Technical Report

Aggressive Flight Performance using Robust Experience-driven Predictive Control Strategies: Experimentation and Analysis

Authors

Mosam Dabhi

Alexandar Spitzer

Nathan Michael

Carnegie Mellon University

5000 Forbes Avenue

Pittsburgh, PA, USA $15213 - 3890$

Abstract

Due to a plethora of applications of resource-constrained Micro Air Vehicles (MAV), there has been an increasing demand to fly aggressively high-speed flights to accomplish the tasks in the minimum amount of time for maintaining the resources. However, as the MAV executes these high-speed behaviors, the safety of the vehicle is often compromised in order to achieve the desired acceleration and aggressive behaviors. We enhance Robust Experience-driven Predictive Control (R-EPC) strategy with the ability to track aggressive a priori specified trajectories perfectly. This is achieved by appropriately handling higher order derivatives of the provided trajectories and appropriately formulating the Quadratic Program optimization problem to account for the higher order derivatives. Implications of leveraging full state dynamics of the MAV in a single control loop over separated cascaded control loop is also considered for situations where the feedforward terms are not readily available.

R-EPC generates a set of parameterized controllers for specific scenarios which can be stored and re-used to reduce the expensive online optimization costs onboard a computationally constrained platform, while the robust behavior tightens those constraint bounds to induce a conservative behavior in the presence of uncertain state estimates to further enforce safety. We further increase the efficiency of R-EPC to allow it to run on severely computationally constrained platforms effectively. This is done using a Markov-chain based control-law transition prediction strategy as well as an intelligent computation cache. Finally, model learning strategies are introduced which can adapt, learn, and compensate unmodeled exogenous disturbances acting on the vehicle online. This technical report also discusses the practical considerations and overall software organization of the predictive control architecture.

Contents

1 Introduction 8

1.1 Feedforward terms . . . 9

1.2 Full State Multirotor Dynamics . . . 9

1.3 Constrained Control with Robust Experience-driven Predictive Control . . . 10

1.4 Model Learning . . . 10

2 Predictive Control Strategies 12

2.1 Full State Multirotor Control Dynamics . . . 13

2.1.1 Multirotor dynamics and linearization . . . 13

2.2 Anticipating Trajectory Behavior using Feedforward terms . . . 18

2.2.1 Deriving feedforward terms . . . 19

2.3 EPC Control Formulation . . . 22

2.4 EPC Algorithm Overview . . . 25

2.5 Practical considerations . . . 27

2.5.1 Tuning cascaded EPC gains . . . 27

2.5.2 Tuning Full State EPC gains . . . 30

2.6 Results . . . 31

2.6.1 Full State EPC controller . . . 31

2.6.2 Feedback and Feedforward response . . . 33

2.7 Conclusion . . . 33

3 Robust Constraint Satisfaction for Aggressive Flights 36

3.1 Explicit EPC with Markov Chain Ordering . . . 37

3.1.1 Controller Database Generation . . . 38

3.1.2 Querying the MC ordered EPC database online . . . 39

3.2 Robust Predictive Control . . . 40

3.3 Results . . . 42

3.3.1 Constrained controller implications for aggressive flights . . . 43

3.3.2 Explicit EPC with Markov Chain Ordering . . . 46

3.3.3 Precomputing and Caching Expensive Matrices . . . 48
3.3.4 Robust Constrained Control . . . 51
3.3.5 Explicit Robust Constrained Control . . . 53

4 Model Learning 54
4.1 Online Model Adaptation . . . 55
4.1.1 Model Learning using LWPR . . . 55
4.1.2 Model Learning using ISSGPR . . . 55
4.1.3 Online model adaptation using Luenberger observer . . . 56
4.2 Practical considerations . . . 56
4.2.1 Bandwidth considerations . . . 56
4.3 Generating wind disturbance in simulation . . . 56
4.3.1 Online model adaptation in windy environment . . . 59

5 Software Organization and Code Interface 61
5.1 High-level Control Architecture . . . 61
5.2 System Interaction . . . 61
5.3 Class Description . . . 62

Bibliography 65

List of Figures

2.1 Reference frames for the multirotor dynamics. World frame is centered at the world origin, W, and body frame is centered at the multirotor center of mass. $\vec{r}_{B,W}$ is the position vector of body frame with respect to world frame expressed in world frame. 14

2.2 Figure 2.2a shows step response analysis confirming that a comparable feedback control response is achieved between full state EPC controller and PD controller strategy. 32

2.3 Figure 2.3a and Fig. 2.3b compares aggressive trajectory tracking performance between full State EPC and cascaded EPC controller without using feedforward terms and with using feedforward terms respectively. Figure 2.3a also shows that full State EPC controller performs better than cascaded EPC controller, which is clearly visible by improved tracking performance in Z direction. 34

2.4 Figure 2.4a and Fig. 2.4b shows overlaid aggressive trajectory execution using EPC controller with feedforward terms enabled and without feedforward terms enabled respectively. 34

2.5 This figure shows feedback response for step inputs given to the full state EPC controller till ≈ 40 seconds, followed by feedback and feedforward response of the controller for the 7^{th} order polynomial trajectory utilizing jerk and snap components. Above figure shows that feedback input is large for step inputs while the feedback component of control input is almost zero for the trajectories with feedforward terms. 35

3.1 Figure 3.1a shows that only the EPC controller with constraints enforced on the outer & inner loop is able to track the extremely aggressive trajectory. Consequently, the trajectory tracking error for this controller reduces in Z after subsequent experience generation and reuse. Figure 3.1b shows controller switching and experience reuse for cascaded EPC controller when the constraints are enforced on desired acceleration (outer loop) as well as roll/pitch angles (inner loop) – y axis represents controller indices and the value on y axis at a given time (x axis) shows the index of the controller that was used. 43

3.2 **(On real systems)** Figure 3.2a shows that unconstrained EPC is unable to track the aggressive trajectory whereas constrained EPC is able to track the trajectory along with storing and switching the corresponding controllers in the database (Fig. 3.2b). Figure 3.2c shows an overlaid image of the constrained controller safely executing aggressive trajectory, while Fig. 3.2d shows an overlaid image of MAV crashing while executing the same aggressive trajectory. This image emphasizes the harmful consequences of not enforcing constraints on the controller. 45

3.3 Figure 3.3a shows that the controller performance is substantially improved when a full state EPC controller is used for handling large step inputs when compared to a PD and a cascaded EPC controller, evident by reduced tracking error in Z direction. Controller switching (y axis represents controller indices) and control loop timing analysis for full state EPC controller is shown in upper and lower parts of Fig. 3.3b respectively 46

3.4 3.4a shows that QP and Markov Chain (MC) Ordered EPC perform well here, while regular EPC (Original EPC) controller performs considerably worse after learning most of the controllers. Figure 3.4b(i) shows query time for MC Ordered EPC is comparatively much less than regular EPC because of the transition matrices logic, but sometimes the timing for MC Ordered EPC is more than QP when the constraints are active. Figure 3.4b(ii) shows that regular EPC controller runs into timing issues as more and more controllers are computed, while MC ordered EPC leverages the previous experience and performs $\approx$ similar to QP controller. Videos executing above aggressive trial in simulation, with excerpts from $70 - 100$ seconds using QP control, regular EPC, and MC Ordered EPC are available here, here, and here respectively. 47

3.5 Improved trajectory tracking using MCO is visible in Fig. 3.5a when the aggressive trial is executed on NVIDIA TX2 (**RH2**). The controller switching in Fig. 3.5b shows that MCO-EPC uses the same controllers from the regular EPC database with a different query priority for choosing the best-transitioned controllers (**RH6**). 48

3.6 Figure 3.6a shows controller switching **(R5)** based on the ordering of controllers (successors) from MC simplification. Figure 3.6b shows velocity constraint satisfaction **(R2)**. 49

3.7 Cached matrices logic shows substantial improvement in total control loop timing resulting in improved trajectory tracking performance for a controller database of more than 100 controllers as shown in Fig. 3.7a, 3.7b. Figure 3.7c and 3.7d shows that the non-cached matrices based EPC calls QP control with a higher frequency compared to cached matrices based EPC and controller database growth comparison between the two techniques, respectively. 51

3.8 Cached matrices logic shows substantial improvement in total control loop timing resulting in improved trajectory tracking performance for a controller database of more than 100 controllers. 52

3.9 Time-varying state constraint bounds onboard a real system changes, that introduce robustness in enforcing the constraints as shown in Fig. 3.9a. Due to the enforced robustness in the presence of noisy odometry data, robust formulation improves the tracking performance in Z direction, as shown in Fig. 3.9b. Real-flight videos showing nominally constrained EPC and robustly constrained EPC are embedded here. 52

3.10 Figure 3.10a shows a distance-varying angular velocity constraint bound that tightens constraints when the MAV enters a specific region in space. Nominal EPC with no robust constraint tightening fails to track the trajectory in the presence of uncertain state estimates. Figure 3.10b shows the roll rate constraints are satisfied for the commanded aggressive trajectory except for 4 violations, resulting from a sudden large step input. 53

4.1 Wind field disturbance visualization in Matlab. 59

4.2 Tracking error in position for a circle trajectory flown in an environment with simulated wind-field disturbance. This figure shows that model learning disturbance compensation strategy outperforms reactive disturbance compensation strategy, as evident by the reduced tracking error. 60

5.1 High-level summary of the major components of the control architecture and their interactions. 62

5.2 High-level summary of the major components of the **`nonlinear_mpc (NMPC)`** package and their interactions shown as a UML activity diagram. For further detail on each of the classes, please refer the **`doxygen`** documentation. 63

1 Introduction

Due to the recent advances in technology, Micro Air Vehicles (MAVs) have become an essential part of critical scenarios such as search and rescue, inspection, and exploration missions. However, due to the limited battery life, a MAV is often tasked to finish the above tasks in a limited amount of time. Due to this constraint, there is a considerable demand for MAVs operating at high speeds while maintaining the safety of the vehicle robustly in such uncertain scenarios.

This report tackles the demand for executing aggressive and accurate trajectory tracking by leveraging feedforward terms in the control law, which improves the trajectory tracking by anticipating the trajectory behavior in a look-ahead fashion. The implications of using full state dynamics of the multirotor in a single control loop are then discussed over the traditional cascaded control loops for executing aggressive trajectories which require sudden angular acceleration changes. Although aggressive flight performance is achieved using simple reactive controllers coupled with higher order feedforward terms [1], constrained control strategies are often required to handle the infeasible state and control inputs demanded from the higher level trajectory planning subsystems. This report introduces constrained Model Predictive Control (MPC) with experience-driven strategies such as Experience-driven Predictive Control (EPC) [2]. While MPC emphasizes the importance of enforcing state and input constraints for executing aggressive trajectories, EPC emphasizes the importance of reusing the parameterized controllers generated beforehand by limiting the use of solving expensive quadratic programs online computationally constrained systems. Although EPC generates a varied set of parameterized controllers online, it is often computationally expensive to check each of the controllers in the database for a specific situation. Thus, Markov chain based ordering is then introduced [3] for efficient querying of the controller database. Experiments onboard a computationally constrained platform along with mean control loop timing is discussed to verify the real-time capabilities of these constrained control strategies. Constrained predictive control during poor state estimation is also discussed, which shows that the safety of the vehicle is further enforced conservatively by varying the bounds of constraints in the presence of uncertain state estimates. Lastly, this technical report discusses the implications of using different online model adaptation strategies coupled with the control strategies introduced in the preceding chapters to fly aggressively in an uncertain environment with considerable external disturbances while maintaining the safety of the vehicle.

1.1 Feedforward terms

While performing accurate and agile trajectory tracking, it is essential to calculate higher order state references and input information of upcoming trajectories using a differentially flat model [4] as evident in Fig. 2.4. Hence, when designing a control law, it is beneficial to anticipate the states from reference trajectories using the feedforward terms. Feedforward terms for the positional control can be calculated from the desired trajectory by extracting the higher order components such as the desired acceleration. Similarly, angular feedforward terms for obtaining desired angular velocity and desired angular acceleration can be calculated from the jerk and snap components of the higher order trajectory. These terms inform the control strategy for future trajectory behavior, and they can be used with any feedback control strategy, such as PD Control, Linear Quadratic Regulator Control, or Model Predictive Control. It is worth noting that feedforward terms are only calculated if some information about the desired trajectory is available; for example, feedforward terms will have no contribution during a step input as shown in Fig. 2.5.

1.2 Full State Multirotor Dynamics

Previous works have shown examples of complex, aggressive maneuvers [4, 5, 6] showing excellent tracking for a given trajectory. State-of-the-art approaches in control and trajectory generation exploit the time-scale separation and differential flatness of the systems [7]. These approaches leverage cascaded control schemes and separate the attitude and positional control dynamics via a geometric tracking controller [8]. Desired acceleration is calculated by the outer loop (position control) which generates the references for inner loop (attitude control), a setup which is directly affected by the over-imposed position and velocity control. Direct attitude control is therefore lost, and the demanded acceleration is dependent on several limitations in the change of rate, magnitude, and direction. Also, using desired acceleration to generate references for attitude control leads to complexities since it ignores the attitude dynamics and thus requires planned smooth trajectories, which should also account for actuator saturation. This requirement fails to hold for high-speed flight, where rapid changes in direction require large attitude swings that do not happen instantaneously. To tackle this problem, this technical report proposes to consider full state dynamics of the MAV, which explicitly model attitude's effect on positional control in a single control loop.

1.3 Constrained Control with Robust Experience-driven Predictive Control

The ability of a MAV to track highly aggressive and fast maneuvers depends on the availability of feedback control strategies which can ensure safety by enforcing constraints on desired acceleration, angular velocity, and other states and inputs. Following [2], a linear MPC formulation is leveraged to design an optimal and safe control strategy in this report. MPC problem can be solved by a constrained linear or quadratic program [9], provided the system does not deviate significantly from the linearization point. However, aggressive flights do not assume a linearized model (generally around hover), and hence MAVs must account for nonlinear dynamics to ensure that the optimization is performed with respect to an accurate system model. Although a nonlinear model predictive control (NMPC) formulation [10] is generally used for dealing with nonlinear dynamics of a multirotor system, the use of nonlinear dynamics in the optimization formulation increases the computational complexity which is detrimental for computationally constrained platforms such as MAVs. Hence, explicit MPC and linear MPC based approaches, such as Experience-driven Predictive Control (EPC) are leveraged in this work, which eliminates the need for online optimization by constructing a database of locally-optimal controllers derived via a receding horizon optimal control formulation [11, 12]. EPC stores a database of locally optimal controllers obtained by solving a Quadratic Program (QP) for specific scenarios with the intent of reusing them in future scenarios. Thus, an adaptive feedback control strategy is presented in this report that leverages a pre-computed database of controllers to ensure accurate trajectory tracking and constraint satisfaction.

The requirement of enforcing the constraints pose a threat of violation when passing through a region of poor state estimates due to the inaccurate odometry data. Thus, a robust formulation of the constrained predictive formulation with experience reuse is also considered, which uses covariance data from the pose estimates to adapt and tighten the constraints in the predictive formulation. This constraint tightening serves as a conservative approach for maintaining the safety of the vehicle, leading to reduced constraint violations.

1.4 Model Learning

Finally, model accuracy for predictive control is addressed, which is required for MPC based approaches to compensate for the unmodeled acceleration and torque disturbances acting on the multirotor system. Model learning formulations seek to estimate the error in the model dynamics and update the predictive model

accordingly [13, 14], by estimating the disturbances acting on the system. High-rate adaptive formulations combine MPC formulation with an online parameter estimator, e.g., a Luenberger observer or Kalman filter, to achieve more accurate, deliberative actions [15, 16, 17]. However, when the robotic system is subjected to complex, exogenous disturbances such as aerodynamic effects, treating all model uncertainty as parameters to estimate in the state-dependent disturbance observer can limit the overall model accuracy. Thus, model learning approaches are leveraged in this work, which augments a structured system model with a non-parametric, online-learned component, e.g., via a Gaussian process [18]. This semi-parametric model is then queried within the MPC formulation while continuing to adapt to the model changes. Since Gaussian process regression-based techniques are known to scale poorly with the amount of training data, kernel-based approaches such as Locally Weighted Projection Regression (LWPR) and Incremental Sparse Spectrum Gaussian Process Regression (ISSGPR) are leveraged in order to incorporate the data more efficiently via linear basis functions [13] and trigonometric approximations [14], respectively.

This technical report is structured in the following way. Importance of feedforward terms along with full state MAV dynamics with predictive control formulation is discussed in Chapter 2. Consequences of enforcing constraints for safe and agile performance with robust experience-driven predictive control formulation is detailed in Chapter 3. Model learning characteristics is discussed in Chapter 4. Finally, a detailed overview of the software organization used to validate the controller strategies proposed throughout this report are introduced in Chapter 5.

2 Predictive Control Strategies

While executing fast and aggressive maneuvers, the MAV consistently moves away from the assumed linearization point. Hence, more accurate trajectory tracking is obtained by leveraging the control strategies which explicitly deals with nonlinear system dynamics. Although applying predictive control techniques such as nonlinear model predictive control (NMPC) to nonlinear systems such as the MAV is beneficial, this strategy incurs a substantial computational penalty or may even be intractable on computationally constrained platforms [19]. Therefore, linear model predictive control (MPC) based strategy, experience-driven predictive control (EPC) [2] is used throughout this report, which employs infrequent online optimization, thus providing a middle ground between simple, reactive controllers and infinite horizon constrained optimal control requirements. This chapter introduces the EPC strategy by Desaraju [2] that leverages an online-updated database of past experiences to achieve high-rate, locally-optimal feedback control with constraint satisfaction. The versatility of this control strategy is discussed in this chapter by considering the full state dynamics of the multirotor in a single control loop instead of separated cascaded control loops [20] for the application of aggressive flight maneuvers.

Moreover, this chapter also introduces the feedforward terms, which anticipates the future state and control inputs into the control law for better trajectory tracking. Feedforward terms for position control loop are calculated by higher-order state references that are derived from the desired trajectory, while the angular feedforward terms for the attitude control loop are calculated using the jerk and snap references.

This chapter starts with describing the system model and linearization of the dynamics when considering full state system dynamics of a multirotor through a single control loop in Sec. 2.1. The derivation for angular feedforward terms is described in Sec. 2.2.1. The predictive control formulation by Desaraju [2] is restated in Sec. 2.4. Practical considerations when using the predictive control formulation using cascaded and full state dynamics is introduced in Sec. 2.5. Finally, the results in a simulation environment and real platforms are presented in Sec. 2.6 showing the real-time capability and improved trajectory tracking performance by the predictive control strategies that are discussed throughout this chapter.

2.1 Full State Multirotor Control Dynamics

Due to the versatile nature of the experience-driven predictive control formulation [2], it becomes simplified to model full state dynamics of the multirotor in a single control loop. Multirotor controllers usually employ a cascaded loop system where an outer loop controls the positional references, and an inner loop controls the attitude references. In this setup, the outer loop computes the desired attitude and thus generates references for the inner loop, which outputs motor RPM values as described in [20]. Such a setup simplifies the system that is not straightforward to control using a single PID control, into two systems that are easily controlled using the cascaded control loop.

Thus, simple control techniques such as PID can be used for adequate control performance. However, since the positional control loop assumes that the desired attitude is realized at all times, this assumption might fail for high-speed flights, where rapid changes in direction necessitate large attitude swings that do not happen instantaneously. Furthermore, since MAVs are assumed to operate near hover due to a linearization at level attitude, this assumption might be harmful while executing high-speed flights. Thus, a control strategy is proposed in this section that is not biased towards the level attitude state by considering the full dynamics of the multirotor in a single loop. This consideration explicitly models the inner loop's effect on the outer loop. The dynamics now consists of 12 degrees of freedom due to consideration of the position and attitude information into a single state vector.

2.1.1 Multirotor dynamics and linearization

The state of the multirotor system for the full state controller dynamics, $\mathbf{x} \in \mathbb{R}^{12}$ is given by the position, velocity of the center of mass, orientation (locally parameterized by Euler angles) and the angular velocity as

$$\mathbf{x} = \begin{bmatrix} p_x, p_y, p_z, v_x, v_y, v_z, \phi, \theta, \psi, \omega_x, \omega_y, \omega_z \end{bmatrix}^\top \tag{2.1}$$

while control input to the system is given by

$$\mathbf{u} = \begin{bmatrix} F, \tau_x, \tau_y, \tau_z \end{bmatrix}^\top \tag{2.2}$$

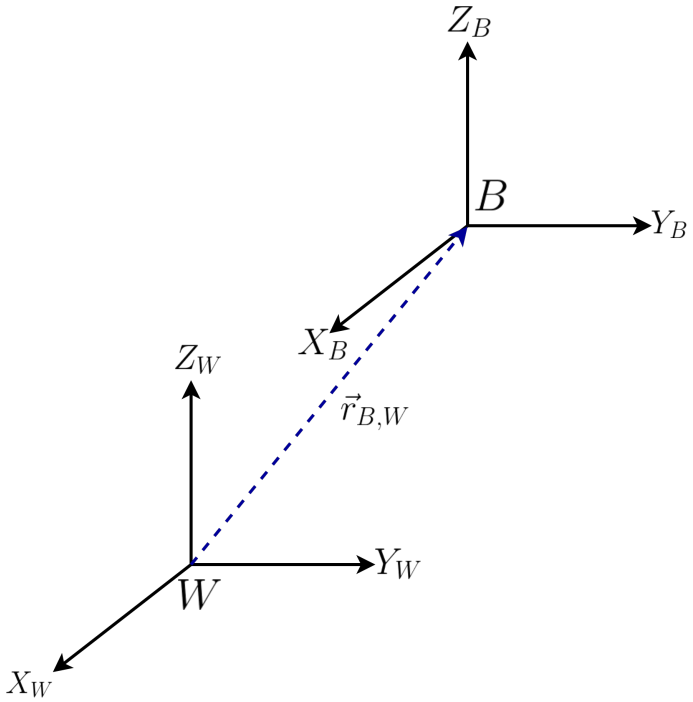


Figure 2.1: Reference frames for the multirotor dynamics. World frame is centered at the world origin, W, and body frame is centered at the multirotor center of mass. $\vec{r}_{B,W}$ is the position vector of body frame with respect to world frame expressed in world frame.

where F is the force from all the propellers and τ_x, τ_y, τ_z are the moments about the body frame axes. $Z-Y-X$ Euler angle notation is used to model the rotation of the multirotor in the world frame. To get from W to B, a rotation takes place about Z_W by the yaw angle, ψ, followed by rotation about the intermediate y-axis by the pitch angle, θ, and finally a rotation about the x-axis by the roll angle, ϕ for the reference frame shown in Fig. 2.1. The rotation matrix for transforming coordinates from B to W in $SO(3)$ is given by

$$
{}^{W}R_B = \begin{bmatrix} \cos\psi\cos\theta & \cos\psi\sin\phi\sin\theta - \cos\phi\sin\psi & \cos\phi\cos\psi\sin\theta + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\psi\sin\theta & \cos\phi\sin\psi\sin\theta - \cos\psi\sin\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix} \tag{2.3}
$$

The forces on the multirotor system are gravity, in the $-Z_W$ direction, and the forces from each of the rotors F, are in the Z_B direction. The equation governing the acceleration of the center of mass is

$$m\ddot{\mathbf{r}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + {}^{W}R_{B} \begin{bmatrix} 0 \\ 0 \\ F \end{bmatrix} \tag{2.4}$$

The time derivatives of position are calculated as

$$\begin{aligned} \dot{p}_x &= v_x \\ \dot{p}_y &= v_y \\ \dot{p}_z &= v_z \end{aligned} \tag{2.5}$$

Substituting (2.3) into (2.4), the time derivatives of velocity are computed as

$$\begin{aligned} \dot{v}_x &= \frac{F}{m}(\cos\phi\cos\psi\sin\theta + \sin\phi\sin\psi) \\ \dot{v}_y &= \frac{F}{m}(-\cos\psi\sin\phi + \cos\phi\sin\theta\sin\psi) \\ \dot{v}_z &= \frac{F}{m}(\cos\theta\cos\phi - g) \end{aligned} \tag{2.6}$$

The angular acceleration determined by the Euler equations is

$$I \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.7}$$

where $\omega = \begin{bmatrix} p \\ q \\ r \end{bmatrix}$ is body frame angular velocity in x, y, z axis respectively and I is the moment of inertia matrix referenced to the center of mass along the $X_B - Y_B - Z_B$ axes. (2.7) can be rewritten as

$$\dot{\omega} = I^{-1}(\tau - \omega \times I\omega) \tag{2.8}$$

The derivatives of orientation states are calculated as

$$\begin{aligned}\dot{\phi} &= \omega_x + \omega_y \sin\phi \tan\theta + \omega_z \cos\phi \tan\theta \\ \dot{\theta} &= \omega_y \cos\phi - \omega_z \sin\phi \\ \dot{\psi} &= \omega_y \frac{\sin\phi}{\cos\theta} + \omega_z \frac{\cos\phi}{\cos\theta}\end{aligned} \tag{2.9}$$

Partial derivatives of the time derivatives of all the state variables with respect to all state and input variables (Eq. (2.5) $-$ (2.9)) are calculated for linearization. The derivatives with respect to state A (having dimension of 12×12) are calculated in (2.10) as

$$\frac{\partial \dot{p}_x}{\partial v_x} = \frac{\partial \dot{p}_y}{\partial v_y} = \frac{\partial \dot{p}_z}{\partial v_z} = 1$$

$$\begin{aligned}\frac{\partial \dot{v}_x}{\partial \phi} &= \frac{F}{m}(\cos\phi \sin\psi - \cos\psi \sin\phi \sin\theta) \\ \frac{\partial \dot{v}_x}{\partial \theta} &= \frac{F}{m}(\cos\phi \cos\psi \cos\theta) \\ \frac{\partial \dot{v}_x}{\partial \psi} &= \frac{F}{m}(\cos\psi \sin\phi - \cos\phi \sin\psi \sin\theta)\end{aligned}$$

$$\begin{aligned}\frac{\partial \dot{v}_y}{\partial \phi} &= \frac{F}{m}(-\cos\phi \cos\psi - \sin\phi \sin\psi \sin\theta) \\ \frac{\partial \dot{v}_y}{\partial \theta} &= \frac{F}{m}(\cos\phi \cos\theta \sin\psi) \\ \frac{\partial \dot{v}_y}{\partial \psi} &= \frac{F}{m}(\sin\phi \sin\psi + \cos\phi \cos\psi \sin\theta) \\ \frac{\partial \dot{v}_z}{\partial \phi} &= \frac{F}{m}(-\cos\theta \sin\phi)\ ; \quad \frac{\partial \dot{v}_z}{\partial \theta} = \frac{F}{m}(-\cos\phi \sin\theta)\ ; \quad \frac{\partial \dot{v}_z}{\partial \psi} = 0\end{aligned}$$

$$\frac{\partial \dot{\phi}}{\partial \phi} = \omega_y \cos\phi \tan\theta - \omega_z \sin\phi \tan\theta \; ; \quad \frac{\partial \dot{\phi}}{\partial \theta} = \omega_z \cos\phi \sec^2\theta + \omega_y \sin\phi \sec^2\theta \; ; \quad \frac{\partial \dot{\phi}}{\partial \psi} = 0$$
$$\frac{\partial \dot{\theta}}{\partial \phi} = -\omega_z \cos\phi - \omega_y \sin\phi \; ; \quad \frac{\partial \dot{\theta}}{\partial \theta} = 0 \; ; \quad \frac{\partial \dot{\theta}}{\partial \psi} = 0$$
$$\frac{\partial \dot{\psi}}{\partial \phi} = \omega_y \frac{\cos\phi}{\cos\theta} - \omega_z \frac{\sin\phi}{\cos\theta} \; ; \quad \frac{\partial \dot{\psi}}{\partial \theta} = (\omega_z \cos\phi + \omega_y \sin\phi) \sin\theta \sec^2\theta \; ; \quad \frac{\partial \dot{\psi}}{\partial \psi} = 0$$

$$\frac{\partial \dot{\phi}}{\partial \omega_x} = 1 \; ; \quad \frac{\partial \dot{\phi}}{\partial \omega_y} = \sin\phi \tan\theta \; ; \quad \frac{\partial \dot{\phi}}{\partial \omega_z} = \cos\phi \tan\theta$$
$$\frac{\partial \dot{\theta}}{\partial \omega_x} = 0 \; ; \quad \frac{\partial \dot{\theta}}{\partial \omega_y} = \cos\phi \; ; \quad \frac{\partial \dot{\theta}}{\partial \omega_z} = -\sin\phi$$
$$\frac{\partial \dot{\psi}}{\partial \omega_x} = 0 \; ; \quad \frac{\partial \dot{\psi}}{\partial \omega_y} = \frac{\sin\phi}{\cos\theta} \; ; \quad \frac{\partial \dot{\psi}}{\partial \omega_z} = \frac{\cos\phi}{\cos\theta}$$

$$\frac{\partial \dot{\omega}}{\partial \omega_x} = -I^{-1}(e_1 \times I\omega + \omega \times I_1) \; ; \quad \frac{\partial \dot{\omega}}{\partial \omega_y} = -I^{-1}(e_2 \times I\omega + \omega \times I_2) \; ; \quad \frac{\partial \dot{\omega}}{\partial \omega_x} = -I^{-1}(e_3 \times I\omega + \omega \times I_3) \tag{2.10}$$

where e_i is the i^{th} unit vector and I_i is the i^{th} column of matrix I. Similarly, derivatives with respect to input B (having dimension of 12×4) are calculated in (2.11) as

$$\begin{aligned}
\frac{\partial \dot{v}_x}{\partial F} &= \frac{1}{m}(\cos\phi \cos\psi \sin\theta + \sin\phi \sin\psi) \\
\frac{\partial \dot{v}_y}{\partial F} &= \frac{1}{m}(-\cos\psi \sin\phi + \cos\phi \sin\theta \sin\psi) \\
\frac{\partial \dot{v}_z}{\partial F} &= \frac{1}{m}\cos\theta \cos\phi \\
\frac{\partial \dot{\omega}}{\partial \tau} &= I^{-1}
\end{aligned} \tag{2.11}$$

Linearization of the full state multirotor dynamics described in (2.10) (2.11) can be used with the predictive control formulation to realize full-state dynamics of the multirotor in a single control loop.

2.2 Anticipating Trajectory Behavior using Feedforward terms

Feedforward terms comprise of desired acceleration for the positional control loop and desired angular velocity, desired angular acceleration for the attitude control. These state-dependent feedforward terms can be realized for reactive as well as predictive control formulations such as PD control and EPC, respectively. It is worth noting that the EPC control law, being an optimal control strategy comprising of state and input variables in the cost function is also able to anticipate the input-dependent references, apart from state-dependent references. This anticipative nature allows for perfect tracking of given higher order trajectories under the idealized point mass model.

Using a predictive control strategy requires specifying a reference to be followed for the duration of the full prediction horizon. In the case of a position controller, the position reference (and its higher order derivatives) comes from a pre-computed desired trajectory that specifies the desired state of the vehicle in space over time. For an attitude controller, the reference needs to specify the desired orientation, angular velocity, and angular acceleration of the vehicle. These can be computed from the position reference using the differential flatness transformation [4]. The position references can be used over the attitude control horizon to compute orientation references. This requirement is fulfilled for the predictive attitude controller if the position controller is also a predictive controller, thus providing acceleration inputs for a horizon which is at least as long as the attitude control horizon. The position control output can be used to build reasonable attitude controller references for the full attitude control horizon. Since the cascaded position control runs at a lower rate than the attitude controller, not all of the resulting acceleration output can be used to build the attitude reference. If the attitude controller runs at 200 Hz and the position controller runs at 100 Hz, and if both the control strategies are using the similar number of time steps in the prediction horizon, then only half of the acceleration output from the outer loop can be used to calculate the inner loop reference. The outer loop output references are doubled for compensating this limitation for the presented formulation. The effects of this limitation can be diminished by leveraging a full state EPC controller which runs at a unique update rate for a single control loop, thus generating required references for the full horizon.

Section 2.6.2 discusses results that shows the importance of enforcing the feedforward terms while executing aggressive trials through simulation as well as hardware studies. Fig. 2.4b shows dire consequences of not utilizing the feedforward terms for aggressive trajectories, that culminates in a crash, resulting in physical

damage to the multirotor system.

2.2.1 Deriving feedforward terms

Following [4], the derivation for the desired angular velocity and desired angular acceleration using the jerk and snap components of the trajectories is presented in this section.

Following terminologies are used for this section. World frame is represented by $\mathcal{W}$, and the body frame of the multirotor is represented by $\mathcal{B}$. The angular velocity of the multirotor is represented by ω_{BW}, denoting angular velocity of the body frame in the world frame. The body frame angular velocity components p, q, and r along $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ axis respectively can then be written as

$$\omega_{\mathcal{BW}} = p\mathbf{x_B} + q\mathbf{y_B} + r\mathbf{z_B} \tag{2.12}$$

The reference position for the multirotor in world frame is denoted by $\mathbf{r}$. Referring the application of Newton's equation of motion, $F = ma$, where m is the mass of the vehicle and a is the reference acceleration, and considering that the forces acting on the vehicle are gravity, in the $-\mathbf{z_W}$ direction and the sum of forces (thrust) from each rotors, u_1, in the $\mathbf{z_B}$, the following vector differential equation can be written

$$m\ddot{\mathbf{r}} = -mg\mathbf{z_W} + u_1\mathbf{z_B} \tag{2.13}$$

$\mathbf{z_B}$ is calculated from the reference acceleration. Desired angular velocity is calculated by taking the first derivative of (2.13). To get an expression containing desired angular acceleration, two derivatives of (2.13) are calculated. The third derivative of position, $\mathbf{r}$, is jerk and the fourth derivative, snap. Thus, the trajectory jerk vector and snap vector are used to calculate the desired angular velocity and desired angular acceleration vectors, respectively. First derivative of (2.13) is found by applying the product rule to (2.13), thus generating the expression for desired angular velocity. Since the world frame z-axis is constant, its derivative comes out to be zero.

$$m\dot{\mathbf{a}} = \dot{u_1}\mathbf{z_B} + u_1\dot{\mathbf{z}}_\mathbf{B} \tag{2.14}$$

For calculating the derivative of $\mathbf{z_B}$, the velocity of a rigid body rotating with $\omega_{\mathcal{BW}}$ is always perpendicular to

the axis of rotation and to the direction vector from the origin to the rigid body. The magnitude of the linear velocity of the rigid body thus depends on the perpendicular distance from the body to the axis of rotation (leading to $v = \omega r$). This distance to the axis of rotation scales with $\sin(\theta)$ where θ is the angle between the axis of rotation and the rigid body, leading to a cross product. Thus, the derivative of $\mathbf{z_B}$ is $\omega_{\mathcal{BW}} \times \mathbf{z_B}$. The desired force vector for the controller is denoted as $\mathbf{F}_{des}$. Following [4], this report makes an assumption to ignore the feedback terms in $\mathbf{F}_{des}$. Using the product rule on $u_1 = \langle \mathbf{F}_{des}, \mathbf{z_B} \rangle$

$$\begin{aligned} \dot{u_1} &= \langle \dot{\mathbf{F}}_{des}, \mathbf{z_B} \rangle + \langle \mathbf{F}_{des}, \dot{\mathbf{z}}_\mathbf{B} \rangle \\ \dot{u_1} &= \langle \dot{\mathbf{F}}_{des}, \mathbf{z_B} \rangle + \langle \mathbf{F}_{des}, \omega_{\mathcal{BW}} \times \mathbf{z_B} \rangle \end{aligned} \tag{2.15}$$

Since feedback terms in $\mathbf{F}_{des}$ are ignored, the second term in (2.15) becomes zero. Thus, $\dot{u_1}$ is

$$\begin{aligned} \dot{u_1} &= \langle \dot{\mathbf{F}}_{des}, \mathbf{z_B} \rangle \\ \dot{u_1} &= \langle m\dot{\mathbf{a}}, \mathbf{z_B} \rangle = \mathbf{z_B} \cdot m\dot{\mathbf{a}} \end{aligned} \tag{2.16}$$

Projecting (2.14) along the body z-axis and using the fact that $\dot{\mathbf{z}}_\mathbf{B} = \omega_{\mathcal{BW}} \times \mathbf{z_B}$, is substituted $\dot{u_1}$ from (2.16) into (2.14) to define the vector $\mathbf{h}_\omega$ as

$$\mathbf{h}_\omega = \omega_{BW} \times \mathbf{z_B} = \frac{m}{u_1}\big(\dot{\mathbf{a}} - (\mathbf{z_B} \cdot \dot{\mathbf{a}})\mathbf{z_B}\big) \tag{2.17}$$

(2.17) enables the calculation of desired roll and pitch angular velocity by projecting $\frac{m}{u_1}\dot{\mathbf{a}}$ onto the $x_B - y_B$ plane. Following the right-hand rule, a positive roll corresponds to a negative motion in the body frame $\mathbf{y}-$axis and a positive pitch corresponds to a positive motion along the body frame $\mathbf{x}-$axis. Desired roll and pitch angular velocity can then be written, from (2.12) as

$$p = -\mathbf{h}_\omega \cdot \mathbf{y_B}, \quad q = \mathbf{h}_\omega \cdot \mathbf{x_B} \tag{2.18}$$

The desired angular velocity in yaw, r is calculated by utilizing the first order desired yaw component, $\dot{\psi}$ from the trajectory as

$$r = \dot{\psi} \cdot \mathbf{z_B} \tag{2.19}$$

For calculating the desired angular acceleration components, a second derivative of (2.14) is calculated by

applying the product rule again

$$\begin{aligned} m\ddot{\mathbf{a}} &= \ddot{u}_1\mathbf{z_B} + \dot{u}_1\dot{\mathbf{z}}_\mathbf{B} + \dot{u}_1\dot{\mathbf{z}}_\mathbf{B} + u_1\ddot{\mathbf{z}}_\mathbf{B} \\ m\ddot{\mathbf{a}} &= \ddot{u}_1\mathbf{z_B} + 2\dot{u}_1\dot{\mathbf{z}}_\mathbf{B} + u_1\ddot{\mathbf{z}}_\mathbf{B} \end{aligned} \tag{2.20}$$

Double derivative $\ddot{\mathbf{z}}_\mathbf{B}$ is calculated by taking a derivative of $\dot{\mathbf{z}}_\mathbf{B} = \omega_{\mathcal{BW}} \times \mathbf{z_B}$. Applying the product rule on this equation results in final double derivative solution: $\ddot{\mathbf{z}}_\mathbf{B} = \dot{\omega}_{BW} \times \mathbf{z_B} + \omega_{\mathcal{BW}} \times (\omega_{BW} \times \mathbf{z_B})$. $\ddot{u}_1$ is calculated in a similar manner by taking another derivative of (2.16)

$$\ddot{u}_1 = m\big(\langle \ddot{\mathbf{a}}, \mathbf{z_B}\rangle + \langle \dot{\mathbf{a}}, \dot{\mathbf{z}}_\mathbf{B}\rangle\big) \tag{2.21}$$

Substituting $\dot{u}_1, \ddot{u}_1, \dot{\mathbf{z}}_\mathbf{B}, \ddot{\mathbf{z}}_\mathbf{B}$ into (2.20) and solving for $\dot{\omega}_{\mathcal{BW}} \times \mathbf{z_B}$

$$\begin{aligned} m\ddot{\mathbf{a}} &= \ddot{u}_1\mathbf{z_B} + 2\dot{\mathbf{u}}_1\dot{\mathbf{z}}_\mathbf{B} + u_1\ddot{\mathbf{z}}_\mathbf{B} \\ m\ddot{\mathbf{a}} &= m\big(\langle \ddot{\mathbf{a}}, \mathbf{z_B}\rangle + \langle \dot{\mathbf{a}}, \dot{\mathbf{z}}_\mathbf{B}\rangle\big)\mathbf{z_B} + 2\langle m\dot{\mathbf{a}}, \mathbf{z_B}\rangle\big(\omega_{\mathcal{BW}} \times \mathbf{z_B}\big) + u_1\big(\dot{\omega}_{\mathcal{BW}} \times \mathbf{z_B} + \omega_{\mathcal{BW}} \times (\omega_{\mathcal{BW}} \times \mathbf{z_B})\big) \\ \frac{u_1}{m}\big(\dot{\omega}_{BW} \times \mathbf{z_B} + \omega_{BW} \times (\omega_{BW} \times \mathbf{z_B})\big) &= \ddot{\mathbf{a}} - \big(\langle \ddot{\mathbf{a}}, \mathbf{z_B}\rangle + \langle \dot{\mathbf{a}}, \dot{\mathbf{z}}_\mathbf{B}\rangle\big)\mathbf{z_B} - 2\langle \dot{\mathbf{a}}, \mathbf{z_B}\rangle\big(\omega_{BW} \times \mathbf{z_B}\big) \\ \dot{\omega}_{BW} \times \mathbf{z_B} &= \frac{m}{u_1}\big(\ddot{\mathbf{a}} - (\langle \ddot{\mathbf{a}}, \mathbf{z_B}\rangle + \langle \dot{\mathbf{a}}, \dot{\mathbf{z}}_\mathbf{B}\rangle)\mathbf{z_B} - 2\langle \dot{\mathbf{a}}, \mathbf{z}_\mathbb{B}\rangle(\omega_{BW} \times \mathbf{z_B})\big) - \omega_{BW} \times \big(\omega_{BW} \times \mathbf{z_B}\big) \end{aligned} \tag{2.22}$$

For computing the desired angular acceleration coefficients, an approach similar to the desired angular velocity calculation is used

$$\begin{aligned} \dot{p} &= -\langle \dot{\omega}_{BW} \times \mathbf{z_B}, \mathbf{y_B}\rangle \\ \dot{q} &= \langle \dot{\omega}_{BW} \times \mathbf{z_B}, \mathbf{x_B}\rangle \end{aligned} \tag{2.23}$$

Finally, the desired angular acceleration in yaw, $\dot{r}$ can be calculated by utilizing the second order desired yaw component, $\ddot{\psi}$ from the trajectory as

$$\dot{r} = \ddot{\psi} \cdot \mathbf{z_B} \tag{2.24}$$

2.3 EPC Control Formulation

For a model predictive control strategy, the affine linear system model with constraints is given by

$$\begin{aligned} \bar{\mathbf{x}}_{k+1} &= \bar{\mathbf{x}}_{k+1}^{\text{nom}} + \hat{\mathbf{p}} \\ &= \mathbf{A}\bar{\mathbf{x}}_k + \mathbf{B}\bar{\mathbf{u}}_k + \mathbf{c} + \hat{\mathbf{p}} \\ &= \mathbf{A}\bar{\mathbf{x}}_k + \mathbf{B}\bar{\mathbf{u}}_k + \tilde{\mathbf{c}} \end{aligned} \tag{2.25}$$

with linear state and input constraints

$$\begin{aligned} \mathbf{G}_{\mathbf{x}}\mathbf{x}_{k+1} &\leq \mathbf{g}_{\mathbf{x}} \\ \mathbf{G}_{\mathbf{u}}\mathbf{u}_k &\leq \mathbf{g}_{\mathbf{u}} \end{aligned} \tag{2.26}$$

where the corresponding prediction output from model learners is given by the vector, $\hat{\mathbf{p}} = \begin{bmatrix} p_0, p_1, \dots \end{bmatrix}^\top$ (detailed in Chap. 4). Here, $\mathbf{x}$ denotes the state vector, and $\mathbf{u}$ denotes the input vector. Affine model (2.25) automatically adapts to capture the effects of nonlinearities and unmodeled dynamics.

Following [2], the receding-horizon control problem as a quadratic program can be formulated as

$$\begin{aligned} \underset{\bar{\mathbf{u}}_k}{\text{argmin}} \sum_{k=0}^{N-1} \frac{1}{2}(\bar{\mathbf{x}}_{k+1} - \bar{\mathbf{r}}_{k+1})^\top \mathbf{Q}_{k+1}(\bar{\mathbf{x}}_{k+1} - \bar{\mathbf{r}}_{k+1}) + \frac{1}{2}(\bar{\mathbf{u}}_k - \bar{\mathbf{u}}_k^{\text{ref}})^\top \mathbf{R}(\bar{\mathbf{u}}_k - \bar{\mathbf{u}}_k^{\text{ref}}) \\ \text{s.t.} \quad \bar{\mathbf{x}}_{k+1} = \mathbf{A}\bar{\mathbf{x}}_k + \mathbf{B}\bar{\mathbf{u}}_k + \tilde{\mathbf{c}} \\ \mathbf{G}_{\mathbf{x}_{k+1}}\bar{\mathbf{x}}_{\mathbf{k+1}} \leq \mathbf{g}_{\mathbf{x}_{k+1}} \\ \mathbf{G}_{\mathbf{u}_k}\bar{\mathbf{u}}_{\mathbf{k}} \leq \mathbf{g}_{\mathbf{u}_k} \\ \forall k = 0, \dots, N-1 \end{aligned} \tag{2.27}$$

where $\bar{\mathbf{u}}_k^{\text{ref}}$ consists of prediction output with feedforward control input references, $\mathbf{x}$ denotes the state vector, $\mathbf{r}$ denotes the reference or desired state, $\mathbf{u}$ denotes the input vector, and Q and R define the LQR-style cost function. Taking the current state as the nominal state, $\mathbf{x}^\star = \mathbf{x}_0$ with N reference states $\mathbf{r_1}, \dots, \mathbf{r_N}$, let $\bar{\mathbf{r}} = \mathbf{r} - \mathbf{x}^\star$.

To simplify the notation, $\boldsymbol{x} = \left[\bar{\mathbf{x}}_1^\top, ..., \bar{\mathbf{x}}_N^\top\right]^\top$, $\boldsymbol{r} = \left[\bar{\mathbf{r}}_1^\top, ..., \bar{\mathbf{r}}_N^\top\right]^\top$, $\boldsymbol{u} = \left[\bar{\mathbf{u}}_0^\top, ..., \bar{\mathbf{u}}_{N-1}^\top\right]^\top$, $\boldsymbol{u}^{\text{ref}} = \left[\bar{\mathbf{u}}_0^{\text{ref}^\top}, ..., \bar{\mathbf{u}}_{N-1}^{\text{ref}^\top}\right]^\top$,

$$\boldsymbol{\mathcal{B}} = \begin{bmatrix} \mathbf{B} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{AB} & \mathbf{B} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \\ \mathbf{A}^{N-1}\mathbf{B} & \mathbf{A}^{N-2}\mathbf{B} & \cdots & \mathbf{B} \end{bmatrix}, \quad \boldsymbol{c} = \begin{bmatrix} \tilde{\mathbf{c}} \\ (\mathbf{A}+\mathbf{I})\tilde{\mathbf{c}} \\ \vdots \\ \sum_{i=0}^{N-1} \mathbf{A}^i\tilde{\mathbf{c}} \end{bmatrix} \tag{2.28}$$

$\boldsymbol{\mathcal{Q}} = \text{diag}(\mathbf{Q}_1, \cdots, \mathbf{Q}_N)$, $\boldsymbol{\mathcal{R}} = \text{diag}(\mathbf{R}_0, \cdots, \mathbf{R}_{N-1})$, $\boldsymbol{\mathcal{G}}_{\mathbf{x}} = \text{diag}(\mathbf{G}_{\mathbf{x}_1}, \cdots, \mathbf{G}_{\mathbf{x}_N})$, $\boldsymbol{\mathcal{G}}_{\mathbf{u}} = \text{diag}(\mathbf{G}_{\mathbf{u}_0}, \cdots, \mathbf{G}_{\mathbf{u}_{N-1}})$, $\boldsymbol{g}_{\mathbf{x}} = \left[\mathbf{g}_{\mathbf{x}_1}^\top, \cdots, \mathbf{g}_{\mathbf{x}_N}^\top\right]^\top$, and $\boldsymbol{g}_{\mathbf{u}} = \left[\mathbf{g}_{\mathbf{u}_0}^\top, \cdots, \mathbf{g}_{\mathbf{u}_{N-1}}^\top\right]^\top$. Provided $\bar{\mathbf{x}}_0 = 0$, we can rewrite (2.27) as

$$\begin{aligned} \underset{\boldsymbol{u}}{\text{argmin}}\ & \frac{1}{2}(\boldsymbol{x}-\boldsymbol{r})^\top \boldsymbol{\mathcal{Q}}(\boldsymbol{x}-\boldsymbol{r}) + \frac{1}{2}(\boldsymbol{u}-\boldsymbol{u}^{\text{ref}})^\top \boldsymbol{\mathcal{R}}(\boldsymbol{u}-\boldsymbol{u}^{\text{ref}}) \\ \text{s.t.}\quad & \boldsymbol{x} = \boldsymbol{\mathcal{B}}\boldsymbol{u} + \boldsymbol{c} \\ & \boldsymbol{\mathcal{G}}_{\mathbf{x}}\boldsymbol{x} \leq \boldsymbol{g}_{\mathbf{x}} \\ & \boldsymbol{\mathcal{G}}_{\mathbf{u}}\boldsymbol{u} \leq \boldsymbol{g}_{\mathbf{u}} \end{aligned} \tag{2.29}$$

We can reconstruct an equivalent QP entirely in terms of $\boldsymbol{u}$ by substituting the dynamics constraints and dropping constant terms in the cost function

$$\begin{aligned} \underset{\boldsymbol{u}}{\text{argmin}}\ & \frac{1}{2}\boldsymbol{u}^\top \boldsymbol{\mathcal{H}}\boldsymbol{u} + \boldsymbol{h}^\top \boldsymbol{u} \\ \text{s.t.}\quad & \boldsymbol{\Gamma}\boldsymbol{u} \leq \boldsymbol{\gamma} \end{aligned} \tag{2.30}$$

where $\boldsymbol{\mathcal{H}} = \boldsymbol{\mathcal{B}}^\top \boldsymbol{\mathcal{Q}}\boldsymbol{\mathcal{B}} + \boldsymbol{\mathcal{R}}$, $\boldsymbol{h} = \boldsymbol{\mathcal{B}}^\top \boldsymbol{\mathcal{Q}}(\boldsymbol{c}-\boldsymbol{r}) - \boldsymbol{\mathcal{R}}\boldsymbol{u}^{\text{ref}}$

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

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

$$\begin{aligned} \boldsymbol{\mathcal{H}}\boldsymbol{u} + \boldsymbol{h} + \boldsymbol{\Gamma}^\top \boldsymbol{\lambda} = 0 \\ \boldsymbol{\Lambda}(\boldsymbol{\Gamma}\boldsymbol{u} - \boldsymbol{\gamma}) = 0 \end{aligned} \tag{2.31}$$

Considering only the active constraints (i.e., with $\boldsymbol{\lambda} > \mathbf{0}$) for a given solution, we can reconstruct $\boldsymbol{u}$ and $\boldsymbol{\lambda}$ by solving a linear system derived from (2.31), where the subscript a indicates rows corresponding to the active constraints

$$\begin{bmatrix} \boldsymbol{\mathcal{H}} & \boldsymbol{\Gamma}_a^\top \\ \boldsymbol{\Gamma}_a & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{\lambda}_a \end{bmatrix} = \begin{bmatrix} -\boldsymbol{h} \\ \boldsymbol{\gamma}_a \end{bmatrix} \tag{2.32}$$

The resulting QP control law, $\boldsymbol{u}$, is affine in the predicted state error, $\boldsymbol{r}$, and parameterized by the system dynamics as

$$\boldsymbol{u} = \boldsymbol{\mathcal{E}}_5 \boldsymbol{r} - \left(\boldsymbol{\mathcal{E}}_5 \boldsymbol{c} - \boldsymbol{\mathcal{E}}_4 \boldsymbol{\mathcal{R}} \boldsymbol{u}^{\text{ref}} + \boldsymbol{\mathcal{E}}_3 \begin{bmatrix} \boldsymbol{g}_{\mathbf{x}}^+ - \boldsymbol{\mathcal{G}}_{\mathbf{x}} \boldsymbol{c} \\ -\boldsymbol{g}_{\mathbf{x}}^- + \boldsymbol{\mathcal{G}}_{\mathbf{x}} \boldsymbol{c} \\ \boldsymbol{g}_{\mathbf{u}}^+ \\ -\boldsymbol{g}_{\mathbf{u}}^- \end{bmatrix}_a \right) \tag{2.33}$$

where $\boldsymbol{\mathcal{E}}_1 = \boldsymbol{\Gamma}_a \boldsymbol{\mathcal{H}}^{-1}$, $\boldsymbol{\mathcal{E}}_2 = -(\boldsymbol{\mathcal{E}}_1 \boldsymbol{\Gamma}_a^\top)^{-1}$, $\boldsymbol{\mathcal{E}}_3 = \boldsymbol{\mathcal{E}}_1^\top \boldsymbol{\mathcal{E}}_2$, $\boldsymbol{\mathcal{E}}_4 = \boldsymbol{\mathcal{H}}^{-1} + \boldsymbol{\mathcal{E}}_3 \boldsymbol{\mathcal{E}}_1$, and $\boldsymbol{\mathcal{E}}_5 = \boldsymbol{\mathcal{E}}_4 \boldsymbol{\mathcal{B}}^\top \boldsymbol{\mathcal{Q}}$. Also, since the coefficients in (2.33) are all functions of $\mathbf{A}, \mathbf{B}$, and $\tilde{\mathbf{c}}$, the overall control law $\kappa(\mathbf{x}_0, \mathbf{r}_1, \cdots, \mathbf{r}_N)$ can be written in terms of a parameterized feedback gain matrix $\mathbf{K}$ and feedforward vector $\mathbf{k}_{\text{ff}}$ and hence can be calculated on each control loop iteration.

$$\kappa(\mathbf{x}_0, \mathbf{r}_1, \cdots, \mathbf{r}_N) = \mathbf{K}(\mathbf{A}, \mathbf{B}, \tilde{\mathbf{c}})\boldsymbol{r} + \mathbf{k}_{\text{ff}}(\mathbf{A}, \mathbf{B}, \tilde{\mathbf{c}}) \tag{2.34}$$

This parameterization then 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

$$\boldsymbol{\lambda}_a = -\boldsymbol{\mathcal{E}}_6 \boldsymbol{r} - \left(\boldsymbol{\mathcal{E}}_6 \boldsymbol{c} - \boldsymbol{\mathcal{E}}_3^\top \boldsymbol{\mathcal{R}} \boldsymbol{u}^{\text{ref}} + \boldsymbol{\mathcal{E}}_2 \begin{bmatrix} \boldsymbol{g}_{\mathbf{x}}^+ - \boldsymbol{\mathcal{G}}_{\mathbf{x}} \boldsymbol{c} \\ -\boldsymbol{g}_{\mathbf{x}}^- + \boldsymbol{\mathcal{G}}_{\mathbf{x}} \boldsymbol{c} \\ \boldsymbol{g}_{\mathbf{u}}^+ \\ -\boldsymbol{g}_{\mathbf{u}}^- \end{bmatrix}_a \right) \tag{2.35}$$

Algorithm 1 Experience-driven Predictive Control

1: $\mathcal{M} \leftarrow \emptyset$ or $\mathcal{M}_{\text{prior}}$
2: **while** control is enabled **do**
3: $\quad \boldsymbol{x} \leftarrow$ current system state
4: $\quad \boldsymbol{r} \leftarrow$ current reference sequence
5: $\quad \mathbf{A}, \mathbf{B}, \tilde{\mathbf{c}} \leftarrow$ current dynamics model from disturbance observer
6: $\quad$ **for** each element $m_i \in \mathcal{M}$ **do**
7: $\quad\quad$ Compute $\boldsymbol{u}, \boldsymbol{\lambda_a}$ via (2.33), (2.35)
8: $\quad\quad$ Store parameterized $\boldsymbol{\mathcal{E}_1} \cdots \boldsymbol{\mathcal{E}_6}$ from (2.33), (2.35) for each set of active constraints
9: $\quad\quad$ **if** $\boldsymbol{x}, \boldsymbol{r}$ satisfy parameterized KKT criteria **then**
10: $\quad\quad\quad$ $\texttt{importance}_i \leftarrow$ current time, sort $\mathcal{M}$
11: $\quad\quad\quad$ $\texttt{controller\_found} \leftarrow$ true
12: $\quad\quad\quad$ Apply affine control law (2.34) from m_i
13: $\quad\quad$ **end if**
14: $\quad$ **end for**
15: $\quad$ **if** `controller_found` is false **then**
16: $\quad\quad$ Apply intermediate control via linear MPC (2.37) with slack variables
17: $\quad\quad$ Update QP formulation with the current model
18: $\quad\quad$ Generate new controller via QP (2.30) (in parallel)
19: $\quad\quad$ **if** $|\mathcal{M}| >$ maximum table size **then**
20: $\quad\quad\quad$ Remove element with minimum `importance`
21: $\quad\quad$ **end if**
22: $\quad\quad$ Add $m_{\text{new}} = (\mathbf{K}, \mathbf{k}_{\text{ff}}, \texttt{importance})$ to $\mathcal{M}$
23: $\quad$ **end if**
24: **end while**

where $\boldsymbol{\mathcal{E}}_6 = \boldsymbol{\mathcal{E}}_3^\top \boldsymbol{\mathcal{B}}^\top \boldsymbol{\mathcal{Q}}$

2.4 EPC Algorithm Overview

As described in Alg. 1, EPC constructs a database defined as a mapping $\mathcal{M}$ from experiences to controllers. For each control iteration, EPC queries the current state and reference, as well as the current affine model from the corresponding disturbance observer, $\mathbf{A}, \mathbf{B}, \tilde{\mathbf{c}}$. The parameterized mapping (line 6) is then queried, and if the KKT conditions are satisfied for an element from $\mathcal{M}$, the corresponding controller is applied. If any element satisfies the criteria, its `importance` value is updated to the current time, and a corresponding affine controller is applied. In this case, no online optimization is required to generate a locally optimal

feedback control law. If no element in $\boldsymbol{\mathcal{M}}$ satisfies the criteria (line 15), a parallelized approach is used to compute and add new element to $\boldsymbol{\mathcal{M}}$ without blocking the main control loop. A local QP is solved (line 17) and corresponding element is added to $\boldsymbol{\mathcal{M}}$. To control the amount of time spent querying the mapping, as new controllers are added to the database, less valuable controllers (indicated by a lower `importance` score) are removed (line 20) to bound the number of elements that may be queried in one control iteration, thereby ensuring computational tractability. With EPC, the local controllers are fully parameterized, allowing controllers computed using past experience to be adapted to the future scenario.

While the new element of $\boldsymbol{\mathcal{M}}$ is being computed, an intermediate/safety controller is used to quickly compute suboptimal commands which ensure stability and constraint satisfaction (line 16). The intermediate controller as a linear MPC with a shorter horizon $\tilde{N}$ and soft constraints is formulated as

$$\begin{aligned}
\underset{\mathbf{u}_k, \boldsymbol{\epsilon}_k}{\operatorname{argmin}} \sum_{k=1}^{\tilde{N}-1} & \frac{1}{2}(\bar{\mathbf{x}}_{k+1} - \bar{\mathbf{r}}_{k+1})^\top \mathbf{Q}_{k+1}(\bar{\mathbf{x}}_{k+1} - \bar{\mathbf{r}}_{k+1}) + \frac{1}{2}(\bar{\mathbf{u}}_k - \bar{\mathbf{u}}_k^{\text{ref}})^\top \mathbf{R}(\bar{\mathbf{u}}_k - \bar{\mathbf{u}}_k^{\text{ref}}) + \frac{1}{2}\boldsymbol{\epsilon}_k^\top \mathbf{S}\boldsymbol{\epsilon}_k \\
\text{s.t.} \quad & \bar{\mathbf{x}}_{k+1} = \mathbf{A}\bar{\mathbf{x}}_k + \mathbf{B}\bar{\mathbf{u}}_k + \tilde{\mathbf{c}} \\
& \mathbf{G}_{\mathbf{x}_{k+1}}\bar{\mathbf{x}}_{\mathbf{k+1}} - \boldsymbol{\epsilon}_k \leq \mathbf{g}_{\mathbf{x}_{k+1}} \\
& \mathbf{G}_{\mathbf{u}_k}\bar{\mathbf{u}}_{\mathbf{k}} \leq \mathbf{g}_{\mathbf{u}_k} \\
& \forall k = 0, ..., N-1
\end{aligned} \tag{2.36}$$

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

As in the local QP, (2.36) can be re-written such that $\mathbf{u}_k$ and $\boldsymbol{\epsilon}_k$ are the only decision variables

$$\begin{aligned}
\underset{\boldsymbol{u}, \boldsymbol{\varepsilon}}{\operatorname{argmin}} \; & \frac{1}{2}\boldsymbol{u}^\top \boldsymbol{\mathcal{H}}\boldsymbol{u} + \boldsymbol{h}^\top \boldsymbol{u} + \frac{1}{2}\boldsymbol{\varepsilon}^\top \boldsymbol{\mathcal{S}}\boldsymbol{\varepsilon} \\
\text{s.t.} \quad & \boldsymbol{\Gamma}\boldsymbol{u} - \boldsymbol{\varepsilon} \leq \boldsymbol{\gamma}
\end{aligned} \tag{2.37}$$

where $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{\varepsilon}$ are aggregated from $\mathbf{S}$ and $\boldsymbol{\epsilon}_k$, respectively.

As this process iterates, $\boldsymbol{\mathcal{M}}$ gets populated by the most useful elements, reducing the dependence on the

intermediate/safety controller (2.37). 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, EPC is able to compute high-rate, stabilizing commands at all times, thus enabling fast, nearly optimization-free control that improves over time.

2.5 Practical considerations

This section introduces some practical insights for leveraging the predictive control formulation strategies by providing considerations for tuning the EPC formulation described in Sec. 2.3 - 2.4. The tuning strategy discussed in this section takes insights from classical control theory of Linear Quadratic Regulator (LQR).

2.5.1 Tuning cascaded EPC gains

For tuning the cascaded predictive control formulation such as EPC, PD control gains are selected which are comparable to an unconstrained version of linear MPC (i.e., finite horizon LQR) to maintain uniformity in the analysis of results. In hindsight, tuning the weights for cascaded EPC formulation is equivalent to tuning this finite horizon LQR formulation. Restating the theory for LQR controller, for a discrete-time system $\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{B}\mathbf{u}_t$, a quadratic cost function is defined as

$$\begin{aligned} J(U) &= \sum_{\tau=0}^{N-1} \left(\mathbf{x}_\tau^\top \mathbf{Q}\mathbf{x}_\tau + \mathbf{u}_\tau^\top \mathbf{R}\mathbf{u}_\tau\right) + \mathbf{x}_N^\top \mathbf{Q}_f \mathbf{x}_N \\ \text{and } U &= \left(\mathbf{u}_0, \cdots, \mathbf{u}_{N-1}\right), \end{aligned} \tag{2.38}$$

where $\mathbf{Q} = \mathbf{Q}^\top \geq 0, \quad \mathbf{Q}_f = \mathbf{Q}_f^\top \geq 0, \quad \mathbf{R} = \mathbf{R}^\top \geq 0$ are state cost, final state cost, and input cost matrices respectively. Here, N is called finite time horizon. First term, second term, and last term measures the state deviation, input/actuator authority, and final state deviation. $\mathbf{Q}$ and $\mathbf{R}$ set relative weights of state deviation and input usage. $\mathbf{R}$ represents any nonzero input adds to cost J. The LQR problem then finds $\mathbf{u}_0^{\text{lqr}}, \cdots, \mathbf{u}_{N-1}^{\text{lqr}}$ that minimizes $J(U)$. Dynamic programming (DP) solution gives an efficient, recursive method to solve this

LQR problem. To adopt the DP approach, a value function (Bellman equation) is defined

$$\mathbf{V}_t(z) = \min_{\mathbf{u}_t,\cdots,\mathbf{u}_{N-1}} \sum_{\tau=t}^{N-1} \left(\mathbf{x}_\tau^\top \mathbf{Q}\mathbf{x}_\tau + \mathbf{u}_\tau^\top \mathbf{R}\mathbf{u}_\tau\right) + \mathbf{x}_N^\top \mathbf{Q}_f \mathbf{x}_N$$

subject to $\mathbf{x}_t = \mathbf{z}$, $\mathbf{x}_{\tau+1} = \mathbf{A}\mathbf{x}_\tau + \mathbf{B}\mathbf{u}_\tau$, $\tau = t, \cdots, T$. $\mathbf{V}_t(z)$ gives the minimum LQR cost-to-go, starting from state z at time t, while $\mathbf{V}_0(\mathbf{x}_0)$ is minimum LQR cost. For the above value function, $\mathbf{V}_t$ is quadratic, i.e., $\mathbf{V}_t(z) = z^\top \mathbf{P}_t z$, where $\mathbf{P}_t = \mathbf{P}_t^\top \geq 0$. $\mathbf{P}_t$ is found recursively, working backward from $t = N$ and the LQR optimal u can be expressed in terms of $\mathbf{P}_t$. Cost-to-go with no time left is then just the final state cost: $\mathbf{V}_N(z) = z^\top \mathbf{Q}_f z$. Thus, $\mathbf{P}_N = \mathbf{Q}_f$ is obtained. The dynamic programming principle states that

$$\mathbf{V}_t(z) = \min_w \left(z^\top \mathbf{Q} z + w^\top \mathbf{R} w + \mathbf{V}_{t+1}(\mathbf{A}z + \mathbf{B}w)\right) \tag{2.39}$$

where $z^\top \mathbf{Q} z + w^\top \mathbf{R} w$ is the cost incurred at time t if $\mathbf{u}_t = w$ and $\mathbf{V}_{t+1}(\mathbf{A}z + \mathbf{B}w)$ is the minimum cost-to-go from $t+1$. Following the fact that minimization is executed in any order, (2.39) can be written as a Bellman equation for LQR as

$$\mathbf{V}_t(z) = z^\top \mathbf{Q} z + \min_w \left(w^\top \mathbf{R} w + \mathbf{V}_{t+1}(\mathbf{A}z + \mathbf{B}w)\right) \tag{2.40}$$

(2.40) gives $\mathbf{V}_t$ recursively in terms of $\mathbf{V}_{t+1}$. Any minimizing w gives optimal $\mathbf{u}_t$ as

$$\mathbf{u}_t^{\text{lqr}} = \operatorname*{argmin}_w \left(w^\top \mathbf{R} w + \mathbf{V}_{t+1}(\mathbf{A}z + \mathbf{B}w)\right)$$

Assuming $\mathbf{V}_{t+1}(z) = z^\top \mathbf{P}_{t+1} z$, with $\mathbf{P}_{t+1} = \mathbf{P}_{t+1}^\top \geq 0$, by DP, (2.40) can be written as

$$\mathbf{V}_t(z) = z^\top \mathbf{Q} z + \min_w \left(w^\top \mathbf{R} w + (\mathbf{A}z + \mathbf{B}w)^\top \mathbf{P}_{t+1}(\mathbf{A}z + \mathbf{B}w)\right) \tag{2.41}$$

(2.41) can be solved by setting the derivative w.r.t. w to zero

$$2w^\top \mathbf{R} + 2(\mathbf{A}z + \mathbf{B}w)^\top \mathbf{P}_{t+1}\mathbf{B} = 0$$

Hence, optimal control input is

$$w^* = -\big(\mathbf{R} + \mathbf{B}^\top \mathbf{P}_{t+1}\mathbf{B}\big)^{-1}\mathbf{B}^\top \mathbf{P}_{t+1}\mathbf{A}z \tag{2.42}$$

while $\mathbf{V}_t(z)$ is

$$\begin{aligned}\mathbf{V}_t(z) &= z^\top \mathbf{Q}z + {w^*}^\top \mathbf{R}w^* + (\mathbf{A}z + \mathbf{B}w^*)^\top \mathbf{P}_{t+1}(\mathbf{A}z + \mathbf{B}w^*) \\ &= z^\top \big(\mathbf{Q} + \mathbf{A}^\top \mathbf{P}_{t+1}\mathbf{A} - \mathbf{A}^\top \mathbf{P}_{t+1}\mathbf{B}(\mathbf{R} + \mathbf{B}^\top \mathbf{P}_{t+1}\mathbf{B})^{-1}\mathbf{B}^\top \mathbf{P}_{t+1}\mathbf{A}\big)z \\ &= z^\top \mathbf{P}_t z\end{aligned}$$

where $\mathbf{P}_t$ for $t = N \cdots 1$ is

$$\mathbf{P}_{t-1} \coloneqq \mathbf{Q} + \mathbf{A}^\top \mathbf{P}_t\mathbf{A} - \mathbf{A}^\top \mathbf{P}_t\mathbf{B}(\mathbf{R} + \mathbf{B}^\top \mathbf{P}_t\mathbf{B})^{-1}\mathbf{B}^\top \mathbf{P}_t\mathbf{A} \tag{2.43}$$

$\mathbf{K}_t$ for $t = 0, \cdots N-1$ from (2.42) is

$$\mathbf{K}_t \coloneqq -\big(\mathbf{R} + \mathbf{B}^\top \mathbf{P}_{t+1}\mathbf{B}\big)^{-1}\mathbf{B}^\top \mathbf{P}_{t+1}\mathbf{A}z \tag{2.44}$$

which follows the optimal control law $\mathbf{u}_t^{\text{lqr}} = \mathbf{K}_t\mathbf{x}_t$. Eq. (2.43), (2.44) run in recursion to compute the finite horizon LQR solution.

For the final application of calculating LQR gains for a comparable feedback response to PD, this report relies on matching the gain matrix generated from LQR formulation to a desired gain matrix obtained from a tuned PD controller. A cost function searches over the values of $\mathbf{Q}$ and $\mathbf{R}$ and minimizes the difference between two gain matrices, $\mathbf{K}_t^{\text{lqr}}$ and $\mathbf{K}_{\text{des}}^{\text{PD}}$. Thus, an optimization solver is ran to solve the following cost function using values from (2.43), (2.44)

$$J(\mathbf{Q}, \mathbf{R}) = \sum \left(\left| \mathbf{K}_t^{\text{lqr}} - \mathbf{K}_{\text{des}}^{\text{PD}} \right| \right) \tag{2.45}$$

With the required components in place, the strategy outlined below is leveraged for obtaining a comparable feedback response of cascaded EPC with respect to a PD controller

Algorithm 2 Gain tuning strategy for cascaded EPC

1: $\mathrm{N} \leftarrow$ Horizon Length
2: $\mathbf{K}_{\text{des}}^{\text{PD}} \leftarrow$ Finite Horizon LQR gain matrix consisting of PD gains initially
3: $\mathbf{A}, \mathbf{B} \leftarrow$ Linearized dynamics model matrices.
4: **while** Updating cost function (2.45) **do**
5: $\quad \mathbf{P} = \mathbf{Q}$
6: $\quad$ **for** Horizon Length from $1, \ldots, \mathrm{N}$ **do**
7: $\quad\quad$ Calculate $\mathbf{K}_t^{\text{lqr}}$ from (2.43), (2.44)
8: $\quad$ **end for**
9: **end while**
10: `gains` $\leftarrow$ Retrieve $\mathbf{Q}$ and $\mathbf{R}$ at the end of optimization routine (line 4)

1. Tune the PD control gains to obtain a performance for comparison and store the gains in a desired gain matrix, $\mathbf{K}_{\text{des}}^{\text{PD}}$. (line 2 of Alg. 2)

2. Generate system linearization matrices, i.e., $\mathbf{A}$ and $\mathbf{B}$, as shown in line 3.

3. Decide the number of optimization variables – this number corresponds to elements inside matrices $\mathbf{Q}$ and $\mathbf{R}$.

4. Leverage the cost function (line 4) defined in (2.45) and calculate the values of $\mathbf{K}_t^{\text{lqr}}$ in recursion using any optimization solver (line 7).

5. The cost function is an absolute difference between the predicted and desired gain matrices. Minimized solution corresponds to the optimal $\mathbf{Q}$ and $\mathbf{R}$ values (line 10).

2.5.2 Tuning Full State EPC gains

PD controller is generally easy to tune because there are only two numbers which need to be altered. However, the full state EPC controller deals with 16 values at the same time. For tuning the full state EPC, the general idea should be to get the feedback response similar to a PD since an unconstrained full state EPC feedback control response is similar to feedback control response obtained from zeroth LQR controller. Consequently, for the experimental purposes of this work, manual hand-tuning of the full state EPC weights is carried out for generating comparable feedback response to PD. To make sure that the feedback response of full state EPC

controller is similar to PD, step inputs in X, Y, and Z directions are analyzed as shown in Fig. 2.2a. Since step responses are based on pure feedback control component, this experiment confirms that the feedback response between full state EPC controller and PD controller is comparable.

2.6 Results

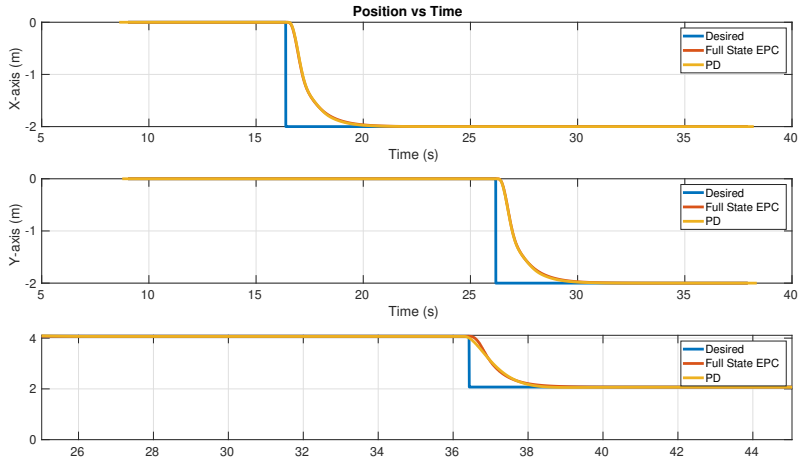
The versatility of the EPC controller is that it can be leveraged for any dynamic model. Throughout this report, illustrative examples for the usage of EPC on position and attitude control (cascaded control), as well as full state controller, are discussed.

This section starts with showing comparable feedback response of full state EPC controller with the PD controller, followed by trials emphasizing the importance of using this full state EPC controller for trajectories requiring sudden acceleration and attitude changes. The real-time performance of full state EPC (FSE) controller on a hexarotor is then presented showing the real-time performance of FSE on a computationally constrained system. This section then presents the results which establish that the feedforward terms are crucial for executing aggressive trajectories.

To assess the performance of the predictive control strategies introduced in this chapter, a set of simulation studies and hardware experiments are conducted with a hexarotor aerial vehicle. Experimental studies with the hexarotor demonstrate the performance of predictive control strategies during aggressive maneuvers. Through this section, following results are demonstrated: **RS1**, real-time computation of control commands; **RS2**, improved trajectory tracking performance while leveraging full state multirotor dynamics; and **RS3**, improved trajectory tracking performance by anticipating future states through feedforward terms. While **RS** shorthand represents the simulation results, **RH** shorthand represents the corresponding hardware results. The experimental hardware platform utilized for this section is a 3.86 kg hexarotor, and the control algorithms are implemented on the hexarotor's NVIDIA TX2 CPU (2 GHz ARM Cortex-A57, 2 GHz NVIDIA Denver2, 8 GB RAM).

2.6.1 Full State EPC controller

To maintain the uniformity in comparison, the full state EPC (FSE) controller is tuned to achieve a comparable feedback response similar to a PD controller. Step responses in X, Y, and Z directions are analyzed as shown



(a) Comparable feedback control response.

Figure 2.2: Figure 2.2a shows step response analysis confirming that a comparable feedback control response is achieved between full state EPC controller and PD controller strategy.

in Fig. 2.2a. The mean control loop time while using FSE for this trial was $\mathbf{2.58}\ ms$, which is well beyond the required control rate (**RS1**), thus proving real-time computation of control commands. A flight trial is also conducted on real system hardware on NVIDIA TX2 using the same gains tuned for simulation trials. The vehicle achieved a mean control loop timing of $\mathbf{5.40}\ ms$ (**RH1**), which is again within the required control rate, thus supporting the real-time control update rate for FSE.

To show the importance of full state controller over a cascaded controller, a high-speed trajectory is commanded without feedforward terms. This experiment emphasizes that full state controller performs relatively well, when executing an aggressive trajectory without the feedforward terms as shown in Fig. 2.3a where the mean absolute tracking error for full state controller in $\mathbf{Z}$ direction was $\mathbf{0.6129}\ cm$ compared to $\mathbf{3.6556}\ cm$ for the cascaded controller. Figure 2.3a shows that full state controller performs well in $\mathbf{Y}$ direction as well, with $\mathbf{2.4505}\ cm$ mean absolute tracking error for full state controller compared to $\mathbf{3.6180}\ cm$ mean absolute tracking error for the cascaded controller.

2.6.2 Feedback and Feedforward response

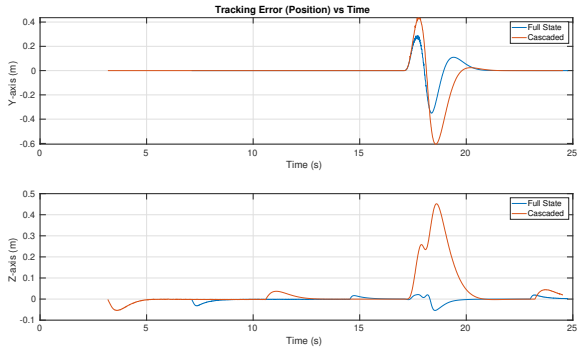
A $4\ m$ long 7^{th} order polynomial trajectory was commanded with a duration of 1.5 seconds on a multirotor platform for conducting this experiment in simulation. Tracking error analysis for this aggressive trajectory is carried out to show improved trajectory tracking with state and input feedforward terms. FSE and Cascaded EPC controller strategy performance are compared with (Fig. 2.3b) and without (Fig. 2.3a) the feedforward terms. Quantitative analysis of the results from Fig. 2.3 suggests that trajectory tracking is substantially improved when state and input references as feedforward terms are utilized in the control formulation (**RS3**).

Finally, a $2.5\ m$ long 7^{th} order polynomial trajectory is commanded with a duration of 1.3 seconds on a hexarotor platform to validate the results on the real hardware system. Flight performance showing improved trajectory tracking is shown in Fig. 2.4 (**RH3**). This figure shows that an aggressive flight is safely executed using feedforward terms (Fig. 2.4a), which otherwise fails when no feedforward terms are utilized (Fig. 2.4b). Aggressive flight video of EPC controller using no feedforward terms, and EPC controller using feedforward terms on a real system are available as embedded links.

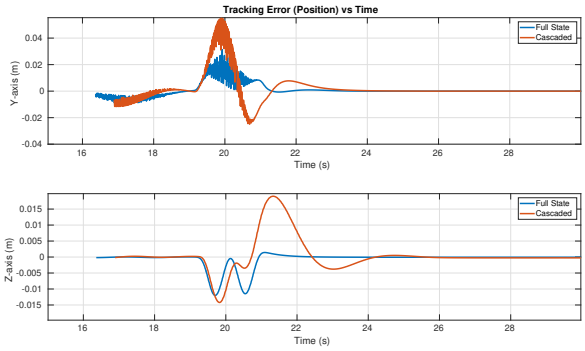
For completeness, the difference between feedback response and feedforward response is shown in Fig. 2.5 for full state EPC controller where the feedback response is visible initially for the commanded step inputs (till ≈ 40 seconds), which diminishes to almost 0 when a higher order (7^{th} order) polynomial trajectory with feedforward terms is commanded.

2.7 Conclusion

This chapter discussed the predictive control formulation, and also introduced the importance of using a controller which considers full state dynamics of the multirotor in a single control loop, instead of a naive cascaded control law strategy. Simulation and hardware results presented in this chapter shows that the proposed predictive control strategies with anticipative nature, derived from feedforward terms improves the trajectory tracking while executing aggressive flights.



(a) Without using feedforward terms.



(b) With using feedforward terms.

Figure 2.3: Figure 2.3a and Fig. 2.3b compares aggressive trajectory tracking performance between full State EPC and cascaded EPC controller without using feedforward terms and with using feedforward terms respectively. Figure 2.3a also shows that full State EPC controller performs better than cascaded EPC controller, which is clearly visible by improved tracking performance in Z direction.

(a) Aggressive trajectory using EPC controller with feedforward terms enabled.

(b) Aggressive trajectory using EPC controller without feedforward terms enabled.

Figure 2.4: Figure 2.4a and Fig. 2.4b shows overlaid aggressive trajectory execution using EPC controller with feedforward terms enabled and without feedforward terms enabled respectively.

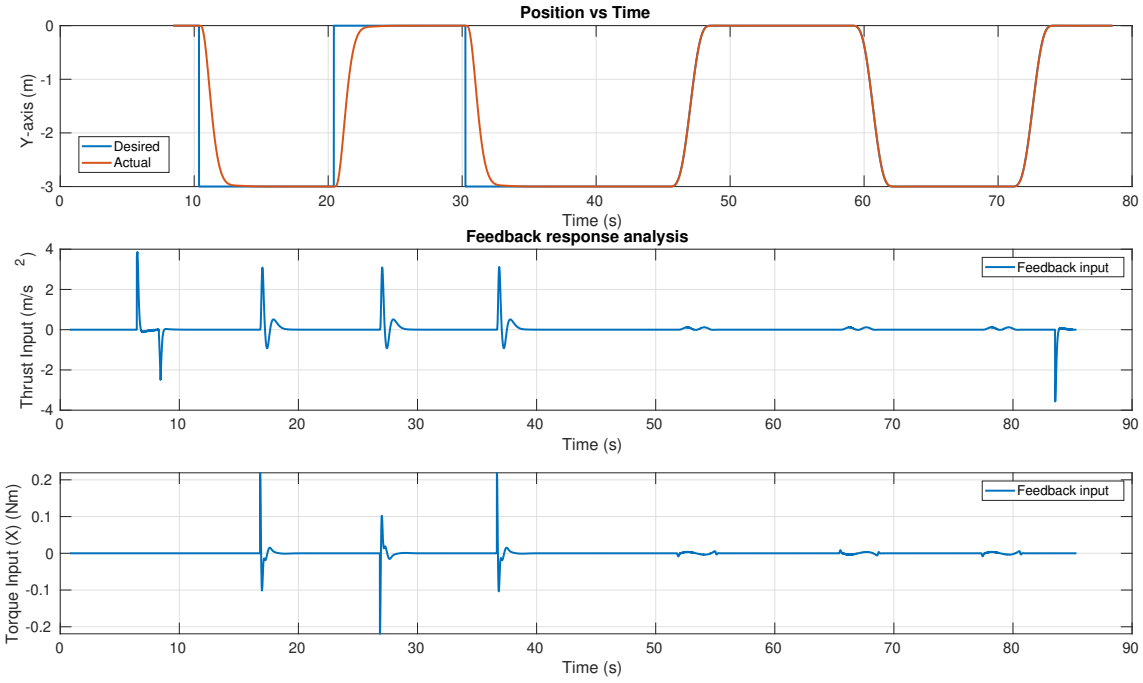


Figure 2.5: This figure shows feedback response for step inputs given to the full state EPC controller till ≈ 40 seconds, followed by feedback and feedforward response of the controller for the 7^{th} order polynomial trajectory utilizing jerk and snap components. Above figure shows that feedback input is large for step inputs while the feedback component of control input is almost zero for the trajectories with feedforward terms.

3 Robust Constraint Satisfaction for Aggressive Flights

This chapter details the importance of enforcing state and control input constraints using a finite-horizon predictive controller strategy, Experience-driven Predictive Control (EPC) described in Sec. 2.3. EPC reuses the experience to reduce online optimization through the use of stored parameterized controllers. Efficient implementation of EPC on computationally constrained platforms is discussed in this chapter by leveraging explicit EPC with Markov Chain simplification [3].

There is a considerable demand for flying aggressive and high speed controlled flights for fast exploration and inspection missions where resources to fly for a long duration are severely constrained [21]. Moreover, drone racing has also become very popular nowadays, demanding accuracy as well as safety while executing such aggressive maneuvers [22]. While flying aggressively, the MAV is prone to disastrous crashes due to the inability of the control strategies for adapting to the sudden changes in the vehicle's angular acceleration. It is thus, essential to enforce safety using predictive control strategies. Predictive control strategies such as Model Predictive Control (MPC) generates safe control commands by enforcing constraints on the system state and control input. Unconstrained controller strategies such as a PD control or an LQR controller cannot handle sudden significant changes in desired control input and state during aggressive flights. This is because, with the unconstrained inputs, the system might hit its physical actuator limits while trying to maneuver the aggressive trajectory. Such significant and sudden changes in the system's attitude can lead to large required angular acceleration, causing motor saturation. Thus, unconstrained control strategies result in poor performance by violating the safety of the MAV for maneuvering aggressive trajectories. Hence, when an unconstrained controller is used to track an aggressive trajectory, a disastrous situation might arise as shown in Fig. 3.2d where the robotic system crashes, resulting in substantial damages to the system. In order to face such situations, nonlinear model predictive control (NMPC) strategies are often leveraged for a nonlinear system such as the MAV. However, due to the shortcomings of NMPC (described in Sec. 2), a linear MPC control formulation is leveraged which stores the parameterized controller in a database for future re-use, using an approach known as EPC [23]. While there are certain merits of using EPC, this approach still needs to solve a QP in parallel and rely on a suboptimal solution such as shorter-horizon QP if the controller is not found in its parameterized controller database. It is thus not efficient to query every parameterized controller in the database which might destabilize the control update rate due to the exceeding time delays.

During real flights, the system only uses a small number of the potential controllers that nominal EPC approach generates. To enhance the efficiency of controller queries, the transitions between the database queries is modeled as a Markov chain empirical transition probabilities. There transition probabilities are used to specify a partial ordering on successors for each controller [3]. This approach improves the efficiency of database queries and permits further reductions of the database size, thereby enabling use on computationally constrained platforms.

The task of enforcing constraints for maintaining the safety of the MAV is challenged in the presence of uncertain state estimates. Robust predictive control strategies deal with such situations by using a conservative solution of tightening/decreasing the constraint bound limit. Due to the tightening of constraints, the system always tries to stay within the constraint bound, thus enforcing safety. Following [23], Robust EPC is presented in this chapter, which tightens the constraint bounds in the presence of uncertain state estimates and stores the parameterized robust controllers in a database for reusing them when a similar scenario is encountered. Explicit Robust EPC with Markov Chain simplification is also presented in this chapter which models the transitions of robust controllers to define an efficient query order for efficient query times.

This chapter is detailed as follows – Section 3.1 details the explicit EPC formulation with Markov Chain ordering simplification algorithm overview. Section 3.2 describes the robust formulation for constrained predictive control strategies using Robust EPC. Finally, Sec. 3.3 presents the results of constrained predictive control using EPC, and explicit EPC with Markov Chain ordering simplification. Results for cached matrices logic for EPC are also presented in this section, that introduces a factor of speedup in querying the database. Along with this, the results for Robust EPC, and explicit Robust EPC with Markov Chain ordering are also discussed in Sec. 3.3.4 - 3.3.5. All the experiments are conducted in simulation as well as real hardware systems that shows the viability of real-time control command generation of the presented control strategies.

3.1 Explicit EPC with Markov Chain Ordering

In this section, the algorithm overview for the efficient implementation of the constrained controller strategies using Markov Chain (MC) simplification is discussed. This MC simplification defines a partial ordering of controller query priority in the database.

Algorithm 3 Controller Database Generation using EPC

1: $\boldsymbol{\mathcal{M}} \leftarrow \emptyset$, $\boldsymbol{\Phi} \leftarrow 0$, $j \leftarrow 0$
2: **while** control is enabled **do**
3: $\quad \boldsymbol{x} \leftarrow$ current system state
4: $\quad \boldsymbol{r} \leftarrow$ current reference sequence
5: $\quad$ **for** each element $m_i \in \boldsymbol{\mathcal{M}}$ **do**
6: $\quad\quad$ Compute $\boldsymbol{u}, \boldsymbol{\lambda_a}$ via (2.33), (2.35)
7: $\quad\quad$ Store parameterized $\boldsymbol{\mathcal{E}_1} \cdots \boldsymbol{\mathcal{E}_6}$ from (2.33), (2.35) for each set of active constraints
8: $\quad\quad$ **if** $\boldsymbol{x}, \boldsymbol{r}$ satisfy parameterized KKT criteria **then**
9: $\quad\quad\quad$ `importance`$_i \leftarrow$ current time, sort $\boldsymbol{\mathcal{M}}$
10: $\quad\quad\quad$ `controller_found` $\leftarrow$ true
11: $\quad\quad\quad$ Apply affine control law (2.34) from m_i
12: $\quad\quad\quad$ $\boldsymbol{\Phi}_{ij} \leftarrow \boldsymbol{\Phi}_{ij} + 1$, $j \leftarrow i$
13: $\quad\quad\quad$ **break**
14: $\quad\quad$ **end if**
15: $\quad$ **end for**
16: $\quad$ **if** `controller_found` is false **then**
17: $\quad\quad$ Apply intermediate control via linear MPC (2.37) with slack variables
18: $\quad\quad$ Update QP formulation with the current model
19: $\quad\quad$ Generate new controller via QP (2.30) (in parallel)
20: $\quad\quad$ **if** $|\boldsymbol{\mathcal{M}}| >$ maximum table size **then**
21: $\quad\quad\quad$ Remove element with minimum `importance`
22: $\quad\quad$ **end if**
23: $\quad\quad$ Add $m_{\text{new}} = (\mathbf{K}, \mathbf{k}_{\text{ff}}, \texttt{importance})$ to $\boldsymbol{\mathcal{M}}$
24: $\quad\quad$ $j \leftarrow |\mathcal{M}|$, $\boldsymbol{\Phi}_{jj} \leftarrow 1$
25: $\quad$ **end if**
26: **end while**
27: Apply a first order Markov Chain to compute ordering ($\boldsymbol{\Omega}$) over the obtained transition matrix ($\boldsymbol{\Phi}$) which consists of transition frequencies between most probable states.

3.1.1 Controller Database Generation

In order to generate a database, $\mathcal{M}$, of affine feedback controllers, regular EPC algorithm is initialized (previously described in Alg. 1) with an empty database, as shown in Alg. 3. A transition frequency matrix, $\boldsymbol{\Phi}$ is introduced, that records the number of transitions between each pair of controllers added to $\mathcal{M}$ (line 1). As the MAV encounters new scenarios and increases the current database of controllers, $\boldsymbol{\Phi}$ is incremented accordingly (line 12). If the relevant controller is not found in the database, EPC solves QP (2.30)(line 19) and adds the resulting controller to the database (increasing the size of $\boldsymbol{\Phi}$ as well) (line 23), while using a

Algorithm 4 Querying MC simplified controller database online

1: **Inputs:** $\mathcal{M}$ and $\boldsymbol{\Omega}$ from Alg. 3, Current $\boldsymbol{x}, \boldsymbol{r}$, Previous controller index $j^\star$
2: **for** each element $m_i \in \mathcal{M}$ ordered by $\Omega_{j^\star}$ **do**
3: **if** $\boldsymbol{x}, \boldsymbol{r}$ satisfy KKT criteria (2.31) **then**
4: `controller_found` $\leftarrow$ true
5: **return** $\boldsymbol{u} \leftarrow \boldsymbol{\kappa}_i(\boldsymbol{x}, \boldsymbol{r})$
6: **end if**
7: **end for**
8: **if** `controller_found` is false **then**
9: Apply intermediate control via linear MPC (2.37) with slack variables
10: **end if**

suboptimal shorter horizon intermediate controller (line 17).

At the end of the trial, an order-1 Markov chain is defined that represents the transitions between controllers, with empirical transition probabilities defined by $\bar{\boldsymbol{\Phi}} = \boldsymbol{\Phi}$ with normalized rows. Sorting the outgoing transitions from each state of the Markov chain according to the probability of occurrence yields a partial ordering, $\boldsymbol{\Omega}$, of the controllers (line 27). This ordering informs the online query process detailed in Sec. 3.1.2. To know more about the benefits of this ordering strategy, the reader is encouraged to refer [3, 24].

3.1.2 Querying the MC ordered EPC database online

After the controller database, $\mathcal{M}$, is populated from Alg. 3, it enables high-rate, online control that recovers the functionality of (2.30). As discussed in Alg. 4, $\mathcal{M}$ is queried in each control iteration to identify the appropriate controller $\boldsymbol{\kappa}_i$. Since every control iteration is a state transition in the Markov chain, the next transition (next controller) is identified by iterating through $\mathcal{M}$ according to the order specified by entry $j^\star$ in $\boldsymbol{\Omega}$, where $j^\star$ is the index of the previous controller applied (line 2). This MC ordering makes it possible to reduce the number of controllers that must be checked (because evaluating the KKT conditions is comparatively more expensive than a lookup table query), thus improving the query efficiency relative to simple ordering from nominal EPC.

Moreover, since the controller database query is only as good as the experiences collected by flying online, it cannot guarantee perfect coverage. Generation of an extensive synthetic experience database as detailed in [3] might be helpful here. Therefore, if no controller in the database is suitable, an intermediate/safety

controller (e.g., a short horizon MPC with soft constraints) is applied until the system returns to the region covered by $\mathcal{M}$.

3.2 Robust Predictive Control

In this section, an extension of the EPC algorithm [25] to achieve high-rate predictive control with robust constraint satisfaction is presented following [23]. Robust EPC algorithm parameterizes the controllers in the database similar to the EPC algorithm presented in Sec. 2.4 by an online updated estimate of the uncertainty in the system state. The state estimate is derived from the estimator covariance and thus enables the use of a belief propagation approach to create an uncertainty tube for the evolution of the state over the prediction horizon. Following nonlinear dynamics and observation model is considered for the Robust EPC algorithm presented in this section

$$\begin{aligned} \mathbf{x}_{k+1} &= f(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{w}_k \\ \mathbf{z}_k &= h(\mathbf{x}_k) + \mathbf{v}_k \end{aligned} \tag{3.1}$$

with $\mathbf{x}_k$ being the system state, $\mathbf{u}_k$ being the control input to the system, $\mathbf{w}_k \sim \mathcal{N}(0, \mathbf{W}_k)$, and $\mathbf{v}_k \sim \mathcal{N}(0, \mathbf{V}_k)$ denoting the process and measurement uncertainties, respectively. To linearize the system dynamics about nominal state $\mathbf{x}^\star$ and nominal input $\mathbf{u}^\star$, first order Taylor-series approximation of (3.1) is required

$$\begin{aligned} \mathbf{x}_{k+1} &\approx \mathbf{A}_k(\mathbf{x}_k - \mathbf{x}^\star) + \mathbf{B}_k(\mathbf{u}_k - \mathbf{u}^\star) + \tilde{\mathbf{c}} + \mathbf{w}_k \\ \mathbf{z}_k &\approx \mathbf{C}_k(\mathbf{x}_k - \mathbf{x}^\star) + \mathbf{v}_k \end{aligned} \tag{3.2}$$

Extending (3.1) to a standard Kalman filter belief state update law following the insight from [23] yields an estimate of the state $\mathbf{x}_k \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, which models the evolution of this uncertain system

$$\begin{aligned} \boldsymbol{\mu}_{k+1} &= f(\boldsymbol{\mu}_k, \mathbf{u}_k) + \mathbf{P}_k \mathbf{C}_k^\top \mathbf{L}_k^{-1} \big(\mathbf{z}_{k+1} - h(\boldsymbol{\mu}_k)\big) \\ \boldsymbol{\Sigma}_k &= \mathbf{P}_k - \mathbf{P}_k \mathbf{C}_k^\top \mathbf{L}_k^{-1} \mathbf{C}_k \mathbf{P}_k \end{aligned} \tag{3.3}$$

where $\mathbf{P}_k = \mathbf{A}_k\boldsymbol{\Sigma}_k\mathbf{A}_k^\top + \mathbf{W}_k$ and $\mathbf{L}_k = \mathbf{C}_k\mathbf{P}_k\mathbf{C}_k^\top + \mathbf{V}_k$. Simplifying further by taking $\mathbf{z}_{k+1} = h(\boldsymbol{\mu}_k)$as the maximum likelihood observation following assumption from [23]

$$\begin{aligned}\boldsymbol{\mu}_{k+1} &= f(\boldsymbol{\mu}_k, \mathbf{u}_k)\\ \boldsymbol{\Sigma}_{k+1} &= \mathbf{P}_k - \mathbf{P}_k\mathbf{C}_k^\top\mathbf{L}_k^{-1}\mathbf{C}_k\mathbf{P}_k\end{aligned} \tag{3.4}$$

Restating [23], the tube MPC formulation enforces state and input constraints by assuming that the admissible state and input sets, $\mathcal{X}_k$ and $\mathcal{U}_k$, respectively can be approximated by polytopes, yielding a set of half-plane constraints

$$\begin{aligned}\mathbf{G}_\mathbf{x}(\mathbf{x}_{k+1} - \mathbf{x}^\star) &\leq \mathbf{g}_\mathbf{x}\\ \mathbf{G}_\mathbf{u}(\mathbf{u}_k^S - \mathbf{u}^\star) &\leq \mathbf{g}_\mathbf{u}\end{aligned} \tag{3.5}$$

where $\mathbf{u}_k^S$ is the combination of MPC output, $\mathbf{u}_k$, and an ancillary stabilizing controller with gain matrix $\mathbf{S}_k$, as proposed by [23]. To maintain stochastic nature of the dynamics model, a chance constrained formulation is applied over (3.5), to hold with probability $1-\alpha$

$$\begin{aligned}P\Big(\mathbf{G}_\mathbf{x}\big(\mathbf{x}_{k+1} - \mathbf{x}^\star\big) \leq \mathbf{g}_\mathbf{x}\Big) &\geq 1-\alpha\\ P\Big(\mathbf{G}_\mathbf{u}\big(\mathbf{u}_k^S - \mathbf{u}^\star\big) \leq \mathbf{g}_\mathbf{u}\Big) &\geq 1-\alpha\end{aligned} \tag{3.6}$$

The ellipsoidal bounds, defined by $\boldsymbol{X}^2$, containing $(1-\alpha)$ of the probability mass is given by

$$(\mathbf{x}-\boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}) = \boldsymbol{X}_n^2(\alpha) \tag{3.7}$$

The ellipsoidal bound $\boldsymbol{\delta}_{k+1}^\mathbf{x}$ is approximated by its axis-aligned bounding box is given by

$$\boldsymbol{\delta}_{k+1}^\mathbf{x} = \sqrt{\boldsymbol{X}_n^2(\alpha)\text{diag}(\boldsymbol{\Sigma}_{k+1})} \tag{3.8}$$

yielding final tightened state constraint bounds. Since the ancillary controller is a function of uncertain future state, a similar bound on control command is formulated

$$\boldsymbol{\delta}_k^\mathbf{u} = \sqrt{\boldsymbol{X}_n^2(\alpha)\text{diag}(\mathbf{S}_k\boldsymbol{\Sigma}_k\mathbf{S}_k^\top)} \tag{3.9}$$

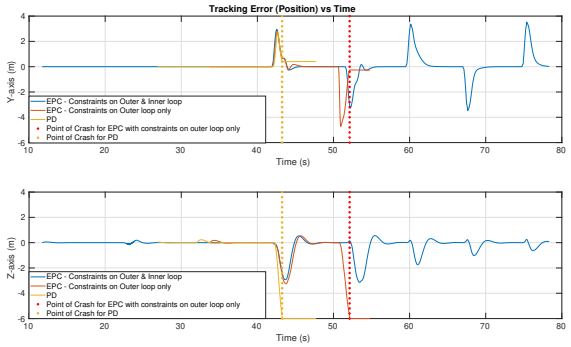
Finally, the generated probabilistic state and input bounds are converted into tightened deterministic constraints as

$$\begin{aligned} \mathbf{G}_{\mathbf{x}}(\boldsymbol{\mu}_{k+1} - \mathbf{x}^{\star}) &\leq \mathbf{g}_{\mathbf{x}} - \mathbf{G}_{\mathbf{x}}\boldsymbol{\delta}^{\mathbf{x}}_{k+1} = \tilde{\mathbf{g}}_{\mathbf{x}} \\ \mathbf{G}_{\mathbf{u}}(\mathbf{u}_{k} - \mathbf{u}^{\star}) &\leq \mathbf{g}_{\mathbf{u}} - \mathbf{G}_{\mathbf{u}}\boldsymbol{\delta}^{\mathbf{u}}_{k} = \tilde{\mathbf{g}}_{\mathbf{u}} \end{aligned} \tag{3.10}$$

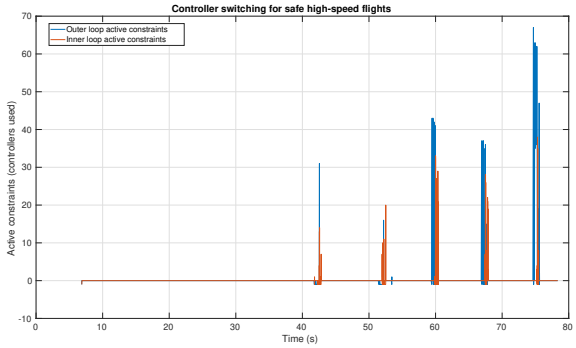
3.3 Results

This section starts with results emphasizing the importance of constrained control strategies over non-constrained control strategies for executing aggressive flight maneuvers. This set is then followed by the discussion of results, which shows that Markov Chain Ordered (MCO) database with explicit EPC gives substantially efficient control update rate with reduced safety control calls. Computational savings memoization is then introduced to further improve the controller database querying by precomputing and caching the expensive matrices in EPC formulation so that the cost is incurred only once on database entry and not every control iteration, i.e., we are proposing to tradeoff memory for CPU time. A similar set of results are also presented for the robust counterpart of EPC for proving the viability of the presented control strategies during varying state uncertainty. To enforce the versatility of the EPC controller, illustrative results for the usage of EPC on cascaded control, as well as full state controller are also discussed similar to Sec. 2.6.

To assess the performance of the predictive control strategies, a set of simulation studies and hardware experiments are conducted with a hexarotor aerial vehicle. Experimental studies with the hexarotor demonstrate the performance of predictive constrained control strategies during aggressive maneuvers. Through this section, following results are demonstrated: **RS1**, real-time computation of control commands; **RS2**, improved trajectory tracking performance; **RS3**, constraint satisfaction; **RS4**, improved tracking performance during uncertain state estimates; **RS5**, constraint satisfaction during uncertain state estimates; **RS6**, reuse of past experiences to improve control update rate and performance; and **RS7**, improved control update rate and performance by precomputing and caching expensive EPC matrices. While **RS** shorthand represents the simulation results, **RH** shorthand represents the corresponding hardware results. The experimental hardware platform utilized for this section is a 3.86 kg hexarotor, and all the control algorithms are implemented in C++ via ROS [26] and run in real time on the hexarotor's NVIDIA TX2 CPU. The communication latency from the TX2 to the motor controllers is sub-millisecond.



(a) Overlaid tracking error in position vs time.



(b) Constraints enforced on outer and inner loop.

Figure 3.1: Figure 3.1a shows that only the EPC controller with constraints enforced on the outer & inner loop is able to track the extremely aggressive trajectory. Consequently, the trajectory tracking error for this controller reduces in Z after subsequent experience generation and reuse. Figure 3.1b shows controller switching and experience reuse for cascaded EPC controller when the constraints are enforced on desired acceleration (outer loop) as well as roll/pitch angles (inner loop) – y axis represents controller indices and the value on y axis at a given time (x axis) shows the index of the controller that was used.

3.3.1 Constrained controller implications for aggressive flights

In this section, the performance analysis of constrained predictive controllers for highly aggressive trajectories is conducted. The experiments in this section show that enforcing constraints helps MAV track the otherwise infeasible aggressive trajectory. For extremely aggressive trajectories, it is often required that constraints on desired acceleration (outer loop) and roll/pitch angles or angular velocity (inner loop) are enforced to maintain the feasibility of the trajectory instead of just enforcing the constraints on desired acceleration (outer loop). To validate the proposed controller strategies, a trajectory with desired speed $\approx$ 12 m/s, and peak acceleration reaching ≈ 20 m/s^2 $\approx 2g$ is commanded to the MAV in simulation.

Figure 3.1a shows that an unconstrained EPC/PD controller, as well as a cascaded EPC controller with a constraint of $14.6\ m/s^2$ enforced only on desired acceleration (outer loop) were not able to execute the aggressive trajectory which required sudden attitude changes and very high desired acceleration values at the same time. Thus, the above trajectory was only realized when the constraints of $14.6\ m/s^2$ and $1.3\ rad$ were enforced on desired acceleration and roll/pitch angles simultaneously. This experiment shows that the simultaneous constraints on state and control inputs enforce better safety and hence provide more stable as well as improved control performance (**RS2**, **RS3**) as compared to the unconstrained controllers. Figure 3.1b

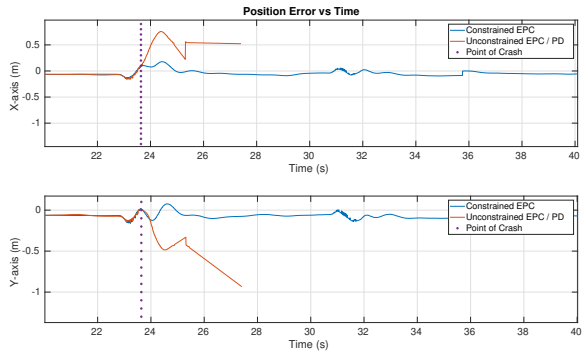
	Query time (ms)		QP (ms)		Intermediate QP(ms)		Total solve time (ms)	
	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
Inner loop	0.1319	0.3250	1.8752	1.3388	2.0582	0.9003	**1.6083**	**1.1232**
Outer loop	0.2571	2.0559	1.3465	0.7490	0.4459	0.2005	**2.2414**	**3.4469**

Table 3.1: Timing characteristics for cascaded EPC controller strategy (shown in Fig. 3.1) when constraints are enforced on the outer and inner loop. Each column represents the mean and standard deviation timing for querying the cascaded EPC controller database, solving the quadratic program when an optimal controller is not found in the database, solving a shorter horizon quadratic program (intermediate QP) while solving full horizon quadratic program, and total control loop response time respectively. The bold values represent the timing values to be well-within the stable control rate required for cascaded outer and inner loops.

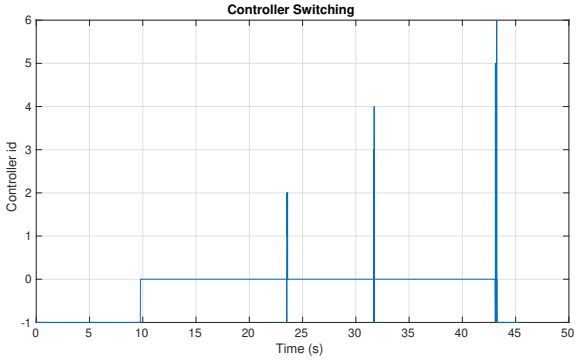
shows that past experiences are reused to improve the trajectory tracking performance (**RS6**). Mean control loop update time for this trial is well within the update rate required for cascaded controllers to function correctly as shown in Table 3.1, thus proving real-time generation of control commands (**RS1**).

A similar experiment is conducted on a real hardware system for validating the real-time control update rate on computationally constrained systems. Aggressive higher order polynomial trajectories are commanded similar to the simulation experiment discussed above, with the state and control input constraints enforced on the controller. A straight line trajectory of 7^{th} order polynomial along the y-axis is commanded to the MAV over 4 meters in 1.2 seconds. A similar comparison study is then carried out between unconstrained EPC / PD controller and constrained cascaded EPC with tight control input constraints of 14.6 m/s^2. Figure 3.2a shows that constrained controllers perform better over unconstrained controllers in that the constrained controllers are able to track the corresponding aggressive trajectories safely without crashing on real systems (**RH2**, **RH3**) along with reusing previous controllers online as shown in Fig. 3.2b (**RH6**). Constrained cascaded EPC controllers takes $\mathbf{2.9}$ ms mean total control loop time, with a standard deviation of $\mathbf{0.5877}$ ms showing real-time computation of control commands (**RH1**). Figure 3.2c shows an overlaid image of successful execution of an aggressive flight trial when constraints are enforced on the controller while Fig. 3.2d shows an overlaid image of MAV crashing due to the non-enforcement of constraints.

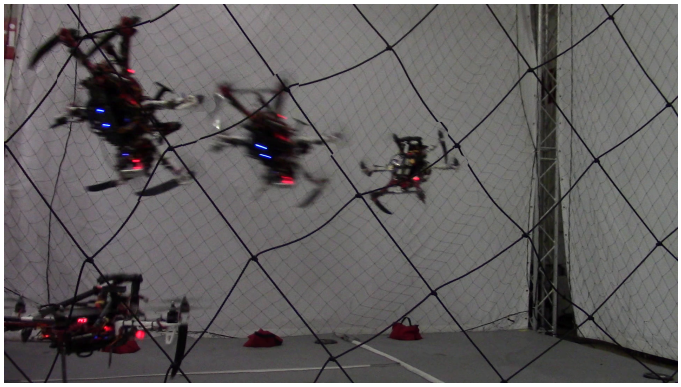
Large step inputs are generally an aggressive form of controller behavior. The next experiment trial is designed to compare the feedback response of the controllers when given a large step response, a common



(a) Position Tracking Error vs. Time



(b) Controller switching

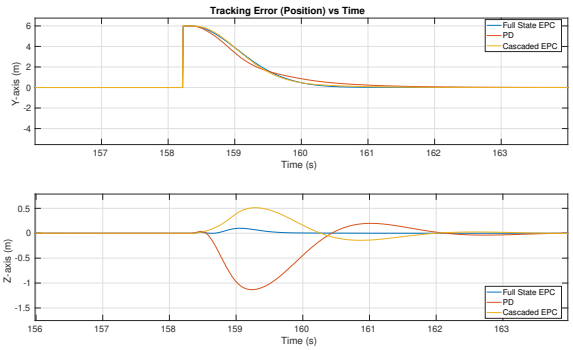


(c) Constrained controller performance (safe)

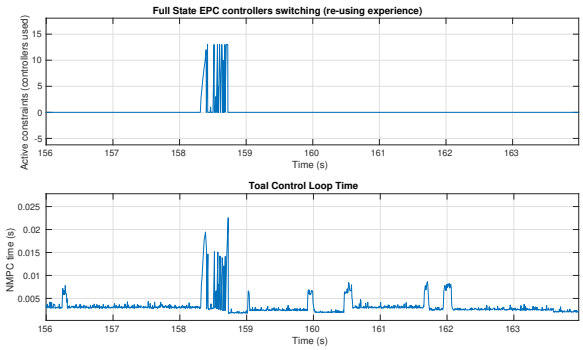
(d) Unconstrained controller performance (fail)

Figure 3.2: **(On real systems)** Figure 3.2a shows that unconstrained EPC is unable to track the aggressive trajectory whereas constrained EPC is able to track the trajectory along with storing and switching the corresponding controllers in the database (Fig. 3.2b). Figure 3.2c shows an overlaid image of the constrained controller safely executing aggressive trajectory, while Fig. 3.2d shows an overlaid image of MAV crashing while executing the same aggressive trajectory. This image emphasizes the harmful consequences of not enforcing constraints on the controller.

situation when trying to execute an aggressive trajectory. To support the versatility of EPC controller, experiments using Full State EPC (FSE) controller with similar constraints on control input and roll/pitch angles as above, are conducted and compared with cascaded EPC and PD controller counterparts. Hence, a 6m step response is analyzed for comparing FSE, cascaded EPC, and PD controller strategy. As evident in Fig. 3.3, FSE provides smoother and comparatively better step response over cascaded PD controller. The cascaded PD controller accumulates substantial error in Z, thereby affecting trajectory tracking performance. (**RS2**, **RS3**). Mean control update time for FSE is $\mathbf{3.1377}$ ms with a standard deviation of $\mathbf{1.6793}$ ms, which shows the real-time computation of control commands (**RS1**). Figure 3.3b also shows that FSE re-uses past



(a) Overlaid tracking error in position vs time.



(b) Controller switching for full state EPC controller.

Figure 3.3: Figure 3.3a shows that the controller performance is substantially improved when a full state EPC controller is used for handling large step inputs when compared to a PD and a cascaded EPC controller, evident by reduced tracking error in Z direction. Controller switching (y axis represents controller indices) and control loop timing analysis for full state EPC controller is shown in upper and lower parts of Fig. 3.3b respectively

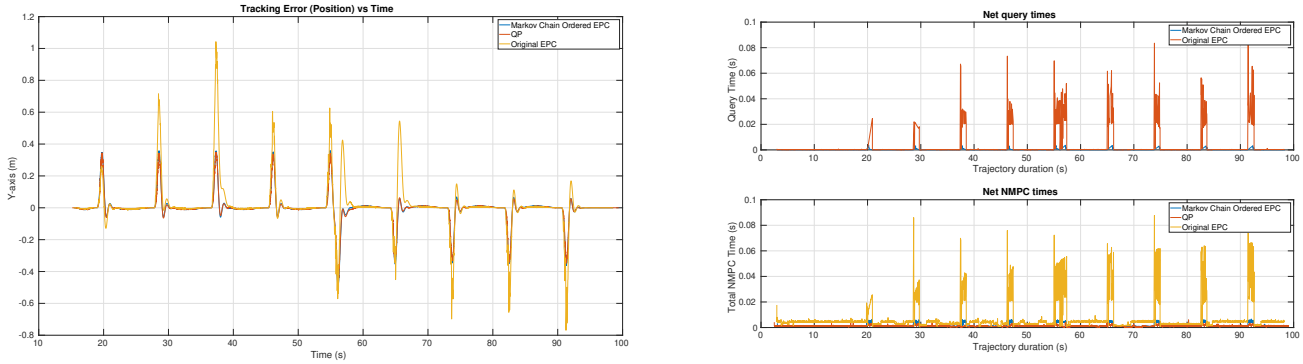
.

experiences over the course of trial for efficient control update rate (**RS6**).

3.3.2 Explicit EPC with Markov Chain Ordering

Cascaded EPC controller with Markov Chain Ordering (MCO) simplification strategy described in Sec. 3.1.2 gives substantially better control-loop update rate by specifying a prioritized ordering of the controller database. While executing long duration trajectories of back and forth trials, regular EPC is unable to maintain the nominal control-loop update rate after a large number of controllers are computed, resulting in subpar control performance. Linear MPC (QP controller), on the other hand, solves a quadratic program on each iteration and gives an optimal solution in comparatively much less time. This control-loop update rate issue is a strong motivation for leveraging Markov Chain (MC) transition logic mentioned in Sec. 3.1.2 for retrieving the optimal control-loop update rate while leveraging previous collective experience. This experiment thus details the comparison between linear MPC (QP controller), regular EPC, and MCO-EPC.

MCO-EPC is capable of performing similar to a linear MPC controller as shown in Fig. 3.4a, with reasonable query times by collectively leveraging previous experience in an efficient query order (**RS1**, **RS2**, **RS6**). On the other hand, regular EPC control-loop update rate is considerably worse, as shown in Fig. 3.4b



(a) Position Tracking Error vs. Time

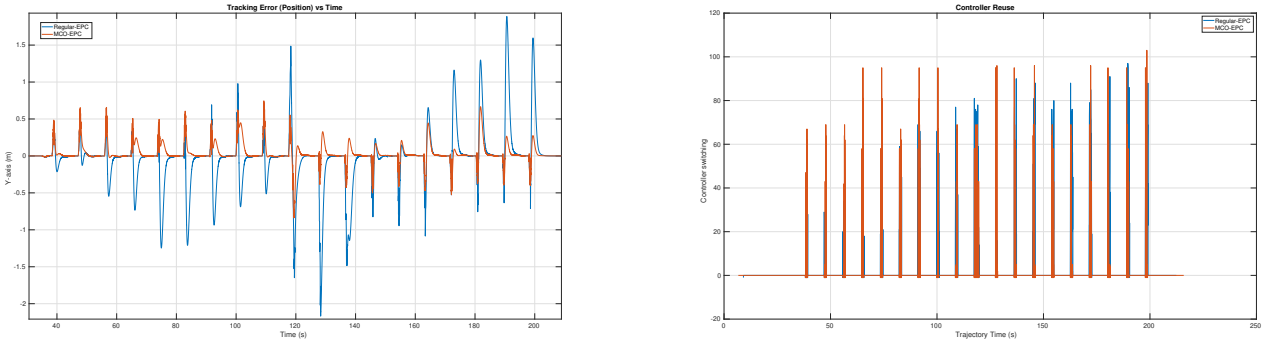
(b) Timing Characteristics

Figure 3.4: 3.4a shows that QP and Markov Chain (MC) Ordered EPC perform well here, while regular EPC (Original EPC) controller performs considerably worse after learning most of the controllers. Figure 3.4b(i) shows query time for MC Ordered EPC is comparatively much less than regular EPC because of the transition matrices logic, but sometimes the timing for MC Ordered EPC is more than QP when the constraints are active. Figure 3.4b(ii) shows that regular EPC controller runs into timing issues as more and more controllers are computed, while MC ordered EPC leverages the previous experience and performs $\approx$ similar to QP controller. Videos executing above aggressive trial in simulation, with excerpts from $70 - 100$ seconds using QP control, regular EPC, and MC Ordered EPC are available here, here, and here respectively.

which explains the subpar control performance in Fig. 3.4a.

Similar aggressive trajectory trial is conducted on a real hardware system using NVIDIA TX2 in order to confirm the real-time viability of MCO-EPC on computationally constrained systems. The tracking performance and controller database usage in Fig. 3.5 signifies improved tracking performance (**RH2**) of MCO-EPC over regular EPC due to efficient ordering specified by Markov Chain simplification (**RH6**). The controller database query time for regular EPC was $\mathbf{1.0671 \pm 6.4080}$, while the query time for MCO-EPC was $\mathbf{0.4100 \pm 2.6307}$ (**RH1**). The stark difference in query time directly correlates with the improvement in tracking performance observed in Fig. 3.5a.

For running the explicit MCO-EPC at the embedded level, position control and attitude control loops are implemented to run on the separate threads so as not to block the control-loop update rate while querying the controller database. To experimentally validate the MCO simplification strategy, a controller database is generated by flying random trajectories in space. Explicit MCO-EPC at the embedded level shows real-time computation of control commands, as (**RH1**) shown in Table 3.2, with a controller coverage of $\mathbf{97.2186}\%$. Constrained control (**RH3**) along-with experience re-use (**RH6**) by controller switching for this experiment



(a) Position Tracking Error vs. Time

(b) Controller database experience reuse

Figure 3.5: Improved trajectory tracking using MCO is visible in Fig. 3.5a when the aggressive trial is executed on NVIDIA TX2 (**RH2**). The controller switching in Fig. 3.5b shows that MCO-EPC uses the same controllers from the regular EPC database with a different query priority for choosing the best-transitioned controllers (**RH6**).

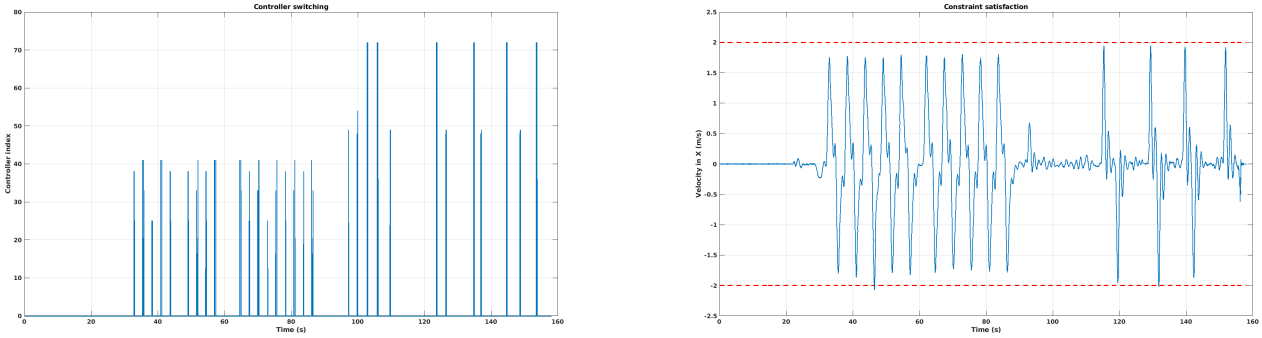
	Mean Query Time	Std. Dev. Query Time	Size of Controller Database	Database queries	Safety Control Calls	Coverage
Regular EPC	2.09 ms	2.39 ms	72	-	-	-
Explicit MCO-EPC	1.96 ms	0.32 ms	72	20637	574	97.2186 %

Table 3.2: Timing statistics showing real-time computation of control commands for a computationally constrained system (**RH1**).

is shown in Fig. 3.6b and Fig. 3.6a respectively.

3.3.3 Precomputing and Caching Expensive Matrices

Precomputing and caching $\boldsymbol{\mathcal{E}}_1, \ldots, \boldsymbol{\mathcal{E}}_6$ matrices of (2.33), (2.35) for each set of active constraints as new controllers and corresponding active constraint sets are calculated, is a potential area of improving the querying efficiency. Analyzing the compute times for EPC, it is evident that precomputing the $\boldsymbol{\mathcal{E}}_1, \ldots, \boldsymbol{\mathcal{E}}_6$ (2.33), (2.35) for each set of constraints can speedup control-loop update rate by reducing the query time, so that the cost is incurred only once on database entry and not on every control iteration, i.e., tradeoff memory for CPU time. Hence, $\boldsymbol{\mathcal{E}}_1, \ldots, \boldsymbol{\mathcal{E}}_6$ can be stored and remembered alongside each controller, as they are generated, since they do not depend on the state or reference.



(a) Controller switching

(b) Velocity constraint satisfaction

Figure 3.6: Figure 3.6a shows controller switching **(R5)** based on the ordering of controllers (successors) from MC simplification. Figure 3.6b shows velocity constraint satisfaction **(R2)**.

For checking the validness of this approach, a trial was conducted by executing the same aggressive back and forth trajectory as previous set of experimental results with constraints enforced on desired acceleration input so that a large number of controllers are generated in the database. Table 3.4 shows that the mean and standard deviation query times, controller addition to the database time, as well as total control-loop update time for cached matrices logic, is substantially better than regular EPC for more than $\mathbf{100}$ controllers (**RS1**). It is worth noting that the cached matrices logic reduces the QP control callback frequency, as shown in Fig. 3.7c by generating a feasible controller database faster compared to non-cached matrices logic as detailed in Fig. 3.7d (**RS6**). Since the control loop timing is improved, the consequent improvement in tracking the trajectories is visible in Fig. 3.7a, which shows reduced tracking error for cached matrices logic compared to regular EPC (**RS7**). The total control loop timing is shown in Fig. 3.7b.

For completeness, another trial was conducted by executing a circular trajectory with tight linear velocity constraints of $\mathbf{1.65}$ m/s in $X - Y$ directions. As expected, cached matrices based EPC gave a substantial improvement in timing efficiency, as shown in Table 3.5. It is worth noting that cached matrices based EPC satisfied the velocity constraints with much fewer violations compared to the non-cached EPC, and almost equivalent to QP based controller strategy as shown in Fig. 3.8a and Table 3.3 below. Table 3.5 also shows that the query times for cached matrices based EPC is less than QP solve times. Moreover, a similar trial with only linear MPC based controller (QP controller) also shows that the total time for linear MPC based controller is $\left(\mathbf{1.3764 \pm 0.5907}\right)$ ms which is more than the cached matrices based EPC total control loop

	Cached (m/s)	**Non-Cached** (m/s)	**QP** (m/s)
X	1.6542	**1.6997**	1.6533
Y	1.6528	**1.6840**	1.6539

Table 3.3: Maximum velocity achieved while enforcing the constraints.

	PE add (ms)		**PE Query (ms)**		**QP (ms)**		**Total time (ms)**	
	Cached	**Non-Cached**	**Cached**	**Non-Cached**	**Cached**	**Non-Cached**	**Cached**	**Non-Cached**
Iterations	182	166	36295	32748	182	166	36497	36497
Mean	**1.1254**	28.8307	**0.5065**	1.0129	**0.5984**	0.6359	**1.3482**	2.2009
STD	**2.5327**	32.7990	**1.3292**	6.3277	**0.2253**	0.2528	**1.4079**	8.3199

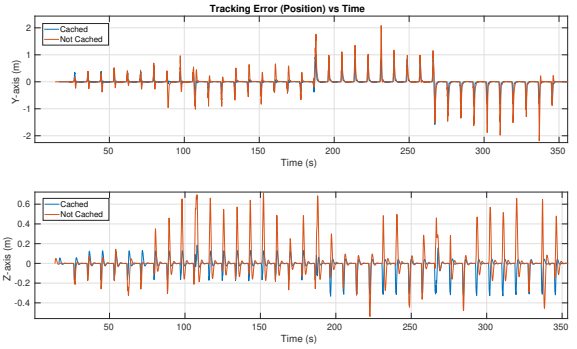
Table 3.4: Timing characteristics for EPC with cached speedup while control input constraints are enforced.

	PE add (ms)		**PE Query (ms)**		**QP (ms)**		**Total time (ms)**	
	Cached	**Non-Cached**	**Cached**	**Non-Cached**	**Cached**	**Non-Cached**	**Cached**	**Non-Cached**
Iterations	114	79	7413	6794	114	79	7528	6920
Mean	**0.1289**	21.0327	**0.1790**	1.1906	**0.8562**	0.9342	**1.0577**	2.5531
STD	**0.6897**	17.4881	**0.4334**	4.4867	**0.4045**	0.4687	**0.7369**	6.3469

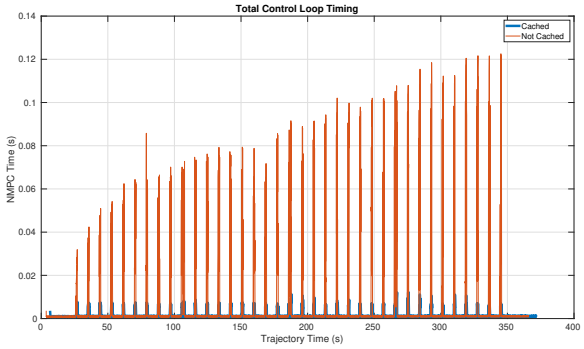
Table 3.5: Timing characteristics for EPC with cached speedup while state constraints (linear velocity) are enforced.

timing of $\left(\mathbf{1.0577 \pm 0.7369}\right)$ ms, as shown in Table 3.5.

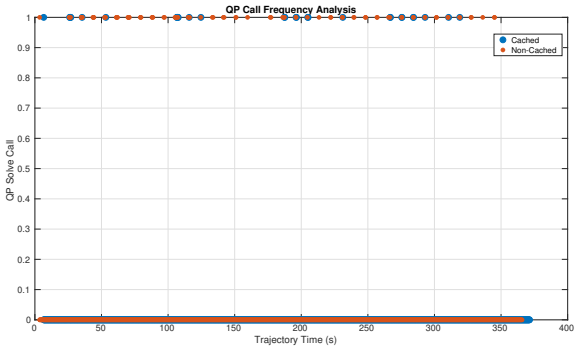
Future avenues for increasing the querying efficiency include computing the cached matrices once per control iteration and reusing them for all the queries in that iteration. Currently, each row corresponding to the active set is selected to compute the above-mentioned cached matrices. We propose a selector matrix that can be introduced as a variable which can restrict the matrix calculation for increasing the querying efficiency. Apart from that, leveraging the transition matrices online during regular EPC by bootstrapping the already learned transitions can also improve the querying efficiency.



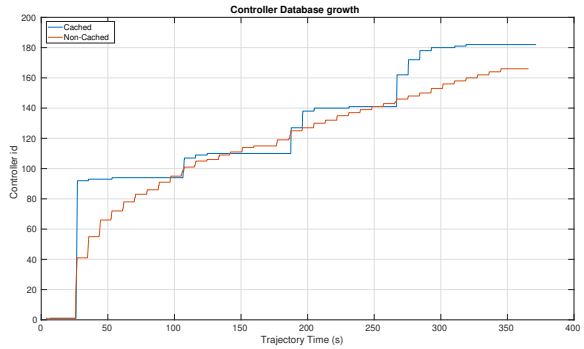
(a) Position Tracking Error vs. Time



(b) Total control loop time



(c) QP solver call frequency analysis



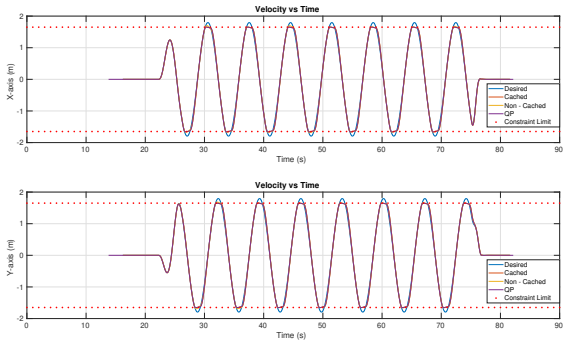
(d) Controller database growth comparison

Figure 3.7: Cached matrices logic shows substantial improvement in total control loop timing resulting in improved trajectory tracking performance for a controller database of more than 100 controllers as shown in Fig. 3.7a, 3.7b. Figure 3.7c and 3.7d shows that the non-cached matrices based EPC calls QP control with a higher frequency compared to cached matrices based EPC and controller database growth comparison between the two techniques, respectively.

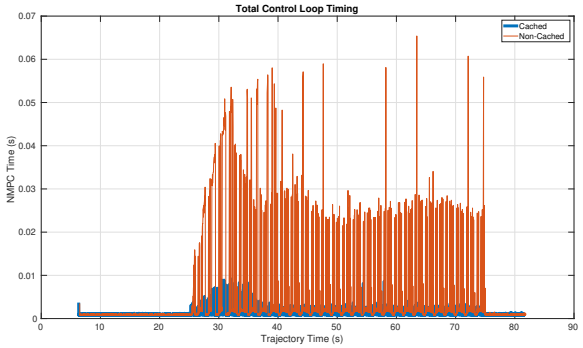
3.3.4 Robust Constrained Control

This section discusses the results of robust constraint tightening for enforcing a safe behavior of the MAV conservatively. All the features discussed above for cascaded and full state EPC controller applies on Robust EPC formulation as well. For proving the viability of running this predictive controller strategy real-time, robust EPC on real hardware systems, as well as on the Hardware-in-Loop (HIL) systems is tested. In the HIL system, the simulation environment onboard the computationally constrained platform is setup to mimic the real flight behavior.

A back and forth trial is conducted on a computationally constrained real system with time-varying state

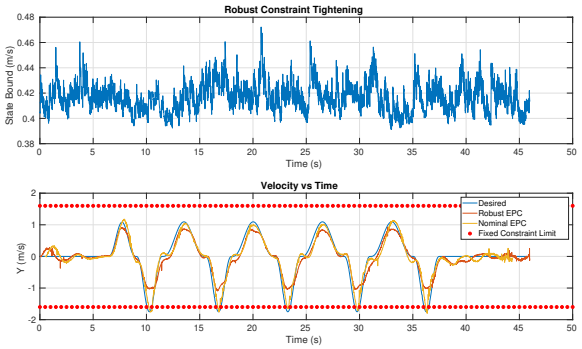


(a) Position Tracking Error vs. Time

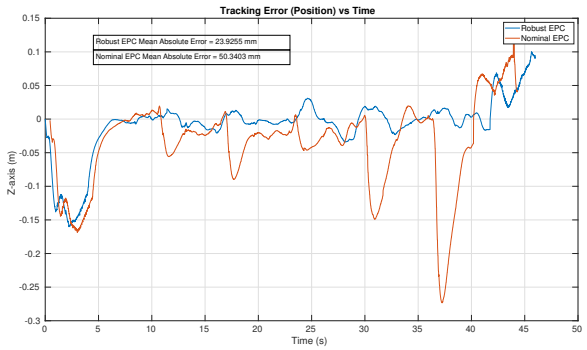


(b) Total control loop time

Figure 3.8: Cached matrices logic shows substantial improvement in total control loop timing resulting in improved trajectory tracking performance for a controller database of more than 100 controllers.



(a) Robustly constrained velocity with varying state constraint bounds

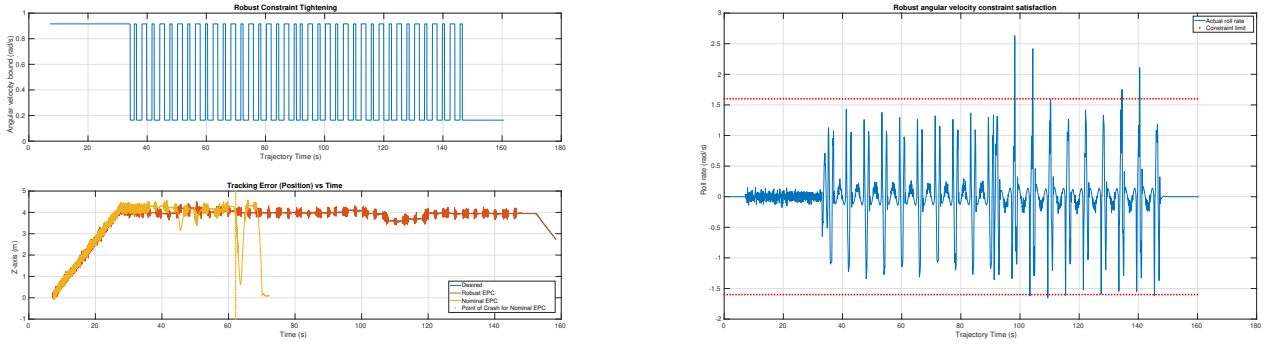


(b) Position tracking in Z direction

Figure 3.9: Time-varying state constraint bounds onboard a real system changes, that introduce robustness in enforcing the constraints as shown in Fig. 3.9a. Due to the enforced robustness in the presence of noisy odometry data, robust formulation improves the tracking performance in Z direction, as shown in Fig. 3.9b. Real-flight videos showing nominally constrained EPC and robustly constrained EPC are embedded here.

uncertainty. Figure 3.9a shows that robust EPC always satisfies the enforced constraint of $1.6\ m/s$ in linear velocity even in the presence of time-varying state uncertainty (**RH5**), resulting in improved tracking behavior in Z direction as shown in Fig. 3.9b (**RH4**).

Angular velocity constraints are crucial for enforcing a reduction in sudden jerks in the vehicle's orientation, thereby enforcing safety. HIL trials with the similar aggressive back and forth trajectory are executed on NVIDIA TX2 by injecting distance varying state uncertainty, which results in constraint tightening as



(a) (b)

Figure 3.10: Figure 3.10a shows a distance-varying angular velocity constraint bound that tightens constraints when the MAV enters a specific region in space. Nominal EPC with no robust constraint tightening fails to track the trajectory in the presence of uncertain state estimates. Figure 3.10b shows the roll rate constraints are satisfied for the commanded aggressive trajectory except for 4 violations, resulting from a sudden large step input.

shown in Fig. 3.10a. Due to the noisy odometry data, nominal EPC with non-robust bounds fails to track the trajectory while robust formulation enforces safety by tightening the roll rate bounds (**RH5**). Improvement in tracking the trajectory is clearly visible in Z direction from Fig. 3.10a (**RH4**). Even though robust formulation tightens the constraints, 4 constraint violations are still visible in Fig. 3.10b due to the chance-constrained nature of the presented robust formulation that enforces constraints with $\alpha = 99.997\%$ probability. Moreover, the query time for robust angular velocity constrained HIL experiment is $\mathbf{0.5426 \pm 2.4979}$, which is well within the stable control-loop update rate for attitude control (**RH1**).

3.3.5 Explicit Robust Constrained Control

Finally, to improve the querying efficiency of Robust EPC, Markov Chain Order (MCO) simplification of the database is applied to the generated controller database. HIL experiment with constraints enforced on desired acceleration is executed to provide real-time viability of the presented control strategies on computationally constrained MAVs. The mean query time for this explicit Robust EPC trial on NVIDIA TX2 is $\mathbf{0.4978 \pm 3.6846}$ while the query time for regular Robust EPC was $\mathbf{1.1808 \pm 7.6683}$, proving that explicit MCO-Robust EPC approach improves query efficiency of the controller database (**RH1**).

4 Model Learning

Model learning strategies can compensate low-frequency noise acting on the MAV such us unmodeled exogenous disturbances. This chapter details the evaluation of model learning strategy with cascaded and full state EPC controller in simulation as well as hardware systems. A strategy to generate wind-field disturbance in the simulation is discussed in this chapter followed by online model adaptation experiments using cascaded EPC controller strategy.

Restating the model learning formulations given in [23], predictive control techniques for nonlinear systems use a nonlinear dynamics model that involves the complexity of solving nonlinear programs, or a computationally efficient local approximation of the nonlinear dynamics. Thus, given the nonlinear dynamics $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$, nominal state $\mathbf{x}^\star$, and nominal control $\mathbf{u}^\star$, we define $\bar{\mathbf{x}} = \mathbf{x} - \mathbf{x}^\star$ and $\bar{\mathbf{u}} = \mathbf{u} - \mathbf{u}^\star$ and correspondingly derive an affine approximation of the dynamics via a first-order Taylor series expansion, $\bar{\mathbf{x}}^{\text{nom}}_{\mathbf{k+1}} = \mathbf{A}\bar{\mathbf{x}}_k + \mathbf{B}\bar{\mathbf{u}}_k + \mathbf{c}$. This model can then be extended with an online-learned component which estimates nonlinear perturbations, modeling errors, and unmodeled exogenous forces acting on the MAV. In this chapter, two online model learning techniques, LWPR and ISSGPR, along with one reactive adaptation strategy, Luenberger disturbance observer is discussed for compensating unmodeled disturbances acting on the MAV. Online model learning techniques model a nonlinear function (from an input $\mathbf{z}$ to an output $\mathbf{p}$) using a Gaussian-weighted combination of simple basis functions (linear and sinusoidal in LWPR and ISSGPR, respectively). Due to their recurrent and incremental online model update capability with the oncoming new data, these model learning strategies are able to retain information on past experiences while adapting their estimates to changing dynamics.

As discussed, this chapter details the implications of leveraging online model learning techniques while flying through a simulated wind-field environment. Unmodeled exogenous disturbances are applied on the vehicle from a random simulated wind field described in Sec. 4.3. Consequently, simulation studies are presented showing the improvement of tracking performance by leveraging online model learning techniques over reactive model adaptation strategies in Sec. 4.3.1.

To close the loop with (2.25), $\mathbf{z}$ is defined as $\mathbf{z} = \begin{bmatrix}\mathbf{x}_k^\top & \mathbf{u}_k^\top\end{bmatrix}^\top$ and $\mathbf{p} = \bar{\mathbf{x}}_{k+1} - \bar{\mathbf{x}}^{\text{nom}}_{k+1}$. The corresponding prediction output $\hat{\mathbf{p}} = \begin{bmatrix}p_0, p_1, \dots\end{bmatrix}^\top$ gives the estimated perturbation at a query point $\mathbf{z}$ which applies to predictive dynamics model given by (2.25) from Chapter 2.

4.1 Online Model Adaptation

4.1.1 Model Learning using LWPR

Partial least squares is used by LWPR for estimating the basis functions, which projects the inputs onto a lower dimensional space defined by projection direction vectors $\boldsymbol{\nu}_r$ and $\boldsymbol{\rho}_r$, as detailed in [13]. It also computes slope coefficients β_r for each projection direction and an offset β_0 to generate a prediction of a scalar output. The dynamics model of the MAV is fit element-wise [27] – for the i^{th} element in $\mathbf{p}$, local linear model j (with r_j projection directions) is given by

$$\begin{aligned} \Psi &= \beta_0 + \begin{bmatrix}\beta_1, \cdots, \beta_{r_j}\end{bmatrix} \begin{bmatrix} \boldsymbol{\nu}_1^\top \\ \boldsymbol{\nu}_2^\top \mathbf{P}_1 \\ \vdots \\ \boldsymbol{\nu}_{r_j}^\top (\mathbf{P}_1 \cdots \mathbf{P}_{r_j-1}) \end{bmatrix} (\mathbf{z} - \mathbf{m}_j) \\ &= \alpha_j + \boldsymbol{\beta}_j^\top (\mathbf{z} - \mathbf{m}_j) \end{aligned} \tag{4.1}$$

where $\mathbf{P}_r = \mathbf{I} - \text{diag}(\boldsymbol{\rho}_r)\begin{bmatrix}\boldsymbol{\nu}_r, \cdots, \boldsymbol{\nu}_r\end{bmatrix}^\top$. 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

$$\begin{aligned} p_i(\mathbf{z}) &= \frac{1}{W} \sum_{j=1}^{N_i} w_j(\mathbf{z})\, \Psi_j(\mathbf{z}) \\ w_j(\mathbf{z}) &= \exp\Big(-\frac{1}{2} (\mathbf{z} - \mathbf{m}_j)^\top \mathbf{D}_j (\mathbf{z} - \mathbf{m}_j) \Big) \\ W &= \sum_{j=1}^{N_i} w_j(\mathbf{z}) \end{aligned} \tag{4.2}$$

4.1.2 Model Learning using ISSGPR

The radial basis function kernel in ISSGPR is approximated by a vector of D sinusoidal features with random frequencies

$$\boldsymbol{\phi}(\mathbf{z}) = \frac{\sigma_f}{\sqrt{D}} [\cos(\boldsymbol{\omega}_1^\top \mathbf{z}), \sin(\boldsymbol{\omega}_1^\top \mathbf{z}), \cdots, \cos(\boldsymbol{\omega}_D^\top \mathbf{z}), \sin(\boldsymbol{\omega}_D^\top \mathbf{z})] \tag{4.3}$$

where the parameter σ_f corresponds to the signal variance, the frequencies $\boldsymbol{\omega} \sim \mathcal{N}(0, \mathbf{M})$, and $\mathbf{M}$ is a diagonal matrix of characteristic length scales. The length scales reflect the size and importance of each input dimension [14]. Detailed formulation on online model learners is given in [23].

4.1.3 Online model adaptation using Luenberger observer

$\mathcal{L}_1$ adaptive control based disturbance observer is also considered in this chapter, which is a reactive adaptation strategy unlike the model learning strategies described above. This approach employs a nonlinear Luenberger disturbance observer which is driven by the difference between the state predicted via a nonlinear dynamics model and the state reported by the state estimator.

4.2 Practical considerations

4.2.1 Bandwidth considerations

For the online model adaptation approaches described in Sec. 4.1, the insight from L1 adaptive control is followed, and a low-pass filter to the output of the observer is applied before it is passed to the controller. The bandwidth parameter of this filter is tuned to the system's response time to avoid destabilizing the system with rapid model perturbations.

4.3 Generating wind disturbance in simulation

In order to test the model learners presented in this chapter, simulated wind-field is generated which adds more variability to the drag disturbance.

For generating the wind-velocity, coordinate location of fans in the space is noted as

$$\begin{aligned} s_x &= \mathbf{d}_b \cos(\hbar - \pi) \\ s_y &= \mathbf{d}_b \sin(\hbar - \pi) \end{aligned} \tag{4.4}$$

where $\mathbf{d}_b$ and $\hbar$ correspond to the perpendicular distance (in the xy-plane) from world origin to the line of wind-generating fans, and wind heading in the world frame respectively.

Direction vector of the row of fans in world frame is

$$\begin{aligned} \mathbf{d}_{b\_\mathbf{x}} &= \cos(\hbar - \pi/2) \\ \mathbf{d}_{b\_\mathbf{y}} &= \sin(\hbar - \pi/2) \end{aligned} \tag{4.5}$$

Defining the linear portion of the wind velocity magnitude as a function of distance in front of the fans

$$\begin{aligned} k &= \frac{(\mathbf{v}_{min} - \mathbf{v}_{max})}{(\mathbf{d}_f - \mathbf{d}_c)} \\ b &= \mathbf{v}_{max} - (\mathbf{d}_c k) \end{aligned} \tag{4.6}$$

where $\mathbf{v}_{min}, \mathbf{v}_{max}$ correspond to minimum wind velocity, and maximum wind velocity, respectively. $\mathbf{d}_f$ is any location in space in front of the fans which has a constant wind velocity of $\mathbf{v}_{min}$ and $\mathbf{d}_c$ is any location in space in front of the fans, or at any location behind the row of fans which has a constant wind velocity.

Radial vector in the world frame is

$$\begin{aligned} \mathbf{V}_{X_\perp} &= (\mathbf{x} - s_x) - ((\mathbf{x} - s_x)(\mathbf{d}_{b\_\mathbf{x}}) + (\mathbf{y} - s_y)(\mathbf{d}_{b\_\mathbf{y}}))\mathbf{d}_{b\_\mathbf{x}} \\ \mathbf{V}_{Y_\perp} &= (\mathbf{y} - s_y) - ((\mathbf{x} - s_x)(\mathbf{d}_{b\_\mathbf{x}}) + (\mathbf{y} - s_y)(\mathbf{d}_{b\_\mathbf{y}}))\mathbf{d}_{b\_\mathbf{y}} \end{aligned} \tag{4.7}$$

where $\mathbf{x}$ and $\mathbf{y}$ correspond to world frame position in x and y respectively. $s_x, s_y, \mathbf{d}_{b\_\mathbf{x}}, \mathbf{d}_{b\_\mathbf{y}}$ are calculated from (4.4), and (4.5) respectively.

Using (4.7), the norm of radial vectors can be calculated as

$$\begin{aligned} \mathbf{V}_{\text{norm}} &= \sqrt{(\mathbf{V}_{X_\perp})^2 + (\mathbf{V}_{Y_\perp})^2} \\ \zeta &= tan^{-1}\frac{\mathbf{V}_{Y_\perp}}{\mathbf{V}_{X_\perp}} \end{aligned} \tag{4.8}$$

where ζ is termed as drag coefficient.

If heading of the radial vector matches that of wind heading, then the query point is in front of the fans; otherwise, it is behind the fans

$$
\begin{aligned}
&\text{if}(\zeta - \hbar) > 0.01 \\
&\mathbf{V}_{X_\perp} = -\mathbf{V}_{X_\perp} \\
&\mathbf{V}_{Y_\perp} = -\mathbf{V}_{Y_\perp} \\
&\mathbf{V}_{\text{norm}} = -\mathbf{V}_{\text{norm}}
\end{aligned}
\tag{4.9}
$$

Final magnitude of velocity based on norm of the velocity and distance from source ((4.6), (4.8)) is

$$
\mathbf{V}_{\text{mag}} = \begin{cases} \mathbf{v}_{max}, & \text{if} \quad \mathbf{V}_{\text{norm}} < \mathbf{d}_c \\ k\mathbf{V}_{\text{norm}} + b, & \text{if} \quad \mathbf{V}_{\text{norm}} < \mathbf{d}_f \\ \mathbf{v}_{min}, & \text{otherwise} \end{cases}
\tag{4.10}
$$

Wind velocity after multiplying norm of radial vectors and random Gaussian noise is

$$
\begin{aligned}
\mathbf{V}_{\text{wind}} &= \begin{bmatrix} (\mathbf{V}_{\text{mag}}\mathbf{V}_{X_\perp})/\mathbf{V}_{\text{norm}} \\ (\mathbf{V}_{\text{mag}}\mathbf{V}_{Y_\perp})/\mathbf{V}_{\text{norm}} \\ 0 \end{bmatrix} \\
\mathbf{V}_{\text{wind}} \quad &+= \quad \mathbf{v}_\sigma \times \text{random}(3,1)
\end{aligned}
\tag{4.11}
$$

where $\mathbf{v}_\sigma$ is the standard deviation of isotropic Gaussian noise added to wind velocity.

Final wind drag acceleration disturbance that is added to the simulator is

$$
\begin{aligned}
\mathbf{v}_{\text{rel}_\text{B}} &= (\mathrm{R}_\text{B}^\text{W})^\top(\mathbf{v}_\text{w} - \mathbf{V}_{\text{wind}}) \\
\mathbf{a}_{\text{wind}} &= -\mathrm{R}_\text{B}^\text{W}\left(\frac{\zeta}{m}\right)\mathbf{v}_{\text{rel}_\text{B}} \\
\mathbf{a}_\text{W} &+= \mathbf{a}_{\text{wind}}
\end{aligned}
\tag{4.12}
$$

where $\mathbf{v}_{\text{rel}_\text{B}}, \mathrm{R}_\text{B}^\text{W}, \mathbf{v}_\text{w}, \mathbf{a}_{\text{wind}}, m$, and $\mathbf{a}_\text{W}$ correspond to relative velocity in body-frame (of MAV), rotation matrix for body-frame to world-frame transformation, linear velocity in world-frame propagated in the simulator, linear acceleration generated in the world frame, mass of the MAV, and linear acceleration in world frame propagated in the simulator respectively. Visualization of wind field disturbance for the results presented in

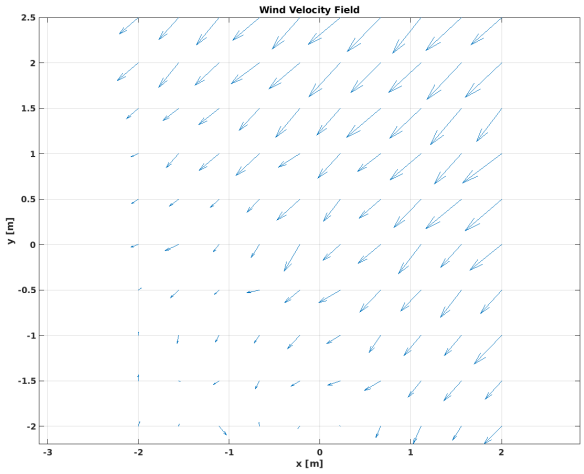


Figure 4.1: Wind field disturbance visualization in Matlab.

Sec. 4.3.1 is shown in Fig. 4.1.

4.3.1 Online model adaptation in windy environment

This experiment is designed to test the online model adaptation capabilities using model learning strategies, followed by reactive disturbance adaptation strategies in the simulated wind-field environment. Cascaded EPC controller with LWPR model learner and Luenberger disturbance observer is leveraged to fly aggressive circle trajectories in an environment with wind flowing diagonally across the MAV at $6\ m/s$. Figure 4.2 and Table 4.1 shows the reduced tracking error for LWPR in all the three directions compared to the Luenberger observer.

Model learning strategies can be coupled with the predictive control strategies with cascaded as well as full state multirotor dynamics (FSE controller) described in previous chapters, for increasingly aggressive flights in uncertain environments.

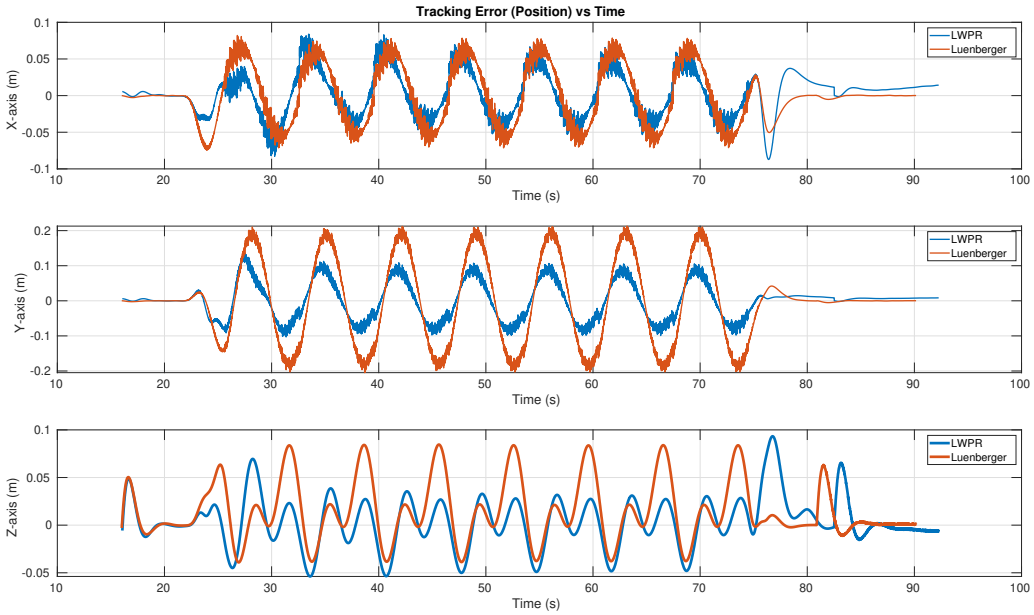


Figure 4.2: Tracking error in position for a circle trajectory flown in an environment with simulated wind-field disturbance. This figure shows that model learning disturbance compensation strategy outperforms reactive disturbance compensation strategy, as evident by the reduced tracking error.

	X (mm)	Y (mm)	Z (mm)
LWPR	**24.3059**	**39.2606**	**19.6424**
Luenberger	30.1953	84.5195	23.8440

Table 4.1: Mean absolute tracking error in position while executing a circle trajectory with simulated wind-field disturbance.

5 Software Organization and Code Interface

In this section, we describe the software system developed for the multirotor control architecture presented in this report. A high-level overview of the interaction between various components in the control architecture is provided in Sec. 5.1, in a package named **`nonlinear_mpc`**. We then provide UML activity diagrams for the components of **`nonlinear_mpc (NMPC)`** components in Sec. 5.2, followed by a description of each class in the **`NMPC`** package in Sec. 5.3. The following codebase supports doxygen; therefore, finer details can be accessed via the compiled doxygen documentation.

5.1 High-level Control Architecture

The control architecture presented in this section serves to provide a generalized interface for the multirotor systems and ensures safe operation as the system transitions between operating modes such as takeoff, landing, providing polynomial inputs for trajectory tracking, and providing step inputs. High-level components such as planners interact with the control architecture by sending trajectories to be tracked by the specified controller, and by sending events to trigger changes in the operating modes. A diagram of system components in a typical configuration is shown in Fig. 5.1.

Each dynamics model in the **`nonlinear_mpc (NMPC)`** package is written as a class and hence the codebase is designed for general usage with any dynamics model. A base class supporting outer, inner, and full state dynamics model is written as **`DynLinQuadOuter, DynLinQuadInner,`** and **`DynNLQuad3D`** respectively. This structure allows for flexible extensions of the proposed framework at different levels, e.g., to extend and switch between different multirotor dynamics, different controller usage, between experience generation and experience re-use for EPC controllers, and so on.

5.2 System Interaction

The control architecture receives current state information from one of the state estimator sources which is passed to a standalone class of motion manager, that consists of the interaction between finite state machines, trajectory managers, controllers, and disturbance observers. Although not explicitly stated, the wrapper for disturbance observer lies alongside the wrapper of controllers in Fig. 5.1.

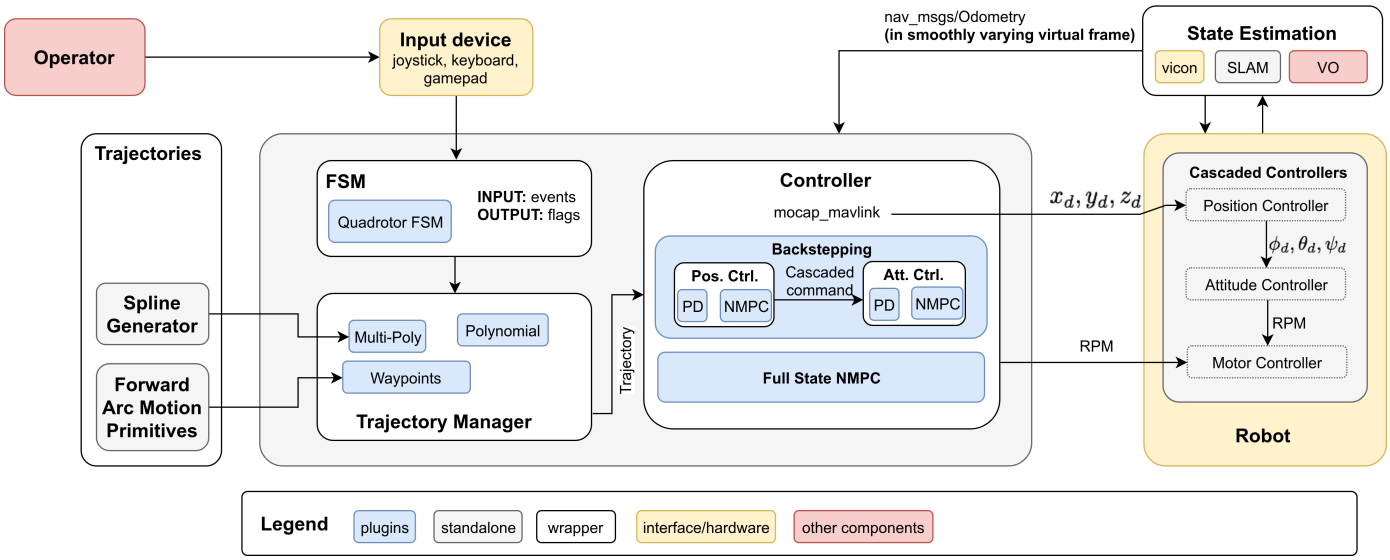


Figure 5.1: High-level summary of the major components of the control architecture and their interactions.

5.3 Class Description

Software for nonlinear model predictive control codebase (NMPC) is detailed in this section. The software organization for **`nonlinear_mpc (NMPC)`** is structured in a class hierarchy to make simplifed setup for changing and prototyping different controller strategies in the system. Codebase (**`NMPC`**) is structured to make modifications and future prototyping easy and hence modularity of different subsystems within **`NMPC`** package is kept into consideration while designing the software. A concise class diagram for the **`NMPC`** codebase is described in Fig. 5.2. Class hierarchy and the corresponding usage of classes are shown as a UML acitivity diagram in Fig. 5.2, which comprises the following

- **`NMPC`**: Top level templated package class supporting the nonlinear model predictive control based controller configurations.
- Following controller configurations are supported in our software architecture:

  `EPC`: Class handling EPC, EPC with Markov Chain simplification (EPC-MCO), and Robust EPC components.

  `QP-based linear MPC`: QP controller solving linear MPC that is called separately from **`NMPC`**.

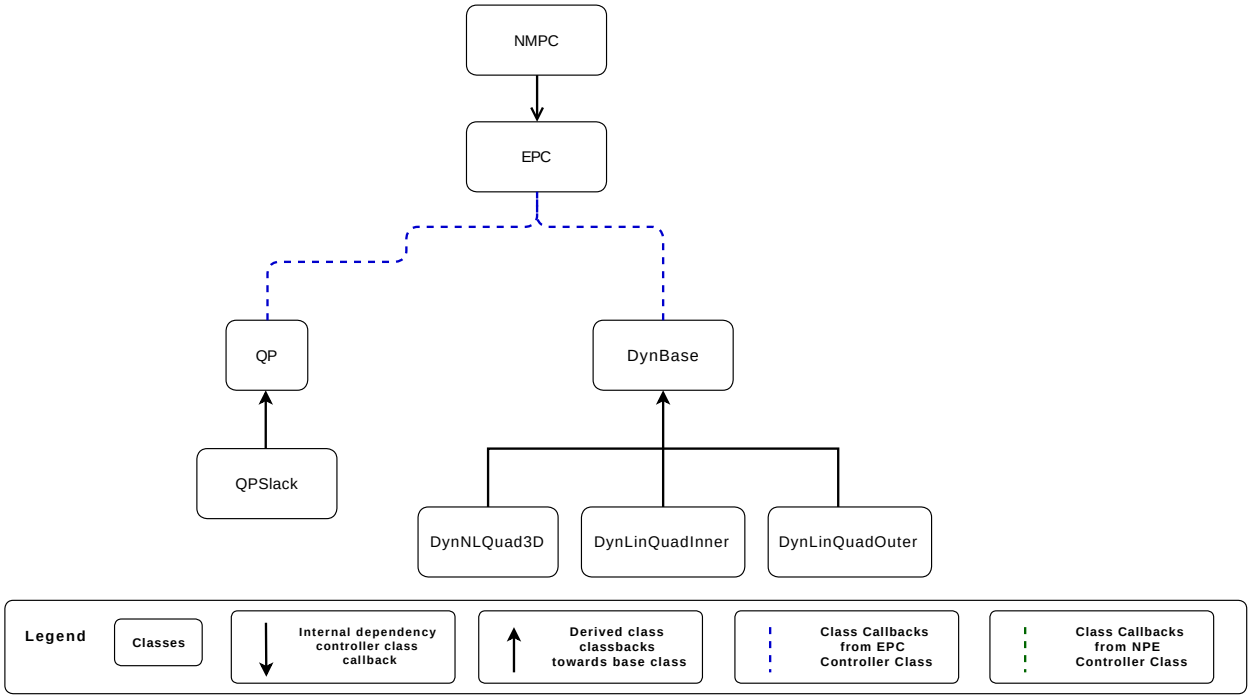


Figure 5.2: High-level summary of the major components of the **`nonlinear_mpc (NMPC)`** package and their interactions shown as a UML activity diagram. For further detail on each of the classes, please refer the **`doxygen`** documentation.

- **`DynBase`**: Base class for handling system dynamics of the MAV.

  `DynLinQuadOuter`: Dynamics for multirotor position control.

  `DynLinQuadInner`: Dynamics for multirotor attitude control.

  `DynNLQuad3D`: Dynamics for multirotor full state control.

- **`QP:`** Class responsible for linearizing dynamics, formulating and solving a quadratic program using third-party libraries such as `qpoases`. Controllers (`EPC, QP-based linear MPC`) use this class for the aforementioned tasks of dynamics linearization and quadratic program setup.

`QPSlack`**:** Inherited from **`QP`** base class. Used to calculate intermediate/safety controllers by adding slack formulation and the corresponding slack variables.

Bibliography

[1] A. Spitzer, X. Yang, J. Yao, A. Dhawale, K. Goel, M. Dabhi, M. Collins, C. Boirum, and N. Michael, "Fast and agile vision-based flight with teleoperation and collision avoidance on a multirotor," in *Proc. of the Intl. Sym. on Exp. Robot.* Buenos Aires, Argentina: Springer, 2018, to be published.

[2] V. R. Desaraju, "Safe, efficient, and robust predictive control of constrained nonlinear systems," 2017.

[3] V. R. Desaraju and N. Michael, "Efficient prioritization in explicit adaptive nmpc through reachable-space search," in *2018 AIAA Guidance, Navigation, and Control Conference*, 2018, p. 1847.

[4] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2520–2525.

[5] M. Müller, S. Lupashin, and R. D'Andrea, "Quadrocopter ball juggling," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 5113–5120.

[6] S. Tang and V. Kumar, "Mixed integer quadratic program trajectory generation for a quadrotor with a cable-suspended payload," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 2216–2222.

[7] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 664–674, 2012.

[8] T. Lee, M. Leoky, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se (3)," in *Decision and Control (CDC), 2010 49th IEEE Conference on*. IEEE, 2010, pp. 5420–5425.

[9] J. Löfberg, *Minimax approaches to robust model predictive control*. Linköping University Electronic Press, 2003, vol. 812.

[10] S. J. Qin and T. A. Badgwell, "An overview of nonlinear model predictive control applications," in *Nonlinear model predictive control*. Springer, 2000, pp. 369–392.

[11] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.

[12] A. Grancharova and T. A. Johansen, *Explicit nonlinear model predictive control: Theory and applications*. Springer Science & Business Media, 2012, vol. 429.

[13] S. Vijayakumar, A. D'souza, and S. Schaal, "Incremental online learning in high dimensions," *Neural computation*, vol. 17, no. 12, pp. 2602–2634, 2005.

[14] A. Gijsberts and G. Metta, "Real-time model learning using incremental sparse spectrum gaussian process regression," *Neural Networks*, vol. 41, pp. 59–69, 2013.

[15] P. Bouffard, A. Aswani, and C. Tomlin, "Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 279–284.

[16] H. Fukushima, T.-H. 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.

[17] Y. K. Ho, H. K. Yeoh, and F. S. Mjalli, "Generalized predictive control algorithm with real-time simultaneous modeling and tuning," *Industrial & Engineering Chemistry Research*, vol. 53, no. 22, pp. 9411–9426, 2014.

[18] C. J. Ostafew, A. P. Schoellig, and T. 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.

[19] B. Khusainov, E. C. Kerrigan, A. Suardi, and G. A. Constantinides, "Nonlinear predictive control on a heterogeneous computing platform," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 11 877–11 882, 2017.

[20] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The grasp multiple micro-uav testbed."

[21] S. Shen, N. Michael, and V. Kumar, "Autonomous multi-floor indoor navigation with a computationally constrained mav," in *Robotics and automation (ICRA), 2011 IEEE international conference on*. IEEE, 2011, pp. 20–25.

[22] E. Kaufmann, A. Loquercio, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep drone racing: Learning agile flight in dynamic environments," *arXiv preprint arXiv:1806.08548*, 2018.

[23] V. R. Desaraju, A. E. Spitzer, C. O'Meadhra, L. Lieu, and N. Michael, "Leveraging experience for robust, adaptive nonlinear mpc on computationally constrained systems with time-varying state uncertainty," *The International Journal of Robotics Research*, p. 0278364918793717, 2018.

[24] M. Dabhi, V. R. Desaraju, and N. Michael, "Evaluation of explicit experience-driven predictive control on a computationally constrained platform."

[25] V. R. Desaraju and N. 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.

[26] M. Quigley, J. Faust, T. Foote, and J. Leibs, "Ros: an open-source robot operating system."

[27] D. Mitrovic, S. Klanke, and S. Vijayakumar, "Adaptive optimal feedback control with learned internal dynamics models," in *From Motor Learning to Interaction Learning in Robots*. Springer, 2010, pp. 65–84.