Predictive Control of Constrained Nonlinear Systems

Lauren Lieu

CMU-RI-TR-18-47
December 2018

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Nathan Michael, *Chair*
Sebastian Scherer
Ashwin Khadke

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

Abstract

Because of advanced autonomous capabilities, mobile robot systems are transitioning from usage in controlled laboratory and factory settings to deployment in unstructured and increasingly complex environments. However, robustness of these field robots to environmental unknowns at the time of deployment remains a challenge. For example, in aerial infrastructure inspection, control of micro air vehicles is challenging due to induced transient flow conditions and varied lighting conditions that degrade state estimation. To safely navigate within these settings, an efficient and adaptive control strategy is essential to mitigate risk and increase system robustness.

This thesis work examines a computationally efficient strategy for solving nonlinear model predictive control problems by learning from past experience. Efficiency is achieved by constructing a library of controllers that map the state, reference, and dynamics model to the locally optimal feedback control laws. An affine dynamics model is combined with an online learned component that estimates nonlinearities and perturbations to accurately model the system. The safety, accuracy, and robustness of this control technique is evaluated by comparing the tracking performance against classic reactive control strategies.

Acknowledgments

First, I would like to thank my advisor, Professor Nathan Michael, for the opportunity to dive into a new research area and steer me on this journey. I am grateful for many thoughtful conversations - counsel on matters both professional and personal, energizing discussions about what it means to create impact, and unwavering belief in my potential.

I am thankful for Vishnu Desaraju's patient guidance, lighting the way as I explored the mysterious realms of EPC. I gratefully acknowledge Vishnu's support for me to build upon his work and replicate in part or full derivations, figures, and text originally presented in his dissertation and prior publications.

Thank you to every member of RISLab who made each day better with awkward humor and honest advice, squished together in the Masters suite. To name a few friends and colleagues: Cormac, for going the extra mile to help prepare on the day of my thesis talk; Arjav, for ideas and feedback that made my slides illustrative and engaging; Wennie, for the odd pep talks and female camaraderie; Tim and Matt, for jumping in last minute to edit this document; and Shaurya, for providing a sounding board and lots of hugs. I also owe a shout-out to those in my cohort who buoyed me up while we were in the same boat: Jerry Hsiung, Ceci Morales, and Ratnesh Madaan. Thanks for checking up on my page count, and showing up with true grit, perspective, and determination.

In trying times, I am fortunate to have received phone calls, care packages, and meals brought to me by Brian Hou, Julia Harreschou, Tuan Truong, and my family. Your thoughtful words and generosity kept me grounded, and this would not have happened without you.

This work was supported by the Department of Energy Environmental Management Traineeship DEEM0004067.

Contents

1 Introduction **1**

1.1 Motivation 2

1.2 Challenges 3

1.3 Research Contributions 4

1.4 Thesis Outline 5

2 Background **7**

2.1 Feedback Control Strategies 7

2.2 Model Predictive Control 10

2.2.1 Fast Nonlinear Model Predictive Control 12

2.2.2 Model Predictive Control with Uncertainty 13

3 Experience-driven Predictive Control **15**

3.1 Approach 15

3.1.1 Online Model Adaptation 16

3.1.2 Receding Horizon Control 18

3.1.3 Experience-driven Predictive Control Algorithm 22

3.2 Quadrotor Dynamics Model and Trajectory Tracking 23

3.2.1 State Formulation 26

3.2.2 Cascaded Control 28

3.2.3 3D Trajectory Control 29
3.3 Results 29

4 Conclusion **35**
4.1 Future Work 36

A Classic Control Strategies **37**
A.1 Proportional-Derivative Control 37
A.2 Linear Quadratic Regulator 38

List of Figures

1.1 Within an autonomous robot's autonomy framework, the control subsystem takes the planned trajectory and current state estimate as inputs. Using this information, the control law generates a set of commands for the vehicle's actuators. . . . 2

1.2 A tethered crawler designed at Savannah River National Labs for inspection of the H-canyon exhaust tunnel in 2009 [28]. This vehicle is part of a series of remote-controlled vehicles built specifically to determine the conditions inside the tunnel and check for structural concerns. 3

1.3 A hexrotor flying above an indoor crane to use its downward facing sensors to inspect the gantry and crane below. 4

2.1 A block diagram of a closed loop control system using a feedback loop to control the process variable by comparing the measured output to the reference, similar to [8]. The error signal is used to compute a control input to eliminate error. . . . 8

2.2 A taxonomy of feedback control strategies with emphasis on techniques which mitigate the effects of disturbances and handle state uncertainty. This diagram is adapted from [8]. 8

2.3 Model predictive control looks ahead over a finite horizon to estimate the evolution of the plant, and minimize cost function parameters such as tracking error and control effort. The plot illustrates the a horizon of length p over which MPC optimizes to compute a sequence of control inputs $\mathbf{u}$. 9

3.1 **Top:** Each control iteration, EPC checks the controller database to see if any of the existing entries is locally optimal. **Middle:** If a suitable controller is not found, the formulated QP is solved to compute a new controller. **Bottom:** The newly computed controller is added to the controller database. 25
3.2 The coordinate systems and free body diagram for a quadrotor, showing the forces and moments acting on the vehicle. This diagram is adapted from [27]. . . 26
3.3 Nested control loops for attitude and position control, following the cascaded model described in [27]. 28
3.4 A quadrotor tracking a circular trajectory using both EPC and PD controllers. . . 31
3.5 Tracking error for PD and EPC controllers for a quadrotor following a circular trajectory. 32
3.6 Y Velocity tracking for a quadrotor following a zig-zag trajectory. The vehicle is subject to a wind disturbance of 1.5 m/s in the $\frac{-3\pi}{4}$ direction in the XY plane. The EPC controller abides by velocity constraints to remain with the bounded velocity limits of 3.8 m/s. 33
3.7 Position tracking for a quadrotor flying in a wind field with and without online model adaptation (Locally Weighted Projection Regression). The aerial robot is following a figure 8 trajectory in an environment with a wind disturbance of 3 m/s in the $\frac{\pi}{4}$ direction in the XY plane. 34

Chapter 1

Introduction

As robotics systems evolve and continue to advance in capability, their usage has expanded from controlled settings into unstructured and dynamic environments. In order for mobile robot systems to perform increasingly complex tasks and navigate outside of laboratory or factory settings, robots need greater levels of autonomy to perceive their surroundings and choose feasible, safe actions as they move through the world. With greater autonomy, mobile robotic systems are useful for performing jobs that would otherwise be strenuous, inaccessible, or hazardous to humans. In the field, robots can assist with tasks such as emergency response, infrastructure inspection, space exploration, and crop monitoring.

Deploying a robot in an unpredictable range of operating condition poses unique challenges to safe navigation, exploration, and inspection. Due to unforeseen hazards, it is important that the robot can move deliberately within the environment, and ideally learn from its interactions to adapt to ensure optimum performance; doing so is largely dependent on three major subsystems: reliable state estimation (belief about the robot's position and orientation), planning, and control. The focus of this thesis work centers on the advancement of robust, adaptive control strategies to achieve these desired capabilities is the focus of this thesis work. Within a mobile robot's autonomy framework, the control subsystem receives information about the vehicle's state estimate and desired trajectory, and interprets these inputs to generate commands for the system's actua-

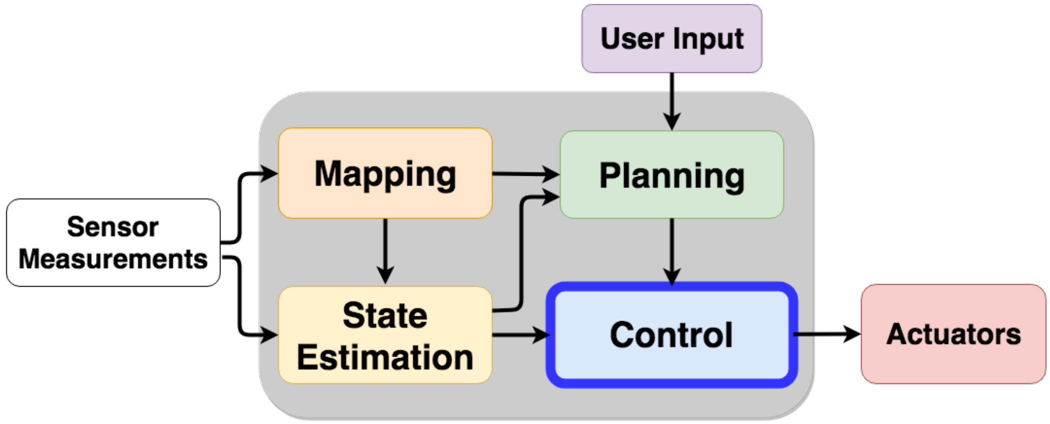

Figure 1.1: Within an autonomous robot's autonomy framework, the control subsystem takes the planned trajectory and current state estimate as inputs. Using this information, the control law generates a set of commands for the vehicle's actuators.

tors. The autonomy framework is comprised of perception, cognition, and action capabilities, as shown in Fig. 1.1.

Within the realm of control theory, there are different techniques that are designed to achieve specific objectives such as precise tracking, robustness to uncertainty, or online adaptation. Depending on the application and chosen model, a suitable control strategy can be applied and tuned to provide desirable characteristics. The target application of this thesis work is to enable micro air vehicles to perform inspection tasks and assess structural integrity in a safe and effective manner.

1.1 Motivation

Infrastructure built to produce, store, and dispose of nuclear materials has degraded over time due to age. Consequentially, the integrity of these containment structures has been compromised. At the time of the Manhattan Project and the ensuing Cold War nuclear arms race (1947-1991) there was substantial investment in government-sponsored nuclear weapons and energy research. Since then, the Department of Energy's Office of Environment Management focused on environmental restoration and waste management to cleanup the environmental legacy left

Figure 1.2: A tethered crawler designed at Savannah River National Labs for inspection of the H-canyon exhaust tunnel in 2009 [28]. This vehicle is part of a series of remote-controlled vehicles built specifically to determine the conditions inside the tunnel and check for structural concerns.

by these projects. In the past 28 years, over $160 billion has been spent on remediation of 91 sites within the United States [38]. The remaining work to be done is projected into the next 50 years and valued at upwards of $250 billion. Two major sites of interest are the Hanford and Savannah River Sites, at which applied research and development teams are pursuing opportunities to deploy remote systems to assist with operations and maintenance [28, 30]. A tethered, remotely operated crawler designed for inspection of a hazardous exhaust tunnel is shown in Fig. 1.2. Robotic systems can be leveraged for tasks such as nuclear facility decommissioning, radioactive waste storage and disposal, and nuclear material management.

Efficient inspection and maintenance is needed to routinely monitor facilities and prevent leakage of harmful chemicals into the environment and reduce risk. These settings are often inaccessible and hazardous to human operators. Remote inspection with robotic platforms can provide broader and more consistent coverage, and reduce risk to personnel [20].

1.2 Challenges

For micro air vehicles operating within an enclosed industrial setting, flight in proximity to surfaces induces aerodynamic effects that can destabilize aerial vehicles. For instance, the hexrotor flying in Figure 1.3 approached the indoor gantry, and its proximity caused transient flow condi-

Figure 1.3: A hexrotor flying above an indoor crane to use its downward facing sensors to inspect the gantry and crane below.

tions that interfered with flight. The presence of radiation, debris, and varied lighting conditions interferes with sensor readings and degrades state estimation performance. There are multiple sources of uncertainty that impact an aerial vehicle's ability to navigate an unpredictable environment. To accurately track trajectories within an enclosed industrial setting, a robust control strategy is needed to bound uncertainty and maintain safety in unknown, dynamic conditions. Operations in constrained environments can prohibit the repair or recovery of the inspection platform. Therefore, safe flight necessitates a high-rate, efficient feedback control system that responds quickly to exogenous perturbations.

1.3 Research Contributions

This thesis investigates the development and performance of Experience-driven Predictive Control (EPC), as proposed by Desaraju [8]. Experience-driven Predictive Control is a robust and adaptive strategy for constrained nonlinear systems with uncertain dynamics and state uncertainty. This approach reduces computational cost by learning from past experiences, and modifying an affine dynamics model online to accurately model the evolution of the system. This fast model predictive control approach is tested to assess model accuracy and tracking performance. EPC is compared to a reactive controller in simulation for micro air vehicles.

1.4 Thesis Outline

In this work we build upon a control strategy that solves nonlinear model predictive control problems efficiently to operate on computationally constrained remote systems. Experience-driven predictive control [8] uses a learning-based function approximation technique to model uncertain, time-varying dynamics and accurately predict the system's evolution. The computational cost of optimal control methods is reduced by learning from past experiences, recognizing similar scenarios, and reusing previously computed feedback control laws.

- **Chapter 2** provides an overview of feedback control strategies and focuses on methods that mitigate the effects of uncertainty, provide constraint satisfaction, and adapt to unknown disturbances. This chapter goes on to describe model predictive control (MPC) and the challenges assciated with MPC for constrained nonlinear systems.
- **Chapter 3** details the Experience-driven Predictive Control algorithm [10]. Individual sections step through the model learning aspect, the optimization problem formulated as a quadratic program, and logical flow of maintaining a controller database. This is followed by a description of 3D quadrotor dynamics and simulation results.
- **Chapter 4** summarizes the contributions of this work and describes future directions to extend this model predictive control technique.

Chapter 2

Background

This thesis investigates the performance of a safe and efficient control strategy for constrained nonlinear systems. The EPC algorithm described in this work was built upon related work in receding horizon control, fast nonlinear model predictive control, and model learning. This chapter provides a description of related work in feedback control, with particular emphasis on control techniques that handle state uncertainty and disturbances.

2.1 Feedback Control Strategies

Feedback control, or closed loop control, determines the control input $\mathbf{u}$ by using the output from the plant to estimate the system's current state $\mathbf{x}$, and measure deviation from the desired state $\mathbf{x_{ref}}$. Based on the computed error, the controller generates an input signal to reduce the error and improve tracking of the reference, as shown in Fig. 2.1. Often in real world applications, there are unpredictable factors that affect the plant as well, such as uncertainty in the state estimate or external disturbances that are not part of the process model. Figure 2.2 categorizes different types of feedback control based on their abilities to handle disturbances, account for uncertainty, and obey state or input constraints.

Reactive control strategies aim to generate a control input based on past and current measure-

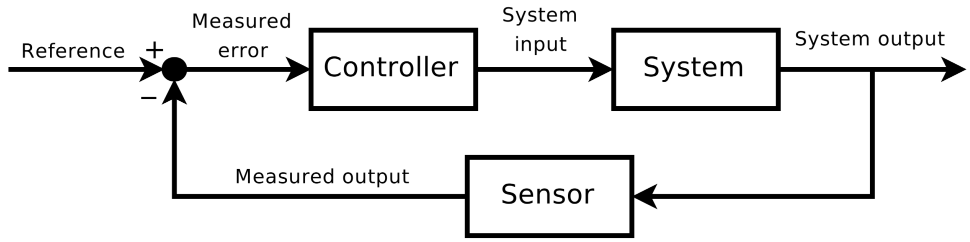

Figure 2.1: A block diagram of a closed loop control system using a feedback loop to control the process variable by comparing the measured output to the reference, similar to [8]. The error signal is used to compute a control input to eliminate error.

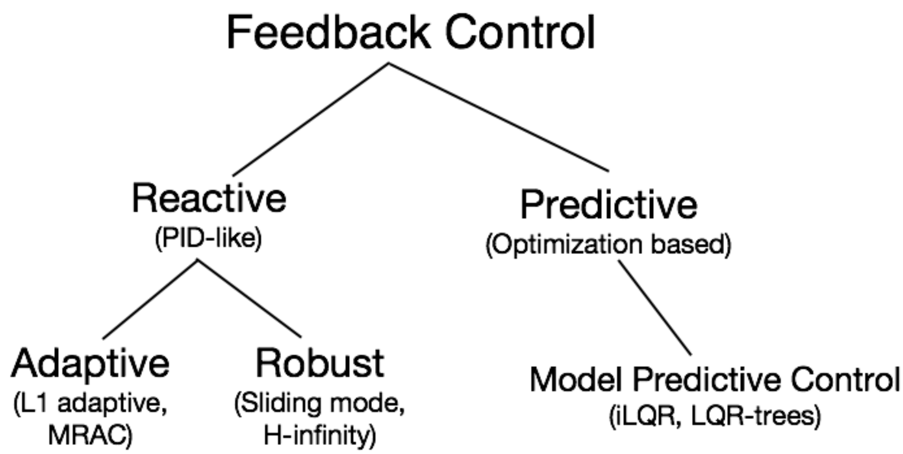

Figure 2.2: A taxonomy of feedback control strategies with emphasis on techniques which mitigate the effects of disturbances and handle state uncertainty. This diagram is adapted from [8].

ments of the system's state and error relative to the reference. The ubiquitous example of reactive control is proportional-integral-derivative (PID) control [1], which can run at high rates due to its simplicity. To account for uncertainty, this method can be augmented by adaptive control strategies such as L1 adaptive control [47] or model reference adaptive control [21]. Adaptive techniques are flexible by estimating and compensating for perturbations to the system online. However, such methods often do not provide guarantees of constraint satisfaction.

Alternatively, robust control strategies such as sliding mode [6, 24] or H-infinity control [35] bound the effects of uncertainty to preserve stability and tracking performance. These techniques use a static control policy and assume specific variables are known or bounded to perform disturbance rejection and provide guarantees. In this way, robust controllers are designed to handle a pre-defined worst case scenario. While these methods provide guarantees on constraint satisfaction and stability, they can be overly conservative and usually don't account for system limitations. In addition, both adaptive and robust control techniques seek to eliminate the effects of un-modeled dynamics, even if they could be beneficial.

In contrast to reactive control strategies, predictive methods look ahead to anticipate the evolution of a system's dynamics, estimate future states, and select control actions to optimize the system's performance for current and future states [40]. Mathematically, this optimization problem seeks to minimize a given cost function that weighs the importance of multiple factors including tracking error, control effort, and constraint satisfaction. Optimization-based strategies provide versatility because they are shaped by a designated cost function and can be formulated to incorporate constraints. However, optimal control strategies are complex and computationally expensive to solve for nonlinear systems [25]. Consequently, optimal control strategies are difficult to run in real time on computationally constrained platforms. Model predictive control (MPC) is an optimal control strategy that is a balance between reactive controllers and infinite horizon optimal control methods [4].

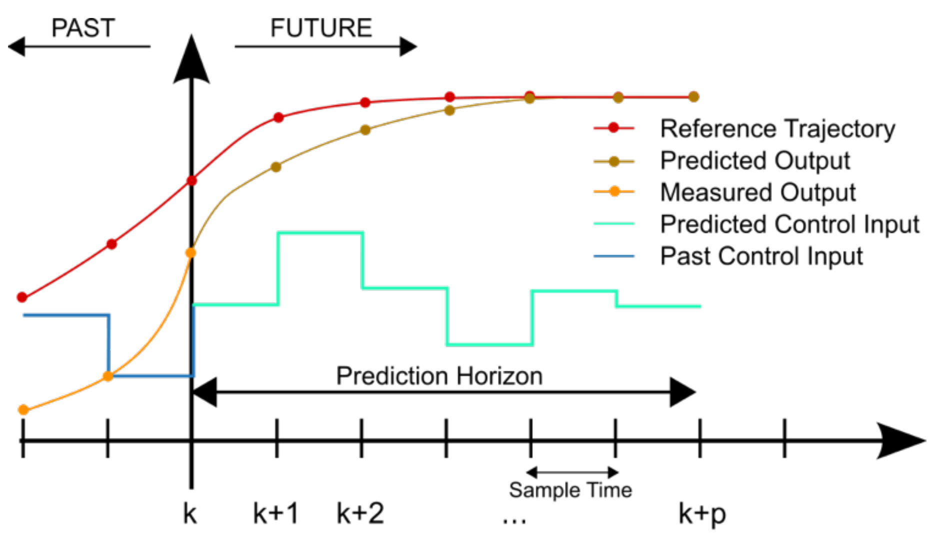

Figure 2.3: Model predictive control looks ahead over a finite horizon to estimate the evolution of the plant, and minimize cost function parameters such as tracking error and control effort. The plot illustrates the a horizon of length p over which MPC optimizes to compute a sequence of control inputs $\mathbf{u}$.

2.2 Model Predictive Control

Model predictive control formulates the control problem as a finite horizon optimization problem with constraints. The evolution of the system dynamics is predicted for an N-step horizon using the process model [2, 26], as shown in Fig. 2.3. For a linear system, or an affine approximation of a nonlinear system,

$$\mathbf{x_{k+1}} = \mathbf{A}\mathbf{x_k} + \mathbf{B}\mathbf{u_k}$$

When the state and input constraints are linear, they can be written as

$$\begin{aligned} \mathbf{G_x}\mathbf{x_{k+1}} &\leq \mathbf{g_x} \\ \mathbf{G_u}\mathbf{u_k} &\leq \mathbf{g_u} \end{aligned}$$

In this case, the optimization problem can be solved as a linear or quadratic program (QP) such as

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

in which $\mathbf{x}$ is the state vector, $\mathbf{r}$ is the desired state, and $\mathbf{u}$ is the optimal control input vector. The positive, semi-definite matrices $\mathbf{Q}$ and $\mathbf{R}$ are weights terms that penalize errors in tracking performance and deviation from the nominal control effort.

While this formulation can be used on nonlinear systems, an affine approximation of the system dynamics degrades the fidelity of the prediction model. Nonlinear model predictive control (NMPC) is more accurate than linear MPC that relies upon a linearized dynamics model. Still, a nonlinear system with possible nonlinear constraints results in a nonlinear programming problem (NLP) that incurs computational penalty. Formally, nonlinear system dynamics are described by

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

and constraints as

$$g(\mathbf{x_{k+1}}, \mathbf{u_k}) \leq 0$$

then the nonlinear programming problem computes the optimal control input $\{\mathbf{u_1}...\mathbf{u_N}\}$ given the current state $\mathbf{x_0}$ and references $\{\mathbf{r_1}...\mathbf{r_N}\}$ from the desired trajectory,

$$\begin{aligned} &\arg\min_{\bar{\mathbf{u}}_k} \sum_{k=0}^{N-1} J(\mathbf{x_{k+1}}, \mathbf{r_{k+1}}, \mathbf{u_k}) \\ \text{s.t.} \quad &\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \\ &g(\mathbf{x_{k+1}}, \mathbf{u_k}) \leq 0 \\ &\forall k = 0, ..., N-1 \end{aligned}$$

The differential equation is enforced by numerical integration. On each iteration, the first input in the control sequence is applied, and for each subsequent iteration the model predictive control problem is re-solved with the current state. In this work, we focus on nonlinear systems with dynamics well modeled by a continuous C^2 function.

2.2.1 Fast Nonlinear Model Predictive Control

To reduce the computational cost of solving a nonlinear program (NLP), there are a range of fast nonlinear model predictive control strategies used: fast online NMPC techniques, explicit offline NMPC methods, and semi-explicit NMPC methods.

Fast online nonlinear model predictive control strategies find solutions efficiently by using techniques such as warm-starting the optimization with a previous solution [15], trading off between speed and optimality [48], and exploiting the structure of the problem. Many techniques use convex approximations [13] or the iterative Linear Quadratic Regulator [45] to find solutions at high rates. These approaches are risky if the optimization cannot necessarily be solved at runtime on constrained platforms.

Explicit MPC and NMPC methods populate a database of controllers in an offline phase, removing the need for online optimization [2, 5]. The drawback of explicit approaches is that generating controllers for every scenario often results in a large database that grows exponentially based on the number of constraints [15]. Searching a large databases can be prohibitively expensive. Furthermore, such databases may contain controllers for scenarios which are improbable for operation on the actual system. One method to improve the efficiency of explicit MPC uses search trees to traverse a reduced state space[18]. Other approaches use function approximation to define a mapping from state to control output to supplant the need for a controller database and partition state space to simplify controller selection [41].

Explicit model predictive control is a subset of a larger class of methods that generate control policies offline. The LQR-Tree algorithm constructs a set of stabilizing local LQR control laws to achieve probabilistic coverage of state space using sum-of-squares optimization [43, 36]. Another approach uses reinforcement learning for an MPC-guided policy search, approximating the optimal control policy by training a neural network [50]. This strategy is limited by the availability of training examples, and thus often focuses on learning specific behaviors [50] to generate a tractable number of control policies.

Semi-explicit MPC techniques are a balance between the online and explicit approaches, storing a controller database and relying on online optimization when an optimal controller is not found in the database. Partial enumeration [33, 34] is an incremental strategy that solves the optimization problem online when a suitable controller is not found, adding it to the existing database. The need-based approach is formulated only for linear systems. Desaraju et al. extended this approach to apply to nonlinear systems [9]. Another approach limits the size of the controller database to retain commonly used controllers, relying on interpolation at run-time to generate a suitable control law. The underlying concepts behind Partial Enumeration are similar to those of transfer learning and lifelong learning. These techniques are similar because they use knowledge from the past to select future actions, generalizing policies to different domains or tasks [32, 42]. Compared to Partial Enumeration, lifelong algorithms [44] differ by maintaining a set of bases to reconstruct task models when new data is received [39]. While Partial Enumeration learns from real experiences, learning-based approaches such as [3] use synthetic experiences to generate additional training data.

2.2.2 Model Predictive Control with Uncertainty

In real world environments, a fixed dynamics model may not accurately predict future states when there are modeling errors or unmodeled, potentially time-varying disturbances affecting the process. This has led to MPC formulations that compensate for different sources of uncertainty by updating the dynamics model or tightening constraints to still provide constraint satisfaction [49]. There are two broad categories of methods used to account for uncertainty: adaptive and robust formulations.

Adaptive MPC approaches estimate uncertainty in the system's dynamics and update the predictive model accordingly. They often assume a structured system model with some uncertain parameters. Such techniques often combine standard MPC with an online parameter estimator, such as a Luenberger observer or Kalman filter [7]. Rather than using parameter estimation to capture model uncertainty, learning-based function approximation techniques are semi-structured approaches that can be useful to describe more complex perturbations that occur. In this case, a structured system dynamics model is modified with a non-parametric, online learned component via a Gaussian process [31]. However, kernel-based approaches such as Locally Weighted Projection Regression (LWPR) [46] and Incremental Sparse Spectrum Gaussian Process Regression [19] scale better with training data, providing incremental updates for fast model learning [29].

The drawback of adaptive techniques is the absence of constraint satisfaction. These strategies are sensitive to delay between the disturbance estimator and its incorporation into the system dynamics. In contrast, robust MPC refines constraints explicitly to include uncertainty and can provide guarantees in the presence of bounded, unpredictable parameters [23, 22]. While robust methods tend to yield conservative controllers, these methods can handle high frequency noise in the state estimate and do not need a disturbance estimator to restrict growth in uncertainty. Techniques such as Tube MPC [14] can achieve better performance by using local feedback control to limit the grown in uncertainty as the system evolves. Another robust strategy modifies robustness bounds online based on the state estimate [37], thus improving performance by relying on a probabilistic representation and adjusting constraints accordingly.

Chapter 3

Experience-driven Predictive Control

In complex, real-world environments, a robot's dynamics change online and can adversely impact the vehicle's performance. To approximate perturbations to the system and adapt to changing conditions, Experience-driven Predictive Control (EPC) is a model predictive control strategy developed by Desaraju [10] that generates a high-rate, adaptive controller with guarantees on safety and constraint satisfaction. The controller improves its performance by constructing a two-part representation of past experiences. First, the system uses an online model adaptation strategy to predict the system's evolution and model uncertain, time-varying dynamics. Second, EPC constructs a database that maps the system states and references to locally optimal controllers. While nonlinear model predictive control often faces a trade-off between model accuracy and computational cost, this approach overcomes the challenge by leveraging non-parametric model learning for a high fidelity dynamics models and strategic storage of prior control laws.

3.1 Approach

In this section we present the Experience-driven Predictive Control algorithm as originally invented by Desaraju [10]. EPC solves NMPC control problems through a receding horizon formulation. Specifically, the system dynamics are linearized, and a quadratic cost function is modelled in the state error. Un-modeled disturbances are accounted for using a model learner, which can learn both errors in the modeled dynamics and external forces acting on the system. Each solution to the quadratic program optimization is optimal for some set of system states and control inputs as defined by the Karush-Kuhn-Tucker (KKT) conditions. EPC leverages this fact by retaining a database of controllers and selecting the control for which the KKT conditions are satisfied. If no such controller can be found, a new controller is added to the database via optimization of the cost function. While the optimization is solved, a relaxed form of the quadratic program is solved with additional slack variance to ensure problem feasibility, allowing for temporary constraint violation.

3.1.1 Online Model Adaptation

For a nonlinear dynamical system, the predictive model is $\mathbf{x_{k+1}} = \mathbf{f}(\mathbf{x_k}, \mathbf{u_k})$. Given the nomial state $(\mathbf{x}^*, \mathbf{u}^*)$, we define $\bar{\mathbf{x}} = \mathbf{x} - \mathbf{x}^*$ and $\bar{\mathbf{u}} = \mathbf{u} - \mathbf{u}^*$. A simpler, affine dynamics model is derived by taking the first-order Taylor series expansion of $\mathbf{f}(\mathbf{x_k}, \mathbf{u_k})$, evaluating it at the nominal state $(\mathbf{x}^*, \mathbf{u}^*)$. This results in a linear approximation of the dynamics:

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

The affine dynamics model simplifies the optimization problem from a nonlinear programming problem (NLP) into a quadratic program, assuming the constraints are linear. Although more computationally efficient, there is a tradeoff in the accuracy of the system model using a local approximation of the nonlinear dynamics. To compensate for this, perturbations to the nominal model are estimated using an online learned component, $\hat{\mathbf{p}}$, to capture modeling errors, such as nonlinearities and un-modeled exogenous forces affecting the system. The modified predictive dynamics model is given by

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

where $\mathbf{\tilde{c}} = \mathbf{c} + \mathbf{\hat{p}}$.

The online model learning strategy considered here is Locally Weighted Projection Regression (LWPR) [46]. The direct measurement of the perturbation model is defined as $\mathbf{p} = \mathbf{\bar{x}_{k+1}} - \mathbf{\bar{x}^{nom}_{k+1}}$ for a query point $\mathbf{z} = \begin{bmatrix} \mathbf{x_k^T} & \mathbf{u_k^T} \end{bmatrix}^T$. The prediction output by the model learner is $\mathbf{\hat{p}} = \begin{bmatrix} p_0, p_1, \ldots \end{bmatrix}^T$.

This model learner updates its estimate by partial least squares, and its computational complexity is linear in the number of inputs. It is viable for real-time operation and can perform incremental updates online, and the rate of adaptation to model changes is controlled by adjusting the effects of prediction error on weight for each basis.

This technique models a nonlinear function by a Gaussian-weighted combination of simple basis functions. These linear basis functions are calculated by taking a linear combination of the state prediction error projected into a lower dimensional space defined by projection direction vectors $\nu_\mathbf{r}$ and $\rho_\mathbf{r}$, as described in [46]. The projection subspace consists of those states on which the disturbances are believed to depend. For example, the disturbance may depend on the velocity of a vehicle but not the robot's orientation. LWPR computes the slope coefficients β_r for each projection direction and an offset β_0 to generate a prediction. The dynamics model is fitted in an element-wise fashion [29], so for the ith element in $\mathbf{p}$, local linear model j (with r_j

projection direction) is

$$\boldsymbol{\Psi}_j(\mathbf{z}) = \beta_0 + \begin{bmatrix}\beta_1, ..., \beta_{r_j}\end{bmatrix} \begin{bmatrix} \boldsymbol{\nu}_\mathbf{1}^\mathbf{T} \\ \boldsymbol{\nu}_\mathbf{2}^\mathbf{T}\mathbf{P_1} \\ \vdots \\ \boldsymbol{\nu}_{\mathbf{r_j}}^\mathbf{T}(\mathbf{P_1}\ldots\mathbf{P}_{\mathbf{r_j-1}}) \end{bmatrix} (\mathbf{z}-\mathbf{m_j}) \\ = \alpha_j + \boldsymbol{\beta}_j^T(\mathbf{z}-\mathbf{m_j}) \tag{3.1}$$

in which $\mathbf{P_r} = \mathbf{I} - \text{diag}(\boldsymbol{\rho}_\mathbf{r})\begin{bmatrix}\boldsymbol{\nu}_r, \ldots, \boldsymbol{\nu}_r\end{bmatrix}^T$ is the projection matrix onto the subspace of basis functions. The prediction model, which consists of N_i local models with weights w_j defined by a Gaussian kernel with mean $\mathbf{m_j}$ and covariance $\mathbf{D_j}$, may be written as

$$p_i(\mathbf{z}) = \frac{1}{W}\sum_{j=1}^{N_i} w_j(\mathbf{z})\boldsymbol{\Psi}_j(\mathbf{z}) \tag{3.2}$$

$$w_j(\mathbf{z}) = exp\big(-\frac{1}{2}(\mathbf{z}-\mathbf{m_j})^T\mathbf{D_j}(\mathbf{z}-\mathbf{m_j})\big) \tag{3.3}$$

$$W = \sum_{j=1}^{N_i} w_j(\mathbf{z}) \tag{3.4}$$

A low-pass filter is applied to the disturbance estimate before adding it to the predictive model. This allows LWPR to learn the perturbation model quickly without adversely impacting the system dynamics, ensuring the disturbance estimate varies at an accurate timescale for the system dynamics.

3.1.2 Receding Horizon Control

The receding horizon control formulation and controller parameterization detailed here is as originally derived by Desaraju in [8, 10]. In this model predictive control formulation, the deviation in the predicted state from the reference is minimized over N steps, where N is the horizon length. The optimization problem adheres to an affine dynamics model, where the state and the input are subject to linear constraints as described by $\mathbf{g_x}$ and $\mathbf{g_u}$. This allows the NMPC problem to be formulated as a quadratric program instead of a nonlinear programming problem.

The current state is defined to be the nominal state, $\mathbf{x}^* = \mathbf{x_0}$. At timestep k, $\bar{\mathbf{x}}_k$ is the predicted state, $\mathbf{r}_k$ is the reference state, and $\mathbf{u}_k$ is the control input. The diagonal matrix $\mathbf{Q}$ describes the cost of deviations from the reference state, and $\mathbf{R}$ penalizes control effort. The constrained QP is as follows:

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

$$\begin{aligned} \text{s.t.} \quad & \bar{\mathbf{x}}_{k+1} = \mathbf{A}\bar{x}_k + \mathbf{B}\bar{\mathbf{u}}_k + \tilde{\mathbf{c}} \\ & \mathbf{G_x}\bar{\mathbf{x}}_{k+1} \leq \mathbf{g_x} \\ & \mathbf{G_u}\bar{\mathbf{u}}_{k+1} \leq \mathbf{g_u} \\ & \forall k = 0, ..., N-1 \end{aligned}$$

The term $\mathbf{u}_{\hat{p}}$ is the nominal control input with an offset incorporated to account for the disturbance estimated by the model learner.

The summation in the QP shown above can be removed by condensing states, references, and control input terms to a vectorized form that encapsulates the entire horizon: $\boldsymbol{x} = \begin{bmatrix} \bar{\mathbf{x}}_1^T, \dots, \bar{\mathbf{x}}_N^T \end{bmatrix}^T$, $\boldsymbol{r} = \begin{bmatrix} \bar{\mathbf{r}}_1^T, \dots, \bar{\mathbf{r}}_N^T \end{bmatrix}^T$, $\boldsymbol{u} = \begin{bmatrix} \bar{\mathbf{u}}_0^T, \dots, \bar{\mathbf{u}}_{N-1}^T \end{bmatrix}^T$, $\boldsymbol{u}_{\hat{\boldsymbol{p}}} = \begin{bmatrix} \mathbf{u_{\hat{p}_0}}^T, \dots, \mathbf{u_{\hat{p}_{N-1}}}^T \end{bmatrix}^T$. Similarly, the evolving system dynamics, cost matrices, and linear constraints can be formulated as block matrices:

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

where $\boldsymbol{\mathcal{Q}} = \mathrm{diag}(\mathbf{Q_1}, \dots, \mathbf{Q_N})$, $\quad \boldsymbol{\mathcal{R}} = \mathrm{diag}(\mathbf{R_0}, \dots, \mathbf{R_{N-1}})$, $\quad \boldsymbol{\mathcal{G}}_{\boldsymbol{x}} = \mathrm{diag}(\mathbf{G_{x_1}}, \dots, \mathbf{G_{x_N}})$,

$$\boldsymbol{\mathcal{G}}_{\boldsymbol{u}} = \mathrm{diag}(\mathbf{G}_{\mathbf{u_0}}, \ldots, \mathbf{G}_{\mathbf{u_{N-1}}}), \quad \boldsymbol{g}_{\boldsymbol{x}} = \left[\mathbf{g}_{\mathbf{x_1}}^{\mathbf{T}}, \ldots, \mathbf{g}_{\mathbf{x_N}}^{\mathbf{T}}\right]^T, \text{ and } \quad \boldsymbol{g}_{\boldsymbol{u}} = \left[\mathbf{g}_{\mathbf{u_0}}^{\mathbf{T}}, \ldots, \mathbf{g}_{\mathbf{u_{N-1}}}^{\mathbf{T}}\right]^T.$$

Note that $\bar{\mathbf{x}}_{\mathbf{0}} = \mathbf{0}$. Then, the QP can be rewritten as

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

Next, we can write an equivalent QP by substituting the dynamics constraints and dropping constant terms in the cost function to yield

$$\begin{aligned} \arg\min_{u} \frac{1}{2}\boldsymbol{u}^{\boldsymbol{T}}\boldsymbol{\mathcal{H}}\boldsymbol{u} + \boldsymbol{h}^{\boldsymbol{T}}\boldsymbol{u} \\ \text{s.t.} \quad \boldsymbol{\Gamma}\boldsymbol{u} \leq \gamma \end{aligned}$$

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

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

By defining $\boldsymbol{\lambda}$ as the vector of Lagrange multipliers and $\boldsymbol{\Lambda} = \mathrm{diag}(\boldsymbol{\lambda})$, the first two Karush-Kuhn-Tucker conditions for optimality, stationarity and complementary slackness are formulated as

$$\begin{aligned} \boldsymbol{\mathcal{H}}\boldsymbol{u} + \boldsymbol{h} + \boldsymbol{\mathit{\Gamma}}^{\boldsymbol{T}}\boldsymbol{\lambda} = \mathbf{0} \\ \boldsymbol{\mathit{\Lambda}}(\boldsymbol{\Gamma}\boldsymbol{u} - \gamma) = \mathbf{0} \end{aligned} \tag{3.5}$$

A constraint is deemed active if its corresponding Lagrange multipliers $\lambda_i > 0$. Solutions to equation 3.5 may be uniquely defined by the active constraints, meaning the control input $\boldsymbol{u}$ and

$\boldsymbol{\lambda}$ may be reconstructed by solving the following linear system

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

The resulting QP control law (assuming the active constraints are linearly independent) is affine in the predicted state error $\boldsymbol{r}$ and parameterized by the system dynamics

$$\boldsymbol{u} = \boldsymbol{\mathcal{E}}_5 \boldsymbol{r} - \left(\boldsymbol{\mathcal{E}}_5 \boldsymbol{c} - \boldsymbol{\mathcal{E}}_4 \boldsymbol{\mathcal{R}} \boldsymbol{u}_{\hat{\mathbf{p}}} + \boldsymbol{\mathcal{E}}_3 \begin{bmatrix} \boldsymbol{g}_{\mathbf{x}}{}^{+} - \boldsymbol{\mathcal{G}}_{\mathbf{x}} \boldsymbol{c} \\ -\boldsymbol{g}_{\mathbf{x}}{}^{-} - \boldsymbol{\mathcal{G}}_{\mathbf{x}} \boldsymbol{c} \\ \boldsymbol{g}_{\mathbf{x}}{}^{+} \\ -\boldsymbol{g}_{\mathbf{x}}{}^{-} \end{bmatrix}_a \right) \tag{3.6}$$

where

$$\boldsymbol{\mathcal{E}}_1 = \boldsymbol{\Gamma}_a \boldsymbol{\mathcal{H}}^{-1} \qquad \boldsymbol{\mathcal{E}}_2 = -(\boldsymbol{\mathcal{E}}_1 \boldsymbol{\Gamma}_a^T)^{-1} \qquad \boldsymbol{\mathcal{E}}_3 = \boldsymbol{\mathcal{E}}_1^T \boldsymbol{\mathcal{E}}_2$$

$$\boldsymbol{\mathcal{E}}_4 = \boldsymbol{\mathcal{H}}^{-1} + \boldsymbol{\mathcal{E}}_3 \boldsymbol{\mathcal{E}}_1 \qquad \boldsymbol{\mathcal{E}}_5 = \boldsymbol{\mathcal{E}}_4 \boldsymbol{\mathcal{B}}^T \boldsymbol{\mathcal{Q}}$$

The coefficients in (3.6) are functions of the affine dynamics model, so the overall control law $\kappa(\mathbf{x_0}, \mathbf{r_1}, ..., \mathbf{r_N})$ can be written in terms of a parameterized feedback gain matrix $\mathbf{K}$ and feedforward vector $\mathbf{k_{ff}}$

$$\kappa(\mathbf{x_0}, \mathbf{r_1}, ..., \mathbf{r_N}) = \mathbf{K}(\mathbf{A}, \mathbf{B}, \tilde{\mathbf{c}}) \boldsymbol{r} + \mathbf{k_{ff}}(\mathbf{A}, \mathbf{B}, \tilde{\mathbf{c}}) \tag{3.7}$$

This parameterization also applies to the KKT condition checks that decide whether a previously computed controller is locally optimal. The active Lagrance multipliers $\boldsymbol{\lambda_a}$ follow a similar form to the control input

$$\boldsymbol{\lambda}_a = -\boldsymbol{\mathcal{E}}_6 \boldsymbol{r} + \left(\boldsymbol{\mathcal{E}}_6 \boldsymbol{c} - \boldsymbol{\mathcal{E}}_3^T \boldsymbol{\mathcal{R}} \boldsymbol{u}_{\hat{\boldsymbol{p}}} + \boldsymbol{\mathcal{E}}_2 \begin{bmatrix} \boldsymbol{g}_{\mathrm{x}}{}^{+} - \boldsymbol{\mathcal{G}}_{\mathrm{x}} \boldsymbol{c} \\ -\boldsymbol{g}_{\mathrm{x}}{}^{-} - \boldsymbol{\mathcal{G}}_{\mathrm{x}} \boldsymbol{c} \\ \boldsymbol{g}_{\mathrm{x}}{}^{+} \\ -\boldsymbol{g}_{\mathrm{x}}{}^{-} \end{bmatrix}_a \right) \tag{3.8}$$

in which $\boldsymbol{\mathcal{E}}_6 = \boldsymbol{\mathcal{E}}_3^T \boldsymbol{\mathcal{B}}^T \boldsymbol{\mathcal{Q}}$. Due to this parameterization, the affine controller gains and Lagrange multipliers do not need to be stored in the controller database. Only the set of active constraints needs to be stored in the controller library. The current affine dynamics model and reference trajectory can be plugged in to apply any controller in the database, and check whether it is locally optimal via the reconstructed active Lagrance multipliers and KKT condtions. This method of storing controllers is memory efficient, making it useful for constrained systems to keep a controller database and adapt each feedback control law as the system evolves.

3.1.3 Experience-driven Predictive Control Algorithm

The EPC algorithm constructs a database that maps from experiences to controllers [8]. Experience is defined to be a library of controllers that each take the current state, reference, and affine dynamics model as input. As output, each controller provides a corresponding feedback control law. As published in [10], Algorithm 3.1.3 formalizes how a controller library is constructed, queried, and selects individual controllers.

At the start of each control iteration, EPC queries for the current state, reference trajectory, and current affine dynamics model (updated by LWPR). Then, EPC iterates through its controller library, querying the parameterized mappings to compute the corresponding feedback control input $\mathbf{u}$ and hypothetically active Lagrange multipliers, $\boldsymbol{\lambda}$. These terms are used to check the KKT conditions, which determine whether a particular controller is locally optimal. If these conditions are met, that feedback control law is applied. If none of the existing controllers are applicable, the receding horizon control problem, formulated as a QP, is solved to compute a new controller mapping. Thereafter, this mapping is added to the database. In the interim, an intermediate controller is used. This intermediate controller is a shorter horizon QP with slack variables augmented to the state constraints. The intermediate controller can be considered a fallback controller, as it enables EPC to operate in real time. An overview of the controller library process is shown in Figure 3.1.

The controller database prioritizes storage of frequently used controllers and limits its size to be memory efficient. To speed up the process of iterating through the controller database, a transition matrix is used to track likely successors to each controller.

By maintaining a controller library that leverages past experiences to reuse and adapt locally optimal feedback control laws, EPC is a computationally efficient model predictive control technique that addresses the challenges of NMPC. Model accuracy is maintained through the use of online model adaptation, and online optimization is reduced via controller reuse. Compared to Nonlinear Partial Enumeration [8], EPC has faster solution times by solving a QP instead of an NLP. Therefore, the faster speed at which new controllers are computed allows EPC to rely less on the intermediate controller and increase the prediction horizon.

Desaraju [8] developed an algorithm to construct controller databases offline using synthetically experience via Rapidly-exploring Random Trees. He also proposed a robust extension to EPC [11, 12] that uses Tube MPC [49] to tighten constraints in the presence of state uncertainty.

3.2 Quadrotor Dynamics Model and Trajectory Tracking

Quadrotor vehicles have six degrees of freedom corresponding to position and orientation in three dimensions. In our state formulation, position is defined as the translation (x,y,z) measured from the world coordinate frame W. Orientation is parameterized by Tait-Bryan Euler angles (roll, pitch, yaw) that follow the Z-Y-X rotation convention. The position and orientation are used to define the pose of the body-fixed coordinate frame B with respect to the world coordinate

Algorithm 1 Experience-driven Predictive Control

$\mathcal{M} \leftarrow \emptyset$ or $\mathcal{M}_{prior}$
while control is enabled **do**
    $\boldsymbol{x} \leftarrow$ current system state
    $\boldsymbol{r} \leftarrow$ current reference sequence
    A, **B**, $\tilde{\mathbf{c}} \leftarrow$ current dynamics model from LWPR
    for each element $m_i \in \mathcal{M}$ **do**
        Compute $\boldsymbol{u}, \boldsymbol{\lambda}$ via (5),(7)
        if $\boldsymbol{x}, \boldsymbol{r}$ satisfy parameterized KKT criteria **then**
            `importance` $\leftarrow$ current time, sort $\mathcal{M}$
            `solution_found` $\leftarrow$ true
            Apply affine control law (6) from m_i
        end if
    end for
    if `solution_found` is false **then**
        Apply intermediate control via (3) with slack variables
        Update QP formulation with current model
        Solve QP (3) to generate new controller
        if $|\mathcal{M}| >$ max table size **then**
            Remove element from $\mathcal{M}$ with minimum `importance`
        end if
        Add new element $m_{new} = (\mathbf{x}_0, \mathbf{K}_1, \mathbf{K}_2, \mathbf{k}_{ff},$ `importance`$)$ to $\mathcal{M}$
    end if
end while

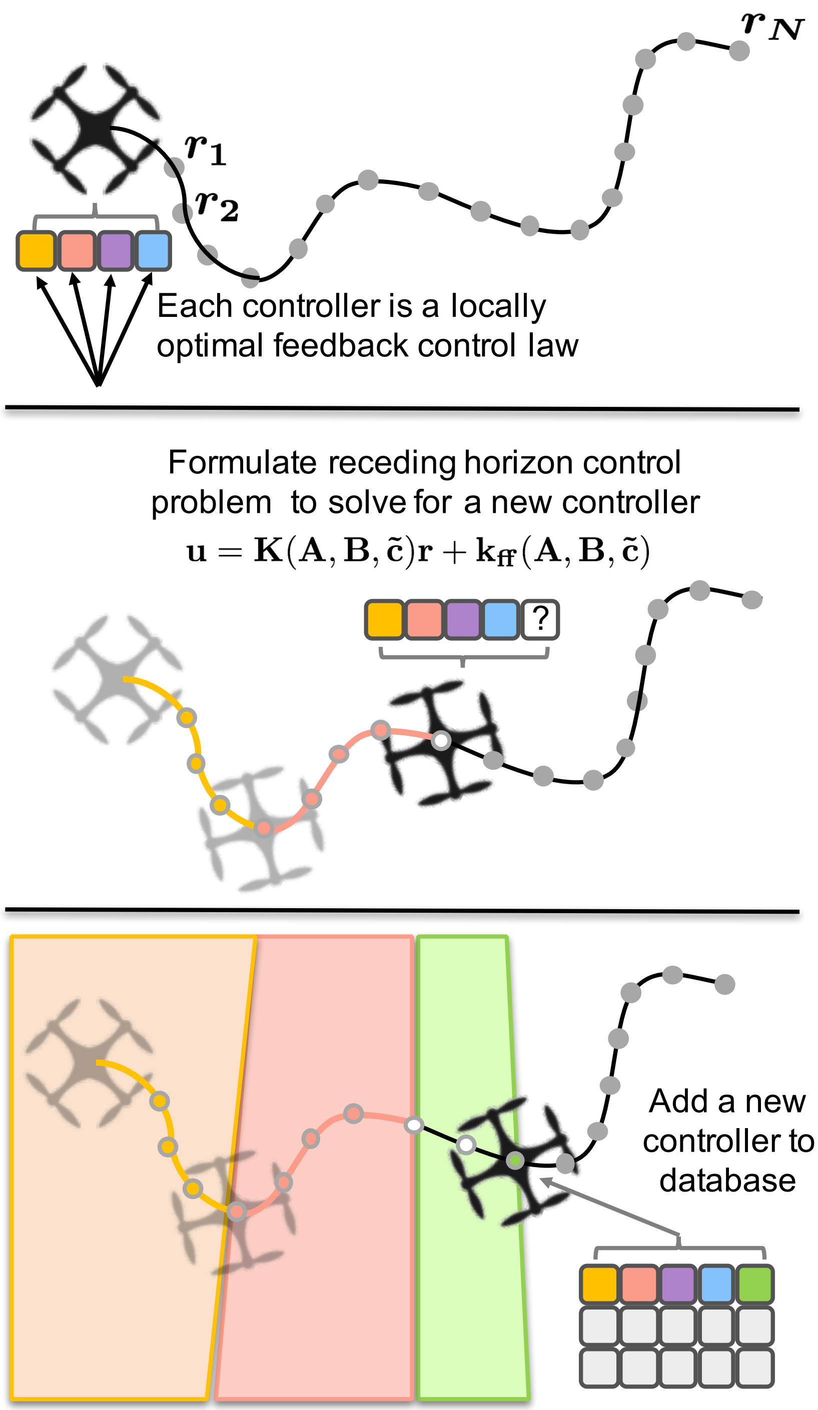

Figure 3.1: **Top:** Each control iteration, EPC checks the controller database to see if any of the existing entries is locally optimal. **Middle:** If a suitable controller is not found, the formulated QP is solved to compute a new controller. **Bottom:** The newly computed controller is added to the controller database.

frame W. The full state of the quadrotor also includes the translational velocities $(\dot{x}, \dot{y}, \dot{z})$ in the world frame W and the rotational velocities in the body frame (p,q,r) in B. Therefore, the dynamic model of the quadrotor has 12 states. The coordinate frames and free body diagram of a quadrotor are shown in Fig. 3.2. The state formulation, cascaded controller design, and equations used for 3D trajectory control are reproduced from [27].

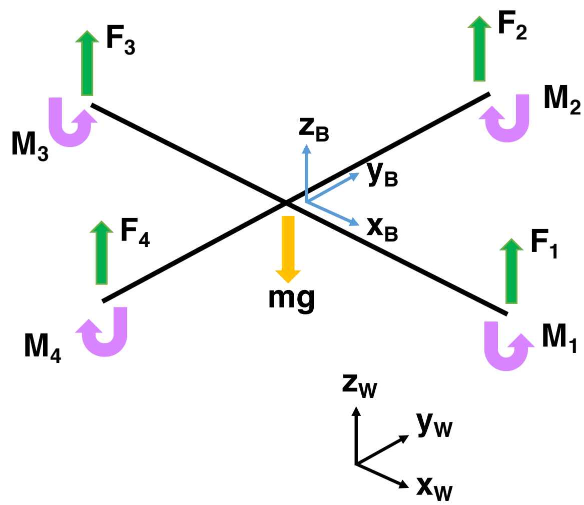

Figure 3.2: The coordinate systems and free body diagram for a quadrotor, showing the forces and moments acting on the vehicle. This diagram is adapted from [27].

3.2.1 State Formulation

To track the position of the quadrotor, the vehicle's position needs to be converted from the body coordinate frame of the quadrotor to the world frame. To change reference frames, a 3D rotation matrix can be constructed by rotating individually about the three coordinate axis. The composition of these three 2D rotation matrices results in the 3D rotation matrix between the world and body coordinate frames.

Within the body frame, the angles are defined by the yaw angle ψ, roll angle ϕ and pitch angle θ. The transformation from the body frame B to the world frame W is defined by

$$\mathbf{R} = \begin{bmatrix} c\psi c\theta - s\phi s\psi s\theta & -c\phi s\psi & c\psi s\theta + c\theta s\phi s\psi \\ c\theta s\psi + c\psi s\phi s\theta & c\phi c\psi & s\psi s\theta - c\psi c\theta s\phi \\ -c\phi s\theta & s\phi & c\phi c\theta \end{bmatrix}$$

in which $c\theta$ and $s\theta$ denote $\cos(\theta)$ and $\sin(\theta)$ respectively. This also follows for the yaw angle ψ and roll angle ϕ.

To track the acceleration of the quadrotor, the vector $\mathbf{r}$ describes the position of the center of mass of the quadrotor in the world frame. Newton's second law of motion, $\mathbf{F} = m\mathbf{a}$ describes the acceleration of the center of mass.

$$m\ddot{\mathbf{r}} = m \begin{bmatrix} \ddot{\mathbf{x}} \\ \ddot{\mathbf{y}} \\ \ddot{\mathbf{z}} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + \mathbf{R} \begin{bmatrix} 0 \\ 0 \\ \sum F_i \end{bmatrix} \tag{3.9}$$

The forces shown in the free body diagram show F_i the vertical forces from each motor and gravity acting on the vehicle. For each motor, the rotor has an angular speed ω_i and produces a vertical force $F_i = k_F \omega_i^2$.

Each rotor also produces a moment perpendicular to the plane of rotation of the blades. Rotors 1 and 3 rotate in the $-z_B$ direction while rotors 2 and 4 rotate in the z_B direction. So, the moments M_1 and M_3 act in the z_B direction while M_2 and M_4 act in the $-z_B$ direction. The moment produced by each rotor is $M_i = k_M \omega_i^2$. Then, the angular acceleration of the quadrotor is determined by the Euler equations for a rigid body.

$$\mathbf{I} \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} L(F_2 - F_4) \\ L(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \mathbf{I} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{3.10}$$

in which L is the distance from the axis of rotation of the rotors to the center of the vehicle. The components of angular velocity of the robot in the body frame are p, q and r. $\mathbf{I}$ is the moment of inertia matrix of the quadrotor.

3.2.2 Cascaded Control

The quadrotor is controlled by attitude and position control loops. The inner control loop controls the pitch, yaw and roll of the vehicle based on sensor readings from the accelerometer and gyroscopes onboard the vehicle. These sensor readings are usually noisy, and thus sensor models may assume some noise distribution, e.g., Gaussian white noise. The outer position control loop uses the estimated position and velocity of the center of mass of the aerial vehicle to control the 3D trajectory. The position controller computes the desired orientations and passes the attitude references to the inner loop, as shown in Fig. 3.3.

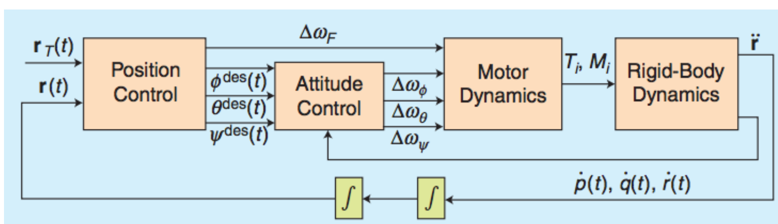

Figure 3.3: Nested control loops for attitude and position control, following the cascaded model described in [27].

The control inputs for the system are the angular speeds ω_i. These desired rotor speeds can be described as a linear combination of four terms:

$$\begin{bmatrix} \omega_1^{des} \\ \omega_2^{des} \\ \omega_3^{des} \\ \omega_4^{des} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 & 1 \\ 1 & 1 & 0 & -1 \\ 1 & 0 & 1 & 1 \\ 1 & -1 & 0 & -1 \end{bmatrix} \begin{bmatrix} \omega_h + \Delta\omega_F \\ \Delta\omega_\phi \\ \Delta\omega_\theta \\ \Delta\omega_\psi \end{bmatrix} \tag{3.11}$$

In the above expression, w_h is the nominal rotor speed to hover in steady state. The $\triangle\omega_F$

term results in a net force along the z_B axis. $\triangle\omega_\phi$, $\triangle\omega_\theta$, and $\triangle\omega_\psi$ result in moments that cause pitch, roll and yaw. Near the hover state, $\dot{\phi} \approx p$, $\dot{\theta} \approx q$, $\dot{\psi} \approx r$.

3.2.3 3D Trajectory Control

The position of the quadrotor in the x_W and y_W planes are controlled using the pitch θ and roll ϕ angles. The $\triangle\omega_\psi$ equation from the previous section controls the yaw angle, while $\triangle\omega_F$ controls the position of the vehicle along the z_W axis.

Linearizing the equation describing the acceleration of the center of mass of the quadrotor results in

$$\begin{aligned}
\ddot{r}_1^{des} &= g(\theta^{des}\cos\psi_T + \phi^{des}\sin\psi_T) \\
\ddot{r}_2^{des} &= g(\theta^{des}\sin\psi_T - \phi^{des}\cos(\psi_T) \\
\ddot{r}_3^{des} &= \frac{8k_F\omega_h}{m}\Delta\omega_F
\end{aligned} \tag{3.12}$$

The above equations describe the relationship between the desired accelerations and the roll and pitch angles. Inverting these equations results in equations for the desired roll and pitch angles for the attitude controller as well:

$$\begin{aligned}
\phi^{des} &= \frac{1}{g}(\ddot{r}_1^{des}\sin\psi_T - \ddot{r}_2^{des}\cos\psi_T) \\
\theta^{des} &= \frac{1}{g}(\ddot{r}_1^{des}\cos\psi_T + \ddot{r}_2^{des}\sin\psi_T) \\
\Delta\omega_F &= \frac{m}{8k_F\omega_h}\ddot{r}_3^{des}
\end{aligned} \tag{3.13}$$

3.3 Results

To demonstrate the performance of EPC, we conduct simulation experiments for controlling a small micro air vehicle using EPC under several different conditions. The test compares the tracking performance of EPC to that of a proportional-derivative controller, for a quadrotor following a circular trajectory. Both control strategies use a cascaded control setup, simulating the inner attitude control loop at 200Hz and the outer position control loop at 100 Hz. The EPC controller is applied to both the translational and rotational control loops.

The tracking performance of PD versus EPC for the circular trajectory is shown in Fig. 3.4. Both controllers track the desired trajectory closely, and Fig. 3.5 shows the absolute tracking error of EPC is less than 2cm. The performance of EPC is comparable with the PD controller, the magnitude of the tracking error is larger than that of PD by less than a centimeter.

In another experiment, the micro air vehicle followed a more aggressive zig-zag trajectory with linear velocities of up to 3.75 meters per second. The quadrotor was commanded to track the reference trajectory in a wind field disturbance of 1.5 meters per second, in the $\frac{-3\pi}{4}$ direction in the XY plane. State constraints were applied to the system to bound the vehicle's translational velocity to less than 3.8 meters per second. Figure 3.6 shows the vehicle's y-velocity remained within the linear velocity state constraints.

To evaluate whether online model adaptation using LWPR improved the vehicle's flight performance, we simulated the robot following a figure 8 trajectory in a wind field of 3 m/s. Figure 3.7 shows the tracking performance of EPC with the system dynamics updated by LWPR, and EPC without model adaptation. LWPR is able to estimate the low-frequency perturbation in the environment and compensate to substantially reduce the vehicle's tracking error.

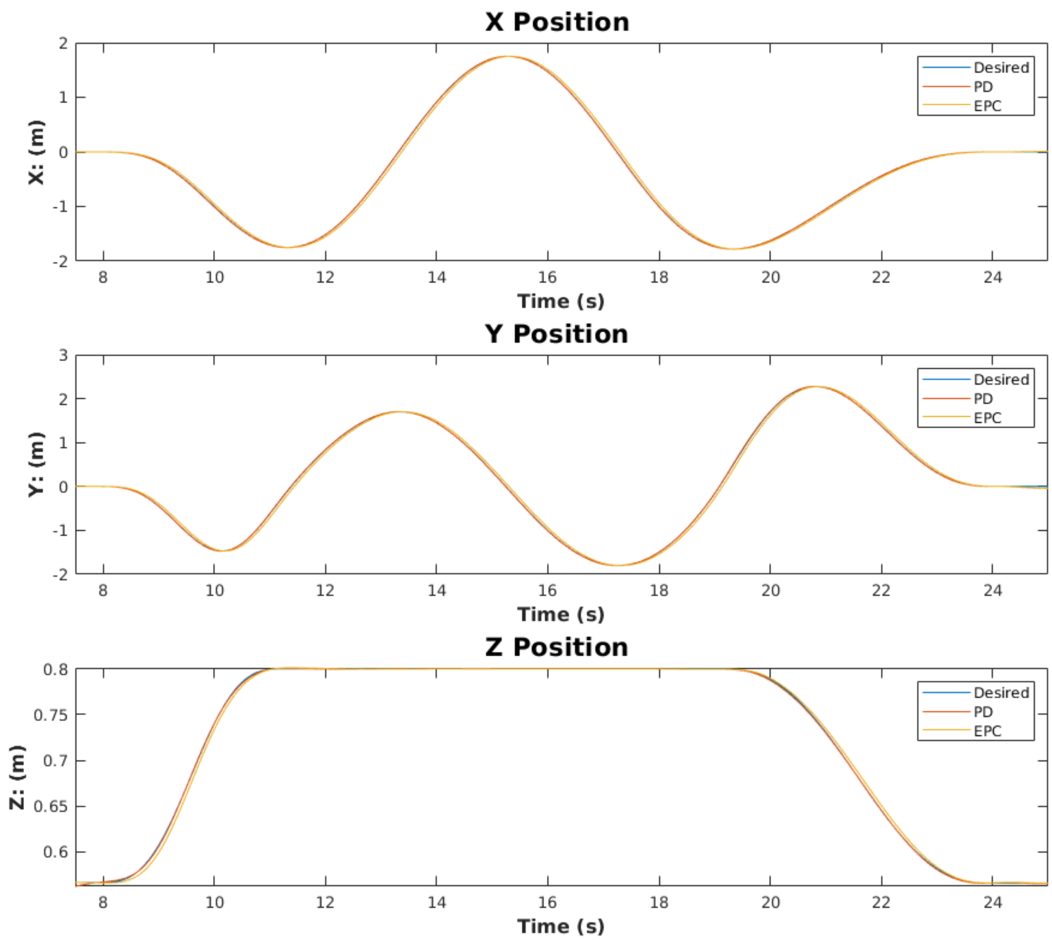

Figure 3.4: A quadrotor tracking a circular trajectory using both EPC and PD controllers.

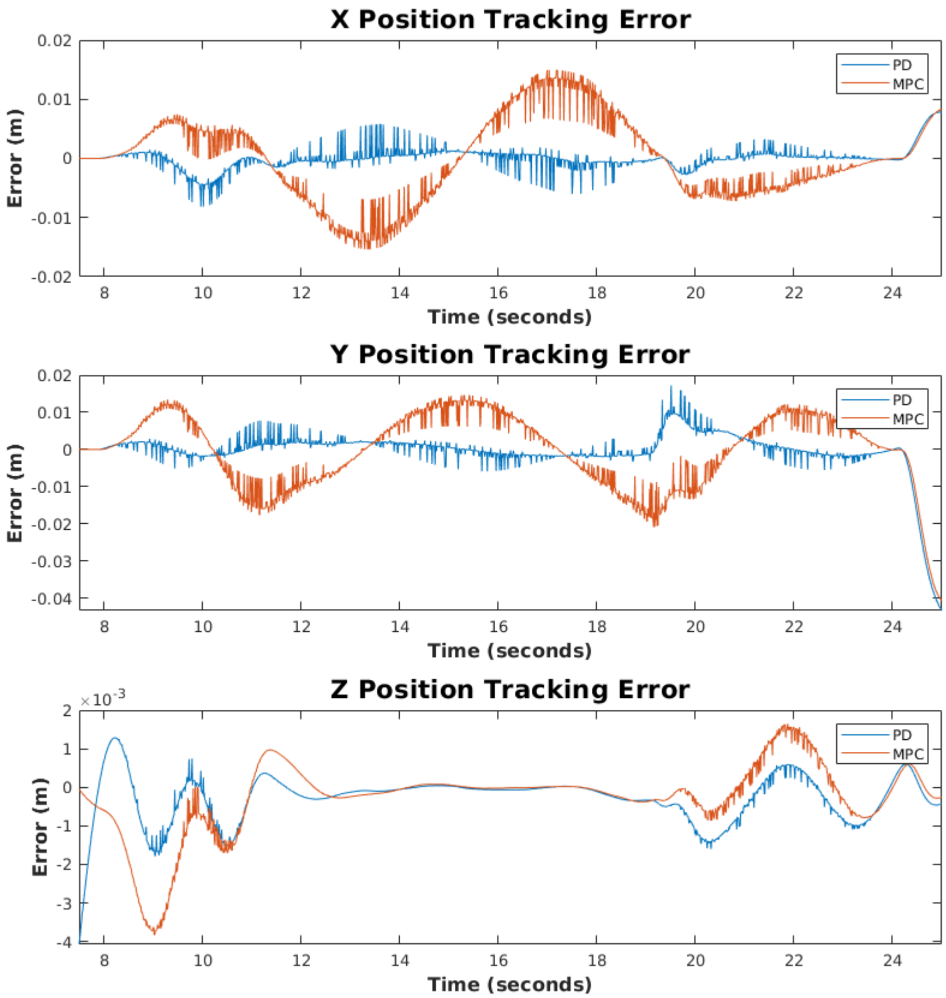

Figure 3.5: Tracking error for PD and EPC controllers for a quadrotor following a circular trajectory.

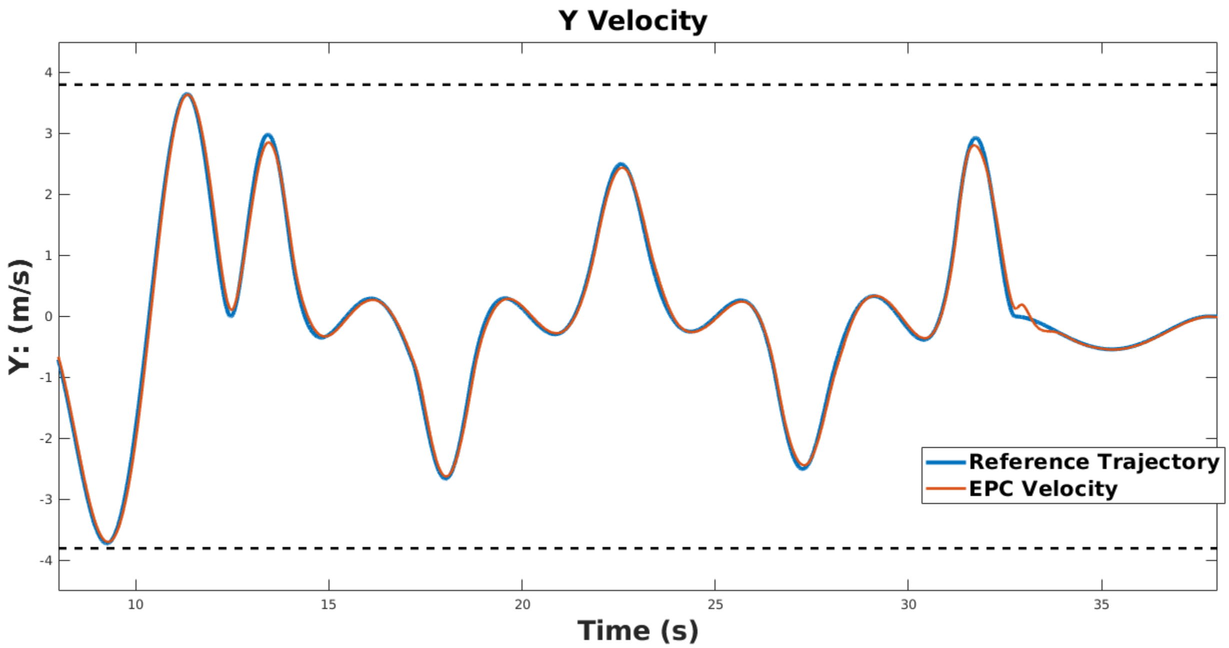

Figure 3.6: Y Velocity tracking for a quadrotor following a zig-zag trajectory. The vehicle is subject to a wind disturbance of 1.5 m/s in the $\frac{-3\pi}{4}$ direction in the XY plane. The EPC controller abides by velocity constraints to remain with the bounded velocity limits of 3.8 m/s.

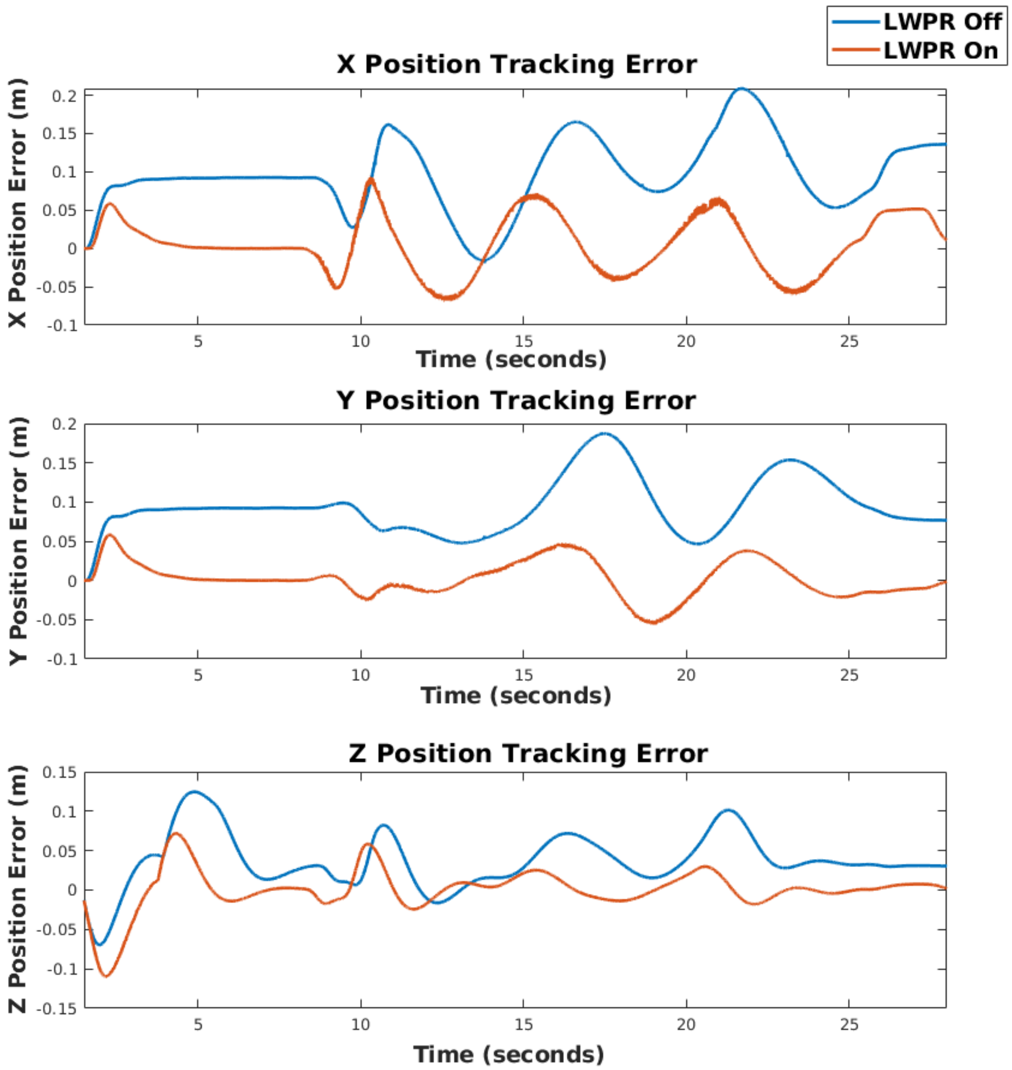

Figure 3.7: Position tracking for a quadrotor flying in a wind field with and without online model adaptation (Locally Weighted Projection Regression). The aerial robot is following a figure 8 trajectory in an environment with a wind disturbance of 3 m/s in the $\frac{\pi}{4}$ direction in the XY plane.

Chapter 4

Conclusion

This thesis investigates and tests a method for safe and accurate control of constrained nonlinear systems in environments with state uncertainty and unpredictable disturbances. Although nonlinear model predictive control is a powerful framework for predictive control of complex systems, a significant limitation of NMPC is the tradeoff between computational cost and model accuracy. Experience-driven Predictive Control is an efficient method that addresses this challenge by using past experiences to improve model accuracy and reduce the need for online optimization [10]. This enables EPC to be used on systems with size, weight, and power constraints. Overall, EPC provides greater control performance and robustness to unknown disturbances, making it well-suited for enabling infrastructure inspection via field robotics.

EPC has shown tracking performance that is comparable with that of a PD controller. Unlike a reactive PD controller, this technique guarantees constraint satisfaction in the presence of disturbances to the system. The semi-parametric learning of perturbations to the system allows for adaptation to low frequency disturbances and improvements in flight performance. With flight performance comparable to classical methods and assurances of safety, EPC is a efficient approach for micro air vehicles to navigate safely within complex indoor environments.

4.1 Future Work

There are multiple avenues for future work that build upon EPC. One area of interest is for transfer learning using EPC's controller database. The controller database can be transferred between robots to examine the safety, coverage, and generality provided by experience accumulated on another vehicle. Also, studying how controller usage varies across operating domains studies how past experiences can be transferred between tasks. Developing an understanding of how environmental factors, state uncertainty, and external disturbances impact controller selection would enable better coverage and safety guarantees.

Furthermore, aggregating controller libraries from multiple robots would pool experiences and increase coverage of the controller database rapidly. Increased coverage reduces the need for online optimization further and reduces calls to the intermediate controller. Evaluating different options for the intermediate controller would enhance the overall stability and performance of Experience-driven Predictive Control algorithm.

Appendix A

Classic Control Strategies

A.1 Proportional-Derivative Control

A simple and frequently used control strategy is proportional-derivative (PD) control. This strategy directly measures the error between the current and desired states as $e(t) = actual - desired$, and the rate of change in the error.

$$u = -K_p e(t) - K_d \frac{de(t)}{dt}$$

While PD controllers can be tuned manually and use heuristic techniques such as the Ziegler-Nichols method, such approaches rely heavily on a designer's intuition, and repeated trial-and-error. A viable strategy that does not rely on the user's expertise is pole placement. Pole placement analyzes the transfer function of a system in the frequency domain, and selects locations for the closed-loop poles such that the plant achieves desired performance characteristics [16].

A.2 Linear Quadratic Regulator

While model predictive control is the focus of this thesis, this work started with studies of the linear-quadratic regulator (LQR) problem. The linear quadratic regulator is one of the simplest optimal control strategies, in which the system dynamics are described by linear differential equations and the cost function is quadratic. LQR operates in a fixed time window, and uses a single optimal solution for the entire time horizon.

Comparatively, MPC optimizes over a receding horizon and a new solution is computed per control iteration. Because MPC optimizes over a finite time window, it may yield a suboptimal solution. However, MPC makes no assumptions of linearity of the system and it can therefore handle hard constraints. This also enables MPC to be applied to nonlinear systems away from its linearized operating point, unlike LQR.

With respect to Experience-driven Predictive Control, the quadratic programming problem described in this approach is similar to LQR. When there are no state or input constraints imposed, EPC is the same as finite-horizon, discrete-time LQR. This relationship has proven invaluable for gain tuning of the $\mathbf{Q}$ and $\mathbf{R}$ cost matrices used in the EPC cost function. By solving the Algebraic Riccati equation, theoretically equivalent gains can be computed that relate to a tuned gain matrix K. The steps to derive the Algebraic Ricatti Equation from a quadratic cost function for the continuous time, infinite horizon case are shown [17].

Nonlinear systems can be linearized to produce a simplified predictive model. This allows the performance to be measured in a quadratic cost function. The dynamics of a linear system can be described by a vector-matrix differential equation:

$$\dot{x} = Ax + Bu$$

The performance criterion are described by the value function

$$V = \int_t^T x^T(\tau)Q(\tau)x(\tau) + u^T(\tau)Ru(\tau)d\tau \tag{A.1}$$

The first term in the above expression penalizes deviations of the state from the origin, and the second term limits the magnitude of the linear control signal $u(t) = -Kx(t)$ to prevent saturation and select inputs within the normal range of operation. When the linear control law is applied, the closed loop dynamics are

$$\dot{x} = Ax - BKx = A_c x \tag{A.2}$$

where $A_c = A - BG$. Since A_c may be time varying, the solution to the differential equation cannot be written as a matrix exponential. Instead, a general state transition matrix can be used:

$$x(\tau) = \phi_c(\tau, t)x(t)$$

ϕ_c is the state transition matrix which corresponds to A_c. The performance index can then be expressed in terms of the initial state

$$\begin{aligned} V &= \int_t^T x^T(\tau)Q(\tau)x(\tau) + u^T(\tau)Ru(\tau)d\tau \\ &= \int_t^T x^T(t)\phi_c^T(\tau,t)\{Q + K^T RK\}\phi_c(\tau,t)x(t)d\tau \end{aligned} \tag{A.3}$$

The initial state $x(t)$ can be moved outside the integral to yield

$$\begin{aligned} V &= x^T(t)M(t,T)x(t) \\ M(t,T) &= \int_t^T \phi_c^T(\tau,t)\{Q + K^T RK\}\phi_c(\tau,t)d\tau \end{aligned} \tag{A.4}$$

The cost function in equation (A.3) can be rewritten as

$$\begin{aligned} V(t) &= \int_t^T x^T(\tau)L(\tau)x(\tau)d\tau \\ L(\tau) &= Q + K^T RK \end{aligned} \tag{A.5}$$

The derivative of the cost function in (A.5) is

$$\frac{dV}{dt} = -x^T(\tau)Lx(\tau)|_{\tau=t} = -x^T(t)Lx(t) \tag{A.6}$$

From equation (A.4), the derivative of V is also

$$\frac{dV}{dt} = \dot{x}^T(t)M(t,T)x(t) + x^T(t)\dot{M}(t,T)x(t) + x^T(t)M(t,T)\dot{x}(t) \tag{A.7}$$

$$\dot{M}(t,T) = \frac{\partial M(t,T)}{\partial t} \tag{A.8}$$

Plugging in the closed loop differential equation in (A.2),

$$\frac{dV}{dt} = x^T(t)\{A_c^T(t)M(t,T) + \dot{M}(t,T) + M(t,T)A_c(t)\}x(t) \tag{A.9}$$

With two expressions for $\frac{dV}{dt}$, the quadratic forms in equations (A.15) and (A.9) can be equal if the underlying matrices are equal. This yields the first-order matrix differential equation

$$\begin{aligned} -L &= A_c M + \dot{M} + M A_c \\ -\dot{M} &= M A_c + A_c^T M + L \end{aligned} \tag{A.10}$$

The arguments are omitted, in the above expression, however,

$$M = M(t,T) \quad A_c = A_c(t) \quad L = L(t)$$

The solution to (A.10) is given in (A.4), and fully defined by the initial condition $M(T,T) = 0$. When the expressions for A_c and L are plugged into the differential equation,

$$-\dot{M} = M(A - BK) + (A^T - K^T B^T)M + Q + K^T RK \tag{A.11}$$

The objective is to minimize the solution to the differential equation for $\dot{M}$. Finding the matrix K that minimizes this quantity also minimizes the cost function $V = x^T M x$. Let us assume that the minimizing gain $K = \hat{K}$ exists and results in a minimum $\hat{M}$. Any other non-optimal gain matrix K and corresponding matrix M can be expressed as:

$$M = \hat{M} + N$$
$$K = \hat{K} + Z$$

Used within the differential equation for $\dot{M}$,

$$-(\dot{\hat{M}} + \dot{N}) = (\hat{M} + N)(A - B(\hat{K} + Z)) + (A^T - (\hat{K}^T + Z^T)B^T)(\hat{M} + N) + Q + (\hat{K}^T + Z)R(\hat{K} + Z) \tag{A.12}$$

Subtracting (A.11) from this expression,

$$-\dot{N} = NA_c + A_c^T N + (\hat{K}^T R - \hat{M}B)Z + Z^T(R\hat{K} - B^T\hat{M}) + Z^T RZ \tag{A.13}$$

in which $A_c = A - BK = A - B(\hat{K} + Z)$. The above expression matches the form of equation (A.10), with L instead being

$$L = (\hat{K}^T R - \hat{M}B)Z + Z^T(R\hat{K} - B^T\hat{M}) + Z^T RZ \tag{A.14}$$

Similarly, the solution to the matrix differential equation (A.11) takes the form

$$N(t, T) = \int_t^T \phi_c^T(\tau, t) L \phi_c(\tau, t) d\tau \tag{A.15}$$

Since $\hat{V}$ is the minimum of the cost function,

$$x^T \hat{M} x \leq x^T (\hat{M} + N) x = x^T \hat{M} x + x^T N x \tag{A.16}$$

The last term indicates that the expression $x^T N x$ must be positive definite or positive semidefinite. Looking at the expression for L in (A.15), the linear terms in the expression must be zero for the control law $\hat{K}$ to be optimum. As a result,

$$R\hat{K} - B^T \hat{M} = 0 \tag{A.17}$$

$$\hat{K} = R^{-1} B^T \hat{M} \tag{A.18}$$

When the optimal value for K is plugged into equation (A.11)

$$-\dot{M} = MA + A^T M + MBR^{-1} B^T M + Q \tag{A.19}$$

The resultant differential equation is known as the Algebraic Ricatti Equation (ARE). Due to the quadratic term in the ARE, there is no general formula for the solution and often $\hat{M}(t, T)$ is solved for using numerical methods. Note: If (i) the system is asymptotically stable or (ii) the system defined by (A, B) is controllable and the system (A, C) where $C^T C = Q$ is observable, then the ARE has a *unique, positive definite* solution M which minimizes V_{inf} when the control law $u = -R^{-1} B^T M x$ is applied.

Bibliography

[1] Samer Abdelmoeti and Raffaella Carloni. "Robust control of UAVs using the parameter space approach". In: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE. 2016, pp. 5632–5637.

[2] Alessandro Alessio and Alberto Bemporad. "A survey on explicit model predictive control". In: *Nonlinear model predictive control*. Springer, 2009, pp. 345–369.

[3] Bram Bakker et al. "Quasi-online reinforcement learning for robots". In: *Proceedings of the International Conference on Robotics and Automation (ICRA-06)*. 2006.

[4] Alberto Bemporad and Manfred Morari. "Robust model predictive control: A survey". In: *Robustness in identification and control*. Springer, 1999, pp. 207–226.

[5] Alberto Bemporad et al. "The explicit linear quadratic regulator for constrained systems". In: *Automatica* 38.1 (2002), pp. 3–20.

[6] Samir Bouabdallah and Roland Siegwart. "Backstepping and sliding-mode techniques applied to an indoor micro quadrotor". In: *None*. LSA-CONF-2005-003. 2005.

[7] Patrick Bouffard, Anil Aswani, and Claire Tomlin. "Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results". In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE. 2012, pp. 279–284.

[8] Vishnu R Desaraju. "Safe, Efficient, and Robust Predictive Control of Constrained Nonlinear Systems". In: (2017).

[9] Vishnu R Desaraju and Nathan Michael. “Fast nonlinear model predictive control via partial enumeration”. In: *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE. 2016, pp. 1243–1248.

[10] Vishnu R Desaraju and Nathan Michael. “Leveraging experience for computationally efficient adaptive nonlinear model predictive control”. In: *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE. 2017, pp. 5314–5320.

[11] Vishnu R Desaraju, Alexander E Spitzer, and Nathan Michael. “Experience-driven predictive control with robust constraint satisfaction under time-varying state uncertainty”. In: *Robotics: Science and Systems Conference (RSS), July*. 2017.

[12] Vishnu R Desaraju et al. “Leveraging experience for robust, adaptive nonlinear MPC on computationally constrained systems with time-varying state uncertainty”. In: *The International Journal of Robotics Research* (2018), p. 0278364918793717.

[13] Moritz Diehl et al. “An efficient algorithm for nonlinear model predictive control of large-scale systems part I: Description of the method (Ein effizienter Algorithmus für die nichtlineare prädiktive Regelung großer Systeme Teil I: Methodenbeschreibung)”. In: *at-Automatisierungstechnik Methoden und Anwendungen der Steuerungs-, Regelungs-und Informationstechnik* 50.12/2002 (2002), p. 557.

[14] Morteza Farrokhsiar, Graham Pavlik, and Homayoun Najjaran. “An integrated robust probing motion planning and control scheme: A tube-based MPC approach”. In: *Robotics and Autonomous Systems* 61.12 (2013), pp. 1379–1391.

[15] Hans Joachim Ferreau, Hans Georg Bock, and Moritz Diehl. “An online active set strategy to overcome the limitations of explicit MPC”. In: *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal* 18.8 (2008), pp. 816–830.

[16] Gene F Franklin et al. *Feedback control of dynamic systems*. Vol. 3. Addison-Wesley Reading, MA, 1994.

[17] Bernard Friedland. *Control system design: an introduction to state-space methods*. Courier Corporation, 2012.

[18] Alexander N Fuchs, CN Jones, and Manfred Morari. “Optimized decision trees for point location in polytopic data sets-application to explicit MPC”. In: *American Control Conference (ACC), 2010*. IEEE. 2010, pp. 5507–5512.

[19] Arjan Gijsberts and Giorgio Metta. “Real-time model learning using incremental sparse spectrum gaussian process regression”. In: *Neural Networks* 41 (2013), pp. 59–69.

[20] Laurent P Houssay. “Robotics and radiation hardening in the nuclear industry”. PhD thesis. State University System of Florida, 2000.

[21] Evgeny Kharisov, Naira Hovakimyan, and Karl Åström. “Comparison of several adaptive controllers according to their robustness metrics”. In: *AIAA Guidance, Navigation, and Control Conference*. 2010, p. 8047.

[22] Yoshiaki Kuwata, Arthur Richards, and Jonathan How. “Robust receding horizon control using generalized constraint tightening”. In: *American Control Conference, 2007. ACC’07*. IEEE. 2007, pp. 4482–4487.

[23] Wilbur Langson et al. “Robust model predictive control using tubes”. In: *Automatica* 40.1 (2004), pp. 125–133.

[24] Daewon Lee, H Jin Kim, and Shankar Sastry. “Feedback linearization vs. adaptive sliding mode control for a quadrotor helicopter”. In: *International Journal of control, Automation and systems* 7.3 (2009), pp. 419–428.

[25] Jay H Lee. “Model predictive control: Review of the three decades of development”. In: *International Journal of Control, Automation and Systems* 9.3 (2011), p. 415.

[26] David Q Mayne. “Model predictive control: Recent developments and future promise”. In: *Automatica* 50.12 (2014), pp. 2967–2986.

[27] Nathan Michael et al. “The grasp multiple micro-uav testbed”. In: *IEEE Robotics & Automation Magazine* 17.3 (2010), pp. 56–65.

[28] R Minichan, R Fogle, and A Marzolf. *H-Canyon Air Exhaust Tunnel Inspection Vehicle Development*. Tech. rep. SRS, 2011.

[29] Djordje Mitrovic, Stefan Klanke, and Sethu Vijayakumar. "Adaptive optimal feedback control with learned internal dynamics models". In: *From Motor Learning to Interaction Learning in Robots*. Springer, 2010, pp. 65–84.

[30] LANDON MR and HANSON CE. *Development and Deployment of the Mobile Arm Retrieval System (MARS)-12187*. Tech. rep. Hanford Site (HNF), Richland, WA (United States), 2012.

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

[32] Sinno Jialin Pan, Qiang Yang, et al. "A survey on transfer learning". In: *IEEE Transactions on knowledge and data engineering* 22.10 (2010), pp. 1345–1359.

[33] Gabriele Pannocchia, James B Rawlings, and Stephen J Wright. "Fast, large-scale model predictive control by partial enumeration". In: *Automatica* 43.5 (2007), pp. 852–860.

[34] Gabriele Pannocchia, Stephen J Wright, and James B Rawlings. "Partial enumeration MPC: Robust stability results and application to an unstable CSTR". In: *Journal of Process Control* 21.10 (2011), pp. 1459–1466.

[35] Guilherme V Raffo, Manuel G Ortega, and Francisco R Rubio. "An integral predictive/nonlinear Hfffdfffdfffd control structure for a quadrotor helicopter". In: *Automatica* 46.1 (2010), pp. 29–39.

[36] Philipp Reist and Russ Tedrake. "Simulation-based LQR-trees with input and state constraints". In: *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE. 2010, pp. 5504–5510.

[37] Arthur Richards and Jonathan How. “Robust model predictive control with imperfect information”. In: *American Control Conference, 2005. Proceedings of the 2005*. IEEE. 2005, pp. 268–273.

[38] Rodrigo Rimando. *Developing EM Roadmap for Robotics and Remote Systems*. Apr. 2017.

[39] Avishek Saha et al. “Online learning of multiple tasks and their relationships”. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. 2011, pp. 643–651.

[40] Robert F Stengel. *Optimal control and estimation*. Courier Corporation, 1994.

[41] Sean Summers et al. “Fast Explicit Nonlinear Model Predictive Control Via Multiresolution Function Approximation with Guaranteed Stability Nonlinear Control Systems”. In: *8th IFAC Symposium on Nonlinear Control Systems (2010)*. EPFL-CONF-169739. 2010.

[42] Matthew E Taylor and Peter Stone. “Transfer learning for reinforcement learning domains: A survey”. In: *Journal of Machine Learning Research* 10.Jul (2009), pp. 1633–1685.

[43] Russ Tedrake et al. “LQR-trees: Feedback motion planning via sums-of-squares verification”. In: *The International Journal of Robotics Research* 29.8 (2010), pp. 1038–1052.

[44] Sebastian Thrun and Tom M Mitchell. “Lifelong robot learning”. In: *The biology and technology of intelligent autonomous agents*. Springer, 1995, pp. 165–196.

[45] Emanuel Todorov and Weiwei Li. “A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems”. In: *American Control Conference, 2005. Proceedings of the 2005*. IEEE. 2005, pp. 300–306.

[46] Sethu Vijayakumar, Aaron D’souza, and Stefan Schaal. “Incremental online learning in high dimensions”. In: *Neural computation* 17.12 (2005), pp. 2602–2634.

[47] Jian Wang et al. “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*. 2013.

[48] Yang Wang and Stephen Boyd. “Fast model predictive control using online optimization”. In: *IEEE Transactions on control systems technology* 18.2 (2010), p. 267.

[49] Shuyou Yu et al. “Tube MPC scheme based on robust control invariant set with application to Lipschitz nonlinear systems”. In: *Systems & Control Letters* 62.2 (2013), pp. 194–200.

[50] Tianhao Zhang et al. “Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search”. In: *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2016, pp. 528–535.