FLIGHT SIMULATION AND HARDWARE IMPLEMENTATION OF DEEP MODEL PREDICTIVE CONTROL

BY

GARRETT GOWAN

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Aerospace Engineering
in the Graduate College of the
University of Illinois Urbana-Champaign, 2021

Urbana, Illinois

Adviser:

Professor Girish Chowdhary

Abstract

This thesis presents the flight simulation and hardware implementation of Deep Model Predictive Control (DMPC) on an experimental setup, which consists of a quadcopter and motion capture system. DMPC aims to adapt abrupt state-dependent matched uncertainties arising due to faults, collects training data for a deep neural network (DNN) to learn slowly varying features, and ensures safety during the learning phase. Training of DNN to learn features is carried out on a parallel machine, while the actual system is controlled by a tube MPC and an adaptive mechanism with fixed features. Under certain verifiable technical conditions, DMPC ensures the asymptotic stability of closed-loop states. Through simulations presented in this thesis, it is shown that DMPC can outperform other control architectures when utilizing an affine model with additive nonlinear disturbance to the control input, and is able to guarantee stability while avoiding unwanted behavior in early learning phases while having long-term learning capabilities. This study demonstrates that DMPC is a powerful and safe control architecture for nonlinear systems.

To Ashley and my family, for their love, support, and sacrifices

Acknowledgments

First and foremost, I would like to thank my advisor, Dr. Girish Chowdhary for his enthusiasm, advice, encouragement, and support. When I first came to UIUC I did not have an advisor, and you decided to give me the opportunity in your group. I am glad to have you as a mentor and for all the effort you put into advising me.

To my colleagues, especially Dr. Prabhat Mishra and Mateus Valverde, for their advice, critiques and encouragements during our time working together. To John Hart, for getting me access to the Intelligent Robotics Lab (IRL), his advice and support during experimentation.

I would like to thank my fiancé, Ashley, for all the love and support she has given me. For being someone to rely on and a rock when things were tough throughout this process. For always being kind, loving, and a source of inspiration. I look forward to my future with you and what it holds for us.

To my parents, Thomas and Donna, and my siblings, Conor, Bryant, and Kaleigh. I will always be grateful for your love, support, and sacrifices you have made over the years. For pushing me to always work my hardest, and chase after my goals. For always being there to talk, relax with, and enjoy each other's company. Without all of you, I would not be where I am today.

To Christa and Jason, thank you for all the support and kindness you have shown me since I moved to Champaign. When I first moved here, I did not know I had family here, but I am glad to have gotten to know you both and share the time that we had together.

To the friends I have made over the years, especially Jon, Mark, Saad, Andrew, and Nil, thank you for your support, kindness, and being there to spend time with.

There are many more people that I have not mentioned that have had an impact on my life in Illinois and beyond, and I apologize for not explicitly mentioning you. But I appreciate the time that you have shared with me and how you have made me the person I am today.

Table of contents

LIST OF ABBREVIATIONS

MPC Model Predictive Control
NN Neural Network
DNN Deep Neural Network
DMPC Deep Model Predictive Control
MRAC Model Reference Adaptive Control
DMRAC Deep Model Reference Adaptive Control
PID Proportional Integral Derivative
NED North East Down
ABC Aircraft Body Center
ISS Input to State Stability
CICS Converging Input Converging State
IMU Internal Measurement Unit
MSE Mean Square Error
COM Center of Momentum

Chapter 1

INTRODUCTION

In recent years, quadcopters have become increasingly popular in civilian and defense applications[1]–[5], and for academic research [6]–[8]. This can be attributed toward the quadcopters maneuverability and the simplicity in its structure. With this increase of usage, and the diversity of its applications, control algorithms that reject disturbances from its environment while giving stability guarantees have become a subject of interest to the research community.

Developing these types of controls while guaranteeing stability and avoid unwanted behaviors is a challenge. These nonlinear disturbances can have significant effects on the dynamics of the robot by changing the operating conditions it was built for, degrading the overall performance, or potentially damaging the vehicle, leading to failure. Under these types of conditions, traditional control architectures, including model-based control architectures, can fail when dynamic changes become too extreme.

Traditionally, Adaptive controllers that can learn online and in real-time to adapt to disturbances have been utilized by quadcopters [9]–[11]. An alternative that has gained popularity is Model Reference Adaptive Control (MRAC) using shallow networks to learn [12]–[14]. One such method, Gaussian Process Model Reference Adaptive Control [13], [15], has had success in adapting to disturbances. While these control architectures can adapt to mitigate instantaneous disturbances, they lack the ability to generalize to similar operating conditions. Recent success in this endeavor has been found utilizing Deep Model Reference Adaptive Control (DMRAC). Quadrotor experiments were conducted to test DMRAC [6], and this control architecture demonstrated long term learning and generalizability, which allowed for improved performance over other PID controllers and Model Reference Adaptive Controllers (MRAC). While DMRAC can learn these unstructured uncertainties in real-time, the system exemplifies unwanted behaviors during the early learning phases and there is no provision to optimally impose hard constraints on control actions. These learning-based

control architectures implementations are prohibited on safety critical systems due to this behavior.

One approach to this problem is to utilize Model Predictive Control (MPC) to ensure safety due to its constraint satisfaction capability. Robust Adaptive MPC [16], [17] has proven to be able to adapt to nonlinear disturbances while ensuring safety, however like other adaptive control systems it lacks the ability to generalize disturbances. New forms of control architectures employ MPC for constraint satisfaction, and deep neural networks (DNN) for function approximation. Recent tests utilizing these types of control architectures have shown that they can handle structured uncertainties through representing the uncertainties as high dimensional basis functions and learning the basis function over time.

1.1 Motivation

A new neuro-adaptive control architecture called Deep Model Predictive Control (DMPC) [18] demonstrates the ability to adapt to nonlinear disturbances while avoiding unwanted behavior by utilizing Deep Learning with a tube-based model predictive control. This ensures the satisfaction of constraints and gives input-to-state stability of the closed-loop states. In this thesis, implementation and evaluation of the DMPC architecture for a quadcopter is simulated in comparison to other algorithms such as PID, tube-based MPC, and Adaptive tube-based MPC and the implementation for hardware is demonstrated.

1.2 Overview

To provide insight into the control architecture and simulations this thesis is organized as follows: In chapter 2, a brief overview of frame of reference, equations of motion and dynamics of quadcopter, and linearization of quadcopter dynamics are discussed. In chapter 3 the Model Predictive Control algorithms are explored and discussed. In chapter 4 the experimental setup and implementations of the control algorithms discussed in chapter 3 are presented. Chapter 5 presents and discusses the results of the control algorithms presented in chapter 3.

Chapter 2

KINEMATICS AND DYNAMICS OF THE QUADCOPTER

The work done in this thesis relies on having accurate models of the kinematics of the Parrot® Mambo Drone. The state-space models derived from the equations of motion are utilized by the MPC to determine the control inputs used at each time step and to simulate the drone with those control inputs at each time step. The Parrot® Mambo Drone is a quadcopter, this means that it is a type of helicopter that uses four upwards directed rotors that are placed in a square formation equidistant to the center of mass of the vehicle. For the control algorithms in this paper, we utilize equations of motion derived for quadcopter applications as the best representation of the Parrot® Mambo Drone used in the experiments.

The kinematics of the quadcopter are well-defined in prior works by Sabatino and Lukkonen [19], [20]. For the purposes of this thesis, we will provide a quick review of the frame of reference, equations of motion of the quadcopter, and linearization of the state-space.

2.1 Frame of Reference

To describe the equations of motion for the quadcopter, we first introduce the reference frame, in which we describe the position and orientation of the system. The two reference frames, inertial and body frame of reference are considered. We describe the inertial and body frame of references using the *North-East Down (NED)*, where the NED convention points the x-axis north, y-axis east and z axis down. For this thesis, we will denote the inertial frame of reference as O_{NED} and the body frame of reference as O_{ABC} for *Aircraft*

Body Center.

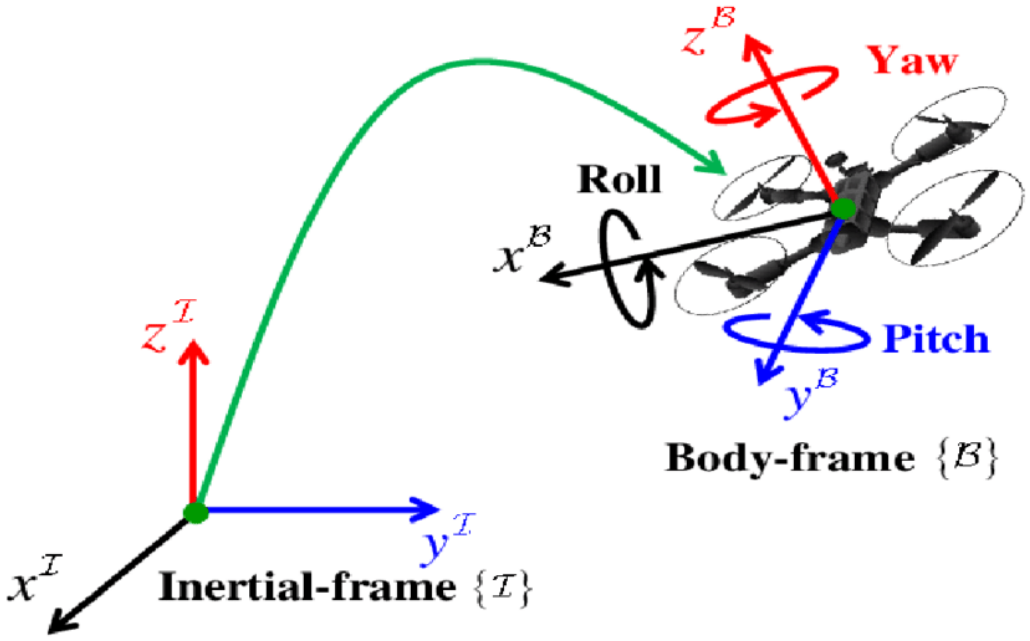


Figure 2.1: Inertial Frame and Body Frame of Reference of Quadcopter [21]

2.2 Equations of Motion

In this section, a mathematical model of the quadcopter will be provided that exploits the Newton and Euler equations. To develop the vector that describes the state-space model, we will utilize the vector $\begin{bmatrix} x & y & z & \phi & \theta & \psi \end{bmatrix}^T$ to describe the linear and angular position of the quadrotor in the earth frame, and the vector $\begin{bmatrix} u & v & w & p & q & r \end{bmatrix}^T$ to describe the linear and angular velocities in the body frame. The states of the quadcopter dynamics can be defined as the combination of these two vectors giving us a 12-dimensional vector of the following form:

$$\mathbf{x} = \begin{bmatrix} x & y & z & \phi & \theta & \psi & u & v & w & p & q & r \end{bmatrix}^T \tag{2.1}$$

where the components of equation (2.1) can be described as:

$$\begin{aligned} x,\ y,\ z &= \text{Position components along NED directions in } O_{NED} \\ \phi,\ \theta, \psi &= \text{Euler angles relating the } O_{ABC} \text{ frame to } O_{NED} \\ u,\ v,\ w &= \text{Velocity components in the NED directions in } O_{ABC} \\ p,\ q,\ r &= \text{Angular rate components relating the } O_{ABC}\text{frame to } O_{NED} \end{aligned}$$

The dynamics can be represented using Newton-Euler equations, which the MPC shown in this thesis utilize for their optimization. The derivation of the state-space model can be found in [19] the final form can be expressed as:

$$\begin{bmatrix} \dot{x}_{t+1} \\ \dot{y}_{t+1} \\ \dot{z}_{t+1} \end{bmatrix} = \begin{bmatrix} w[S\phi_t S\psi_t + C\phi_t C\psi_t S\theta_t] - v[C\phi_t S\psi_t - C\phi S\psi_t S\theta_t] + u[C\phi_t C\theta_t] \\ v[C\phi_t C\psi_t + S\phi_t S\psi_t S\theta_t] - w[C\phi_t S\psi_t - C\phi S\psi_t S\theta_t] + u[C\phi_t S\theta_t] \\ w[C\phi_t C\theta_t] - u[S\theta_t] + v[C\theta_t S\phi_t] \end{bmatrix} \tag{2.2}$$

$$\begin{bmatrix} \dot{\phi}_{t+1} \\ \dot{\theta}_{t+1} \\ \dot{\psi}_{t+1} \end{bmatrix} = \begin{bmatrix} p_t + r_t[C\phi_t T\theta_t] + q_t[S\phi_t T\theta_t] \\ q_t[C\phi_t] - r_t[S\phi_t] \\ r_t \frac{C\phi_t}{C\theta_t} + q_t \frac{S\phi_t}{C\theta_t} \end{bmatrix} \tag{2.3}$$

$$\begin{bmatrix} \dot{u}_{t+1} \\ \dot{v}_{t+1} \\ \dot{w}_{t+1} \end{bmatrix} = \begin{bmatrix} r_t v_t - q_t w_t - g[S\theta_t] \\ p_t w_t - r_t u_t - g[S\phi_t C\theta_t] \\ q_t u_t - p_t v_t - g[C\theta_t C\phi_t] \end{bmatrix} - \frac{T}{m} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.4}$$

$$\begin{bmatrix} \dot{p}_{t+1} \\ \dot{q}_{t+1} \\ \dot{r}_{t+1} \end{bmatrix} = \begin{bmatrix} \frac{I_y - I_z}{I_x} q_t r_t \\ \frac{I_z - I_z}{I_y} p_t r_t \\ \frac{I_x - I_y}{I_z} p_t q_t \end{bmatrix} + \begin{bmatrix} \frac{\tau_\phi}{I_x} \\ \frac{\tau_\theta}{I_y} \\ \frac{\tau_\psi}{I_z} \end{bmatrix} \tag{2.5}$$

In which T is the total thrust, τ_ϕ, and τ_ψ are the thrust in the direction of the corresponding body frame angles, and ω_Γ is the total angular velocity of the rotors which correspond to the following equations:

$$T = k \sum_{i=1}^{4} \omega_i \tag{2.6}$$

$$\tau_B = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} lk(-\omega_2^2 + \omega_4^2) \\ lk(-\omega_1^2 + \omega_3^2) \\ \sum_{i=1}^{4} b\omega_i^2 + I_m \dot{\omega_i} \end{bmatrix} \tag{2.7}$$

$$\omega_\gamma = \omega_1 - \omega_2 + \omega_3 - \omega_4 \tag{2.8}$$

Where k in equations (2.6) and (2.7) is the lift constant, l, b, and I_M in equation (2.7) are the distance between the rotor and the center of mass, drag constant, and inertia moment of the rotor for the quadcopter.

2.3 Linerization

As mentioned previously, the state-space model for MPC is critical for accuracy in control inputs for the system. The main drawback, however, of using accurate nonlinear state-space models is computational time and power needed to run these systems. For real-time application driven programs, computational time and power is critical for responding to disturbance and tracking trajectory for the operating vehicle. Researchers have demonstrated [22] that while nonlinear model based MPC is slightly more accurate than Linear MPC, the run-times for these algorithms are significantly higher than their linear counterparts.

In our simulation, accuracy is measured by tracking error of the trajectory in nominal conditions. Looking at the standard deviation, and R-Squared error, we can get a better idea of how well our system is performing. To illustrate the differences between utilizing a Linear and Nonlinear model for our simulation, we will simulate the quadcopter in Nominal Conditions, utilizing Nominal MPC for a simulation, and simulate the quadcopter rising to the altitude $z_{t=10} = 2\ m$ in $t = 10\ s$ and then following a fig-8 trajectory for 20 s. Under these conditions, the run time for the Linear and Nonlinear model was $t_L = 4.5938\ s$ and $t_{NL} = 143.8750\ s$ respectively and the R-Squared error was $R_L^2 = 0.0615\ m^2$ and $R_{NL}^2 = 0.0174\ m^2$ respectively. Using the Linear model vs the Nonlinear model, we see an increase in accuracy of 71.74%, but we also see an increase in run time of 3272.75%. For quadcopter applications, run time is critical, and the system needs to be able to make a decision in real-time. These results show that while the nonlinear model based MPC is more accurate, it wouldn't be able to run in real-time. While we see a decrease in accuracy for the trajectory tracking, the linear model based MPC is able to run in real-time. For our real-time application, we use the linearized form of the quadcopter state-space in order to keep computational time down while maintaining a fair amount of accuracy.

For small oscillations and movements, a simplified state-space model is used and linearized for the quadcopter system. The system can be simplified by approximating the sine function with its argument and the cosine function with unity. This approximation gives us the following equations:

$$\begin{bmatrix} \dot{x}_{t+1} \\ \dot{y}_{t+1} \\ \dot{z}_{t+1} \end{bmatrix} = \begin{bmatrix} w_t(\phi_t\psi_t + \theta_t) - v_t(\psi_t - \phi_t\theta_t) + u_t \\ v_t(1 + \phi_t\psi_t\theta_t) - v_t(\phi_t - \psi_t\theta_t) + u\psi \\ w_t - u_t\theta_t + v_t\phi_t \end{bmatrix} \tag{2.9}$$

$$\begin{bmatrix} \dot{\phi}_{t+1} \\ \dot{\theta}_{t+1} \\ \dot{\psi}_{t+1} \end{bmatrix} = \begin{bmatrix} p_t + r_t\theta_t + q_t\phi_t\theta_t \\ q_t - r_t\phi_t \\ r_t + q_t\phi_t \end{bmatrix} \tag{2.10}$$

$$\begin{bmatrix} \dot{u}_{t+1} \\ \dot{v}_{t+1} \\ \dot{w}_{t+1} \end{bmatrix} = \begin{bmatrix} r_t v_t - q_t w_t - g\theta_t \\ p_t w_t - r_t u_t + g\phi_t \\ q_t u_t - p_t v_t + g \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -\frac{T}{m} \end{bmatrix} \tag{2.11}$$

$$\begin{bmatrix} \dot{p}_{t+1} \\ \dot{q}_{t+1} \\ \dot{r}_{t+1} \end{bmatrix} = \begin{bmatrix} \frac{I_y - I_z}{I_x} q_t r_t \\ \frac{I_z - I_z}{I_y} p_t r_t \\ \frac{I_x - I_y}{I_z} p_t q_t \end{bmatrix} + \begin{bmatrix} \frac{\tau_\phi}{I_x} \\ \frac{\tau_\theta}{I_y} \\ \frac{\tau_\psi}{I_z} \end{bmatrix} \tag{2.12}$$

Using these simplified equations, we can linearize the system around an equilibrium point $\bar{x}$ for a fixed input $\bar{u}$. One such equilibrium point is when the drone is hovering, this equilibrium point is described as follows:

$$\bar{\mathbf{x}} = \begin{bmatrix} \bar{x} & \bar{y} & \bar{z} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \in \mathbb{R}^{12} \tag{2.13}$$

$$\bar{\mathbf{u}} = \begin{bmatrix} mg & 0 & 0 & 0 \end{bmatrix}^T \in \mathbb{R}^4 \tag{2.14}$$

The associated matrices for linearizing the state-space matrix along this equilibrium point for equations (2.9) through (2.12) is described below:

$$\mathbf{A} = \left.\frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}}\right|_{\substack{x = \bar{x} \\ u = \bar{u}}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -g & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.15}$$

$$\mathbf{B} = \left.\frac{\partial \mathbf{f}\left(\mathbf{x},\mathbf{u}\right)}{\partial \mathbf{x}}\right|_{\substack{x=\bar{x}\\u=\bar{u}}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{m} & 0 & 0 & 0 \\ 0 & \frac{1}{I_x} & 0 & 0 \\ 0 & 0 & \frac{1}{I_y} & 0 \\ 0 & 0 & 0 & \frac{1}{I_z} \end{bmatrix} \tag{2.16}$$

Utilizing linearization of the State-Space matrix, this matrix can be tailored for different stability points and simplify calculations for small oscillations.

2.4 Overall dynamics

The overall dynamics of the Parrot® Mambo Drone can be written as a linear time invariant discrete time system with state dependent uncertainty as follows:

$$x_{t+1} = Ax_t + B\left(u_t + \Delta(x_t)\right), \tag{2.17}$$

Using the following constants:

$$\begin{aligned} \text{mass, } m &= 0.063\ kg \\ \text{distance between rotor and COM, } l &= 0.0624\ m \\ \text{lift constant, } k &= 0.0107 \\ \text{drag constant, } b &= 0.78264 * 10^{-3} \\ \text{element of inertia matrix, } I_x &= 0.0001\ \frac{kgm^2}{s} \\ \text{element of inertia matrix, } I_y &= 0.000071691\ \frac{kgm^2}{s} \\ \text{element of inertia matrix, } I_z &= 0.000058286\ \frac{kgm^2}{s} \end{aligned}$$

If we rewrite the control inputs to be the angular velocities as represented in equations (2.6)-(2.8), and set the equilibrium point to be the quadcopter hovering at the desired altitude

of $\bar{z} = 2$, the equilibrium point can be written as:

$$\bar{\mathbf{x}} = \begin{bmatrix} 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \in \mathbb{R}^{12} \tag{2.18}$$

$$\bar{\mathbf{u}} = \begin{bmatrix} 14.4 & 14.4 & 14.4 & 14.4 \end{bmatrix}^T \in \mathbb{R}^4 \tag{2.19}$$

where discrete time system can be written as:

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -9.81 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 9.81 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.20}$$

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.1698 & 0.1698 & 0.1698 & 0.1698 \\ 0 & -11.4552 & 0 & -11.4552 \\ -9.3133 & 0 & 9.3133 & 0 \\ -7.8264 & 7.8264 & -7.8264 & 7.8264 \end{bmatrix} \tag{2.21}$$

In the simulation, our disturbance, $\Delta(x_t)$, will be generated in a few different ways to try to mimic some of the conditions that quadcopters experience in various operation environments and usages. More detail about these disturbances will be explored in our experimental results in Chapter 5 where we discuss the simulation setup.

Chapter 3

MODEL PREDICTIVE CONTROL

In this chapter, an overview and salient features of model predictive control (MPC) are provided. In particular, we first summarize tube-based MPC because of its inherent robustness compatibility with learning based techniques. By breaking the control command in two components – MPC component and adaptive component, and by considering adaptive component as an additional disturbance for MPC, we introduce adaptive MPC. The MPC control component aims to achieve input-to-state stability (ISS) with respect to total experienced disturbance, and the adaptive control component tries to adapt the state-dependent uncertainties. In case of matched and structured uncertainties, this method ensures convergence of states to origin due to converging input converging state (CICS) property of ISS. For unstructured uncertainties, a single layer neural network (NN) can be employed to get an approximate structure. This approach, *shallow MPC*, demonstrates a reasonable performance but is unable to possess long term learning and generalization. Therefore, we replace the single layer NN by a deep NN. This final technique is called deep MPC. In this thesis, these different techniques are utilized and compared for the autonomous flight of a quad-copter simulation.

Consider a discrete time dynamical system

$$x_{t+1} = Ax_t + Bu_t, \tag{3.1}$$

where $x_t \in \mathcal{X}$ and $u_t \in \mathbb{U}$. We first fix an optimization horizon N and choose positive definite matrices $Q, R \succ 0$. The matrices $Q, R \succ 0$ are used to construct the cost per stage function. The terminal constraint set $\mathcal{X}_f$ and the terminal penalty matrix $Q_f \succ 0$ are constructed as per the following assumption:

Assumption 1. There exists a control $u' \in \mathbb{U}$ such that the following holds:

$$\|Ax + Bu'\|_{Q_f}^2 - \|x\|_{Q_f}^2 \leq -\|x\|_Q^2 + \|u'\|_R^2 \tag{3.2}$$

for every $x \in \mathcal{X}_f$.

The standard MPC algorithm is given as follows:

Algorithm 1 MPC

0: **for** each t **do**
1: measure x_t
2: solve

$$\begin{aligned} \min_{(v_i)_{i=0}^{N-1}} \quad & \|z_N\|_{Q_f}^2 + \sum_{i=0}^{N-1} \|z_i\|_Q^2 + \|v_i\|_R^2 \\ \text{s.t.} \quad & z_0 = x_t, \\ & z_{i+1} = Az_i + Bv_i \text{ for } i = 0, \ldots, N-1, \\ & z_i \in \mathcal{X} \text{ for } i = 1, \ldots, N-1, \\ & v_i \in \mathbb{U} \text{ for } i = 0, \cdots, N-1, \\ & z_N \in \mathcal{X}_f. \end{aligned} \tag{3.3}$$

3: apply $u_t = v_0$ to the system

The above algorithm guarantees satisfaction of state and control constraints, and convergence of states to origin. When the system is affected by disturbances, then robust and adaptive versions of the above algorithm are required. In this chapter, we will highlight the modifications needed in the above algorithm to control the Parrot® Mambo Drone.

3.1 Tube-Based Model Predictive Control

Tube based MPC is based on ensuring that the closed-loop states in the presence of disturbance stay within a tube around a nominal reference trajectory. We obtain a reference trajectory by solving a reference governor problem offline, and utilize constraint tightening in our reference governor to satisfy the state constraints of our system. To tighten our constraints, knowledge of the exact bounds on disturbance is needed. The disturbance we see in our dynamical system (2.17) at time t is $B\Delta(x_t)$. We assume that the state-dependent uncertainty $\Delta(x_t)$ is bounded. Therefore, $B\Delta(x_t)$ is also bounded. To tighten our constraints, we must obtain the sets of our various states. To do this We construct a series of polyhedron that contain the bounds of our possible states, control authority, and disturbance using the upper and lower bounds of the possible states, control authority and disturbance respectively. These polyhedron represent the sets of our State $\mathbb{X}$, Control Authority $\mathbb{U}$, and disturbance $\mathbb{W}$. We

then compute the invariant set $S_K(t)$ such that:

$$S_K(t) = \sum_{t=0}^{t-1} (A + BK)^t \mathbb{W} \tag{3.4}$$

We can then use the invariant set to compute the nominal sets of our state and control authority such that:

$$\bar{\mathbb{X}} = \mathbb{X} \ominus S_K \tag{3.5}$$

$$\bar{\mathbb{U}} = \mathbb{U} \ominus K S_K \tag{3.6}$$

These tightened constraint sets can then be used to find the upper and lower bounds for the nominal state outputs and control inputs. The derivation of equations 3.5 and 3.6 can be found in Rawlings [23].

We then obtain a reference trajectory by solving the optimal control problem with penalty matrices $Q, R \succ 0$, optimization horizon $N \in \mathbb{Z}_+$ and tightened sets $\mathbb{X}_r$ and $\mathbb{U}_r$:

$$\begin{aligned} \min_{(u_i^r)_{i=0}^{N-1}} \quad & \sum_{i=0}^{N-1} \|x_i^r\|_Q^2 + \|u_i^r\|_R^2 \\ \text{s.t.} \quad & x_0^r = x_0, x_N^r = \mathbf{0}, \\ & x_{i+1}^r = A x_i^r + B u_i^r, x_i^r \in \mathbb{X}_r \subset \mathbb{X}, \\ & u_i^r \in \mathbb{U}_r \subset \mathbb{U}; i = 0, \cdots, N-1. \end{aligned} \tag{3.7}$$

The solution to this optimization problems produces the following reference trajectory:

$$\begin{aligned} (x_t^r)_{t \in \mathbb{N}_0} &:= \{(x_t^r)_{t=0}^{\mathbb{N}-1}, 0, \dots\}, \\ (u_t^r)_{t \in \mathbb{N}_0} &:= \{(u_t^r)_{t=0}^{\mathbb{N}-1}, 0, \dots\} \end{aligned} \tag{3.8}$$

For the online reference tracking MPC, we need to choose an optimization horizon $N \in \mathbb{Z}_+$, and positive definite matrices $Q, R \succ 0$. These parameters do not need to match the same parameters used to solve the offline reference governor. Let:

$$c_s(x_{t+i|t}, u_{t+i|t}) := \|x_{t+1|t} - x_{t+i}^r\|_Q^2 + \|u_{t+1|t} - u_{t+i}^r\|_R^2 \tag{3.9}$$

be the cost per state at time $t + i$ predicted at time t, and let:

$$c_f(x) := x^T Q_f x \tag{3.10}$$

be the terminal cost with $Q_f \succ 0$. The terminal cost c_f is treated as a local control Lyapunov

function within a terminal set:

$$\mathcal{X}_f := \left\{x \in \mathbb{R}^d | c_f(x) \leq \alpha; \alpha > 0\right\} \tag{3.11}$$

by making the following assumption:

Assumption 2. There exists a control $u' \in \mathbb{U}$ such that the following holds:

$$c_f(\bar{f}(x, u\prime)) - c_f(x) \leq -c_s(x, u') \tag{3.12}$$

for every $x \in \mathcal{X}_f$.

This assumption is standard in literature and can be referred to in mayne [24]. The terminal set $\mathcal{X}_f$ is defined as a level set of terminal cost.

The optimal control problem for the online reference tracking MPC can be represented as:

$$\begin{aligned} \min_{(u_{t+i|t})_{i=0}^{N-1}} \quad & x_{t+N|t}^\top Q_f x_{t+N|t} + \sum_{i=0}^{N-1} \|x_{t+i|t} - x_{t+i}^r\|_Q^2 + \|u_{t+i|t} - u_{t+i}^r\|_R^2 \\ \text{s.t.} \quad & x_{t|t} = x_t, \\ & x_{t+i+1|t} = Ax_{t+i|t} + Bu_{t+i|t} \text{ for } \ i = 0, \cdots, N-1, \\ & u_{t+i|t} \in \mathbb{U} \text{ for } i = 0, \cdots, N-1. \end{aligned} \tag{3.13}$$

The first component of $u_{t|t}$ is called the MPC component, u_t^m which is applied to the quadcopter at time t and gives us a result that satisfies our control constraints. The overall algorithm for offline and online portions of the Tube MPC have been rewritten as:

Algorithm 2 Tube MPC

Require: x_0, Q, R, N

1: get the reference trajectory $(x_t^r)_{t\in\mathbb{N}}, (u_t^r)_{t\in\mathbb{N}}$

1: **for** each t **do**

2: measure x_t

3: solve

$$\begin{aligned} \min_{(u_{t+i|t})_{i=0}^{N-1}} \quad & x_{t+N|t}^\top Q_f x_{t+N|t} + \sum_{i=0}^{N-1} \|x_{t+i|t} - x_{t+i}^r\|_Q^2 + \|u_{t+i|t} - u_{t+i}^r\|_R^2 \\ \text{s.t.} \quad & x_{t|t} = x_t, \\ & x_{t+i+1|t} = Ax_{t+i|t} + Bu_{t+i|t} \text{ for } \ i = 0, \cdots, N-1, \\ & u_{t+i|t} \in \mathbb{U} \text{ for } i = 0, \cdots, N-1. \end{aligned} \tag{3.14}$$

4: apply $u_t = u_{t|t}$ to the system

The Tube-Based Model Predictive Control and overall Control System Architecture can be simplified in model form. In figure 3.1 and figure 3.2 below, a model of the Tube MPC System is represented. After initializing the Nominal and Tube MPC's, we input the current state $y(t)$, and the reference trajectory $r(t)$. The Nominal MPC then generates a nominal state and control trajectory,$x_n(t)$ and $u_n^m(t)$, that satisfies the tightened constraints and passes those trajectories to the Tube MPC. The Tube MPC then takes the current states of the system, nominal state trajectory, and nominal control trajectory, and represented determines a control input to keep the state and control of the system close to the nominal trajectory.

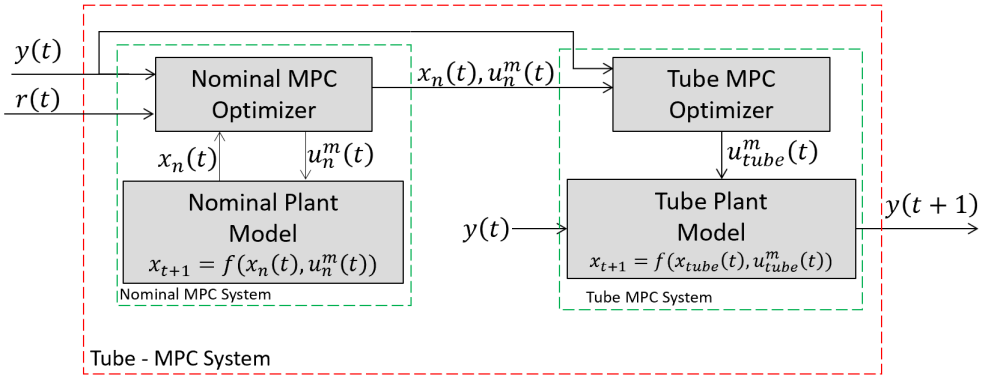


Figure 3.1: MPC Control System Plant Model

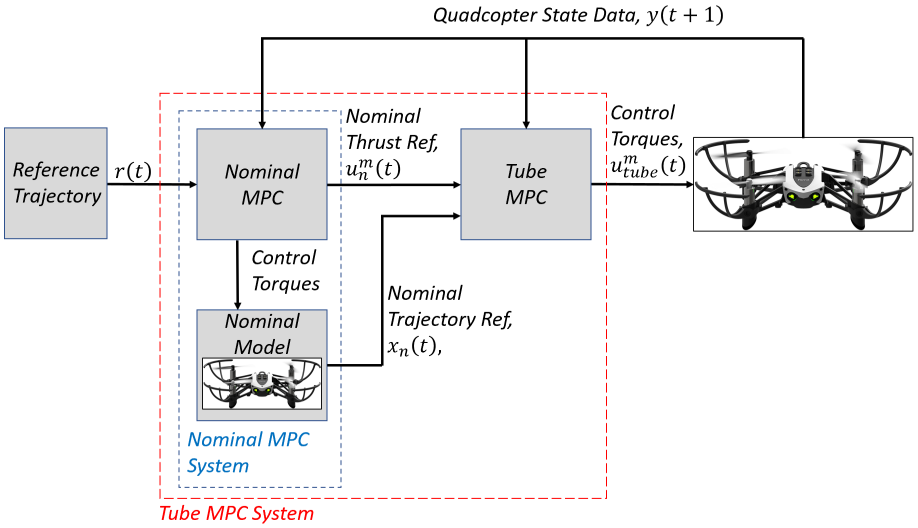


Figure 3.2: Control System Architecture

3.2 Adaptive Tube-Based Model Predictive Control

For Adaptive Tube-Based Model Predictive Control and Deep Tube-Based Model Predictive Control, we make slight modifications to the previously discussed control architecture. Consider the dynamical system (2.17). For the Adaptive MPC and Deep MPC, the control input u_t is broken up into:

$$u_t = u_t^m + u_t^a \tag{3.15}$$

where u_t^m represents the MPC control input and u_t^a represents the neuro-adaptive control input. A disturbed system utilizing the Adaptive MPC controller can best be described as:

$$x_{t+1} = Ax_t + B(u_t^m + u_t^a + \Delta(x_t)) \tag{3.16}$$

From this equation, it can be noted that the MPC experiences the adaptive control input as a disturbance. In this sense, u_t^m is responsible for input-to-state stability of the close loop system in the presence of bounded disturbances, and u_t^a is responsible for approximating $-\Delta(x_t)$ and staying within a known bound. In the case that the uncertainty is structured, given that there exists a known basis function, $\phi(x) \in \mathbb{R}^n$, and a set of ideal weights that correspond to that basis function, W^*, the disturbance can be represented such that:

$$\Delta x_t = W^{*T}\phi(x_t) \tag{3.17}$$

And the adaptive control element can be represented as:

$$u_t^a = -K_t^\top \phi(x_t) \tag{3.18}$$

In the case where uncertainty is unstructured, given that the disturbance is continuous and is defined over a compact set, then:

$$\Delta(x_t) = W^{*\top}\Phi x_t + \varsigma_1(x_t) \tag{3.19}$$

where $\phi(x)$ is a Gaussian Basis Function Network, in the case of Adaptive MPC [25], or the basis function of a Neural Network adaptive element [26], in the case of Deep MPC, then the adaptive control element can be represented as:

$$u_t^a = -K_t^\top \Phi(x_t) \tag{3.20}$$

The Gaussian radial basis function is used for our adaptive control law and is expressed as:

$$\phi(x_t) = e^{-\left(\frac{x-\mu}{\sigma}\right)^2} \tag{3.21}$$

where μ is the radial basis function center, and σ is the standard deviation.

For our purposes, we employ the following weight update law for our adaptive control law:

$$W_{t+1} = W_t + \frac{\Gamma}{\|\phi(x_t)\|^2}\phi\left(x_t\right)\left(B^{\dagger}\left(x_{t+1} - \bar{x}_{t+1}\right)\right)^T \tag{3.22}$$

where Γ is our learning rate, $\phi\left(x_t\right)$ is our feature basis function, $B^{\dagger}$ is the pseudo-inverse of B. By taking the difference between the nominal states determined by the MPC, $\bar{x_t}$, and the actual state, x_{t+1}, we get the added disturbance to the system. To ensure boundedness of our weights we employ a discrete project method as follows:

$$K_t = Proj\ \bar{K_t} = \begin{cases} \bar{K_t} & \text{if } \|\bar{K_t}\| \leq \bar{w} \\ \frac{\bar{w}}{\|\bar{K_t}\|}\bar{K_t} & \text{otherwise} \end{cases} \tag{3.23}$$

Looking back at the model of our Adaptive MPC model with added disturbance,

$$x_{t+1} = Ax_t + B(u_t^m + u_t^a + \Delta(x_t)) \tag{3.24}$$

we notice that a part of the control input to the system is utilized by the adaptive controller, so we rewrite the control set utilized by the offline reference governor described in the previous section. The control set can be rewritten as:

$$\mathbb{U}' := \{v \in \mathbb{R}^m \mid \|v\|_\infty \leq u_{max} - u_{max}^a\} \tag{3.25}$$

We also redefine the online reference tracking MPC as:

$$\begin{aligned} \min_{(u_{t+i|t})_{i=0}^{N-1}} \quad & x_{t+N|t}^\top Q_f x_{t+N|t} + \sum_{i=0}^{N-1} \|x_{t+i|t} - x_{t+i}^r\|_Q^2 + \|u_{t+i|t} - u_{t+i}^r\|_R^2 \\ \text{s.t.} \quad & x_{t|t} = x_t, u_{t|t} + u_t^a \in \mathbb{U}', \\ & x_{t+i+1|t} = A\overline{x}_{t+i|t} + B\overline{u}_{t+i|t} \text{ for } i = 0, \cdots, N-1, \\ & u_{t+i|t} \in \mathbb{U}' \text{ for } i = 0, \cdots, N-1. \end{aligned} \tag{3.26}$$

We then take the first component of the optimizer of the above problem, $u_{t|t}^*$, to be u_t^m and apply it along with u_t^a to the quadcopter at time t and gives us a result that satisfies our control constraints. The overall control system for Adaptive MPC has been rewritten as:

Algorithm 3 Adaptive Tube MPC

Require: $x_0, Q, R, N\phi$

choose Γ

initialize $K_0 = \mathbf{0}, t = 0$

get the reference trajectory $(x_t^r)_{t\in\mathbb{N}}, (u_t^r)_{t\in\mathbb{N}}$

for each t **do**

measure x_t

compute $u_t^a = -K_t\phi(x_t)$

solve

$$\begin{aligned} \min_{(u_{t+i|t})_{i=0}^{N-1}} \quad & V(x_t, (u_{t+i|t})_{i=0}^{N-1} \\ \text{s.t.} \quad & x_{t|t} = x_t, \\ & u_{t|t} = u_t \in \mathbb{U}, \\ & x_{t+i+1|t} = Ax_{t+i|t} + Bu_{t+i|t} \text{ for } \ i = 0, \cdots, N-1, \\ & u_{t|t} + u_t^a \in \mathbb{U} \text{ for } i = 0, \cdots, N-1. \end{aligned} \tag{3.27}$$

set $u_t^m = u_{t|t}$

apply $u_t = u_t^m + u_t^a$ to the system and measure x_{t+1}

compute K_{t+1} by the weight update law:

$$K_{t+1} = K_t - \frac{\Gamma B^\dagger (x_{t+1} - \overline{x}_{t+1})\phi(x_t)^T}{\varepsilon + \|\phi(x_t)\|^2}$$

Finite Horizon Cost Function:

$$V(x_t, u_{t+i|t})_{i=0}^{N-1}) := c_f(x_{t+N|t}) + \sum_{i=0}^{N-1} c_s(x_{t+i|t}, u_{t+i|t})$$

3.3 Deep Tube-Based Model Predictive Control

Deep Tube-Based MPC is a learning-based controller that incorporates deep learning and the Adaptive Tube-Based MPC algorithm. While Adaptive Tube-Based MPC outperforms conventional Tube-Based MPC, single layer neurons utilized to calculate the adaptive control input cannot update properly. To improve the performance of these neurons, they need to be states that the system is experiencing in real time. By utilizing deep neural networks to represent the complex nonlinearities experienced by the system and the stability guarantees associated with it, the system will see better performance in the face of nonlinear disturbances. Details regarding in-depth derivations of equations, stability proofs and boundedness of Deep Tube-Based Model Predictive control can be found in [2].

The general architecture of Deep Tube-Based Model Predictive Control utilizes a similar architecture to [1] with some slight modifications:

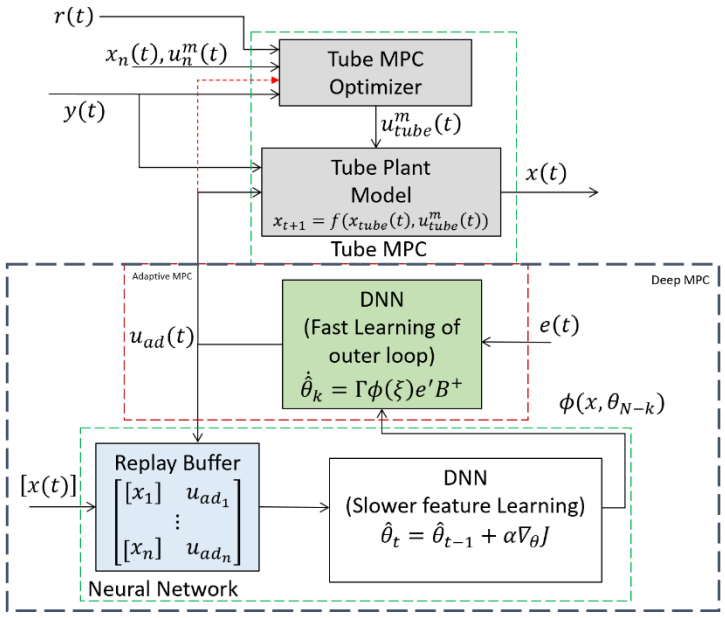


Figure 3.3: Deep Tube-Based MPC Architecture

The control architecture can be split into four major elements: the Tube MPC, Fast Learning Outer Loop (Adaptive Control Law, DNN), Replay Buffer, and the DNN (Slow feature learning). The Tube MPC takes the current position, nominal trajectory and nominal control inputs and optimizes its next control input as described in Section 3.2.1. It then takes the next control input and the adaptive control input determined and simulates the next state of the system. The Fast-Learning Outer Loop, represented by the Adaptive Control Law or the DNN, takes the estimate state and determines the estimated uncertainty. The estimated uncertainty will then be used by the system as the control input for its next time step and is stored along with the state at that time inside the replay buffer as a data point. Random batches of data are then taken from the replay buffer and used to train a neural network utilizing stochastic gradient descent method to map states to use estimated uncertainty values. The basis function $\phi(x_t)$ is then calculated and passed to the Fast-Learning Outer Loop, which completes the update loop. The Deep Tube-Based Model Predictive Control algorithm is written as:

For Deep Tube-Based Model Predictive Control, we want to train the neural network based on the disturbances and states seen during the Tube-MPC and Adaptive Tube-MPC process. As the adaptive control law is running at every time step, the neural network is training off of the data collected in the replay buffer at every time step. While both functions are performed at every time step, we do not update the basis function of the adaptive control

Algorithm 4 Deep Tube MPC: Online Reference Tracking

Require: x_0

choose $\theta \leq \frac{1}{\sigma^2}$

Generate the reference trajectory from algorithm 1 under the tightened constraints
initialize $K_0 = \mathbf{0}, t = 0$ and randomly assign weights of inner layer to get ϕ
initialize a generative network on a parallel machine

for each t **do**

update ϕ if training in the generative network is complete, otherwise move to the next step

compute $u_t^a = -K_t \phi(x_t)$

solve algorithm 2, set $u_t^m = u_{t|t}$

apply $u_t = u_t^m + u_t^a$ to the system and measure x_{t+1}

compute K_{t+1} by the weight update law:

$$K_{t+1} = K_t - \frac{\Gamma B^{\dagger}(x_{t+1} - \overline{x}_{t+1})\phi(x_t)^T}{\varepsilon + \|\phi(x_t)\|^2}$$

law at every time step. After a few training iterations we update the basis function of the adaptive control law which is based on the uncertainties and state dated collected. This allows us to produce better estimates of the uncertainties and improve tracking performance for the system.

Chapter 4

EXPERIMENTAL SETUP

In this chapter, we will introduce the hardware and testing facilities that will be used to validate the results seen in the simulations from chapter 4. This chapter continues the discussion in chapter 3 where we introduced the model predictive control architecture and continue with the methods proposed for hardware implementation of the different MPC control architecture.

4.1 Parrot Mambo Mini Drone

For our flight test, we will be using the Parrot® Mambo Drone that were utilized in [1].

Figure 4.1: Parrot Mambo Mini Drone [27]

These drones are relatively small, low cost and ease of testing control algorithms due to its compatibility with Simulink. The parrot mambo mini drone utilizes an IMU (3 axis accelerometers and 3 axis gyroscope), ultrasound sensors and camera to aid in its state estimation. Similar to the DMRAC experiments, the algorithm will utilize an on-board, off-board architecture, which we will discuss later in this chapter.

4.2 Test Facility

Figure 4.2: UIUC Mobile Robotics Arena [28]

The initial and continued experiments for testing the different MPC control architectures will be conducted in CSL Studio VICON facility and drone arena at the University of Illinois at Urbana Champaign. To give accurate position tracking for the state estimator, the VICON arena's motion capture system will be used to give us greater accuracy. The VICON tracking arena is equipped with 8 VICON T40 motion capture cameras that can run at 370 frames per second and have millimeter accuracy.

4.3 Control Architecture Implementations

For the various MPC architecture implementation, we utilize an on-board, off-board architecture due to the limitations of the drone. The Mambo Parrot Mini drone has a dedicated processor to perform its on-board control, sensor processing, and state estimation. While this processor is sufficient for performing some additional calculations, it is unable to perform the optimization algorithms required for MPC as well as the additional NN needed for the Deep-MPC architecture. To solve this, we utilize an asynchronous on-board and off-board control system, as shown in 4.3. We utilize Simulink off-board to collect position data from the VICON arena's motion capture system, as well as iterate the NN with the data captured by the drone. We then transmit the data via Bluetooth to the quadrotor which performs the MPC element on-board to determine the next control input.

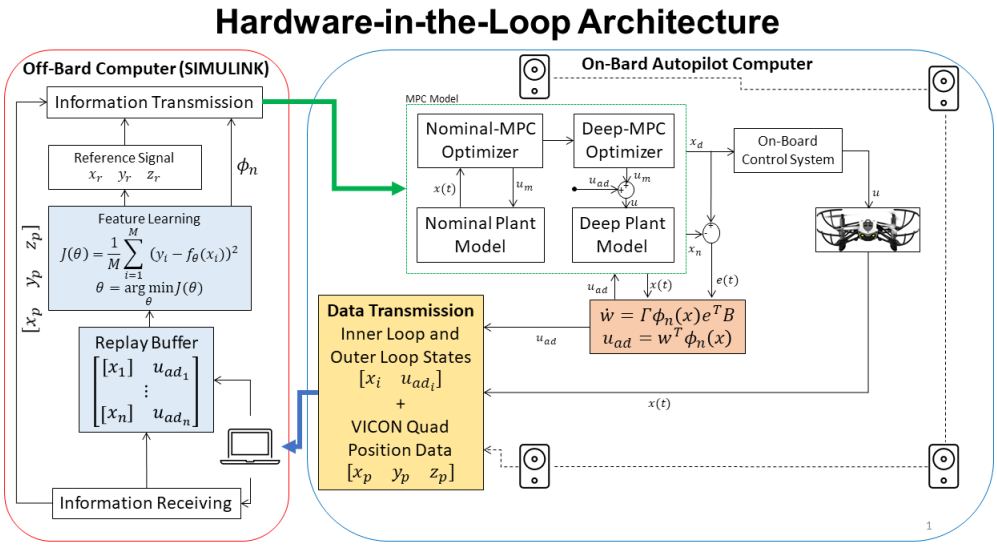


Figure 4.3: Hardware-In-the-Loop Architecture for Deep MPC. This utilizes the algorithms described in Section 3, and Quadrotor Models described in Section 2

Chapter 5

SIMULATION RESULTS

In this section, the flight simulation results will be presented running the different MPC algorithms utilizing a linear quadrotor dynamic system with various disturbances and different trajectories. The code for these simulations is available on GitHub for reference and external use [29]. This chapter is split into four sections where we will review different disturbances that were simulated. Further results are presented in Appendix A for review. Performance for tracking state trajectories from the nominal trajectory will be done by average XY mean square error for the figure 8 and circle maneuver, and altitude mean square error for the hover maneuver.

5.1 Reference Trajectory tracking: Nominal and High Wind Velocity Conditions

In these simulations, each controller's performance is evaluated for tracking a Parrot® Mambo that is moving in a figure-8 trajectory for nominal and high wind velocity conditions.The hyperparameters for the Tube, Adaptive and Deep MPC are tuned on this case to ensure each controller performs equally well. For each plot we will use the colors Red, Blue, and Green to distinguish Tube, Adaptive, and Deep MPC results. The parameters obtained from this experiment are kept fixed throughout the remainder of the other experiments. In Figures 5.1 and 5.2 below we demonstrate the tracking capabilities of the different MPC algorithms in Nominal Conditions. Since each controller is tuned to perform at its best the difference between each controller is negligible, for future reference the XY position mean square error (MSE) is $MSE_{XY} = 0.02016$.

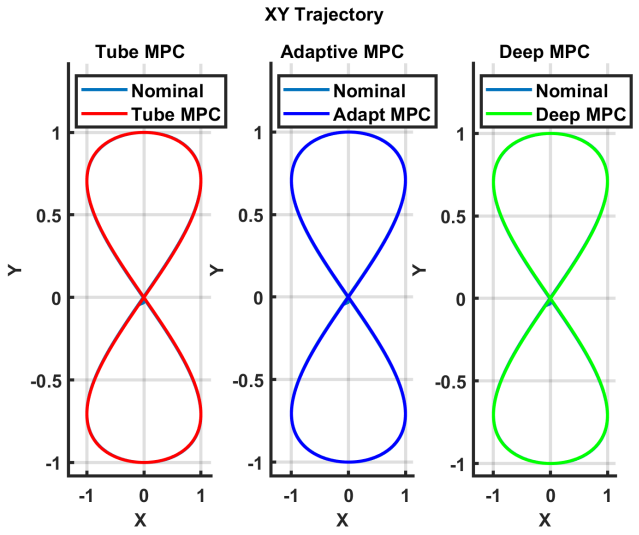


Figure 5.1: MPC Controls performance in tracking a trajectory while following a hover maneuver in nominal conditions.

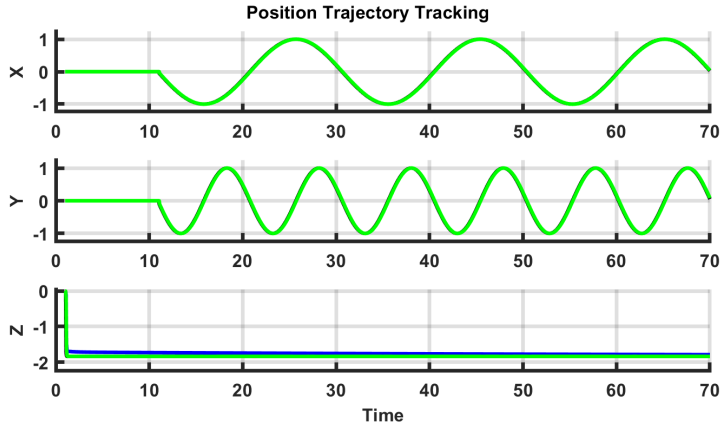


Figure 5.2: MPC Controls performance in tracking position while following a hover maneuver in nominal conditions

In these next set of results, the quadrotor experiences a high velocity wind disturbance while performing the figure 8 maneuver. This simulation is designed to test the MPCs capabilities in trajectory tracking while experiencing a disturbance. We inject the disturbance at the 35-second mark of the simulation and stop the disturbance at the 55-second mark. It can be seen that all controllers perform very similarly in that they have difficulty tracking the exact position of the figure 8 trajectory, but still execute the pitch and roll maneuvers

that give us the figure. These results are presented below in Figures 5.3 and 5.4.

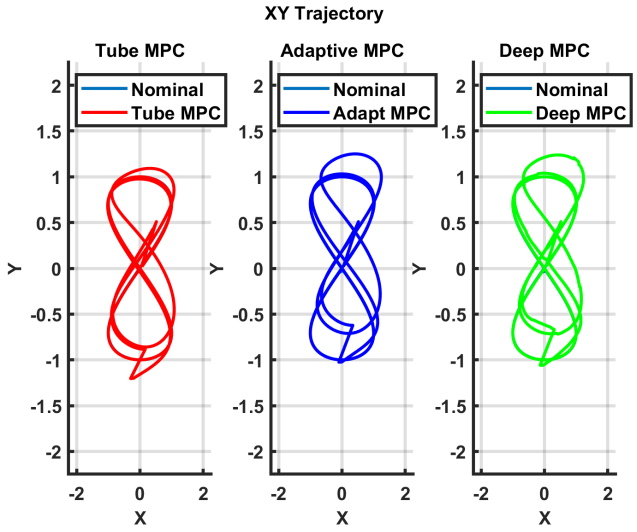


Figure 5.3: MPC Controls performance in tracking a trajectory while following a hover maneuver in high wind velocity conditions.

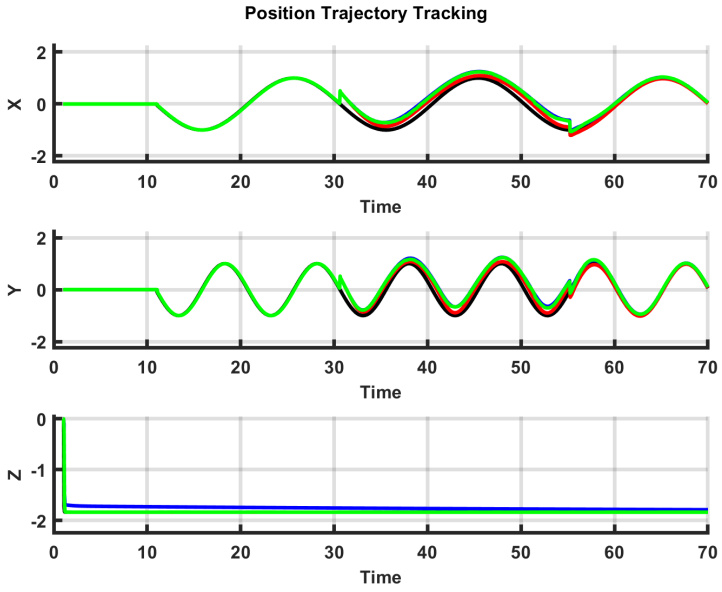


Figure 5.4: MPC Controls performance in tracking position while following a hover maneuver in high wind velocity.

5.2 Reference Trajectory Tracing: Change in Center of Momentum

In these simulations, we shift the center of momentum of the quadrotor slightly to create a nonlinear disturbance. In the field, some vehicles experience changes in the center of momentum (COM) for various reasons. This change in COM cause a change in the dynamics of the original system in a way that will make it so control systems that are preset on the original model will have difficulty to track their intended trajectory. As discussed in Chapter 3, Linear MPC is initialized on a preset model of the system in order to generate the control inputs to track the trajectory. By changing the model of the system after initialization, the model of the MPC no longer matches the model of the system. For these simulations, we change the I_x and I_y values slightly at time $t = 15s$. In Figures 5.5 and 5.6 below we can see that this change of COM an impact on the tracking of all MPC controllers. We do see a shift in the tracking for the Adaptive and Deep MPC as they attempt to correct their control inputs to track the original dynamics of the quadrotor. From these results we can analyze the XY position tracking error of each of the controllers, and from those results we see that Deep MPC is able to beat that of Tube and Adaptive MPC where we see that $MSE_{Deep} = 0.0536$, $MSE_{Adapt} = 0.0539$, and $MSE_{Tube} = 0.0683$. This correlates to an 27% improvement from Tube to Deep MPC, and an 0.5% improvement from Adaptive to Deep MPC. The plot of the XY position tracking error is presented in Figure 5.7.

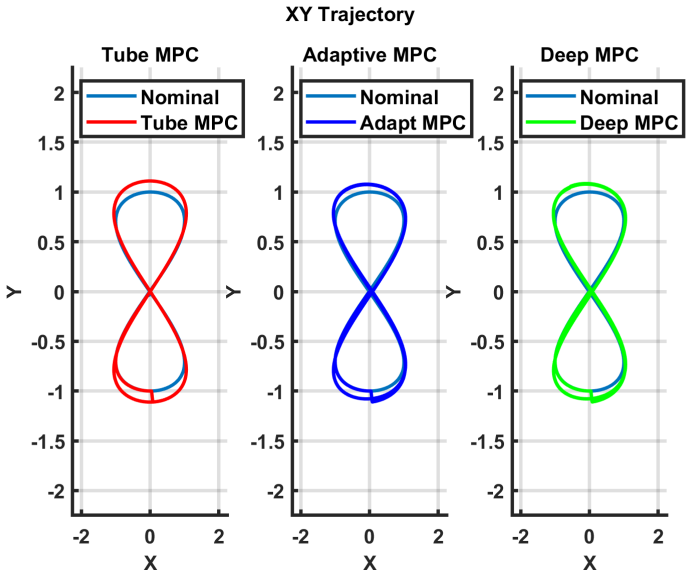


Figure 5.5: MPC Controls performance in trajectory tracking position when experiencing a COM shift in a figure 8 maneuver. Fault is injected at $t = 15s$

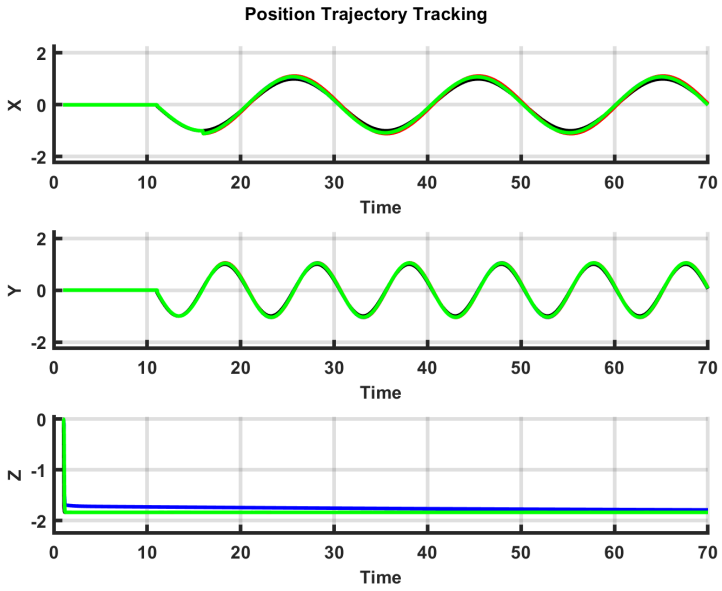


Figure 5.6: MPC Controls performance in tracking position when experiencing a COM shift in a figure 8 maneuver. Fault is injected at $t = 15s$

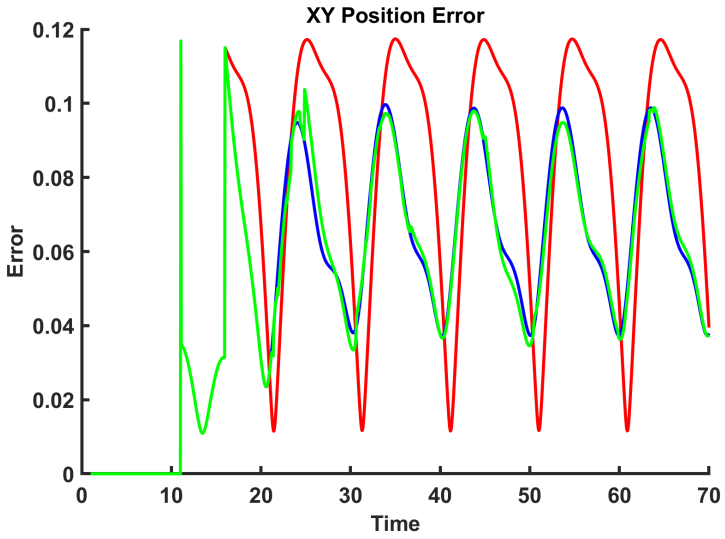


Figure 5.7: Position deviation for each of the control implementations when there is a COM shift. Fault is injected at $t = 15s$

5.3 Reference Trajectory Tracking: Fault Tolerance

In these next simulations, we demonstrate the fault-tolerance capabilities of the quadcopter during flight. Quadrotor blades and motors can be damaged during operation, especially if not well maintained, and damages to these reduces the thrust and control capacity of the quadcopter. For this simulation we demonstrate the fault tolerance capabilities for a quadrotor that is hovering and at time $t = 15s$ we reduce the control input for one of the motors to simulate a damaged motor. The results of this simulation are presented in Figure 5.8. From these simulations, we can see that Deep MPC is able to outperform Adaptive and Tube MPC by a good margin. It is also able to get the closest to the desired altitude of $Z = 2m$. This is also be seen in the altitude error displayed in 5.8. The altitude error for each of the controllers is as follows: $MSE_{Deep} = 0.01296$, $MSE_{Adapt} = 0.09537$, and $MSE_{Tube} = 0.015428$. This correlates to a 19% improvement from Tube MPC to Deep MPC.

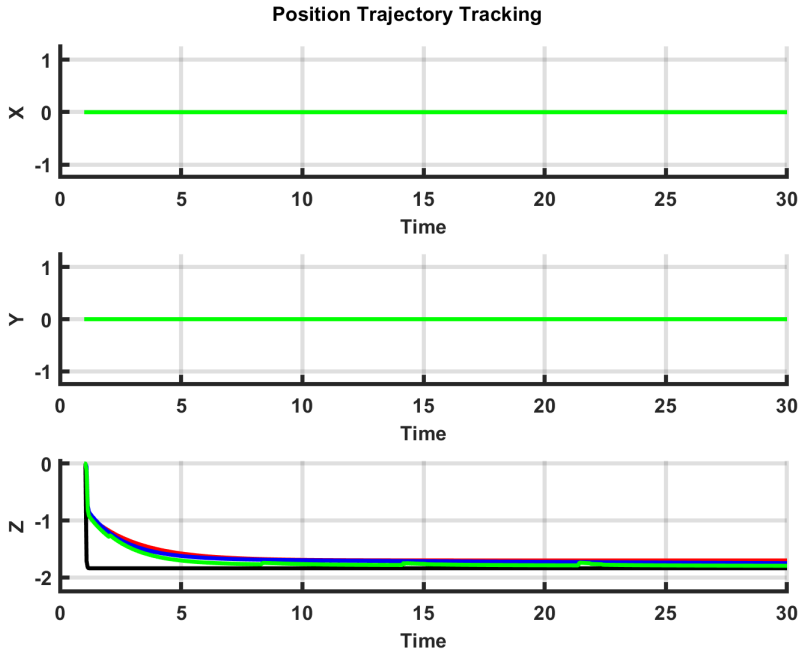


Figure 5.8: MPC Controls performance in tracking position while following a hover maneuver. Fault is injected at $t = 15s$

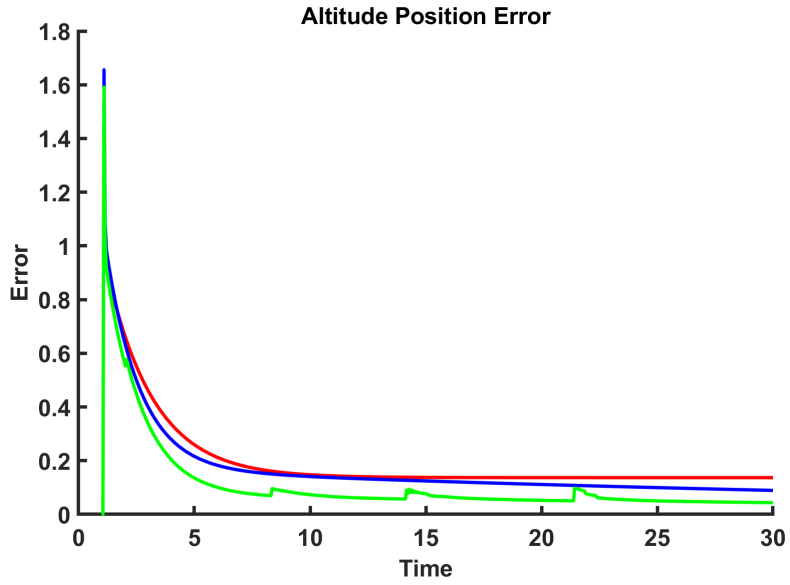


Figure 5.9: Altitude deviation for each of the control implementations when the control input is reduced for the system through a fault. Fault is injected at $t = 15s$

5.4 Reference Trajectory: Payload Drop

In these simulations, a change in the equilibrium point of the quadcopter mid-flight. Imagine a drone that is carrying a payload to a destination. In order to maintain altitude, the drone's thrust needs to match the total mass of the drone and the payload. The drone arrives at its drop off point and releases the package, now its total mass has changed and so does the minimum thrust requirements to maintain altitude. As discussed in chapter 2, when you linearize the state dynamics of a system, you linearize it about an equilibrium point. For these simulations, we increase the mass of the quadrotor at initialization and the control set point so that the minimum thrust is met. The control set point for these parameters is $u_{eq} = 15.3568$ for each control variable. At time $t = 15s$ we simulate dropping the payload by reducing the mass and setting the control set point back to the nominal set point shown in section 2.4. The results of this simulation can be seen in Figures 5.10 and 5.11 below. From these results we can see that both Deep and Adaptive MPC are able to begin to readjust to the nominal dynamics, and if we take a closer look at the state variables we can see that Deep MPC approaches this point faster and earlier than Adaptive MPC. The XY Position MSE for each controller is as follows: $MSE_{Deep} = 0.11368$, $MSE_{Adapt} = 0.11443$, and $MSE_{Tube} = 0.12190$. This correlates to a 7.23% performance increase from Tube to Deep MPC and a 0.66% improvement from Adaptive to Deep MPC. The XY position tracking

error is presented in Figure 5.12

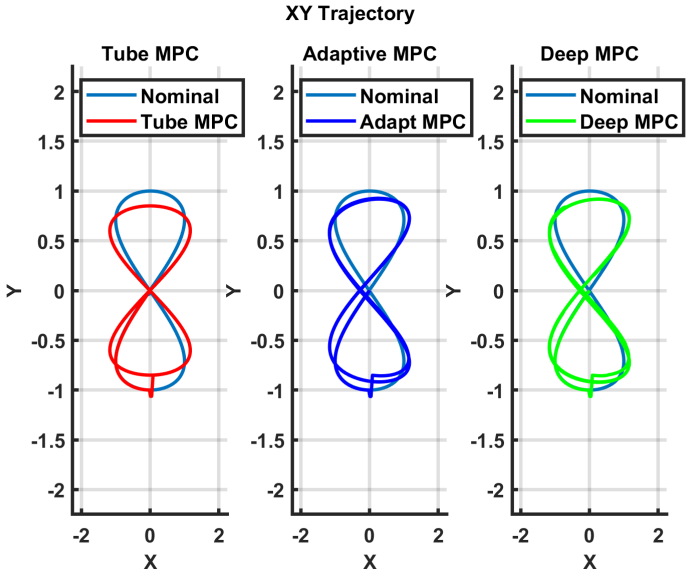


Figure 5.10: MPC Controls performance in tracking while following a figure 8 maneuver. Payload is dropped at $t = 15s$

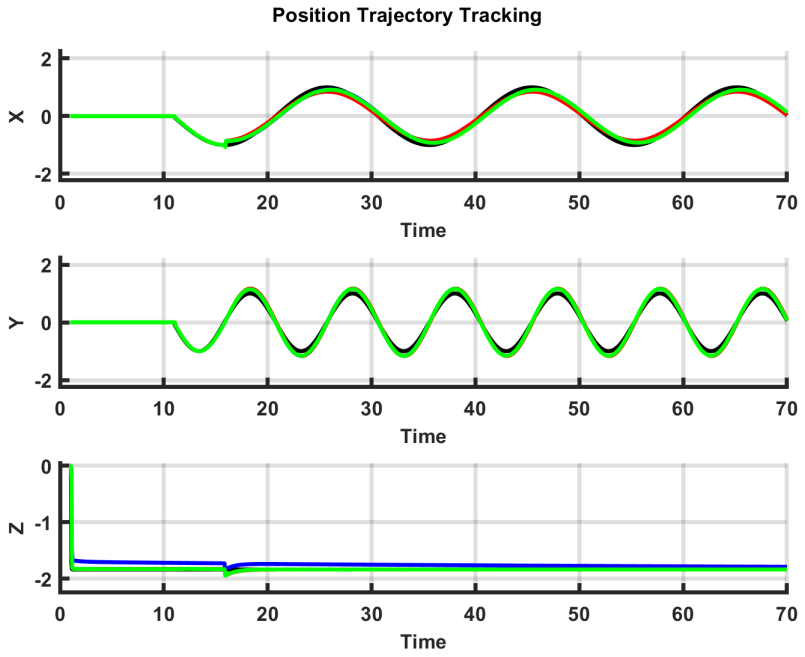


Figure 5.11: MPC Controls performance in tracking position while following the figure 8

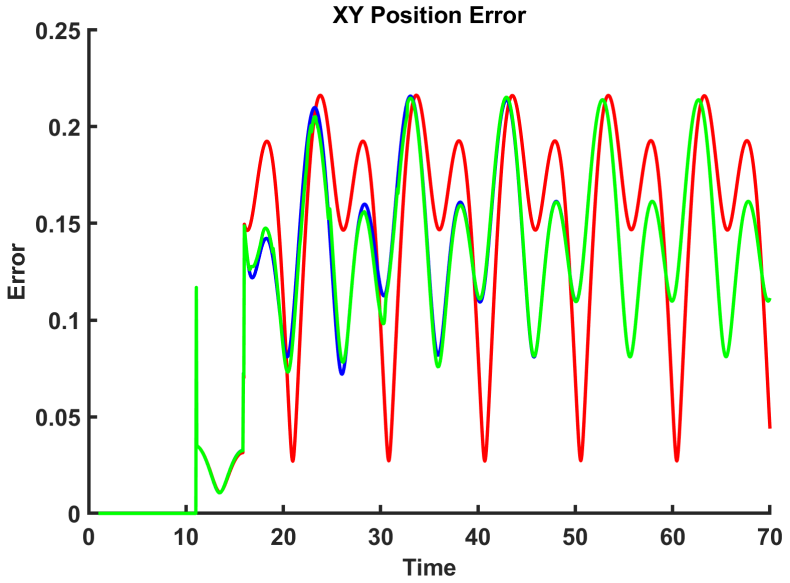


Figure 5.12: XY Position deviation for each of the control implementations when there is a reduction in the system's mass. Fault is injected at $t = 15s$

Chapter 6

CONCLUSION AND FUTURE WORK

In this thesis, it has been demonstrated that successful implementation of Deep MPC improves the tracking capability of the quadrotors when they experience disturbances that change the system's dynamic model. It has also been demonstrated that even in the wake of compounding disturbances that the Deep MPC is able to minimize error while providing stability guarantees. The disturbances demonstrated are not uncommon for quadrotors and mobile robots used in the field today. These results are good indications that implementation of Deep MPC on hardware could improve overall performance in these circumstances.

To validate these results for real world application, the MPC architectures discussed in this thesis are being implemented on the Parrot® Mambo Drone hardware. For this implementation we will replicate the experiments seen in this thesis as well as in [6]. One future possible directions for this work is developing methods that are capable of utilizing the nonlinear model in real-time for our simulation. As discussed in the second chapter, we see an increase in accuracy with regard to tracking the trajectory when we use a nonlinear model of the quadcopter for the MPC, but we also see an increase in run time that is too high to run for a real-time simulation. The ability to use this model would potentially allow us to see an increase in tracking performance of the DMPC during nominal and disturbed conditions. Another area of improvement is to test the DMPC control architecture on different platforms and outdoor environments. To achieve this, the hardware will need to be reconfigured for the drone to generate it's MPC trajectory data on-board and communicate with GPS. By utilizing other vehicles, such as larger drones or fixed wind aircraft, and testing in outdoor environments, the control algorithm can be validated in settings that will closer resemble its real-world applications and provide insight on the feasibility of the use of Deep MPC for real world autonomous flight applications.

REFERENCES

[1] R. Amin, L. Aijun, and S. Shamshirband, "A review of quadrotor uav: Control methodologies and performance evaluation," *International Journal of Automation and Control*, vol. 10, no. 2, p. 87, 2016. DOI: 10.1504/ijaac.2016.076453.

[2] M. A. Tofigh, M. J. Mahjoob, and M. Ayati, "Dynamic modeling and nonlinear tracking control of a novel modified quadrotor," *International Journal of Robust and Nonlinear Control*, vol. 28, no. 2, pp. 552–567, 2017. DOI: 10.1002/rnc.3885.

[3] V. S. Juan, M. Santos, and J. M. Andújar, "Intelligent uav map generation and discrete path planning for search and rescue operations," *Complexity*, vol. 2018, pp. 1–17, 2018. DOI: 10.1155/2018/6879419.

[4] R. Szabolcsi, "The quadrotor-based night watchbird uav system used in the force protection tasks," *International conference KNOWLEDGE-BASED ORGANIZATION*, vol. 21, no. 3, pp. 749–755, 2015. DOI: 10.1515/kbo-2015-0126.

[5] R. Grassi, P. Rea, E. Ottaviano, and P. Maggiore, "Application of an inspection robot composed by collaborative terrestrial and aerial modules for an operation in agriculture," in *Advances in Service and Industrial Robotics*, C. Ferraresi and G. Quaglia, Eds., Cham: Springer International Publishing, 2018, pp. 539–546, ISBN: 978-3-319-61276-8.

[6] J. Virdi, *Flight evaluation of deep model reference adaptive control*, May 2020. [Online]. Available: http://hdl.handle.net/2142/108017.

[7] L. Ding and Z. Wang, "A robust control for an aerial robot quadrotor under wind gusts," *Journal of Robotics*, vol. 2018, pp. 1–8, 2018. DOI: 10.1155/2018/5607362.

[8] J. E. Sierra and M. Santos, "Wind and payload disturbance rejection control based on adaptive neural estimators: Application on quadrotors," *Complexity*, vol. 2019, pp. 1–20, 2019. DOI: 10.1155/2019/6460156.

[9] P. Ioannou and J. Sun, "Theory and design of robust direct and indirect adaptive-control schemes," *International Journal of Control*, vol. 47, no. 3, pp. 775–813, 1988. DOI: 10.1080/00207178808906054.

[10] G. Tao, "Adaptive control design and analysis," *Adaptive and Learning Systems for Signal Processing, Communications, and Control*, 2003. DOI: 10.1002/0471459100.

[11] J.-B. Pomet and L. Praly, "Adaptive nonlinear regulation: Estimation from the lyapunov equation," *IEEE Transactions on Automatic Control*, vol. 37, no. 6, pp. 729–740, 1992. DOI: 10.1109/9.256328.

[12] G. Chowdhary, M. Mühlegg, and E. Johnson, "Exponential parameter and tracking error convergence guarantees for adaptive controllers without persistency of excitation," *International Journal of Control*, vol. 87, no. 8, pp. 1583–1603, 2014. DOI: 10.1080/00207179.2014.880128.

[13] R. C. Grande, G. Chowdhary, and J. P. How, "Experimental validation of bayesian nonparametric adaptive control using gaussian processes," *Journal of Aerospace Information Systems*, vol. 11, no. 9, pp. 565–578, 2014. DOI: 10.2514/1.i010190.

[14] G. Chowdhary, E. N. Johnson, R. Chandramohan, M. S. Kimbrell, and A. Calise, "Guidance and control of airplanes under actuator failures and severe structural damage," *Journal of Guidance, Control, and Dynamics*, vol. 36, no. 4, pp. 1093–1104, 2013. DOI: 10.2514/1.58028.

[15] G. Chowdhary, H. A. Kingravi, J. P. How, and P. A. Vela, "Bayesian nonparametric adaptive control of time-varying systems using gaussian processes," *2013 American Control Conference*, 2013. DOI: 10.1109/acc.2013.6580235.

[16] A. Didier, A. Parsi, J. Coulson, and R. Smith, *Robust adaptive model predictive control of quadrotors*, Feb. 2021.

[17] N. Xuan-Mung and S. K. Hong, "Robust adaptive formation control of quadcopters based on a leader–follower approach," *International Journal of Advanced Robotic Systems*, vol. 16, no. 4, p. 172 988 141 986 273, 2019. DOI: 10.1177/1729881419862733.

[18] P. K. Mishra, M. Valverde Gasparino, A. Velasquez, and G. Chowdhary, *Deep model predictive control with stability guarantee*, Apr. 2021.

[19] F. Sabatino, *Quadrotor control: Modeling, nonlinearcontrol design, and simulation*, 2015.

[20] T. Luukkonen, "Modelling and control of quadcopter," *Independent research project in applied mathematics, Espoo*, vol. 22, p. 22, 2011.

[21] S. Albaradie, "Modeling, simulation and implementation of stabilizing controller for a quadcopter," Ph.D. dissertation, May 2020. DOI: 10.13140/RG.2.2.25064.49925.

[22] L. N. Petersen, N. K. Poulsen, H. H. Niemann, C. Utzen, and J. B. Jørgensen, "Comparison of linear and nonlinear model predictive control for optimization of spray dryer operation," *IFAC-PapersOnLine*, vol. 48, no. 23, pp. 218–223, 2015. DOI: 10.1016/j.ifacol.2015.11.286.

[23] J. Rawlings, D. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2017, ISBN: 9780975937730. [Online]. Available: https://books.google.com/books?id=MrJctAEACAAJ.

[24] D. Q. Mayne, E. C. Kerrigan, E. J. van Wyk, and P. Falugi, "Tube-based robust nonlinear model predictive control," *International Journal of Robust and Nonlinear Control*, vol. 21, no. 11, pp. 1341–1353, 2011. DOI: https://doi.org/10.1002/rnc.1758. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rnc.1758. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rnc.1758.

[25] R. M. Sanner and J.-J. E. Slotine, "Gaussian networks for direct adaptive control," *1991 American Control Conference*, 1991. DOI: 10.23919/acc.1991.4791778.

[26] F. Lewis, "Nonlinear network structures for feedback control," *Asian Journal of Control*, vol. 1, no. 4, pp. 205–228, 2008. DOI: 10.1111/j.1934-6093.1999.tb00021.x.

[27] *Parrot mambo fly review - mini quadcopter rc drone for beginners*. [Online]. Available: https://www.aniwaa.com/product/drones/parrot-mambo-fly/.

[28] *Lab manual*. [Online]. Available: https://robotics.illinois.edu/lab-manual/.

[29] G. T. Gowan and M. Valverde, *Deepmpc*, Dec. 2021. [Online]. Available: https://github.com/ggowan95/DeepMPC.

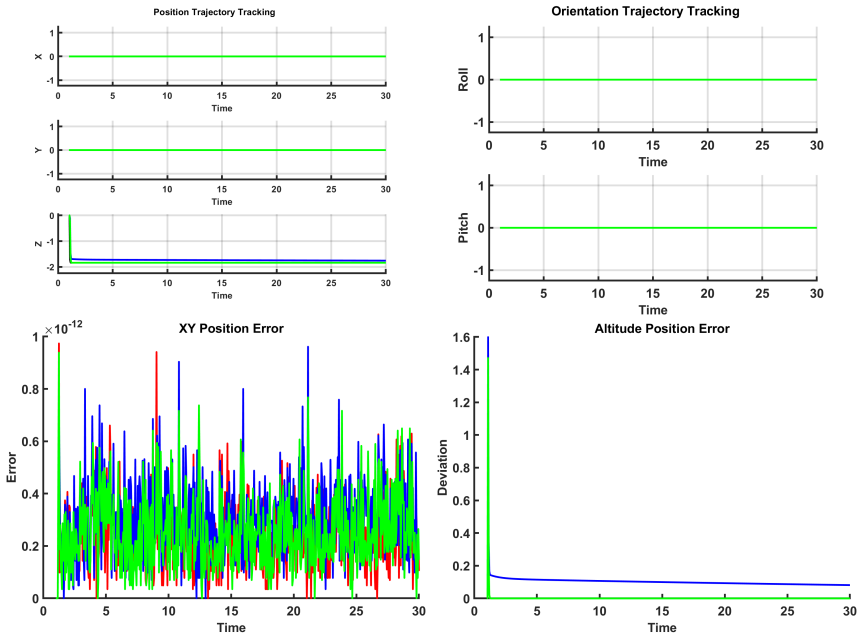
APPENDIX A: ADDITIONAL RESULTS

A Additional Results

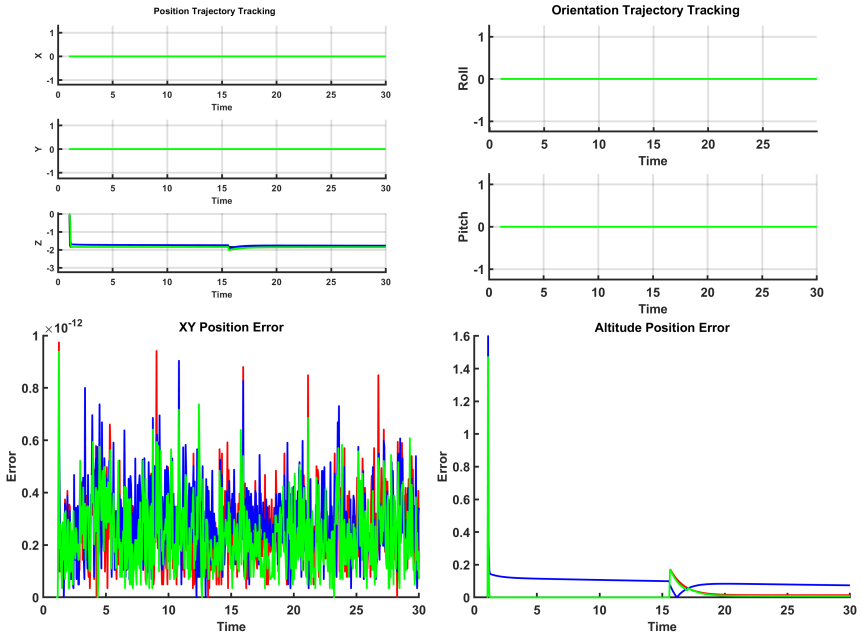
A.1 Hover Maneuver

In this section, we will present the results of a quadrotor that is hovering.

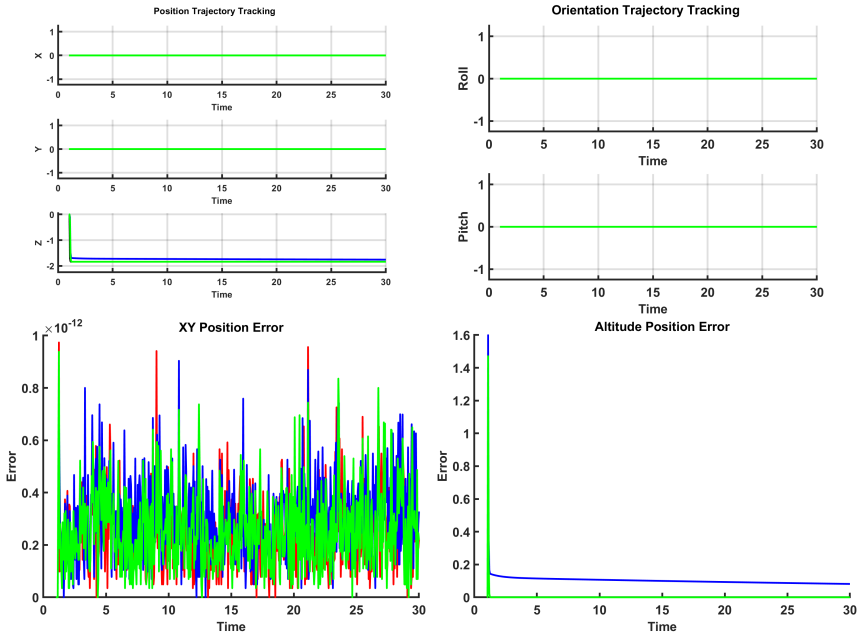
Hover Maneuver: Nominal Conditions



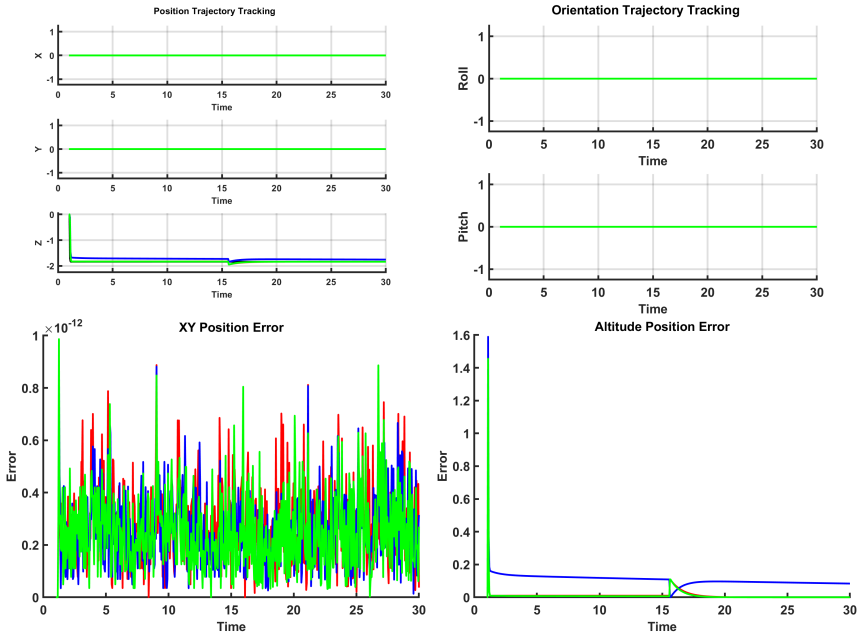
Hover Maneuver: High Wind Conditions



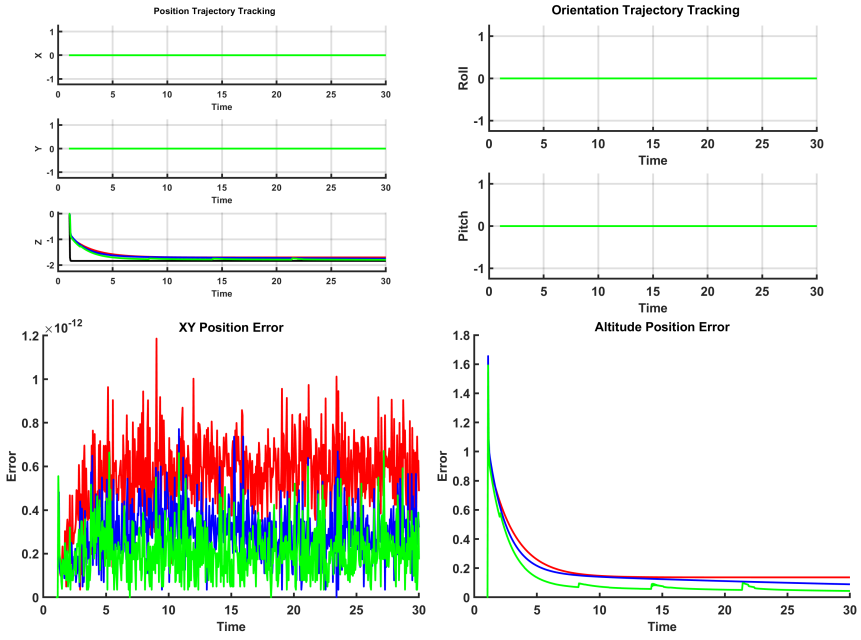
Hover Maneuver: COM Shift



Hover Maneuver: Payload Drop



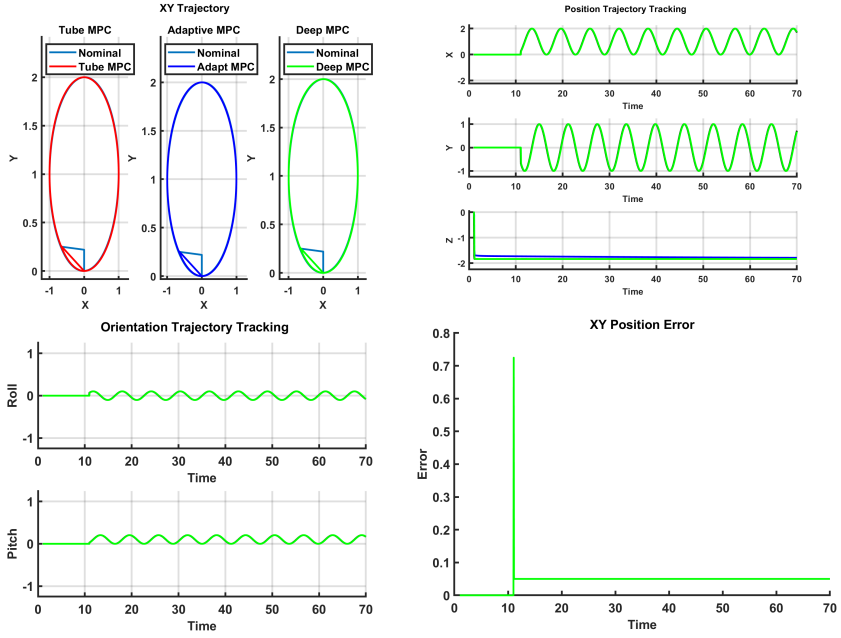
Hover Maneuver: Blade Break



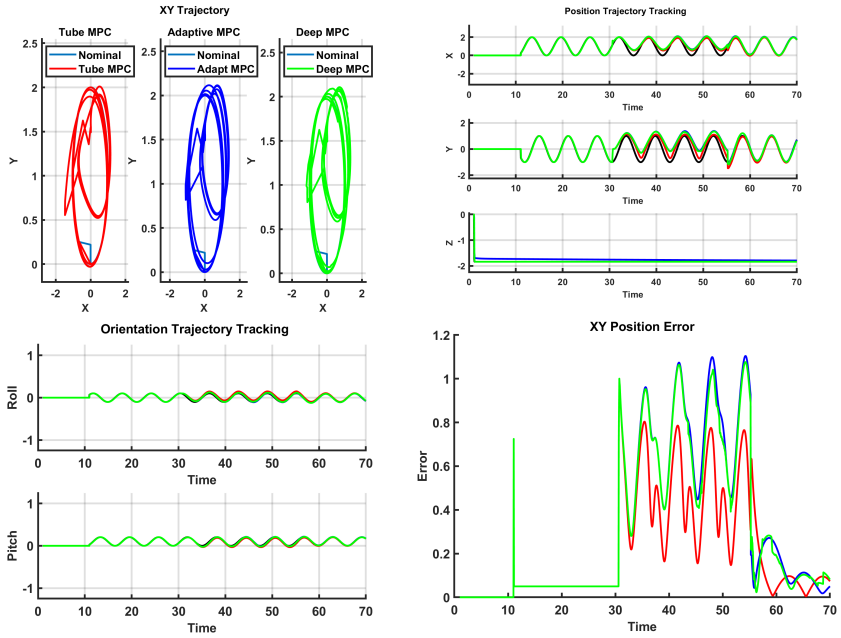
A.2 Circle Maneuver

In this section, we will present the results of a quadrotor that is performing a circle maneuver.

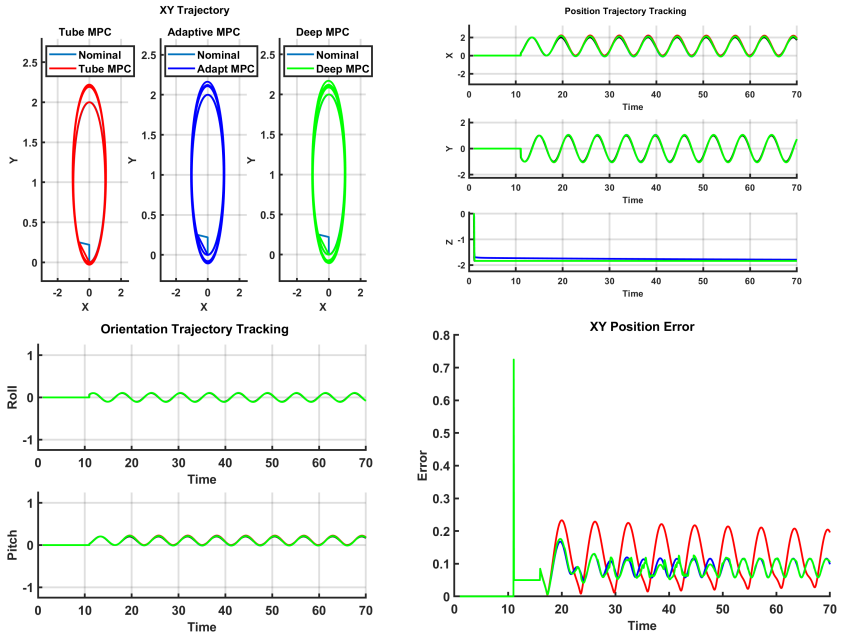
Circle Maneuver: Nominal Conditions



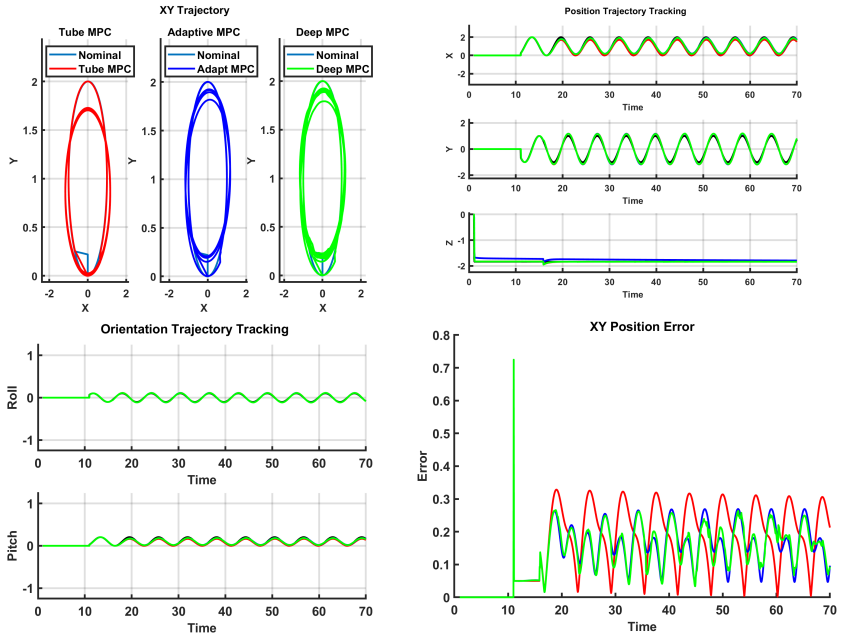
Circle Maneuver: High Wind Conditions



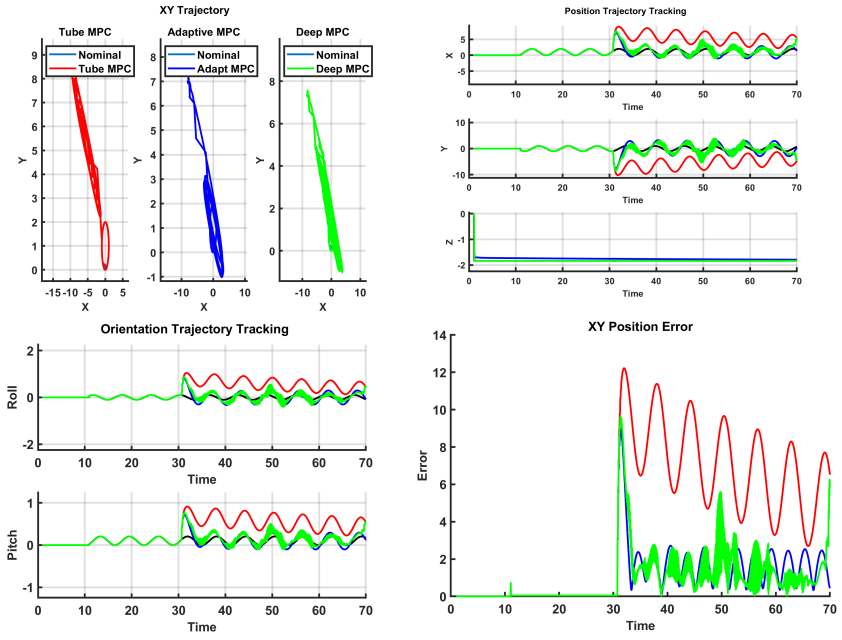
Circle Maneuver: COM Shift



Circle Maneuver: Payload Drop



Circle Maneuver: Blade Break



A.3 Figure-8 Maneuver

In this section, we will present the results of a quadrotor that is performing a Figure-8 maneuver.

Figure-8 Maneuver: Nominal Conditions

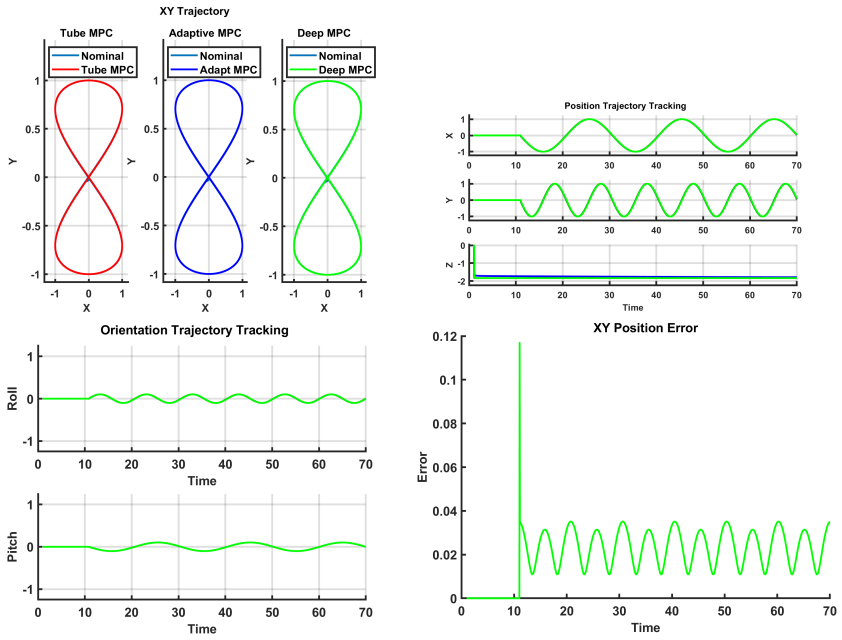


Figure-8 Maneuver: High Wind Conditions

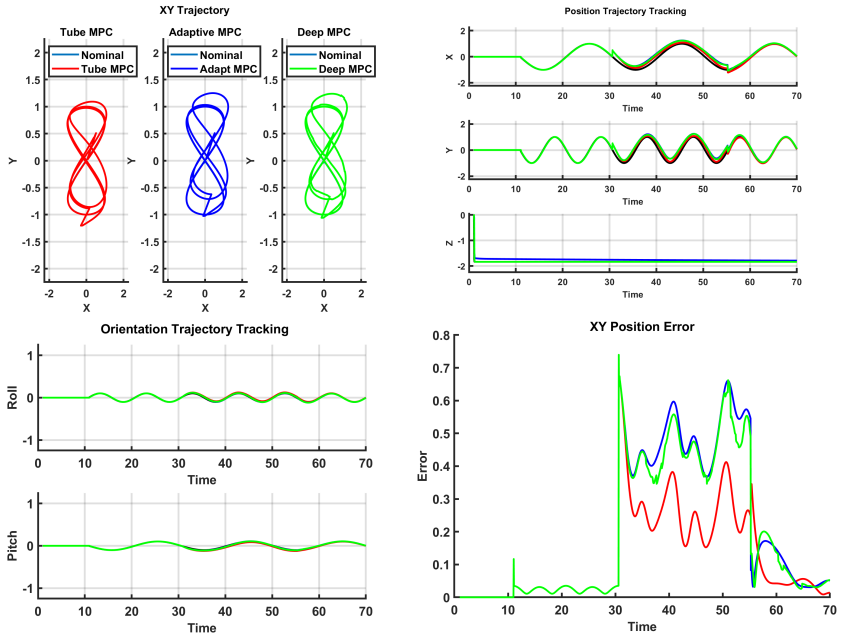


Figure-8 Maneuver: COM Shift

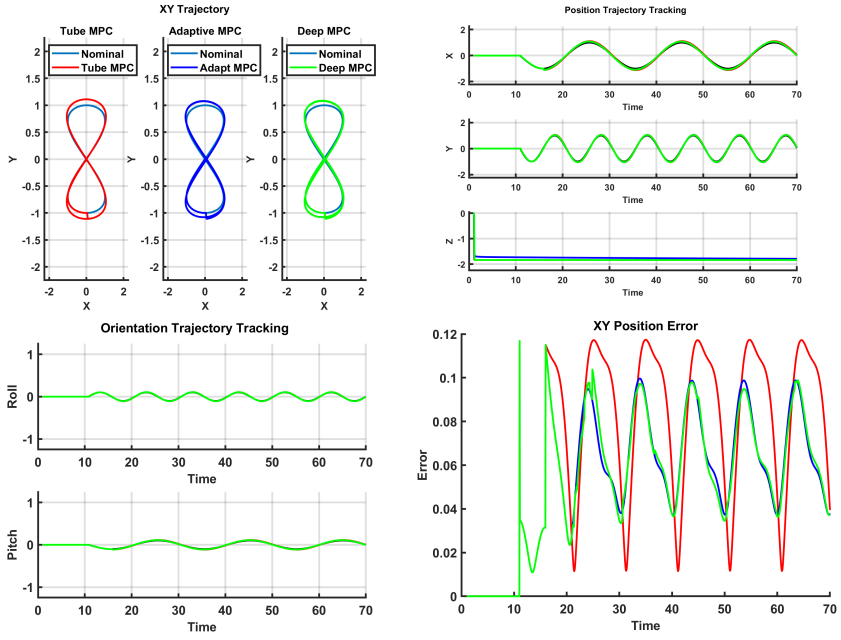


Figure-8 Maneuver: Payload Drop

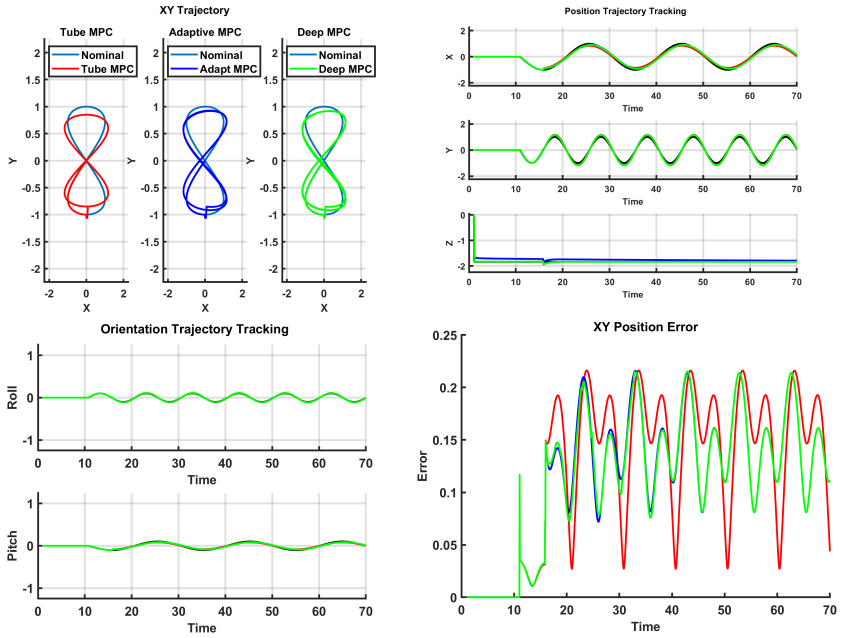


Figure-8 Maneuver: Blade Break