Safe, Efficient, and Robust Predictive Control of Constrained Nonlinear Systems

Vishnu R. Desaraju

April 2017

CMU-RI-TR-17-25

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Robotics

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Nathan Michael, Chair
Maxim Likhachev
Koushil Sreenath
Nicholas Roy, MIT

Abstract

As autonomous systems are deployed in increasingly complex and uncertain environments, safe, accurate, and robust feedback control techniques are required to ensure reliable operation. Accurate trajectory tracking is essential to complete a variety of tasks, but this may be difficult if the system's dynamics change online, e.g., due to environmental effects or hardware degradation. As a result, uncertainty mitigation techniques are also necessary to ensure safety and accuracy.

This problem is well suited to a receding-horizon optimal control formulation via Nonlinear Model Predictive Control (NMPC). NMPC employs a nonlinear model of the plant dynamics to compute non-myopic control policies, thereby improving tracking accuracy relative to reactive approaches. This formulation ensures constraints on the dynamics are satisfied and can compensate for uncertainty in the state and dynamics model via robust and adaptive extensions. However, existing NMPC techniques are computationally expensive, and many operating domains preclude reliable, high-rate communication with a base station. This is particularly difficult for small, agile systems, such as micro air vehicles, that have severely limited computation due to size, weight, and power restrictions but require high-rate feedback control to maintain stability. Therefore, the system must be able to operate safely and reliably with typically limited onboard computational resources.

In this thesis, we propose a series of non-myopic, computationally-efficient, feedback control strategies that enable accurate and reliable operation in the presence of unmodeled system dynamics and state uncertainty. The key concept underlying these techniques is the reuse of past experiences to reduce online computation and enhance control performance in novel scenarios. These experiences inform an online-updated estimate of the system dynamics model and the choice of controller to optimize performance for a given scenario. We present a set of simulation and experimental studies with a small aerial robot operating in windy environments to assess the performance of the proposed control methodologies. These results demonstrate that leveraging past experiences to inform feedback control yields high-rate, constrained, robust-adaptive control and enables the deployment of predictive control techniques on systems with severe computational constraints.

Acknowledgments

I initially decided to do a Ph.D. to hone my research skills and to develop a deeper understanding of motion planning and controls for robotics. However, I have gained much more than that over the past few years thanks to the incredible people I have had the privilege to work with.

First and foremost, I would like to thank my advisor, Prof. Nathan Michael, for his guidance, feedback, and insight throughout my time at CMU. It is rare to find a thesis advisor whose research interests align so well with your own, and I feel that has made the Ph.D. a far more enjoyable and fruitful experience. His ability to dissect a problem and clearly communicate the research challenges has been immensely insightful and has helped shape my approach to research. The opportunity to help lay the foundation for the Robust Adaptive Systems Lab has also been an invaluable experience and has taught me far more about the challenges in deploying real-world robotic systems than I could have imagined. I am also grateful to my thesis committee members, Prof. Maxim Likhachev, Prof. Koushil Sreenath, and Prof. Nicholas Roy, for their feedback and insight throughout this process. In hindsight, I realized that my committee also reflects my time at Michigan and MIT, which helped shape my research interests.

I also have to thank all my friends and colleagues who have made the past few years a truly memorable and enjoyable experience, especially John Yao, Ellen Cappo, Humphrey Hu, and Kumar Shaurya Shankar. When conversations can change topics from Lyapunov stability to quadrotor cupcakes to ancient Rome to the curiously recurring template pattern, there is never a dull moment. It has been a pleasure working with Alex Spitzer over the past year, and I am extremely grateful for all his help in conducting and analyzing many of the flight experiments that have helped solidify this thesis. I would also like to thank Arjav Desai and Matt Collins for their help with the Crazyflie platform and Cormac O'Meadhra and Lauren Lieu for their help with the ground robot simulations.

I am also grateful to Chuck Whittaker, Curt Boirum, Derek Mitchell, Wennie Tabib, Xuning Yang, Micah Corah, Vibhav Ganesh, Erik Nelson, Shihyun Lo, and so many others in the Robust Adaptive Systems Lab, the Field Robotics Center, and elsewhere who have helped with everything from infrastructure to research discussions to software development to just relaxing and de-stressing throughout this endeavor. I am also thankful for all the administrative support provided by Karen Widmaier, Ashley McClinton, Lynnetta Miller, Nora Kazour, and Suzanne Lyons Muth. We would certainly descend into chaos without them.

Finally, a sincere thanks to friends around the world, and especially to my parents and my sister for all their support and encouragement over the course of the Ph.D. It is fascinating to see how little has changed in the Ph.D. experience between generations. Thanks to them, I have been able to maintain some semblance of sanity during even the most trying of times.

Contents

List of Figures

List of Tables

List of Algorithms

Notation

Throughout this thesis, we use the following notation to indicate different variable types

Type	Examples
Scalar:	x, r, u, λ, N
Vector:	$\mathbf{x}, \mathbf{r}, \mathbf{u}, \mathbf{c}, \mathbf{g}_{\mathbf{x}}, \boldsymbol{\lambda}$
Concatenated Vector:	$\boldsymbol{x}, \boldsymbol{r}, \boldsymbol{u}, \boldsymbol{c}, \boldsymbol{g}_{\mathbf{x}}, \boldsymbol{\lambda}$
Matrix:	$\mathbf{A}, \mathbf{B}, \mathbf{R}, \mathbf{G}_{\mathbf{x}}, \mathbf{M}$
Concatenated Matrix:	$\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}, \boldsymbol{\mathcal{R}}, \boldsymbol{\mathcal{G}}_{\mathbf{x}}, \boldsymbol{\mathcal{M}}$
Set:	$\mathcal{X}_k, \mathcal{R}_T, \mathcal{M}$
Function:	$J(\cdot), f(\cdot), g(\cdot), \kappa(\cdot)$

Chapter 1

Introduction

Safe, accurate, and reliable motion is fundamental to the deployment of agile, autonomous robotic systems to execute challenging tasks in complex, real-world environments. These systems must employ feedback control techniques that enable them to track trajectories while obeying system limitations (e.g., actuator constraints) and operational bounds (e.g., speed limits for traversing a region of the environment or to satisfy sensor limitations). However, to ensure safety and reliability, they must also be able to mitigate the effects of uncertainty stemming from several sources including errors in the dynamics model, exogenous forces that alter the system's motion, and state estimate uncertainty introduced by inaccuracies in sensing.

These sources of uncertainty are prevalent across domains (Fig. 1.1). For example, a micro air vehicle (MAV) operating in a windy or outdoor scenario is subject to significant perturbations to its dynamics [3, 4, 5]. Similarly, MAVs operating in confined environments or near structures must be able to operate safely and reliably in the presence of aerodynamic forces induced by the vehicle's thrust interacting with nearby surfaces, such as the ground [6, 7] or rooftops [8]. Similarly, underwater vehicles are subject to severe external disturbances, including currents and surge, as well as variable payloads [9, 10]. Surface vehicles must additionally compensate for the effects of wind and waves [11]. Autonomous ground vehicles operating in challenging terrestrial or space environments may be required to traverse sloped surfaces [2] or loose terrain [12] where

(a) (b) (c)

Figure 1.1: Environmental interactions introduce perturbations to motion in a variety of domains including (a) an aerial robot flying in a turbulent wind field, (b) an underwater robot encountering currents and other flow fields [1], and (c) a ground robot traversing a sandy slope [2].

terramechanics heavily influences the vehicle's motion.

Additionally, onboard sensing and perception systems are inherently noisy, and the uncertainty in the resulting state estimates can lead to control actions that compromise the safety and reliability of the system [13, 14]. Moreover, the uncertainty in the state estimates may vary over time as the system traverses different parts of the environment. For example, transitioning from a feature-rich region to one that is feature-sparse or encountering sudden illumination changes may result in drastic changes in state uncertainty [15]. However, even with these changes in state uncertainty, the controller must be able to enforce state and input constraints to ensure safe and successful operation.

This underscores the need for high precision control techniques that are able to mitigate the effects of uncertainty arising from diverse operating conditions. However, autonomous systems are often severely limited in their available onboard computational resources, either in terms of the resources allocated to motion control [16] or in many cases the total onboard compute [17, 18]. Therefore, these techniques must account for the system's nonlinear dynamics, physical and operational limitations, and computational capabilities to ensure safe, accurate, and reliable operation.

1.1 Core Challenges

As a result, we identify three core challenges that must be addressed to enable safe, accurate, and reliable feedback control on uncertain nonlinear systems with performance and computational constraints:

- **Uncertainty mitigation:** Compensate for the effects of state uncertainty and perturbations to the dynamics model.
- **Constraint satisfaction:** Ensure commands and states satisfy hard constraints derived from safety or mission requirements.
- **Computational efficiency:** Achieve sufficiently high control rates for stability of highly dynamic systems with limited compute.

The problem of safe and accurate feedback control for constrained nonlinear systems is well suited to a Nonlinear Model Predictive Control (NMPC) formulation [19]. NMPC is a receding-horizon optimal control technique that enforces constraints on the system state and control inputs while optimizing system evolution according to a nonlinear dynamics model (see Chapter 2 for additional details). The resulting non-myopic control policies yield improved trajectory tracking performance while ensuring safety through constraint satisfaction.

However, to apply NMPC to agile robotic systems operating in real-world settings, we must ensure safe, accurate, and robust trajectory tracking despite motion uncertainty and computational constraints. NMPC performance is dependent on the fidelity of the plant model used to forward-predict and optimize system evolution. Therefore, model uncertainty due to hardware degradation or exogenous perturbations may lead to inaccurate trajectory tracking and potentially catastrophic constraint violations. State estimate uncertainty also incurs similar penalties. Robust and adaptive control formulations seek to mitigate these problems by estimating the effects of uncertainty, e.g., via a Kalman filter, but yield delayed disturbance compensation due to the reactive nature of these estimators. Computational tractability concerns stem from the limited onboard

processing capabilities available on autonomous systems. While some applications may permit offboard computation, many complex operating domains preclude the reliable, low-latency communication required for computationally-intensive offboard control. Agile, small-scale systems, such as micro aerial vehicles, further complicate this issue as they require high-rate control to maintain stability but have severely limited onboard computation due to fundamental size, weight and power restrictions. As a result, many powerful nonlinear control techniques, such as constrained optimal control via nonlinear programming, are not viable for these systems.

1.2 Thesis Contributions

This thesis seeks to address these challenges through the development and validation of a set of nonlinear predictive control methodologies that leverage past experiences to improve tracking performance and computational efficiency. We construct an experience database that encapsulates previously encountered scenarios and the corresponding optimal control laws, thus enabling reuse of past control calculations for new scenarios. Past experiences also inform an online model learner, allowing the system to anticipate and robustly adapt to uncertainty in the state and plant dynamics. The contributions of this thesis are summarized below and detailed in the subsequent chapters:

- **Chapter 3 – Nonlinear Partial Enumeration:** A NMPC solution technique that constructs online a database of local controllers that can be reused in future control iterations. This approach reduces the dependence on online optimization, thus enabling high-rate NMPC on systems with insufficient computational resources to solve nonlinear programs in real-time [20].
- **Chapter 4 – Experience-driven Predictive Control:** An adaptive, predictive controller based on an online-constructed database of past experiences. This approach eliminates the need to solve nonlinear programs, thus enabling high-rate control and rapid acquisition of experience to enhance controller performance. It combines these controllers with an online, experience-

based dynamics model learner, thus enabling adaptation to changes in the system dynamics [21].

- **Chapter 5 – Robust EPC:** An extension of the EPC algorithm to account for state uncertainty and enable robust adaptation. This extension enhances system performance and safety by ensuring robust constraint satisfaction in addition to the real-time capabilities of EPC. It also enables integration with non-idealized state estimation systems and operation in variable environmental conditions.
- **Chapter 6 – Efficient Explicit Adaptive NMPC:** An explicit adaptive NMPC formulation that employs an offline reachable-space search to generate realistic synthetic experiences for EPC. The resulting database avoids the exponential growth of other explicit MPC formulations. A Markov chain representation of the database further enhances the computational efficiency of queries, thereby enabling NMPC without online optimization.

Additionally, each of the developed techniques seeks to demonstrate or enhance a common set of objectives. Throughout the thesis, any result corresponding to a given objective from the list below is highlighted with one of the following labels:

- **R1**: Stable control performance
- **R2**: Constraint satisfaction
  - **R2.1**: Constraint satisfaction in the presence of time-varying dynamics models
  - **R2.2**: Constraint satisfaction in the presence of time-varying sensor uncertainty
  - **R2.3**: Robust constraint satisfaction during aggressive trajectory tracking
- **R3**: Real-time computation of control commands
- **R4**: Improved trajectory tracking performance
  - **R4.1**: Adaptation performance
- **R5**: Reuse of past experiences to improve performance

Chapter 2

Background

This thesis seeks to develop safe, robust, and computationally efficient control methodologies for constrained nonlinear systems with uncertain dynamics and state uncertainty. We therefore leverage and extend techniques from several areas, including explicit and semi-explicit Model Predictive Control (MPC) for safety and computational efficiency, adaptive and robust MPC for uncertainty mitigation, and learning-based control for efficient control law selection and reuse. This chapter provides a brief overview of related work in these areas.

2.1 Feedback Control

Feedback control aims to compute control actions, u, to minimize the error between the state of the plant, x and a desired state or set of states, x^{desired}. As illustrated in Fig. 2.1, we assume the plant is subject to uncertain external disturbances and that x is provided by a (potentially uncertain) state estimation system. Figure 2.2 summarizes the relationships between several control strategies that are commonly employed due to their computational efficiency, ability to mitigate the effects of uncertainty, or capacity to obey state and input constraints.

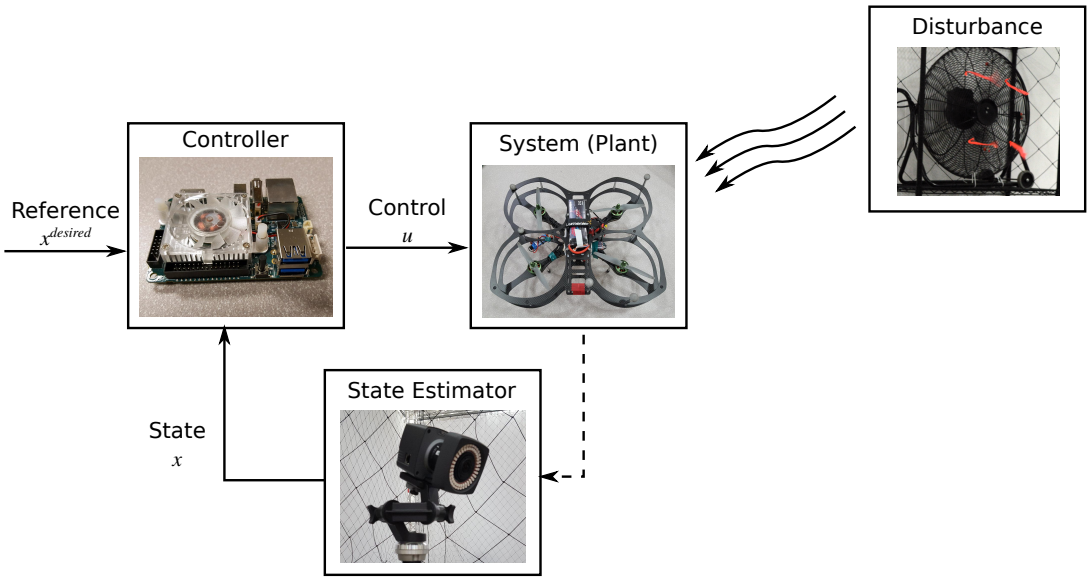


Figure 2.1: General feedback control diagram depicting the control computer that generates commands to drive the plant (e.g., an aerial robot) to track the reference based on state estimates (e.g., from a motion capture system or onboard sensing) and uncertain disturbances (e.g., wind).

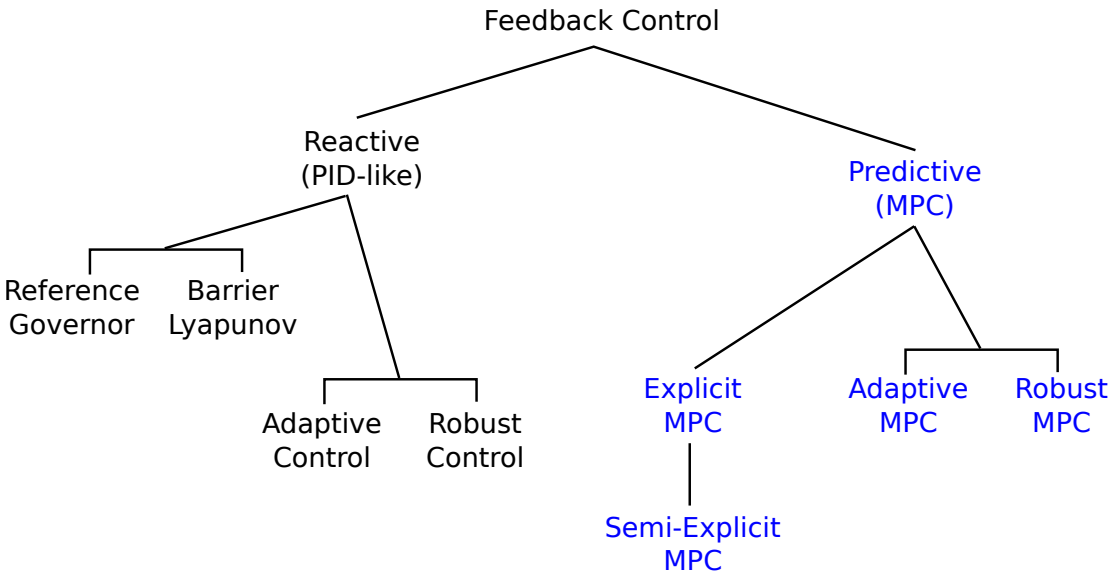


Figure 2.2: Taxonomy of several common classes of control methodologies that address the challenges of computational efficiency, uncertainty mitigation, or constraint satisfaction. The approaches detailed in this thesis represent Semi-Explicit and Explicit MPC solution strategies that leverage Adaptive and Robust MPC formulations to address all three challenges concurrently.

Reactive feedback controllers, such as Proportional-Integral-Derivative (PID) feedback control, achieve high rates through their simplicity. These techniques can also mitigate the effects of uncertainty via adaptive control formulations, such as model-reference adaptive control [22] and $\mathcal{L}_1$ adaptive control [23], that estimate and compensate for the effects of perturbations to the system. Alternatively, robust control formulations, such as H_∞ control [24, 25] and sliding mode control [26], aim to bound the effects of uncertainty to preserve stability and tracking performance. However, these techniques typically do not account for system limitations (e.g., actuator constraints). Additionally, these purely reactive techniques seek to eliminate the effects of unmodeled dynamics, even when they may be beneficial. As a result, they can lead to degraded performance in more challenging settings outside their nominal operating regime. More advanced feedback controllers are able to account for constraints via techniques such as barrier Lyapunov functions [27, 28, 29] or reference governors [30, 31, 32]. In contrast, optimal control techniques can explicitly enforce constraints while computing optimal commands for all future times to improve overall control performance [33]. However, applying these techniques to general nonlinear systems incurs a substantial computational penalty [34] or may even be intractable [35]. Therefore, we look to Model Predictive Control (MPC) to provide a middle ground between simple, reactive controllers and infinite-horizon optimal control formulations [36, 37].

2.2 Model Predictive Control

MPC balances these extremes by casting the control problem as a finite horizon, constrained optimization. It ensures the generated commands obey actuator and operating limits by optimizing the predicted evolution of the system dynamics over an N-step horizon. When applied to a linear system, or a system that does not deviate significantly from a nominal operating point,

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k$$

with linear state and input constraints,

$$\mathbf{G_x x}_{k+1} \leq \mathbf{g_x}$$

$$\mathbf{G_u u}_k \leq \mathbf{g_u}$$

the linear MPC problem can be formulated and solved efficiently as either a constrained linear or quadratic program [38] such as

$$\begin{aligned} \underset{\mathbf{u}_k}{\operatorname{argmin}} \sum_{k=0}^{N-1} \frac{1}{2}(\mathbf{x}_{k+1} - \mathbf{r}_{k+1})^{\mathrm{T}}\mathbf{Q}(\mathbf{x}_{k+1} - \mathbf{r}_{k+1}) + \frac{1}{2}\mathbf{u}_k^{\mathrm{T}}\mathbf{R}\mathbf{u}_k \\ \text{s.t.} \quad \mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k \\ \mathbf{G_x x}_{k+1} \leq \mathbf{g_x} \\ \mathbf{G_u u}_k \leq \mathbf{g_u} \\ \forall\, k = 0, \ldots, N-1 \end{aligned} \tag{2.1}$$

where $\mathbf{x}$ denotes the state vector, $\mathbf{r}$ denotes the reference or desired state, $\mathbf{u}$ denotes the control input vector, and $\mathbf{Q}$ and $\mathbf{R}$ define an LQR-style cost function.

Although these techniques can be applied to nonlinear systems via linearization, the accuracy of the resulting motion is dependent on the fidelity of the prediction model used. This motivates the use of Nonlinear MPC (NMPC), as the nonlinear dynamics model will predict system evolution more accurately than a linear approximation about a nominal operating point. However, due to this nonlinear model and other potentially nonlinear constraints, NMPC is formulated as a nonlinear program (NLP).

More precisely, for a system with nonlinear dynamics given by

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$$

and constraints given by

$$g(\mathbf{x}_{k+1}, \mathbf{u}_k) \leq 0$$

this NLP computes the control sequence $\{\mathbf{u}_1, \ldots, \mathbf{u}_N\}$ given the current state $\mathbf{x}_0$ and references $\mathbf{r}_1, \ldots, \mathbf{r}_N$ (e.g., from a desired trajectory),

$$\begin{aligned} &\underset{\mathbf{u}_k}{\operatorname{argmin}} \sum_{k=0}^{N-1} J(\mathbf{x}_{k+1}, \mathbf{r}_{k+1}, \mathbf{u}_k) \\ \text{s.t.} \quad &\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) \\ &g(\mathbf{x}_{k+1}, \mathbf{u}_k) \leq 0 \; \forall k = 0, \ldots, N-1 \end{aligned} \tag{2.2}$$

where the cost function, $J(\cdot)$, is typically selected to penalize tracking error and extraneous control effort and the differential equation constraint is enforced via numerical integration. In each control iteration, the problem is re-solved from the current state and the first element of the resulting control sequence is applied to the system. While MPC and NMPC formulations exist for different classes of systems [39], in this thesis we restrict our focus nonlinear systems with dynamics that are well modeled by a continuous and smooth (e.g., C^2) function, $f(\cdot)$.

2.3 Fast Nonlinear Model Predictive Control

The NLP formulation of NMPC (2.2) results in a far more computationally expensive controller than the linear variant (2.1). As a result, there are a variety of fast NMPC solution techniques that comprise three main categories: fast online optimization, explicit NMPC, and semi-explicit approaches.

2.3.1 Online NMPC

Online MPC formulations compute solutions at high rates through the use of efficient convex optimization techniques including warm-starting the optimization using a previous solu-

tion [40], exploiting problem structure, and trading speed for optimality [41]. Consequently, many fast NMPC techniques rely on convex approximations, such as sequential quadratic programming [42, 43, 44] or iterative Linear Quadratic Regulator formulations [45], to achieve computational efficiency. However, these techniques assume the optimization problem can be solved quickly and reliably at runtime, which may not be feasible for computationally constrained systems. Additionally, the reliance on online optimization may raise software certifiability concerns for some applications [46].

2.3.2 Explicit MPC and NMPC

Explicit MPC and NMPC approaches eliminate the need for online optimization by constructing a database of locally-optimal controllers derived via a receding horizon optimal control formulation [39, 47, 48]. At runtime, these methods query the database to identify and apply the appropriate controller. However, explicit MPC techniques are known to scale poorly due to the exponential growth in the number of controllers with the number of constraints [40], leading to a prohibitively expensive database search. Consequently, numerous approximate explicit MPC strategies are proposed that seek to improve the efficiency of database queries, for example, through the introduction of search trees connecting partitions of a reduced state-space [49, 50]. Alternatively, some approaches leverage function approximation techniques to compute a continuous mapping from state to control output that replaces the controller database [51] and can be combined with a partitioning strategy to yield a hierarchical model [52].

Explicit MPC is part of a broader class of methodologies that generate a set of control policies during an offline training phase to enable high-rate online control. For example, the LQR-Tree algorithm leverages offline sums-of-squares optimization to construct sequences of controllers that are applied online to stabilize the system along specific trajectories [53, 54]. Other techniques, such as MPC-guided policy search [55] and Learning Global Optima [56], employ demonstrations of optimal behavior to train a learner (e.g., k-NN or neural network model) that seeks to

approximate the optimal control policy at a given query point. However, these techniques are limited by the type and number of optimal training examples that can be generated and therefore often focus on learning specific behaviors [55] to avoid the same exponential growth in the number of policies as in explicit MPC [56].

2.3.3 Semi-Explicit MPC

Semi-explicit MPC techniques combine a controller database with online optimization to balance the strengths and weaknesses of each. The Partial Enumeration (PE) [57] technique incrementally constructs a controller database by solving the MPC optimization problem online if a locally-optimal controller is not found in the current database. The solution is introduced into the database for future use, thereby reducing the occurrence of online optimization computations. Due to its incremental and need-based optimization formulations, PE yields smaller databases than explicit MPC, enabling rapid online queries. However, the approach is specifically formulated for linear systems. Another strategy is to restrict the size of the database by selecting a subset of the most commonly used controllers and interpolating between them at runtime [39].

The construction of a database from past actions in order to facilitate choosing future actions is also the foundation of transfer learning and lifelong learning algorithms. These learning-based approaches consider executing tasks, which, by analogy to the PE algorithm, can be viewed as a particular state-reference sequence. Transfer learning seeks to use knowledge about past tasks to bootstrap learning a new task [58], similar to efficient MPC strategies [40]. Lifelong learning shares similarities with the PE algorithm in that it makes this knowledge transfer bidirectional to learn policies that maximize performance over all past and present tasks [59]. However, the PE algorithm maintains a finite set of controllers that are updated through infrequent computation and do not permit interpolation. Whereas lifelong learning algorithms, such as ELLA [59] or OMTL [60], maintain a set of bases that aid in reconstructing task models whenever new data is received.

While semi-explicit techniques learn from real experience gained through operation, many learning-based techniques employ synthetic experiences in a similar way to generate or refine their policies. Action model-based approaches use prior experiences to construct a model that mimics the behavior of the world. The models aim to find optimal future actions via dynamic programming [61] or stochastic shortest path [62], which can then be used to provide synthetic experiences. Depending on the underlying learner, these experiences can be used to perform additional refinement (e.g., additional value iteration steps in a Q-learning framework [63]) or to guide the learner toward better solutions [62]. Other approaches aim to improve learning speed and performance by introducing "imagined" experiences to the training data provided to the learner [64]. For example, a recurrent neural network trained on actual experiences can be used to predict comparable but imaginary scenarios to serve as more general training data for the original learner [65].

2.4 MPC with Plant and State Uncertainty

For systems operating in real-world environments, a fixed nonlinear dynamics model may be insufficient to accurately predict motion due to modeling errors and unmodeled, time varying, exogenous disturbances. This motivates the use of MPC formulations that directly account for these sources of uncertainty, either by modifying the dynamics or tightening the constraints to ensure accurate motion prediction and constraint satisfaction. Consequently, there are two general approaches to mitigating the effects of uncertainty in predictive control: adaptive MPC formulations and robust MPC formulations.

2.4.1 Adaptive MPC

The issue of model accuracy for predictive control can be addressed through various adaptation and learning-based approaches. Adaptive formulations seek to estimate the uncertainty in the

dynamics and update the predictive model to more accurately anticipate the system's interaction with the constraints. Many existing adaptive MPC approaches assume a structured system model with uncertain parameters that can be estimated online. These approaches then combine a standard MPC formulation with an online parameter estimator, e.g., a Luenberger observer or Kalman filter, to achieve more accurate, deliberative actions [14, 66, 67].

However, treating all model uncertainty as parameters to estimate can limit overall model accuracy, especially when the system is subject to complex, exogenous perturbations, such as aerodynamic effects on an aerial vehicle. Therefore, learning-based function approximation techniques are also applied to address this issue. The resulting semi-structured approaches augment a structured system model with a non-parametric, online-learned component, e.g., via a Gaussian process [12]. The resulting semi-parametric model is then queried within the NMPC formulation while continuing to adapt to model changes. While techniques such as Gaussian process regression scale poorly with the amount of training data, other kernel-based approaches, such as Locally Weighted Projection Regression (LWPR) and Incremental Sparse Spectrum Gaussian Process Regression (ISSGPR) incorporate data more efficiently via linear basis functions [68] or trigonometric approximations [69], respectively. The resulting incremental updates enable fast model learning that is suitable for finite-horizon control [70].

2.4.2 Robust MPC

However, in practice, adaptive formulations may still lead to constraint violations due to the difference in timescales between the disturbance estimator and high-frequency noise in the state estimate. Therefore, Robust MPC techniques are also employed frequently to mitigate the effects of state and model uncertainty. In contrast to adaptive approaches, robust formulations refine the constraints to explicitly include uncertainty. Robust MPC techniques provide constraint satisfaction guarantees in the presence of bounded, uncertain parameters [38, 71, 72, 73]. For linear dynamics, the effects of bounded uncertainty can be represented by disturbance-invariant sets [74]

that can be subtracted from the set of feasible states via the Pontryagin difference operation. In the nonlinear case, this can be generalized to min-max formulations to optimize with respect to the maximal state deviations [75]. These techniques yield more conservative controllers than the adaptive approaches, but as a result, are able to account for any variations within the bounded uncertainty set without requiring a disturbance estimator that can track rapid changes.

A subset of these Robust MPC techniques employ local feedback control laws to restrict the anticipated growth of uncertainty. This yields constraint tightening and Tube MPC approaches that enable more aggressive performance [72, 73, 76]. While many formulations assume the uncertainty set is known a priori (e.g., as a disturbance invariant set or via the min-max calculation), some approaches permit online modification of robustness bounds driven by online estimates of the uncertainty bounds [13]. An extension of this idea replaces the deterministic uncertainty set with a probabilistic representation, e.g., as a multivariate Gaussian distribution [77]. This enables the use of a Kalman filter to predict the evolution of state uncertainty instead of the recursive Pontryagin difference operations required for deterministic sets [78]. Many adaptive MPC formulations also include a robust component that is coupled to estimator uncertainty [66, 75, 79]. The resulting robust-adaptive formulations allow the adaptive component to estimate and compensate for low frequency components of the uncertainty, while variability about the current estimate is mitigated by the robust constraints.

2.5 Online Dynamics Model Learning

As mentioned in Sect. 2.4.1, adaptive MPC formulations mitigate the effects of uncertainty in the dynamics model. These techniques estimate corrections to the nominal dynamics model based on discrepancies between the system's predicted and observed state evolution [14]. Although this thesis does not aim to develop a novel online model learning strategy, the algorithms proposed in many of the following chapters do incorporate an adaptive component. Therefore, we briefly review three approaches used throughout this thesis for online estimation of perturbations to the

dynamics model.

We first consider a purely reactive adaptation strategy based on $\mathcal{L}_1$ adaptive control [23]. This approach employs a nonlinear Luenberger observer driven by the difference between the state predicted via the nominal nonlinear dynamics model and the state reported by the state estimator. Additionally, this approach is representative of the observer or Kalman filter based techniques that are prevalent in the literature [14, 66, 67].

The second approach applies Locally Weighted Projection Regression (LWPR), which we consider an experience-based technique rather than a purely reactive one. LWPR learns corrections to a nominal dynamics model via a Gaussian-weighted combination of local linear basis functions that are updated incrementally via partial least squares [68]. These basis functions encapsulate all past dynamics information, in contrast to storing all past training data as in a Gaussian process, and new bases are introduced as required when the existing set are insufficient to represent new data with the desired accuracy. LWPR also has a forgetting factor to control the rate of adaptation to model changes by adjusting the effects of prediction error on the weight for each basis. Additionally, the computationally efficient update policy makes LWPR well-suited to real-time operation [70]. Therefore, given a state-control pair, LWPR returns the anticipated error between the predicted and actual next state.

Finally, we also consider Incremental Sparse Spectrum Gaussian Process Regression (ISSGPR) [69]. ISSGPR is a regression algorithm that projects input data onto a set of trigonometric basis functions with randomly chosen frequencies. Regularized linear regression is then performed in this feature space to obtain the predictive mean. In addition, a Bayesian perspective allows the algorithm to be cast as a Gaussian Process Regression (GPR), which provides the predictive variance. Although standard GPR has cubic run time in the number of data points, ISSGPR achieves constant time by using an explicit feature space, thereby avoiding the expensive computation of the Gramian matrix. Similarly to LWPR, ISSGPR regresses to a scalar output. Thus we fit the dynamics model element-wise.

Chapter 3

Nonlinear Partial Enumeration

This chapter presents Nonlinear Partial Enumeration (NPE), a Nonlinear Model Predictive Control (NMPC) technique that combines online and offline computation to yield a nonlinear version of Partial Enumeration MPC, thereby dramatically decreasing the solution time per NMPC iteration and making it viable for use on systems with dynamics that evolve on the order of milliseconds. The proposed approach leverages a parallelized structure, ensuring that a feasible solution is returned at the required rate while enabling slower optimization techniques to learn local control laws that capture the functionality of standard NMPC. Using the problem of aggressive MAV flight as a guiding example, we leverage this formulation to demonstrate through a set of simulation and experimental trials the functionality and performance of NPE, as well as its ability to enable online learning of reusable local feedback control laws.

3.1 Approach

We propose a novel nonlinear extension of the Partial Enumeration (PE) technique [80] to construct online a piecewise-affine control law as the solution to a NMPC problem. Just as linear PE leverages explicit MPC techniques based on multi-parametric quadratic programming (mp-QP), we employ ideas from explicit NMPC to combine solutions to a nonlinear program (NLP) with

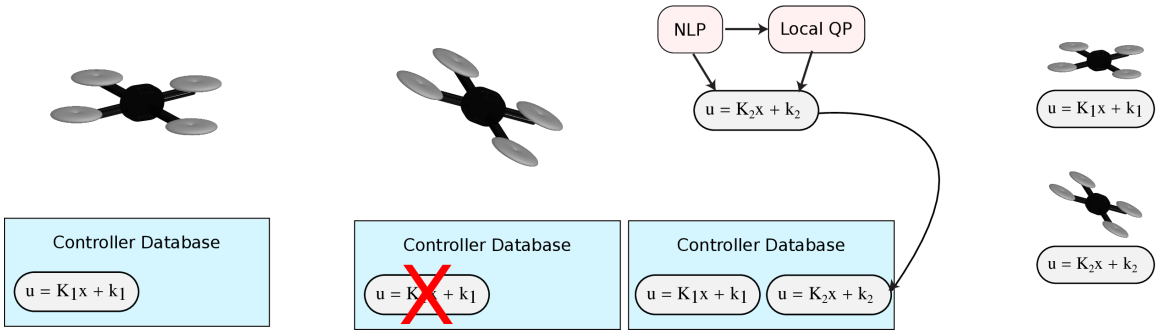


Figure 3.1: Overview of the proposed approach that constructs a reusable controller database to recover the functionality of NMPC. (a) A MAV initially operates away from constraint boundaries enabling it to apply a controller in the database. (b) As the MAV transitions to more aggressive flight and approaches a constraint boundary, a new controller is added to the database that enforces this constraint. (c) The MAV reuses controllers in the database according to its state to satisfy constraints.

local mp-QPs [48], thereby reducing the number of NLPs that must be solved online. The resulting NPE algorithm is summarized in Fig. 3.1, as applied to a quadrotor micro air vehicle (MAV), and detailed below.

3.1.1 Receding-Horizon Control Formulation

We first formulate a finite horizon NLP to compute the control sequence $\{\mathbf{u}_1, \ldots, \mathbf{u}_N\}$ given the current state $\mathbf{x}_0$ and references $\mathbf{r}_1, \ldots, \mathbf{r}_N$ (e.g., from a desired trajectory),

$$\begin{aligned} \underset{\mathbf{u}_k}{\operatorname{argmin}} \sum_{k=0}^{N-1} & J_k(\mathbf{x}_{k+1}, \mathbf{r}_{k+1}, \mathbf{u}_k) \\ \text{s.t.} \quad & \dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) \\ & g(\mathbf{x}_{k+1}, \mathbf{u}_k) \leq 0 \\ & \forall\, k = 0, \ldots, N-1 \end{aligned} \tag{3.1}$$

where the differential equation constraint is enforced via numerical integration. The resulting optimal control sequence must satisfy the first-order KKT conditions

$$\nabla_{\mathbf{u}} L(\mathbf{x}, \mathbf{r}, \mathbf{u}, \boldsymbol{\lambda}) = \mathbf{0} \tag{3.2}$$

$$\boldsymbol{\Lambda} g(\mathbf{x}, \mathbf{u}) = \mathbf{0} \tag{3.3}$$

$$\boldsymbol{\lambda} \geq \mathbf{0} \tag{3.4}$$

$$g_k(\mathbf{x}, \mathbf{u}) \leq \mathbf{0} \tag{3.5}$$

where $\boldsymbol{\lambda}$ is a vector of Lagrange multipliers, $L(\mathbf{x}, \mathbf{r}, \mathbf{u}, \boldsymbol{\lambda}) = J(\mathbf{x}, \mathbf{r}, \mathbf{u}) + \boldsymbol{\lambda}^{\mathrm{T}} g(\mathbf{x}, \mathbf{u})$ is the corresponding Lagrangian, and $\boldsymbol{\Lambda} = \mathrm{diag}(\boldsymbol{\lambda})$. In a standard online NMPC framework, the first element of this sequence would be applied and the problem re-solved from the updated state.

Instead, given a sequence of control inputs $\{\mathbf{u}_k^*\}$ that are the solution to (3.1) at $\mathbf{x}^*$, we define difference variables $\bar{\mathbf{x}} = \mathbf{x} - \mathbf{x}^*$, $\bar{\mathbf{r}} = \mathbf{r} - \mathbf{x}^*$, and $\bar{\mathbf{u}} = \mathbf{u} - \mathbf{u}^*$ and formulate a local QP as

$$\begin{aligned} \underset{\mathbf{u}_k}{\operatorname{argmin}} \sum_{k=0}^{N-1} \frac{1}{2} (\bar{\mathbf{x}}_{k+1} - \bar{\mathbf{r}}_{k+1})^{\mathrm{T}} \mathbf{Q}_{k+1} (\bar{\mathbf{x}}_{k+1} - \bar{\mathbf{r}}_{k+1}) + \frac{1}{2} \bar{\mathbf{u}}_k^{\mathrm{T}} \mathbf{R}_k \bar{\mathbf{u}}_k \\ \text{s.t.} \quad \bar{\mathbf{x}}_{k+1} = \mathbf{A} \bar{\mathbf{x}}_k + \mathbf{B} \bar{\mathbf{u}}_k \\ \mathbf{G}_{\mathbf{x}_{k+1}} \bar{\mathbf{x}}_{k+1} \leq \mathbf{g}_{\mathbf{x}_{k+1}} \\ \mathbf{G}_{\mathbf{u}_k} \bar{\mathbf{u}}_k \leq \mathbf{g}_{\mathbf{u}_k} \\ \forall\, k = 0, \ldots, N-1 \end{aligned} \tag{3.6}$$

where $\mathbf{Q}_{k+1}$ and $\mathbf{R}_k$ are given by the Hessian of $J_k(\mathbf{x}, \mathbf{r}, \mathbf{u})$ and $\mathbf{A}, \mathbf{B}, \mathbf{G}_{\mathbf{x}_{k+1}}, \mathbf{G}_{\mathbf{u}_k}, \mathbf{g}_{\mathbf{x}_{k+1}}, \mathbf{g}_{\mathbf{u}_k}$ are given by the linearization of $f(\mathbf{x}, \mathbf{u})$ and $g_k(\mathbf{x}, \mathbf{u})$ about $\{\mathbf{u}_k^*\}$ and $\mathbf{x}^*$. To simplify the formulation, we note that the linearized dynamics over N steps can be rewritten as $\boldsymbol{x} = \boldsymbol{\mathcal{A}} \bar{\mathbf{x}}_0 + \boldsymbol{\mathcal{B}} \boldsymbol{u}$,

where

$$\boldsymbol{x} = \begin{bmatrix} \bar{\mathbf{x}}_1 \\ \bar{\mathbf{x}}_2 \\ \vdots \\ \bar{\mathbf{x}}_N \end{bmatrix} \qquad \boldsymbol{u} = \begin{bmatrix} \bar{\mathbf{u}}_0 \\ \bar{\mathbf{u}}_1 \\ \vdots \\ \bar{\mathbf{u}}_{N-1} \end{bmatrix} \qquad \boldsymbol{\mathcal{A}} = \begin{bmatrix} \mathbf{A} \\ \mathbf{A}^2 \\ \vdots \\ \mathbf{A}^N \end{bmatrix} \qquad \boldsymbol{\mathcal{B}} = \begin{bmatrix} \mathbf{B} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{AB} & \mathbf{B} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \\ \mathbf{A}^{N-1}\mathbf{B} & \mathbf{A}^{N-2}\mathbf{B} & \dots & \mathbf{B} \end{bmatrix}$$

Additionally, let $\boldsymbol{\mathcal{H}} = \boldsymbol{\mathcal{B}}^{\mathrm{T}} \boldsymbol{\mathcal{Q}} \boldsymbol{\mathcal{B}} + \boldsymbol{\mathcal{R}}$, where $\boldsymbol{\mathcal{Q}} = \mathrm{diag}(\mathbf{Q}_1, \dots, \mathbf{Q}_N)$ and $\boldsymbol{\mathcal{R}} = \mathrm{diag}(\mathbf{R}_0, \dots, \mathbf{R}_{N-1})$, and $\boldsymbol{h} = \boldsymbol{\mathcal{B}}^{\mathrm{T}} \boldsymbol{\mathcal{Q}} (\boldsymbol{\mathcal{A}} \bar{\mathbf{x}}_0 - \boldsymbol{r})$, where $\boldsymbol{r}$ is defined analogously to $\boldsymbol{x}$. Similarly, let $\boldsymbol{\mathcal{G}}_{\mathbf{x}} = \mathrm{diag}(\mathbf{G}_{\mathbf{x}_1}, \dots, \mathbf{G}_{\mathbf{x}_N})$, $\boldsymbol{\mathcal{G}}_{\mathbf{u}} = \mathrm{diag}(\mathbf{G}_{\mathbf{u}_0}, \dots, \mathbf{G}_{\mathbf{u}_{N-1}})$, $\boldsymbol{g}_{\mathbf{x}} = \begin{bmatrix} \mathbf{g}_{\mathbf{x}_1}^{\mathrm{T}}, \dots, \mathbf{g}_{\mathbf{x}_N}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$, $\boldsymbol{g}_{\mathbf{u}} = \begin{bmatrix} \mathbf{g}_{\mathbf{u}_0}^{\mathrm{T}}, \dots, \mathbf{g}_{\mathbf{u}_{N-1}}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$, and

$$\boldsymbol{\Gamma} = \begin{bmatrix} \boldsymbol{\mathcal{G}}_{\mathbf{x}} \boldsymbol{\mathcal{B}} \\ \boldsymbol{\mathcal{G}}_{\mathbf{u}} \end{bmatrix} \qquad \boldsymbol{\gamma} = \begin{bmatrix} \boldsymbol{g}_{\mathbf{x}} - \boldsymbol{\mathcal{G}}_{\mathbf{x}} \boldsymbol{\mathcal{A}} \bar{\mathbf{x}}_0 \\ \boldsymbol{g}_{\mathbf{u}} \end{bmatrix}$$

We can then rewrite (3.6) in an equivalent form (by dropping constant terms in the cost function) as

$$\begin{aligned} \underset{\boldsymbol{u}}{\mathrm{argmin}} \ & \frac{1}{2} \boldsymbol{u}^{\mathrm{T}} \boldsymbol{\mathcal{H}} \boldsymbol{u} + \boldsymbol{h}^{\mathrm{T}} \boldsymbol{u} \\ \text{s.t.} \quad & \boldsymbol{\Gamma} \boldsymbol{u} \leq \boldsymbol{\gamma} \end{aligned} \tag{3.7}$$

This form facilitates writing the KKT conditions (3.2) and (3.3) for the local QP as

$$\begin{aligned} \boldsymbol{\mathcal{H}} \boldsymbol{u} + \boldsymbol{h} + \boldsymbol{\Gamma}^{\mathrm{T}} \boldsymbol{\lambda} &= \mathbf{0} \\ \boldsymbol{\Lambda} (\boldsymbol{\Gamma} \boldsymbol{u} - \boldsymbol{\gamma}) &= \mathbf{0} \end{aligned} \tag{3.8}$$

If we only consider the active constraints (i.e., with $\lambda > 0$) for a given solution, we can reconstruct $\boldsymbol{u}$ and $\boldsymbol{\lambda}$ by solving a linear system derived from (3.8), where the subscript a indicates

rows corresponding to the active constraints

$$\begin{bmatrix} \boldsymbol{\mathcal{H}} & \boldsymbol{\Gamma}_a^{\mathrm{T}} \\ \boldsymbol{\Gamma}_a & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{\lambda}_a \end{bmatrix} = \begin{bmatrix} -\boldsymbol{h} \\ \boldsymbol{\gamma}_a \end{bmatrix} \tag{3.9}$$

Assuming the active constraints are linearly independent (Bemporad, et al. [47] suggest alternatives if this assumption fails), the resulting local QP control law $\boldsymbol{u}^*$ is affine in $\bar{\mathbf{x}}_0$ and $\boldsymbol{r}$. If we denote the NLP solution as $\boldsymbol{u}_{\mathrm{NL}}$, the overall control law $\kappa(\mathbf{x}_0, \boldsymbol{r})$ then consists of an affine feedback term computed via the local QP and a constant feedforward term determined by the NLP. We can then combine these terms into a single feedback control law, $\kappa(\bar{\mathbf{x}}_0, \boldsymbol{r})$, defined by the gain matrices $\mathbf{K}_1$ and $\mathbf{K}_2$ and a feedforward vector $\mathbf{k}_{\mathrm{ff}}$,

$$\begin{aligned} \kappa(\bar{\mathbf{x}}_0, \boldsymbol{r}) &= \boldsymbol{u}^*(\bar{\mathbf{x}}_0, \boldsymbol{r}) + \boldsymbol{u}_{\mathrm{NL}} \\ &= \mathbf{K}_1 \bar{\mathbf{x}}_0 + \mathbf{K}_2 \boldsymbol{r} + \mathbf{k}_{\mathrm{ff}} \end{aligned} \tag{3.10}$$

However, as the feedback term is derived from a local approximation, we must determine the region of validity of this solution. A similar region is computed in mp-QP explicit NMPC [48] by formulating additional optimization problems to find hard bounds on the suboptimality of $\boldsymbol{u}^*$ relative to the NLP solution. However, this is intractable for online computation. We instead follow PE and determine the region of validity by first checking the remaining KKT conditions (3.4) and (3.5) for the local QP. We can then further restrict the region of validity via commonly used measures of suboptimality, such as the KKT tolerance criteria used to determine when to terminate iterations in sequential quadratic programming [44],

$$\begin{aligned} |\nabla_u L(\mathbf{x}, \mathbf{r}, \mathbf{u}, \boldsymbol{\lambda})| &\leq \epsilon \\ -\boldsymbol{\delta} \leq \boldsymbol{\Lambda} g(\mathbf{x}, \mathbf{u}) &\leq \boldsymbol{\delta} \end{aligned} \tag{3.11}$$

where ϵ and $\boldsymbol{\delta}$ are predefined tolerance parameters.

Algorithm 3.1 Nonlinear Partial Enumeration

1: $\mathcal{M} \leftarrow \emptyset$ or $\mathcal{M}_{\text{prior}}$
2: `solution_found` $\leftarrow$ false
3: `nco_running` $\leftarrow$ false
4: **while** control is enabled **do**
5: $\quad \mathbf{x}_0 \leftarrow$ current system state
6: $\quad \boldsymbol{r} \leftarrow$ current reference sequence
7: $\quad$ **for** each element $m_i \in \mathcal{M}$ **do**
8: $\quad\quad$ Compute $\boldsymbol{u}, \boldsymbol{\lambda}$ via (3.9)
9: $\quad\quad$ **if** $\mathbf{x}_0, \boldsymbol{r}$ satisfy KKT criteria (3.4), (3.5) applied to (3.7) and (3.11) **then**
10: $\quad\quad\quad$ `importance`$_i \leftarrow$ current time, sort $\mathcal{M}$
11: $\quad\quad\quad$ `solution_found` $\leftarrow$ true
12: $\quad\quad\quad$ Apply affine control law (3.10) from m_i
13: $\quad\quad\quad$ **break**
14: $\quad\quad$ **end if**
15: $\quad$ **end for**
16: $\quad$ **if** `solution_found` is false **then**
17: $\quad\quad$ **if** `nco_running` is false **then**
18: $\quad\quad\quad$ Start Alg. 3.2 (parallel thread)
19: $\quad\quad$ **end if**
20: $\quad\quad$ Apply intermediate control via linear MPC (3.13)
21: $\quad$ **end if**
22: **end while**

3.1.2 NPE Algorithm

Introducing these nonlinear KKT criteria enables us to extend the state of the art for fast NMPC by defining a Nonlinear Partial Enumeration (NPE) strategy, as described in Algorithm 3.1. As in linear PE, we aim to construct a mapping $\mathcal{M}$ from regions of the state space to local, affine controllers. Each element $m \in \mathcal{M}$ is defined by a nominal state, an affine controller, and an `importance` score that is used to order the elements. Intuitively, the system is not expected to transition between regions frequently, so we choose to order the elements by when they were last used (Pannocchia et al. [80] discuss other strategies). $\mathcal{M}$ can either be initialized as an empty set or with information from previous trials to reduce the need for online optimization. In each control iteration, we first evaluate the KKT criteria at the current state and reference for each element in $\mathcal{M}$ (lines 7-9). If any element satisfies the criteria, we update its `importance` value

Algorithm 3.2 NPE: New Controller Optimization

1: `nco_running` $\leftarrow$ true
2: $\boldsymbol{u}_{\text{NL}} \leftarrow$ Solution to NLP (3.1) at $\mathbf{x}_0$
3: $(\mathbf{K}_1, \mathbf{K}_2, \mathbf{k}_{\text{ff}}) \leftarrow$ Local QP (3.7) solution about $\mathbf{x}_0, \boldsymbol{u}_{\text{NL}}$
4: **if** NLP and local QP solutions are found **then**
5: **if** $|\mathcal{M}| >$ max table size **then**
6: Remove element from $\mathcal{M}$ with minimum `importance`
7: **end if**
8: Add new element $m_{\text{new}} = (\mathbf{x}_0, \mathbf{K}_1, \mathbf{K}_2, \mathbf{k}_{\text{ff}}, \texttt{importance})$ to $\mathcal{M}$
9: **end if**
10: `nco_running` $\leftarrow$ false

to the current time (line 10) and apply the corresponding affine controller (line 12). In this case, no online optimization is required to generate a locally optimal feedback control law.

If no element in $\mathcal{M}$ satisfies the criteria, we use a parallelized approach to compute and add a new element to $\mathcal{M}$ without blocking the main control loop. As described in Alg. 3.2, we solve the NLP and local QP (lines 2-3) and in line 8 add the corresponding element to $\mathcal{M}$ (as defined in (3.10)). To control the amount of time spent querying the mapping, we can restrict its size and, if necessary, remove the lowest-`importance` element prior to adding m_{new}, as shown in lines 5-6.

While the new element of $\mathcal{M}$ is being computed, we use an intermediate controller to quickly compute suboptimal commands that ensure stability and constraint satisfaction (Alg. 3.1, line 20). As Pannocchia et al. note, there are several options for the intermediate controller [80]. For example, we can formulate the intermediate controller as a linear MPC with a shorter horizon $\tilde{N}$

and soft constraints:

$$\begin{aligned}
\underset{\mathbf{u}_k, \boldsymbol{\varepsilon}_k}{\operatorname{argmin}} \sum_{k=0}^{\tilde{N}-1} \frac{1}{2}(\bar{\mathbf{x}}_{k+1} - \bar{\mathbf{r}}_{k+1})^{\mathrm{T}} \mathbf{Q_{k+1}} (\bar{\mathbf{x}}_{k+1} - \bar{\mathbf{r}}_{k+1}) + \frac{1}{2} \bar{\mathbf{u}}_k^{\mathrm{T}} \mathbf{R_k} \bar{\mathbf{u}}_k + \frac{1}{2} \boldsymbol{\varepsilon}_k^{\mathrm{T}} \mathbf{S} \boldsymbol{\varepsilon}_k \\
\text{s.t.} \quad \bar{\mathbf{x}}_{k+1} = \mathbf{A}\bar{\mathbf{x}}_k + \mathbf{B}\bar{\mathbf{u}}_k \\
\mathbf{G}_{\mathbf{x}_{k+1}} \bar{\mathbf{x}}_{k+1} - \boldsymbol{\varepsilon}_k \leq \mathbf{g}_{\mathbf{x}_{k+1}} \\
\mathbf{G}_{\mathbf{u}_k} \bar{\mathbf{u}}_k \leq \mathbf{g}_{\mathbf{u}_k} \\
\forall\, k = 0, \ldots, N-1
\end{aligned} \tag{3.12}$$

The bounds on the control inputs are enforced as hard constraints to ensure the resulting commands are feasible, while slack variables $\boldsymbol{\varepsilon}_k$ are added to the state constraints to allow violations with some cost penalty, $\mathbf{S}$. The slack variables are unconstrained to ensure existence of a solution.

As in the local QP, this can be re-written such that $\mathbf{u}_k$ and $\boldsymbol{\varepsilon}_k$ are the only decision variables,

$$\begin{aligned}
\underset{\boldsymbol{u}, \boldsymbol{\varepsilon}}{\operatorname{argmin}} \frac{1}{2} \boldsymbol{u}^{\mathrm{T}} \boldsymbol{\mathcal{H}} \boldsymbol{u} + \boldsymbol{h}^{\mathrm{T}} \boldsymbol{u} + \frac{1}{2} \boldsymbol{\varepsilon}^{\mathrm{T}} \boldsymbol{\mathcal{S}} \boldsymbol{\varepsilon} \\
\text{s.t.} \quad \boldsymbol{\Gamma} \boldsymbol{u} - \boldsymbol{\varepsilon} \leq \boldsymbol{\gamma}
\end{aligned} \tag{3.13}$$

where $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{\varepsilon}$ aggregate $\mathbf{S}$ and $\boldsymbol{\varepsilon}_k$, respectively.

As this process iterates, $\mathcal{M}$ will be populated by the most useful elements, reducing the dependence on the intermediate controller. The combination of controllers queried from $\mathcal{M}$ and the intermediate controller ensures the existence of a locally optimal feedback controller at every iteration. Since the computationally expensive components of the algorithm are run in parallel, NPE will compute high-rate, stabilizing commands at all times, thereby enabling fast, nearly optimization-free but minimally-suboptimal control that improves over time.

3.2 Results

To assess the performance of the proposed NPE algorithm, we aim to demonstrate the following results (from the set enumerated in Chapter 1): **R1**: stable control performance, **R4**: improved trajectory tracking performance, **R5**: online controller learning and reuse, **R2**: constraint satisfaction, and **R3**: realtime computation to enable high-rate control. We consider the specific case of a quadrotor micro aerial vehicle (MAV) tracking aggressive trajectories. To predict system evolution, we employ a nonlinear dynamics model for the quadrotor consisting of 12 states and four control inputs, as detailed in Appendix A.3.1. To formulate the NMPC problem, we select a standard linear-quadratic cost function

$$J(\mathbf{x}, \mathbf{r}, \mathbf{u}) = \frac{1}{2}(\mathbf{x} - \mathbf{r})^{\mathrm{T}}\mathbf{Q}(\mathbf{x} - \mathbf{r}) + \frac{1}{2}\mathbf{u}^{\mathrm{T}}\mathbf{R}\mathbf{u}$$

and define $g(\mathbf{x}, \mathbf{u})$ to enforce the following constraints on the velocity, orientation, and control commands,

$$\mathbf{v}_{\text{min}} \leq \mathbf{v} \leq \mathbf{v}_{\text{max}}$$

$$\boldsymbol{\xi}_{\text{min}} \leq \boldsymbol{\xi} \leq \boldsymbol{\xi}_{\text{max}}$$

$$\mathbf{u}_{\text{min}} \leq \mathbf{u} \leq \mathbf{u}_{\text{max}}$$

By using (A.1) as the dynamics constraint in (3.1), the controller will directly compute force and moment commands from $\mathbf{r}$, without the need for intermediate commands often seen in quadrotor control [81].

3.2.1 Simulation Studies

We conduct a series of quadrotor flights using the high-fidelity simulation environment described in Sect. A.1 running on a 2.9 GHz Intel mobile processor. In practice, quadrotor attitude controllers are often run at high rates (e.g., greater than 200 Hz) for reliable stabilization. As our

formulation directly computes the forces and moments (as an attitude controller would, see Sect. A.3.1), we require the controller to return solutions at 200 Hz. Given the relative degree of the quadrotor dynamics, we choose a ten-step prediction horizon with a step size of 20ms, thereby allowing the controls to have a non-trivial effect on position states over the course of the predicted motion.

We first consider a scenario (shown in Fig. 3.2) in which the quadrotor must track a linear trajectory that requires increasing speeds every lap (ranging from 0.6 m/s to 3.0 m/s). We compare the performance of NPE against a proportional-derivative (PD) controller and linear MPC. The linear MPC follows the formulation in (3.13) and uses the same parameters as in NPE. It uses a model of the system dynamics that is linearized about the nominal hover state and commands. The PD control gains are also selected to be comparable to an unconstrained version of the linear MPC (i.e., finite horizon LQR). These controllers are chosen as they can achieve the 200 Hz update requirement. A standard NMPC implementation is three orders of magnitude slower due to the NLP solver (see Table 3.1), and therefore is not viable for comparison in this scenario.

As Fig. 3.3 illustrates, PD control is able to track the trajectory well at lower speeds but degrades at higher speeds due to overshoot and the resulting large oscillations. Linear MPC does not exhibit this overshoot due to its predictive model. However, it does suffer from sustained tracking error and large roll angles due to the choice of linearization point. Linearizing about a non-zero roll command can actually eliminate this error in one direction, but consequently increases the error during the return lap. Since NPE uses the NLP to provide a feed-forward term when computing a new controller, the linearization point for the local QP and resulting feedback controllers is chosen intelligently, resulting in stable flight (**R1**) with reduced tracking error and less severe roll angles (**R4**).

To better illustrate NPE's ability to learn, use, and reuse controllers, we consider another scenario in which the quadrotor is repeatedly commanded to take-off and land aggressively, i.e., by commanding a 1.5 m step change in the desired altitude. NPE is initialized with an empty set of controllers, and Fig. 3.4 shows the evolution of the control strategy over repeated takeoff and

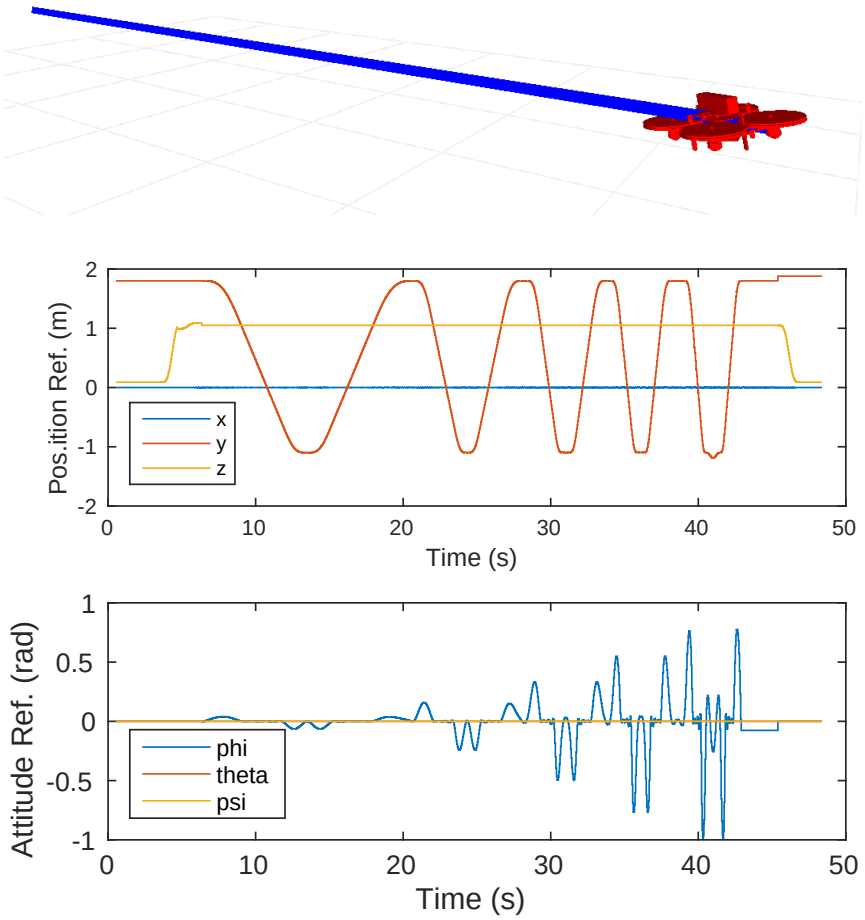


Figure 3.2: Reference trajectory for the first test scenario

landing sequences. The first column illustrates the dependence on the intermediate controller, while the remaining columns correspond to the online computed controllers as they are added to the mapping $\mathcal{M}$. The first local feedback controller is computed about a common operating mode (near hover, away from all constraint boundaries) and is analogous to a finite-horizon LQR solution. Consequently, it is applicable to non-aggressive portions of the test scenario, as is shown by the high usage times in the second column, but still requires substantial use of the intermediate controller for aggressive motion (nearly a 2:1 ratio for usage duration). However, as additional, specialized controllers are computed, NPE's reliance on the intermediate controller decreases until the system operates solely using the learned local feedback control laws (**R5**).

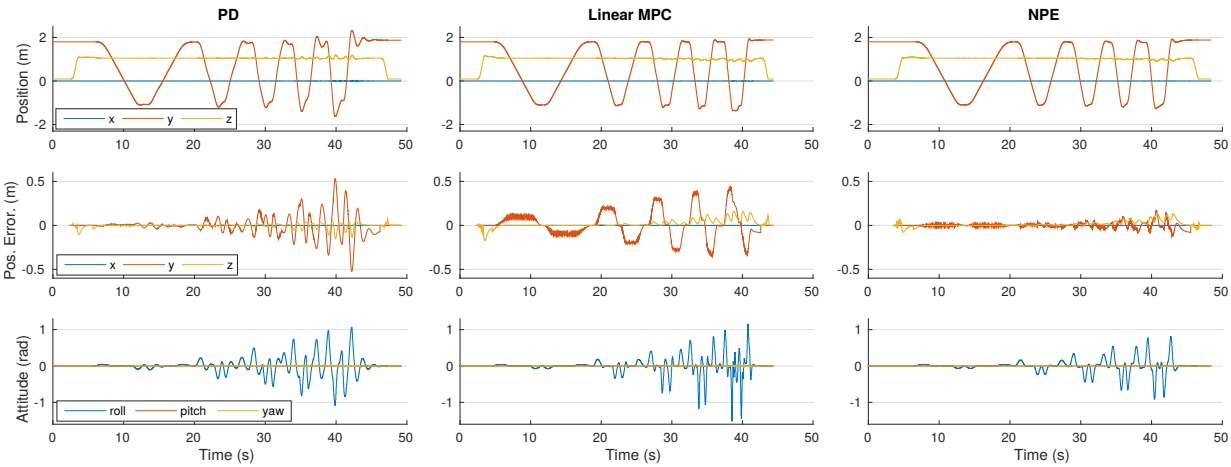
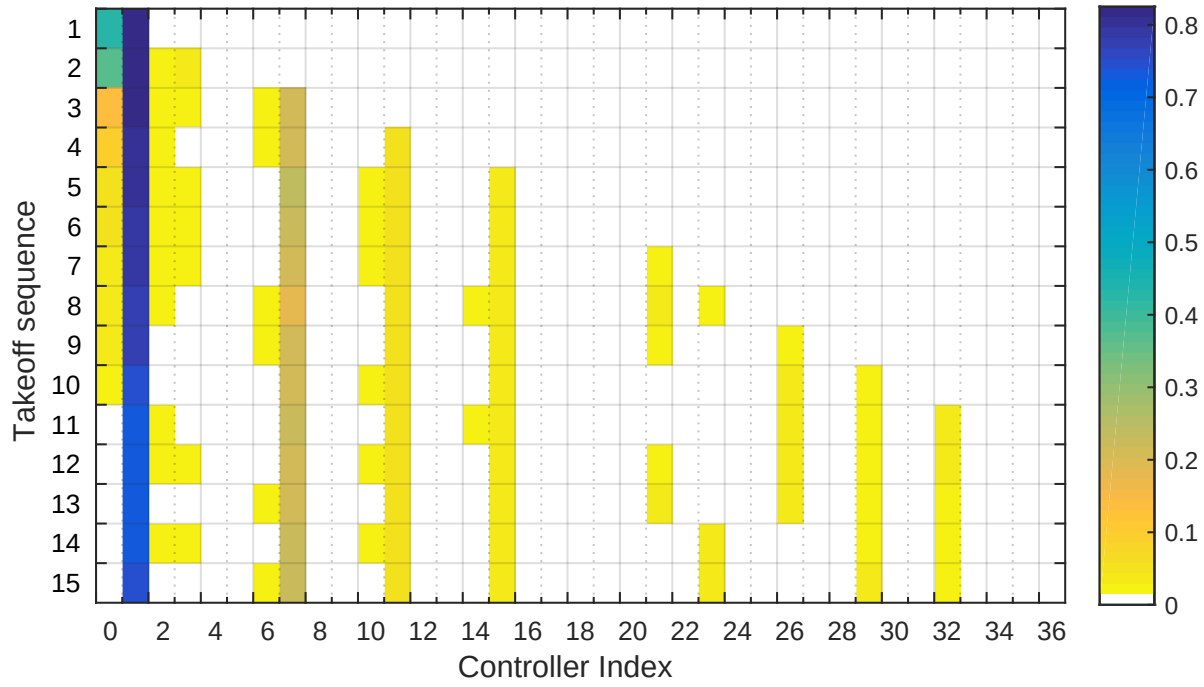


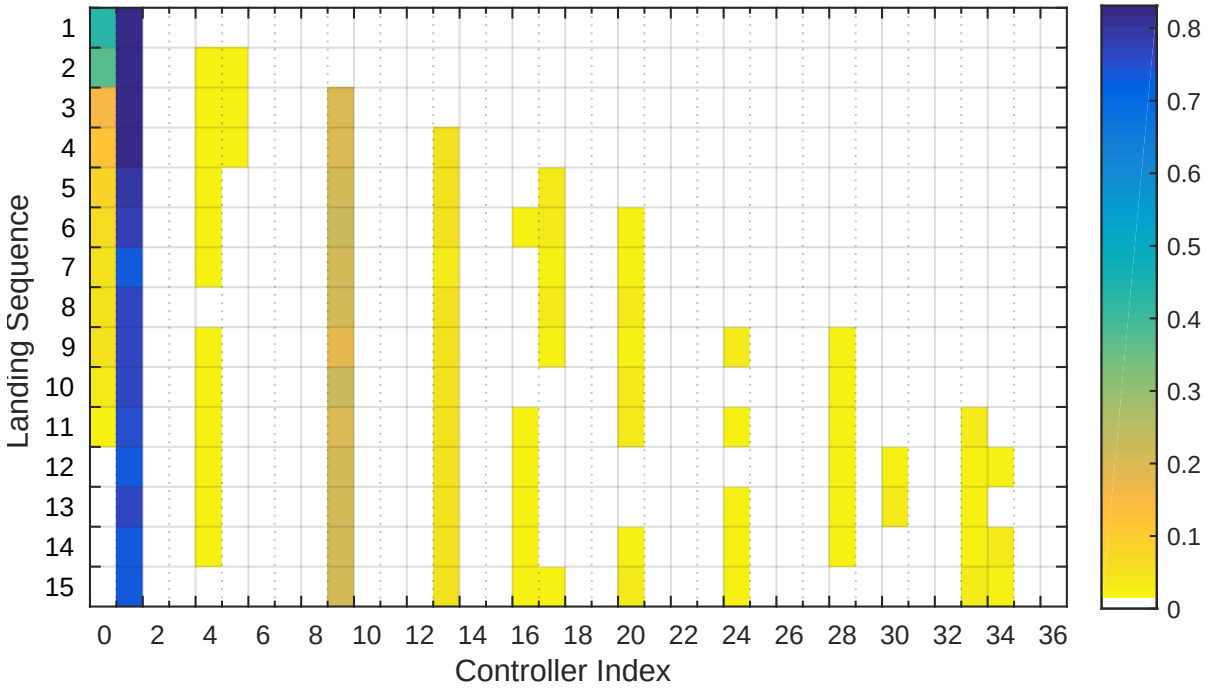
Figure 3.3: Comparison of position, trajectory tracking error, and attitude for the three controllers considered (PD, Linear MPC, and NPE). NPE yields substantially improved tracking performance with reduced overshoot and oscillations.

Although this takeoff-hover-land maneuver only activates the constraints on z-velocity and thrust, as shown in Fig. 3.5, NPE computes 36 different controllers corresponding to combinations of these constraints over the prediction horizon. More diverse maneuvers will activate far more constraints, especially due to the coupling in the nonlinear dynamics, further emphasizing the need for a bounded set of candidate controllers. Figure 3.5 also shows that NPE largely satisfies these constraints (**R2**). The minimal violations observed are due to unmodeled dynamics (such as the motor time constant) resulting small prediction errors. However, this effect is independent of the NPE formulation and further illustrates the importance of model fidelity in predictive control.

These learned controllers can be reused in subsequent trials, enabling the NPE-controlled system to leverage previous computation to operate more efficiently. Figure 3.6 shows the controller usage for another set of takeoff and landing sequences where NPE is initialized with the set of controllers learned in the previous trials. As expected, the system immediately leverages the previously computed controllers, and as a result, the intermediate controller is never applied (**R5**). Figure 3.7 shows an overlay of the transitions between controllers for these takeoff and landing sequences, illustrating a consistent behavior in terms of controller switching. The slight



(a) Controller usage over multiple takeoff sequences



(b) Controller usage over multiple landing sequences

Figure 3.4: Total time a controller is applied (in seconds, indicated by color) during a sequence of similar actions. The first column (index 0) corresponds to the intermediate controller, while index 1 corresponds to the first computed controller.

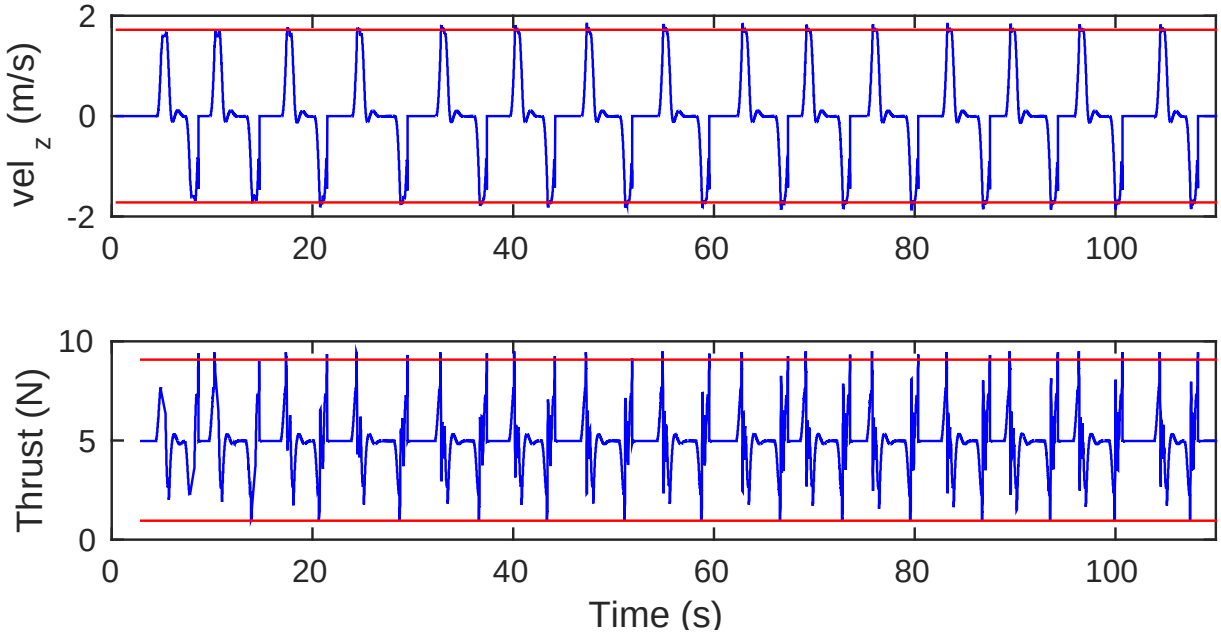
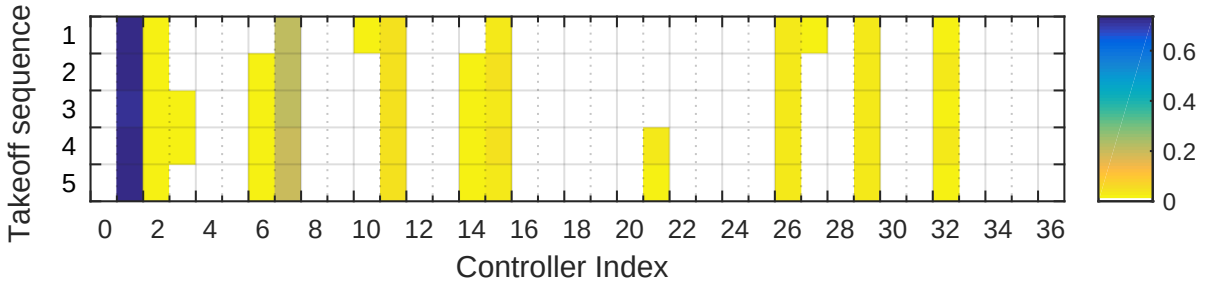


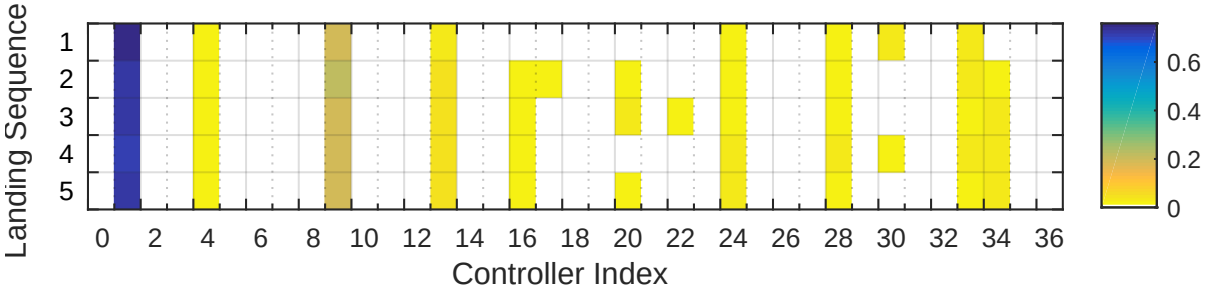
Figure 3.5: Vehicle velocity (along the world z-axis) and commanded thrust over repeated takeoff-hover-land sequences. Red lines indicate constraints enforced in NPE.

variations in the switches can be attributed to the overlap between the valid regions for the learned controllers. This is an effect of the nonlinear dynamics model and KKT tolerance criteria, and removing these overlaps and any redundant controllers is an avenue for future investigation.

One of the key performance metrics for NPE is solution speed. The low computational cost of NPE is illustrated in Table 3.1, which provides statistics on the compute times per component (controller query, intermediate controller, NLP, local QP, and adding a new controller) for the scenario shown in Fig. 3.4. The controller query and intermediate controller easily achieve the requisite 200 Hz, while the more computationally expensive components are run in parallel with decreasing frequency (**R3**). The first row of the table also shows that the NLP is only solved 36 times, which is in stark contrast to standard NMPC solutions that would require solving the NLP in each of the 20807 control iterations. This demonstrates that NPE is an effective real-time model predictive control methodology for nonlinear dynamic systems, such as a quadrotors, where linearity assumptions are degraded during aggressive motions.

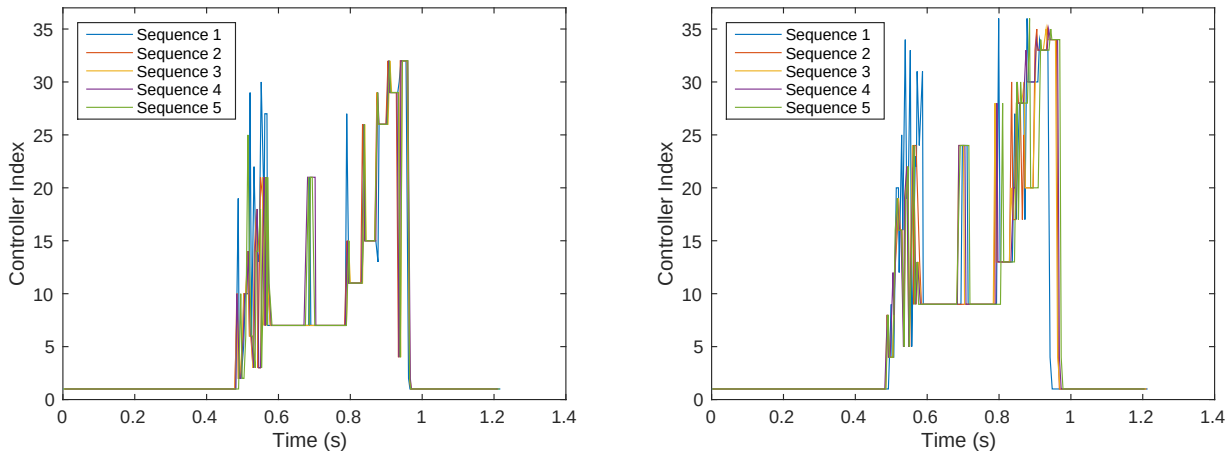


(a) Learned controller usage over multiple takeoff sequences



(b) Learned controller usage over multiple landing sequences

Figure 3.6: Total controller application time for a sequence of actions using previously computed controllers. The first column shows the intermediate controller is never used.



(a) (b)

Figure 3.7: Overlay of controller switches illustrating similarity across trials.

Table 3.1: Solution times for NPE in simulation, including the number of control iterations over which the statistics are computed.

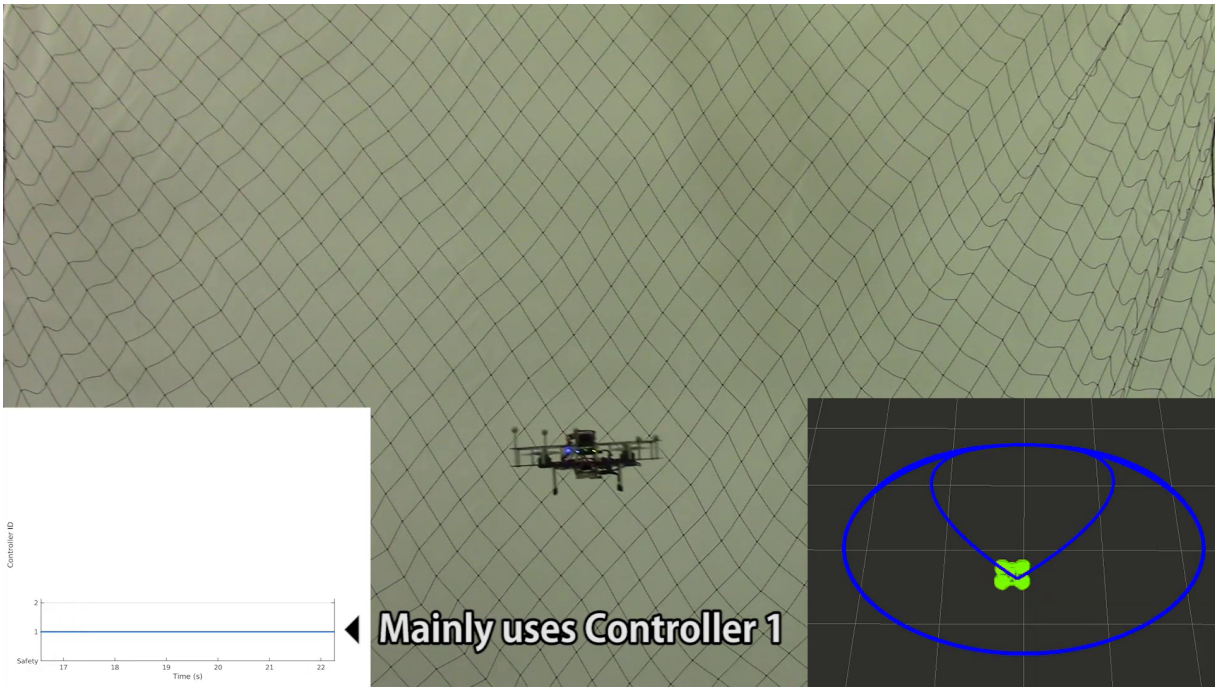
	Query	Interm. Ctrl.	NLP	Local QP	Add Element
Iterations	20807	1061	36	36	36
Mean (ms)	**1.107**	0.923	1412.6	6.232	1.404
Std. Dev. (ms)	**0.678**	0.781	1063.5	1.310	0.827

3.2.2 Experimental Validation

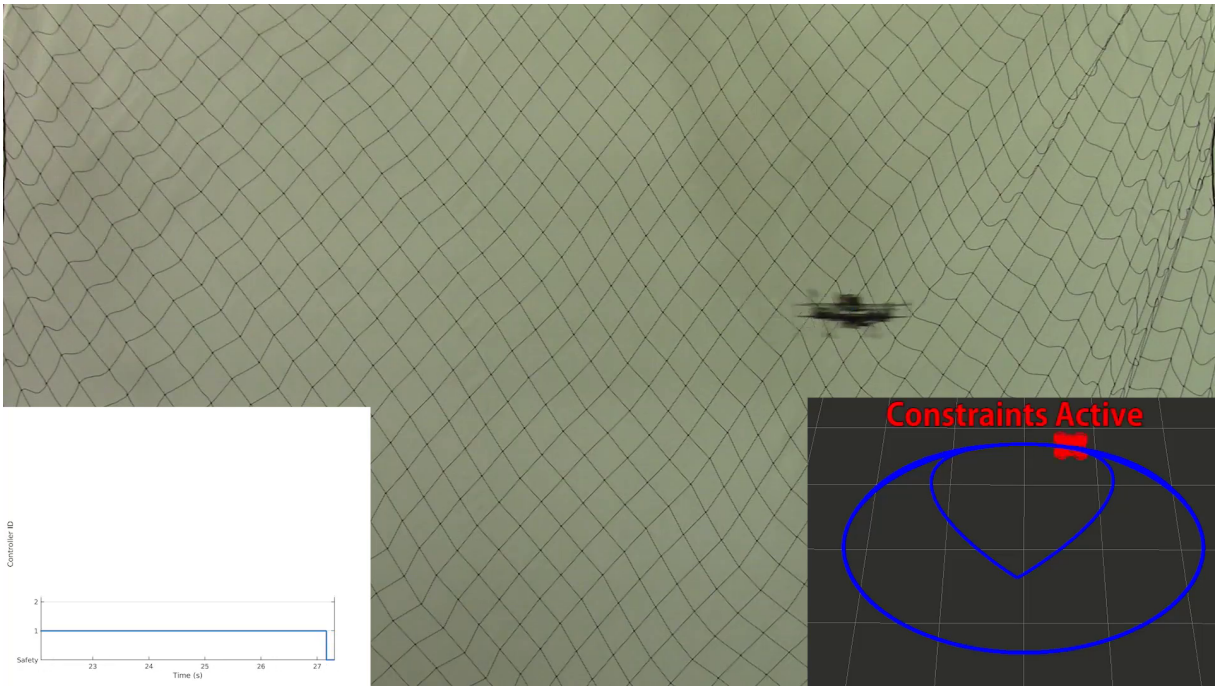
We further demonstrate the realtime constrained trajectory tracking capabilities of the NPE algorithm via flight experiments using the small 790 g quadrotor described in Appendix A.3.1. We employ a cascaded control setup [81] running instances of NPE for both the outer control loop (translational dynamics) and inner control loop (rotational dynamics). Each NPE instance operates over a 20-step (control iteration) horizon with constraints on velocity, attitude, and control commands. Figure 3.8 shows a series of snapshots of the quadrotor tracking the elliptical trajectory shown in the insets on the right. The insets on the left indicate the controller being applied and whether a new controller is being computed. As these snapshots show, NPE achieves sufficiently fast update times to maintain stability (**R1**) and correctly identifies when the quadrotor is approaching constraint boundaries, thus generating new, reusable controllers (**R5**).

To evaluate constraint satisfaction performance, we note that the elliptical trajectory commands a maximum velocity of 1.1 m/s in the y-axis, while NPE is given a speed limit of 1.0 m/s (x-axis velocity, z-axis velocity, attitude, and control input constraints are omitted as they are rarely activated during this trajectory). Figure 3.9 compares the velocity profiles for tracking this ellipse using finite-horizon LQR and NPE. As LQR is an unconstrained formulation, it represents the nominal velocity profile expected from tracking this trajectory with the given cost function. However, NPE enforces the 1.0 m/s constraint, and this results in an altered velocity profile that largely obeys the velocity bound with minor violations (**R2**) likely stemming from unmodeled dynamics.

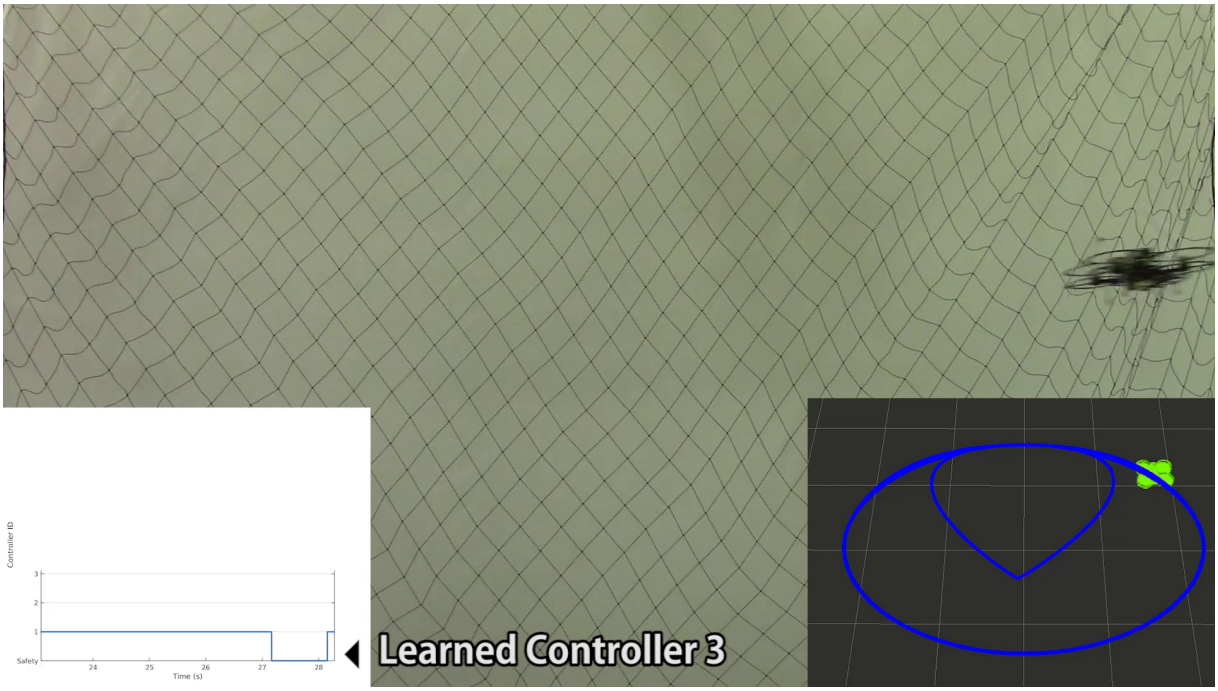
With both instances of NPE running onboard the quadrotor's ODROID computer (detailed in Appendix A.3.1), we also obtain a realistic assessment of the algorithm's computational efficiency. Table 3.2 and Table 3.3 show similar patterns to the simulation results with the NPE query times (in bold) dramatically outperforming the NLP and QP. Additionally, both the outer- and inner-loop instances of NPE reliably achieve query times that permit the desired control rates of 100 Hz and 200 Hz, respectively (**R3**). As noted above, the ellipse trajectory does not activate the



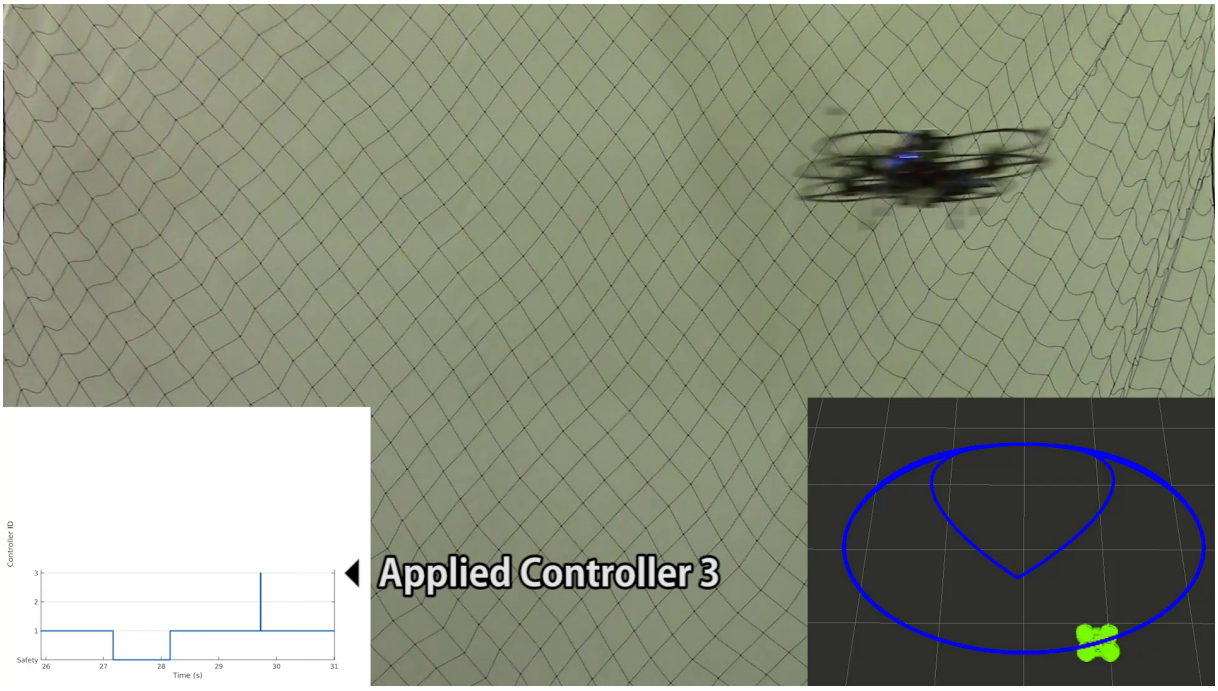
(a) Controller 1 enables nominal flight



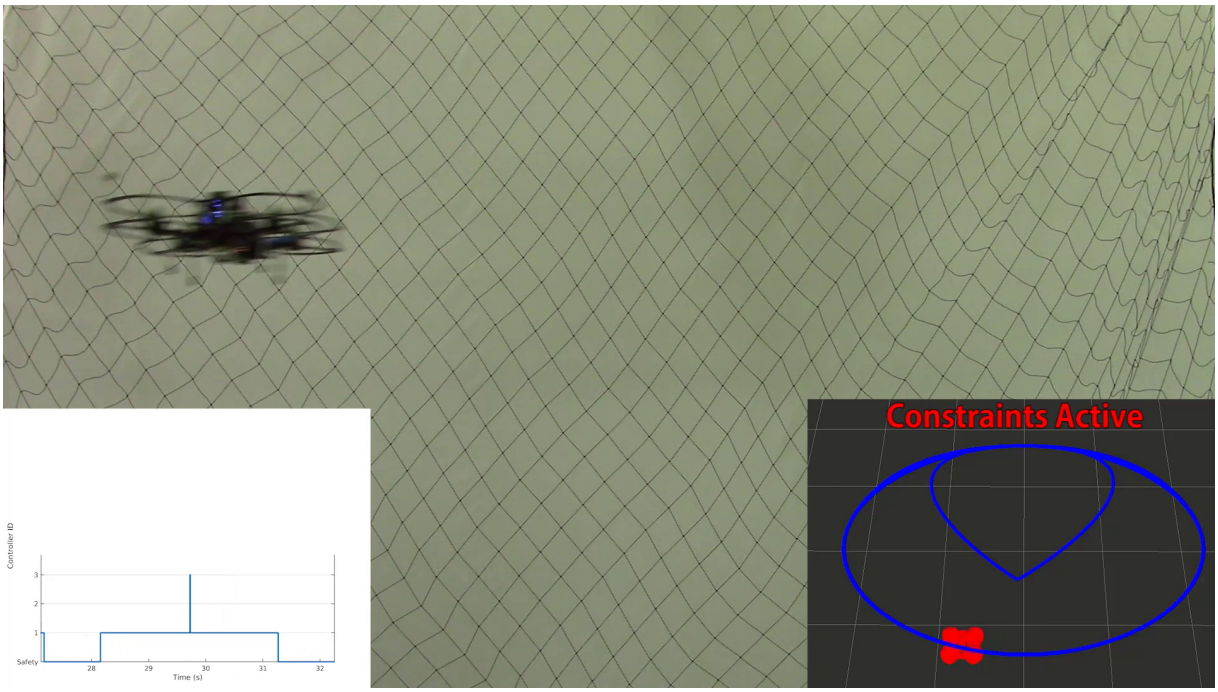
(b) Velocity constraint becomes activate and starts new controller computation



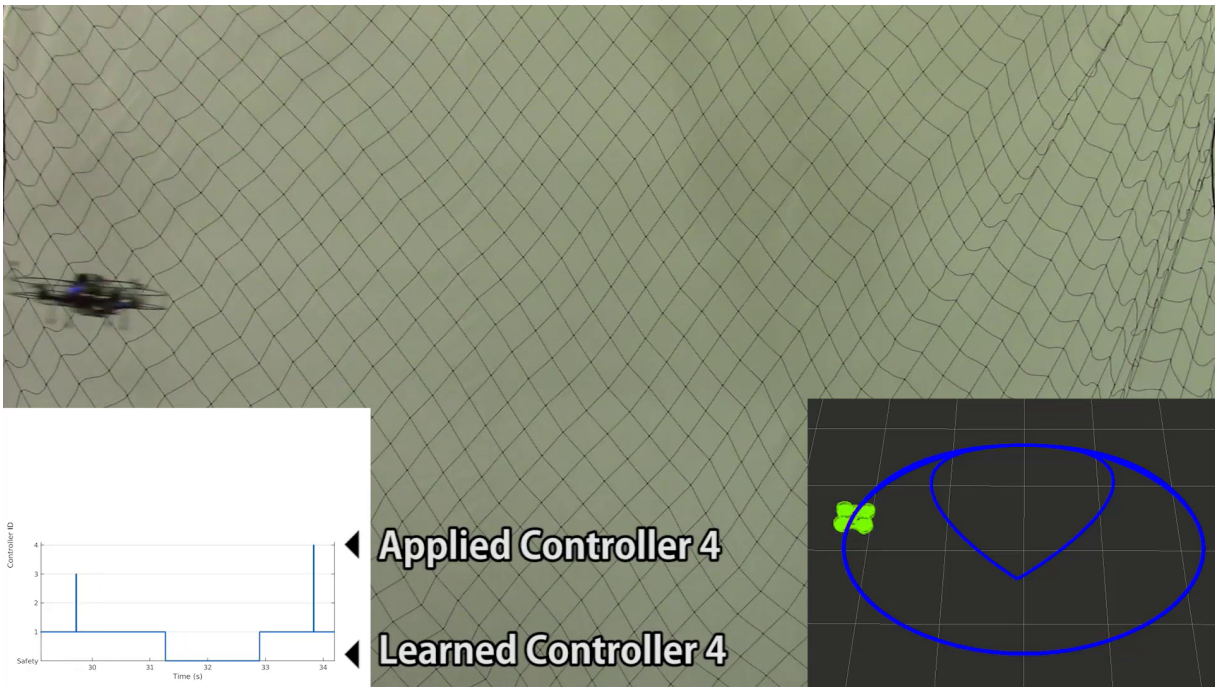
(c) Controller 3 learned to handle velocity constraint



(d) Reapplies Controller 3 to obey velocity constraint at another point in the trajectory



(e) Different velocity constraint become active



(f) Learns Controller 4 and reuses it to obey velocity constraint again

Figure 3.8: Snapshots of the quadrotor tracking an ellipse using Nonlinear Partial Enumeration. The inset images in each frame show controller usage (left) and trajectory visualization with active constraint alerts (right).

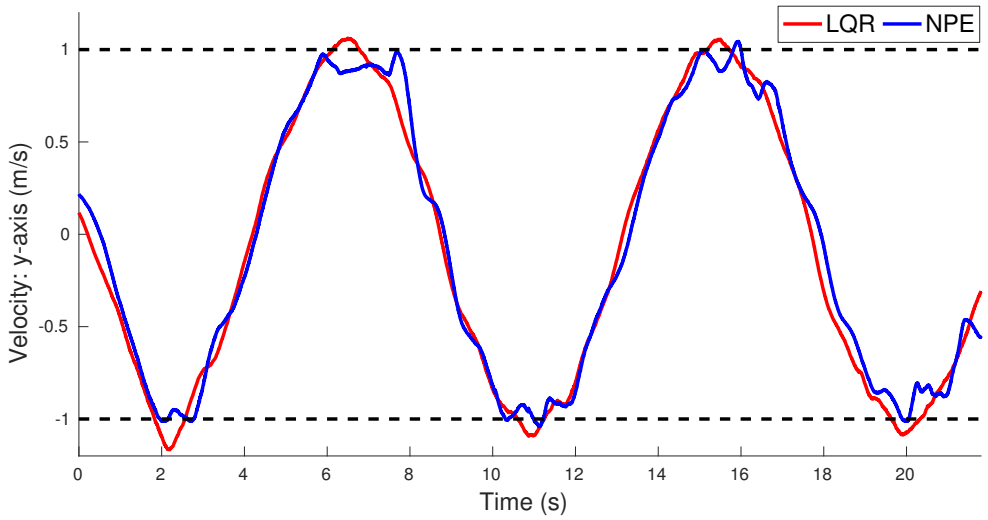


Figure 3.9: Comparison of y-axis velocity profiles for the experimental platform tracking the elliptical trajectory using LQR and NPE, where NPE aims to enforce the velocity limits indicated by the dashed lines.

inner-loop constraints frequently, and this is reflected in the low number of controllers computed and the sub-millisecond query times. These sub-millisecond query times also indicate that the inner-loop can easily compute and query multiple additional controllers without compromising runtime.

Table 3.2: Solution times for the *outer*-loop NPE running at 100 Hz onboard the experimental platform, including the number of control iterations over which the statistics are computed.

	Query	Interm. Ctrl.	NLP	Local QP	Add Element
Iterations	4854	433	7	7	7
Mean (ms)	**3.132**	4.575	2504.47	32.961	29.994
Std. Dev. (ms)	**3.282**	2.083	1.852	5.624	9.868

Table 3.3: Solution times for the *inner*-loop NPE running at 200 Hz onboard the experimental platform, including the number of control iterations over which the statistics are computed.

	Query	Interm. Ctrl.	NLP	Local QP	Add Element
Iterations	11305	11	2	2	2
Mean (ms)	**0.597**	4.043	10.248	49.506	27.105
Std. Dev. (ms)	**0.322**	1.868	2.742	4.465	9.551

3.3 Conclusions

In this section, we have presented Nonlinear Partial Enumeration (NPE) as a fast solution strategy for nonlinear model predictive control (NMPC). NPE extends the linear PE algorithm to use a nonlinear model for more accurate motion prediction and, with minimal online optimization, produces a piecewise-affine controller covering the relevant regions of the state-space. A set of simulation studies focused on aggressive trajectory control for a quadrotor micro aerial vehicle and experimental trials with NPE running onboard a small-scale quadrotor validate the algorithm's real-time control capabilities. Moreover, these results demonstrate that NPE outperforms other fast control methodologies and enables reuse of learned controllers in subsequent flights, thereby reducing the overhead associated with re-solving the optimizations online.

Chapter 4

Experience-driven Predictive Control

Although the Nonlinear Partial Enumeration (NPE) algorithm enables high-rate control that aims to replicate NMPC performance, it also assumes the plant dynamics are known. Moreover, the use of the nonlinear program (NLP) solution as a feedforward term conflicts with the introduction of an adaption scheme to mitigate the effects of uncertain dynamics. Therefore, in this chapter we propose an Experience-driven Predictive Control (EPC) methodology that combines aspects of NPE with online model learning, e.g. via Locally Weighted Projection Regression (LWPR) or Incremental Sparse Spectrum Gaussian Process Regression (ISSGPR). As in NPE, EPC leverages an online-updated database of past experiences in order to achieve high-rate, locally-optimal feedback control with constraint satisfaction. However, we sacrifice the NLP-based feedforward term to parameterize the learned feedback control laws by the system dynamics, thus enabling online adaptation to model perturbations.

4.1 Approach

This section details the proposed Experience-driven Predictive Control (EPC) algorithm for fast, adaptive, nonlinear model predictive control. In the context of predictive control, we first define experience to be the relationship between previous states, references, and system dynamics

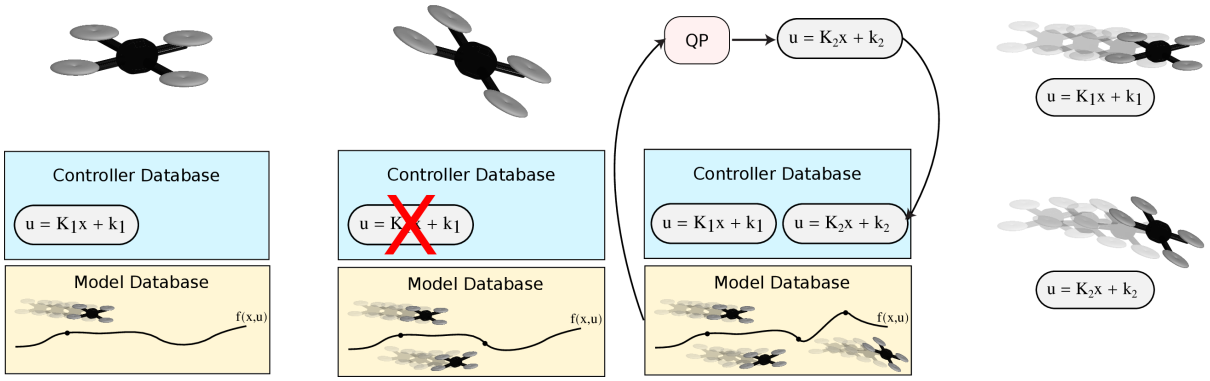


Figure 4.1: Overview of the proposed approach that constructs online an experience database consisting of parameterized feedback controllers and dynamics models. (a) A MAV operates away from constraint boundaries enabling it to apply a controller in the database while the dynamics model continues to be updated. (b) A new controller is added to the experience database as the MAV transitions to more aggressive flight and the updated dynamics model predicts that the system state is approaching a constraint boundary. (c) The MAV reuses controllers in the database based on the state evolution predicted by the current estimate of its dynamics model.

models and the optimal control law applied at that time. Past dynamics models capture the effects of uncertainty on observed system evolution, while previous states capture the system's behavior under optimal control policies for a given dynamics model. Therefore, EPC constructs and leverages a two-part representation of past experiences to improve the accuracy of its finite-horizon lookahead. The first is the set of basis functions maintained by the online model learner (either Locally Weighted Projection Regression (LWPR) [68] or Incremental Sparse Spectrum Gaussian Process Regression (ISSGPR) [69]) that capture observed variations in the system dynamics. The second is a mapping from states and references to locally optimal controllers that is updated online and is parameterized by the current estimate of the vehicle dynamics. The resulting algorithm is summarized in Fig. 4.1, as applied to a quadrotor micro air vehicle (MAV), and detailed below.

4.1.1 Online Model Adaptation

Predictive control techniques for nonlinear systems employ either a nonlinear dynamics model that incurs the complexity of solving nonlinear programs or a more computationally efficient local approximation of the nonlinear dynamics. Therefore, given the nonlinear dynamics $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$, nominal state $\mathbf{x}^*$, and nominal control $\mathbf{u}^*$, we define $\bar{\mathbf{x}} = \mathbf{x} - \mathbf{x}^*$ and $\bar{\mathbf{u}} = \mathbf{u} - \mathbf{u}^*$ and derive an affine approximation of the dynamics via a first-order Taylor series expansion, $\bar{\mathbf{x}}_{k+1}^{\text{nom}} = \mathbf{A}\bar{\mathbf{x}}_k + \mathbf{B}\bar{\mathbf{u}}_k + \mathbf{c}$. We can then extend this model with an online-learned component that estimates perturbations to the nominal model, including nonlinearities, modeling errors, and unmodeled exogenous forces. As mentioned above, we consider two online model learning techniques, LWPR and ISSGPR (see Ch. 2.5 for additional details). Both techniques model a nonlinear function (from an input $\mathbf{z}$ to an output $\mathbf{p}$) by a Gaussian-weighted combination of simple basis functions (linear in LWPR and sinusoidal in ISSGPR). As these models can be updated incrementally with new data, they are able to retain information on past experiences while adapting their estimates to changing dynamics.

LWPR-based Model Learner

LWPR updates its estimate incrementally via partial least squares, which has $\mathcal{O}(|\mathbf{z}|)$ complexity, making it well-suited to real-time operation. Partial least squares projects the inputs onto a lower dimensional space defined by projection direction vectors $\boldsymbol{\nu}_r$ and $\boldsymbol{\rho}_r$, as detailed in [68]. It also computes slope coefficients β_r for each projection direction and an offset β_0 to generate a prediction of a scalar output. Therefore, following Mitrovic, et al. [70], we fit the dynamics model element-wise: for the i^{th} element in $\mathbf{p}$, local linear model j (with r_j projection directions)

is given by

$$\begin{aligned}\Psi_j(\mathbf{z}) &= \beta_0 + \begin{bmatrix}\beta_1, \ldots, \beta_{r_j}\end{bmatrix} \begin{bmatrix}\boldsymbol{\nu}_1^{\mathrm{T}} \\ \boldsymbol{\nu}_2^{\mathrm{T}}\mathbf{P}_1 \\ \vdots \\ \boldsymbol{\nu}_{r_j}^{\mathrm{T}}(\mathbf{P}_1 \cdots \mathbf{P}_{r_j-1})\end{bmatrix} (\mathbf{z} - \mathbf{m}_j) \\ &= \alpha_j + \boldsymbol{\beta}_j^T (\mathbf{z} - \mathbf{m}_j)\end{aligned}$$

where $\mathbf{P}_r = \mathbf{I} - \text{diag}(\boldsymbol{\rho}_r) \begin{bmatrix}\boldsymbol{\nu}_r, \ldots, \boldsymbol{\nu}_r\end{bmatrix}^{\mathrm{T}}$. The prediction model (consisting of N_i local models with weights w_j defined by a Gaussian kernel with mean $\mathbf{m}_j$ and covariance $\mathbf{D}_j$) is

$$\begin{aligned}p_i(\mathbf{z}) &= \frac{1}{W} \sum_{j=1}^{N_i} w_j(\mathbf{z}) \Psi_j(\mathbf{z}) \\ w_j(\mathbf{z}) &= \exp\left(-\frac{1}{2}(\mathbf{z} - \mathbf{m}_j)^T \mathbf{D}_j (\mathbf{z} - \mathbf{m}_j)\right) \\ W &= \sum_{j=1}^{N_i} w_j(\mathbf{z})\end{aligned}$$

ISSGPR-based Model Learner

Although LWPR can more accurately represent accumulated experiences via its local models, we also consider ISSGPR due to its potential for faster model adaptation [82]. Prediction via ISSGPR follows a similar structure to standard Gaussian process regression [83], except the radial basis function kernel is approximated by a vector of D sinusoidal features with random frequencies,

$$\boldsymbol{\phi}(\mathbf{z}) = \frac{\sigma_f}{\sqrt{D}} \left[\cos(\boldsymbol{\omega}_1^{\mathrm{T}}\mathbf{z}), \sin(\boldsymbol{\omega}_1^{\mathrm{T}}\mathbf{z}), \ldots, \cos(\boldsymbol{\omega}_D^{\mathrm{T}}\mathbf{z}), \sin(\boldsymbol{\omega}_D^{\mathrm{T}}\mathbf{z})\right]^{\mathrm{T}}$$

where the parameter σ_f corresponds to the signal variance, the frequencies $\boldsymbol{\omega} \sim \mathcal{N}(\mathbf{0}, \mathbf{M})$, and $\mathbf{M}$ is a diagonal matrix of characteristic lengths scales that reflect the size and importance of

each input dimension [69]. As ISSGPR also produces a scalar output, we again fit the dynamics model element-wise.

Dynamics Model Error Prediction

With either learning algorithm, we define $\mathbf{z} = \begin{bmatrix} \mathbf{x}_k^{\mathrm{T}} & \mathbf{u}_k^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$ and $\mathbf{p} = \bar{\mathbf{x}}_{k+1} - \bar{\mathbf{x}}_{k+1}^{\mathrm{nom}}$. The corresponding prediction output $\hat{\mathbf{p}} = \begin{bmatrix} p_0, p_1, \ldots \end{bmatrix}^{\mathrm{T}}$ gives the estimated perturbation to the system dynamics at a query point $\mathbf{z}$. The total predictive dynamics model is then given by

$$\begin{aligned} \bar{\mathbf{x}}_{k+1} &= \bar{\mathbf{x}}_{k+1}^{\mathrm{nom}} + \hat{\mathbf{p}} \\ &= \mathbf{A}\bar{\mathbf{x}}_k + \mathbf{B}\bar{\mathbf{u}}_k + \mathbf{c} + \hat{\mathbf{p}} \\ &= \mathbf{A}\bar{\mathbf{x}}_k + \mathbf{B}\bar{\mathbf{u}}_k + \tilde{\mathbf{c}} \end{aligned} \tag{4.1}$$

As we learn the perturbation model online, the learner (particularly LWPR) may initially return high-variance estimates when the system enters a new region of the input space (i.e., values of $\mathbf{z}$ for which the system has minimal experience). Therefore, to limit the effects of the resulting transients in the estimate, we introduce a simple gate based on the model uncertainty maintained by the learner. If model uncertainty is high at a given query point, we instead use a zero-order hold on the previous estimate. As the system continues to gain experience in its operating domain, this gate will cease to be applied.

Finally, following the insight from $\mathcal{L}_1$ adaptive control [23], we introduce a low-pass filter on the disturbance estimate before it is incorporated into the predictive model (4.1). This enables LWPR and ISSGPR to learn the perturbation model quickly while limiting changes to system dynamics to be within the bandwidth of the system.

4.1.2 Receding-Horizon Control Formulation

The use of an affine model (4.1) that automatically adapts to capture the effects of nonlinearities and unmodeled dynamics permits a simplified optimal control formulation for EPC relative to techniques such as nonlinear partial enumeration (NPE), which requires solving a nonlinear program due to the general nonlinear dynamics model. Taking the current state as the nominal state, $\mathbf{x}^* = \mathbf{x}_0$, and given N reference states $\mathbf{r}_1, \ldots, \mathbf{r}_N$, let $\bar{\mathbf{r}} = \mathbf{r} - \mathbf{x}^*$. We can then formulate the receding-horizon control problem as a quadratic program:

$$\begin{aligned} \underset{\bar{\mathbf{u}}_k}{\operatorname{argmin}} \sum_{k=0}^{N-1} \frac{1}{2}(\bar{\mathbf{x}}_{k+1} - \bar{\mathbf{r}}_{k+1})^{\mathrm{T}} \mathbf{Q}_{k+1}(\bar{\mathbf{x}}_{k+1} - \bar{\mathbf{r}}_{k+1}) + \frac{1}{2}(\bar{\mathbf{u}}_k - \bar{\mathbf{u}}_{\hat{\mathbf{p}}})^{\mathrm{T}} \mathbf{R}_k(\bar{\mathbf{u}}_k - \bar{\mathbf{u}}_{\hat{\mathbf{p}}}) \\ \text{s.t.} \quad \bar{\mathbf{x}}_{k+1} = \mathbf{A}\bar{\mathbf{x}}_k + \mathbf{B}\bar{\mathbf{u}}_k + \tilde{\mathbf{c}} \\ \mathbf{G}_{\mathbf{x}_{k+1}} \bar{\mathbf{x}}_{k+1} \leq \mathbf{g}_{\mathbf{x}_{k+1}} \\ \mathbf{G}_{\mathbf{u}_k} \bar{\mathbf{u}}_k \leq \mathbf{g}_{\mathbf{u}_k} \\ \forall\, k = 0, \ldots, N-1 \end{aligned} \tag{4.2}$$

where $\tilde{\mathbf{c}} = \mathbf{c} + \hat{\mathbf{p}}$. If a control input, $\bar{\mathbf{u}}_{\hat{\mathbf{p}}}$, can be derived from the model adaptation term (e.g., if $\hat{\mathbf{p}}$ is an acceleration disturbance, $\bar{\mathbf{u}}_{\hat{\mathbf{p}}}$ is the corresponding force) we subtract it in the cost function to avoid penalizing disturbance compensation.

To simplify notation, define $\boldsymbol{x} = \left[\bar{\mathbf{x}}_1^{\mathrm{T}}, \ldots, \bar{\mathbf{x}}_N^{\mathrm{T}}\right]^{\mathrm{T}}$, $\boldsymbol{r} = \left[\bar{\mathbf{r}}_1^{\mathrm{T}}, \ldots, \bar{\mathbf{r}}_N^{\mathrm{T}}\right]^{\mathrm{T}}$, $\boldsymbol{u} = \left[\bar{\mathbf{u}}_0^{\mathrm{T}}, \ldots, \bar{\mathbf{u}}_{N-1}^{\mathrm{T}}\right]^{\mathrm{T}}$, $\boldsymbol{u}_{\hat{\mathbf{p}}} = \left[\bar{\mathbf{u}}_{\hat{\mathbf{p}}}^{\mathrm{T}}, \ldots, \bar{\mathbf{u}}_{\hat{\mathbf{p}}}^{\mathrm{T}}\right]^{\mathrm{T}}$

$$\boldsymbol{\mathcal{B}} = \begin{bmatrix} \mathbf{B} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{AB} & \mathbf{B} & \ldots & \mathbf{0} \\ \vdots & \vdots & \ddots & \\ \mathbf{A}^{N-1}\mathbf{B} & \mathbf{A}^{N-2}\mathbf{B} & \ldots & \mathbf{B} \end{bmatrix}, \; \boldsymbol{c} = \begin{bmatrix} \tilde{\mathbf{c}} \\ (\mathbf{A} + \mathbf{I})\,\tilde{\mathbf{c}} \\ \vdots \\ \sum_{i=0}^{N-1} \mathbf{A}^i \tilde{\mathbf{c}} \end{bmatrix},$$

$\boldsymbol{\mathcal{Q}} = \operatorname{diag}(\mathbf{Q}_1, \ldots, \mathbf{Q}_N)$, $\boldsymbol{\mathcal{R}} = \operatorname{diag}(\mathbf{R}_0, \ldots, \mathbf{R}_{N-1})$, $\boldsymbol{\mathcal{G}}_{\mathbf{x}} = \operatorname{diag}(\mathbf{G}_{\mathbf{x}_1}, \ldots, \mathbf{G}_{\mathbf{x}_N})$,

$\boldsymbol{\mathcal{G}}_{\mathbf{u}} = \text{diag}(\mathbf{G}_{\mathbf{u}_0}, \ldots, \mathbf{G}_{\mathbf{u}_{N-1}})$, $\boldsymbol{g}_{\mathbf{x}} = \left[\mathbf{g}_{\mathbf{x}_1}^{\mathrm{T}}, \ldots, \mathbf{g}_{\mathbf{x}_N}^{\mathrm{T}}\right]^{\mathrm{T}}$, and $\boldsymbol{g}_{\mathbf{u}} = \left[\mathbf{g}_{\mathbf{u}_0}^{\mathrm{T}}, \ldots, \mathbf{g}_{\mathbf{u}_{N-1}}^{\mathrm{T}}\right]^{\mathrm{T}}$. Also, noting that $\bar{\mathbf{x}}_0 = \mathbf{0}$, we can rewrite (4.2) as

$$\begin{aligned} \underset{\boldsymbol{u}}{\operatorname{argmin}} \; & \frac{1}{2}(\boldsymbol{x}-\boldsymbol{r})^{\mathrm{T}}\boldsymbol{\mathcal{Q}}(\boldsymbol{x}-\boldsymbol{r}) + \frac{1}{2}(\boldsymbol{u}-\boldsymbol{u}_{\hat{\mathbf{p}}})^{\mathrm{T}}\boldsymbol{\mathcal{R}}(\boldsymbol{u}-\boldsymbol{u}_{\hat{\mathbf{p}}}) \\ \text{s.t.} \quad & \boldsymbol{x} = \boldsymbol{\mathcal{B}}\boldsymbol{u} + \boldsymbol{c} \\ & \boldsymbol{\mathcal{G}}_{\mathbf{x}}\boldsymbol{x} \leq \boldsymbol{g}_{\mathbf{x}} \\ & \boldsymbol{\mathcal{G}}_{\mathbf{u}}\boldsymbol{u} \leq \boldsymbol{g}_{\mathbf{u}} \end{aligned}$$

As in NPE, we construct an equivalent QP entirely in terms of $\boldsymbol{u}$ by substituting the dynamics constraints and dropping constant terms in the cost function

$$\begin{aligned} \underset{\boldsymbol{u}}{\operatorname{argmin}} \; & \frac{1}{2}\boldsymbol{u}^{\mathrm{T}}\boldsymbol{\mathcal{H}}\boldsymbol{u} + \boldsymbol{h}^{\mathrm{T}}\boldsymbol{u} \\ \text{s.t.} \quad & \boldsymbol{\Gamma}\boldsymbol{u} \leq \boldsymbol{\gamma} \end{aligned} \tag{4.3}$$

where $\boldsymbol{\mathcal{H}} = \boldsymbol{\mathcal{B}}^{\mathrm{T}}\boldsymbol{\mathcal{Q}}\boldsymbol{\mathcal{B}} + \boldsymbol{\mathcal{R}}$, $\boldsymbol{h} = \boldsymbol{\mathcal{B}}^{\mathrm{T}}\boldsymbol{\mathcal{Q}}(\boldsymbol{c}-\boldsymbol{r}) - \boldsymbol{\mathcal{R}}\boldsymbol{u}_{\hat{\mathbf{p}}}$,

$$\boldsymbol{\Gamma} = \begin{bmatrix} \boldsymbol{\mathcal{G}}_{\mathbf{x}}\boldsymbol{\mathcal{B}} \\ \boldsymbol{\mathcal{G}}_{\mathbf{u}} \end{bmatrix}, \text{ and } \boldsymbol{\gamma} = \begin{bmatrix} \boldsymbol{g}_{\mathbf{x}} - \boldsymbol{\mathcal{G}}_{\mathbf{x}}\boldsymbol{c} \\ \boldsymbol{g}_{\mathbf{u}} \end{bmatrix}$$

Defining $\boldsymbol{\lambda}$ as the vector of Lagrange multipliers and $\boldsymbol{\Lambda} = \text{diag}(\boldsymbol{\lambda})$, the first two Karush-Kuhn-Tucker (KKT) conditions for optimality (stationarity and complementary slackness) for the QP can then be written as

$$\begin{aligned} \boldsymbol{\mathcal{H}}\boldsymbol{u} + \boldsymbol{h} + \Gamma^{\mathrm{T}}\boldsymbol{\lambda} &= \mathbf{0} \\ \boldsymbol{\Lambda}(\boldsymbol{\Gamma}\boldsymbol{u} - \boldsymbol{\gamma}) &= \mathbf{0} \end{aligned} \tag{4.4}$$

If we only consider the active constraints (i.e., with $\lambda > 0$) for a given solution, we can reconstruct $\boldsymbol{u}$ and $\boldsymbol{\lambda}$ by solving a linear system derived from (4.4), where the subscript a indicates

rows corresponding to the active constraints

$$\begin{bmatrix} \boldsymbol{\mathcal{H}} & \boldsymbol{\Gamma}_a^{\mathrm{T}} \\ \boldsymbol{\Gamma}_a & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{\lambda}_a \end{bmatrix} = \begin{bmatrix} -\boldsymbol{h} \\ \boldsymbol{\gamma}_a \end{bmatrix}$$

Assuming the active constraints are linearly independent (Bemporad, et al. [47] suggest alternatives if this assumption fails), the resulting QP control law, $\boldsymbol{u}$, is affine in the predicted state error, $\boldsymbol{r}$, and parameterized by the system dynamics

$$\boldsymbol{u} = \boldsymbol{\mathcal{E}}_5 \boldsymbol{r} - \left(\boldsymbol{\mathcal{E}}_5 \boldsymbol{c} - \boldsymbol{\mathcal{E}}_4 \boldsymbol{\mathcal{R}} \boldsymbol{u}_{\hat{\mathbf{p}}} + \boldsymbol{\mathcal{E}}_3 \begin{bmatrix} \boldsymbol{g}_{\mathbf{x}}^{+} - \boldsymbol{\mathcal{G}}_{\mathbf{x}} \boldsymbol{c} \\ -\boldsymbol{g}_{\mathbf{x}}^{-} + \boldsymbol{\mathcal{G}}_{\mathbf{x}} \boldsymbol{c} \\ \boldsymbol{g}_{\mathbf{u}}^{+} \\ -\boldsymbol{g}_{\mathbf{u}}^{-} \end{bmatrix}_a \right) \tag{4.5}$$

where $\boldsymbol{\mathcal{E}}_1 = \boldsymbol{\Gamma}_a \boldsymbol{\mathcal{H}}^{-1}$, $\boldsymbol{\mathcal{E}}_2 = -(\boldsymbol{\mathcal{E}}_1 \boldsymbol{\Gamma}_a^{\mathrm{T}})^{-1}$, $\boldsymbol{\mathcal{E}}_3 = \boldsymbol{\mathcal{E}}_1^{\mathrm{T}} \boldsymbol{\mathcal{E}}_2$, $\boldsymbol{\mathcal{E}}_4 = \boldsymbol{\mathcal{H}}^{-1} + \boldsymbol{\mathcal{E}}_3 \boldsymbol{\mathcal{E}}_1$, and $\boldsymbol{\mathcal{E}}_5 = \boldsymbol{\mathcal{E}}_4 \boldsymbol{\mathcal{B}}^{\mathrm{T}} \boldsymbol{\mathcal{Q}}$. Moreover, since the coefficients in (4.5) are all functions of $\mathbf{A}$, $\mathbf{B}$, and $\tilde{\mathbf{c}}$, the overall control law $\kappa(\mathbf{x}_0, \mathbf{r}_1, \ldots, \mathbf{r}_N)$ can be written in terms of a parameterized feedback gain matrix $\mathbf{K}$ and feedforward vector $\mathbf{k}_{\text{ff}}$

$$\kappa(\mathbf{x}_0, \mathbf{r}_1, \ldots, \mathbf{r}_N) = \mathbf{K}(\mathbf{A}, \mathbf{B}, \tilde{\mathbf{c}}) \boldsymbol{r} + \mathbf{k}_{\text{ff}}(\mathbf{A}, \mathbf{B}, \tilde{\mathbf{c}}) \tag{4.6}$$

This parameterization also extends to the KKT condition checks to determine whether a previously computed controller is locally optimal. The active Lagrange multipliers $\boldsymbol{\lambda}_a$ follow a similar

form to the control law

$$\boldsymbol{\lambda}_a = -\boldsymbol{\mathcal{E}}_6 \boldsymbol{r} + \left(\boldsymbol{\mathcal{E}}_6 \boldsymbol{c} - \boldsymbol{\mathcal{E}}_3^{\mathrm{T}} \boldsymbol{\mathcal{R}} \boldsymbol{u}_{\hat{\mathbf{p}}} + \boldsymbol{\mathcal{E}}_2 \begin{bmatrix} \boldsymbol{g}_{\mathbf{x}}^{+} - \boldsymbol{\mathcal{G}}_{\mathbf{x}} \boldsymbol{c} \\ -\boldsymbol{g}_{\mathbf{x}}^{-} + \boldsymbol{\mathcal{G}}_{\mathbf{x}} \boldsymbol{c} \\ \boldsymbol{g}_{\mathbf{u}}^{+} \\ -\boldsymbol{g}_{\mathbf{u}}^{-} \end{bmatrix}_a \right) \tag{4.7}$$

where $\boldsymbol{\mathcal{E}}_6 = \boldsymbol{\mathcal{E}}_3^{\mathrm{T}} \boldsymbol{\mathcal{B}}^{\mathrm{T}} \boldsymbol{\mathcal{Q}}$.

Therefore, instead of storing the affine controller gains and Lagrange multipliers required to evaluate the KKT conditions, it is sufficient to store only the set of active constraints. This enables a memory-efficient implementation for constrained systems. The controller and KKT matrices can then be reconstructed online using (4.5), (4.7), and the current $\mathbf{A}, \mathbf{B}, \tilde{\mathbf{c}}$. Consequently, this parameterized formulation enables us to adapt and apply any previously computed controller, when appropriate according to the KKT conditions, even as the system dynamics evolve. The complete algorithm is detailed below.

4.1.3 EPC Algorithm

As described in Alg. 4.1, EPC constructs a database defined as a mapping $\mathcal{M}$ from experiences to controllers. At the beginning of each control iteration, EPC queries the current state and reference, as well as the current affine model from LWPR, $(\mathbf{A}, \mathbf{B}, \tilde{\mathbf{c}})$. It then queries the parameterized mapping (line 6), and if the KKT conditions are met for an element, applies the corresponding controller. If no controller from prior experience is applicable (line 14), it solves the QP (4.3) to add a new parameterized element to the mapping, updating the stored experiences with the current scenario. In parallel, EPC applies commands from a short-horizon intermediate QP with slack on state constraints (to ensure problem feasibility), in order to maintain a desired control update rate (line 15). As new controllers are added to the database, less valuable controllers (indicated by a lower `importance` score) are removed (line 19) to bound the number of elements

Algorithm 4.1 Experience-driven Predictive Control

$\mathcal{M} \leftarrow \emptyset$ or $\mathcal{M}_{\text{prior}}$
while control is enabled **do**
    $\boldsymbol{x} \leftarrow$ current system state
    $\boldsymbol{r} \leftarrow$ current reference sequence
    $\mathbf{A}, \mathbf{B}, \tilde{\mathbf{c}} \leftarrow$ current dynamics model from LWPR
    for each element $m_i \in \mathcal{M}$ **do**
        Compute $\boldsymbol{u}, \boldsymbol{\lambda}_a$ via (4.5),(4.7)
        if $\boldsymbol{x}, \boldsymbol{r}$ satisfy parameterized KKT criteria **then**
            $\texttt{importance}_i \leftarrow$ current time, sort $\mathcal{M}$
            $\texttt{solution\_found} \leftarrow$ true
            Apply affine control law (4.6) from m_i
        end if
    end for
    if `solution_found` is false **then**
        Apply interm. control via (4.3) with slack variables
        Update QP formulation with current model
        Generate new controller via QP (4.3) (in parallel)
        if $|\mathcal{M}| >$ maximum table size **then**
            Remove element with min. `importance`
        end if
        Add $m_{\text{new}} = (\mathbf{K}, \mathbf{k}_{\text{ff}}, \texttt{importance})$ to $\mathcal{M}$
    end if
end while

that may be queried in one control iteration, thereby ensuring computational tractability.

In addition to introducing adaptation to unmodeled dynamics, the parameterization by experience and the introduction of an online updated linear dynamics model eliminates the most computationally expensive component of NPE - the nonlinear program. Although the nonlinear program does not limit the control rate in NPE, it does limit how quickly new controllers can be computed, consequently limiting the practical horizon length and increasing the dependence on the intermediate controller. With its quadratic program formulation, EPC has the advantage of faster solution times in the parallel thread that can be leveraged to reduce the dependence on the intermediate controller or increase the prediction horizon. Additionally, the nonlinear program solutions in NPE serve as fixed feedforward terms in the resulting affine control laws, precluding a completely adaptive control strategy. With EPC, the local controllers are fully parameterized,

allowing controllers computed using past experience to be adapted to the present scenario.

4.2 Results

To validate the performance of the EPC algorithm, we seek to demonstrate the following results (from the set enumerated in Chapter 1): **R1**: stable control performance, **R2**: constraint satisfaction, **R3**: real-time computation of control commands, **R4.1**: adaptation performance, and **R5**: applicability of experiences to novel scenarios. These results are shown through a set of simulation and experimental studies applying EPC to the problem of controlling a small quadrotor micro air vehicle.

4.2.1 Simulation Studies

We first conduct a series of hardware-in-the-loop simulations on an ODROID-XU4 (2 GHz ARM processor with 2 GB RAM) that satisfies the size, weight, and power limitations of a small, 790 g quadrotor (see Appendix A for details). The commanded trajectories cross a region where strong, exogenous disturbances (e.g., wind) act on the vehicle. We employ a standard hierarchical control setup [81] to track the trajectories, applying EPC separately to the translational and rotational dynamics.

The quadrotor is commanded to fly ten laps at 0.7 m/s around an elliptical trajectory (Fig. 4.2) that intersects a region in which a constant disturbance torque is applied about the x and y axes. Given that the disturbance acts on the rotational dynamics, we focus on the EPC used for attitude control in the following results. The attitude dynamics are modeled by six states (body frame Euler angles and rates) with one torque input about each axis [81], and we select a horizon (N) of 15 control iterations to ensure that the predicted state evolution captures the effects of the control inputs. As attitude controllers are commonly run at rates exceeding 200 Hz to ensure stability of these fast dynamics [20], we note that a viable attitude controller must consistently return a

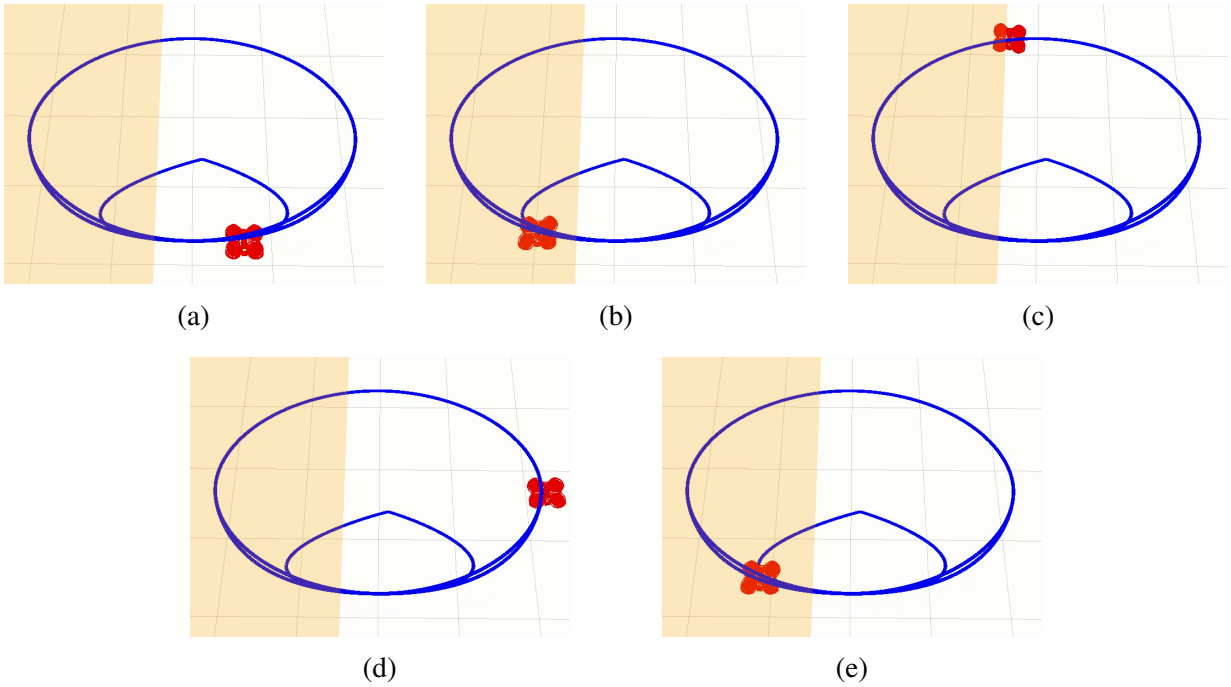


Figure 4.2: Snapshots of the quadrotor executing the elliptical trajectory that traverses the disturbance region (highlighted).

control command within 5 ms.

Constraint Satisfaction

To demonstrate safety under limited control authority, we enforce constraints on the torque control inputs that are more restrictive than the nominal commands that would be applied to track the trajectory. As a result, these constraints are activated repeatedly as the vehicle tracks the trajectory. In order to satisfy these constraints, EPC learns 22 different parameterized feedback control laws, as shown in Fig. 4.3. Moreover, the intermediate controller (denoted controller 0) is only applied in the early laps, indicating that the majority of the controllers are learned quickly and then reused in subsequent laps. This intelligent controller switching also yields reliable constraint satisfaction (**R2**), as shown in Fig. 4.4.

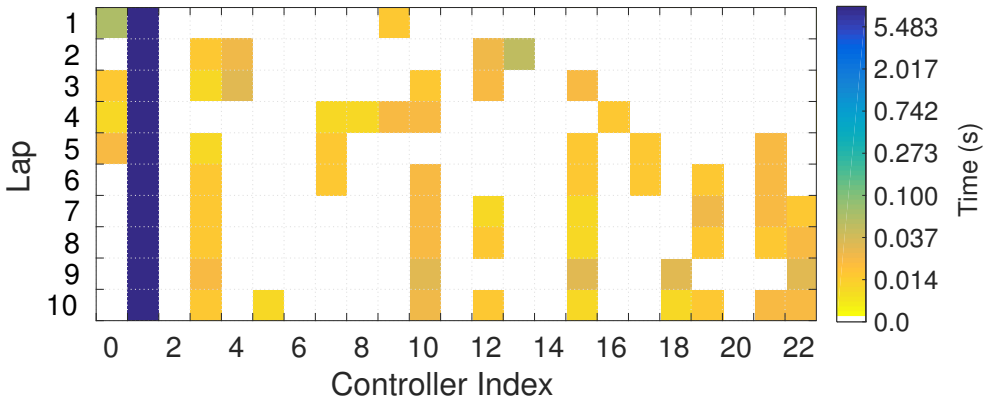


Figure 4.3: Learned controllers are reused in subsequent laps, ultimately eliminating the dependence on the intermediate controller (column 0). Colors denote the total usage time (in seconds) for each controller.

Real-time Computation

Over the course of this trial, the mean time required to query the controller database is 0.77 ms with a standard deviation of 0.75 ms. In contrast, the mean time to solve the equivalent QP is 4.7 ms with a standard deviation of 3.2 ms, which violates the consistent 5 ms command requirement. This confirms that EPC is a computationally efficient approach for adaptive nonlinear model predictive control suitable for high-rate applications, such as attitude control of a quadrotor, even on computationally constrained platforms (**R3**).

Adaptation Performance

In addition to constraint satisfaction, EPC substantially improves trajectory tracking accuracy in the presence of sudden changes to the system dynamics, as shown in Fig. 4.5. As expected, tracking performance improves over time with the accumulation of experience. In addition to extending the controller database, this experience refines the LWPR model. Consequently, the model yields increasingly accurate estimates of the exogenous torques, as shown in Fig. 4.6.

Figure 4.7 illustrates the performance of EPC relative to two baseline approaches: $\mathcal{L}_1$ adaptive control (L1AC)[23] and an adaptive MPC formulation based on a state predictor (Luenberger observer). The gains for the L1AC are selected to match the nominal gains computed by EPC.

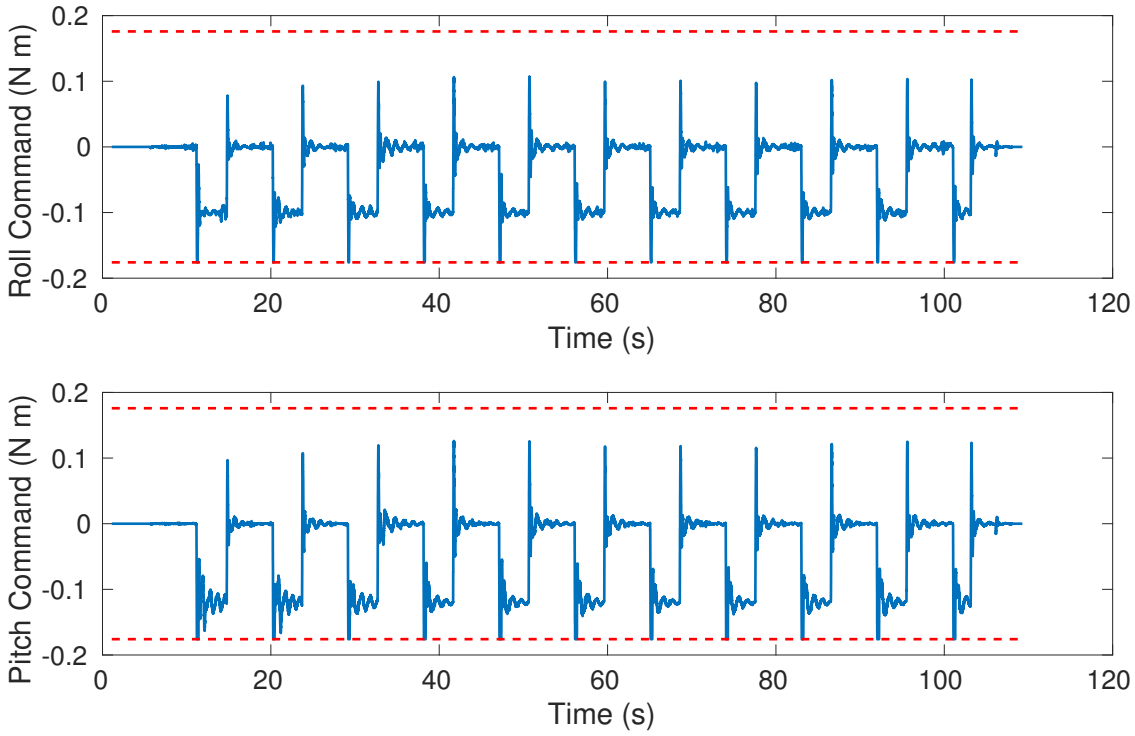


Figure 4.4: EPC successfully satisfies roll and pitch control input constraints (dashed red lines) via controller switching.

The low-pass filter bandwidth is equivalent for both controllers to ensure a fair comparison of the adaptation laws. As the core EPC formulation is equivalent to a quadratic program-based MPC, we consider EPC with the Luenberger observer as the second baseline. Additionally, while EPC embeds the disturbance estimate in the prediction model to enable constraint satisfaction, L1AC adds it as a compensation term to the resulting command. It therefore lacks any safe means of constraint satisfaction, precluding a comparison of constrained control performance. We therefore loosen the EPC control input constraints to aid comparison.

As Fig. 4.7 shows, EPC (after obtaining sufficient experience) reduces peak tracking error by an average of 26.8% relative to $\mathcal{L}_1$ adaptive control. EPC (with LWPR) also reduces peak tracking error by an average of 17.2% relative to the variant with a Luenberger observer, confirming that the improvement relative to L1AC is not simply due to integrating the estimate into the prediction model. Moreover, these results show that the combination of a predictive controller driven by an online learned, reusable model yields significantly improved tracking performance (**R4.1**).

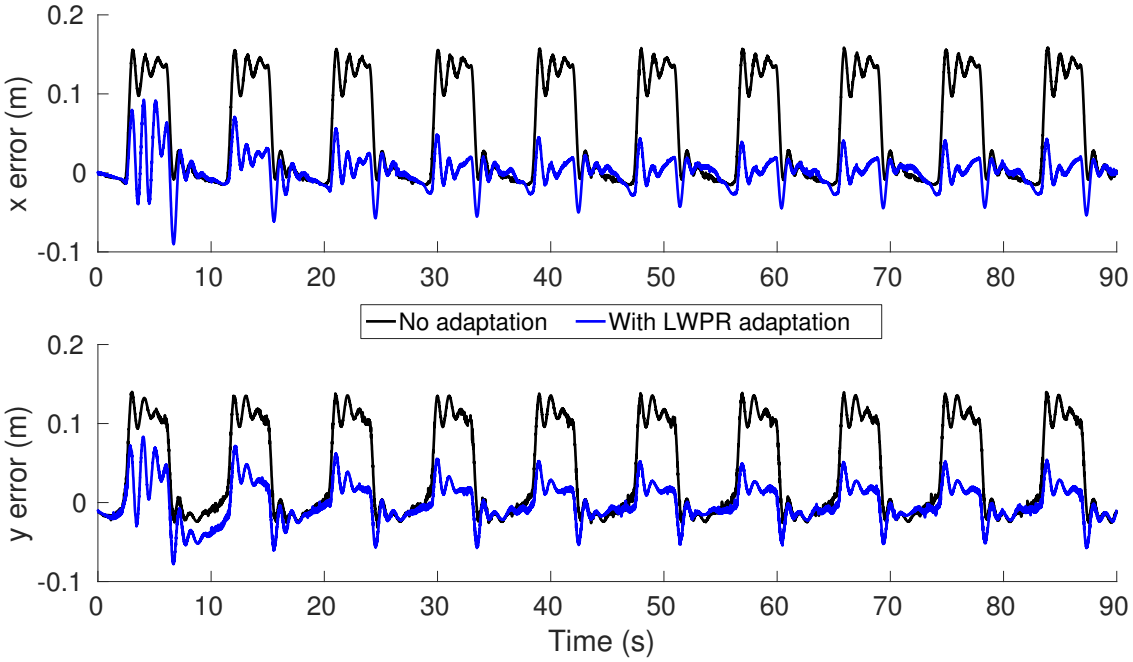


Figure 4.5: Comparison of EPC tracking performance with and without LWPR-based adaptation.

Application to Novel Scenarios

Finally, to evaluate the generalizability of experience, we consider a more complex scenario. Over the course of this 1000 s trial, the quadrotor is commanded to track a series of smooth but random trajectories through the same environment as before. Figures 4.8 and 4.9 show these trajectories, which achieve maximum commanded velocities of 1.7 m/s and accelerations of 5.1 m/s^2. The vehicle dynamics are also perturbed by a stochastic process emulating turbulent air flow, introducing noise into the LWPR training data.

Due to the randomization, the quadrotor enters and exits the disturbance region following a variety of trajectories. The resulting disturbance estimate (Fig. 4.10) shows transient behavior during the initial traversals of the disturbance region (e.g. during the first 200 s of the trial), with disturbance estimate rise times greater than 1.5 s. However, these transients do not reappear, even as the vehicle traverses the region in previously unseen ways while executing this diverse set of trajectories. Moreover, the disturbance estimate has a consistent rise time of approximately 0.5 s

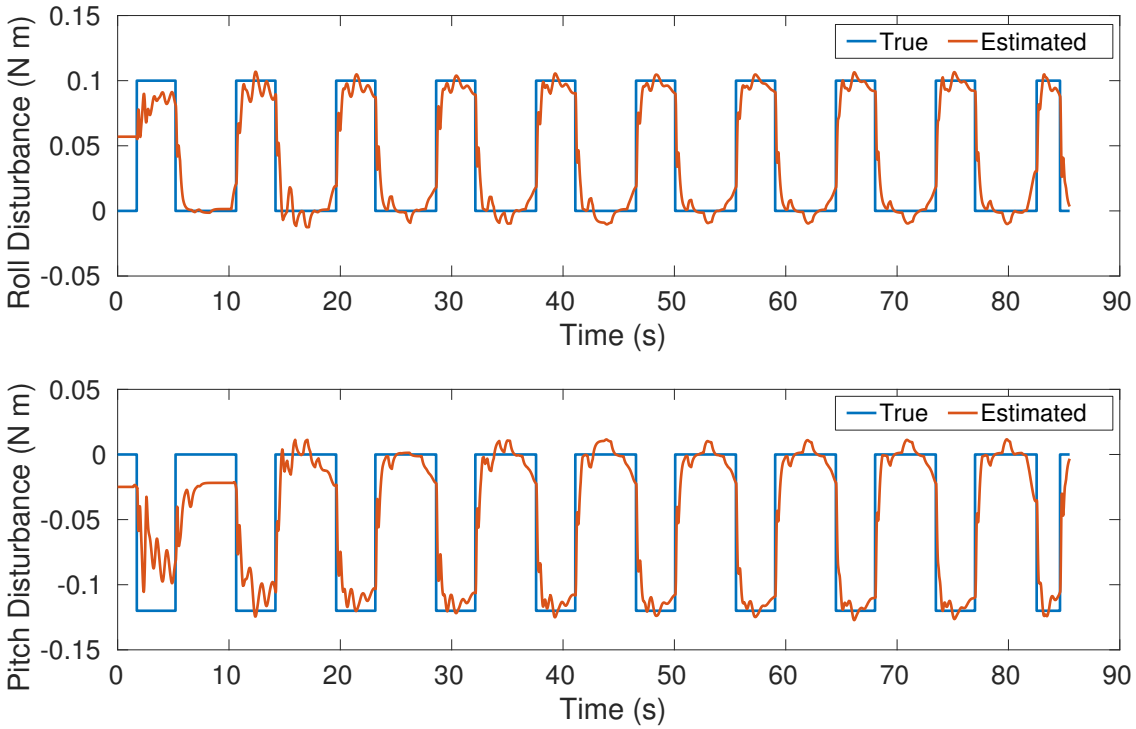


Figure 4.6: LWPR accurately estimates the torque disturbances about the x- and y-axes as it tracks the elliptical trajectory.

for the remainder of the trial (**R4.1**). This indicates that the experience gained through the initial traversals is applicable to the numerous novel scenarios encountered in the future and yields a consistent improvement in disturbance estimation performance (**R4.1**).

The controller also yields stable tracking as expected (**R1**). Even for this long trial with diverse trajectories, EPC only computes 52 controllers to maintain constraint satisfaction, as shown in Fig. 4.11 (**R2**). Additionally, the time to query this database has a mean of 0.30 ms with a variance of 0.29 ms (**R2**). These results again illustrate the computational efficiency of this Experience-driven Predictive Control approach and its suitability for use on flight-viable constrained computational platforms (**R3**).

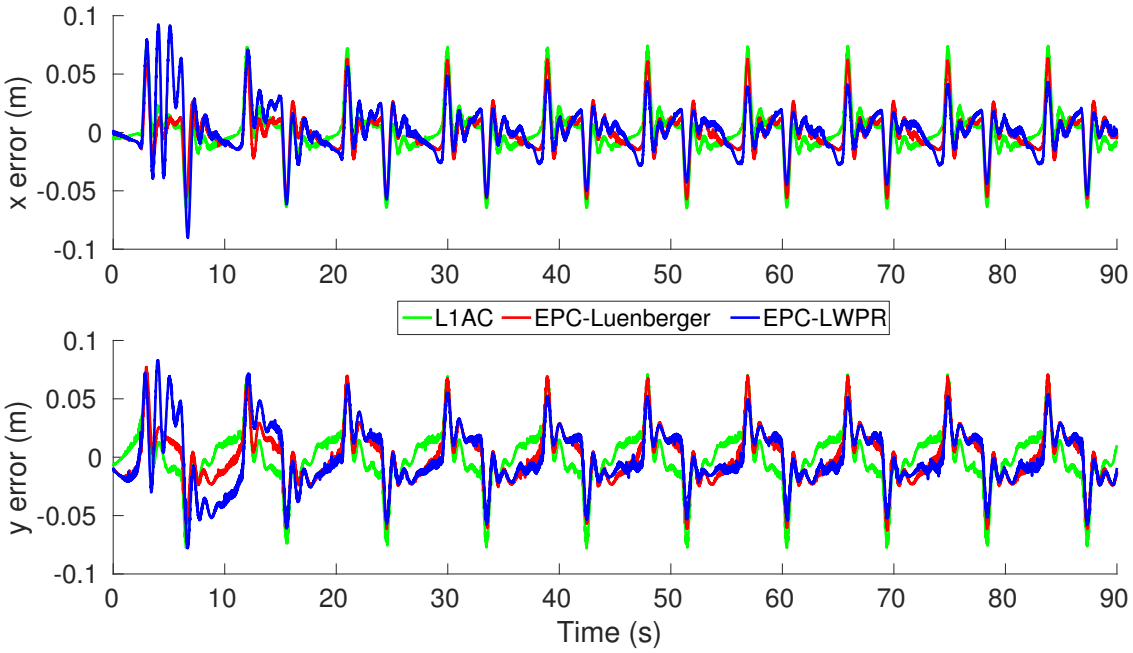


Figure 4.7: EPC with LWPR yields improved position tracking error compared to $\mathcal{L}_1$ adaptive control (L1AC) and EPC with a simple state predictor (EPC-Luenberger).

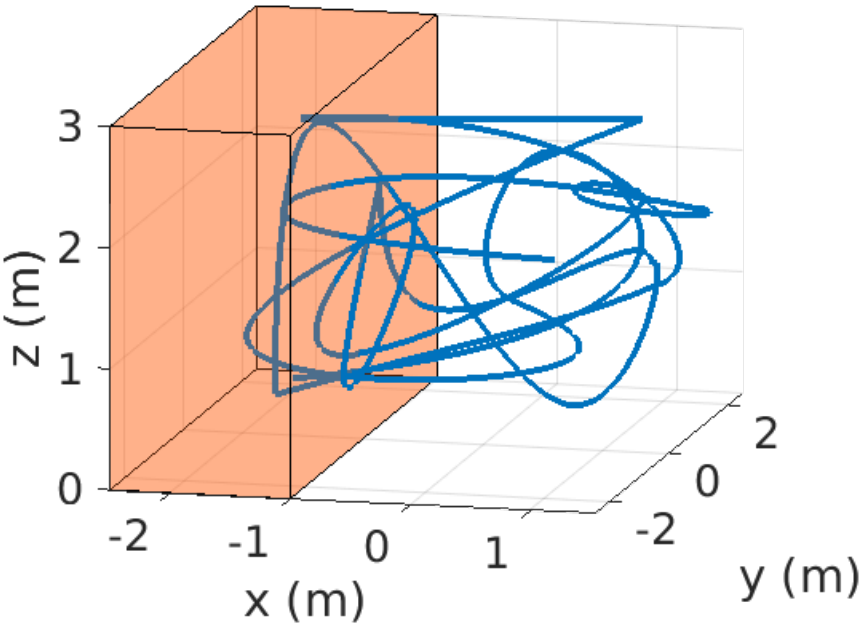


Figure 4.8: Representative trajectories entering and exiting the disturbance region (highlighted), taken from a 100 s window of the randomized trial.

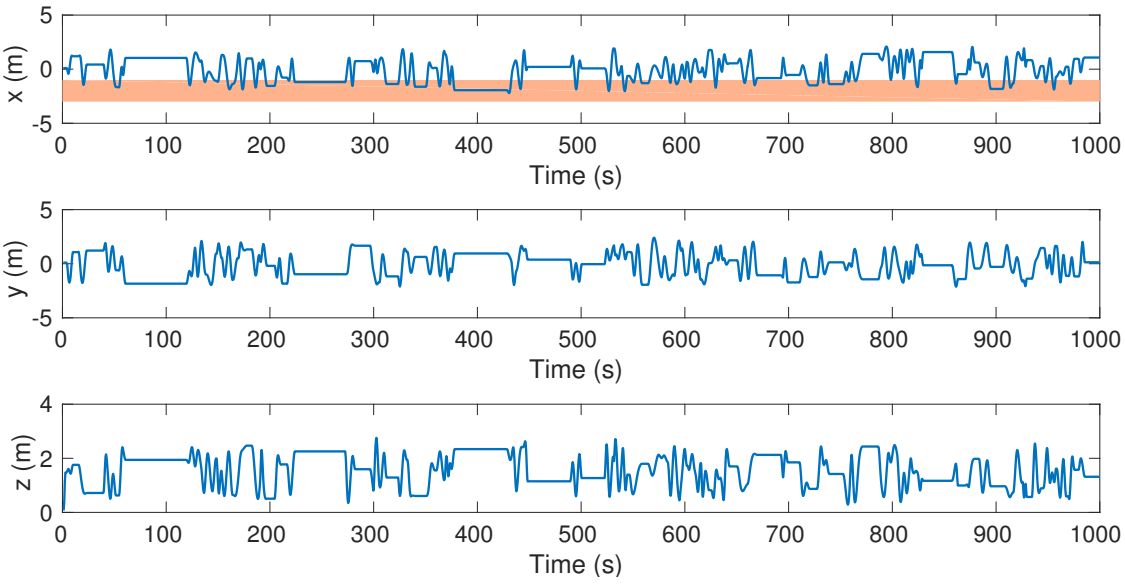


Figure 4.9: Reference trajectory components for the randomized trial with the disturbance region highlighted along the x-axis

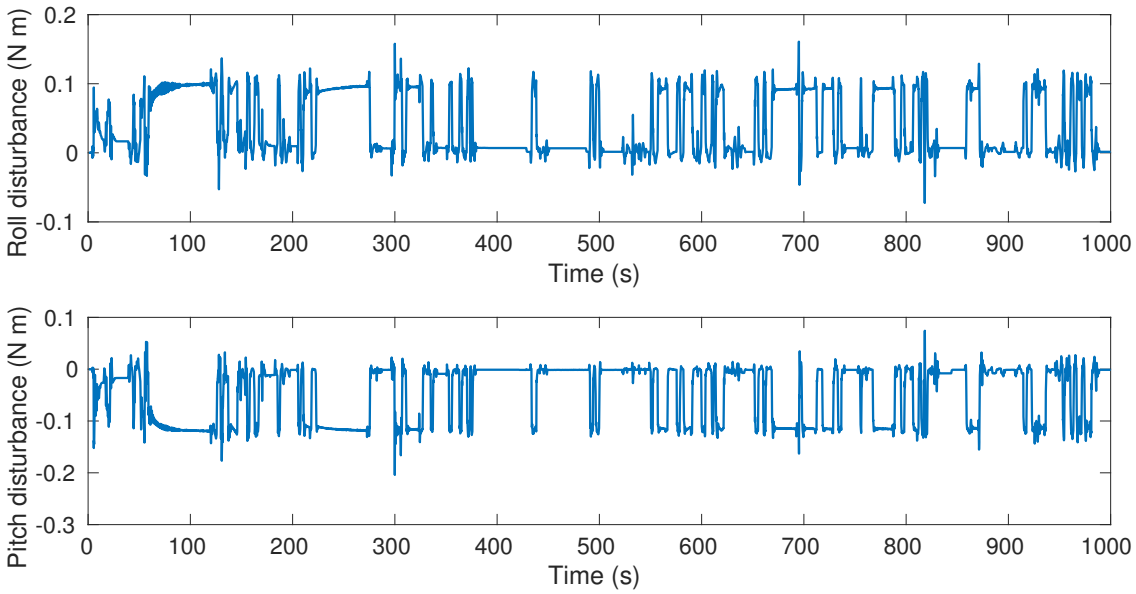


Figure 4.10: Roll and pitch disturbance estimates for the randomized trial show an initial transient but have consistent performance for the remainder of the trial

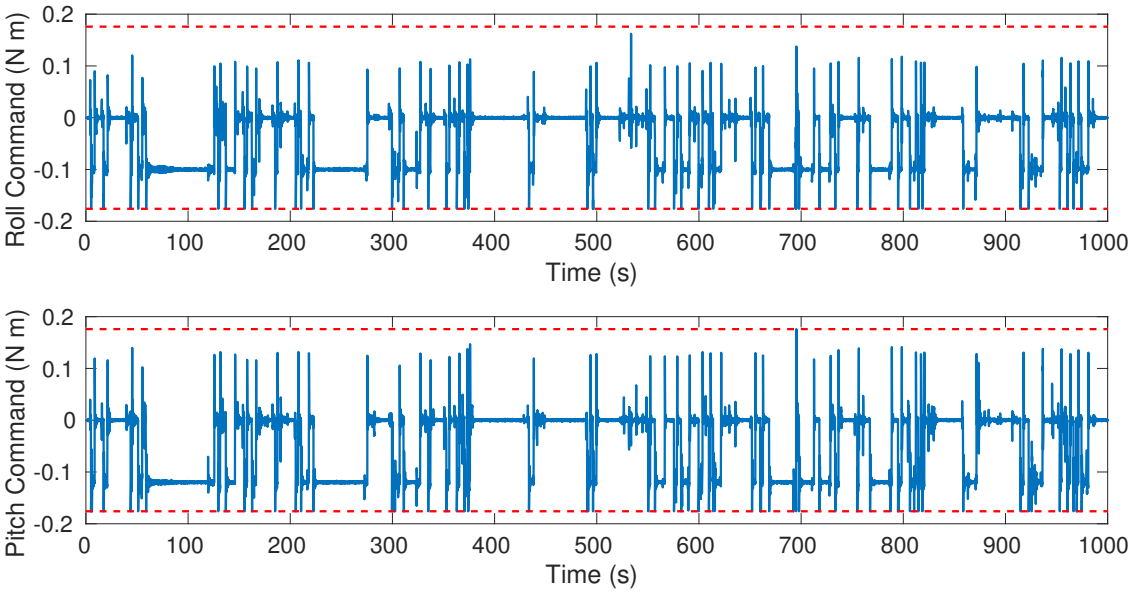


Figure 4.11: EPC satisfies control input constraints for the entire duration of the randomized trial while tracking a diverse set of trajectories

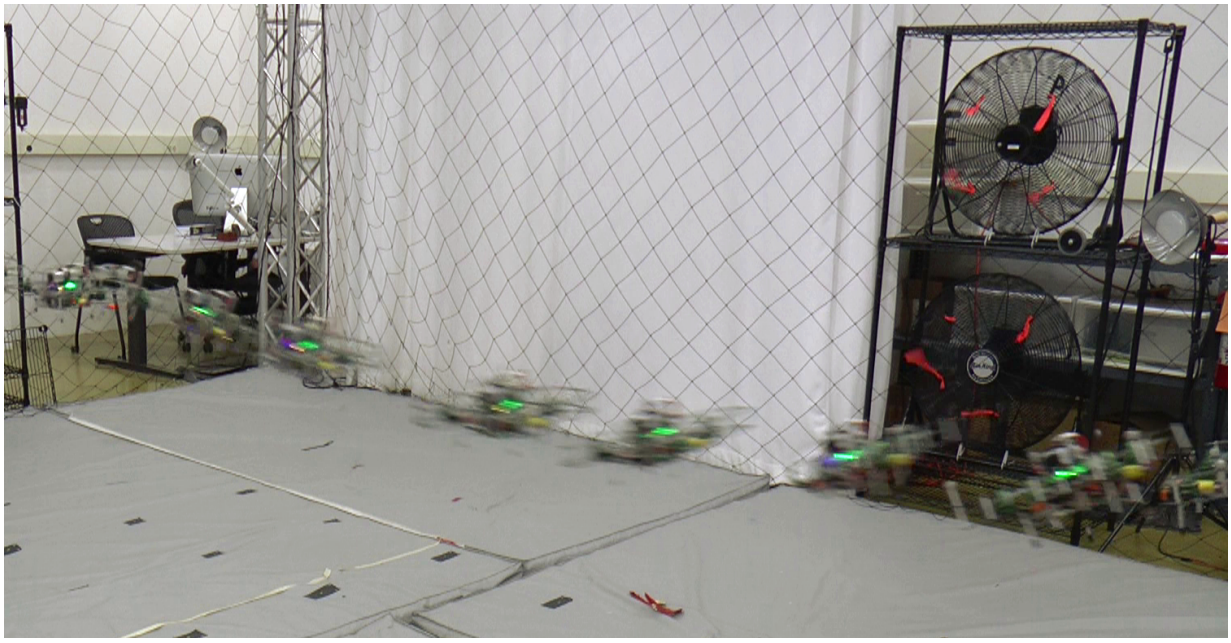


Figure 4.12: Snapshots of the line trajectory executed in a spatially-varying wind field generated via a pair of high-power fans

4.2.2 Experimental Validation

To validate the performance of EPC with realistic perturbations to the dynmics, we conduct a series of flight experiments with a small quadrotor platform (detailed in Appendix A.3.1). The quadrotor is commanded to fly 12 laps along the line trajectory shown in Fig. 4.12 with two fans generating a spatially-varying wind disturbance field. Each fan generates approximately a 6 m/s wind that decays minimally over the dimensions of the flight arena [5]. The quadrotor is commanded to track this trajectory with three instantiations of EPC as the position controller, one each with the Luenberger observer, LWPR, and ISSGPR as the disturbance adaptation strategy, as well as with $\mathcal{L}_1$ adaptive control. As Fig. 4.14 shows, applying an unconstrained controller ($\mathcal{L}_1$) yield repeated violations of the desired velocity limit. However, all three instances of EPC show reliable constraint satisfaction, with the LWPR version yielding only minor violations (**R2.1**).

To assess flight performance in the presence of the wind field, we first evaluate the model perturbations estimated by each of the four controller instances. As the wind acts along the negative x-axis, Fig. 4.15 shows the resulting disturbance acceleration estimates over the course of

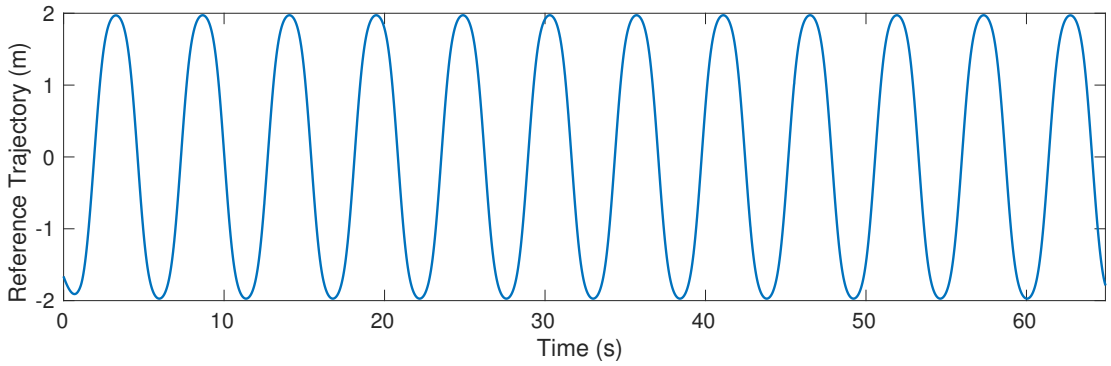
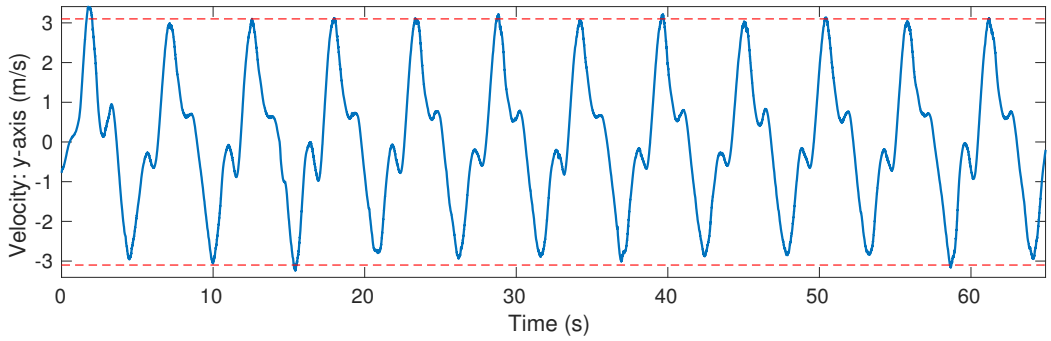


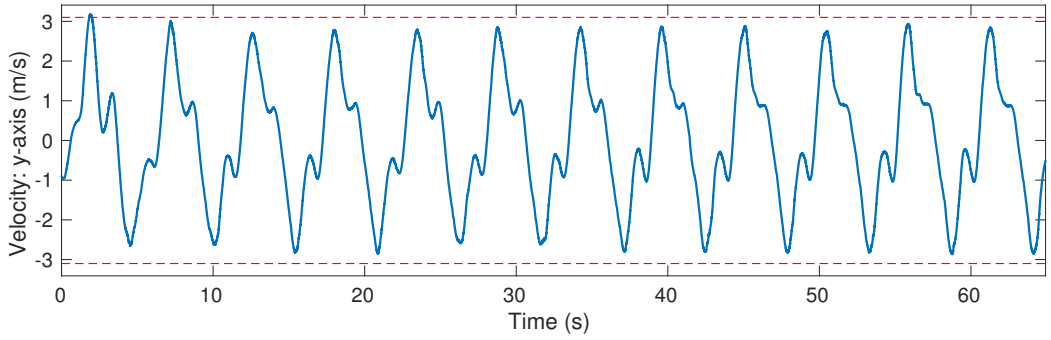
Figure 4.13: Reference trajectory along the y-axis for the 12-lap line flight experiments

the 12 laps. As both $\mathcal{L}_1$ and the first EPC instance use the same Luenberger observer, it is unsurprising to see consistent disturbance estimates from both trials. Additionally, these estimates are consistent across laps. However, with the experience-based model learning methodologies, there is a substantial temporal component to the estimate that is most noticeable with ISSGPR. This evolution of the estimate across laps shows that the model learner is initially accumulating experience on the effects of the wind field, but by the fourth lap, it has sufficient experience to maintain a consistent estimate of the wind disturbance acceleration.

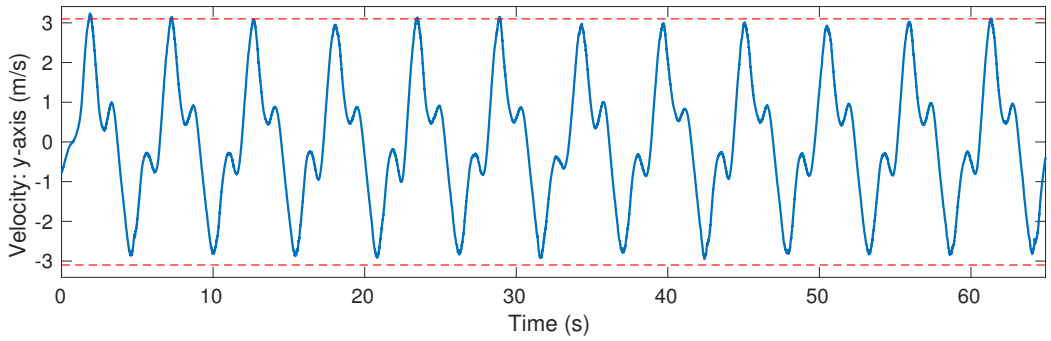
However, the merits of the experience-based model learning strategies is evident from a comparison of tracking performance with all four controllers. As the trajectory is constrained to the y-axis and the wind acts orthogonally to it, we use the cross-track error (i.e., deviations from the trajectory along the x-axis) to evaluate tracking performance due to adaptation. As Fig. 4.16 illustrates, both $\mathcal{L}_1$ and EPC with the Luenberger observer yield comparable performance that is consistent with their comparable disturbance estimates. However, these trials also exhibit a non-zero steady-state mean in the cross-track error. This likely stems from the reactive nature of the observer having insufficient time to fully mitigate the effect of the wind. In contrast, the LWPR- and ISSGPR-based EPC instantiations show nearly zero-mean cross-track error (Table 4.1). ISSGPR also yields a reduced variance that is consistent with empirical observations of smoother and less oscillatory flight. The last two columns in Table 4.1 also show the cross-track error for LWPR and ISSGPR evaluated over the last six laps, i.e., after gaining experience. This again



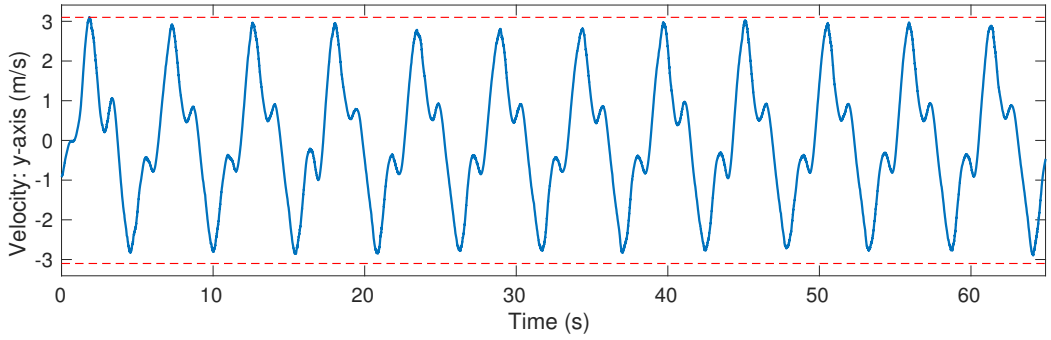
(a) $\mathcal{L}_1$ Adaptive Control



(b) EPC with Luenberger Observer (N = 25)



(c) EPC with LWPR (N = 25)



(d) EPC with ISSGPR (N = 25)

Figure 4.14: Comparison of y-velocity profiles for the experimental platform tracking the line trajectory using EPC with different disturbance estimation strategies and $\mathcal{L}_1$ adaptive control. All three EPC instances follow the velocity constraints (dashed lines).

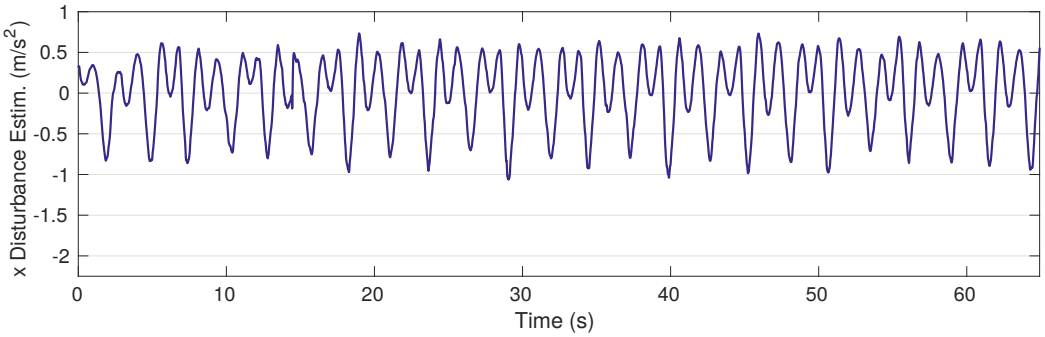
demonstrates that tracking performance improves as the system accumulates experience (**R4.1**).

Table 4.1: Cross-track error statistics for the high-wind line trajectory. The last two columns provide statistics for the experience-based approaches taken over the second half of the trial.

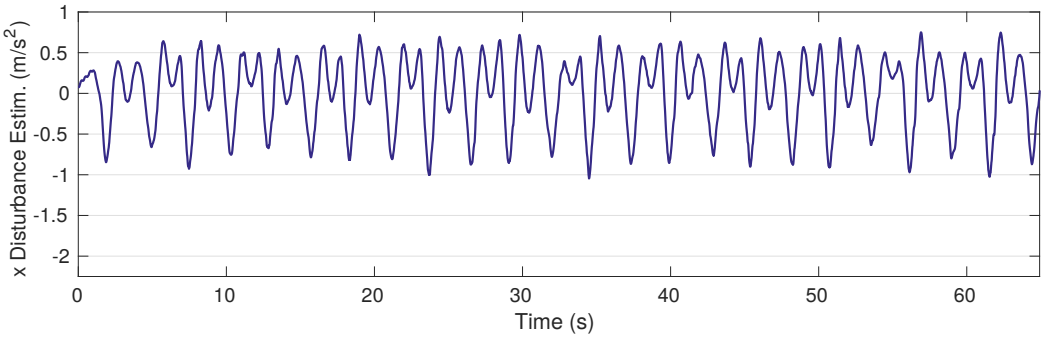
	$\mathcal{L}_1$ Adaptive Control	EPC with Luenberger	EPC with LWPR	EPC with ISSGPR	EPC+LWPR (2nd half)	EPC+ISSGPR (2nd half)
Mean (m)	0.0422	0.0334	**-0.0019**	**0.0032**	**-0.0016**	**0.0001**
Std. Dev. (m)	0.0518	0.0527	0.0623	0.0514	0.0587	0.0494

4.3 Conclusions

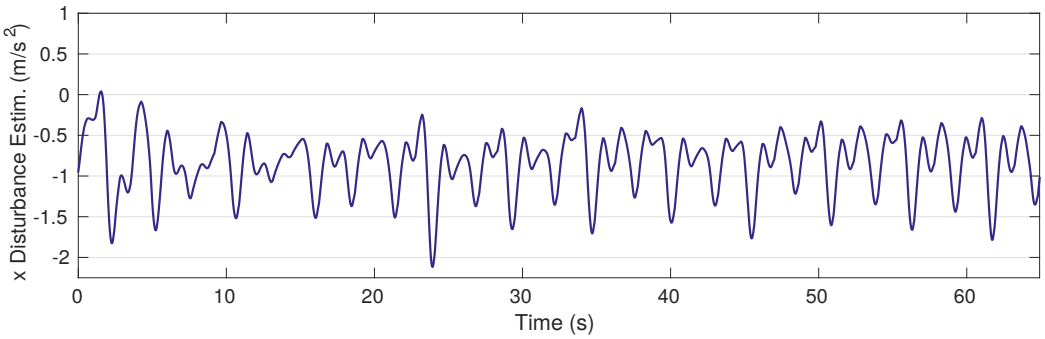
In this chapter, we have presented the Experience-driven Predictive Control (EPC) algorithm for fast, adaptive, nonlinear model predictive control. EPC constructs a database of reusable feedback controllers that are parameterized by the system dynamics. When combined with an online-learned model of the system dynamics based on Locally-Weighted Projection Regression (LWPR) or Incremental Sparse Spectrum Gaussian Process Regression (ISSGPR), this enables online adaption to perturbations to the dynamics model. As the system gains experience through operation, both the controller database and the dynamics model are improved to yield increased tracking accuracy, even in the presence of sudden changes to the dynamics model. This also implies that if the system were to start with some experience (e.g., from past operation), it could further reduce the transient effects of learning.



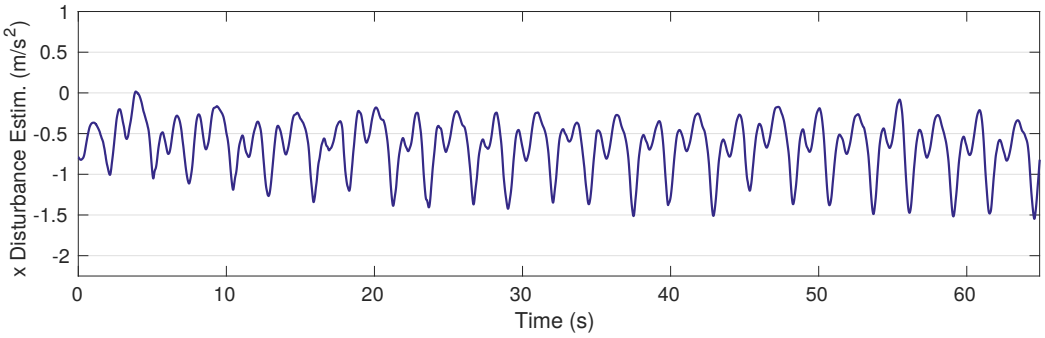
(a) $\mathcal{L}_1$ Adaptive Control



(b) EPC with Luenberger Observer (N = 25)

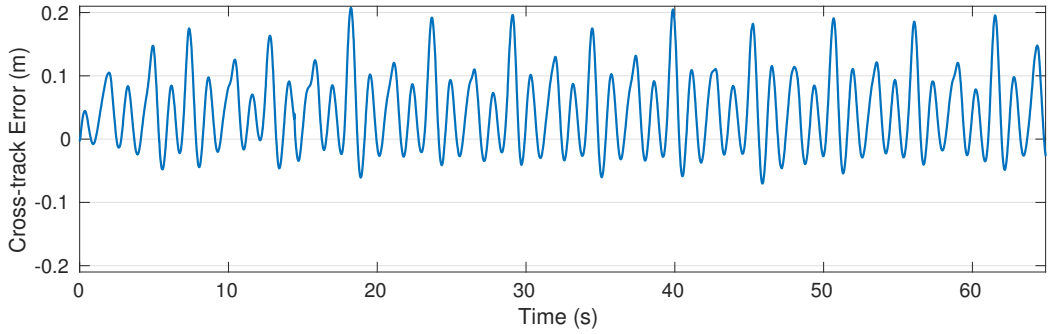


(c) EPC with LWPR (N = 25)

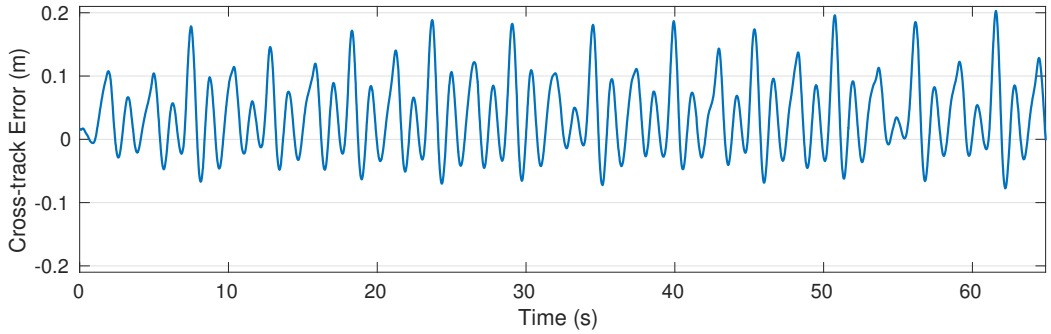


(d) EPC with ISSGPR (N = 25)

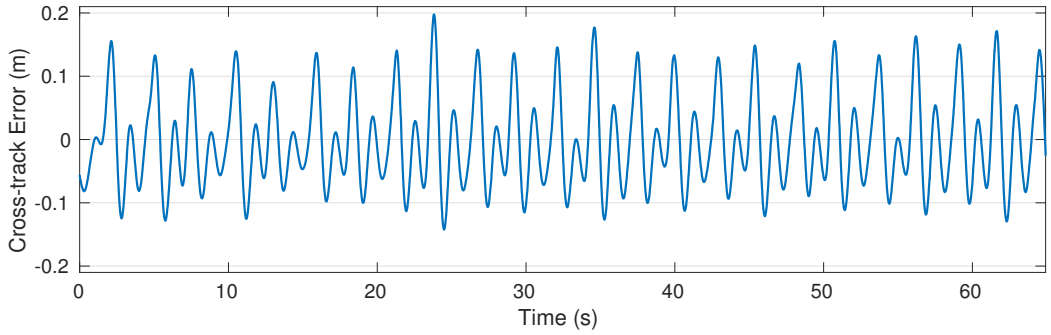
Figure 4.15: Comparison of the x-axis acceleration disturbance estimated by each controller's model adaptation component. ISSGPR (and LWPR to some extent) shows a clear trend in the estimates that stabilizes after acquiring sufficient experience.



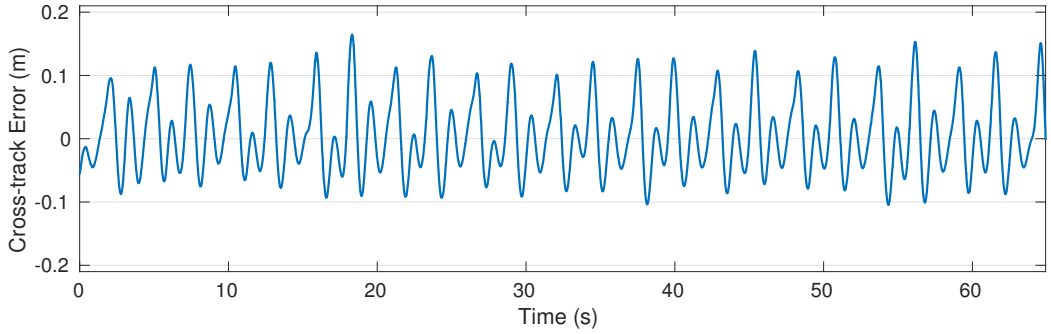
(a) $\mathcal{L}_1$ Adaptive Control



(b) EPC with Luenberger Observer (N = 25)



(c) EPC with LWPR (N = 25)



(d) EPC with ISSGPR (N = 25)

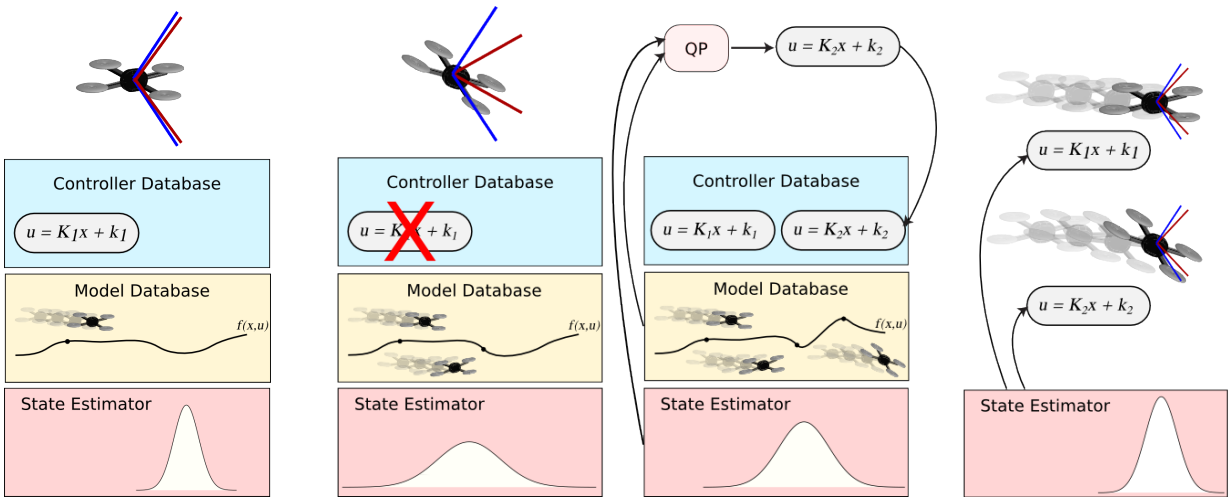
Figure 4.16: Comparison of cross-track error induced by the wind disturbance acting orthogonally to the line trajectory. Both LWPR and ISSGPR are able to mitigate the mean error, unlike the Luenberger observer-based configurations.

Chapter 5

Robust EPC

As shown in the previous chapter, the Experience-driven Predictive Control (EPC) algorithm enables high-rate, constrained Model Predictive Control (MPC) for uncertain, nonlinear systems. However, the model adaptation strategies EPC employs are limited in bandwidth and, as a result, are only able to mitigate the effects of low frequency disturbances (relative to the timescale of the dynamics). The introduction of high frequency sources of uncertainty, often via imperfect state estimation, can therefore lead to constraint violations even in the presence of online model adaptation.

In this chapter we aim to mitigate the effects of imperfect state estimation by leveraging uncertainty information from the state estimator to ensure constraints on the system state and control inputs are satisfied, even in the presence of time-varying state uncertainty. We propose a constrained, predictive control strategy that leverages EPC for computational efficiency and adaptation to low-frequency components of the uncertainty. We extend the underlying control problem to a chance-constrained Tube MPC formulation to capture the effects of time-varying state uncertainty (e.g., due to sensors with environment-dependent performance) in the robustness bounds. The resulting Robust EPC algorithm (illustrated in Fig. 5.1) ensures probabilistic constraint satisfaction in the presence of state uncertainty modeled by a multivariate Gaussian distribution provided by a Kalman filter based state estimator.



(a) (b) (c)

Figure 5.1: Overview of the proposed approach that combines an online learned controller database with estimates of the dynamics model and state uncertainty. As uncertainty changes, the tightened constraints (red) on the MAV automatically adjust to ensure robust satisfaction of the requested constraints (blue), even as the MAV switches between controllers. Panel (b) shows the addition of a new controller to the experience database to accommodate higher sensor uncertainty. In panel (c), the state uncertainty parameterizes all controllers in the database.

5.1 Approach

In this section, we present an extension of the Experience-driven Predictive Control (EPC) algorithm [84] to achieve high-rate predictive control with robust constraint satisfaction. EPC constructs online a two-part experience database consisting of previously used locally optimal controllers and observed perturbations to the system's dynamics model (illustrated by the blue and yellow boxes in Fig. 5.1). As the controllers are parameterized by the dynamics model, they automatically adapt to changes in the model. We therefore propose the Robust EPC algorithm by similarly parameterizing the controllers in the database by an online updated estimate of the uncertainty in the system state. This estimate is derived from the state estimator covariance and enables the use of a belief propagation approach to construct an uncertainty tube for the evolution of the state over the prediction horizon.

5.1.1 Adaptive Stochastic Dynamics Model

We consider the general nonlinear dynamics and observation models

$$\begin{aligned} \mathbf{x}_{k+1} &= f(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{w}_k \\ \mathbf{z}_k &= h(\mathbf{x}_k) + \mathbf{v}_k \end{aligned} \tag{5.1}$$

where $\mathbf{x}_k \in \mathbb{R}^n$ is the system state, $\mathbf{u}_k \in \mathbb{R}^m$ is the control input, and $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{W}_k)$ and $\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{V}_k)$ denote the process and measurement uncertainty, respectively. The corresponding first order approximations about a nominal state $\mathbf{x}^*$ and nominal control $\mathbf{u}^*$ are

$$\begin{aligned} \mathbf{x}_{k+1} &\approx \mathbf{A}_k(\mathbf{x}_k - \mathbf{x}^*) + \mathbf{B}_k(\mathbf{u}_k - \mathbf{u}^*) + \tilde{\mathbf{c}} + \mathbf{w}_k \\ \mathbf{z}_k &\approx \mathbf{C}_k(\mathbf{x}_k - \mathbf{x}^*) + \mathbf{v}_k \end{aligned} \tag{5.2}$$

where $\tilde{\mathbf{c}}$ combines the constant term in the Taylor series approximation and the estimate of the model error, $\hat{\mathbf{p}}$, returned by the model learner (Sect. 4.1.1). Updating this estimate via online observations also captures the effects of unmodeled dynamics, thus enabling adaptation to external perturbations (detailed in Sect. 5.1.4).

To model the evolution of this uncertain system, we extend (5.1) to a standard EKF belief state update law that estimates the state mean, $\boldsymbol{\mu}_k$, and covariance, $\boldsymbol{\Sigma}_k$,

$$\begin{aligned} \boldsymbol{\mu}_{k+1} &= f(\boldsymbol{\mu}_k, \mathbf{u}_k) + \mathbf{P}_k \mathbf{C}_k^{\mathrm{T}} \mathbf{L}_k^{-1} \left(\mathbf{z}_{k+1} - g(\boldsymbol{\mu}_k)\right) \\ \boldsymbol{\Sigma}_{k+1} &= \mathbf{P}_k - \mathbf{P}_k \mathbf{C}_k^{\mathrm{T}} \mathbf{L}_k^{-1} \mathbf{C}_k \mathbf{P}_k \end{aligned}$$

where $\mathbf{P}_k = \mathbf{A}_k \boldsymbol{\Sigma}_k \mathbf{A}_k^{\mathrm{T}} + \mathbf{W}_k$ and $\mathbf{L}_k = \mathbf{C}_k \mathbf{P}_k \mathbf{C}_k^{\mathrm{T}} + \mathbf{V}_k$. Following Platt et al. [85], we take $\mathbf{z}_{k+1} = h(\boldsymbol{\mu}_k)$ as the maximum likelihood observation to obtain a simplified belief state update

law

$$\begin{aligned} \boldsymbol{\mu}_{k+1} &= f(\boldsymbol{\mu}_k, \mathbf{u}_k) \\ \boldsymbol{\Sigma}_{k+1} &= \mathbf{P}_k - \mathbf{P}_k \mathbf{C}_k^{\mathrm{T}} \mathbf{L}_k^{-1} \mathbf{C}_k \mathbf{P}_k \end{aligned} \tag{5.3}$$

5.1.2 Chance-constrained Tube MPC

To incorporate this uncertainty propagation model into a robust control framework, we propose a Tube MPC formulation where the control applied to the system, $\mathbf{u}_k^S$, is the combination of the MPC output, $\mathbf{u}_k$, and an ancillary stabilizing controller with gain matrix $\mathbf{S}_k$,

$$\mathbf{u}_k^S = \mathbf{u}_k + \mathbf{S}_k(\mathbf{x}_k - \boldsymbol{\mu}_k) \tag{5.4}$$

This gain, $\mathbf{S}_k$, is designed to stabilize the nominal system via an unconstrained MPC formulation [86] given in Sect. 5.1.3. The introduction of the ancillary controller restricts deviations from the predicted state mean [78] and enables the MPC formulation to account for the reduction in uncertainty due to local feedback. This results in a slight change in the belief state update law,

$$\mathbf{P}_k = (\mathbf{A}_k - \mathbf{B}_k \mathbf{S}_k) \boldsymbol{\Sigma}_k (\mathbf{A}_k - \mathbf{B}_k \mathbf{S}_k)^{\mathrm{T}} + \mathbf{W}_k$$

The Tube MPC formulation also enforces state and input constraints, $\mathbf{x}_k \in \mathcal{X}_k, \mathbf{u}_k^S \in \mathcal{U}_k$. In this work, we assume the admissible state sets, $\mathcal{X}_k$, and input sets, $\mathcal{U}_k$, are polytopic, or can be approximated by polytopes. This yields a set of half-plane constraints,

$$\begin{aligned} \mathbf{G}_{\mathbf{x}_{k+1}}(\mathbf{x}_{k+1} - \mathbf{x}^*) &\leq \mathbf{g}_{\mathbf{x}_{k+1}} \\ \mathbf{G}_{\mathbf{u}_k}(\mathbf{u}_k^S - \mathbf{u}^*) &\leq \mathbf{g}_{\mathbf{u}_k} \end{aligned} \tag{5.5}$$

However, due to the stochastic dynamics model, we instead employ a chance constrained formu-

lation by requiring (5.5) to hold with probability $1-\alpha$,

$$\begin{aligned} P\left(\mathbf{G}_{\mathbf{x}_{k+1}}(\mathbf{x}_{k+1}-\mathbf{x}^*) \leq \mathbf{g}_{\mathbf{x}_{k+1}}\right) &\geq 1-\alpha \\ P\left(\mathbf{G}_{\mathbf{u}_k}(\mathbf{u}_k^S-\mathbf{u}^*) \leq \mathbf{g}_{\mathbf{u}_k}\right) &\geq 1-\alpha \end{aligned} \tag{5.6}$$

Given that the belief state corresponds to a multivariate Gaussian, $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, its probability mass level sets are ellipsoids defined by a χ^2 value. The ellipsoid containing $1-\alpha$ of the probability mass is given by $(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}) = \chi_n^2(\alpha)$. Therefore, a given chance constraint threshold, $1-\alpha$, yields an ellipsoid defining the state uncertainty bounds.

Ensuring robust constraint satisfaction requires tightening (5.5) by these bounds [78], as illustrated in Fig. 5.2. Consequently, to retain the linear form of the constraints, we follow Domes et al. [87] to approximate the ellipsoid by its axis-aligned bounding box with side lengths given by

$$\boldsymbol{\delta}_{k+1}^{\mathbf{x}} = \sqrt{\chi_n^2(\alpha)\mathrm{diag}(\boldsymbol{\Sigma}_{k+1})} \tag{5.7}$$

where $\mathrm{diag}(\cdot)$ returns the diagonal elements of the argument as a vector.

While the MPC output, $\mathbf{u}_k$, does not introduce any control input uncertainty, the ancillary controller is a function of the uncertain future state. This yields a similar bound on the control command,

$$\boldsymbol{\delta}_{k}^{\mathbf{u}} = \sqrt{\chi_n^2(\alpha)\mathrm{diag}(\mathbf{S}_k\boldsymbol{\Sigma}_k\mathbf{S}_k^{\mathrm{T}})} \tag{5.8}$$

Given these bounding box dimensions, we convert the probabilistic state and input constraints (5.6) to tightened deterministic constraints, $\mathbf{x}_k \in \tilde{\mathcal{X}}_k, \mathbf{u}_k \in \tilde{\mathcal{U}}_k$,

$$\begin{aligned} \mathbf{G}_{\mathbf{x}_{k+1}}(\boldsymbol{\mu}_{k+1}-\mathbf{x}^*) &\leq \mathbf{g}_{\mathbf{x}_{k+1}} - \mathbf{G}_{\mathbf{x}_{k+1}}\boldsymbol{\delta}_{k+1}^{\mathbf{x}} = \tilde{\mathbf{g}}_{\mathbf{x}_{k+1}} \\ \mathbf{G}_{\mathbf{u}_k}(\mathbf{u}_k-\mathbf{u}^*) &\leq \mathbf{g}_{\mathbf{u}_k} - \mathbf{G}_{\mathbf{u}_k}\boldsymbol{\delta}_{k}^{\mathbf{u}} = \tilde{\mathbf{g}}_{\mathbf{u}_k} \end{aligned} \tag{5.9}$$

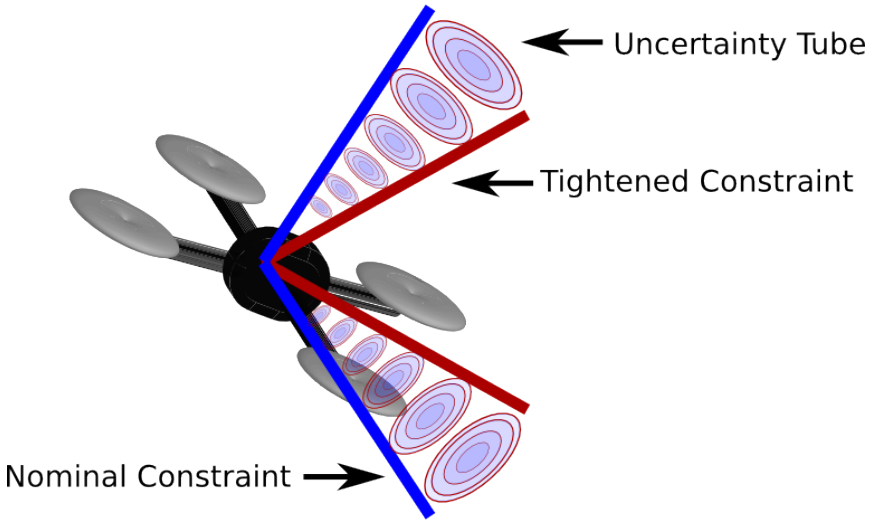


Figure 5.2: Nominal state constraints (blue lines) are tightened (red lines) according to a chance-constraint bound on the predicted Gaussian uncertainty.

Although the bounding box is generally a conservative approximation of the ellipsoid, we observe that for any axis-aligned box constraint, tightening by the bounding box is equivalent to the exact approach of tightening by the axis-aligned suprema over the ellipsoid [88].

5.1.3 Robust EPC formulation

Although this chance-constrained Tube MPC formulation permits an optimization-based solution, in this work, we propose a novel extension to the Experience-driven Predictive Control (EPC) algorithm [84] to enable Robust MPC on computationally constrained systems. The proposed Robust EPC algorithm leverages this tube-based formulation to enforce robust constraint satisfaction while retaining the computational efficiency and model adaptation properties of EPC.

As in EPC, we can formulate the receding-horizon control problem as a quadratic program (QP) due to the model adaptation term, $\tilde{\mathbf{c}}$, that captures the nonlinearities and other unmodeled dynamics. The QP is formulated about a nominal state, $\mathbf{x}^*$ and input, $\mathbf{u}^*$, to track a sequence of

N reference states $\mathbf{r}_1, \ldots, \mathbf{r}_N$,

$$\begin{aligned} \underset{\bar{\mathbf{u}}_k}{\operatorname{argmin}} \sum_{k=0}^{N-1} \frac{1}{2}(\bar{\mathbf{x}}_{k+1} - \bar{\mathbf{r}}_{k+1})^{\mathrm{T}} \mathbf{Q}_{k+1}(\bar{\mathbf{x}}_{k+1} - \bar{\mathbf{r}}_{k+1}) + \frac{1}{2}(\bar{\mathbf{u}}_k - \bar{\mathbf{u}}_{\hat{\mathbf{p}}})^{\mathrm{T}} \mathbf{R}_k(\bar{\mathbf{u}}_k - \bar{\mathbf{u}}_{\hat{\mathbf{p}}}) \\ \text{s.t.} \quad \bar{\mathbf{x}}_{k+1} = \mathbf{A}\bar{\mathbf{x}}_k + \mathbf{B}\bar{\mathbf{u}}_k + \tilde{\mathbf{c}} \\ \mathbf{G}_{\mathbf{x}_{k+1}} \bar{\mathbf{x}}_{k+1} \leq \tilde{\mathbf{g}}_{\mathbf{x}_{k+1}} \\ \mathbf{G}_{\mathbf{u}_k} \bar{\mathbf{u}}_k \leq \tilde{\mathbf{g}}_{\mathbf{u}_k} \\ \forall\, k = 0, \ldots, N-1 \end{aligned} \tag{5.10}$$

where $\bar{\mathbf{x}}_k = \boldsymbol{\mu}_k - \mathbf{x}^*$, $\bar{\mathbf{r}}_k = \mathbf{r}_k - \mathbf{x}^*$, $\bar{\mathbf{u}}_k = \mathbf{u}_k - \mathbf{u}^*$. If it is possible to derive a control input, $\bar{\mathbf{u}}_{\hat{\mathbf{p}}}$, from the model adaptation term (e.g., if $\hat{\mathbf{p}}$ is an acceleration disturbance, $\bar{\mathbf{u}}_{\hat{\mathbf{p}}}$ would be the corresponding force) we subtract it in the cost function to avoid penalizing model error compensation [84].

Given that we can forward predict the mean and covariance evolution via (5.3), we can simplify notation by defining $\boldsymbol{x} = \begin{bmatrix} \bar{\mathbf{x}}_1^{\mathrm{T}}, \ldots, \bar{\mathbf{x}}_N^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$, $\boldsymbol{r} = \begin{bmatrix} \bar{\mathbf{r}}_1^{\mathrm{T}}, \ldots, \bar{\mathbf{r}}_N^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$, $\boldsymbol{u} = \begin{bmatrix} \bar{\mathbf{u}}_0^{\mathrm{T}}, \ldots, \bar{\mathbf{u}}_{N-1}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$, $\boldsymbol{u}_{\hat{\mathbf{p}}} = \begin{bmatrix} \bar{\mathbf{u}}_{\hat{\mathbf{p}}}^{\mathrm{T}}, \ldots, \bar{\mathbf{u}}_{\hat{\mathbf{p}}}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$,

$$\boldsymbol{\mathcal{B}} = \begin{bmatrix} \mathbf{B} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{AB} & \mathbf{B} & \ldots & \mathbf{0} \\ \vdots & \vdots & \ddots & \\ \mathbf{A}^{N-1}\mathbf{B} & \mathbf{A}^{N-2}\mathbf{B} & \ldots & \mathbf{B} \end{bmatrix}, \; \boldsymbol{c} = \begin{bmatrix} \tilde{\mathbf{c}} \\ (\mathbf{A} + \mathbf{I})\,\tilde{\mathbf{c}} \\ \vdots \\ \sum_{i=0}^{N-1} \mathbf{A}^i \tilde{\mathbf{c}} \end{bmatrix},$$

$\boldsymbol{\mathcal{Q}} = \operatorname{diag}(\mathbf{Q}_1, \ldots, \mathbf{Q}_N)$, $\boldsymbol{\mathcal{R}} = \operatorname{diag}(\mathbf{R}_0, \ldots, \mathbf{R}_{N-1})$, $\boldsymbol{\mathcal{G}}_{\mathbf{x}} = \operatorname{diag}(\mathbf{G}_{\mathbf{x}_1}, \ldots, \mathbf{G}_{\mathbf{x}_N})$, and $\boldsymbol{\mathcal{G}}_{\mathbf{u}} = \operatorname{diag}(\mathbf{G}_{\mathbf{u}_0}, \ldots, \mathbf{G}_{\mathbf{u}_{N-1}})$, where $\operatorname{diag}(\cdot)$ here diagonally concatenates matrices. Similarly, let $\boldsymbol{g}_{\mathbf{x}} = \begin{bmatrix} \tilde{\mathbf{g}}_{\mathbf{x}_1}^{\mathrm{T}}, \ldots, \tilde{\mathbf{g}}_{\mathbf{x}_N}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$ and $\boldsymbol{g}_{\mathbf{u}} = \begin{bmatrix} \tilde{\mathbf{g}}_{\mathbf{u}_0}^{\mathrm{T}}, \ldots, \tilde{\mathbf{g}}_{\mathbf{u}_{N-1}}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$ to capture the tightened constraints (5.9).

Finally, we define $\boldsymbol{\mu}_0$ to be a parameter of the optimization constrained by the current state (see Sect. 5.1.3) rather than directly using the current state as in EPC. Therefore, the nominal

state, $\mathbf{x}^* = \boldsymbol{\mu}_0$, $\bar{\mathbf{x}}_0 = \mathbf{0}$, and (5.10) simplifies to

$$\begin{aligned} \underset{\boldsymbol{u}}{\operatorname{argmin}} \frac{1}{2}(\boldsymbol{x}-\boldsymbol{r})^{\mathrm{T}}\boldsymbol{\mathcal{Q}}(\boldsymbol{x}-\boldsymbol{r}) + \frac{1}{2}(\boldsymbol{u}-\boldsymbol{u}_{\hat{\mathbf{p}}})^{\mathrm{T}}\boldsymbol{\mathcal{R}}(\boldsymbol{u}-\boldsymbol{u}_{\hat{\mathbf{p}}}) \\ \text{s.t.} \quad \boldsymbol{x} = \boldsymbol{\mathcal{B}}\boldsymbol{u} + \boldsymbol{c} \\ \boldsymbol{\mathcal{G}}_{\mathbf{x}}\boldsymbol{x} \leq \boldsymbol{g}_{\mathbf{x}} \\ \boldsymbol{\mathcal{G}}_{\mathbf{u}}\boldsymbol{u} \leq \boldsymbol{g}_{\mathbf{u}} \end{aligned}$$

Incorporating the dynamics into the cost and constraints yields an equivalent QP that facilitates the state space partitioning and local controller computation steps of EPC,

$$\begin{aligned} \underset{\boldsymbol{u}}{\operatorname{argmin}} \frac{1}{2}\boldsymbol{u}^{\mathrm{T}}\boldsymbol{\mathcal{H}}\boldsymbol{u} + \boldsymbol{h}^{\mathrm{T}}\boldsymbol{u} \\ \text{s.t.} \quad \boldsymbol{\Gamma}\boldsymbol{u} \leq \boldsymbol{\gamma} \end{aligned} \tag{5.11}$$

where $\boldsymbol{\mathcal{H}} = \boldsymbol{\mathcal{B}}^{\mathrm{T}}\boldsymbol{\mathcal{Q}}\boldsymbol{\mathcal{B}} + \boldsymbol{\mathcal{R}}$, $\boldsymbol{h} = \boldsymbol{\mathcal{B}}^{\mathrm{T}}\boldsymbol{\mathcal{Q}}(\boldsymbol{c}-\boldsymbol{r}) - \boldsymbol{\mathcal{R}}\boldsymbol{u}_{\hat{\mathbf{p}}}$,

$$\boldsymbol{\Gamma} = \begin{bmatrix} \boldsymbol{\mathcal{G}}_{\mathbf{x}}\boldsymbol{\mathcal{B}} \\ \boldsymbol{\mathcal{G}}_{\mathbf{u}} \end{bmatrix}, \text{ and } \boldsymbol{\gamma} = \begin{bmatrix} \boldsymbol{g}_{\mathbf{x}} - \boldsymbol{\mathcal{G}}_{\mathbf{x}}\boldsymbol{c} \\ \boldsymbol{g}_{\mathbf{u}} \end{bmatrix}$$

As in EPC, the partitioning of the state-space for Robust EPC is determined by the Karush-Kuhn-Tucker (KKT) conditions for optimality,

$$\begin{aligned} \boldsymbol{\mathcal{H}}\boldsymbol{u} + \boldsymbol{h} + \Gamma^{\mathrm{T}}\boldsymbol{\lambda} = \mathbf{0} \\ \boldsymbol{\Lambda}(\boldsymbol{\Gamma}\boldsymbol{u} - \boldsymbol{\gamma}) = \mathbf{0} \end{aligned} \tag{5.12}$$

where $\boldsymbol{\lambda}$ is the vector of Lagrange multipliers and $\boldsymbol{\Lambda} = \operatorname{diag}(\boldsymbol{\lambda})$. Therefore, given a set of active constraints (i.e., with $\lambda > 0$), we can solve for the optimal control sequence $\boldsymbol{u}$ and corresponding

Lagrange multipliers by solving a linear system derived from (5.12),

$$\begin{bmatrix} \boldsymbol{\mathcal{H}} & \boldsymbol{\Gamma}_a^{\mathrm{T}} \\ \boldsymbol{\Gamma}_a & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{\lambda}_a \end{bmatrix} = \begin{bmatrix} -\boldsymbol{h} \\ \boldsymbol{\gamma}_a \end{bmatrix}$$

where the subscript a denotes rows corresponding to active constraints.

For any linearly independent set of active constraints [47], the resulting $\boldsymbol{u}$ is affine in the predicted state mean error, $\boldsymbol{r}$,

$$\boldsymbol{u} = \boldsymbol{\mathcal{E}}_5 \boldsymbol{r} - \left(\boldsymbol{\mathcal{E}}_5 \boldsymbol{c} - \boldsymbol{\mathcal{E}}_4 \boldsymbol{\mathcal{R}} \boldsymbol{u}_{\hat{\mathbf{p}}} + \boldsymbol{\mathcal{E}}_3 \begin{bmatrix} \boldsymbol{g}_{\mathbf{x}}^{+} - \boldsymbol{\mathcal{G}}_{\mathbf{x}} \boldsymbol{c} \\ -\boldsymbol{g}_{\mathbf{x}}^{-} + \boldsymbol{\mathcal{G}}_{\mathbf{x}} \boldsymbol{c} \\ \boldsymbol{g}_{\mathbf{u}}^{+} \\ -\boldsymbol{g}_{\mathbf{u}}^{-} \end{bmatrix}_a \right) \tag{5.13}$$

where $\boldsymbol{\mathcal{E}}_1 = \boldsymbol{\Gamma}_a \boldsymbol{\mathcal{H}}^{-1}$, $\boldsymbol{\mathcal{E}}_2 = -(\boldsymbol{\mathcal{E}}_1 \boldsymbol{\Gamma}_a^{\mathrm{T}})^{-1}$, $\boldsymbol{\mathcal{E}}_3 = \boldsymbol{\mathcal{E}}_1^{\mathrm{T}} \boldsymbol{\mathcal{E}}_2$, $\boldsymbol{\mathcal{E}}_4 = \boldsymbol{\mathcal{H}}^{-1} + \boldsymbol{\mathcal{E}}_3 \boldsymbol{\mathcal{E}}_1$, and $\boldsymbol{\mathcal{E}}_5 = \boldsymbol{\mathcal{E}}_4 \boldsymbol{\mathcal{B}}^{\mathrm{T}} \boldsymbol{\mathcal{Q}}$. Moreover, the coefficients in (5.13) are all functions of $\mathbf{A}, \mathbf{B}, \tilde{\mathbf{c}}, \boldsymbol{\delta}^{\mathbf{x}}$, and $\boldsymbol{\delta}^{\mathbf{u}}$. Therefore, the final control law $\kappa(\mathbf{x}_0, \mathbf{r}_1, \ldots, \mathbf{r}_N)$ is given by a parameterized feedback gain matrix $\mathbf{K}$, a feedforward vector $\mathbf{k}_{\mathrm{ff}}$, and the ancillary control gain matrix, $\mathbf{S}$,

$$\begin{aligned} \kappa(\mathbf{x}_0, \mathbf{r}_1, \ldots, \mathbf{r}_N) &= \mathbf{K}(\mathbf{A}, \mathbf{B}, \tilde{\mathbf{c}}, \boldsymbol{\delta}^{\mathbf{x}}, \boldsymbol{\delta}^{\mathbf{u}}) \boldsymbol{r} \\ &\quad + \mathbf{k}_{\mathrm{ff}}(\mathbf{A}, \mathbf{B}, \tilde{\mathbf{c}}, \boldsymbol{\delta}^{\mathbf{x}}, \boldsymbol{\delta}^{\mathbf{u}}) \\ &\quad + \left[\mathbf{S}_0 (\mathbf{x}_0 - \boldsymbol{\mu}_0)^{\mathrm{T}}, \ldots, \mathbf{S}_{N-1} (\mathbf{x}_{N-1} - \boldsymbol{\mu}_{N-1})^{\mathrm{T}} \right]^{\mathrm{T}} \end{aligned} \tag{5.14}$$

The KKT condition matrices, which determine whether a previously computed controller is locally optimal, are similarly parameterized, and the active Lagrange multipliers, $\boldsymbol{\lambda}_a$, are given

by

$$\boldsymbol{\lambda}_a = -\boldsymbol{\mathcal{E}}_6 \boldsymbol{r} + \left(\boldsymbol{\mathcal{E}}_6 \boldsymbol{c} - \boldsymbol{\mathcal{E}}_3^{\mathrm{T}} \boldsymbol{\mathcal{R}} \boldsymbol{u}_{\hat{\mathbf{p}}} + \boldsymbol{\mathcal{E}}_2 \begin{bmatrix} \boldsymbol{g}_{\mathbf{x}}^{+} - \boldsymbol{\mathcal{G}}_{\mathbf{x}} \boldsymbol{c} \\ -\boldsymbol{g}_{\mathbf{x}}^{-} + \boldsymbol{\mathcal{G}}_{\mathbf{x}} \boldsymbol{c} \\ \boldsymbol{g}_{\mathbf{u}}^{+} \\ -\boldsymbol{g}_{\mathbf{u}}^{-} \end{bmatrix}_a \right) \tag{5.15}$$

where $\boldsymbol{\mathcal{E}}_6 = \boldsymbol{\mathcal{E}}_3^{\mathrm{T}} \boldsymbol{\mathcal{B}}^{\mathrm{T}} \boldsymbol{\mathcal{Q}}$. Therefore, given a set of active constraints, the corresponding controller and KKT matrices can be reconstructed online using (5.13), (5.15), and the current $\mathbf{A}, \mathbf{B}, \tilde{\mathbf{c}}, \boldsymbol{\delta}^{\mathbf{x}}$ and $\boldsymbol{\delta}^{\mathbf{u}}$. Therefore, each controller automatically evolves with both the estimated system dynamics and state uncertainty. This also enables the construction of a controller database that recovers the functionality of (5.10) by switching between controllers according to the KKT conditions, thus providing the foundation for the Robust EPC algorithm detailed in Sect. 5.1.5.

Ancillary Controller

In addition to the introduction of a chance-constrained formulation, the extension of EPC to Robust EPC requires two key components. The first is the ancillary controller, which aims to drive the uncertain state, $\mathbf{x}_k$, to the state mean sequence, $\boldsymbol{\mu}_k$, produced by (5.11). The corresponding unconstrained MPC formulation,

$$\underset{\mathbf{u}_k}{\operatorname{argmin}} \sum_{k=0}^{N-1} \frac{1}{2} (\mathbf{x}_{k+1} - \boldsymbol{\mu}_{k+1})^{\mathrm{T}} \mathbf{Q}_{k+1} (\mathbf{x}_{k+1} - \boldsymbol{\mu}_{k+1}) + \frac{1}{2} \mathbf{u}_k^{\mathrm{T}} \mathbf{R}_k \mathbf{u}_k$$

yields an equivalent set of feedback control gains computed analogously to (5.14) without constraints,

$$\operatorname{diag}(\mathbf{S}_0, \ldots, \mathbf{S}_{N-1}) = (\boldsymbol{\mathcal{B}}^{\mathrm{T}} \boldsymbol{\mathcal{Q}} \boldsymbol{\mathcal{B}} + \boldsymbol{\mathcal{R}})^{-1} \boldsymbol{\mathcal{B}}^{\mathrm{T}} \boldsymbol{\mathcal{Q}} \tag{5.16}$$

Initial State Selection

The second component is the initial state mean parameter, $\boldsymbol{\mu}_0$. Due to the uncertainty in the state, $\boldsymbol{\mu}_0$ is not necessarily set to the initial state, $\mathbf{x}_0$. Instead, the underlying tube-based formulation permits selecting $\boldsymbol{\mu}_0$ such that

$$\mathbf{x}_0 \in \boldsymbol{\mu}_0 \oplus Box(\boldsymbol{\delta}_0^x) \tag{5.17}$$

where $Box(\boldsymbol{\delta}_0^x)$ is the bounding box with dimensions given by $\boldsymbol{\delta}_0^x$ [89] and $\oplus$ denotes the Minkowski sum. We therefore propose a piecewise definition of $\boldsymbol{\mu}_0$,

$$\boldsymbol{\mu}_0 = \begin{cases} \mathbf{x}_0, & \mathbf{x}_0 \in \tilde{\mathcal{X}}_0 \\ \text{proj}_{\tilde{\mathcal{X}}}(\mathbf{x}_0), & \mathbf{x}_0 \in \mathcal{X}_0 \backslash \tilde{\mathcal{X}}_0 \end{cases} \tag{5.18}$$

where the $\text{proj}_{\tilde{\mathcal{X}}}(\cdot)$ operator projects the state onto the tightened constraint set, $\tilde{\mathcal{X}}$. If $\mathbf{x}_0 \in \tilde{\mathcal{X}}_0$, the initial state satisfies (5.17) and can be assigned to $\boldsymbol{\mu}_0$. Otherwise, we assume only the noisy state is outside $\tilde{\mathcal{X}}_0$ and use the projection operation to find the closest point in $\tilde{\mathcal{X}}_0$. Due to the chance-constrained formulation, infrequent constraint violations are possible. Therefore, if $\mathbf{x}_0 \notin \mathcal{X}_0$, an intermediate controller is applied as part of the Robust EPC algorithm detailed in Sect. 5.1.5 to recover from the constraint violation.

5.1.4 Online Model Adaptation

In addition to robust constraint satisfaction, the parameterized controllers (5.14) generated via Robust EPC retain the adaptation properties of EPC, thus providing a means to mitigate both high and low frequency components of uncertainty. We consider three online model adaptation strategies to assess their effects on robust constraint satisfaction. Locally Weighted Projection Regression (LWPR) and Incremental Sparse Spectrum Gaussian Process Regression (ISSGPR) constitute model adaptation techniques that leverage past experience to improve performance over time, while the Luenberger disturbance observer is a more traditional estimation strategy

Algorithm 5.1 Robust Experience-driven Predictive Control

$\mathcal{M} \leftarrow \emptyset$ or $\mathcal{M}_{\text{prior}}$
while control is enabled **do**
  $\mathbf{x}_0 \leftarrow$ current system state estimate mean
  $\mathbf{r}_1, \ldots, \mathbf{r}_N \leftarrow$ current reference sequence
  $\mathbf{A}, \mathbf{B}, \tilde{\mathbf{c}} \leftarrow$ current dynamics model via adaptation
  Compute $\mathbf{S}$ via (5.16) and $\boldsymbol{\delta}^{\mathbf{x}}, \boldsymbol{\delta}^{\mathbf{u}}$ via (5.7),(5.8)
  Select $\boldsymbol{\mu}_0$ via (5.18)
  for each element $m_i \in \mathcal{M}$ **do**
    Compute $\boldsymbol{u}, \boldsymbol{\lambda}$ via (5.13),(5.15)
    if $\boldsymbol{x}, \boldsymbol{r}$ satisfy parameterized KKT criteria **then**
      $\texttt{importance}_i \leftarrow$ current time, sort $\mathcal{M}$
      `solution_found` $\leftarrow$ true
      Apply control law (5.14) from m_i
    end if
  end for
  if `solution_found` is false **then**
    Apply interm. control via (5.11) with slack variables
    Update QP formulation with $(\mathbf{A}, \mathbf{B}, \tilde{\mathbf{c}}, \boldsymbol{\delta}^{\mathbf{x}}, \boldsymbol{\delta}^{\mathbf{u}})$
    Generate new controller via QP (5.11) (in parallel)
    if $|\mathcal{M}| =$ maximum table size **then**
      Remove element with min. `importance`
    end if
    Add $m_{\text{new}} = (\mathbf{K}, \mathbf{k}_{\text{ff}}, \texttt{importance})$ to $\mathcal{M}$
  end if
end while

that reacts to perturbations. Additional details on the three techniques are given in Sect. 2.5 and Sect. 4.1.1.

5.1.5 Algorithm Overview

The Robust EPC algorithm leverages this formulation to achieve high-rate adaptive control while providing robust constraint satisfaction, as illustrated in Fig. 5.1 and detailed in Alg. 5.1. We incrementally construct a mapping, $\mathcal{M}$, from experiences to controllers that can be queried in future control iterations to recover the functionality of (5.10). In every control iteration, Robust EPC obtains the current state estimate, $\mathbf{x}_0$, reference sequence, $\mathbf{r}_1, \ldots, \mathbf{r}_N$, and dynamics model

$(\mathbf{A}, \mathbf{B}, \tilde{\mathbf{c}})$ updated via adaptation. It also computes the robustness bounds, $\boldsymbol{\delta}^{\mathbf{x}}$ and $\boldsymbol{\delta}^{\mathbf{u}}$, via the current state estimate covariance and the ancillary controller gains, and sets the initial state, $\boldsymbol{\mu}_0$, according to (5.18). The algorithm then searches $\mathcal{M}$ and assesses the optimality of each element via the parameterized KKT conditions (line 8). If the optimality criteria are met for any element, the search terminates and the corresponding parameterized controller is augmented with the ancillary controller (5.14) and applied.

If no element satisfies the KKT conditions (line 16), a new element is computed via (5.11) and added to $\mathcal{M}$ to extend the stored experiences to include the current scenario. To avoid blocking the control loop during this computation, a short-horizon intermediate MPC with slack on state constraints (line 17) is applied in parallel. The short horizon is selected to achieve the required control rate at the expense of degraded performance, while the slack constraints ensure feasibility even in the presence of constraint violations. Robust EPC also bounds search time by limiting the size of $\mathcal{M}$. Each element is given an `importance` score based on how recently it was used, and $\mathcal{M}$ is sorted in order of decreasing `importance`. When a new element is added, the element of $\mathcal{M}$ with the minimum `importance` may be removed to maintain the size limit (line 21). As this algorithm runs, $\mathcal{M}$ will be populated with the appropriate controllers for the current situation, thereby reducing the dependence on the intermediate controller. Additionally, due to the parameterized form of the controller gains (5.13) and KKT matrices (5.15), the elements of $\mathcal{M}$ automatically adapt to changes in the dynamics model and robustness bounds, thus maintaining robust constraint satisfaction.

5.2 Results

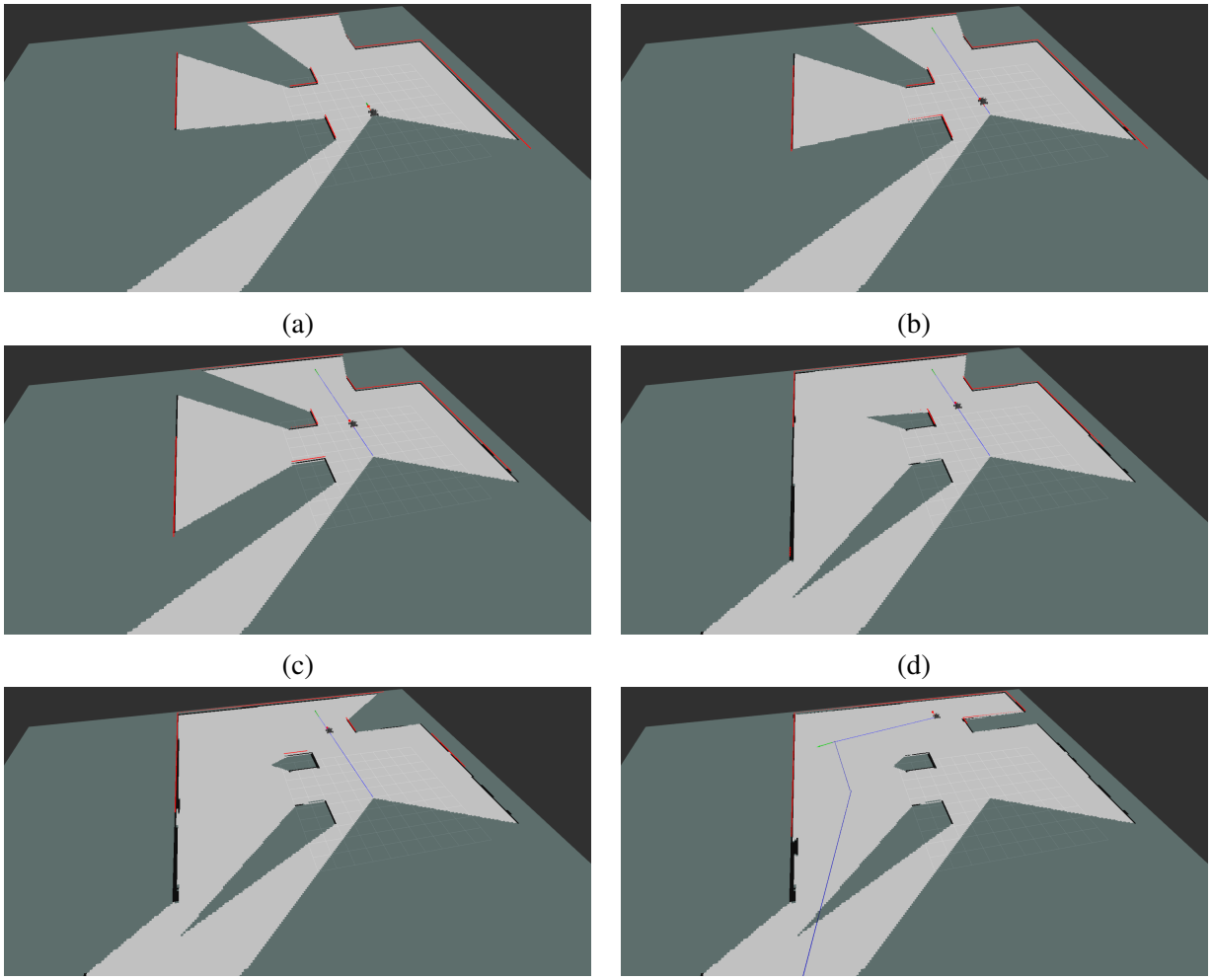
To assess the performance of the proposed Robust EPC algorithm, we aim to demonstrate the following results through simulation studies with a skid-steer ground robot and a series of flight experiments with a quadrotor micro air vehicle: **R1**: stable control performance, **R3**: real-time computation of control commands, **R5**: experience reuse, **R2.2**: constraint satisfaction in the

presence of time-varying sensor uncertainty (i.e., robust constraint satisfaction) **R4**: improved trajectory tracking performance while satisfying constraints, and **R2.3**: robust constraint satisfaction during aggressive flight.

5.2.1 Simulation Studies

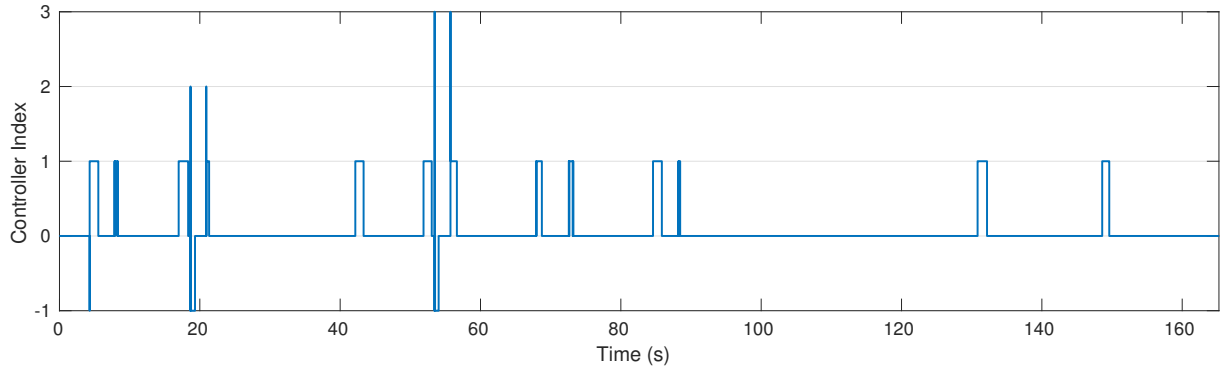
We first consider a simulated ground robot with skid-steer dynamics and odometry obtained via a simulated laser-based localization strategy (see Appendix A.3.2). The ground robot is commanded to track a set of trajectories through the environment shown in Fig. 5.3 (e.g., for exploration or mapping applications). The localization system provides state estimates but also introduces uncertainty in these estimates due to imperfect registration of laser returns, thus replicating a common source of state estimate degradation in physical robotic systems. The dynamics model in (A.2) yields an MPC formulation with $n = 8$ states and $m = 2$ inputs. We apply a horizon of $N = 10$ steps at the controller update rate (200 Hz) and enforce constraints on the two control inputs (linear and angular velocity commands) as well as the translational and rotational rates $(\dot{x}, \dot{y}, \dot{\theta})$. The chance constraint parameter α is set to 0.001 to yield a constraint satisfaction probability of 99.9%.

To evaluate robust constraint satisfaction performance in the presence of imperfect state information, we compare Robust EPC with the nominal EPC when commanded to track the same trajectory. Although the simulations are not run on a compute-constrained system (2.9 GHz Intel mobile processor), the relative query times demonstrate that the chance constrained extension does not significantly increase the compute times over regular EPC. For this trial, EPC yields a mean database query time of 0.1305 ms with a standard deviation of 0.0927 ms, while Robust EPC yields a mean of 0.1767 ms with a standard deviation of 0.1548 ms (**R3**). Additionally, both approaches learn and reuse controllers to enforce constraints, as expected. However, as Fig. 5.4 illustrates, Robust EPC computes and reuses more entries in its controller database, i.e., 17 entries, as opposed to four for EPC (**R5**). This increase is consistent with the constraint tightening

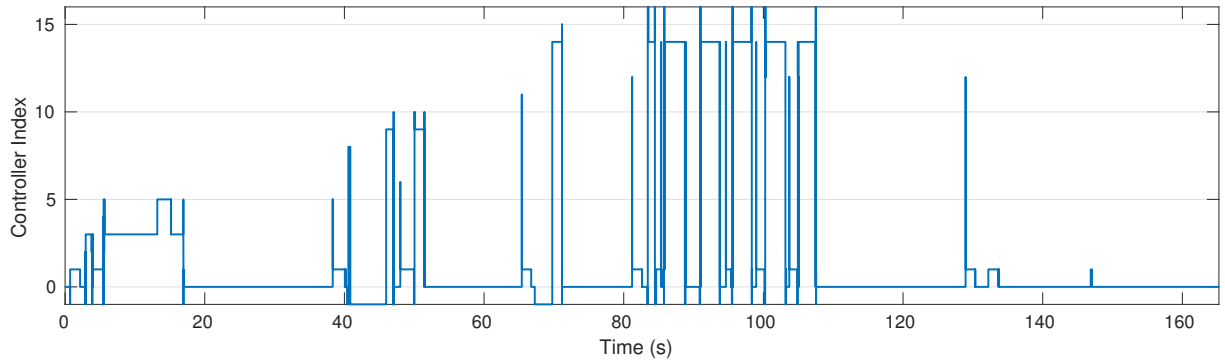


(a) (b) (c) (d) (e) (f)

Figure 5.3: A series of snapshots showing a segment of the ground robot simulation trial. The blue lines denote the trajectory being tracked by the ground robot as it traverses the unknown environment. The successive frames illustrate the simulated laser scanner (red dots denote simulated laser returns) building a map of the environment that drives the localization subsystem.



(a) EPC controller changes over the duration of the trial

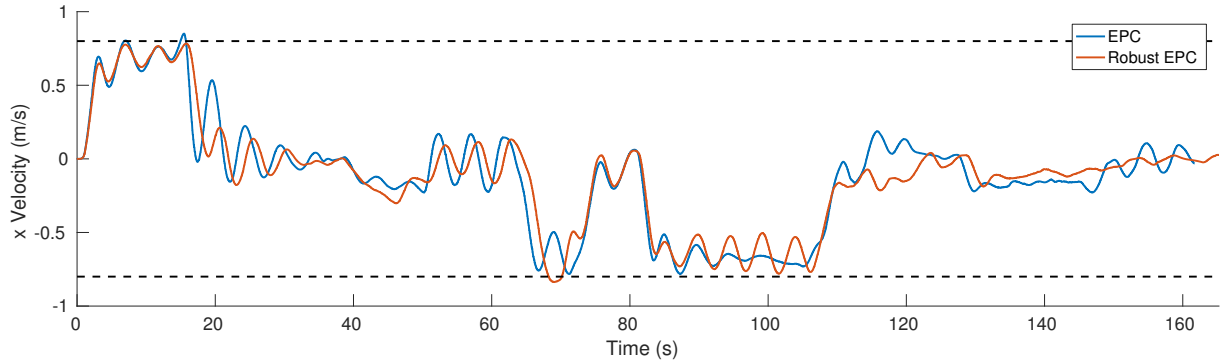


(b) Robust EPC controller changes over the duration of the trial

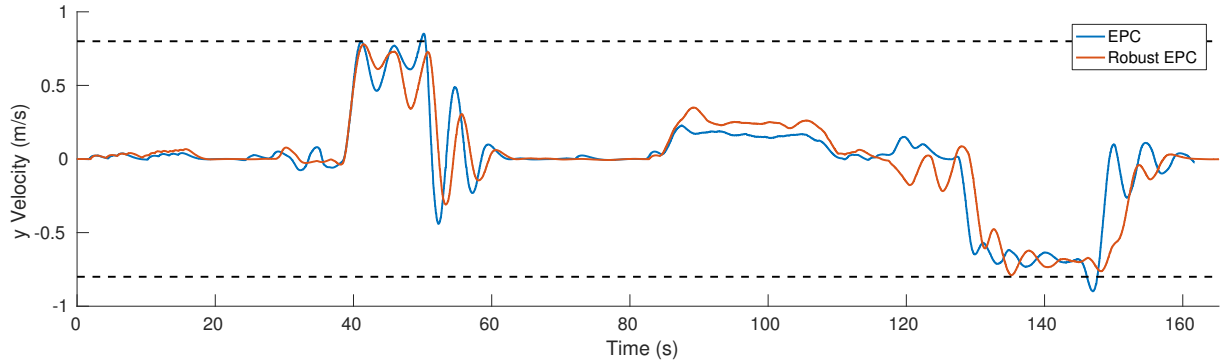
Figure 5.4: (a) EPC computes and reuses four controllers (indexed 0-3) to enforce the nominal state and input constraints, while (b) Robust EPC applies 17 controllers to ensure robust constraint satisfaction (an index of -1 denotes application of the intermediate controller).

formulation, as Robust EPC is expected to encounter the tightened constraints more frequently than EPC encounters the nominal constraints.

The effects of this increased database size and application of the corresponding controllers is evident in the resulting velocity profiles. Although both EPC and Robust EPC yield stable trajectory tracking (**R1**), as Fig. 5.5 shows, the nominal EPC formulation yields multiple velocity constraint violations along both axes. However, Robust EPC only yields one small violation of the x-axis velocity constraint, thus demonstrating robust constraint satisfaction in the presence of imperfect state information derived from the laser-based localization system (**R2.2**).



(a)



(b)

Figure 5.5: Velocity profiles for the ground robot tracking the commanded trajectory using EPC and Robust EPC. The robust formulation yields more reliable constraint satisfaction (velocity constraints shown by dashed lines).

5.2.2 Experimental Evaluation

The experimental platform is a small, 790 g quadrotor equipped with an ODROID-XU4 (2 GHz ARM processor with 2 GB RAM), as detailed in Appendix A.3.1. As state feedback is, in part, provided by a motion capture system with low variance on the estimates (Appendix A.2) we inject Gaussian noise with changing variance into the motion capture data to emulate a lower-quality sensor that exhibits changes in performance as a function of the environment (e.g., a vision-based sensor transitioning between feature-rich and feature-sparse regions). The changing uncertainty in the motion capture data is also broadcast to the state estimator and Robust EPC to inform belief state propagation via the measurement covariance term in (5.1).

For these experiments, we consider the problem of controlling the translational dynamics of

the quadrotor [81], subject to velocity and control constraints. This yields an MPC formulation with $n = 6$ states and $m = 3$ inputs. We also consider a horizon of $N = 25$ steps at the control update rate (100 Hz) for the main Robust EPC formulation. We use $\alpha = 0.001$ for a constraint satisfaction probability of 99.9%. The intermediate controller is formulated with a horizon of $N = 10$ to yield comparable solution times to Robust EPC. The cost function weight matrices for these two horizons are selected such that a finite-horizon LQR using the either set of weights (and the corresponding horizon) would yield the same controller gains. The proportional and derivative gains for the $\mathcal{L}_1$ adaptive controller used as a baseline also match this LQR formulation.

Timescale Separation with Model Adaptation

Since Robust EPC leverages EPC for its adaptive capabilities, the choice of model adaptation strategies (described in Sect. 5.1.4) should influence overall uncertainty mitigation performance. However, as these techniques seek to estimate the effects of slowly-varying disturbances acting on the system, the experiments detailed below do not require the system to operate long enough to acquire sufficient experience. As Fig. 5.6 shows, this results in comparable performance across all three adaptation strategies. While model adaptation will improve performance over an extended duration or in scenarios with significant exogenous perturbations to the system (e.g., wind acting on the quadrotor) [21], for the scenarios considered below, the choice of adaptation strategy does not significantly impact robust constraint satisfaction. Therefore, we do not distinguish between LWPR and ISSGPR in the remainder of these experiments.

Robust Constraint Satisfaction

We first evaluate Robust EPC's trajectory tracking performance along a trajectory that makes five straight line laps between two waypoints about 3.6 meters apart (Fig. 5.7a). Figure 5.7(b) shows that Robust EPC adequately stabilizes the system to track the trajectory, which achieves a

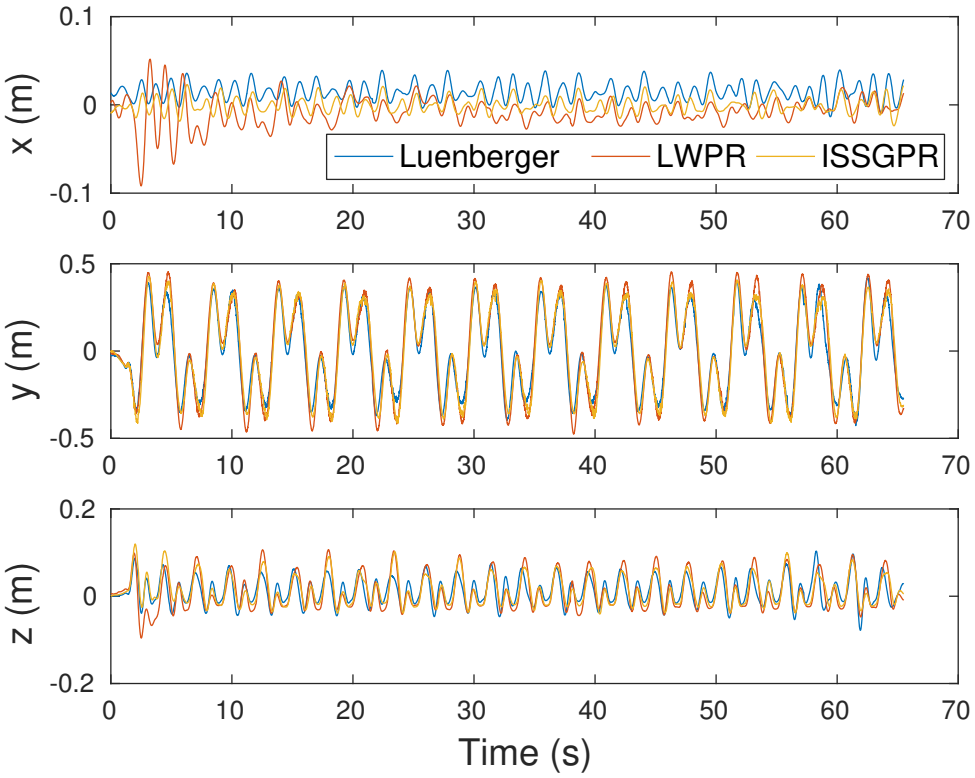


Figure 5.6: Position tracking error of the three model adaptation strategies: Luenberger, LWPR, and ISSGPR. Performance is comparable across all three.

maximum linear velocity of 2.7 m/s (**R1**).

Table 5.1 shows the compute times for the different components of Robust EPC from one representative trial. This demonstrates that both the Query and Intermediate controller components, which constitute the primary control thread, run in real-time on the computationally constrained flight hardware (**R3**).

In order to show robust constraint satisfaction in the presence of time-varying sensor uncertainty, we set up an experiment where Gaussian noise is injected into the motion capture data. To increase the difficulty of velocity constraint satisfaction, we inject noise in the center of each lap of the trajectory, where the velocity of the vehicle is highest. In the experiments that follow, we use mean-zero Gaussian noise with a standard deviation of 0.03 that is applied when the position of the vehicle along the y-axis is between -0.5 meters and 0.5 meters.

In addition to Robust EPC, we consider three baseline control strategies: $\mathcal{L}_1$ adaptive control,

EPC, and a Robust MPC (R-MPC) formulation that solves the QP online with $N = 10$ (to achieve comparable solution times) and slack on state constraints (to ensure problem feasibility). Figure 5.9 shows the resulting velocity profiles with the constraint bounds shown by the dashed lines. $\mathcal{L}_1$ adaptive control shows unconstrained control performance, which naturally violates the constraints as the reference velocity has a maximum of 2.7 m/s. The enforcement of constraints in EPC yields smaller constraint violations, but the non-robust formulation of the constraints fails to mitigate the effects of measurement uncertainty. R-MPC also exhibits substantial constraint violations. To confirm that the degraded performance of R-MPC is due to the short horizon and not the slack constraints, we also compared performance of R-MPC and Robust EPC with $N = 25$ using a high-fidelity simulator on a more powerful computer and observed comparable performance and robust constraint satisfaction. Therefore, these results illustrate that over repeat trials, only Robust EPC consistently satisfies the velocity constraints (**R2.2**).

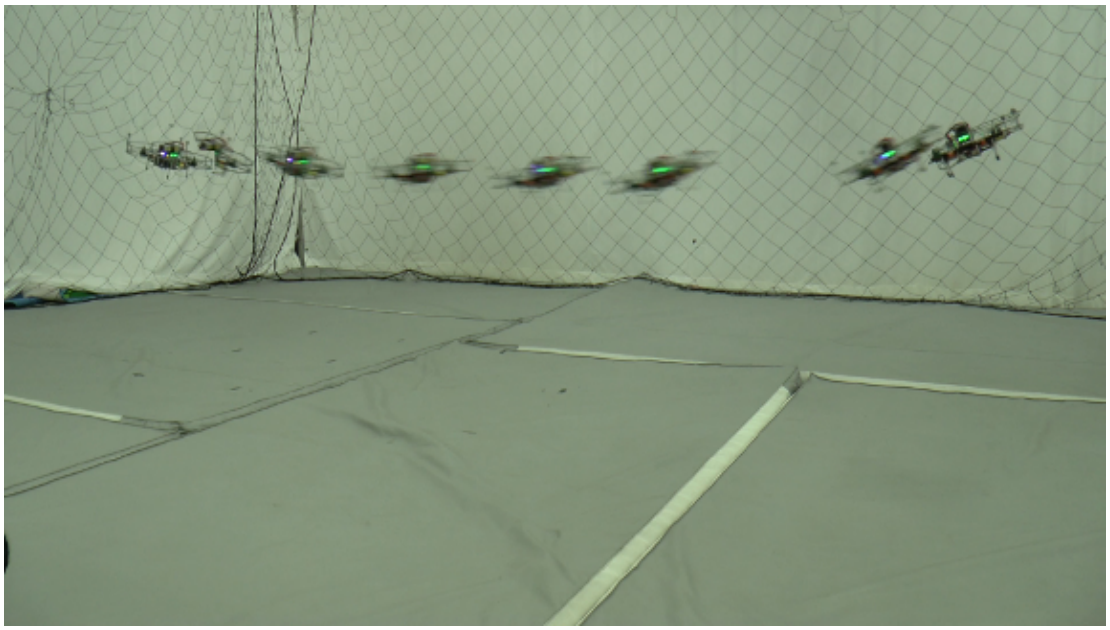
Figure 5.8 illustrates controller generation and reuse as indicated by the amount of time each controller is applied. Note that the intermediate controller (index 1) is only used in the first few laps, while controller 2 (corresponding to operation away from constraints) is applied frequently (**R5**). This indicates that over time, all of the necessary controllers needed to track the trajectory and satisfy constraints are enumerated and available for use in the controller table.

Time-Varying Uncertainty Prediction

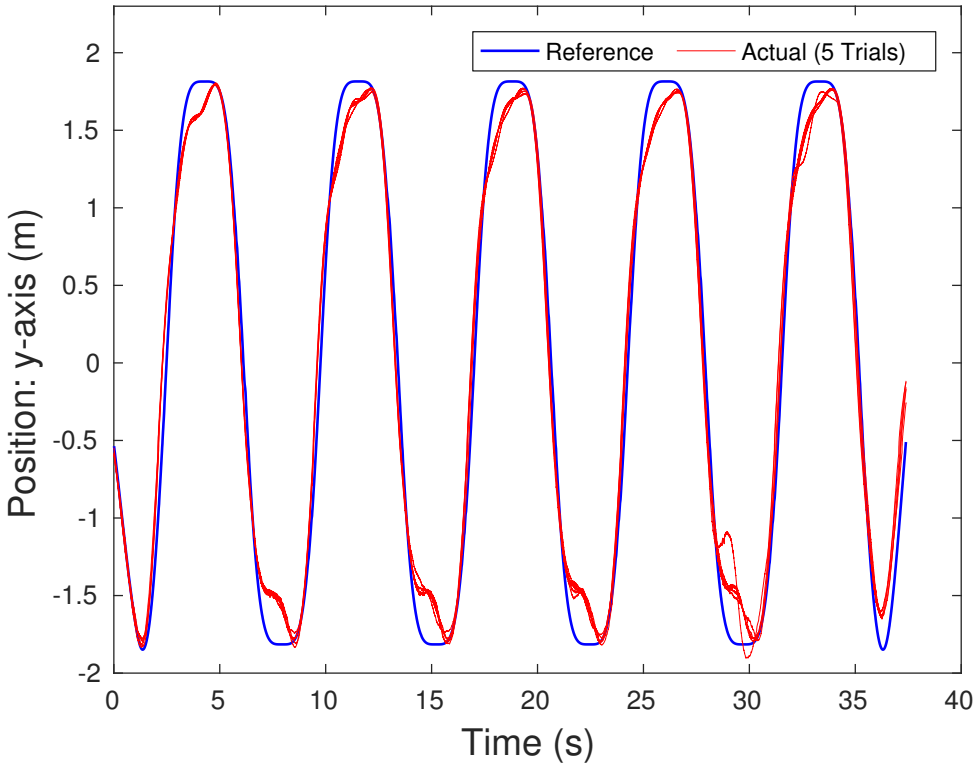
To show that Robust EPC leverages the Gaussian nature of the state estimator output and exploits regions of low uncertainty to improve performance over more conservative approaches, we inves-

Table 5.1: Compute times for Robust EPC components, including the number of control iterations over which the statistics are computed.

	Databse Query	Intermediate Controller	Solve QP	Add Element
Iterations	5949	18	12	12
Mean (ms)	**1.089**	1.303	4.427	4.891
Std. Dev. (ms)	**1.463**	0.886	2.720	5.393



(a) The back and forth trajectory visualized using video stills.



(b) Tracking performance of Robust EPC while executing the back and forth trajectory.

Figure 5.7: Vehicle executing a back and forth trajectory with five laps.

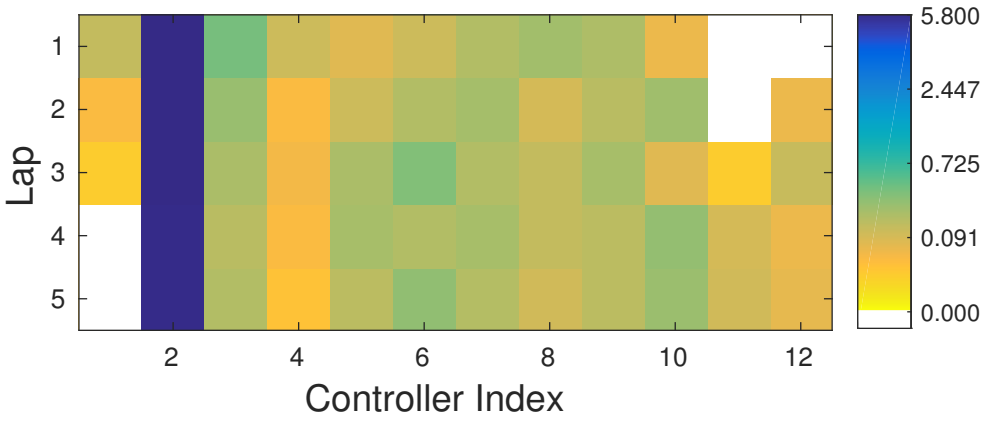
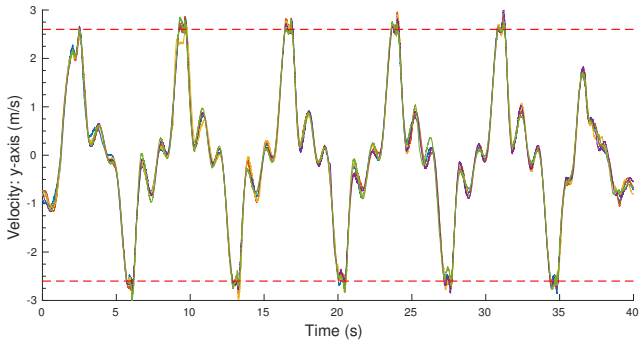
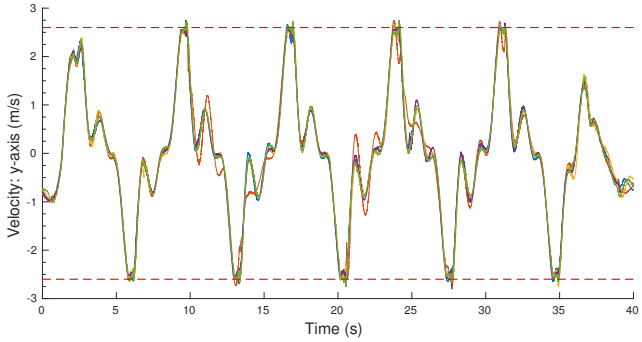


Figure 5.8: Time spent using each controller per lap. Note that multiple controllers are learned and reused and that the intermediate controller (index 1) ceases to be used past lap 3.

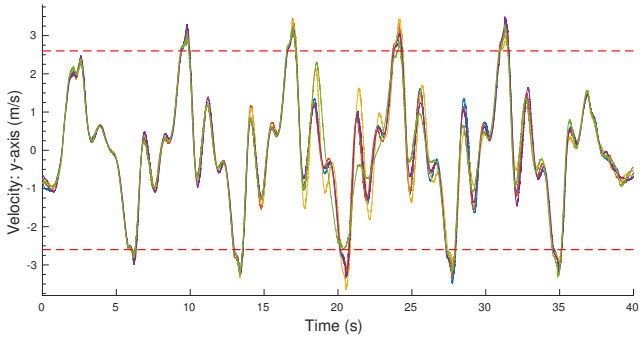
tigate its performance compared to an instantiation of Robust EPC that uses a fixed upper bound on the uncertainty. We take the maximum bound applied by Robust EPC during a run of the trajectory as the uncertainty value for this fixed bound approach. The quadrotor is commanded to track a vertical circle trajectory (Fig. 5.10) while Gaussian noise with a standard deviation of 0.03 is injected when the vehicle is below 1 meter in height. We expect that Robust EPC will exploit the low noise region above 1 meter and achieve better performance than the conservative approach. Figure 5.11 shows tracking results for Robust EPC using Gaussian belief propagation, the fixed bound approach using the true upper bound as described above, and the fixed bound approach using the highest bound that allows for stable trajectory tracking. The fixed bound approach is unable to complete the trajectory with the true upper bound, and Robust EPC yields reduced tracking error compared to the less conservative fixed bound approach (**R4**). We also consider an uncertainty propagation strategy based on recursive application of the Pontryagin difference with the uncertainty set [73]. However, even with the ancillary controller, this results in an infeasible problem for the longer horizons permitted by Robust EPC. Figure 5.12 illustrates the tube growth with a 25-step horizon for the two approaches.



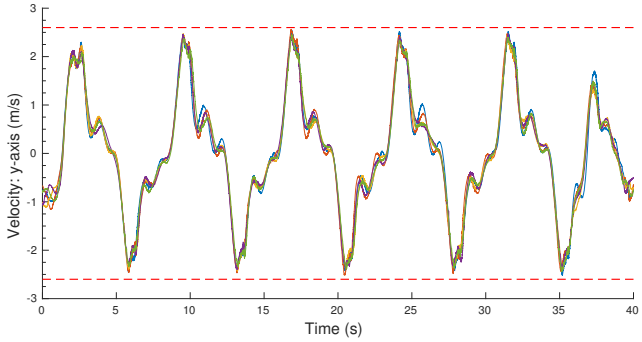
(a) $\mathcal{L}_1$ Adaptive Control



(b) EPC (N = 25)



(c) Robust MPC (QP, N = 10)



(d) Robust EPC (N = 25)

Figure 5.9: Comparison of y-velocity profiles for the line trajectory across 10 trials of each controller. Only Robust EPC satisfies the nominal velocity constraints (dashed lines).

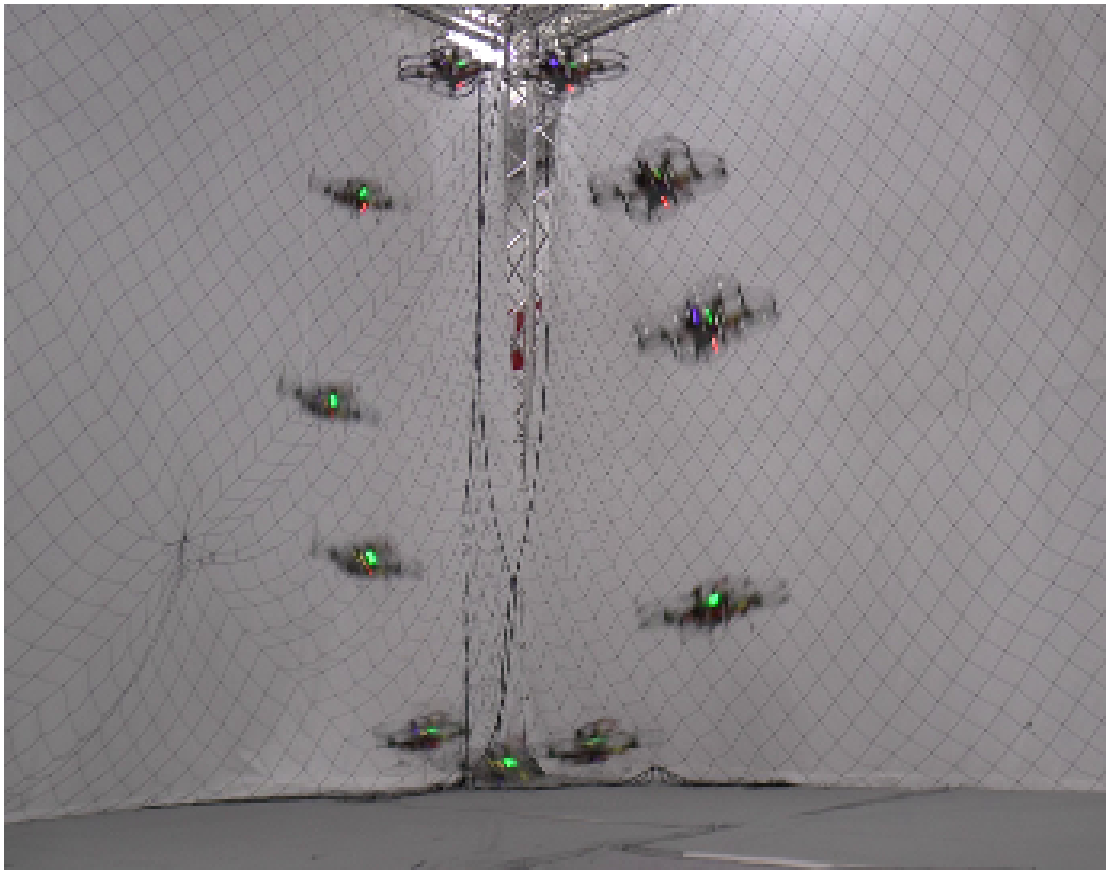


Figure 5.10: The vertical circle trajectory used to assess belief propagation, visualized using video stills.

Aggressive Flight

To further assess the performance of Robust EPC, we consider two aggressive scenarios. The first scenario aims to test constraint satisfaction on a high speed back and forth trajectory with a maximum velocity of 3.6 m/s. Figure 5.13 displays the velocity along the trajectory. Constraints are satisfied throughout with the exception of a 0.03 m/s violation during the final, fastest lap. Due to the chance-constrained nature of Robust EPC, there is a nonzero probability of constraint violation (0.1% in our experiments). In addition, higher speeds typically accentuate any modeling errors present in the system and, as a result, may yield degraded tracking performance if the model learning components are unable to adapt at a sufficiently high rate or have not accumulated sufficient experience (as noted in Sect. 5.2.2).

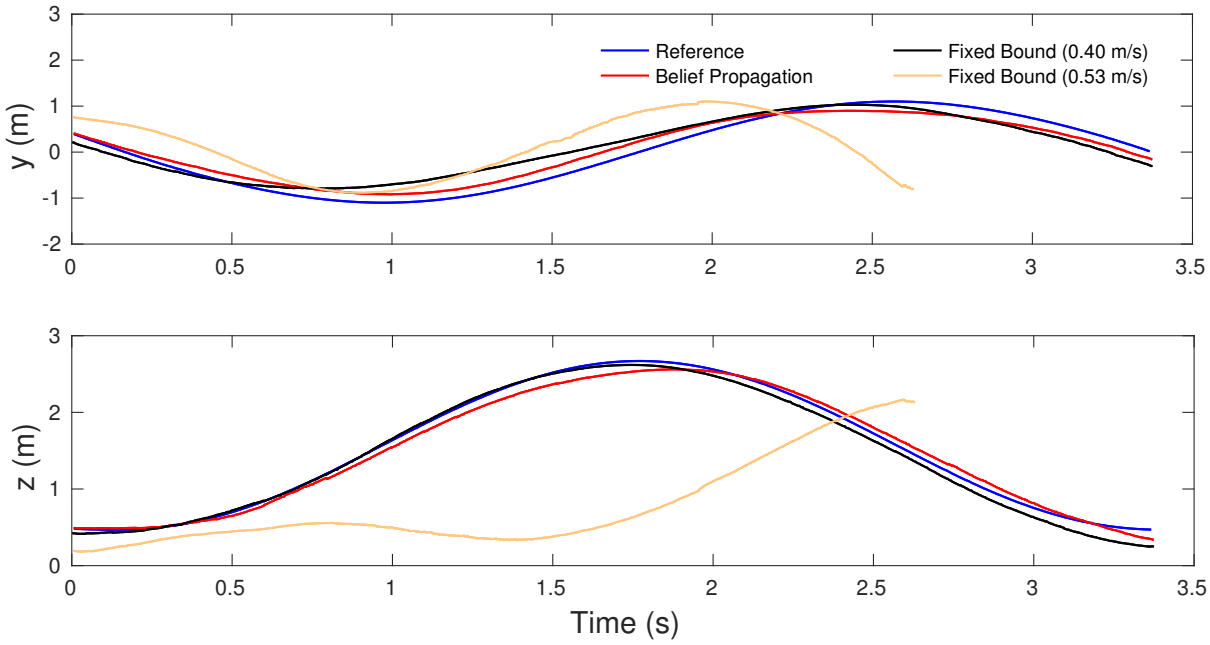


Figure 5.11: Position along the y and z axes for Robust EPC and the fixed bound approach as compared to the reference trajectory. The fixed bound approach that uses the true upper bound fails to track the trajectory. The mean and max error for Robust EPC along the y-axis are 0.22 and 0.41, respectively, while for the successful fixed bound approach, 0.24 and 0.51.

In the second aggressive flight scenario, the quadrotor is commanded to fly three laps around a circle in the x-y plane that traverses a turbulent wind field generated by eight, high-power fans, as illustrated in Figs. 5.14 and 5.15. The average wind velocity directly in front of each fan is approximately 6 m/s [5], and the placement of the fans around the flight volume results in significant spatial variation in the disturbance forces acting on the vehicle. The reference trajectory commands a maximum velocity of 2.0 m/s, but due to the wind field, the vehicle may often overshoot the command. However, Robust EPC enforces a velocity limit of 2.3 m/s, and as Fig. 5.16 shows, the resulting velocity profile satisfies this constraint with just one minor violation. As a result, we conclude that Robust EPC adequately handles constraints even during aggressive flight (**R2.3**).

Due to the velocity constraints being activated in each lap, it is difficult to assess flight performance via the tracking error (e.g., a high velocity may be needed to overcome wind-induced lag). We therefore look at cross-track error as a measure of the deviation from the trajectory, as shown

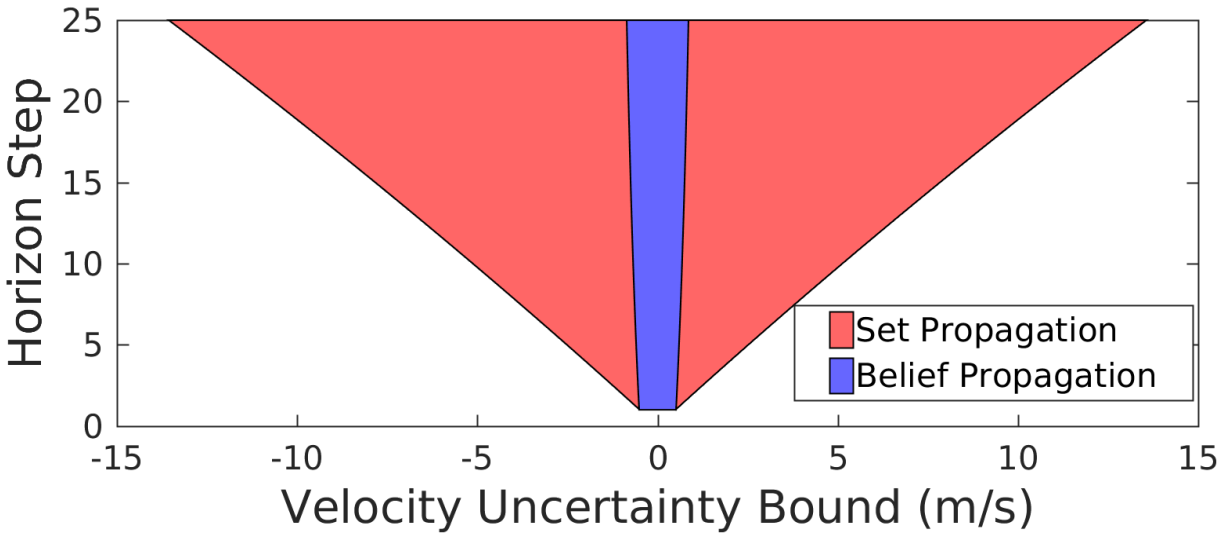


Figure 5.12: Overlay of tube growth for Set Propagation and Belief Propagation based on the bounds computed by each at the start of trajectory tracking. Set Propagation growth is too fast to yield feasible constraints.

in Fig. 5.17. As Table 5.2 shows, the cross-track error about all three axes improves significantly by the third lap as the controller database and model learner accumulate sufficient experience. This also matches empirical observations during the flight test that the vehicle exhibits improved stability and smoothness over successive laps (**R4.1**).

Table 5.2: Cross-track error statistics for the high-wind circle trajectory

	x-axis		y-axis		z-axis	
	Mean (m)	Std. Dev. (m)	Mean (m)	Std. Dev. (m)	Mean (m)	Std. Dev. (m)
Lap 1	0.0287	0.0893	0.0075	0.0694	0.0161	0.0364
Lap 2	0.0266	0.1060	0.0061	0.1057	0.0242	0.0528
Lap 3	0.0110	0.0785	0.0012	0.0740	0.0022	0.0378

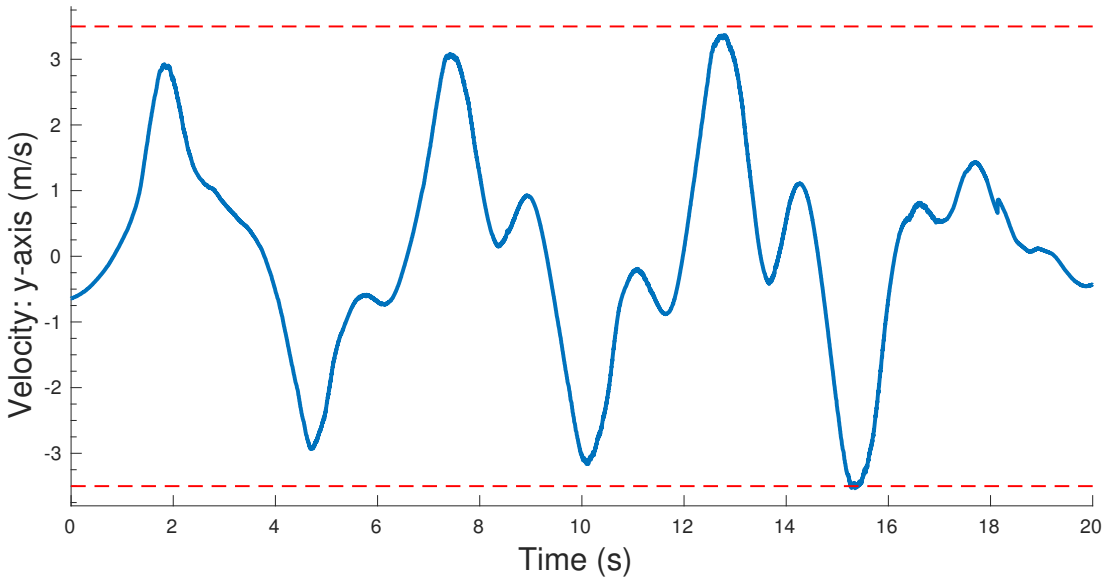


Figure 5.13: Velocity of Robust EPC along a high-speed back and forth trajectory. There is a small constraint violation of 0.03 m/s during the last lap.

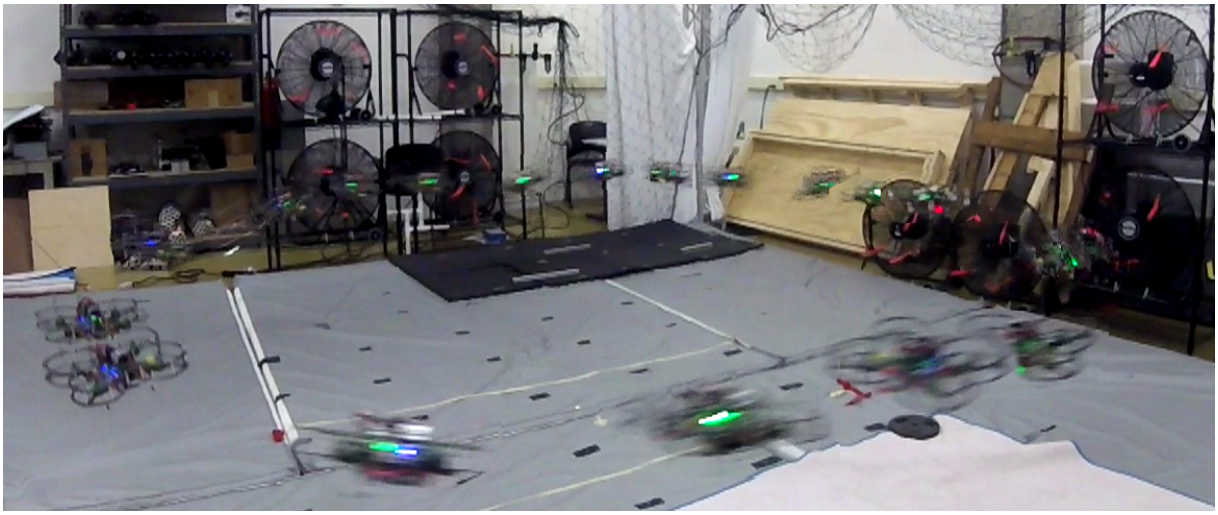


Figure 5.14: Snapshots of the horizontal circle trajectory executed in a high-speed, turbulent wind field generated via a set of eight high-power fans

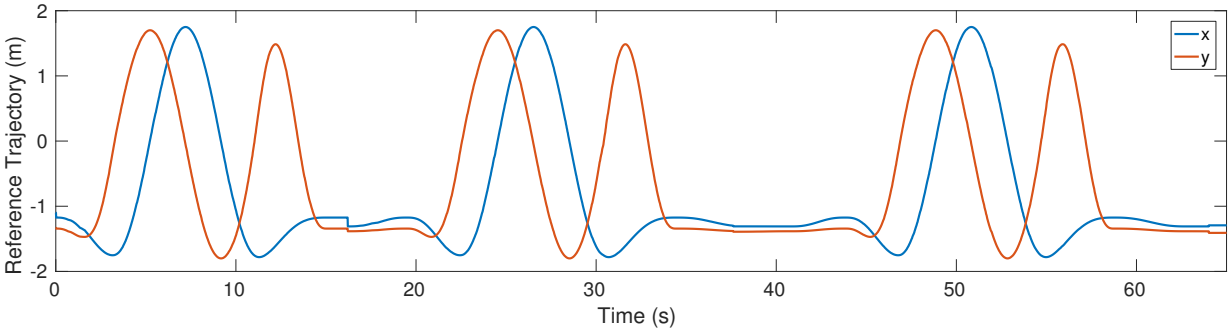


Figure 5.15: x and y components of the horizontal circle trajectory showing the three laps executed

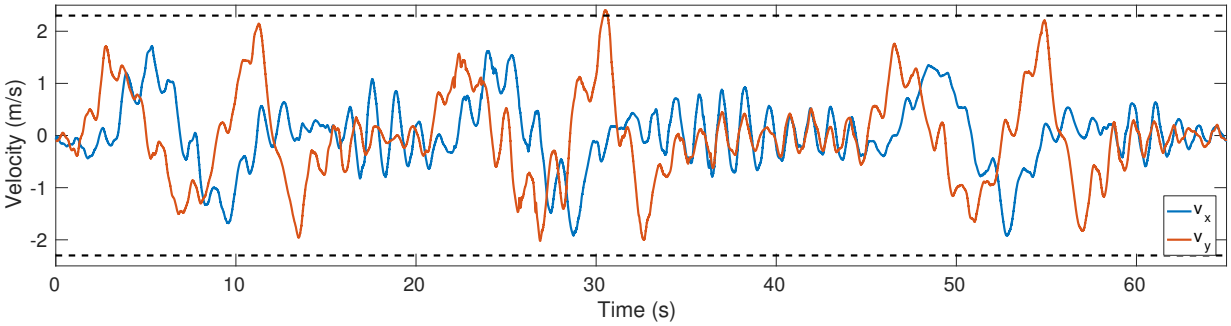


Figure 5.16: Velocity of Robust EPC along the circle trajectory in the high-wind scenario. The velocity obeys the constraint bound aside from one minor constraint violation of 0.09 m/s.

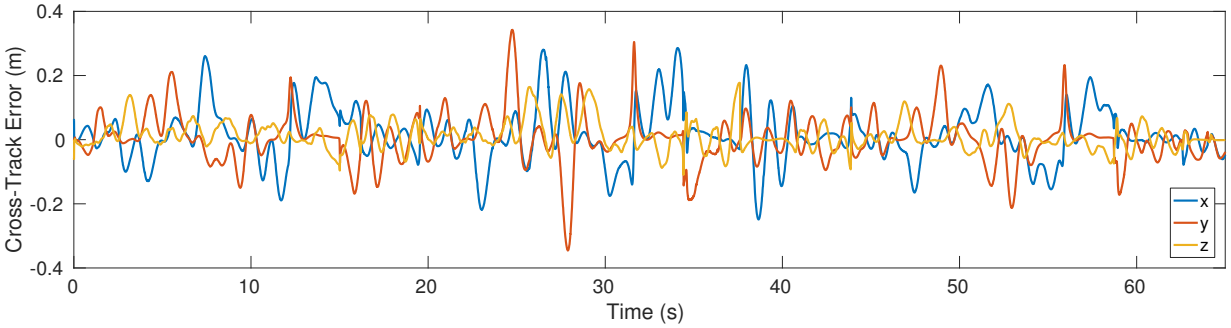


Figure 5.17: Cross-track error while executing the circle trajectory in the high-wind scenario is nearly zero-mean and shows some improvement over time.

5.3 Conclusions

In this chapter, we have presented an extension to EPC that allows for robust constraint satisfaction in the presence of time and state-dependent uncertainty. We have shown that the proposed system, Robust EPC, successfully stabilizes the vehicle along a variety of trajectories (**R1**), easily meets computational requirements on a compute-constrained system (**R3**), properly satisfies constraints in the presence of time-varying sensor uncertainty (**R2.2**) while improving tracking performance as compared to conservative methods (**R4**), and maintains constraint satisfaction and tracking improvement properties during aggressive flight (**R2.3**).

Chapter 6

Efficient Explicit Adaptive Nonlinear MPC

In this chapter we propose an explicit adaptive NMPC algorithm that builds upon the previously described semi-explicit techniques by learning from synthetic experiences. One of the key observations from semi-explicit NMPC is that, in practice, the system only uses a small number of the potential controllers that an explicit approach would compute. Therefore, we propose to compute a limited database consisting of the controllers required to track feasible trajectories, e.g., trajectories that lie within the reachable space of the system controlled by NMPC. To do so, we construct a randomized search tree through the reachable space to generate synthetic experiences and apply the EPC algorithm to incrementally construct a controller database from these experiences. We model transitions between controllers in the database as a Markov chain and use empirical transition probabilities to specify a partial ordering on successors for each controller. This approach improves the efficiency of database queries and permits further reductions of the database size, thereby enabling use on computationally constrained platforms.

While the reachable-space search shares similarities with the LQR-Tree algorithm [53], the proposed approach addresses the problem of constrained optimal control for arbitrary (feasible) trajectory tracking rather than the generation of stabilizing trajectory or controller sequences. Alessio, et al. [39] also propose a means of generating a reduced explicit MPC database through repeated simulation, but the proposed approach extends this technique by using a tree of fea-

sible trajectories rather than arbitrary states and references to guide the simulations. Finally, the proposed approach enables the controllers queried online to automatically adapt to changes in the system dynamics, due to its use of EPC. The full adaptive NMPC formulation, database generation, and online query algorithms are detailed in the following section.

6.1 Approach

Explicit Nonlinear Model Predictive Control (NMPC) constructs a database of locally optimal affine feedback controllers that solve a given NMPC problem without the need for online optimization. The explicit NMPC technique proposed in this section combines the Experience-driven Predictive Control (EPC) algorithm (Ch. 4) with a randomized search tree to generate synthetic experiences that can populate a database of local feedback controllers. This offline-constructed database can then be queried during online operation for high-rate control that is equivalent to solving the NMPC problem but requires no online optimization.

6.1.1 Predictive Control Formulation

We consider the adaptive NMPC formulation used in EPC, summarized below, that leverages an online-learned model of plant dynamics to simplify the optimization problem. As described in Ch. 4, EPC employs techniques such as Locally Weighted Projection Regression (LWPR) or Incremental Sparse Spectrum Gaussian Process Regression (ISSGPR) to learn perturbations to the dynamics model online. Given a state-control pair, $(\mathbf{x}, \mathbf{u})$, the model learner returns the anticipated error, $\hat{\mathbf{p}}$, between the predicted and actual next state. Combining $\hat{\mathbf{p}}$ with a first order Taylor-series approximation of the nonlinear equations of motion (computed about the current state $\mathbf{x}_0$ and a nominal control input $\mathbf{u}_\mathrm{r}$ derived from the trajectory) yields an affine model of the

system's dynamics that evolves over time,

$$\begin{aligned}\bar{\mathbf{x}}_{k+1} &= \mathbf{A}\bar{\mathbf{x}}_k + \mathbf{B}\bar{\mathbf{u}}_k + \mathbf{c} + \hat{\mathbf{p}} \\ &= \mathbf{A}\bar{\mathbf{x}}_k + \mathbf{B}\bar{\mathbf{u}}_k + \tilde{\mathbf{c}},\end{aligned}$$

where $\bar{\mathbf{x}}_k = \mathbf{x}_k - \mathbf{x}_0$, $\bar{\mathbf{u}}_k = \mathbf{u}_k - \mathbf{u}_\mathrm{r}$, and $\tilde{\mathbf{c}} = \mathbf{c} + \hat{\mathbf{p}}$.

This affine model enables the N-step receding-horizon control problem to be formulated as a quadratic program (QP) rather than as a nonlinear program,

$$\begin{aligned}\underset{\bar{\mathbf{u}}_k}{\operatorname{argmin}} \sum_{k=0}^{N-1} \frac{1}{2}(\bar{\mathbf{x}}_{k+1} - \bar{\mathbf{r}}_{k+1})^\mathrm{T}\mathbf{Q}_{k+1}(\bar{\mathbf{x}}_{k+1} - \bar{\mathbf{r}}_{k+1}) + \frac{1}{2}(\bar{\mathbf{u}}_k - \bar{\mathbf{u}}_{\hat{\mathbf{p}}})^\mathrm{T}\mathbf{R}_k(\bar{\mathbf{u}}_k - \bar{\mathbf{u}}_{\hat{\mathbf{p}}}) \\ \text{s.t.} \quad \bar{\mathbf{x}}_{k+1} = \mathbf{A}\bar{\mathbf{x}}_k + \mathbf{B}\bar{\mathbf{u}}_k + \tilde{\mathbf{c}} \\ \mathbf{G}_{\mathbf{x}_{k+1}}\bar{\mathbf{x}}_{k+1} \leq \mathbf{g}_{\mathbf{x}_{k+1}} \\ \mathbf{G}_{\mathbf{u}_k}\bar{\mathbf{u}}_k \leq \mathbf{g}_{\mathbf{u}_k} \\ \forall\, k = \{0, \ldots, N-1\}\end{aligned}$$

where $\{\mathbf{r}_1, \ldots, \mathbf{r}_N\}$ denote N reference states along the current trajectory, $\bar{\mathbf{r}}_k = \mathbf{r}_k - \mathbf{x}_0$, and $\bar{\mathbf{u}}_{\hat{\mathbf{p}}}$ is the compensatory control derived from the LWPR output. To simplify notation, we define $\boldsymbol{x} = \begin{bmatrix}\bar{\mathbf{x}}_1^\mathrm{T}, \ldots, \bar{\mathbf{x}}_N^\mathrm{T}\end{bmatrix}^\mathrm{T}$, $\boldsymbol{r} = \begin{bmatrix}\bar{\mathbf{r}}_1^\mathrm{T}, \ldots, \bar{\mathbf{r}}_N^\mathrm{T}\end{bmatrix}^\mathrm{T}$, $\boldsymbol{u} = \begin{bmatrix}\bar{\mathbf{u}}_0^\mathrm{T}, \ldots, \bar{\mathbf{u}}_{N-1}^\mathrm{T}\end{bmatrix}^\mathrm{T}$, $\boldsymbol{u}_{\hat{\mathbf{p}}} = \begin{bmatrix}\bar{\mathbf{u}}_{\hat{\mathbf{p}}}^\mathrm{T}, \ldots, \bar{\mathbf{u}}_{\hat{\mathbf{p}}}^\mathrm{T}\end{bmatrix}^\mathrm{T}$,

$$\boldsymbol{\mathcal{B}} = \begin{bmatrix}\mathbf{B} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{AB} & \mathbf{B} & \ldots & \mathbf{0} \\ \vdots & \vdots & \ddots & \\ \mathbf{A}^{N-1}\mathbf{B} & \mathbf{A}^{N-2}\mathbf{B} & \ldots & \mathbf{B}\end{bmatrix}, \; \boldsymbol{c} = \begin{bmatrix}\tilde{\mathbf{c}} \\ (\mathbf{A}+\mathbf{I})\,\tilde{\mathbf{c}} \\ \vdots \\ \sum_{i=0}^{N-1}\mathbf{A}^i\tilde{\mathbf{c}}\end{bmatrix}$$

$\boldsymbol{\mathcal{Q}} = \operatorname{diag}(\mathbf{Q}_1, \ldots, \mathbf{Q}_N)$, $\boldsymbol{\mathcal{R}} = \operatorname{diag}(\mathbf{R}_0, \ldots, \mathbf{R}_{N-1})$, $\boldsymbol{\mathcal{G}}_\mathbf{x} = \operatorname{diag}(\mathbf{G}_{\mathbf{x}_1}, \ldots, \mathbf{G}_{\mathbf{x}_N})$, $\boldsymbol{\mathcal{G}}_\mathbf{u} = \operatorname{diag}(\mathbf{G}_{\mathbf{u}_0}, \ldots, \mathbf{G}_{\mathbf{u}_{N-1}})$, $\boldsymbol{g}_\mathbf{x} = \begin{bmatrix}\mathbf{g}_{\mathbf{x}_1}^\mathrm{T}, \ldots, \mathbf{g}_{\mathbf{x}_N}^\mathrm{T}\end{bmatrix}^\mathrm{T}$, and $\boldsymbol{g}_\mathbf{u} = \begin{bmatrix}\mathbf{g}_{\mathbf{u}_0}^\mathrm{T}, \ldots, \mathbf{g}_{\mathbf{u}_{N-1}}^\mathrm{T}\end{bmatrix}^\mathrm{T}$. Since

$\bar{\mathbf{x}}_0 = \mathbf{0}$, the QP (6.1) can we rewritten as

$$\begin{aligned} \underset{\boldsymbol{u}}{\operatorname{argmin}} \; & \frac{1}{2}(\boldsymbol{x}-\boldsymbol{r})^{\mathrm{T}}\boldsymbol{\mathcal{Q}}(\boldsymbol{x}-\boldsymbol{r}) + \frac{1}{2}(\boldsymbol{u}-\boldsymbol{u}_{\hat{\mathbf{p}}})^{\mathrm{T}}\boldsymbol{\mathcal{R}}(\boldsymbol{u}-\boldsymbol{u}_{\hat{\mathbf{p}}}) \\ \text{s.t.} \quad & \boldsymbol{x} = \boldsymbol{\mathcal{B}}\boldsymbol{u} + \boldsymbol{c} \\ & \boldsymbol{\mathcal{G}}_{\mathbf{x}}\boldsymbol{x} \leq \boldsymbol{g}_{\mathbf{x}} \\ & \boldsymbol{\mathcal{G}}_{\mathbf{u}}\boldsymbol{u} \leq \boldsymbol{g}_{\mathbf{u}} \end{aligned}$$

Substituting the system dynamics into the cost and constraints and dropping constant terms in the cost function yields an equivalent QP in terms of $\boldsymbol{u}$,

$$\begin{aligned} \underset{\boldsymbol{u}}{\operatorname{argmin}} \; & \frac{1}{2}\boldsymbol{u}^{\mathrm{T}}\boldsymbol{\mathcal{H}}\boldsymbol{u} + \boldsymbol{h}^{\mathrm{T}}\boldsymbol{u} \\ \text{s.t.} \quad & \boldsymbol{\Gamma}\boldsymbol{u} \leq \boldsymbol{\gamma} \end{aligned} \tag{6.1}$$

where $\boldsymbol{\mathcal{H}} = \boldsymbol{\mathcal{B}}^{\mathrm{T}}\boldsymbol{\mathcal{Q}}\boldsymbol{\mathcal{B}} + \boldsymbol{\mathcal{R}}$, $\boldsymbol{h} = \boldsymbol{\mathcal{B}}^{\mathrm{T}}\boldsymbol{\mathcal{Q}}(\boldsymbol{c}-\boldsymbol{r}) - \boldsymbol{\mathcal{R}}\boldsymbol{u}_{\hat{\mathbf{p}}}$,

$$\boldsymbol{\Gamma} = \begin{bmatrix} \boldsymbol{\mathcal{G}}_{\mathbf{x}}\boldsymbol{\mathcal{B}} \\ \boldsymbol{\mathcal{G}}_{\mathbf{u}} \end{bmatrix}, \text{ and } \boldsymbol{\gamma} = \begin{bmatrix} \boldsymbol{g}_{\mathbf{x}} - \boldsymbol{\mathcal{G}}_{\mathbf{x}}\boldsymbol{c} \\ \boldsymbol{g}_{\mathbf{u}} \end{bmatrix}$$

As in other explicit and semi-explicit MPC techniques, we apply a standard result from multi-parametric QPs to derive optimal, local controllers from the Karush-Kuhn-Tucker (KKT) conditions for optimality[46],

$$\begin{aligned} \boldsymbol{\mathcal{H}}\boldsymbol{u} + \boldsymbol{h} + \Gamma^{\mathrm{T}}\boldsymbol{\lambda} &= \mathbf{0} \\ \boldsymbol{\Lambda}(\boldsymbol{\Gamma}\boldsymbol{u} - \boldsymbol{\gamma}) &= \mathbf{0}, \end{aligned} \tag{6.2}$$

where $\boldsymbol{\lambda}$ is the vector of Lagrange multipliers and $\boldsymbol{\Lambda} = \operatorname{diag}(\boldsymbol{\lambda})$. This allows $\boldsymbol{u}$ and $\boldsymbol{\lambda}$ to be

reconstructed by solving a linear system derived from (6.2),

$$\begin{bmatrix} \boldsymbol{\mathcal{H}} & \boldsymbol{\Gamma}_a^{\mathrm{T}} \\ \boldsymbol{\Gamma}_a & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{\lambda}_a \end{bmatrix} = \begin{bmatrix} -\boldsymbol{h} \\ \boldsymbol{\gamma}_a \end{bmatrix}$$

where the subscript a denotes rows corresponding to the active constraints (i.e., with $\lambda > 0$) for a given solution. Assuming a linearly independent set of active constraints [47], the resulting $\boldsymbol{u}$ is affine in the predicted state error $\boldsymbol{r}$,

$$\boldsymbol{u} = \boldsymbol{\mathcal{E}}_5 \boldsymbol{r} - \left(\boldsymbol{\mathcal{E}}_5 \boldsymbol{c} - \boldsymbol{\mathcal{E}}_4 \boldsymbol{\mathcal{R}} \boldsymbol{u}_{\hat{\mathbf{p}}} + \boldsymbol{\mathcal{E}}_3 \begin{bmatrix} \boldsymbol{g}_{\mathbf{x}}^{+} - \boldsymbol{\mathcal{G}}_{\mathbf{x}} \boldsymbol{c} \\ -\boldsymbol{g}_{\mathbf{x}}^{-} + \boldsymbol{\mathcal{G}}_{\mathbf{x}} \boldsymbol{c} \\ \boldsymbol{g}_{\mathbf{u}}^{+} \\ -\boldsymbol{g}_{\mathbf{u}}^{-} \end{bmatrix}_a \right), \tag{6.3}$$

where $\boldsymbol{\mathcal{E}}_1 = \boldsymbol{\Gamma}_a \boldsymbol{\mathcal{H}}^{-1}$, $\boldsymbol{\mathcal{E}}_2 = -(\boldsymbol{\mathcal{E}}_1 \boldsymbol{\Gamma}_a^{\mathrm{T}})^{-1}$, $\boldsymbol{\mathcal{E}}_3 = \boldsymbol{\mathcal{E}}_1^{\mathrm{T}} \boldsymbol{\mathcal{E}}_2$, $\boldsymbol{\mathcal{E}}_4 = \boldsymbol{\mathcal{H}}^{-1} + \boldsymbol{\mathcal{E}}_3 \boldsymbol{\mathcal{E}}_1$, and $\boldsymbol{\mathcal{E}}_5 = \boldsymbol{\mathcal{E}}_4 \boldsymbol{\mathcal{B}}^{\mathrm{T}} \boldsymbol{\mathcal{Q}}$. Moreover, since the coefficients in (6.3) are all functions of $\mathbf{A}$, $\mathbf{B}$, and $\tilde{\mathbf{c}}$, the overall control law $\boldsymbol{\kappa}(\mathbf{x}_0, \mathbf{r}_1, \ldots, \mathbf{r}_N)$ can be written in terms of a parameterized feedback gain matrix $\mathbf{K}$ and feedforward vector $\mathbf{k}_{\text{ff}}$,

$$\boldsymbol{\kappa}(\mathbf{x}_0, \mathbf{r}_1, \ldots, \mathbf{r}_N) = \mathbf{K}(\mathbf{A}, \mathbf{B}, \tilde{\mathbf{c}}) \boldsymbol{r} + \mathbf{k}_{\text{ff}}(\mathbf{A}, \mathbf{B}, \tilde{\mathbf{c}}). \tag{6.4}$$

The KKT condition checks (used to determine whether a previously computed controller is locally optimal) have a similar structure. The active Lagrange multipliers, $\boldsymbol{\lambda}_a$, also take a similar

form to the control law,

$$\boldsymbol{\lambda}_a = -\boldsymbol{\mathcal{E}}_6 \boldsymbol{r} + \left(\boldsymbol{\mathcal{E}}_6 \boldsymbol{c} - \boldsymbol{\mathcal{E}}_3^{\mathrm{T}} \boldsymbol{\mathcal{R}} \boldsymbol{u}_{\hat{\mathbf{p}}} + \boldsymbol{\mathcal{E}}_2 \begin{bmatrix} \boldsymbol{g}_{\mathbf{x}}^{+} - \boldsymbol{\mathcal{G}}_{\mathbf{x}} \boldsymbol{c} \\ -\boldsymbol{g}_{\mathbf{x}}^{-} + \boldsymbol{\mathcal{G}}_{\mathbf{x}} \boldsymbol{c} \\ \boldsymbol{g}_{\mathbf{u}}^{+} \\ -\boldsymbol{g}_{\mathbf{u}}^{-} \end{bmatrix}_a \right) \tag{6.5}$$

where $\boldsymbol{\mathcal{E}}_6 = \boldsymbol{\mathcal{E}}_3^{\mathrm{T}} \boldsymbol{\mathcal{B}}^{\mathrm{T}} \boldsymbol{\mathcal{Q}}$.

Therefore, given the current state, references, affine dynamics model, and set of active constraints, we can compute and validate the optimal affine feedback law via (6.3) and (6.5) and thus construct a database of controllers simply by storing the appropriate sets of active constraints and reconstructing the control output online. This online reconstruction of the control and KKT matrices also ensures that the computed control response is locally optimal even as the system dynamics evolve.

6.1.2 Controller Search via Randomized Trajectories

Explicit MPC techniques employ formulations similar to those detailed in Sect. 6.1.1 to exhaustively enumerate all controllers for a given MPC formulation (i.e., all possible combinations of active constraints). However, this enumeration results in a controller database that is exponentially large in the number of constraints[1]. We therefore seek to construct a database containing only the controllers required by the system to traverse trajectories in the reachable set with time horizon T,

$$\mathcal{R}(\mathbf{x}_0, \boldsymbol{u}, T) = \left\{ \int_0^T f(\mathbf{x}_0, \boldsymbol{u}) dt \; \forall \; \boldsymbol{u} \in \mathcal{U} \right\}$$

[1] Assuming c constraints of the form $a_{\min} \leq a \leq a_{\max}$, there may be up to 3^{cN} controllers)[57].

where $f(\cdot)$ denotes the nonlinear dynamics model, $\mathcal{U}$ is the set of controls permitted by (6.1), and the trajectories are reasonably tracked by the system. As the control input at any point is of the form $\boldsymbol{u} = \mathbf{K}\boldsymbol{r} + \mathbf{k}_{\text{ff}}$, the reachable set under (6.1) is given by $\mathcal{R}_T = \mathcal{R}(\mathbf{x}_0, \mathbf{K}\boldsymbol{r} + \mathbf{k}_{\text{ff}}, T) \; \forall \; \boldsymbol{r} \in \mathcal{X}^N$, where $\mathcal{X}$ is the set of states permitted by (6.1). Therefore, we pursue a database that consists of local, affine controllers required to track trajectories in $\mathcal{R}_T$ by searching over feasible trajectories in $\mathcal{R}_T$.

Due to the high dimensionality of the search space, we choose to randomly sample trajectories in $\mathcal{R}_T$ using the Closed-loop Rapidly-exploring Random Tree (CL-RRT) algorithm [90]. CL-RRT grows a tree of dynamically feasible trajectories through $\mathcal{X}$ by randomly sampling points in the reference space of the feedback controller (e.g., $\boldsymbol{r} \in \mathcal{X}^N$ for (6.1)), finding the closest node in the current tree, and forward-simulating the closed-loop system dynamics from that node to track the reference [91]. Consequently, the resulting tree of trajectories can be interpreted as a sampling-based approximation of $\mathcal{R}_T$ [92]. As CL-RRT is probabilistically complete [93, 94], in the limit the tree of trajectories will be arbitrarily dense in any part of $\mathcal{R}_T$ with a diverse set of trajectories[2]. Moreover, if the system is differentially flat, the samples may be drawn from the lower-dimensional space of flat outputs, $\mathcal{S} \subset \mathcal{X}$, and used to compute smooth polynomial trajectories, $\mathbf{s}(t) \in \mathcal{S} \cap \mathcal{R}_T$, that provide full state references $\boldsymbol{r} = [\mathbf{s}(t_0), \dot{\mathbf{s}}(t_0), \ldots, \mathbf{s}^{(k)}(t_{N-1})]$ to the controller [91]. This formulation allows us to solve (6.1) at any point along any branch of the tree and generate a set of affine feedback controllers, as in Sect. 6.1.1. We can also assess if a given set of controllers is sufficient to cover a variety of relevant operating domains (i.e, at each simulation step of each branch).

[2]We intentionally do not use RRT* here as it includes an additional rewiring step that effectively homogenizes the branches in the tree to obtain asymptotic optimality. In contrast, standard RRT and CL-RRT enable branches to grow in arbitrary directions. This results in a greater diversity of trajectory segments and therefore improved coverage of $\mathcal{R}_T$.

6.1.3 Controller Database Generation via Sampling

We propose the following approach to generate a database, $\mathcal{M}$, of affine feedback controllers. As described in Alg. 6.1, the database is initialized to be empty, and we introduce a transition frequency matrix, $\mathbf{\Phi}$, that records the number of transitions between each pair of controllers added to $\mathcal{M}$. We initialize the search tree with a nominal state, $\mathbf{x}_0 \in \mathcal{X}$, and its corresponding flat outputs, $\mathbf{s}_0 \in \mathcal{S}$. For each random sample $\mathbf{s}'$ generated, we compute $\mathbf{s}(t)$ as detailed in Sect. 6.1.2. The system dynamics are forward-simulated along $\mathbf{s}(t)$ using EPC with the current database of controllers (the simulation step size, Δt, matches the desired control rate) and $\mathbf{\Phi}$ is incremented accordingly. If the appropriate controller is not found in the database, EPC solves the QP (6.1) and adds the resulting controller to the database (increasing the size of $\mathbf{\Phi}$ as well).

The forward-simulation terminates upon reaching a predefined maximum segment duration, T, and a new node is added to the tree containing the end state and index of the final controller used. Following Reist, et al. [54], we terminate growing of the tree after we have generated M consecutive samples with resulting trajectories covered by the controllers in $\mathcal{M}$, leading to no change in $\mathcal{M}$. After the database is complete, we can define an order-1 Markov chain that represents the transitions between controllers, with empirical transition probabilities defined by $\bar{\mathbf{\Phi}} = \mathbf{\Phi}$ with normalized rows. Sorting the outgoing transitions from each state of the Markov chain according to probability of occurrence yields a (strict) partial ordering, $\mathbf{\Omega}$, of the controllers. This ordering informs the online query process detailed in Sect. 6.1.4.

The proposed Markov chain representation enables two additional simplifications of the controller database. The first is model reduction via elimination of low-occupancy states to reduce storage requirements and query times, which is essential for deployment on systems with severe computational constraints. This is achieved by retaining only the K most frequently visited states defined by the K columns of $\bar{\mathbf{\Phi}}$ with the highest sums, or via more advanced approaches such as conservation of transition rates [95]. The second simplification reduces the number of transitions out of each state so as to limit the query time per control iteration by retaining the K highest

Algorithm 6.1 Controller Database Generation

$\mathcal{M} \leftarrow \emptyset$, $\mathbf{\Phi} \leftarrow \mathbf{0}$, $k \leftarrow 0$
Add root node $n_0 = (\mathbf{x}_0, \mathbf{s}_0)$ to tree $\mathcal{T}$
while $k < M$ **do**
    Generate sample $\mathbf{s}' \in \mathcal{S}$
    Select parent node $n^* = \text{argmin}_{n \in \mathcal{T}} \|\mathbf{s}_n - \mathbf{s}'\|$
    Compute spline $\mathbf{s}(t)$ from $\mathbf{s}_{n^*}$ to $\mathbf{s}'$
    $t_{\text{sim}} \leftarrow 0$, $\mathbf{x}_{\text{sim}} \leftarrow \mathbf{x}_{n^*}$, $j \leftarrow i_{n^*}$
    while $t_{\text{sim}} < T$ **do**
        Get $\boldsymbol{r}$ from $\mathbf{s}(t)$
        for each element $\mathbf{m}_i \in \mathcal{M}$ **do**
            if $\mathbf{x}_{\text{sim}}, \boldsymbol{r}$ satisfy KKT criteria (6.2) **then**
                $\boldsymbol{u} \leftarrow \boldsymbol{\kappa}_i(\mathbf{x}_{\text{sim}}, \boldsymbol{r})$, `controller_found` $\leftarrow$ true
                $\mathbf{\Phi}_{ij} \leftarrow \mathbf{\Phi}_{ij} + 1$, $j \leftarrow i$, $k \leftarrow k + 1$
                break
            end if
        end for
        if `controller_found` is false **then**
            Solve QP (6.1) to generate new controller, $\boldsymbol{\kappa}_{\text{new}} = (\mathbf{K}, \mathbf{k}_{\text{ff}})$
            Add new element $\mathbf{m}_{\text{new}}$ containing $\boldsymbol{\kappa}_{\text{new}}$ to $\mathcal{M}$
            $\boldsymbol{u} \leftarrow \boldsymbol{\kappa}_{\text{new}}(\mathbf{x}_{\text{sim}}, \boldsymbol{r})$, $j \leftarrow |\mathcal{M}|$, $k \leftarrow 0$
        end if
        $t_{\text{sim}} \leftarrow t_{\text{sim}} + \Delta t_{\text{sim}}$, $\mathbf{x}_{\text{sim}} \leftarrow \mathbf{x}_{\text{sim}} + \int_0^{\Delta t_{\text{sim}}} f(\mathbf{x}, \mathbf{u}) d\tau$
    end while
    Add new node $n = (\mathbf{x}_{\text{sim}}, \mathbf{s}_{\text{sim}}, j)$ to $\mathcal{T}$
end while
Compute ordering $\mathbf{\Omega}$ based on transition frequencies in $\mathbf{\Phi}$

probability transitions out of each state such that the sum of the retained transitions satisfies a threshold, ρ, or by leveraging more advanced techniques, such as entropy-based methods [96]. Combinations of these two simplifications may also be applied, by preserving a limited set of states $\mathcal{M}'$ such that $\bar{\mathbf{\Phi}}_{ij} \leq \epsilon \; \forall \; \mathbf{m}_i \in \mathcal{M}', \mathbf{m}_j \in \mathcal{M} \cap \mathcal{M}'$ over the next N queries.

6.1.4 Online Database Query

Once the controller database, $\mathcal{M}$, is populated, it enables high-rate, online control that recovers the functionality of (6.1). As described in Alg. 6.2, we query $\mathcal{M}$ in each control iteration to identify the appropriate controller $\boldsymbol{\kappa}_i$. However, as each control iteration represents a state transition

Algorithm 6.2 Controller Database Query

Inputs: $\mathcal{M}$ and $\boldsymbol{\Omega}$ from Alg. 6.1, Current $\mathbf{x}, \boldsymbol{r}$, Previous controller index j^*
for each element $\mathbf{m}_i \in \mathcal{M}$ ordered by Ω_{j^*} **do**
    if $\mathbf{x}, \boldsymbol{r}$ satisfy KKT criteria (6.2) **then**
        return $\boldsymbol{u} \leftarrow \boldsymbol{\kappa}_i(\mathbf{x}, \boldsymbol{r})$
    end if
end for
return $\boldsymbol{u} \leftarrow$ safety controller

in the Markov chain (including self-loops), we identify the next transition by iterating through $\mathcal{M}$ according to the order specified by entry j^* in $\boldsymbol{\Omega}$. This ordering aims to reduce the number of controllers that must be evaluated (since evaluating the KKT conditions is more expensive than a simple lookup table query), thereby improving the query efficiency relative to orderings based on measures of controller utility.

Although the termination condition in Alg. 6.1 ensures a high probability of coverage, it cannot guarantee perfect coverage. Therefore, if no controller in the database is suitable, we apply a safety controller (e.g., a short horizon MPC with soft constraints [20] or any other globally stabilizing controller) until the system returns to the region covered by $\mathcal{M}$.

Finally, the controllers applied online retain the adaptive properties of EPC presented in Ch. 4. As $\boldsymbol{\kappa}$ is parameterized by the system dynamics (Sect. 6.1.1), it can be combined with an online model learner for online adaptation to changing plant dynamics undergoing exogenous disturbances.

6.2 Simulation Results

To demonstrate the performance of the proposed explicit adaptive NMPC algorithm, we seek to demonstrate the following results through a series of simulation trials: **R6.1**: severely reduced controller database size, **R6.2**: improved computational efficiency via the Markov chain-based ordering and simplification, and **R6.3**: database coverage of controllers required at run-time. We also leverage these properties to demonstrate the following results from the set enumerated in

Chapter 1: **R1**: stable control performance, **R2**: constraint satisfaction, **R3**: real-time computation of control commands, and **R4.1**: adaptation performance. We consider two applications: 1) pendulum control to demonstrate basic functionality of the proposed approach and 2) quadrotor attitude control to demonstrate the value of Markov chain simplification to achieve the control rates required for stability. The offline computed controller database for each application is evaluated through a series of real-time simulations that require the system to track a variety of random, smooth trajectories.

6.2.1 Pendulum Control

The first system is a simple pendulum modeled by two states (angle and angular velocity) and one control input (torque). The corresponding MPC formulation has a 10-step prediction horizon and enforces constraints on angular velocity and torque.

Database Computation

To generate the controller database offline, we apply Alg. 6.1 with a termination threshold, M, of 50000 consecutive queries, and as Fig. 6.1 illustrates, the database satisfies this threshold after computing 268 controllers. This number is in stark contrast to the $3^{20} \approx 3.49 \times 10^9$ possible controllers that a standard explicit MPC would aim to enumerate (**R6.1**). Figure 6.2 shows the Markov chain transition probabilities derived from the search process. Table 6.1 summarizes the database generation results.

Database Query Performance

To evaluate Alg. 6.2 with this controller database, the pendulum is commanded to track a series of random, smooth trajectories over the course of a 600 s trial. As Table 6.2 shows, the limited number of controllers in the database permit tracking a variety of trajectories with negligible use of the safety controller. This demonstrates that the proposed reachable-space search does

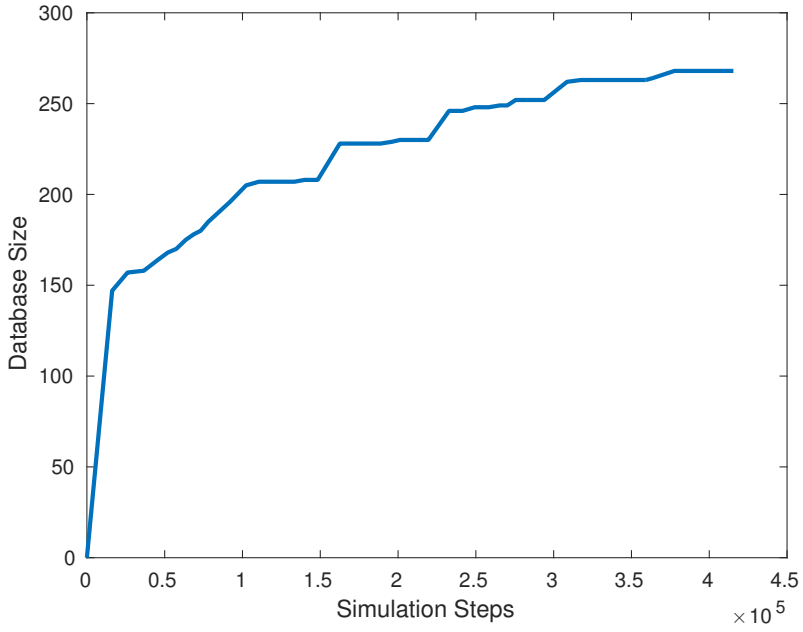


Figure 6.1: Growth of the pendulum controller database during the offline search.

cover the space of control scenarios that the system my encounter during operation (**R6.3**). Additionally, repeating the same trial with a simple search ordering (as employed by EPC) yields substantially longer query times than the proposed ordering based on Markov chain transition probabilities (**R6.2**). Figures 6.3 and Fig. 6.4 illustrate the control performance provided by this database in terms of trajectory tracking error (**R4**) and constraint satisfaction (**R2**).

Table 6.1: Statistics for the pendulum control database computation

Tree Branches	Simulation Steps	Control Database Size	Explicit MPC Bound
560	415517	**268**	3^{20}

Table 6.2: Statistics for the 600 s pendulum database evaluation

	Queries	Coverage	Mean Query Time	Safety Control Calls
Markov chain ordering	119165	**99.987%**	**0.1316 ms**	15
Simple ordering	119373	99.975%	0.7778 ms	30

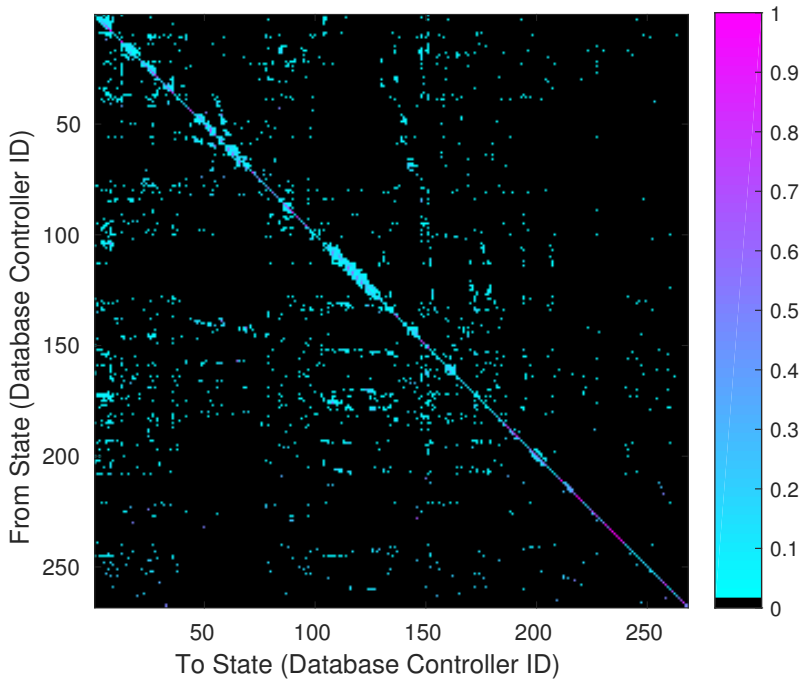


Figure 6.2: Markov chain transition probabilities with states that correspond to the relevant database controller enumeration for the pendulum feedback control system

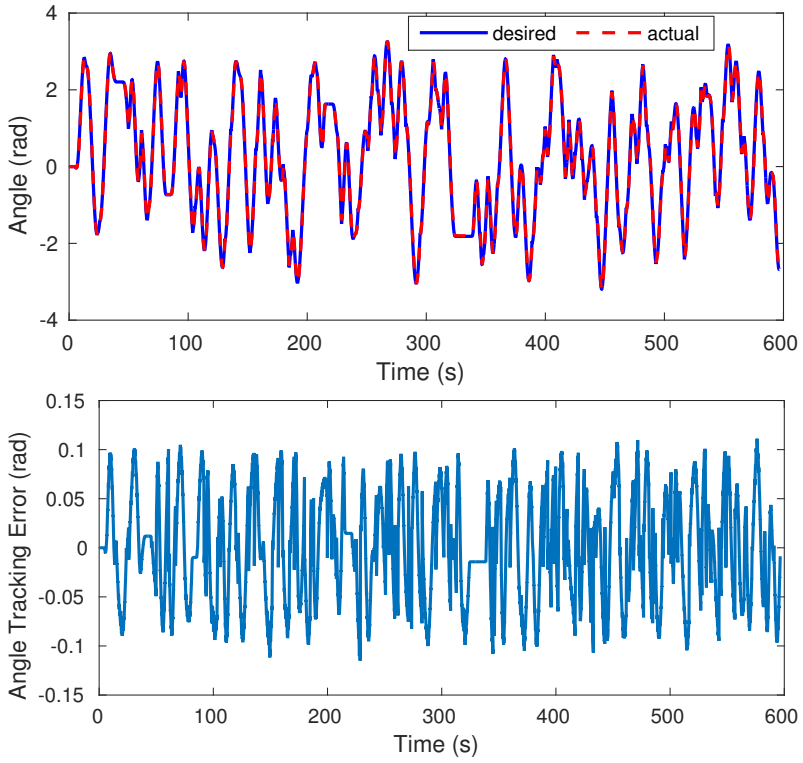


Figure 6.3: Trajectory tracking performance via the pendulum control database

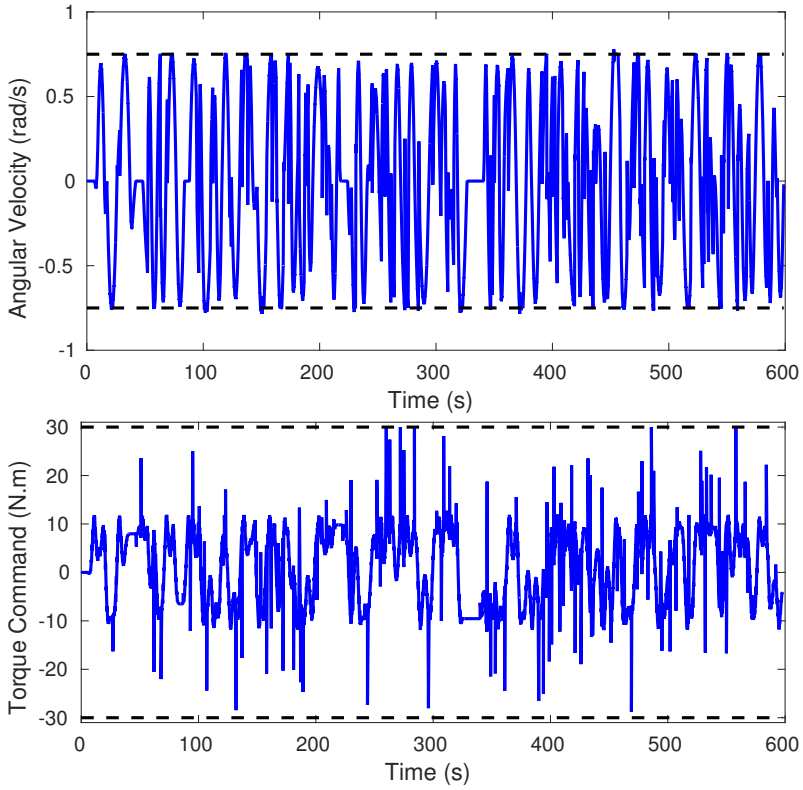


Figure 6.4: State and input constraint satisfaction via the pendulum control database

6.2.2 Quadrotor Attitude Control

The second application we consider is attitude control of a small quadrotor micro air vehicle simulated via the architecture described in Appendix A.1. Quadrotors are highly dynamic systems that require attitude controllers that run at high feedback rates (e.g., 200 Hz), and as a result, are sensitive to computational delays in the controller. As described in Appendix A.3.1, the attitude dynamics are modeled by six states (attitude and angular velocities) and three control inputs (torque about each axis). The corresponding MPC formulation has a 15-step prediction horizon and enforces constraints on roll, pitch, and the three torques.

Database Computation

As with the pendulum, we first apply Alg. 6.1 with a termination threshold, M, of 50000 consecutive queries. As Fig. 6.6 illustrates, the database satisfies this threshold after computing 570 controllers. This number is in stark contrast to the $3^{75} \approx 6.08 \times 10^{35}$ possible controllers that a standard explicit MPC would aim to enumerate (**R6.1**). Figure 6.7 shows the Markov chain transition probabilities derived from the search process, and Table 6.3 summarizes the database generation results.

Database Query Performance

To evaluate online control using the database, the quadrotor is commanded to track a series of random, smooth trajectories for a 600 s trial (commanded trajectories are in position and yaw but contain sufficiently aggressive sections to excite the attitude dynamics, and for simplicity, the translational dynamics are controlled via LQR). We consider both the search order based on Markov chain transition probabilities and a simple search order as employed by EPC [84]. During the first 250 s of the trial, the proposed Markov chain-based ordering strategy yields faster solution times than the simple ordering (**R6.2**), as shown in Table 6.4. Furthermore, there are no calls to the safety controller in this time, demonstrating that the proposed reachable-space search

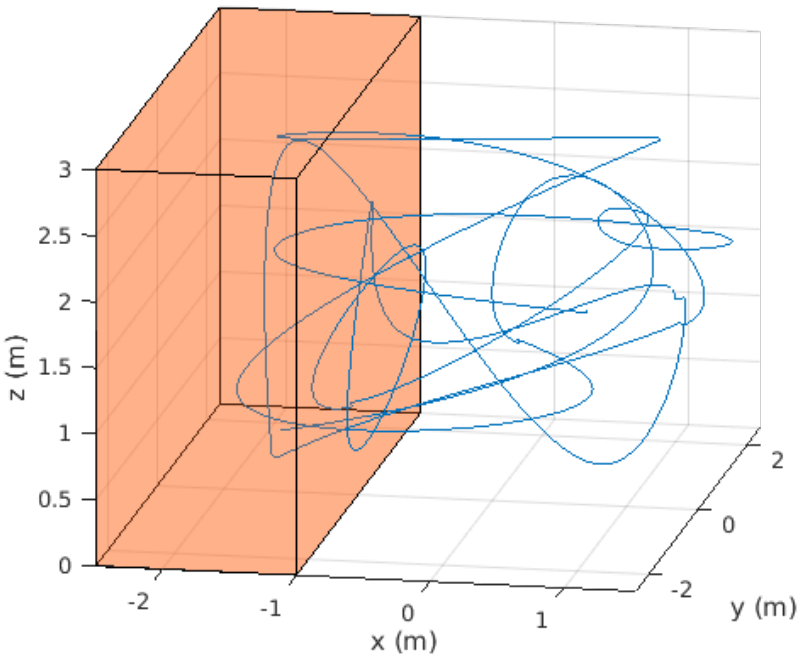


Figure 6.5: Disturbance region highlighted in orange with example trajectories.

does cover the space of control scenarios that the system may encounter during operation (**R6.3**).

However, we also observe that the database is still too large for reliable, realtime operation as an extended query causes the quadrotor to crash partway through the trial with either search ordering. This failure stems from an aggressive trajectory commanded approximately 250 s into the simulation that requires a controller not found in the database. Both ordering strategies are unable to detect this omission without searching the entire database, resulting in a delay of over 100 ms that causes the failure.

Database Simplification

We therefore simplify the Markov chain by preserving the highest-probability outgoing edges for each state such that they represent 99% of transitions in the original model. This simplification allows queries to terminate after exploring the high-probability options, resulting in successful completion of the 600 s trial (**R1**) with millisecond query times that enable high-rate attitude control (**R3**) and negligible safety controller usage (**R6.3**), as shown in Table 6.4. For comparison,

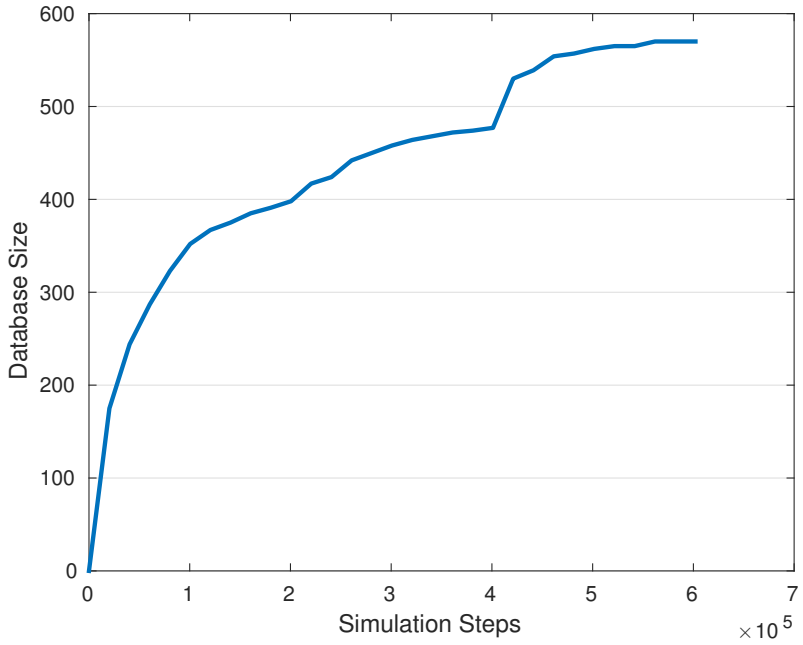


Figure 6.6: Growth of the attitude controller database during the offline search.

an equivalent QP solution would be nearly an order of magnitude slower [21]. Figure 6.8 illustrates the attitude reference tracking error for this trial, and Fig. 6.9 verifies that the torque input constraints are satisfied for the entire 600 s (the roll and pitch constraints are rarely activated as the input constraints dominate) (**R2**). These results demonstrate that the proposed approach for generating an explicit NMPC database enables real-time constrained control of systems with fast timescales.

Table 6.3: Statistics for the quadrotor attitude control database computation

Tree Branches	Simulation Steps	Database Size	Explicit MPC Bound
3000	602060	**570**	3^{75}

Table 6.4: Statistics for the 600 s quadrotor attitude control database evaluation. Entries in red indicate failures due to a prolonged database search before applying the safety controller.

	Number of Queries	Percent Coverage	Mean Loop Time	Safety Control Calls
EPC ordering	49166	100.00%	1.49 ms	Failed on first call
MC ordering	49074	100.00%	1.24 ms	Failed on first call
MC simplification	118956	**99.97%**	**1.45 ms**	34

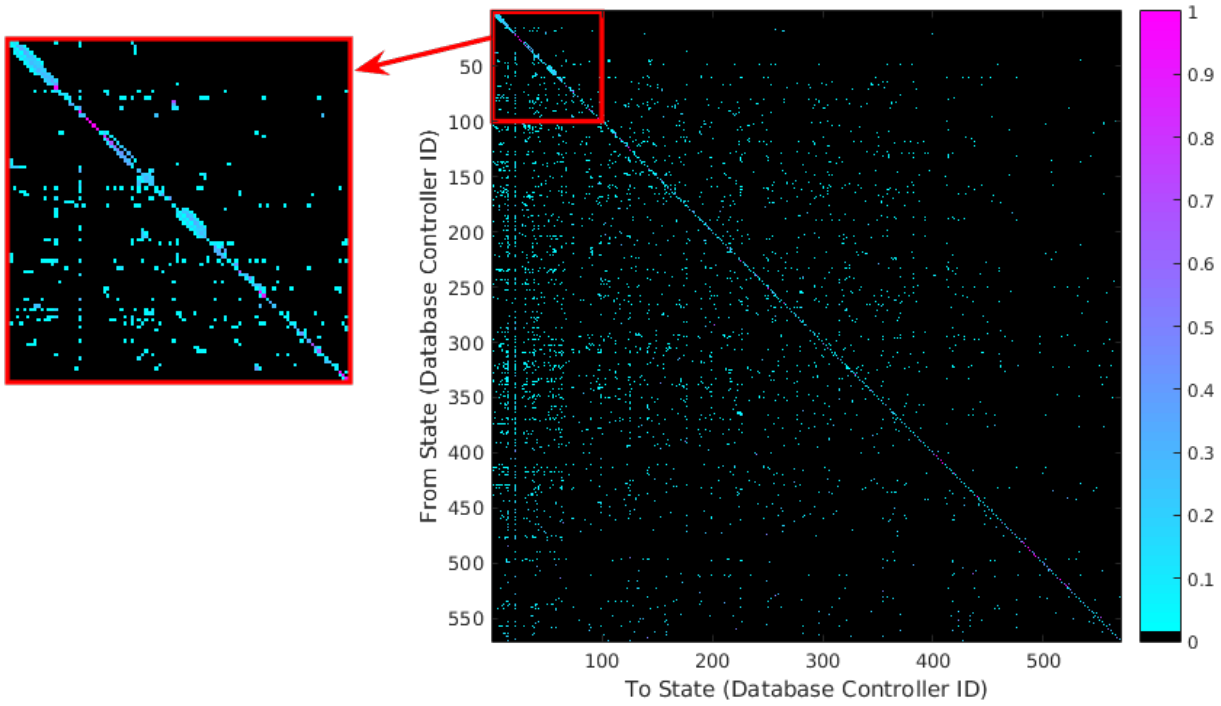
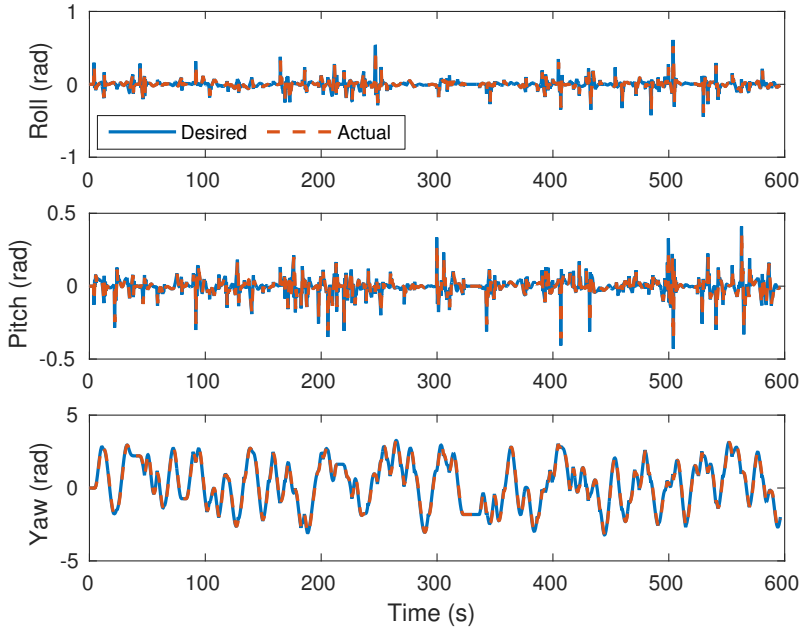


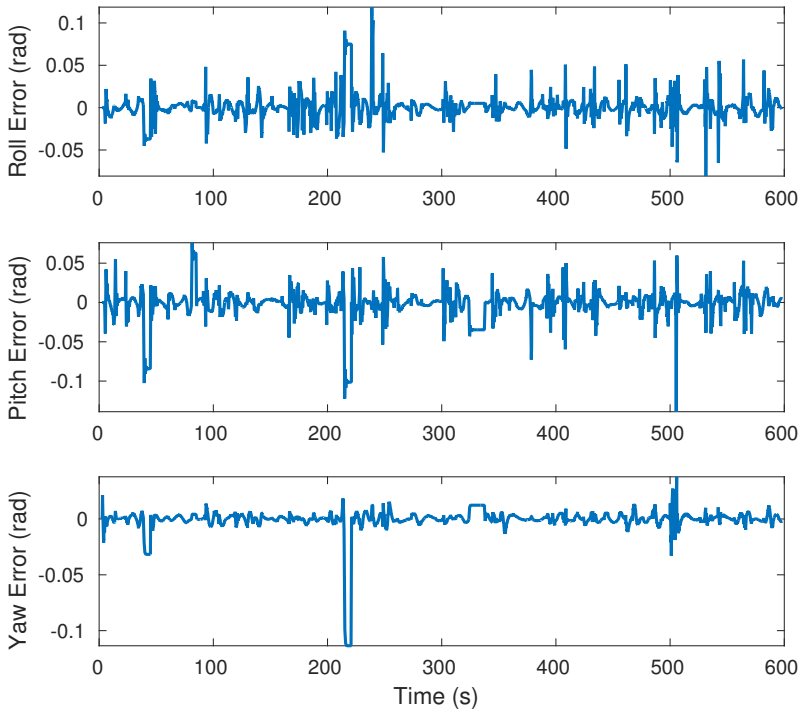
Figure 6.7: Transition probabilities for the Markov chain with states corresponding to controllers in the database. The entries for the first 100 controllers are magnified for clarity.

Adaptation Performance

Finally, the as controllers in the database are parameterized by the system dynamics, we introduce the online model learner used in EPC to enable online adaptation of the control laws. A constant torque disturbance is applied to the vehicle in the region highlighted in Fig. 6.5. The magnitude of this exogenous torque is approximately 30% of the roll and pitch command limits enforced in the MPC formulation. Despite this reduced control authority, the commanded torques never exceed the constraints (**R2.1**), as shown in Fig. 6.10, and combining the previously computed controller database with an online-updated estimate of the dynamics yields a 35% improvement in roll and pitch tracking error in this region. As Table 6.5 also shows, there is a 25% improvement in the resulting position tracking error. Thus, the improvement due to the proposed approach is comparable to the performance obtained by directly applying EPC [84] but does not require the computational resources to perform online optimization (**R4.1**).



(a) Attitude tracking comparison



(b) Attitude tracking error

Figure 6.8: Quadrotor attitude reference tracking performance via the control database.

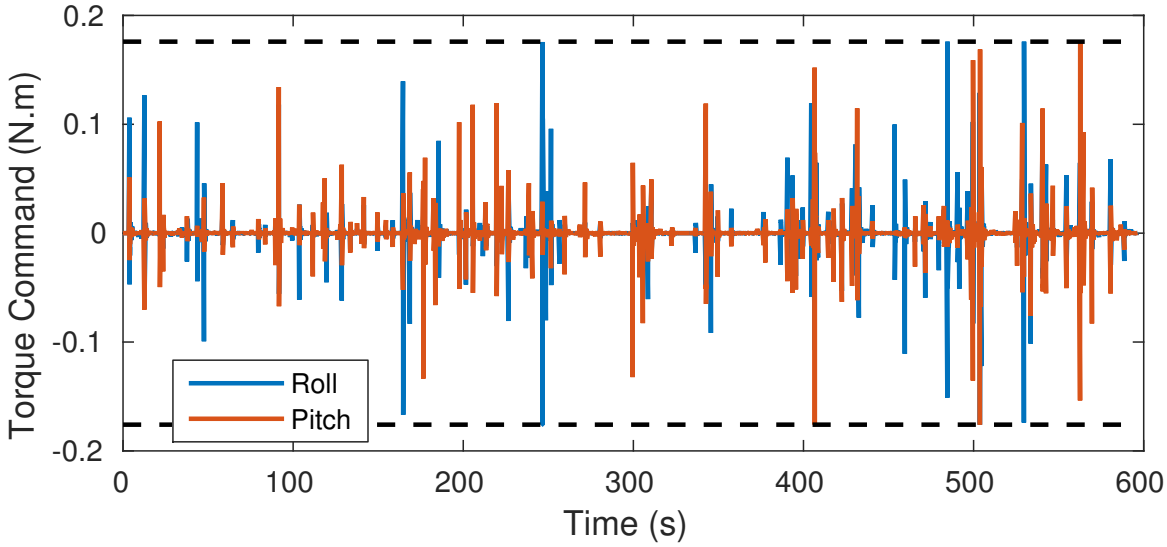


Figure 6.9: Roll and pitch torque constraints are satisfied for the duration of the attitude control evaluation.

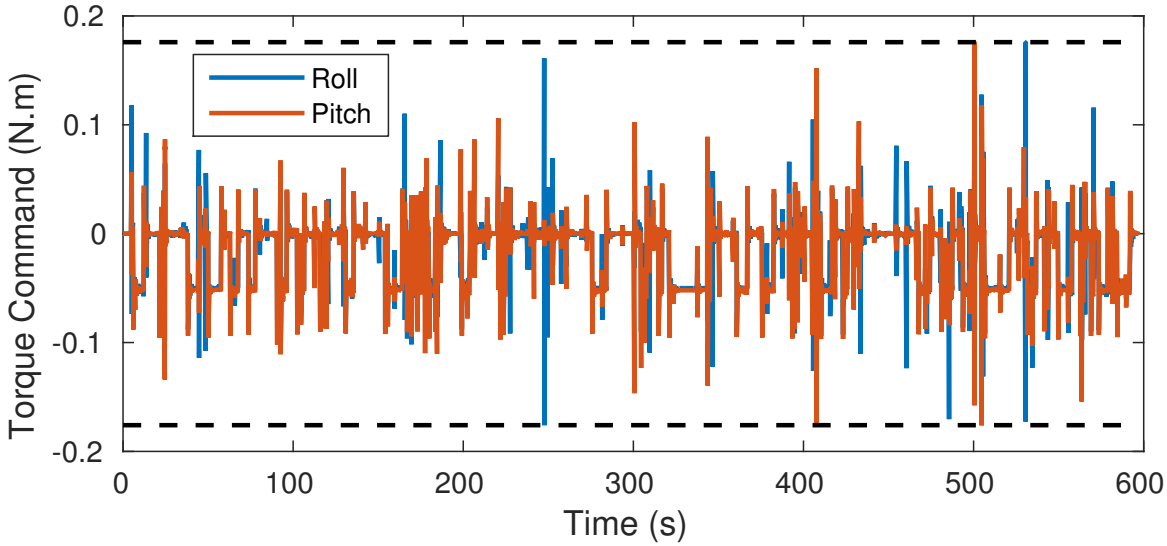


Figure 6.10: Roll and pitch torque commands satisfy constraints even in the presence of a 30% max torque disturbance.

Table 6.5: Comparison of mean tracking errors with and without adaptation

	Roll (rad)	Pitch (rad)	x-Position (m)	y-Position (m)
No adaptation	0.0908	0.0909	0.0400	0.0360
With adaptation	0.0597	0.0585	0.0299	0.0271

6.3 Experimental Evaluation

To assess the real-time control performance of the proposed efficient explicit formulation, we implement the controller database onboard a Crazyflie quadrotor, a 32 g, severely compute-constrained platform with a 168 MHz processor and 192 KB of SRAM (see Appendix A.3.1 for details). In this section, we consider the problem of controlling the translational dynamics (outer loop) of the quadrotor, rather than the attitude dynamics as in Sect. 6.2.2. The database for the outer loop is computed offline via Alg. 6.1 and applied online via Alg. 6.2 at a rate of 100 Hz. Following a standard cascaded control formulation [81], we apply a PD control law for the inner loop to track the outer loop outputs.

We generate the controller database with a horizon length of $N = 10$ control iterations to maintain tractability and with constraints on x-velocity, y-velocity, and the three control inputs. This yields a database with only 191 entries, as opposed to the $3^{50} \approx 7.18 \times 10^{23}$ total possible controllers (**R6.1**). Figure 6.11 illustrates the connectivity in the corresponding Markov chain before and after simplification. As the database only requires the active set and the simplified list of successors for each controller, the total memory required to store the database is only 3.9 KB. However, to reduce redundant online matrix operations, we precompute and cache some of the EPC matrices, raising the total memory usage to 8.8 KB.

To evaluate online control performance, the vehicle is commanded to track a linear trajectory, as shown in Fig. 6.12. Figure 6.13 depicts a set of transitions between controllers in the database as the vehicle executes this trajectory, and Fig. 6.14 shows the corresponding query times for the controller database. This trial demonstrates that the proposed approach is computationally tractable on a severely constrained platform and yields stable control performance (**R1**). Moreover, the total position control loop time, including the database query step, achieves a mean of 3.956 ms with a sufficiently low standard deviation to reliably achieve a 100 Hz update rate (**R3**). Table 6.6 summarizes the online control performance.

This realtime control performance stems from the Markov chain simplification. Each con-

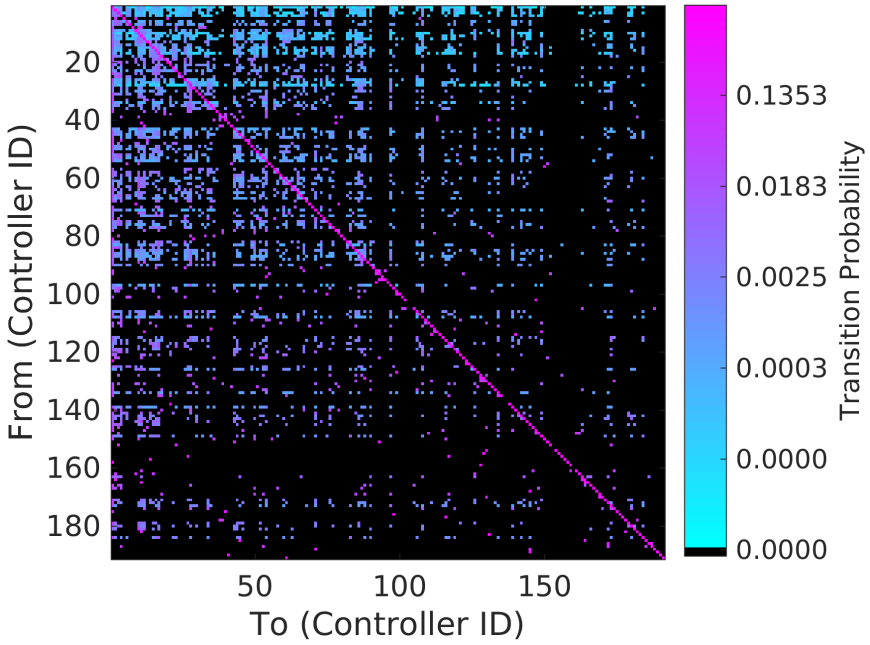
troller in the database has on average only 9 successors to evaluate (**R6.2**). Without the simplification, each controller would have 191 successors to evaluate and could cause the controller to block for nearly 100 ms, destabilizing the vehicle. Finally, even after the simplification, we observe minimal application of the safety controller (Table 6.6). As a result, the database yields 97% coverage of the set of controllers required to track this trajectory (**R6.3**).

Table 6.6: Statistics for the Crazyflie evaluation trial.

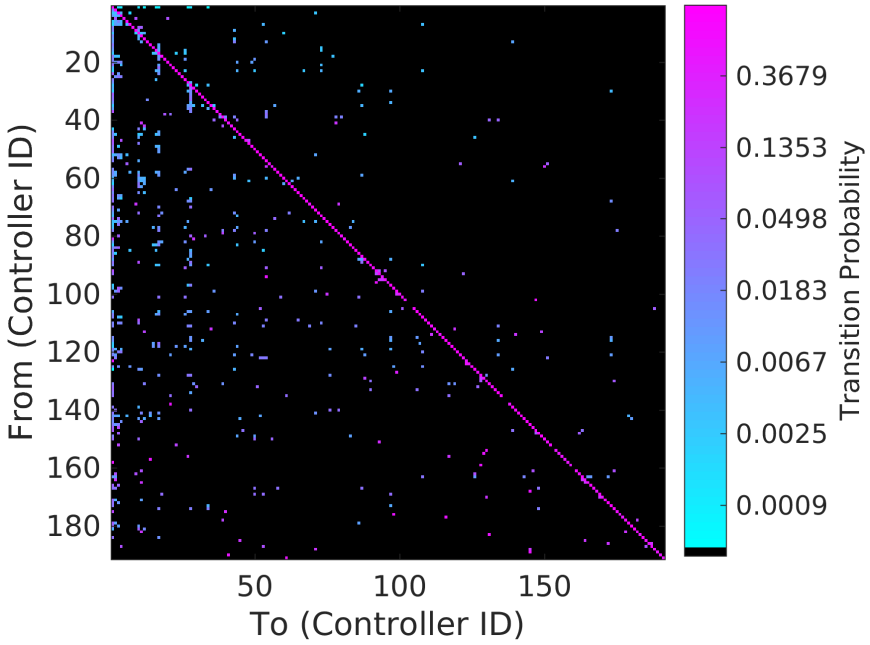
Num. Queries	Coverage	Query Time (Mean)	Query Time (Std.Dev)	Safety Control Calls
8268	**97.443%**	**3.956 ms**	**5.212 ms**	217

6.4 Conclusions

In this chapter, we have presented a technique for constructing an explicit NMPC controller database that leverages reachable-space search to identify a limited set of controllers that cover nearly all potential operating domains. Transitions between controllers in the resulting database are modeled via a Markov chain that facilitates further simplification to enable high-rate constrained control. Furthermore, since the controllers in the database are parameterized by the system dynamics, they can be used in conjunction with an online model learner to enable constrained adaptive control. A set of simulation trials demonstrate the offline and online performance of the proposed approach applied to a simple pendulum and the high-rate, constrained, adaptive nonlinear control problem of quadrotor attitude control. Additionally, we demonstrate that this approach enables real-time predictive control on a nano quadrotor with severe computational constraints. Finally, although we have presented this explicit NMPC approach as an application of EPC (Ch. 4), it is readily extended to incorporate Robust EPC (Ch. 5) as well.



(a)



(b)

Figure 6.11: Transition probability matrices (a) before and (b) after simplification of the Markov chain underlying the position controller in the Crazyflie flight experiments.

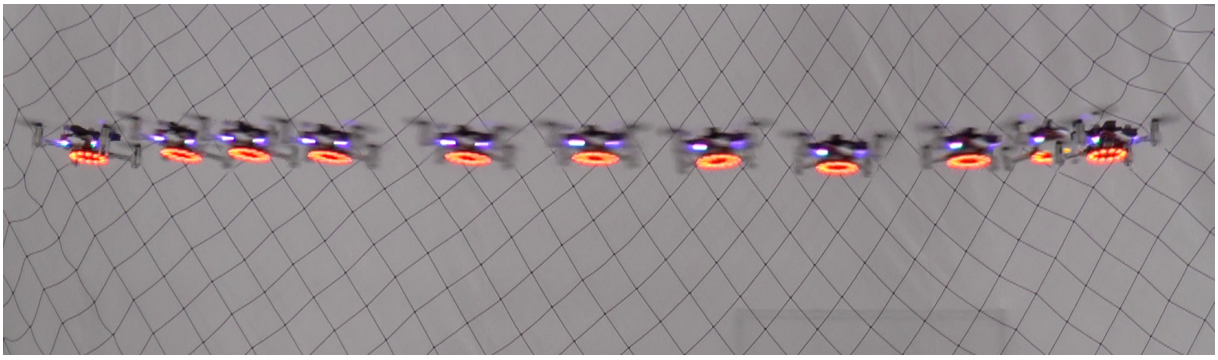


Figure 6.12: Snapshots of the Crazyflie tracking one lap of the linear trajectory used to evaluate real-time control feasibility

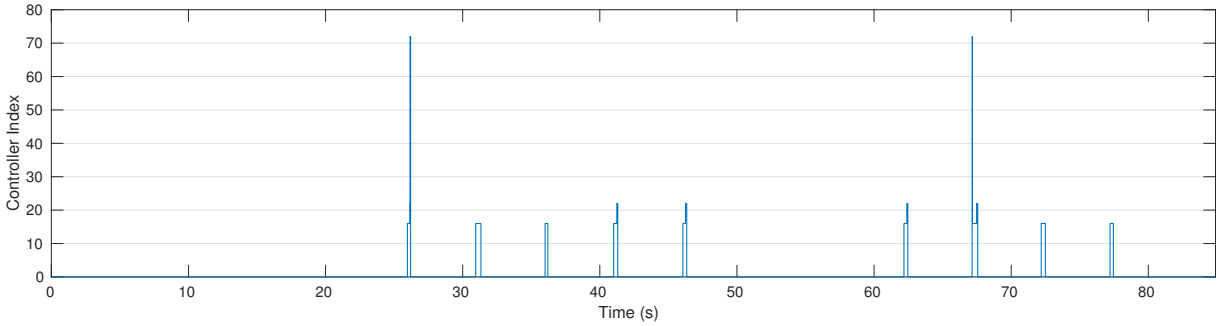


Figure 6.13: The Crazyflie transitions between multiple controllers while executing the linear trajectory.

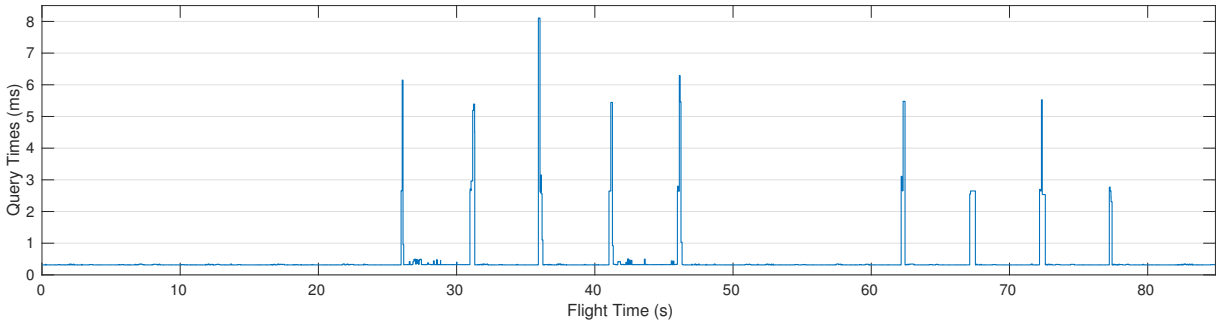


Figure 6.14: Controller database query times onboard the Crazyflie where even the spikes corresponding to the controller changes are below the 10 ms desired threshold.

Chapter 7

Conclusion

This thesis addresses the problem of achieving safe, accurate, and computationally efficient feedback control of constrained nonlinear systems with state and plant model uncertainty. Nonlinear Model Predictive Control (NMPC) provides a natural framework to address this class of control problems, but it is too computationally intensive to apply to small, agile autonomous systems with size, weight, and power constraints. We therefore introduce the idea of experience-driven control to enable the system to improve performance over time by leveraging experience in the form of learned dynamics models and control laws. Consequently, we propose a series of feedback control techniques that

1. Ensure safety via a constrained robust-adaptive NMPC formulation
2. Improve trajectory tracking accuracy via semi-parametric learning of changes in the dynamics model due to modeling error or exogenous perturbations
3. Improve computational efficiency by leveraging past experiences to compute and store locally optimal feedback control laws that can be reused as appropriate

7.1 Summary of Contributions

Chapters 3 - 6 present a set of control methodologies that leverage experience to achieve these objectives. Thus, the four key contributions of this thesis are:

- **Computationally efficient NMPC via Nonlinear Partial Enumeration:** NPE solves the NMPC problem by constructing online a database of controllers that locally recover the performance of nonlinear optimization-based NMPC. The algorithm reuses these controllers as appropriate in future control iterations, thus reducing its dependence on online optimization. As a result, NPE achieves the millisecond solution times requires for high-rate NMPC on computationally constrained systems that are unable to solve nonlinear programs in real-time (Ch. 3).
- **Adaptation to plant model uncertainty via Experience-driven Predictive Control:** EPC retains the millisecond solution times from NPE but eliminates the need to solve nonlinear programs by introducing online dynamics model learning. We define a notion of experience for predictive control as the combination of an online learned dynamics model that reflects observed system evolution and a parameterized controller database that captures past locally-optimal control laws. The controller automatically enhances tracking performance via adaptation to nonlinearities and unmodeled dynamics as well as unknown external disturbances acting on the system. (Ch. 4).
- **Reliable constraint satisfaction with state uncertainty via Robust EPC:** Robust EPC extends the previous fast, adaptive formulation to account for time-varying state uncertainty and yield robust constraint satisfaction. We employ a chance-constrained formulation to tighten constraints according to the covariance from the state estimator and extend the parameterization of the controllers in the database to include uncertainty bounds. As a result, Robust EPC enables integration with non-idealized state estimation systems and safe operation in scenarios where varying state uncertainty may otherwise cause constraint violations (Ch. 5).
- **Tractable NMPC on severely limited systems via Efficient Explicit Adaptive NMPC:** This

efficient solution strategy for the explicit NMPC formulation applies EPC to construct a controller database offline based on realistic synthetic experiences. These synthetic experiences are generated via simulations of a dynamically-feasible, randomized search tree that explores the reachable space of the closed-loop system controlled by EPC. The resulting controller database avoids the exponential growth of other explicit formulations but maintains coverage of the set of controllers required during online operation. The empirical transition probabilities for the database entries define a Markov chain and a partial ordering on transitions. By applying Markov chain simplification strategies, we obtain a controller database that retains coverage while enabling the use of NMPC on systems with severe computational constraints (Ch. 6).

7.2 Future Work

There are numerous avenues for future research that build upon the approaches presented in this thesis. A primary focus of this work is the mitigation of various sources of uncertainty, including unmodeled dynamics, external disturbances, and imperfect state estimation. However, there are additional sources of uncertainty that could be addressed. For example, while model adaptation mitigates the effects of low frequency disturbances, this adaptation is not instantaneous. As a result, the transient behavior of the model learner introduces additional uncertainty into the predicted system evolution, and this uncertainty could potentially cause constraint violations such as those in Sect. 5.2.2. The Robust EPC framework can capture this uncertainty via the process noise model, but additional care must be taken to ensure that the model learner covariance or confidence estimates evolve in a predictable manner. Otherwise, a sudden, unexpected change in the uncertainty bound could again lead to constraint violations due to rapid changes in the tightened constraint bounds.

Appendix A

Experimental Architecture and Platforms

A.1 Software Architecture

We have developed a C++ ROS-based [97] software architecture to enable rapid development, simulation testing, and experimental validation of different control techniques, including those developed in this thesis. Figure A.1 illustrates this modular framework in which controllers are implemented as dynamically loaded plug-ins. This greatly facilitates changing between different control methodologies to compare their performance on a given scenario. The architecture also includes a state machine and a modular trajectory generation component to ensure safe and repeatable testing. For the majority of the results presented in this thesis, we use trajectory generator that publishes a polynomial spline fit to a predefined set of waypoints [98].

The controllers developed in this thesis are implemented as a C++ library templated on the system dynamics model. This further illustrates the platform-independent nature of these algorithms and simplifies the process of applying them to a new system or subsystem. All controller implementations use the qpOASES library to solve quadratic programs via the online active set strategy [99]. NPE (Ch. 3) also uses the NLopt library to solve nonlinear programs via an augmented Lagrangian approach and the Subplex algorithm [100]. It also uses Boost odeint [101] for numerical integration of the nonlinear dynamics constraint. With the exception of the experimen-

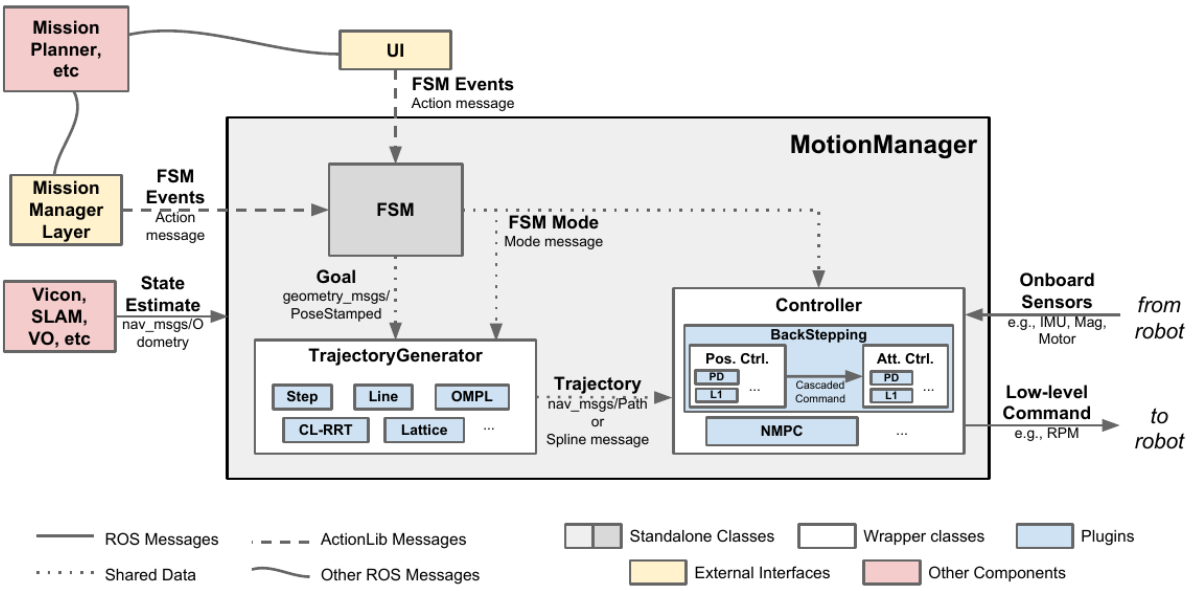


Figure A.1: Block diagram of the modular planning and control architecture that enables simulation and experimental evaluation of the ideas proposed in this thesis

tal results in Ch. 6, all linear algebra operations are implemented via the Armadillo library [102] integrated with the optimized linear algebra routines in OpenBLAS [103].

The state inputs and control outputs from this architecture are either connected to a custom simulator that employs a high-fidelity, nonlinear model of the vehicle dynamics, or they are connected to the appropriate hardware interface nodes. This enables seamless transitions from full-software simulations to hardware-in-the-loop simulations to experimental testing.

A.2 Infrastructure

We conduct experimental studies in the indoor flight arena shown in Fig. A.2. To facilitate rapid development, evaluation, and analysis of the control algorithms proposed in this thesis, we obtain accurate state feedback at 100-200 Hz from a Vicon motion capture system installed in the arena. This nominal accuracy also permits arbitrary degradation to emulate other sources of odometry, as in Ch. 5. The flight arena is also equipped with a set of eight high-power fans (a subset of which are shown in Fig. A.3) capable of generating wind speeds of up to 6 m/s [5]. As these fans

Figure A.2: Flight arena used to experimentally validate the proposed algorithms. The arena is equipped with a Vicon motion capture that obtains accurate, high-rate state feedback.

can be arranged arbitrarily around the flight arena, they are able to produce turbulent flow in a variety of directions that reflects the variability observed in windy outdoor settings.

A.3 Evaluation Platforms

A.3.1 Quadrotor Micro Air Vehicle

The dynamics of a quadrotor can be modeled as a 12 dimensional nonlinear system whose state $\mathbf{x} = \begin{bmatrix} \mathbf{p}^{\mathrm{T}} & \mathbf{v}^{\mathrm{T}} & \boldsymbol{\xi}^{\mathrm{T}} & \boldsymbol{\omega}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$ consists of position ($\mathbf{p}$), velocity ($\mathbf{v}$), attitude ($\boldsymbol{\xi} = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^{\mathrm{T}}$), and angular velocity ($\boldsymbol{\omega}$). Attitude is represented by roll (ϕ), pitch (θ), and yaw (ψ) angles following the ZYX convention. The control input $\mathbf{u} = \begin{bmatrix} F & \boldsymbol{\tau}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$ consists of thrust along the $+z$ body axis (F) and torques about each of the 3 body axes ($\boldsymbol{\tau} = \begin{bmatrix} \tau_\phi & \tau_\theta & \tau_\psi \end{bmatrix}^{\mathrm{T}}$). The system's time

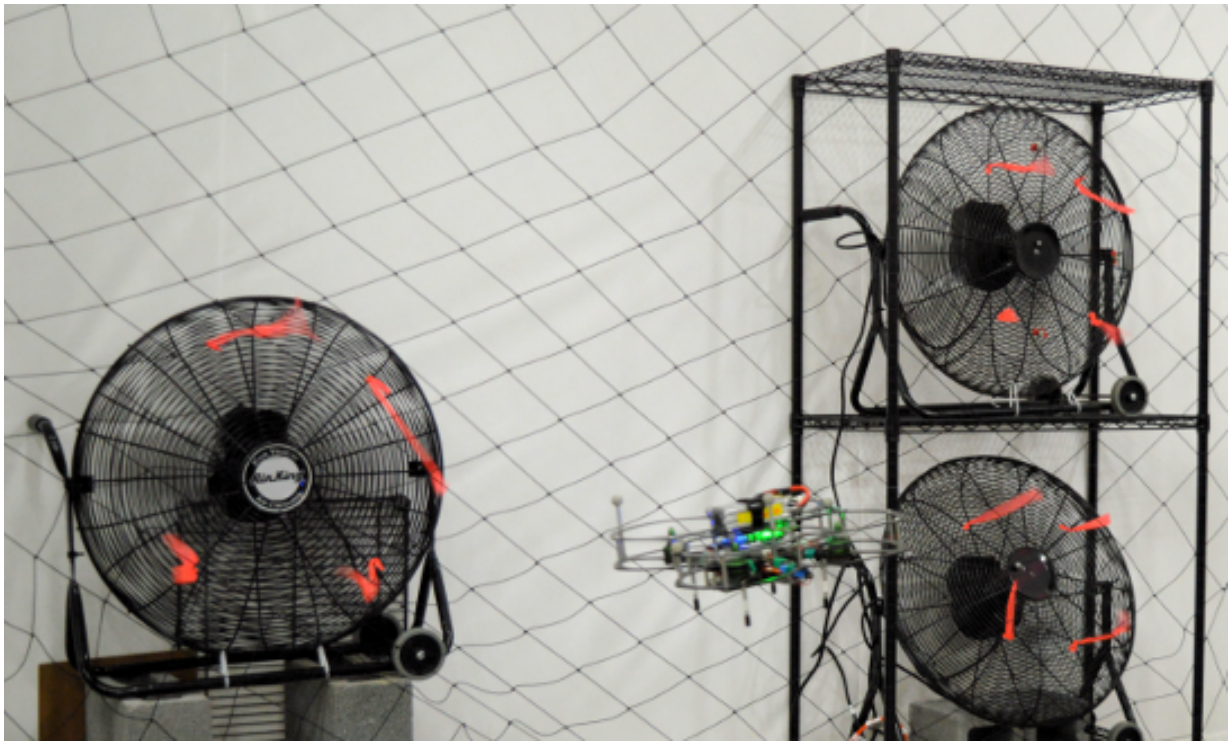


Figure A.3: Fans in the flight arena used to generate turbulent flow to assess online model adaptation in the proposed techniques. Colored streamers on the fans aid in visualizing the wind.

evolution is governed by $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$, where

$$f(\mathbf{x}, \mathbf{u}) = \begin{cases} \dot{\mathbf{p}} & = \mathbf{v} \\ \dot{\mathbf{v}} & = \frac{1}{m} F \mathbf{R}_{\boldsymbol{\xi}} \mathbf{e}_3 - g\mathbf{e}_3 \\ \dot{\boldsymbol{\xi}} & = \mathbf{S}_{\boldsymbol{\xi}} \boldsymbol{\omega} \\ \dot{\boldsymbol{\omega}} & = \mathbf{J}^{-1} \left(\boldsymbol{\tau} - \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} \right) \end{cases} \tag{A.1}$$

The constants g, m, and $\mathbf{J}$ denote gravity, vehicle mass, and inertia, respectively. The vector $\mathbf{e}_3$ is the third column of the 3×3 identity matrix, $\mathbf{R}_{\boldsymbol{\xi}}$ denotes the rotation matrix formed from the ZYX Euler angles $\boldsymbol{\xi}$ that takes vectors from body frame to world frame, and $\mathbf{S}_{\boldsymbol{\xi}}$ is the inverse of the Jacobian that relates ZYX Euler angle rates to angular velocities [81].

Figure A.4: The small quadrotor equipped with an ODROID-XU4 used for experimental validation of NPE (Ch. 3), EPC (Ch. 4), and Robust EPC (Ch. 5).

Small Custom Quadrotor

The primary experimental platform used in this thesis is the small, 790 g custom quadrotor shown in Fig. A.4. The quadrotor is equipped with an ODROID-XU4 computer (Cortex-A15 2 GHz and Cortex-A7 octa-core CPU and 2 GB of RAM) that satisfies the size, weight, and power limitations of the system. All control algorithms are implemented in C++ via ROS [97] and run in real-time on the ODROID. The resulting force and torque commands are sent to a Pixhawk autopilot (via the Mavlink protocol) where they are converted to motor commands and applied to the system. Odometry from the motion capture system is broadcast wirelessly to the ODROID where it is combined with IMU data from the Pixhawk via an Unscented Kalman Filter to produce state estimates. The ODROID also enables hardware-in-the-loop simulations by running the full control architecture and vehicle simulator (Sect. A.1) to provide realistic solution times for the various components in the proposed algorithms.

Figure A.5: The Crazyflie quadrotor used for experimental validation of the efficient explicit adaptive NMPC technique (Ch. 6).

Crazyflie Quadrotor

We also use a Bitcraze Crazyflie 2.0 nano quadrotor to demonstrate the efficiency of the algorithm developed in Ch. 6. The Crazyflie has a total flight mass of 32 g and is severely compute-constrained with a 168 MHz STM32F405 Cortex-M4 processor, 192 KB of SRAM, and 1 MB of flash memory. Odometry and position setpoints are sent to the platform wirelessly, and an onboard Kalman filter integrates odometry and IMU to generate state estimates. All control is performed onboard the Crazyflie, and the control algorithms are implemented in C to facilitate cross-compilation. As the EPC-based techniques developed in Ch. 6 rely heavily on matrix operations, we have developed a set of custom linear algebra routines that exploit the sparse and block-concatenated structure of the matrices in EPC in order to optimize runtime.

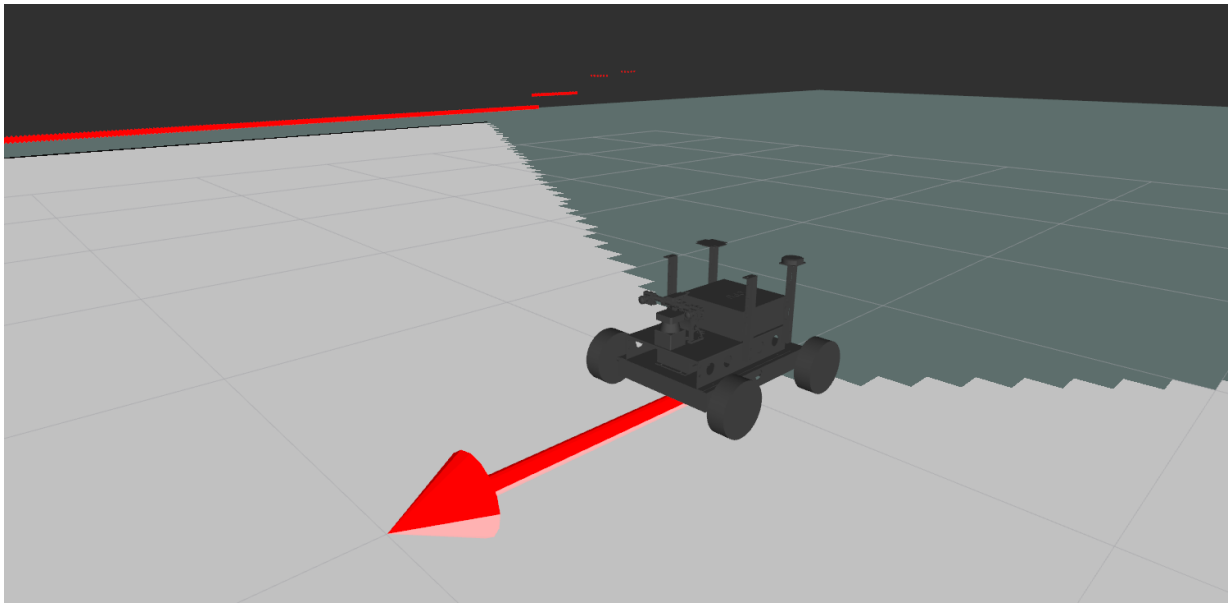


Figure A.6: The simulated ground robot used to evaluate performance of Robust EPC (Ch. 5). The large arrow denotes the vehicle heading, while the simulated laser scan returns are shown by the red points in the background.

A.3.2 Skid-steer Ground Robot

Chapter 5 presents a set of simulation results with a skid-steer ground robot (Fig. A.6) equipped with a planar laser-scanner (270° field of view, 1081 beams, 30 m maximum range). The simulated robot obtains state feedback via a Simultaneous Localization and Mapping (SLAM) architecture that employs an unscented Kalman filter (UKF) to fuse estimates from ICP-based laser odometry and histogram filter-based localization [104]. The UKF also provides covariances estimates that capture the uncertainty in the state estimate due to imperfect ICP solutions, thereby enabling use of the proposed Robust EPC algorithm.

We consider a dynamics model that captures the translational dynamics in the $x-y$ plane, heading, θ, and the angular velocities of the left and right wheels, w_l and w_r, respectively. As the robot is modeled after a skid-steer platform with low-level velocity control, the control inputs available to Robust EPC are the body-frame velocity and angular velocity commands, v^{des} and ω^{des}, respectively. The resulting nonlinear dynamics model is given by

$$f(\mathbf{x}, \mathbf{u}) = \begin{cases} \dot{x} & = v\cos(\theta) \\ \ddot{x} & = \dot{v}\cos(\theta) - v\dot{\theta}\sin(\theta) \\ \dot{y} & = v\sin(\theta) \\ \ddot{y} & = \dot{v}\sin(\theta) + v\dot{\theta}\cos(\theta) \\ \dot{\theta} & = \omega \\ \ddot{\theta} & = \dot{\omega} \\ \dot{w}_l & = \frac{1}{R}\dot{v} - \frac{1}{2}\frac{T}{R}\dot{\omega} \\ \dot{w}_r & = \frac{1}{R}\dot{v} + \frac{1}{2}\frac{T}{R}\dot{\omega} \end{cases} \tag{A.2}$$

where $v = \frac{1}{2}R(w_l + w_r)$, $\omega = \frac{R}{T}(w_r - w_l)$, $\dot{v} = -K_f(v - v^{\text{des}})$, $\dot{\omega} = -K_\tau(\omega - \omega^{\text{des}})$ R is the wheel radius, T is the vehicle track (distance between left and right wheels), and K_f and K_τ are the low-level velocity control gains.

Appendix B

Stability Properties

The preceding chapters detail a series of feedback control methodologies and demonstrate stable trajectory tracking performance through numerous simulation and experimental studies. This empirical stability assessment is motivated by the practical nature of the challenges we aim to address (e.g., state and model uncertainty, limited compute). However, as these techniques are derived from control strategies with well known analytical stability properties, we include a brief discussion of these properties below.

B.1 NPE Stability

NPE is a hybrid controller with two discrete modes. The first mode corresponds to operation within the current database, while the second corresponds to application of the intermediate controller. Additionally, NPE does not constitute a fundamental change in the optimal control formulation, but rather, it is an alternate solution strategy similar to explicit MPC [47, 48] and Partial Enumeration [80]. Therefore, to show stability with NPE, it is sufficient to show that 1) each mode preserves stability of the underlying control formulation and that 2) switching between stable modes preserves stability.

B.1.1 Mode 1: In-Database Operation

Switching between controllers in the database is identical to explicit MPC, and, by construction, the explicit approach preserves all stability and performance properties of the optimization-based solution [47, 48]. This implies that if the optimization is formulated such that it yields stability, the first operating mode in NPE will preserve it.

We therefore formulate the nonlinear receding horizon optimal control problem in NPE as

$$\begin{aligned} \underset{\mathbf{u}_k}{\operatorname{argmin}} & \sum_{k=0}^{N-1} J(\mathbf{x}_{k+1}, \mathbf{r}_{k+1}, \mathbf{u}_k) \\ \text{s.t.} \quad & \dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) \\ & g(\mathbf{x}_{k+1}, \mathbf{u}_k) \leq 0 \\ & \forall k = 0, \ldots, N-1 \end{aligned}$$

and the corresponding local control problem as

$$\begin{aligned} \underset{\bar{\mathbf{u}}_k}{\operatorname{argmin}} & \sum_{k=0}^{N-1} \frac{1}{2}(\bar{\mathbf{x}}_{k+1} - \bar{\mathbf{r}}_{k+1})^{\mathrm{T}} \mathbf{Q}_{k+1}(\bar{\mathbf{x}}_{k+1} - \bar{\mathbf{r}}_{k+1}) + \frac{1}{2}\bar{\mathbf{u}}_k^{\mathrm{T}} \mathbf{R}_k \bar{\mathbf{u}}_k \\ \text{s.t.} \quad & \bar{\mathbf{x}}_{k+1} = \mathbf{A}\bar{\mathbf{x}}_k + \mathbf{B}\bar{\mathbf{u}}_k \\ & \mathbf{G}_{\mathbf{x}_{k+1}} \bar{\mathbf{x}}_{k+1} \leq \mathbf{g}_{\mathbf{x}_{k+1}} \\ & \mathbf{G}_{\mathbf{u}_k} \bar{\mathbf{u}}_k \leq \mathbf{g}_{\mathbf{u}_k} \\ & \forall\, k = 0, \ldots, N-1 \end{aligned}$$

where the current state is taken to be the nominal state, $\mathbf{x}^* = \mathbf{x}_0$, $\mathbf{r}_1, \ldots, \mathbf{r}_N$ denote N reference states, and $\bar{\mathbf{r}} = \mathbf{r} - \mathbf{x}^*$ (see Ch. 3 for details). We consider this general formulation as it permits several techniques commonly used to ensure stability in MPC and NMPC. This includes terminal state penalties [105] via appropriate choice of $J(\cdot)$ and $\mathbf{Q}_N$; terminal state constraints [106] via appropriate choice of $g(\cdot)$, $\mathbf{G}_{\mathbf{x}_N}$, and $\mathbf{g}_{\mathbf{x}_N}$; and "dual-mode" formulations that combine terminal set constraints (again given by an appropriate choice of $g(\cdot)$, $\mathbf{G}_{\mathbf{x}_N}$, and $\mathbf{g}_{\mathbf{x}_N}$) with a locally-

stabilizing controller [107].

B.1.2 Mode 2 - Intermediate Controller

Although there are several options for the choice of intermediate controller for Partial Enumeration based approaches [80], we consider a short horizon (online optimization-based) MPC with slack on the state constraints to ensure feasibility.

One such soft constrained formulation adds a vector of slack variables, $\boldsymbol{\varepsilon}$, to each state constraint and penalizes any deviation from $\mathbf{0}$ by a weighting matrix, $\mathbf{S}$,

$$\underset{\bar{\mathbf{u}}_k}{\operatorname{argmin}} \sum_{k=0}^{N-1} \frac{1}{2}(\bar{\mathbf{x}}_{k+1} - \bar{\mathbf{r}}_{k+1})^{\mathrm{T}}\mathbf{Q}_{k+1}(\bar{\mathbf{x}}_{k+1} - \bar{\mathbf{r}}_{k+1}) + \frac{1}{2}(\bar{\mathbf{u}}_k - \bar{\mathbf{u}}_{\hat{\mathbf{p}}})^{\mathrm{T}}\mathbf{R}_k(\bar{\mathbf{u}}_k - \bar{\mathbf{u}}_{\hat{\mathbf{p}}}) + \frac{1}{2}\boldsymbol{\varepsilon}_k^{\mathrm{T}}\mathbf{S}_k\boldsymbol{\varepsilon}_k$$

$$\begin{aligned} \text{s.t.} \quad & \bar{\mathbf{x}}_{k+1} = \mathbf{A}\bar{\mathbf{x}}_k + \mathbf{B}\bar{\mathbf{u}}_k \\ & \mathbf{G}_{\mathbf{x}_{k+1}}\bar{\mathbf{x}}_{k+1} \leq \mathbf{g}_{\mathbf{x}_{k+1}} + \boldsymbol{\varepsilon}_k \\ & \mathbf{G}_{\mathbf{u}_k}\bar{\mathbf{u}}_k \leq \mathbf{g}_{\mathbf{u}_k} \\ & \forall\, k = 0, \ldots, N-1 \end{aligned}$$

Although this formulation is commonly used in practice [47, 108] to ensure problem feasibility, it is not guaranteed to yield asymptotic stability [109]. Therefore, to ensure asymptotic stability, we can employ various techniques as appropriate for a given control problem, Examples include requiring that the slack variables be monotonically decreasing [110], limiting the number of constraint violations over the horizon [111], explicitly minimizing the violation duration [112], and restricting the relaxation via modifications to the terminal constraints [109]. Again, as NPE does not prescribe a specific intermediate controller, any of these options with known stability properties can be applied to ensure the asymptotic stability of the second mode.

B.1.3 Stable Switching

Assuming that the two modes are stabilized via their respective controllers, we finally wish to ensure asymptotic stability across mode transitions. While Lyapunov-based methods can be applied to the switching system, assessing stability of a system with arbitrarily frequent transitions requires the identification of a common Lyapunov function that satisfies stability criteria for both modes [113]. An alternate class of approaches yield asymptotic stability by regulating the transitions between modes. Dwell time based approaches formalize the notion that sufficiently slow switching between modes will preserve stability [113]. A family of Lyapunov functions (corresponding to each mode) can be used to show asymptotic stability, but this requires the Lyapunov functions and dwell time in each mode to be selected such that the value of the corresponding Lyapunov function decreases beyond the final value from the previous mode [114]. However, in the case of NPE, a dwell time requirement cannot be enforced in the first mode (in-database) as a transition may be triggered at any time by violations of the KKT conditions.

We therefore consider the Average Dwell Time (ADT) requirement [115]. A dwell time parameter, τ_D, denotes the duration for which a mode must be sustained to ensure stability. ADT extends this to permit rapid mode transitions as long as the average time spent in a mode satisfies τ_D [115]. Although NPE is expected to operate primarily in the first mode, we cannot enforce a dwell time requirement on it as noted above. Therefore, we can ensure asymptotic stability by enforcing the dwell time requirement on the second mode (intermediate controller). As a result, the controller will either operate in mode 1 with infrequent transitions to mode 2, thereby implicitly satisfying the dwell time requirement for mode 1, or it will experience numerous transitions to mode 2 (e.g., while constructing the database), each of which is required to dwell for sufficient time to maintain an average dwell time of τ_D. As a result, the controller avoids potentially destabilizing chattering between modes.

B.2 EPC Stability

The EPC stability argument largely follows NPE as it retains the hybrid control structure and solves a very similar generalized receding horizon optimal control formulation. However, there are few additional considerations in mode 1 due to its incorporation of semi-parametric dynamics model learning.

We wish to control a nonlinear system subject to (in general) nonlinear constraints, and as a result, paralleled an explicit NMPC formulation [48] in NPE. In EPC, we instead approximate the resulting nonlinear optimization problem via a series of local quadratic programs. This is similar to single-step SQP solution strategies for NMPC [43, 44] that are known to preserve stability [116]. As a result, we can formulate the underlying receding horizon optimal control problem in EPC as

$$\underset{\bar{\mathbf{u}}_k}{\operatorname{argmin}} \sum_{k=0}^{N-1} \frac{1}{2}(\bar{\mathbf{x}}_{k+1} - \bar{\mathbf{r}}_{k+1})^{\mathrm{T}}\mathbf{Q}_{k+1}(\bar{\mathbf{x}}_{k+1} - \bar{\mathbf{r}}_{k+1}) + \frac{1}{2}(\bar{\mathbf{u}}_k - \bar{\mathbf{u}}_{\hat{\mathbf{p}}})^{\mathrm{T}}\mathbf{R}_k(\bar{\mathbf{u}}_k - \bar{\mathbf{u}}_{\hat{\mathbf{p}}})$$

$$\text{s.t.} \quad \bar{\mathbf{x}}_{k+1} = \mathbf{A}\bar{\mathbf{x}}_k + \mathbf{B}\bar{\mathbf{u}}_k + \tilde{\mathbf{c}}$$

$$\mathbf{G}_{\mathbf{x}_{k+1}}\bar{\mathbf{x}}_{k+1} \leq \mathbf{g}_{\mathbf{x}_{k+1}}$$

$$\mathbf{G}_{\mathbf{u}_k}\bar{\mathbf{u}}_k \leq \mathbf{g}_{\mathbf{u}_k}$$

$$\forall\, k = 0, \ldots, N-1$$

where the current state is taken to be the nominal state, $\mathbf{x}^* = \mathbf{x}_0$, $\mathbf{r}_1, \ldots, \mathbf{r}_N$ denote N reference states, $\bar{\mathbf{r}} = \mathbf{r} - \mathbf{x}^*$, and $\tilde{\mathbf{c}} = \mathbf{c} + \hat{\mathbf{p}}$ (see Ch. 4 for details). Again this general formulation is compatible with a variety of techniques known to yield a stabilizing controller. EPC supports both parametric adaptation strategies, which yield well-established adaptive MPC stability criteria similar to those mentioned above [14, 66], and semiparametric adaptation strategies, e.g. based on LWPR or ISSGPR. Although these semiparametric techniques lack inherent bounds that yield similar stability criteria, we can recover guarantees on stability by enforcing limits on the model learner outputs (e.g., as part of the filtering strategy mentioned in Sect. 4.1.1).

Both norm bounds [14] and Lipschitz bounds [75] on the overall dynamics model permit stable formulations.

The stability properties of mode 2 and the ADT-based approach to ensuring stability across mode transitions follow directly from Sect. B.1.2 and Sect. B.1.3, respectively.

Bibliography

[1] D. Lang, "My underwater robot." TED Talks, Feb. 2013.

[2] H. Inotsume, M. Sutoh, K. Nagaoka, K. Nagatani, and K. Yoshida, "Modeling, Analysis, and Control of an Actively Reconfigurable Planetary Rover for Traversing Slopes Covered with Loose Soil," *J. Field Robot.*, vol. 30, pp. 875–896, Nov. 2013.

[3] A. Mokhtari and A. Benallegue, "Dynamic feedback controller of Euler angles and wind parameters estimation for a quadrotor unmanned aerial vehicle," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, (New Orleans, LA), pp. 2359–2366, Apr. 2004.

[4] J. How, E. King, and Y. Kuwata, "Flight demonstrations of cooperative control for UAV teams," in *AIAA Unmanned Unlimited*, (Chicago, IL), Sept. 2004.

[5] J. Yao, V. R. Desaraju, and N. Michael, "Experience-Based Models of Surface Proximal Aerial Robot Flight Performance in Wind," in *Proc. of the Intl. Sym. on Exp. Robot.*, (Tokyo, Japan), pp. 1–10, Oct. 2016.

[6] C. Powers, D. Mellinger, A. Kushleyev, B. Kothmann, and V. Kumar, "Influence of Aerodynamics and Proximity Effects in Quadrotor Flight," in *Proc. of the Intl. Sym. on Exp. Robot.*, (Quebec City, Canada), pp. 1–14, June 2012.

[7] S. Shen, N. Michael, and V. Kumar, "Autonomous multi-floor indoor navigation with a computationally constrained MAV," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, (Shanghai, China), pp. 20–25, May 2011.

[8] V. R. Desaraju, N. Michael, M. Humenberger, R. Brockers, S. Weiss, J. Nash, and

L. Matthies, "Vision-based landing site evaluation and informed optimal trajectory generation toward autonomous rooftop landing," *Auton. Robots*, vol. 39, pp. 445–463, Oct. 2015.

[9] G. Antonelli, S. Chiaverini, N. Sarkar, and M. West, "Adaptive Control of an Autonomous Underwater Vehicle: Experimental Results on ODIN," *IEEE Trans. Control Syst. Technol.*, vol. 36, no. 9, pp. 1420–1423, 2001.

[10] LT. N. D. Valladarez and N. E. Du Toit, "Robust Adaptive Control of Underwater Vehicles for Precision Operations," in *OCEANS*, (Washington, DC), pp. 1–7, Oct. 2015.

[11] Z. Liu, Y. Zhang, X. Yu, and C. Yuan, "Unmanned surface vehicles: An overview of developments and challenges," *Annual Reviews in Control*, vol. 41, pp. 71–93, 2016.

[12] C. Ostafew, A. Schoellig, and T. Barfoot, "Learning-Based Nonlinear Model Predictive Control to Improve Vision-Based Mobile Robot Path-Tracking in Challenging Outdoor Environments," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pp. 4029–4036, IEEE, May 2014.

[13] A. Richards and J. How, "Robust Model Predictive Control with Imperfect Information," in *Proc. of the Amer. Control Conf.*, (New York City, NY), July 2005.

[14] P. Bouffard, A. Aswani, and C. Tomlin, "Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, (St. Paul, MN), May 2012.

[15] C. Brunner and T. Peynot, "Visual metrics for the evaluation of sensor data quality in outdoor perception," in *Proc. of the Workshop on Perf. Metrics for Intell. Sys.*, (Baltimore, MD), pp. 1–8, Sept. 2010.

[16] R. Brockers, M. Humenberger, S. Weiss, and L. Matthies, "Towards Autonomous Navigation of Miniature UAV," in *IEEE Conf. on Comp. Vision and Pattern Recog. Workshops*, (Columbus, OH), June 2014.

[17] "Crazyflie micro quadrotor." http://www.bitcraze.se/crazyflie/, 2016.

[18] M. Hofer, M. Muehlebach, and R. DAndrea, "Application of an Approximate Model Predictive Control Scheme on an Unmanned Aerial Vehicle," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, (Stockholm, Sweden), May 2016.

[19] R. Findeisen and L. Imsland, "State and output feedback nonlinear model predictive control: An overview," *Euro. J. of Control*, no. Apr., pp. 179–195, 2003.

[20] V. R. Desaraju and N. Michael, "Fast Nonlinear Model Predictive Control via Partial Enumeration," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, (Stockholm, Sweden), May 2016.

[21] V. R. Desaraju and N. Michael, "Leveraging Experience for Computationally Efficient Adaptive Nonlinear Model Predictive Control," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, (Singapore), May 2017.

[22] K. S. Narendra and A. M. Annaswamy, *Stable Adaptive Systems*. Prentice Hall, 1989.

[23] J. Wang, F. Holzapfel, E. Xargay, and N. Hovakimyan, "Non-Cascaded Dynamic Inversion Design for Quadrotor Position Control with L1 Augmentation," in *Proc. of the CEAS Specialist Conf. on Guidance, Navigation & Control*, (Delft, Netherlands), Apr. 2013.

[24] P. Khargonekar, I. Petersen, and K. Zhou, "Robust stabilization of uncertain linear systems: quadratic stabilizability and H/sup infinity / control theory," *IEEE Trans. Autom. Control*, vol. 35, pp. 356–361, Mar. 1990.

[25] G. V. Raffo, M. G. Ortega, and F. R. Rubio, "An integral predictive/nonlinear H_∞ control structure for a quadrotor helicopter," *Automatica*, vol. 46, no. 1, pp. 29–39, 2010.

[26] M. Ö. Efe, "Robust low altitude behavior control of a quadrotor rotorcraft through sliding modes," in *Medit. Conf. on Control and Autom.*, (Athens, Greece), pp. 0–5, July 2007.

[27] K. B. Ngo, R. Mahony, and Z.-P. Jiang, "Integrator backstepping using barrier functions for systems with multiple state constraints," in *Proc. of the IEEE Conf. on Decision and*

Control, (Seville, Spain), pp. 8306–8312, Dec. 2005.

[28] K. P. Tee, S. S. Ge, and E. H. Tay, "Barrier Lyapunov Functions for the control of output-constrained nonlinear systems," *Automatica*, vol. 45, no. 4, pp. 918–927, 2009.

[29] G. Wu and K. Sreenath, "Safety-critical and constrained geometric control synthesis using control Lyapunov and control Barrier functions for systems evolving on manifolds," in *Proc. of the Amer. Control Conf.*, (Chicago, IL), pp. 2038–2044, July 2015.

[30] E. G. Gilbert, I. Kolmanovsky, and K. T. Tan, "Discrete-time reference governors and the nonlinear control of systems with state and control constraints," *International Journal of Robust and Nonlinear Control*, vol. 5, no. 5, pp. 487–504, 1995.

[31] A. Bemporad, "Reference governor for constrained nonlinear systems," *IEEE Trans. Autom. Control*, vol. 43, pp. 415–419, Mar. 1998.

[32] S. J. Sun and I. V. Kolmanovsky *IEEE Trans. Control Syst. Technol.*, pp. 911–920, Nov.

[33] D. E. Kirk, *Optimal control theory: an introduction*. Courier Corporation, 2012.

[34] W. H. Chen, D. J. Ballance, and P. J. Gawthrop, "Optimal control of nonlinear systems: a predictive control approach," *Automatica*, vol. 39, pp. 633–641, 2003.

[35] J. H. Lee, "Model predictive control: Review of the three decades of development," *Intl. J. of Control, Autom. and Sys.*, vol. 9, no. 3, pp. 415–424, 2011.

[36] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, pp. 789–814, June 2000.

[37] A. Bemporad and M. Morari, "Robust model predictive control: A survey," *Robustness in identification and control*, pp. 207–226, 1999.

[38] J. Löfberg, *Minimax approaches to robust model predictive control*. PhD thesis, Linköping University, 2003.

[39] A. Alessio and A. Bemporad, "A Survey on Explicit Model Predictive Control," in *Nonlinear Model Predictive Control*, vol. 384 of *Lecture Notes in Control and Information*

Sciences, pp. 345–369, Springer Berlin Heidelberg, 2009.

[40] H. J. Ferreau, H. G. Bock, and M. Diehl, "An online active set strategy to overcome the limitations of explicit MPC," *International Journal of Robust and Nonlinear Control*, vol. 18, pp. 816–830, May 2008.

[41] Y. Wang and S. Boyd, "Fast Model Predictive Control Using Online Optimization," *IEEE Trans. Control Syst. Technol.*, vol. 18, pp. 267–278, Mar. 2010.

[42] M. Diehl, R. Findeisen, S. Schwarzkopf, I. Uslu, F. Allgöwer, H. G. Bock, E.-D. Gilles, and J. P. Schlöder, "An Efficient Algorithm for Nonlinear Model Predictive Control of Large-Scale Systems Part I: Description of the Method," *Automatisierungstechnik*, vol. 50, no. 12, pp. 557–567, 2012.

[43] B. Houska, H. J. Ferreau, and M. Diehl, "An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range," *Automatica*, vol. 47, no. 10, pp. 2279–2285, 2011.

[44] F. Debrouwere, M. Vukov, R. Quirynen, M. Diehl, and J. Swevers, "Experimental validation of combined nonlinear optimal control and estimation of an overhead crane," in *Proc. of the Intl. Fed. of Autom. Control*, (Cape Town, South Africa), pp. 9617–9622, Aug. 2014.

[45] M. Neunert, C. de Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. Buchli, "Fast Nonlinear Model Predictive Control for Unified Trajectory Optimization and Tracking," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, (Stockholm, Sweden), May 2016.

[46] A. Bemporad, "Model Predictive Control Design: New Trends and Tools," in *Proceedings of the 45th IEEE Conference on Decision and Control*, no. 1, pp. 6678–6683, Ieee, 2006.

[47] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, pp. 3–20, 2002.

[48] A. Grancharova and T. Johansen, *Explicit Nonlinear Model Predictive Control*, vol. 429.

Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[49] T. A. Johansen and A. Grancharova, "Approximate Explicit Constrained Linear Model Predictive Control via Orthogonal Search Tree," *IEEE Trans. Autom. Control*, vol. 48, pp. 810–815, May 2003.

[50] A. N. Fuchs, C. N. Jones, and M. Morari, "Optimized decision trees for point location in polytopic data sets - application to explicit MPC," in *Proc. of the Amer. Control Conf.*, (Baltimore, MD), pp. 5507–5512, June 2010.

[51] A. Domahidi, M. N. Zeilinger, M. Morari, and C. N. Jones, "Learning a Feasible and Stabilizing Explicit Model Predictive Control Law by Robust Optimization," in *Proc. of the IEEE Conf. on Decision and Control*, (Orlando, FL), pp. 513–519, Dec. 2011.

[52] S. Summers, D. M. Raimondo, C. N. Jones, J. Lygeros, and M. Morari, "Fast explicit nonlinear model predictive control via multiresolution function approximation with guaranteed stability," in *IFAC*, 2010.

[53] R. Tedrake, "LQR-Trees: Feedback motion planning on sparse randomized trees," in *Artificial Intelligence*, (Seattle, WA), p. 8, June 2009.

[54] R. Reist, P. Preiswerk, and R. Tedrake, "Feedback-motion lanning with simulation-based LQR-trees," *Intl. Journal of Robotics Research*, 2016.

[55] T. Zhang, G. Khan, S. Levine, and P. Abbeel, "Learning Deep Control Policies for Autonomous Aerial Vehicles with MPC-Guided Policy Search," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, (Stockholm, Sweden), May 2016.

[56] K. Hauser, "Learning the Problem-Optimum Map: Analysis and Application to Global Optimization in Robotics," in *Robot Learn. and Plan. Workshop at RSS*, (Ann Arbor, MI), June 2016.

[57] G. Pannocchia, S. J. Wright, B. T. Stewart, and J. B. Rawlings, "Efficient Cooperative Distributed MPC using Partial Enumeration," in *IFAC Intl. Symposium on Adv. Control of*

Chemical Process., pp. 607–612, July 2008.

[58] M. E. Taylor and P. Stone, "Transfer Learning for Reinforcement Learning Domains: A Survey," *J. of Machine Learning Research*, vol. 10, pp. 1633–1685, 2009.

[59] P. Ruvolo and E. Eaton, "ELLA: An efficient lifelong learning algorithm," in *Intl. Conf. on Machine Learning*, pp. 507–515, 2013.

[60] A. Saha, P. Rai, H. D. e, and S. Venkatasubramanian, "Online learning of multiple tasks and their relationships," in *Intl. Conf. on Artif. Intell. and Stat.*, pp. 643–651, 2011.

[61] R. S. Sutton, "Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming," in *Int. Conf. on Machine Learning*, vol. 02254, pp. 216–224, 1990.

[62] W. Y. Kwon, I. H. Suh, and S. Lee, "SSPQL: Stochastic shortest path-based q-learning," *International Journal of Control, Automation and Systems*, vol. 9, no. 2, pp. 328–338, 2011.

[63] B. Bakker, V. Zhumatiy, G. Gruener, and J. Schmidhuber, "Quasi-online reinforcement learning for robots," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2997–3002, 2006.

[64] L. A. Stein, "Imagination and situated cognition," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 6, no. 4, pp. 393–407, 1994.

[65] A. G. Di Nuovo, D. Marocco, S. Di Nuovo, and A. Cangelosi, "Autonomous learning in humanoid robotics through mental imagery," *Neural Networks*, vol. 41, pp. 147–155, 2013.

[66] H. Fukushima, T. Kim, and T. Sugie, "Adaptive model predictive control for a class of constrained linear systems based on the comparison model," *Automatica*, vol. 43, no. 2, pp. 301–308, 2007.

[67] Y. K. Ho, H. K. Yeoh, and F. S. Mjalli, "Generalized Predictive Control Algorithm

with Real-Time Simultaneous Modeling and Tuning," *Industrial & Eng. Chem. Research*, vol. 53, no. 22, pp. 9411–9426, 2014.

[68] S. Vijayakumar, A. DSouza, and S. Schaal, "Incremental Online Learning in High Dimensions," *Neural Comp.*, vol. 17, no. 12, pp. 2602–2634, 2005.

[69] A. Gijsberts and G. Metta, "Real-time model learning using Incremental Sparse Spectrum Gaussian Process Regression," *Neural Networks*, vol. 41, pp. 59–69, 2013.

[70] D. Mitrovic, S. Klanke, and S. Vijayakumar, "Adaptive optimal feedback control with learned internal dynamics models," *From Motor Learn. to Inter. Learn. in Rob.*, vol. 264, pp. 65–84, 2010.

[71] W. Langson, I. Chryssochoos, S. V. Raković, and D. Q. Mayne, "Robust model predictive control using tubes," *Automatica*, vol. 40, pp. 125–133, Jan. 2004.

[72] D. Q. Mayne, M. M. Seron, and S. V. Raković, "Robust model predictive control of constrained linear systems with bounded disturbances," *Automatica*, vol. 41, pp. 219–224, Feb. 2005.

[73] A. Richards, "Robust Model Predictive Control for Time-Varying Systems," in *Proc. of the IEEE Conf. on Decision and Control*, (Seville, Spain), pp. 3747–3752, Dec. 2005.

[74] I. Kolmanovsky and E. G. Gilbert, "Theory and computation of disturbance invariant sets for discrete-time linear systems," *Mathematical Problems in Engineering*, vol. 4, no. 4, pp. 317–367, 1998.

[75] V. Adetola and M. Guay, "Robust adaptive MPC for constrained uncertain nonlinear systems," *Intl. J. of Adaptive Control & Signal Process.*, vol. 25, no. 2, pp. 155–167, 2011.

[76] Y. Kuwata, A. Richards, and J. How, "Robust Receding Horizon Control using Generalized Constraint Tightening," in *Proc. of the Amer. Control Conf.*, (New York City, NY), pp. 4482–4487, July 2007.

[77] J. Yan and R. R. Bitmead, "Incorporating state estimation into model predictive control

and its application to network traffic control," *Automatica*, vol. 41, pp. 595–604, 2005.

[78] D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, pp. 2967–2986, 2014.

[79] M. Farrokhsiar, G. Pavlik, and H. Najjaran, "An integrated robust probing motion planning and control scheme: A tube-based MPC approach," *Robot. Auton. Syst.*, vol. 61, no. 12, pp. 1379–1391, 2013.

[80] G. Pannocchia, J. B. Rawlings, and S. J. Wright, "Fast, large-scale model predictive control by partial enumeration," *Automatica*, vol. 43, no. 5, pp. 852–860, 2007.

[81] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "Experimental evaluation of multirobot aerial control algorithms," *IEEE Robotics & Automation Magazine*, Sept. 2010.

[82] A. Droniou, S. Ivaldi, V. Padois, and O. Sigaud, "Autonomous online learning of velocity kinematics on the iCub: A comparative study," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, (Vilamoura, Algarve, Portugal), pp. 3577–3582, Oct. 2012.

[83] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[84] V. R. Desaraju and N. Michael, "Experience-driven Predictive Control," in *Robot Learn. and Plan. Workshop at RSS*, (Ann Arbor, MI), June 2016.

[85] R. Platt, Jr., R. Tedrake, L. Kaelbling, and T. Lozano-Perez, "Belief space planning assuming maximum likelihood observations," in *Proc. of Robot.: Sci. and Syst.*, 2010.

[86] D. Q. Mayne, E. C. Kerrigan, E. J. van Wyk, and P. Falugi, "Tube-based robust nonlinear model predictive control," *J. Robust and Nonlin. Control*, vol. 21, p. 13411353, 2011.

[87] F. Domes and A. Neumaier, "Rigorous Enclosures of Ellipsoids and Directed Cholesky Factorizations," *SIAM J. on Matrix Analysis and Appl.*, vol. 32, p. 62285, 2011.

[88] C. Conte, M. N. Zeilinger, M. Morari, and C. N. Jones, "Robust distributed model predictive control of linear systems," in *Euro. Control Conf.*, p. 5, July 2013.

[89] D. Q. Mayne and W. Langson, "Robustifying model predictive control of constrained linear systems," , vol. 37, pp. 1422–1423, 2001.

[90] Y. Kuwata, S. Karaman, J. Teo, E. Frazzoli, J. How, and G. Fiore, "Real-Time Motion Planning With Applications to Autonomous Urban Driving," *IEEE Trans. Control Syst. Technol.*, vol. 17, pp. 1105–1118, Sept. 2009.

[91] V. R. Desaraju and N. Michael, "Hierarchical adaptive planning in environments with uncertain, spatially-varying disturbance forces," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, (Hong Kong, China), pp. 5171–5176, May 2014.

[92] G. S. Aoude, B. D. Luders, J. P. How, and T. E. Pilutti, "Sampling-Based Threat Assessment Algorithms for Intersection Collisions Involving Errant Drivers," in *IFAC Symposium on Intell. Auton. Vehicles*, (Lecce, Italy), Sept. 2010.

[93] S. M. LaValle and J. J. Kuffner, "Randomized Kinodynamic Planning," *Intl. Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[94] B. D. Luders, *Robust Sampling-based Motion Planning for Autonomous Vehicles in Uncertain Environments*. PhD thesis, Massachusetts Institute of Technology, 2014.

[95] G. Ullah, W. J. Bruno, and J. E. Pearson, "Simplification of Reversible Markov Chains by Removal of States With Low Equilibrium Occupancy," *J. of Theoretical Biology*, vol. 311, p. 117129, 2012.

[96] G. E. Henter and W. B. Kleijn, "Minimum Entropy Rate Simplification of Stochastic Processes," *IEEE Trans. Pattern Analysis and Machine Learn.*, vol. 38, 2016.

[97] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An open-source Robot Operating System," in *ICRA Workshop on open source software*, (Kobe, Japan), p. 5, 2009.

[98] C. Richter, A. Bry, and N. Roy, "Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments," in *Proc. of the Intl. Sym. of Robot. Research*,

(Singapore), Dec. 2013.

[99] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[100] S. G. Johnson, "The NLopt nonlinear-optimization package," 2014.

[101] K. Ahnert and M. Mulansky, "Odeint Solving Ordinary Differential Equations in C++," in *Intl. Conf. on Num. Analysis and Applied Math.*, vol. 1586, (Halkidiki, Greece), pp. 1586–1589, Sept. 2011.

[102] C. Sanderson and R. Curtin, "Armadillo: a template-based C++ library for linear algebra," *J. of Open Source Software*, vol. 1, p. 26, 2016.

[103] "OpenBLAS: An optimized BLAS library." http://www.openblas.net/, 2016.

[104] E. A. Nelson, *Environment Model Adaptation for Autonomous Exploration*. PhD thesis, Carnegie Mellon University, 2015.

[105] W. H. Kwon, A. M. Bruckstein, and T. Kailath, "Stabilizing state-feedback design via the moving horizon method," *Intl. J. of Control*, vol. 37, no. 3, pp. 631–643, 1983.

[106] W. Kwon and A. Pearson, "A modified quadratic cost problem and feedback stabilization of a linear system," *IEEE Trans. Autom. Control*, vol. 22, pp. 838–842, Oct. 1977.

[107] H. Michalska and D. Q. Mayne, "Robust receding horizon control of constrained nonlinear systems," *IEEE Trans. Autom. Control*, vol. 38, no. 11, pp. 1623–1633, 1993.

[108] A. Zheng and M. Morari, "Stability of model predictive control with mixed constraints," *IEEE Trans. Autom. Control*, vol. 40, no. 10, pp. 1818–1823, 1995.

[109] M. N. Zeilinger, M. Morari, and C. N. Jones, "Soft constrained model predictive control with robust stability guarantees," *IEEE Trans. Autom. Control*, vol. 59, no. 5, pp. 1190–1202, 2014.

[110] A. Bemporad, F. Borrelli, and M. Morari, "Optimal controllers for hybrid systems: sta-

bility and piecewise linear explicit form," in *Proc. of the IEEE Conf. on Decision and Control*, (Sydney, NSW, Australia), pp. 1810–1815, Dec. 2000.

[111] M. Cannon, B. Kouvaritakis, and X. Wu, "Probabilistic Constrained MPC for Multiplicative and Additive Stochastic Uncertainty," *IEEE Trans. Autom. Control*, vol. 54, no. 7, pp. 1626–1632, 2009.

[112] P. Mhaskar, N. H. El-Farra, and P. D. Christofides, "Stabilization of nonlinear systems with state and control constraints using Lyapunov-based predictive control," *Systems & Control Letters*, vol. 55, pp. 650–659, 2006.

[113] D. Liberzon and A. S. Morse, "Basic Problems in Stability and Design of Switched Systems," *IEEE Control Sys. Mag.*, pp. 59–70, Oct. 1999.

[114] P. Peleties and R. DeCarlo, "Asymptotic Stability of m-Switched Systems using Lyapunov-Like Functions," in *Proc. of the Amer. Control Conf.*, (Boston, MA), pp. 1679–1684, June 1991.

[115] J. P. Hespanha and A. S. Morse, "Stability of switched systems with average dwell-time," in *Proc. of the IEEE Conf. on Decision and Control*, vol. 3, (Phoenix, AZ), pp. 2655–2660, Dec. 1999.

[116] M. Diehl, H. Ferreau, and N. Haverbeke, "Efficient numerical methods for nonlinear MPC and moving horizon estimation," in *Nonlinear Model Predictive Control*, pp. 391–417, 2009.