On-board Model Predictive Control of a Quadrotor Helicopter: Design, Implementation, and Experiments

Patrick Bouffard

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2012-241
http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-241.html

December 13, 2012

Acknowledgement

The author gratefully acknowledges the financial support of a Natural Sciences and Engineering Research Council of Canada (NSERC) postgraduate doctoral (PGS-D) scholarship. This work has also been supported in part by NSF under CPS:ActionWebs (CNS-931843), by ONR under the HUNT (N0014-08-0696) and SMARTS (N00014-09-1-1051) MURIs and by grant N00014-12-1-0609, and by AFOSR under the CHASE MURI (FA9550-10-1-0567).

On-board Model Predictive Control of a Quadrotor Helicopter

Design, Implementation, and Experiments

by Patrick Bouffard

Department of Electrical Engineering and Computer Sciences
College of Engineering
University of California, Berkeley

On-board Model Predictive Control of a Quadrotor Helicopter:
Design, Implementation, and Experiments
by Patrick Bouffard

This document was created with the document preparation system LyX.

Abstract

On-board Model Predictive Control of a Quadrotor Helicopter:
Design, Implementation, and Experiments

by Patrick Bouffard

This report describes work in applying model predictive control (MPC) techniques to the control of quadrotor helicopters, a type of micro aerial vehicle (MAV) platform that has gained great popularity in recent years both in research and commercial/military settings. MPC is a form of optimal control which is attractive in part because it allows engineering requirements to be addressed directly in the design of the controller in terms of costs to be minimized and constraints to be satisfied in an optimization problem. Furthermore, for many engineering problems of interest, the optimization to be performed is convex, meaning that a global optimum can be efficiently computed. MPC first found broad early application in the process industry, where the typically longer time scales were compatible with the time necessary to solve the optimization problem. More recently with both the exponential increase in available computing power and the development of more efficient solution techniques, MPC has become an option for control of systems with faster dynamics, such as quadrotors.

We bring together results from our application of two distinct variants of MPC. The common thread is that we seek advanced control algorithms that can be applied to an autonomous MAV like the quadrotor, ideally without requiring any external resources, i.e. we aim to perform all computations required for real-time closed-loop control on-board the vehicle.

The first variant is known as explicit MPC, where in a sense the heavy numerical work of solving optimization problems is done a priori and off-line, such that the on-line implementation requires minimal computation. In this report we describe the design and implementation of three explicit MPC controllers of increasing complexity, and experiments in which these controllers were executed on a quadrotor's on-board computer to control the vehicle in hovering flight. We describe the results of these experiments, with particular emphasis on the resulting performance, in terms of each controller's ability to maintain the quadrotor near a static hover condition.

The second variant is learning-based model predictive control (LBMPC). LBMPC seeks to combine techniques from statistical learning which can help improve performance, with tools and concepts from control theory which provide guarantees about safety, robustness, and convergence. Prior to this work, LBMPC had been implemented in systems quite different from the quadrotor, such as an air-conditioning testbed. Our LBMPC controller for the quadrotor helps demonstrate the formulation's versatility, and some of the particulars of this problem required extensions to LBMPC which we describe. Our main focus here is on demonstrating properties of LBMPC controllers on the quadrotor testbed, with an implementation of LBMPC that runs in real-time on the quadrotor's on-board computer. Robustness to "mis-learning" is one aspect of LBMPC that we demonstrate in an experiment where we deliberately mis-tune a learning algorithm. We also demonstrate the improvement in performance possible when a well-tuned learning algorithm is used, show learning used to update the model in a physically meaningful way, and demonstrate the use of the LBMPC controller in an integrated robotic task requiring speed and precision: we design a controller that enables the quadrotor to catch balls.

Finally, another aspect of MPC research that is important in applying MPC to systems with fast dynamics and limited computation is the development of so-called "fast MPC" methods. These techniques leverage the sparsity pattern that is characteristic of the MPC optimization in dedicated solvers that scale better in the MPC horizon than do general-purpose solvers. However, since LBMPC can be considered a superset of classical MPC, existing fast MPC solvers are not able to solve the optimization problems that arise in LBMPC. We have implemented an LBMPC solver based on an infeasible start interior point method and in simulations demonstrate that its runtime scales linearly in the horizon length. We also show results of experiments where this new solver is run on-board the quadrotor for closed-loop control, and compare its performance in this setting to that of two existing general-purpose active-set solvers.

Acknowledgments

Firstly, I am deeply indebted to my advisor, Professor Claire Tomlin, for accepting me into her research group and providing generous support, insightful advice and kind encouragement throughout my time at Berkeley to date. No matter how discouraged I might at times feel when entering her office for a status meeting, I always leave with renewed energy and determination.

That Professor Ruzena Bajcsy is a remarkable person is well-known to anyone who has crossed her path. I am most thankful to her for agreeing to be Second Reader for this report and I am looking forward to our future collaborations together. I am also thankful to Professor Francesco Borrelli for a providing thorough and proper introduction to MPC in his ME290J class and agreeing to supervise the project on explicit MPC for the quadrotor. I also thank my partner for that project, Cameron Rose.

I greatly enjoyed collaborating with Dr. Anil Aswani on the LBMPC work. Always patient with me when I struggled to grasp some of the theoretical concepts, he also provided immensely useful suggestions on practical implementation of controllers and estimators that will be useful quite beyond the work described herein. I have greatly enjoyed our conversations about a wide range of topics, which we often undertook during long hours in the laboratory while waiting for code to recompile, but also over lunches, dinners, and late-night food runs in the cafes and restaurants along Euclid Avenue. It was also a pleasure working with George (Xiaojing) Zhang to whom most of the credit for the work on the PD IIPM LBMPC solver in Chapter 5 is due.

The Hybrid Systems Lab has always been a place where people are ready to offer keen criticism and advice on any and all topics, academic or otherwise. For this I thank Anil Aswani, Maximilan Balandat, Young-Hwan Chang, Mo Chen, Vera Dadok, Roel Dobbe, Jeremy Gillula, Qie Hu, Haomiao Huang, Sleiman Itani, Maryam Kamgarpour, Eugene Li, Neal Master, Tony Mercer, Selina Pan, Pangun Park, Mac Schwager, Andrew Sy, Ryo Takei, Michael Vitus, Insoon Yang, Melanie Zeilinger, Wei Zhang, and Zhengyuan Zhou.

Without the open-source software generously released by several individuals around the world, much of the implementation and experimental part of the work in this report would have been considerably more difficult. I am therefore thankful to the authors of ROS, the ROS AscTec drivers, as well as those of Linux and an uncountable number of supporting packages for this wonderful operating system.

I would like to gratefully acknowledge the financial support of a Natural Sciences and Engineering Research Council of Canada (NSERC) postgraduate doctoral (PGS-D) scholarship. This work has also been supported in part by NSF under CPS:ActionWebs (CNS-931843), by ONR under the HUNT (N0014-08-0696) and SMARTS (N00014-09-1-1051) MURIs and by grant N00014-12-1-0609, and by AFOSR under the CHASE MURI (FA9550-10-1-0567).

Finally, my deepest and most sincere appreciation goes to those who are dearest to me: My parents Michael and Beth Bouffard have always shown through both words and actions that their love for their children is unconditional and complete; also my sister Danielle, whose hard work and dedication to family are more of an inspiration than she knows. I am grateful to my parents-in-law Roy and Misao Katsuyama, who have never questioned the wisdom of their son-in-law's somewhat unorthodox career detour. Most of all, I thank my darling wife Noriko, whose love and strength have never flagged through the many ups and downs of this journey; and my children, Kiyomi and Keiji, whose smiles upon my return from a too-long day on campus always lift my spirits.

Patrick Bouffard
Berkeley, California
December, 2012

Contents

List of Figures

List of Tables

Chapter 1

Introduction

There has been much recent interest in the use of small unmanned aerial vehicles (UAVs) for security, surveillance/sensor networks (Schwager et al., 2011), and search-and-rescue (Hoffmann, Waslander, and Tomlin, 2006; Michael et al., 2012) applications, and such vehicles have seen use in areas as diverse as recent political upheavals (Austen, 2011) and even as high-tech toys[1]. Due to these applications, and because of their relatively small size, ability to hover, and mechanical simplicity, quadrotor helicopter UAVs are a popular choice among researchers in control and robotics (Hoffmann et al., 2004; How et al., 2008; Michael et al., 2010; Huang et al., 2011a; Lupashin et al., 2011; Meier et al., 2012).

Prior work in our lab has included application of various types of control systems to the quadrotor; for example, in (Waslander et al., 2005), reinforcement learning and integral sliding mode control were investigated for outdoor altitude control of the quadrotor, and PID control of attitude and altitude were examined in (Hoffmann et al., 2007). In (Ding et al., 2011) a reachability-based control synthesis technique was used to control the lateral motion of a quadrotor. In a prior work in vision-based control (Brockers, Bouffard, Ma, Matthies, and Tomlin, 2011), a PID-type controller was used on the same type of quadrotor platform used in the experiments herein. Much work has been focused on higher-level control and autonomous behaviors (Vitus et al., 2008; Hoffmann and Tomlin, 2008; Huang et al., 2011b). However, application of one of the major classes of control techniques, model predictive control (MPC), had not previously been investigated in detail on our platform.

MPC is a technique which poses the control problem as an optimization, where a given cost function is minimized over decision variables that include the values of the control inputs for the current and future (for some finite horizon–for this reason MPC is also known as receding horizon control (RHC)) timesteps, subject to constraints on state and inputs. MPC is therefore able to naturally consider safety considerations and actuator saturation, as long as these can be formulated as state and input constraints. When the cost function is quadratic in the states and inputs, and the constraints affine, the optimization is a quadratic program (QP), and the resulting control law turns out to be piecewise affine (PWA) when parametrized by the initial condition, over a collection of polytopic regions of the state space.

MPC in applications dates to the late 1970's, where it first found use in the process industry and was termed Model Predictive Heuristic Control (MPHC) (Richalet et al., 1976, 1978). In this setting, relatively long time scales (e.g. sampling periods on the order of minutes) meant that the modest and unreliable digital computing capabilities of the day were not a barrier to successful adoption of these new methods. In the intervening decades as computing has become faster and more reliable, the types of systems that MPC can address has broadened, and concurrently new theoretical developments have also made the application of MPC techniques feasible on a larger range of systems.

There has been some prior work in application of MPC to quadrotors. In (Raffo and Ortega, 2008), a method for controlling the quadrotor using a combination of MPC and $\mathcal{H}_\infty$ control was described and tested in simulations. In (Lopes, Ara, and Ishihara, 2011), an MPC controller for a quadrotor was developed and compared in simulations to conventional linear PID control and nonlinear backstepping control. In (Alexis et al., 2011; Alexis, Nikolakopoulos, and Tzes, 2011), a switching (among 3 piecewise-affine (PWA) systems) explicit MPC controller is presented, with experimental results. Finally, in some recent work (Burri et al., 2012) the authors describe experiments with an MPC controller for a quadrotor designed for inspections of a thermal power plant boiler.

[1] Parrot AR.Drone, http://ardrone2.parrot.com/

1.1 Overview

In this report, we bring together results from our application of two distinct variants of MPC for control of the quadrotor. The common thread is that we seek advanced control algorithms that can be applied to an autonomous MAV like the quadrotor, ideally without requiring any external resources, i.e. we aim to perform all computations required for real-time closed-loop control on-board the vehicle. The two variants are known as "explicit MPC" and "learning based MPC".

Explicit MPC

The controller resulting from solving the (quadratic program) MPC optimization problem is PWA over polytopic regions of the initial state (Bemporad et al., 2002). This fact is leveraged by a form of MPC called explicit MPC, in which these PWA control laws and the polytopic regions are pre-computed. The advantage is that the resulting controller can be implemented on-line very efficiently, as the determination of the control input at each timestep amounts to a look-up of the appropriate PWA control law based on which polytopic region contains the current state. This efficiency at run-time comes at a cost–the computation of the polytopic regions scales poorly as the number of states in the system and number of steps in the horizon increase, and can both limit the achievable performance (by limiting the length of horizon that can be handled) and also slow down the task of control design, as any change to the controller parameters (e.g. weighting matrices in the cost function) requires a re-computation of the lookup table, which can require a considerable amount of time, even for systems with relatively small state spaces such as the quadrotor, with a small number of timesteps. We sought to explore the use of explicit MPC for the quadrotor; because of the quadrotor's small payload capacity it carries a relatively modest on-board computer, so a scheme like explicit MPC that is efficient at run-time is attractive in this context. In this report we describe the design and implementation of three explicit MPC controllers of increasing complexity, and experiments in which these controllers were executed on a quadrotor's on-board computer to control the quadrotor in hovering flight. We describe the results of these experiments, with particular emphasis on the resulting performance, in terms of each controller's ability to maintain the quadrotor near a static hover condition.

Learning-based MPC (LBMPC)

A recently developed control technique with roots in MPC, adaptive, and learning-based control is called learning based model predictive control (LBMPC) (Aswani et al., 2012a). It seeks to combine attributes of MPC (most notably, the ability to enforce constraints, which encode safety requirements) with elements of adaptive or learning schemes which promise to improve performance by improving system models based on data obtained on-line. The issue with using the adaptive/learning-based techniques is that, alone, they do not provide any safety guarantees; it is possible in most such techniques to learn an arbitrarily bad model, and controlling based on such a model could lead to compromising safety. LBMPC seeks to reconcile this by using learning to update a model that can improve performance by its incorporation into the cost function, but without compromising safety. The safety property is achieved my maintaining a nominal model, with bounds on the modeling error, and using this bounded-uncertainty nominal model in constraint satisfaction. The result is that even if the learning algorithm fails to improve the model, safety is maintained. This robustness to "mis-learning" is one aspect of LBMPC that we demonstrate with an implementation of LBMPC that runs in real-time on the quadrotor's on-board computer, in an experiment where we deliberately mis-tune a learning algorithm. We also demonstrate the improvement in performance possible when a well-tuned learning algorithm is used, show learning used to update the model in a physically meaningful way, and demonstrate the use of the LBMPC controller in an integrated robotic task requiring speed and precision: we design a controller that enables the quadrotor to catch balls.

Exploiting structure for fast LBMPC solutions

The sparsity structure in MPC problems has been noted by previous authors, e.g. (Wang and Boyd, 2010; Rao, Wright, and Rawlings, 1998). In a short coda, we describe experiments and simulations where a new primal-dual infeasible-start interior-point method (PD IIPM) sparse solver is used for the control of the quadrotor using LBMPC. The results confirm theoretical results on the linearity of the solve time in the LBMPC horizon length. We compare the new solver against two other solvers as well.

1.2 Organization

This report is concerned with the design and implementation of different types of MPC controllers for a quadrotor, and experiments testing the use of these controllers on a real quadrotor in flight, with the controller running entirely on-board the quadrotor. The material presented here has appeared in part in a term project report (Bouffard and Rose, 2011) and in two papers (Bouffard, Aswani, and Tomlin, 2012; Aswani, Bouffard, and Tomlin, 2012). In the following we describe the organization of the present report.

The following section in this chapter describes notation used in this document. Chapter 2 describes dynamical models for the quadrotor helicopter that are relevant to the controllers that we design for it, and also describes how we identified relevant system parameters for these models. Chapter 3 describes our design and implementation of, and experiments with, an explicit MPC controller for the quadrotor. Chapter 4 describes the LBMPC technique and its application to the quadrotor, and design and implementation of an LBMPC controller for the quadrotor. We also describe experiments on this platform which demonstrate some of the important features of LBMPC. Chapter 5 briefly describes the design and implementation of a new solver using "fast MPC" techniques on the LBMPC problem, and focuses on simulation and experimental results including comparison between this new solver and some existing general-purpose solvers. Finally, in Chapter 6 we offer some concluding remarks and directions for future work. We reserve Appendix A for a detailed description of the experimental testbed including the quadrotor itself as well as supporting hardware and software. Appendix B describes some details of the modeling, estimation, and prediction of the trajectories of balls in free flight that are pertinent to one of the LBMPC experiments.

1.3 Notation

Here, we define the notation used in this report. First we will describe some general conventions used, and then each symbol used is listed, along with its dimensionality/set membership (if applicable) and a short description. Acronyms/initialisms are defined where they are first used, but see also page 67 for a complete list of these.

1.3.1 General conventions

Vectors and matrices: Vectors are not typeset specially, but are identified as such when introduced (e.g., $v \in \mathbb{R}^{10}$). All vectors are column vectors, and the transpose of a vector or matrix is denoted with a superscript T (e.g., v^T). The notation $\mathbf{diag}\left\{d_1, \ldots, d_n\right\}$ denotes a diagonal matrix with the bracketed values along the diagonal; similarly, the notation $\mathbf{blkdiag}\left\{D_1, \ldots, D_N\right\}$ denotes a block-diagonal matrix with the bracketed matrices along the diagonal blocks.

Discrete-time indices and difference equations: Variables that change at each discrete timestep have the time index either denoted by the subscript (e.g. v_k), or in square brackets (e.g. $v\left[k\right]$). However, in equations describing the update of such a variable, a superscript $+$ on the variable indexed by time indicates the subsequent time index of the variable. For example, $v^+ = 0.5v + 0.1$ is equivalent to $v_{i+1} = 0.5v_i + 0.1$.

Where a time-indexed variable is included without a subscript, this refers to the value of the variable at the "most recent" timestep in a sense that should be clear from the context.

Quadratic form: The notation $\|v\|_M^2$ denotes the quadratic form $v^T M v$.

Inertial frame axis subscripts: The indices 1, 2, or 3 are subscripted on vectors to denote the component in the corresponding axis of the inertial frame.

Time derivative: Symbols with a dot above are the time derivative of that symbol (e.g., $\dot{x} = \frac{dx}{dt}$). Double dots indicate a second time derivative (e.g. $\ddot{x} = g/m$).

Sets: Sets are denoted by calligraphic capital letters, e.g. $\mathcal{U}$. The Minkowski sum of two sets $\mathcal{U}, \mathcal{W}$ is defined as $\mathcal{U} \oplus \mathcal{W} = \{u + w \mid u \in \mathcal{U}, w \in \mathcal{W}\}$. The $\ominus$ symbol denotes the Pontryagin set difference operator (Kolmanovsky and Gilbert, 1998), e.g. $\mathcal{X} \ominus \mathcal{D} = \{x \mid \{x\} \oplus \mathcal{D} \subseteq \mathcal{X}\}$.

Limiting values: Whenever an underbar or overbar is used it indicates the upper or lower limit resp. for under/overlined quantity. For vector quantities the limits are taken element-wise. For example, $\bar{x}$ is the vector of upper limits of the components of x.

1.3.2 List of symbols

We endeavor here to describe every symbol used in this report. We have tried not to overload notation too much, but where a symbol is overloaded we warn the reader.

Sets

$\mathbb{R}$	The set of all real numbers
$\mathbb{R}_+$	The set of all non-negative real numbers
$\mathbb{R}_{>0}$	The set of all positive real numbers
$\mathbb{R}^n$	The n-dimensional real linear space
$\mathbb{Z}$	The set of all integers
$\mathbb{Z}_+$	The set of all non-negative integers
$\mathbb{N}$	The set of all positive integers
$\mathcal{X} \subseteq \mathbb{R}^n$	set of feasible states
$\mathcal{X}_f \subseteq \mathbb{R}^n$	set of feasible terminal states
$\mathcal{U} \subseteq \mathbb{R}^m$	set of feasible inputs
$\Omega \subseteq \mathcal{X} \times \mathbb{R}^3$	maximal output admissible disturbance invariant set
$\mathcal{D} \subset \mathbb{R}^n$	model uncertainty bounds (per timestep)

Scalars

(elements of $\mathbb{R}$ unless otherwise specified)

δt	discrete-time model timestep
θ	attitude angle (pitch or roll depending on context)
$\theta_0, \theta_1, \theta_2$	yaw, pitch, roll angles respectively
g	acceleration due to gravity
m	(overloaded) mass of quadrotor
$m \in \mathbb{N}$	(overloaded) number of inputs (dimension of input vector), or, current discrete timestep
$n \in \mathbb{N}$	number of states (dimension of state vector)
$p \in \mathbb{N}$	number of outputs (dimension of output vector)
$N \in \mathbb{N}$	MPC prediction horizon
n_0, d_0, d_1	attitude subsystem transfer function polynomial coefficients
T	total thrust
T_x	lateral component of total thrust (in some lateral axis)
T_z	vertical component of total thrust
x_i	i-th component of position of quadrotor in $\mathbf{F}_I$
u_i	i-th input
K_T	thrust coefficient
$k \in \mathbb{Z}$	timestep index, in discrete-time model
t	time, in continuous-time model
δ	tuning parameter in dual EKF
ρ	density of air
C_D	coefficient of air friction

Reference frames

$\mathbf{F}_I$	Inertial reference frame
$\mathbf{F}_B$	Quadrotor body reference frame

Vectors and matrices

$\mathbf{0}, \mathbf{0}_{m \times n}$	The zero matrix (without subscript, dimensions should be clear from context; subscript when present denotes the size)
I, I_n	The identity matrix (without subscripts, dimensions should be clear from context; subscript when present denotes the size)
$\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3$	The coordinate axes of $\mathbf{F}_B$
$\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$	The coordinate axes of $\mathbf{F}_I$
$A_c \in \mathbb{R}^{2 \times 2}$	attitude subsystem continuous-time model dynamics matrix
$A_\theta \in \mathbb{R}^{2 \times 2}$	attitude subsystem discrete-time model dynamics matrix
$A_t \in \mathbb{R}^{2 \times 2}$	lateral translational subsystem discrete-time model dynamics matrix

$A_l \in \mathbb{R}^{4\times 4}$	combined attitude and lateral translation discrete-time model dynamics matrix
$A_8 \in \mathbb{R}^{2\times 2}$	combined attitude and lateral translation (both axes) discrete-time model dynamics matrix
$A_z \in \mathbb{R}^{2\times 2}$	vertical dynamics subsystem discrete-time model dynamics matrix
$A \in \mathbb{R}^{10\times 10}$	combined (10-state) quadrotor dynamics discrete-time model dynamics matrix
$A_b \in \mathbb{R}^{2\times 2}$	ball model dynamics matrix (one axis)
$B_l \in \mathbb{R}^{4\times 1}$	combined attitude and lateral translation discrete-time model input-to-state map
$B_c \in \mathbb{R}^{2\times 1}$	attitude subsystem continuous-time model input-to-state map
$B_\theta \in \mathbb{R}^{2\times 1}$	attitude subsystem discrete-time model input-to-state map
$B_t \in \mathbb{R}^{2\times 1}$	lateral translational subsystem discrete-time model input-to-state map
$B_8 \in \mathbb{R}^{8\times 2}$	combined attitude and lateral translation (both axes) discrete-time model input-to-state map
$B_z \in \mathbb{R}^{2\times 1}$	vertical dynamics subsystem discrete-time model input-to-state map
$B \in \mathbb{R}^{10\times 3}$	combined (10-state) quadrotor dynamics discrete-time model input-to-state map
$C_\theta \in \mathbb{R}^{1\times 2}$	attitude subsystem state-to-output map (discrete- or continuous-time model)
$C_t \in \mathbb{R}^{1\times 2}$	lateral translational subsystem discrete-time model state-to-output map
$C \in \mathbb{R}^{5\times 10}$	combined (10-state) quadrotor dynamics discrete-time model state-to-output map
$C_b \in \mathbb{R}^{3\times 6}$	ball model state-to-output map
$D \in \mathbb{R}^{p\times m}$	input-to-output map (feed-through)
$q_i \in \mathbb{R}^4$	state vector of 4-state combined lateral and attitude model, $i = 1, 2$
$q_3 \in \mathbb{R}^2$	state vector of vertical dynamics subsystem model
$x_l \in \mathbb{R}^8$	state vector of 8-state combined lateral and attitude discrete-time model for both lateral axes
$u_l \in \mathbb{R}^2$	input vector of 8-state combined lateral and attitude discrete-time model for both lateral axes
$k_z \in \mathbb{R}^{2\times 1}$	affine term in vertical dynamics subsystem model
$k_l \in \mathbb{R}^{4\times 1}$	affine term in 8-state combined lateral and attitude discrete-time model for both lateral axes
$x \in \mathbb{R}^{10}$	combined (10-state) quadrotor dynamics discrete-time model state vector
$y \in \mathbb{R}^5$	combined (10-state) quadrotor dynamics discrete-time model output vector
$\epsilon \in \mathbb{R}^5$	measurement noise in combined (10-state) quadrotor dynamics discrete-time model output equation
$P_N \in \mathbb{R}^{n\times n}$	final state cost weighting matrix in MPC cost function
$Q \in \mathbb{R}^{n\times n}$	stage state cost weighting matrix in MPC cost function
$R \in \mathbb{R}^{m\times m}$	stage input cost weighting matrix in MPC cost function
$\delta u \in \mathbb{R}^m$	differential input vector in δu-formulation MPC model
$r \in \mathbb{R}^p$	reference output in δu-formulation MPC model
$\hat{x} \in \mathbb{R}^{10}$	estimated quadrotor state
$\tilde{x} \in \mathbb{R}^{10}$	quadrotor state predicted with learned model
$\bar{x} \in \mathbb{R}^{10}$	quadrotor state predicted with nominal model

$x \in \mathbb{R}^{10}$	actual quadrotor state vector
$x_b \in \mathbb{R}^6$	actual ball state vector
$\hat{x}_b \in \mathbb{R}^6$	estimated ball state vector
$\hat{u} \in \mathbb{R}^n$	estimated input
$U_0 \in \mathbb{R}^{mn}$	stacked vector of all input vectors over MPC horizon
$w \in \mathbb{R}^4$	state noise vector (explicit MPC model for one axis)
$F(\beta) \in \mathbb{R}^{10\times 10}$	oracle updates to dynamics matrix
$H(\beta) \in \mathbb{R}^{10\times 3}$	oracle updates to input-to-state map
$z(\beta) \in \mathbb{R}^{10}$	oracle updates to affine term of dynamics model
$K \in \mathbb{R}^{3\times 10}$	nominal feedback gain
$\bar{K} \in \mathbb{R}^{3\times 10}$	feedback gain used for computations of terminal set
$\hat{K} \in \mathbb{R}^{10\times 5}$	feedback gain matrix used in dual EKF state estimate update
$K_b \in \mathbb{R}^{6\times 3}$	feedback gain matrix used in ball observer state estimate update
$\beta \in \mathbb{R}^{12}$	vector of true parameters in oracle model
$\hat{\beta} \in \mathbb{R}^{12}$	vector of estimated parameters in oracle model
$\xi \in \mathbb{R}^3$	vector parametrizing points that can be feasibly tracked with a linear controller
$\breve{u} \in \mathbb{R}^3$	input vector in LBMPC optimization
$c^* \in \mathbb{R}^3$	optimal input
$y_s \in \mathbb{R}^5$	steady-state output
$x_s \in \mathbb{R}^{10}$	steady-state state
$u_s \in \mathbb{R}^3$	steady-state input
$\Lambda \in \mathbb{R}^{10\times 3}$	subspace projection of ξ into state space
$\Psi \in \mathbb{R}^{3\times 3}$	subspace projection of ξ into input space
$\Pi \in \mathbb{R}^{5\times 3}$	subspace projection of ξ into output space
$x_0 \in \mathbb{R}^{10}$	steady-state affine subspace offset vector in state space
$u_0 \in \mathbb{R}^3$	steady-state affine subspace offset vector in input space
$y_0 \in \mathbb{R}^5$	steady-state affine subspace offset vector in output space
$\mu \in \mathbb{R}^{12}$	parameter noise vector
$P_2 \in \mathbb{R}^{10\times 12}$	cross-covariance matrix between state and parameter estimates
$P_3 \in \mathbb{R}^{12\times 12}$	covariance matrix of parameter estimate
$M \in \mathbb{R}^{10\times 12}$	matrix of partial derivatives of oracle update equation with respect to parameters
$\Xi \in \mathbb{R}^{5\times 5}$	covariance matrix of measurement noise
$\Upsilon \in \mathbb{R}^{12\times 12}$	covariance matrix of parameter process noise
$L \in \mathbb{R}^{12\times 5}$	feedback gain in dual EKF parameter update equation
$\bar{x}_{\mathcal{D}} \in \mathbb{R}^{10}$	bounds on state uncertainty
$F_G \in \mathbb{R}^3$	force due to gravity
$F_D \in \mathbb{R}^3$	force due to air drag
$V \in \mathbb{R}^3$	instantaneous velocity vector of ball
$x_b \in \mathbb{R}^6$	state vector of ball
$\hat{x}_c \in \mathbb{R}^{10}$	predicted intercept of ball trajectory with plane at catching altitude

Other

$G(s)$ attitude subsystem input-output transfer function

$J^*(x) : \mathbb{R}^n \to \mathbb{R}$ optimal value of cost function in mp-QP for given state x

$J(z, x) : \mathbb{R}^s \times \mathbb{R}^n \to \mathbb{R}$ cost function in mp-QP for given vector of decision variables z and state vector x

$J_0^*(x(0)) : \mathbb{R}^n \to \mathbb{R}$ optimal value of cost function in CFTOC mp-QP for given initial state $x(0)$

$J_0(x(0), U_0) : \mathbb{R}^n \times \mathbb{R}^{mN} \to \mathbb{R}$ cost function in mp-QP for given vector of decision variables z and state vector x

$h(x, u) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ unmodeled dynamics in system model

$\mathcal{F}_{\mathcal{N}}(x, u) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ LBMPC nominal dynamics model (difference equation)

$\mathcal{O}_m(x, u) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ oracle updates to model at timestep m

Chapter 2

Quadrotor Dynamics Model

2.1 Overview

In this chapter we first briefly describe the theory of operation of quadrotor helicopters. We then describe the models used for the controller design in later chapters, and conclude with a description of the system identification that was performed to obtain estimates of the key parameters of these models.

2.2 Quadrotor theory of operation

The basic principle of operation of a quadrotor helicopter consists in the generation of net force and torque through variation of the rotational speeds of the four rotors. Detailed treatment of the dynamics of quadrotor motion can be found in, e.g., (Hoffmann et al., 2011). Here, we assume a simplified model that is suitable for an operating regime around steady hover, and we summarize in this section the aspects of quadrotor operation that are relevant to near-hover flight.

Quadrotors are so named because of their characteristic set of four identical rotors. We will assume here only fixed-pitch rotors, although variable-pitch quadrotors are being investigated by some researchers (Cutler, 2012). The rotors are the only actuators in this system, and at each instant each will exert a force and moment on the quadrotor's airframe. The rotors are set up in counter-rotating pairs as shown in Fig. 2.1, such that when they are all rotating at the same speed, the moments exactly cancel. Further, when the motors rotate at a particular equal speed, and the plane of the rotors is perpendicular to the vertical, sufficient overall thrust is produced so that the vehicle neither gains nor loses altitude. If the quadrotor is also at zero velocity relative to some inertial frame then this is considered to be the *hover condition* of the quadrotor.

The quadrotor's position and orientation in space can be modified from the hover condition by varying the speeds of the motors from their hover speed. To induce an angular acceleration about the quadrotor's yaw axis (the axis coincident with vertical, when the quadrotor is level with the ground), rotors 1 and 3 (with reference to Fig. 2.1 for the rotor numbering) would together increase/decrease their rate while rotors 2 and 4 decrease/increase their rate. Angular accelerations about roll and pitch axes (the axes labeled $\mathbf{b}_1$ and $\mathbf{b}_2$ in Fig. 2.1 resp.) are induced when one rotors on the alternate axis increases its speed while the other decreases. For example, to induce a positive pitch acceleration, rotor 1 would increase in speed while rotor 3 would decrease. Lateral accelerations are induced whenever roll and/or pitch are nonzero (they are both zero when the quadrotor is level with the ground) by virtue of the overall net thrust vector of the quadrotor (i.e. the sum of the four thrust vectors, one from each rotor) having nonzero components in the horizontal plane when this is the case. To increase or decrease altitude requires a nonzero net force in the vertical axis; this is achieved by simultaneously increasing or decreasing the speeds of all four rotors by the same amount. These translational accelerations will be the ones we are most interested in, and because they are coupled to the angular rotation, this coupling is an important consideration in any dynamical model of the quadrotor.

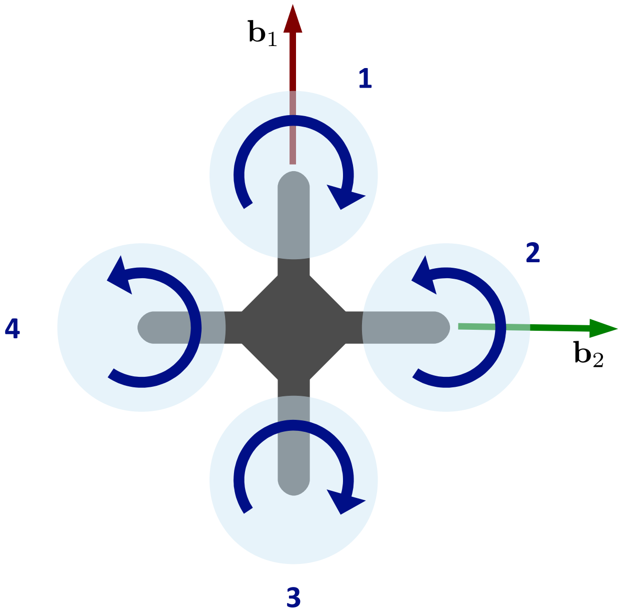


Figure 2.1: Quadrotor rotor numbering convention and sense of rotation of rotors. View is from above, looking down at the quadrotor. Two of the body-fixed axes ($\mathbf{b}_1$ and $\mathbf{b}_2$) are also shown.

2.3 Quadrotor models for near-hover operation

In this section we describe in detail the specific models for the quadrotor that are used in later chapters. Quadrotors, like all rotorcraft, are subject to a number of important aerodynamic effects, beyond those that are behind its basic principle of operation: each motor develops a force and moment that are roughly proportional to the motor's speed, as the attached rotor effects momentum change of the surrounding air. However, we have in mind applications (e.g. mobile sensor networks, surveillance) in which the translational position and velocity of the quadrotor are of primary interest, while the vehicle's attitude is secondary. By this we mean that changes in pitch and roll attitude serve to induce lateral translational accelerations to change the vehicle's translational position and velocity, but we usually are only interested in changing pitch and roll as a means to an end (that of inducing lateral accelerations).

In all the models considered here, we treat the yaw angle as fixed; equivalently yaw is handled by a completely separate controller and the models we describe here can be considered to be transformed such that "pitch" and "roll" are interpreted as angles in a body frame unaffected by the yaw angle, i.e. always oriented such that the body axis $\mathbf{b}_1$ coincides with inertial axis $\mathbf{x}_1$, and body axis $\mathbf{b}_2$ with inertial axis $\mathbf{x}_2$ (with reference to Fig. 2.2).

In general, we start by imagining the quadrotor as a free body subject to two forces: that of gravity, and the total thrust from the rotors, which is taken to act in a direction opposite to the body z axis. We can consider the motion in each of the two perpendicular vertical planes (the $\mathbf{x}_1 - \mathbf{x}_3$ plane and the $\mathbf{x}_2 - \mathbf{x}_3$ plane) separately; since gravity always acts in the same direction, the only variation in the forces relates to the direction and magnitude of the thrust vector. However, to completely decouple the axes, and obtain a linear model, we will assume when considering the lateral (horizontal) dynamics that the vertical dynamics are in static equilibrium, and vice-versa. This fixes the magnitude of the thrust vector (since it must be that which coincides with zero vertical acceleration) and the only remaining variable is the direction of the thrust in each vertical plane, or equivalently, the angle between the (projected in a given plane) thrust vector and the vertical.

Now the dynamics of this angle (call it θ) are due to moments generated about an axis perpendicular to the plane by unequal thrust on either side of the vehicle. Conceptually it is perhaps easiest to think

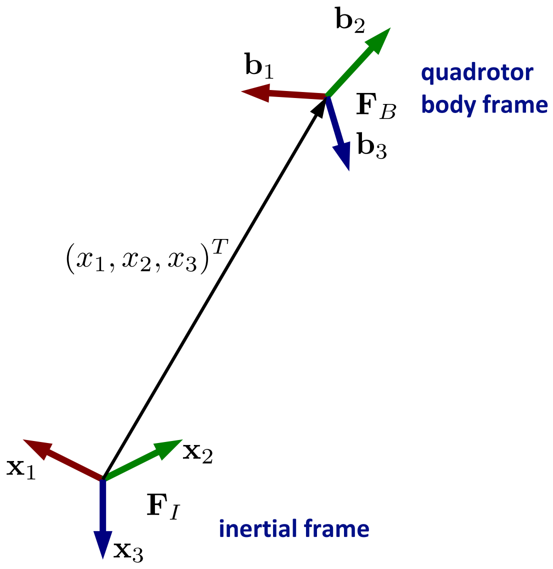


Figure 2.2: Relationship between inertial frame $\mathbf{F}_I$ and quadrotor body frame $\mathbf{F}_B$.

of this as the difference in thrust between opposing pairs of rotors, though in general the plane in consideration may be such that a mixture of rotors is in play. At any rate, it suffices to note that the thrust from a given rotor is effectively proportional to its rate of rotation; indeed the whole control task at the low level for a quadrotor consists in judicious modulation of the speeds of each of the four rotors to obtain a desired motion of the vehicle.

For our purposes though, we will delve no further into the physics behind the attitude dynamics. The main reason for this is a practical one: the actual system that we will perform experiments on, a quadrotor based on the Ascending Technologies (AscTec) Pelican, is equipped with closed-loop attitude regulation "out of the box". That is, it accepts desired pitch and roll angles and implements a feedback controller to drive the error between commanded and actual angles towards zero. The exact structure of this controller is in fact not known to us; it is implemented in proprietary code on the "low level" (LL) microcontroller on the Pelican (see Appendix A). AscTec does provide a utility that allows the user to modify some control gains, in particular proportional (P) and derivative (D) gains for the roll and pitch axes can be selected. So we assume that the attitude controller is some type of PD based controller, though we don't know the specifics. This leads us towards system identification, which we will discuss in more detail below. But here it is worth pointing out the main conclusion we have drawn about the attitude controller based on our testing: it seems it may be reasonably well-modeled by a second-order transfer function.

Ultimately what we will obtain is a model in which each lateral axis has a rotational subsystem that describes the dynamics of the angle θ, and this angle is the input to a translational subsystem which is simply a double integrator; as we will see the angle θ (in radians) is effectively identically the acceleration in that lateral axis, in g's. That is, if $\theta = 0.1\,\mathrm{rad}$, then the corresponding translational axis will undergo an acceleration of $0.1\,\mathrm{g} = 0.981\,\mathrm{m/s^2}$. The vertical dynamics model will be even simpler: just a double integrator with a constant force of mg in the (inertial frame) $+z$ direction, and a controlled input of $0 \leq T \leq \bar{T}$ in the $-z$ direction (where $\bar{T}$ is the maximum thrust of the quadrotor).

In the following subsections we develop these ideas to show the specific form of discrete-time, linear time-invariant (LTI) models. We start by modeling the attitude subsystem, next the lateral translational subsystem, then the combined attitude and lateral dynamics for one lateral direction, then we form a model combining two attitude/lateral systems together to consider both lateral axes in one model, then a model of the dynamics of the vertical subsystem, and finally we put the last two models together for a full model of the translational dynamics of the quadrotor, including the pitch/roll attitude subsystems.

2.3.1 Preliminaries

The quadrotor's position and orientation are expressed in terms of a body-fixed frame with axes $\mathbf{F}_B := \{\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3\}$, with respect to an inertial frame with axes $\mathbf{F}_I := \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}$. Define the state of the system $x = (x_1, \dot{x}_1, \theta_1, \dot{\theta}_1, x_2, \dot{x}_2, \theta_2, \dot{\theta}_2, x_3, \dot{x}_3)^T \in \mathbb{R}^{10}$, where x_1, x_2, x_3 are the components of the vector from $\mathbf{F}_I$ to $\mathbf{F}_B$, expressed in $\mathbf{F}_I$, and $\theta_0, \theta_1, \theta_2$ are the rotations (in radians) in a 3-2-1 (yaw-pitch-roll) rotating axes Euler sequence taking $\mathbf{F}_I$ to $\mathbf{F}_B$. In the present work, we assume that θ_0 (yaw) is held fixed which is why we do not include it in the state vector. We already illustrated the inertial and quadrotor body frames in Fig. 2.2, and Fig. 2.3 illustrates the axes of rotation of the quadrotor.

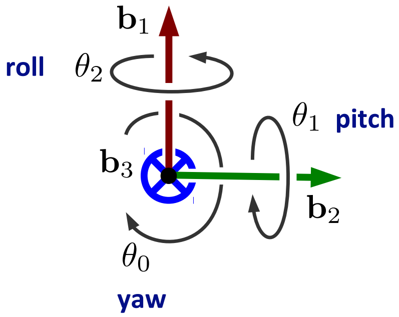


Figure 2.3: Axes of rotation: θ_0 (yaw) about $\mathbf{b}_3$ axis (going into the page); θ_1 (pitch) about $\mathbf{b}_2$ axis; and θ_2 (roll) about $\mathbf{b}_1$ axis.

2.3.2 Attitude Dynamics

We assume that the closed-loop attitude dynamics can be approximated by a second-order system (i.e. one that can be thought of as a torsional inertia-spring-damper). This is based on observations from experiments with step inputs which we detail below. We will assume that the pitch and roll dynamics are decoupled and identical to one another; in the following for ease of presentation we will simply refer to the angular dynamics of the pitch axis.

We therefore assume an input-output (where input is desired pitch angle θ_r and output is actual pitch angle θ) transfer function model of the form

$$G(s) = \frac{n_0}{s^2 + d_1 s + d_0}. \tag{2.1}$$

Note that this corresponds to the differential equation

$$\frac{1}{n_0}\ddot{\theta}(t) = -\frac{d_1}{n_0}\dot{\theta}(t) - \frac{d_0}{n_0}\theta(t).$$

An equivalent continuous-time state-space single-input single-output (SISO) model (in observable canonical form) is

$$\begin{aligned} \frac{d}{dt}x(t) &= A_c x(t) + B_c u(t) && (2.2)\\ &= \begin{bmatrix} -d_1 & 1 \\ -d_0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ n_0 \end{bmatrix} u(t), && (2.3)\\ y(t) &= \begin{bmatrix} 1 & 0 \end{bmatrix} x(t), \end{aligned}$$

where $x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$ is the state, the input $u(t) = \theta_r(t)$ (reference input), and the output $y(t) = \theta(t)$. Finally, for implementation in our MPC framework, we convert this to a discrete-time representation , with sample time δt , using the exact discretization (assuming zero-order-hold of the input, i.e. $u(t+s) = u(t)$ for $s \in [0, \delta t]$)

$$A_\theta = e^{A_c \delta t}, \quad B_\theta = \int_0^{\delta_t} e^{A_c \tau} d\tau B_c \quad . \tag{2.4}$$

yielding a difference equation discrete-time linear, time-invariant (LTI) model,

$$\begin{aligned} x_\theta^+ &= A_\theta x_\theta + B_\theta u_\theta \\ \theta = y_\theta &= C_\theta x_\theta . \end{aligned} \tag{2.5}$$

2.3.3 Lateral translational dynamics

We again assume identical, decoupled axes for the lateral motion. We will assume a point mass model and neglect air drag. The lateral (horizontal) acceleration is due to the horizontal component T_x of the quadrotor's total thrust T, which always acts opposite the quadrotor $\mathbf{b}_3$ axis. In the vertical direction, the acceleration is due to the net force considering the vertical component T_z of the quadrotor's thrust and the force due to gravity mg. Fig. 2.4 illustrates this. We assume that the timescale of the thrust dynamics is sufficiently fast that we can consider the vertical dynamics to be in equilibrium, thus the vertical component of the thrust $T_z = T\cos\theta = mg$, so $T = \frac{mg}{\cos\theta}$ and so the lateral thrust for some given θ is $T_x = mg\frac{\sin\theta}{\cos\theta}$. With $x(t)$ here representing the lateral position in a particular axis the model has the form,

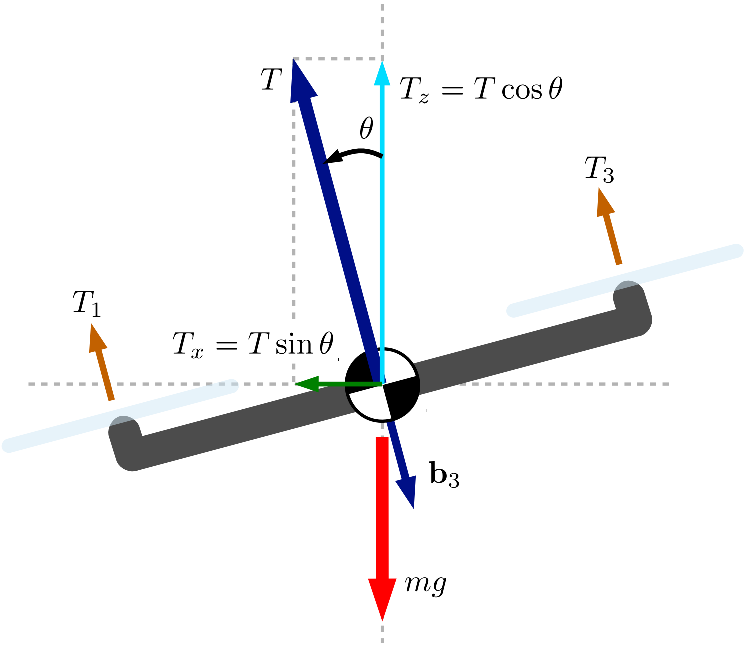


Figure 2.4: Free body diagram for quadrotor lateral translational dynamics. Lateral acceleration is achieved by developing a lateral force T_x by changing the roll or pitch attitude angle θ.

$$\begin{aligned} m\ddot{x}(t) &= mg\frac{\sin\theta}{\cos\theta} \\ \ddot{x}(t) &= g\tan\theta \end{aligned}$$

We linearize (i.e. small angles assumption, $\tan\theta \approx \theta$ near $\theta = 0$). The linearized model is thus,

$$\ddot{x}(t) = g\theta(t).$$

The corresponding transfer function is

$$G(s) = \frac{g}{s^2},$$

which yields a continuous-time observer canonical LTI realization ,

$$\begin{aligned} \frac{d}{dt}x(t) &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ g \end{bmatrix} u(t), \\ y(t) &= \begin{bmatrix} 1 & 0 \end{bmatrix} x(t), \end{aligned} \tag{2.6}$$

and a corresponding discrete-time LTI model,

$$\begin{aligned} x_t[k+1] &= A_t x[k] + B_t u_t[k] \\ y_t[k] &= C_t x_t[k] \end{aligned} \tag{2.7}$$

where

$$A_t = \begin{bmatrix} 1 & \delta t \\ 0 & 1 \end{bmatrix}, \quad B_t = \begin{bmatrix} \frac{\delta t^2}{2} \\ \delta t \end{bmatrix} g, \quad C_t = \begin{bmatrix} 1 & 0 \end{bmatrix},$$

using the same exact discretization of (2.4).

2.3.4 Combined attitude and lateral translational dynamics

Here we describe combining the SISO attitude dynamics, in which the input is the commanded angle and the output is the actual angle, with the translational dynamics, in which the input is essentially the actual angle in radians (for the system linearized about the hover condition). Since the input u_t to the translational dynamics system (2.7) is the output y_θ of the attitude system (2.5), we have

$$B_t u_t = B_t C_\theta x_\theta = g \begin{bmatrix} \frac{\delta t^2}{2} \\ \delta t \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} x_\theta = \begin{bmatrix} \frac{g\delta t^2}{2} & 0 \\ g\delta t & 0 \end{bmatrix} x_\theta.$$

4-state model for one lateral axis

We now combine the translational and attitude dynamics for a given axis (i.e. roll ϕ and +y, pitch θ and -x) to form a 4-state linear time-invariant discrete-time model for that axis:

$$\begin{aligned} q_i^+ &= \begin{bmatrix} x_i \\ \dot{x}_i \\ \theta_i \\ \dot{\theta}_i \end{bmatrix}^+ = A_l q_i + B_l u_i \\ &= \begin{bmatrix} A_t & B_t C_\theta \\ 0 & A_\theta \end{bmatrix} q_i + \begin{bmatrix} 0 \\ B_\theta \end{bmatrix} u_i \end{aligned} \tag{2.8}$$

For clarity, here $q_i \in \mathbb{R}^4$, $x_t, x_\theta \in \mathbb{R}^2$, $u_i \in \mathbb{R}$, $A_l \in \mathbb{R}^{4\times4}$, $B_l \in \mathbb{R}^{4\times1}$, $A_t, A_\theta \in \mathbb{R}^{2\times2}$, $B_\theta \in \mathbb{R}^{2\times1}$. The index $i \in \{1, 2\}$ denotes which of the two lateral axes is modeled.

8-state model for both lateral axes

Further, it is trivial to then concatenate two copies of (2.8) block-wise, to derive an 8-state model which models the dynamics of both lateral axes simultaneously:

$$\begin{aligned}
x_l^+ &= \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}^+ = A_8 x_l + B_8 u_l \\
&= \begin{bmatrix} A_l & 0 \\ 0 & A_l \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} + \begin{bmatrix} B_l & 0 \\ 0 & B_l \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \\
&= \begin{bmatrix} A_t & B_t C_\theta & 0 & 0 \\ 0 & A_\theta & 0 & 0 \\ 0 & 0 & A_t & B_t C_\theta \\ 0 & 0 & 0 & A_\theta \end{bmatrix} x + \begin{bmatrix} 0 & 0 \\ B_\theta & 0 \\ 0 & 0 \\ 0 & B_\theta \end{bmatrix} u
\end{aligned} \tag{2.9}$$

For clarity, here $x_l = \left[q_1^T, q_2^T\right]^T \in \mathbb{R}^8$, $q_1, q_2 \in \mathbb{R}^4$, $u_l = [u_1, u_2]^T \in \mathbb{R}^2$, $A_8 \in \mathbb{R}^{8\times 8}$, $B_8 \in \mathbb{R}^{8\times 1}$, $A_l \in \mathbb{R}^{4\times 4}$, $B_l \in \mathbb{R}^{4\times 1}$, $A_t, A_\theta \in \mathbb{R}^{2\times 2}$, $B_\theta \in \mathbb{R}^{2\times 1}$.

2.3.5 Vertical translational dynamics

The vertical dynamics have no rotational component and can be written in discrete-time for timestep δt as

$$q_3^+ = A_z q_3 + B_z u_3 + k_z, \tag{2.10}$$

where $q_3 = [x, \dot{x}_3]^T$, $A_z = \left[\begin{smallmatrix} 1 & \delta t \\ 0 & 1 \end{smallmatrix}\right]$, $B_z = -\frac{K_T}{m}\left[\frac{\delta t^2}{2}, \delta t\right]^T$ and $K_T > 0$ is a thrust-to-command ratio to be determined empirically, the input u_3 is the commanded thrust, and

$$k_z = \frac{g}{m} \begin{bmatrix} \frac{\delta t^2}{2} \\ \delta t \end{bmatrix} \tag{2.11}$$

represents the acceleration due to gravity.

2.3.6 Combined lateral and vertical dynamics model

Finally, we describe the form of an overall model of both the lateral (horizontal, parallel to the ground) and vertical dynamics of the quadrotor. The combined model has a total of 10 states; 3 translational positions, 2 angles, and the derivatives of each. Since we intend to use this model for MPC, we will only consider a discrete-time model for a given discrete time step δt. To form this model, we simply concatenate block-wise the 8-state model for the lateral axes (2.9) with the 2-state model for the vertical dynamics (2.10) to obtain

$$
\begin{aligned}
x^{+} &= Ax + Bu + k \qquad (2.12)\\
&= \begin{bmatrix} A_8 & 0 \\ 0 & A_z \end{bmatrix} \begin{bmatrix} x_l \\ q_3 \end{bmatrix} + \begin{bmatrix} B_8 & 0 \\ 0 & B_z \end{bmatrix} \begin{bmatrix} u_l \\ u_3 \end{bmatrix} + \begin{bmatrix} k_l \\ k_z \end{bmatrix}\\
&= \begin{bmatrix} A_t & B_t C_\theta & 0 & 0 & 0 \\ 0 & A_\theta & 0 & 0 & 0 \\ 0 & 0 & A_t & B_t C_\theta & 0 \\ 0 & 0 & 0 & A_\theta & 0 \\ 0 & 0 & 0 & 0 & A_z \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} + \begin{bmatrix} B_l & 0 & 0 \\ 0 & B_l & 0 \\ 0 & 0 & B_z \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ k_z \end{bmatrix}\\
&= \begin{bmatrix}
1 & \delta t & \frac{g\delta t^2}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & g\delta t & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & A_\theta^{11} & A_\theta^{12} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & A_\theta^{21} & A_\theta^{22} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & \delta t & \frac{g\delta t^2}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & g\delta t & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & A_\theta^{11} & A_\theta^{12} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & A_\theta^{21} & A_\theta^{22} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \delta t \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix} x_1 \\ \dot{x}_1 \\ \theta_1 \\ \dot{\theta}_1 \\ x_2 \\ \dot{x}_2 \\ \theta_2 \\ \dot{\theta}_2 \\ x_3 \\ \dot{x}_3 \end{bmatrix}
+ \begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
B_\theta^1 & 0 & 0 \\
B_\theta^2 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & B_\theta^1 & 0 \\
0 & B_\theta^2 & 0 \\
0 & 0 & \frac{-K_T}{m}\frac{\delta t^2}{2} \\
0 & 0 & \frac{-K_T}{m}\delta t
\end{bmatrix}
\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}
+ \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{g}{m}\frac{\delta t^2}{2} \\ \frac{g}{m}\delta t \end{bmatrix}
\end{aligned}
$$

where A_θ^{ij} refers to the ij-th entry of the system matrix of the discretized rotational dynamics A_θ and B_θ^i refers to the i-th entry of the input-to-state map B_θ for the discretized rotational dynamics. In (2.12) we have expanded all matrices for clarity and to emphasize the block structure. This form also makes clear which quantities need to be measured or identified: the rotational subsystem input-to-state map B_θ, which will be a function of continuous-time transfer function (2.1) coefficients d_0, d_1, n_0 and discrete time step δt; the rotational subsystem dynamics matrix A_θ, a function of d_0, d_1 and δt; the thrust coefficient K_T; and the mass m (note that the last two only appear in the vertical dynamics subsystem and the first two only in the attitude subsystems).

2.4 System Identification

In this section we describe how we identified (or measured) the vehicle-dependent quantities that appear in the dynamics models of Sec. 2.3. First we describe the identification of the parameters of the attitude dynamics model, and then of the vertical dynamics model. We conclude by forming the full model discretized to the nominal control frequency for the quadrotor (40 Hz) with the identified parameters included. The models with parameters identified are used for our controller development, and have also been used in other work, for example in hierarchical stochastic motion planning (Vitus, 2012).

2.4.1 Attitude dynamics

To realize a system model that is a good approximation of our actual quadrotors, we performed a rudimentary system identification of the attitude dynamics. The objective was to identify the unknown coefficients n_0, d_0, d_1 of the transfer function (2.1). This involved issuing step-input pitch (or roll) commands starting from a hover condition, and measuring the corresponding response of the actual angle $\theta(t)$. The step inputs were performed for differing magnitudes of the step and in different directions.

Fig. 2.5 shows a plot of a typical response, and the overall response for all these tests can be summarized as follows. The typical response was for the measured angle $\theta(t)$ to reach a maximum of 60% of the commanded step value in just under 0.4 seconds. The steady-state value $\theta_{ss}(t)$ appears to be approximately 50% of the command[1]. Note that there exist simple relationships between these observed quantities and the unknown coefficients we seek. Let the proportion of overshoot PO be the fraction by which the maximum value of the response exceeds its steady state value; thus we have $PO = (0.6 - 0.5)/0.5 = 0.2$. The peak time t_p is the time required for the response to reach the peak (overshoot) value, so $t_p = 0.4$. Let the DC gain G_{dc} be the proportion of the steady-state value of the signal (relative to the input); so $G_{dc} = 0.5$. It is customary to define a quantity called the damping ratio and use for it the notation ζ, and also to call the undamped natural frequency ω_n. The relationships between the observed quantities and the coefficients we seek are then

$$\begin{aligned}
\zeta &= \sqrt{\frac{\ln^2 PO}{\ln^2 PO + \pi^2}} \\
\omega_n &= \frac{\pi}{t_p\sqrt{1-\zeta^2}} \\
n_0 &= G_{dc}\omega_n^2 \\
d_0 &= \omega_n^2 \\
d_1 &= 2\zeta\omega_n.
\end{aligned}$$

With these relationships we obtain $n_0 = 38.9372$, $d_0 = 77.8743$, $d_1 = 8.0472$. Recognizing that our methodology is rather coarse, we took only the first significant digit and so selected the parameter values

$$n_0 = 40,\ d_1 = 8,\ d_0 = 80. \tag{2.13}$$

With these parameters we can now form the continuous time model (2.2) of the attitude dynamics

$$\begin{aligned}
\frac{d}{dt}x(t) &= \begin{bmatrix} -8 & 1 \\ -80 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 40 \end{bmatrix} u(t), \\
y(t) &= \begin{bmatrix} 1 & 0 \end{bmatrix} x(t).
\end{aligned} \tag{2.14}$$

Taking a discretization timestep of $\delta t = 0.025\,\mathrm{s}$ (40 Hz, the default control frequency for our quadrotor), and gravitational constant $g = 9.81\,\mathrm{m/s}^2$, we can then obtain through exact discretization (2.4)[2] the transition map and input map matrices of the equivalent discrete-time system (2.5),

$$A_\theta = \begin{bmatrix} 0.7969 & 0.2247 \\ -1.798 & 0.9767 \end{bmatrix}, \quad B_\theta = \begin{bmatrix} 0.01166 \\ 0.9921 \end{bmatrix} .$$

A simulated step response from continuous- and discrete-time models using these parameters is shown in Fig. 2.6.

Finally, we form the combined 4-state discrete-time (for $\delta t = 0.025\,\mathrm{s}$) SISO state-space model for either the body y or body $-x$ axis, with corresponding input θ_{cmd} (pitch) or ϕ_{cmd} (roll) respectively by

[1] Initially, we were surprised to observe the large steady-state offset. However, we find in the literature a model and supporting experimental results for non-trivial opposing moments on a similar quadrotor with PD attitude control. This is a manifestation of a phenomenon on rotorcraft known as 'blade flapping' (Hoffmann et al., 2007). The angles and velocities here are comparable to those in that work, and though its authors do not present step input results, it seems reasonable to conjecture that the same phenomenon may be at work in the present study. Additional analysis and/or experiments are needed to draw more certain conclusions, but this is left as a direction for future investigation.

[2] e.g. using the MATLAB Control System Toolbox command `c2d`, which produces an exact discretization assuming zero-order-hold of the input.

substituting into 2.8:

$$\begin{aligned}
x^{+} &= \begin{bmatrix} x_t \\ x_\theta \end{bmatrix}^{+} = Ax + Bu \\
&= \begin{bmatrix} A_t & B_t C_\theta \\ 0 & A_\theta \end{bmatrix} \begin{bmatrix} x_t \\ x_\theta \end{bmatrix} + \begin{bmatrix} 0 \\ B_\theta \end{bmatrix} u_\theta \\
&= \begin{bmatrix} 1 & 0.0250 & 0.0031 & 0 \\ 0 & 1 & 0.2453 & 0 \\ 0 & 0 & 0.7969 & 0.0225 \\ 0 & 0 & -1.7976 & 0.9767 \end{bmatrix} \begin{bmatrix} x_t \\ x_\theta \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0.01 \\ 0.9921 \end{bmatrix} u_\theta
\end{aligned} \tag{2.15}$$

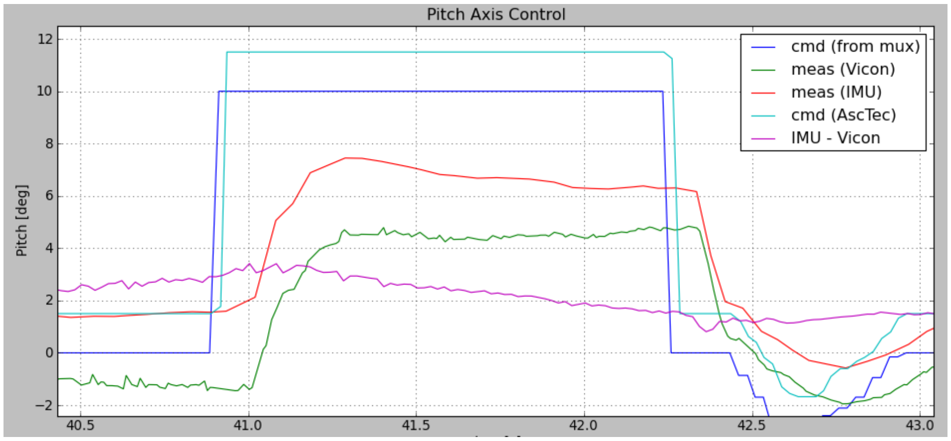


Figure 2.5: Typical experimental step response from testing data (input in cyan, output in red). Here the command is a step change of 10 degrees in the pitch axis. (The horizontal axis is time in seconds)

2.4.2 Vertical dynamics

As noted in Sec. 2.3.6, to have a model of the vertical dynamics we need to know the mass m and a thrust factor K_T. The mass was measured with a scale and found to be 1.3 kg. To determine the thrust factor, we flew the quadrotor at a hover condition (far enough from the ground that ground effect was not present) and observed the command input that was required to maintain equilibrium. We obtained a value of $K_T = 0.91$.

2.4.3 Combined lateral and vertical dynamics

Combining the parameters identified in the above, here we present the complete 10-state discrete-time (for timestep $\delta t = 0.025\,\mathrm{s}$) LTI model for the quadrotor near hover, i.e. (2.12) with all parameters identified,

$$
\begin{aligned}
x^{+} &= Ax + Bu + k \\
&= \begin{bmatrix} A_8 & 0 \\ 0 & A_z \end{bmatrix} \begin{bmatrix} x_l \\ q_3 \end{bmatrix} + \begin{bmatrix} B_8 & 0 \\ 0 & B_z \end{bmatrix} \begin{bmatrix} u_l \\ u_3 \end{bmatrix} + \begin{bmatrix} k_l \\ k_z \end{bmatrix} \\
&= \begin{bmatrix} A_t & B_t C_\theta & 0 & 0 & 0 \\ 0 & A_\theta & 0 & 0 & 0 \\ 0 & 0 & A_t & B_t C_\theta & 0 \\ 0 & 0 & 0 & A_\theta & 0 \\ 0 & 0 & 0 & 0 & A_z \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} + \begin{bmatrix} B_l & 0 & 0 \\ 0 & B_l & 0 \\ 0 & 0 & B_z \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ k_z \end{bmatrix} \\
&= \begin{bmatrix}
1 & 0.025 & 0.0031 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0.2453 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.7969 & 0.0225 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1.7976 & 0.9767 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0.025 & 0.0031 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0.2453 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.7969 & 0.0225 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1.7976 & 0.9767 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0.025 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix} x_1 \\ \dot{x}_1 \\ \theta_1 \\ \dot{\theta}_1 \\ x_2 \\ \dot{x}_2 \\ \theta_2 \\ \dot{\theta}_2 \\ x_3 \\ \dot{x}_3 \end{bmatrix} \\
&\quad + \begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0.01 & 0 & 0 \\
0.9921 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0.01 & 0 \\
0 & 0.9921 & 0 \\
0 & 0 & -0.00021875 \\
0 & 0 & -0.0175
\end{bmatrix}
\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0.0031 \\ 0.2453 \end{bmatrix}.
\end{aligned}
\tag{2.16}
$$

Finally, the measurement $y_k \in \mathbb{R}^5$ at timestep k for this system are obtained from the states and subject to some measurement noise $\epsilon_k \in \mathbb{R}^5$, which is considered to be an independent and identically distributed (i.i.d.), bounded stochastic quantity. The measurement equation is

$$
\begin{aligned}
y_k &= Cx_k + \epsilon_k \\
&= \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{bmatrix} x_k + \epsilon_k.
\end{aligned}
\tag{2.17}
$$

Thus, the measurement vector $y_k = [x_1, \theta_1, x_2, \theta_2, x_3]^T + \epsilon$.

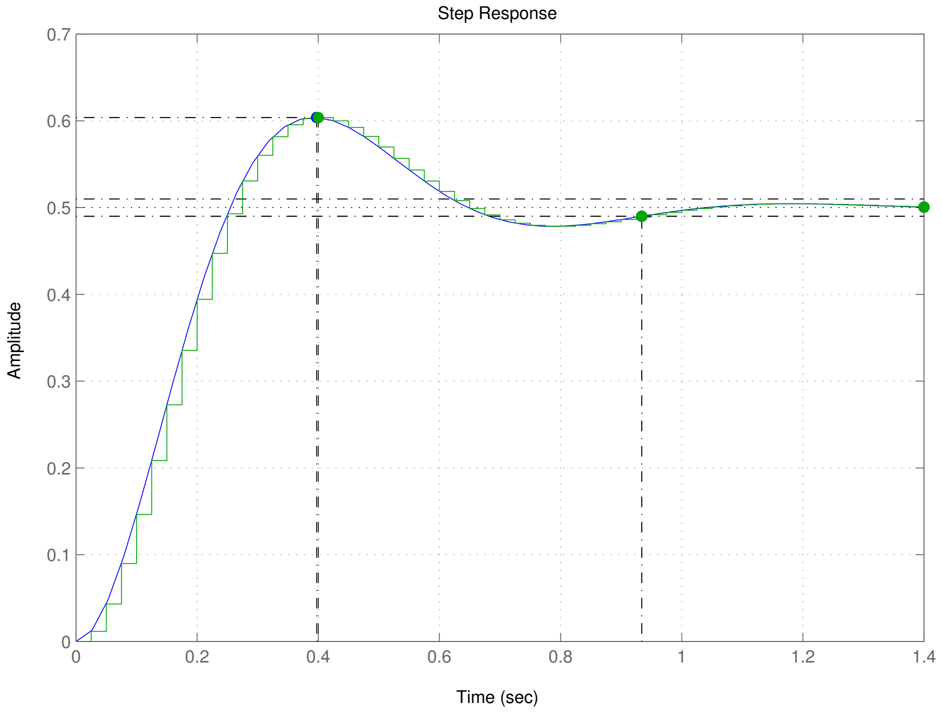


Figure 2.6: Simulated response of closed-loop attitude dynamics model, continuous and discretized ($dT = 0.025$ s), for selected parameter values, to a unity step input. Dashed vertical lines illustrate rise time of 0.4 s to a peak value of 0.6, a DC gain of 0.5, and a settling time (to within 1% of the steady-state value) of about 0.9 s.

Chapter 3

Explicit MPC Control

3.1 Overview

In this chapter we investigate the use of explicit MPC (Bemporad et al., 2002; Tondel, Johansen, and Bemporad, 2001; Mariethoz, Domahidi, and Morari, 2009) to control the quadrotor in hover. In explicit MPC, the optimization problem whose solution determines the optimal sequence of inputs is effectively solved ahead of time. That is, a "multi-parametric" quadratic programming problem is posed, in which the initial state is the parameter; the solution of this is piecewise affine over polyhedral domains of the state space. The control law can thus effectively be encoded as a look-up table, resulting in a very computationally efficient on-line controller implementation.

The main objective here was to explore the use of MPC for the real-time control of the quadrotor. There is very little prior work described in the literature on explicit MPC for a quadrotor; notably one group from the University of Patras in Greece has published several papers (Alexis, Nikolakopoulos, and Tzes, 2010a,b, 2011, 2012; Alexis et al., 2011), on a switching (among 3 piecewise-affine (PWA) systems) explicit MPC controller (though the earlier papers from this group do not use the term MPC) for a quadrotor, including some experimental results.

Our goal was to implement the reference tracking and disturbance rejection solely using MPC. We predicted that the nature of MPC, its constant prediction of future inputs based on the state of the system, would allow for superior disturbance rejection than the original PID controller. As a direct result, a more stable hover would be achieved.

Since the translational dynamics of the near-hover quadrotor in the vertical direction (perpendicular to the ground) are generally faster than those in the lateral (parallel to the ground) direction, we elected to control altitude with an existing linear (proportional-integral-derivative, PID) controller and focus on regulating the lateral states. Initially we did so separately for each of the translational axes, effectively completely decoupling the control problem into two models each with a 4-dimensional state space and single input. Next we formed a composite model by combining two copies of the 4-state model into one 8-state model with a 2-dimensional input. For each model we implemented a standard MPC controller using the control synthesis capabilities of the Multi-Parametric Toolbox (MPT) (Kvasnica, Grieder, and Baotic, 2004), and performed flight experiments with compiled versions of these controllers running on a computer on-board a quadrotor. We finish by implementing a controller that is based on the "δu formulation" for offset-free MPC (Maeder, Borrelli, and Morari, 2009); this formulation allows the MPC controller to have a sort of integral action similar to that in a PID controller, which enables the controller to achieve zero steady-state offset. Again we implemented this controller using the MPT and performed flight experiments.

The work described in this chapter was part of a semester project with Cameron Rose for ME290J (Model Predictive Control, Instructor: Prof. F. Borrelli) in Spring 2011 (Bouffard and Rose, 2011).

3.2 Problem formulation

Here we briefly recapitulate the salient points of what we here call ‘standard’ quadratic program (QP) MPC, as well as a special class of linear offset-free MPC known as the delta input (δu) formulation.

3.2.1 Standard QP-MPC

Central to the MPC technique is an optimization problem. If the objective is a (convex) quadratic function and the constraints are affine functions of the decision variables, then this is known as a quadratic program (QP) (Boyd and Vandenberghe, 2009). Further, if we parametrize the QP by a vector of parameters $x \in \mathbb{R}^n$, this is considered a multiparametric QP (mp-QP) (Borrelli, Bemporad, and Morari, 2012). The optimization problem is then of the form

$$\begin{aligned} J^*(x) = \quad & \min_{z} && J(z,x) = \frac{1}{2} z^T H z \\ & \text{subj. to} && Gz \leq w + Sx. \end{aligned} \tag{3.1}$$

Here, the parameter x is the current state. More specifically, the MPC problem is one of solving the (parametrized) constrained finite-time optimal control (CFTOC) problem

$$\begin{aligned} J_0^*(x(0)) = \quad & \min_{U_0} && J_0\left(x(0), U_0\right) \\ & \text{subj. to} && x_{k+1} = Ax_k + Bu_k,\ k = 0, \ldots, N-1 \\ & && x_k \in \mathcal{X},\ u_k \in \mathcal{U},\ k = 0, \ldots, N-1 \\ & && x_N \in \mathcal{X}_f \\ & && x_0 = x(0), \end{aligned}$$

where N is the time horizon and $\mathcal{X}_f \subseteq \mathbb{R}^n$ is a terminal polyhedron. The vector of decision variables is $U_0 = \left[u_0^T, \ldots, u_{N-1}^T\right]^T \in \mathbb{R}^s$, $s = mN$, where $u_k \in \mathbb{R}^m$, $k = 0, \ldots, N-1$ are the inputs at each timestep in the horizon. $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times m}$ are the dynamics and input-to-state map respectively for the discrete-time LTI system. The sets $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{U} \subseteq \mathbb{R}^m$ are polyhedral state and input constraints respectively. The objective is the sum of a final state cost quadratic in the final state x_N and stage costs that are quadratic in the state and inputs at each timestep

$$\begin{aligned} J_0(x(0), U_0) &= x_N^T P_N x_N + \sum_{k=0}^{N-1} x_k^T Q x_k + u_k^T R u_k, \\ &= \|x_N\|_P^2 + \sum_{k=0}^{N-1} \|x_k\|_Q^2 + \|u_k\|_R^2 \end{aligned} \tag{3.2}$$

where $P_N, Q \in \mathbb{R}^{n \times n}$ and $R \in \mathbb{R}^{m \times m}$ are positive semi-definite matrices which can be designed to express the relative importance of minimizing individual state components; typically these are chosen as diagonal matrices, and there are thus $2n + m$ ’knobs’ for the designer to adjust in this control scheme.

3.2.2 Offset-free QP-MPC using δu formulation

Offset-free MPC (Maeder, Borrelli, and Morari, 2009) seeks to augment the standard MPC formulation such that an steady-state errors are driven to zero asymptotically. In this sense offset-free MPC is similar to the addition of an integral term to a PD controller.

In the δu formulation we consider a system in which there is uncertainty in the mapping between inputs and states. The basic idea is to reformulate the system such that we are now trying to determine a change in input (hence δu) from the previous timestep, rather than the actual input itself. This amounts to forming a new system with state dimension $n+m$, where the previous timestep's input are the new m state variables. In the δu formulation the system model is thus

$$\begin{aligned}\begin{bmatrix} x_{k+1} \\ u_k \end{bmatrix} &= \begin{bmatrix} A & B \\ 0 & I \end{bmatrix} \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix} + \begin{bmatrix} B \\ I \end{bmatrix} \delta u_k \\ y_k &= \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix}. \end{aligned}$$

So that the MPC problem is well-defined, we need an estimate of all the states in this new system. We thus design an observer of the form

$$\begin{bmatrix} \hat{x}_{k+1} \\ \hat{u}_k \end{bmatrix} = \begin{bmatrix} A & B \\ 0 & I \end{bmatrix} \begin{bmatrix} \hat{x}_k \\ \hat{u}_{k-1} \end{bmatrix} + \begin{bmatrix} B \\ I \end{bmatrix} \delta u_k + \begin{bmatrix} L_x \\ L_u \end{bmatrix} \left(-y_k + C\hat{x}_k\right). \tag{3.3}$$

Furthermore, the cost function is modified such that there is no terminal cost and the stage state cost is a function of the difference between reference and desired outputs

$$J_0(x(0), U_0) = \sum_{k=0}^{N-1} \|y_k - r_k\|_Q^2 + \|\delta u_k\|_R^2 . \tag{3.4}$$

The MPC optimization problem is then

$$\begin{array}{lll} \min\limits_{U_0} & J_0(x(0), U_0) & \\ \text{subj. to} & x_k \in \mathcal{X},\ u_k \in \mathcal{U},\ k = 0, \ldots, N-1 & \\ & x_{k+1} = Ax_k + Bu_k, & k = 0, \ldots, N-1 \\ & y_k = Cx_k, & k \geq 0 \\ & u_k = u_{k-1} + \delta u_k, & k \geq 0 \\ & u_{-1} = \hat{u}_{t-1} & \\ & x_0 = \hat{x}_t & \end{array}$$

Here, the cost function is per 3.4, y_k, $k = 0, \ldots, N-1$ is a sequence of reference outputs to track, and the vector of decision variables $U_0 = \left[\delta u_0^T, \ldots, \delta u_{N-1}^T\right]^T \in \mathbb{R}^s$ now consists of the difference between the inputs and their values at the prior timestep. At run time, the control input is the sum of the previous input and the first optimal δu from the optimizer of the QP:

$$u(t) = \delta u_0^* + u(t-1).$$

3.3 Experiments

To design the explicit QP-MPC controllers we used the Multi-Parametric Toolbox (MPT)(Kvasnica, Grieder, and Baotic, 2004). The MPT allows one to specify the MPC problem in terms of the key parameters, including system matrices A, B, C, D of an LTI system, the prediction horizon N, the cost function weighting matrices Q, P_N and R (from a cost function of the form (3.2)), input and state constraints $\mathcal{U}$ and $\mathcal{X}$. MPT will then automatically form a QP of the form (3.1) and produce a solution parametrized by the initial state $x(0)$.

In this section we describe the design, implementation, and testing in flight on-board the quadrotor, of three different explicit MPC controllers. The first two controllers are based on the standard QP-MPC formulation described in (Sec. 3.2.1), and the third uses the

3.3.1 Dual, decoupled standard explicit CFTOC MPC

3.3.1.1 Controller design and implementation

We first designed a standard explicit MPC controller solving the CFTOC problem. For simplicity we implemented the control of each lateral axis (x/pitch and y/roll) of the quadrotor completely separately using the same explicit controller on each axis (hence, 'dual, decoupled'). The key parameters of the controller design are shown in Table 3.1. We encoded these parameters into the `sysStruct` and `probStruct` structures and used the function `mpt_control` to generate the explicit MPC controller. The `mpt_control` command took 31.6 seconds to complete[1] and the resulting controller consisted of 675 polyhedral regions.

We used the `mpt_exportc` command to generate a C header file encoding the controller in terms of the polyhedral regions and piecewise affine control law for each region. The header file consists primarily of the array `MPT_H` of type `float` (32-bit) with 33325 entries, which thus should occupy 33325*4/1024 = 130 kB in RAM at run-time. This header file was used together with a new controller code implemented in C++ and integrated into the existing quadrotor software framework (Bouffard, 2011). Compiled code was transferred to the quadrotor's computer (see Sec. A.1) to run on-board during flight experiments.

Symbol(s)	Value	Description, units
N	10	prediction horizon [steps]
n	4	dimension of state space
m	1	dimension of input space
δt	0.1	discrete-time model timestep [s]
A, B, C, D	per (2.8), given parameters (2.13) and (2.4)	LTI system matrices
$\overline{u}$	$10\left(\frac{\pi}{180}\right)$	maximum input [rad]
$\underline{u}$	$-10\left(\frac{\pi}{180}\right)$	minimum input [rad]
$\overline{x}$	$\left[1, 5, 10\left(\frac{\pi}{180}\right), 50\left(\frac{\pi}{180}\right)\right]^T$	state constraints (upper limit of box constraint) [m, m/s, rad, rad/s]
$\underline{x}$	-$\overline{x}$	state constraints (lower limit of box constraint) [m, m/s, rad, rad/s]
Q	$\textbf{diag}\{10, 0.1, 0.1, 0.1\}$	cost function stage weight on states
R	0.001	cost function stage weight on input
P_N	$\textbf{diag}\{10, 0.1, 0.1, 0.1\}$	cost function weight on final state
$\overline{w}$	$[0.0001, 0.005, 0.0001, 0.005]^T$	bounded additive noise (see (Kvasnica, Grieder, and Baotic, 2006, p. 32)

Table 3.1: Key parameters for dual, decoupled explicit MPC controller for one axis.

3.3.1.2 Experimental results

The experiments using the dual, decoupled CFTOC controller were preliminary, exploratory work that led to the final offset-free formulation. We include here our subjective observations from these initial experiments, that were meant to help us determine the final form of our problem.

The regulation performance was good, though the lack of coupling between the axes was noticeable when the controller recovered from large disturbances; typically the path back to the origin would not be along a straight line but rather one or the other axis would be zeroed out considerably before the other. This motivated the second preliminary formulation, described next.

[1] Thinkpad T410, Intel Core i7 M 620 CPU 2.67 GHz, 4 MB cache, Ubuntu 12.04 64-bit, MATLAB R2011b 64-bit

3.3.2 Coupled standard explicit CFTOC MPC

Next, we designed another standard explicit MPC controller for the CFTOC problem, but this time we formed an 8-state LTI system to represent the full planar/roll/pitch dynamics in one model. The motivation was to explore whether the explicit MPC technique could handle a higher-order system. Even though the system consists of two completely decoupled subsystems (i.e. system matrices are block-diagonal), the closed-loop controller implicitly couples the two subsystems via the cost function. That said, since our weighting matrices are all diagonal, we did not expect this coupling to be evident in the controller's performance. Again, we were mostly interested here in the problem size. The key parameters of the controller design are shown in Table 3.2.

Symbol(s)	Value	Description, units
N	4	prediction horizon [steps]
n	8	dimension of state space
m	2	dimension of input space
δt	0.25	discrete-time model timestep [s]
A, B, C, D	per (2.9), given parameters (2.13) and (2.4)	LTI system matrices
$\overline{u}$	$10\left(\frac{\pi}{180}\right)\cdot[1,1]^T$	maximum input [rad]
$\underline{u}$	$-10\left(\frac{\pi}{180}\right)\cdot[1,1]^T$	minimum input [rad]
$\overline{x}$	$\left[1,5,10\left(\frac{\pi}{180}\right),\pi,\ 1,5,10\left(\frac{\pi}{180}\right),\pi\right]^T$	state constraints (upper limit of box constraint) [m, m/s, rad, rad/s]
$\underline{x}$	-$\overline{x}$	state constraints (lower limit of box constraint) [m, m/s, rad, rad/s]
Q	$\mathbf{diag}\{10,0.1,0.1,0.1,\ 10,0.1,0.1,0.1\}$	cost function stage weight on states
R	$\mathbf{diag}\,\{0.001,0.001\}$	cost function stage weight on input
P_N	$\mathbf{diag}\{10,0.1,0.1,0.1,\ 10,0.1,0.1,0.1\}$	cost function weight on final state
$\overline{w}$	$[0.0001,0.005,0.0001,0.005]^T$	bounded additive noise (see (Kvasnica, Grieder, and Baotic, 2006, p. 32)

Table 3.2: Key parameters for combined 8-state system explicit MPC controller for both lateral axes

Note that we found that a horizon of $N = 10$ took a very long time to compute (and moreover, during computations would also emit an error message about numerical problems with the linear program (LP) solver), and so we reduced the horizon to $N = 4$ and increased the discrete time step to $\delta t = 0.25\,\mathrm{s}$ (such that the horizon would be longer than the settling time of the attitude dynamics).

The mpt_control command took 38.5 seconds to complete[2] and the resulting controller consisted of 361 polyhedral regions. We further used the mpt_simplify command to perform a simplification which reduces the number of polyhedral regions using a heuristic greedy merging algorithm. This simplification took 26.5 seconds to complete and resulted in a controller based on 224 polyhedral regions.

We again used the mpt_exportc command to generate a C header file, in which the MPT_H array consisted of 42490 entries, which thus should occupy 42490*4/1024 = 166 kB in RAM at run-time.

3.3.2.1 Experimental results

The performance was again good, in terms of maintaining the quadrotor's hover position within some small ϵ of a mean position $\bar{x}_{\text{hover}}$, however that mean position $\bar{x}_{\text{hover}}$ was clearly offset from the origin, which is the desired position in this problem. Fig. 3.1 shows this offset; the mean position of the quadrotor is approximately (0.05, 0.01) m. This illustrated one of the drawbacks of the standard MPC framework for regulation, namely that there is no 'integral action' that automatically adds to the input based on an accumulation of integrated steady-state error. This motivated us to ultimately investigate an implementation of offset-free MPC.

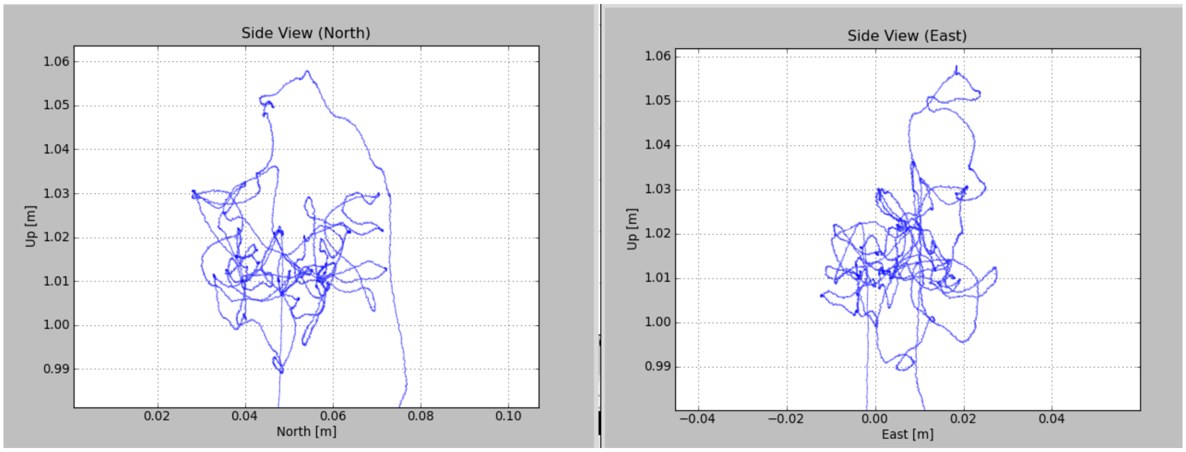


Figure 3.1: Standard MPC: trajectory of the quadrotor at hover altitude, two views towards perpendicular headings.

3.3.3 Offset-free MPC

We finally implemented an offset-free, delta input (δu) formulation, MPC in an attempt to address the steady state offset. Implementing the delta input formulation took somewhat longer as an estimator had to be designed and implemented, and several bugs shaken out of the first implementations.

3.3.3.1 Controller design and implementation

The key parameters of the controller design are shown in Table 3.3.

The mpt_control command took 1514 seconds to complete[3] and the resulting controller consisted of 4963 polyhedral regions (in 12 dimensions, recall that for the δu formulation the parametrization is over the augmented state). We further used the mpt_simplify command to perform a simplification

[2] Thinkpad T410, Intel Core i7 M 620 CPU 2.67 GHz, 4 MB cache, Ubuntu 12.04 64-bit, MATLAB R2011b 64-bit
[3] Thinkpad T410, Intel Core i7 M 620 CPU 2.67 GHz, 4 MB cache, Ubuntu 12.04 64-bit, MATLAB R2011b 64-bit

which reduces the number of polyhedral regions using a heuristic greedy merging algorithm. This simplification took 2094 seconds to complete and resulted in a controller based on 4084 polyhedral regions.

We again used the `mpt_exportc` command to generate a C header file, in which the `MPT_H` array consisted of 1621060 entries, which thus should occupy 1621060*4/1024 ≈ 6.3 MB in RAM at run-time.

Symbol(s)	Value	Description, units
N	4	prediction horizon [steps]
n	8	dimension of state space
m	2	dimension of input space
δt	0.25	discrete-time model timestep [s]
A, B, C, D	per (2.9), given parameters (2.13) and (2.4)	LTI system matrices
$\overline{u}$	$10\left(\frac{\pi}{180}\right)\cdot[1,1]^T$	maximum input [rad]
$\underline{u}$	$-10\left(\frac{\pi}{180}\right)\cdot[1,1]^T$	minimum input [rad]
$\overline{x}$	$\left[1, 5, 10\left(\frac{\pi}{180}\right), \pi, 1, 5, 10\left(\frac{\pi}{180}\right), \pi\right]^T$	state constraints (upper limit of box constraint) [m, m/s, rad, rad/s]
$\underline{x}$	-$\overline{x}$	state constraints (lower limit of box constraint) [m, m/s, rad, rad/s]
Q	$\mathbf{diag}\{10, 0.1, 0.1, 0.1, 10, 0.1, 0.1, 0.1\}$	cost function stage weight on states
R	$\mathbf{diag}\,\{10, 10\}$	cost function stage weight on input
P_N	$\mathbf{diag}\{10, 0.1, 0.1, 0.1, 10, 0.1, 0.1, 0.1\}$	cost function weight on final state
$\overline{w}$	$[0.0001, 0.005, 0.0001, 0.005]^T$	bounded additive noise (see (Kvasnica, Grieder, and Baotic, 2006, p. 32)
Q_y	$\mathbf{diag}\,\{10, 10\}$	

Table 3.3: Key parameters for combined 8-state system explicit MPC controller for both lateral axes

3.3.3.2 Estimator design and implementation

The innovation gain matrix was computed by solving a discrete time algebraic Riccati equation for P

$$P = A^T PA - (A^T PB)(R + B^T PB)^{-1}(B^T PA) + Q.$$

The Q and R matrices are covariance matrices for the estimated state vector and estimated input vector, respectively, and were chosen as

$$
\begin{aligned}
Q &= \textbf{diag}\,\{1000, 1000, 1000, 1000, 1000, 1000, 1000, 1000, 10, 10\}\ , \\
R &= \textbf{diag}\,\{1, 4, 1, 4\}\ .
\end{aligned}
$$

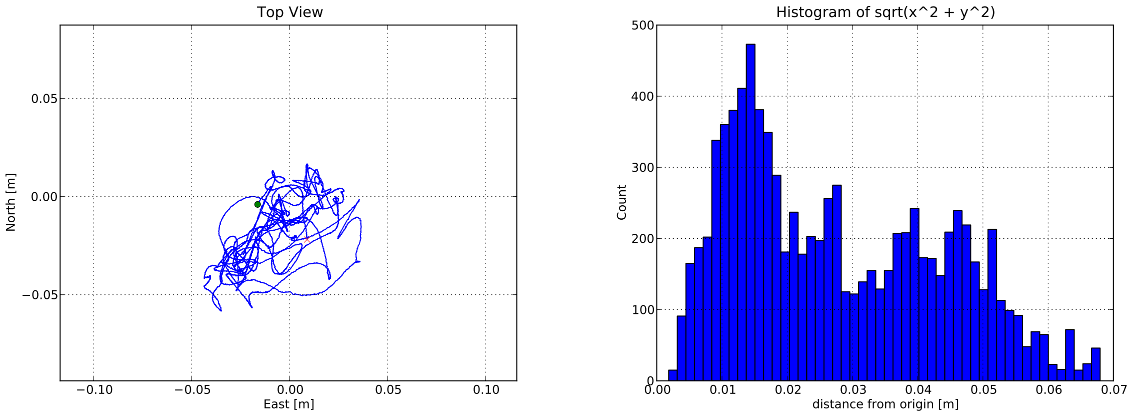
The innovation gain matrix was then computed by using the solution P

$$\begin{bmatrix} L_x \\ L_u \end{bmatrix} = (B^T P B + R)^{-1}(B^T P A).$$

3.3.3.3 Experimental results

A notable pitfall with this formulation was that the MPC seemed much more prone to being infeasible than the earlier controllers. We noticed that the input estimate $\hat{u}$ would often spike above the input constraint $\bar{u}$ coincident with the onset of infeasibility. More investigation is required but it was observed that modifying the covariance matrices in the Kalman filter design has an impact on the rate of occurrence of infeasibility, so we suspect that further tuning of the filter might be of use.

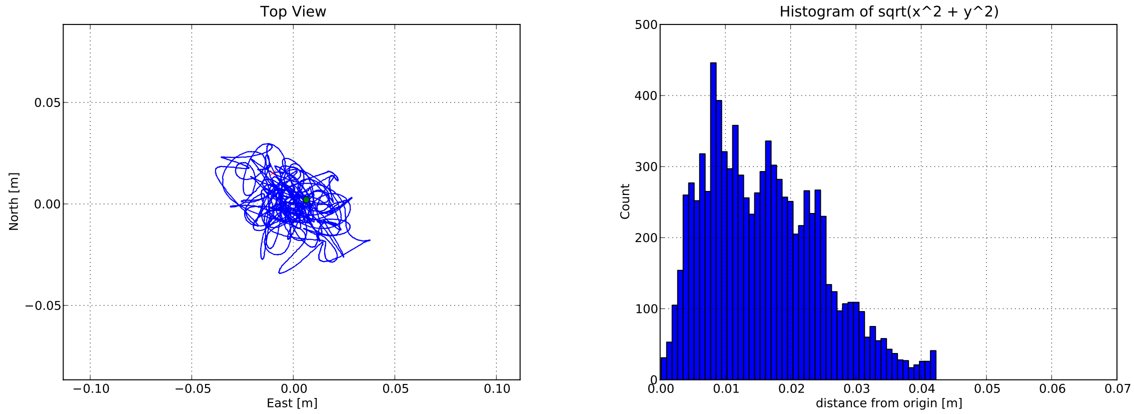
The primary performance metric of interest in this application is the accuracy of the hover point relative to the commanded reference r (the origin). We are interested in the mean deviation from the reference when the quadrotor is in steady hovering flight, with no artificial disturbances added. This can be seen in a phase plot showing the two position states in the plane (essentially, a 'top view' of the trajectory of the quadrotor) and in a histogram of the offsets from the origin, as seen in Fig. Fig. 3.2. Note that despite the zero-offset form used, it appears that the MPC controller allows a non-zero offset to persist. The mean (x, y) position during this time period is $(0.0172, 0.004)$ m. By comparison, the PID controller performs better in this regard as can be seen in Fig. Fig. 3.3; the mean position for the PID controller is $(0.0003, -0.0003)$ m. Nonetheless, this should not at this point be taken as an indication that the MPC controller is not capable of besting the PID in this performance metric; only that further tuning should be attempted.



(a) Phase plot ('top view') of trajectory for steady hover

(b) Histogram for steady hover

Figure 3.2: δu formulation offset free MPC: Position-holding performance of MPC controller



(a) Phase plot ('top view') of trajectory for steady hover

(b) Histogram for steady hover

Figure 3.3: PID controller in steady hover

3.4 Discussion

In this chapter we have described the design, implementation, and testing in experiment of explicit MPC controllers for the quadrotor. One of the objectives of this work was to test the explicit MPC concept in application on a system of interest. Considering that MPC's heritage is from the process control industry, where timescales are generally quite long[4], the ability to perform MPC on-line for the real-time control of the quadrotor is perhaps a testament to the power of Moore's Law in the intervening decades.

In terms of general observations, we can report that the explicit MPC controller's runtime CPU requirements were not an issue, something that was not obvious at the outset given the large ($>$ 6 MB, much larger then the Atom CPU's cache size of 512 kB) size of the data table for the brute-force search. From a design point of view though, the most striking difference from the PID controller is the number of free parameters available for tuning the MPC controller (at minimum, the diagonal dimensions of the Q and R matrices for both the MPC problem and the estimator) as compared to simply 6 (3 per axis) for the PID. This could be considered either an advantage or a disadvantage depending on how nuanced one wants to be in the control design, or how closely one's performance requirements can be expressed as a quadratic cost function.

In designing the explicit MPC controllers, particularly the δu formulation controller where the augmented state exacerbates this issue, we were quite aware of its main drawback: We had to reduce the horizon (we would have preferred to use a horizon of at least $N = 15$ steps at the full control sampling rate of 40 Hz) and consequently increase the MPC sampling rate; this degrades the performance of the resulting controller. We had to do this because the computation time for the explicit controller grows exponentially in the number of timesteps, and size of state and input vectors. Now certainly Moore's Law will (eventually) help out here as well, but even if we succeed in computing an explicit MPC controller for a 'larger' MPC problem, the resulting control law will in general be defined over an exponentially increasing number of polytopes, so at some point memory (both consumption and speed of access) could become an issue at runtime. These experiences helped motivate our looking into on-line MPC, and the controller considered in the next chapter is one of this type.

[4] in the introductory lecture to his MPC class, Prof. Borrelli includes a film from the 1970's in which state measurements are fed into an IBM mainframe computer by punch cards and the optimal inputs manually relayed from the computer's output to control operators by telephone!

Chapter 4

Learning-Based MPC Control

In this chapter, we present details of the real time implementation on-board a quadrotor helicopter of learning-based model predictive control (LBMPC). LBMPC rigorously combines statistical learning with control engineering, while providing levels of guarantees about safety, robustness, and convergence. Experimental results show that LBMPC can learn physically based updates to an initial model, and how as a result LBMPC improves transient response performance. We demonstrate robustness to mis-learning. Finally, we show the use of LBMPC in an integrated robotic task demonstration. The quadrotor is used to catch a ball thrown with an *a priori* unknown trajectory.

The content of this chapter has in part been previously presented in two papers (Bouffard, Aswani, and Tomlin, 2012; Aswani, Bouffard, and Tomlin, 2012).

4.1 Overview

Recent results in the applications of learning techniques to robotic systems (e.g., (Abbeel, Coates, and Ng, 2010; Tedrake et al., 2010)) suggest exploring how they might integrate with control techniques; indeed, this is an active area of research (Gillula and Tomlin, 2011). Learning-Based Model Predictive Control (LBMPC) (Aswani, Gonzalez, Sastry, and Tomlin, 2012a) is a new model-based control strategy that also allows for on-line updates to the model to improve performance, while maintaining certain guarantees about safety, robustness, and convergence. LBMPC combines aspects of learning-based control and model predictive control (MPC, (Langson et al., 2004)). Adaptive control (Åström and Wittenmark, 1994; Sastry and Bodson, 1994) aims to match the performance of a reference model with on line updates, and MPC optimizes a cost function subject to system constraints. Like robust control, LBMPC can deal with uncertainty directly, but also allows the designer to specify performance objectives to optimize and explicitly incorporates on line model updates to further improve performance. LBMPC is compatible with many learning techniques; previous work has employed a modified Nadaraya-Watson estimator with Tikhonov regularization (Aswani et al., 2012a) and a semi-parametric regression estimator (Aswani et al., 2012b).

In this chapter, we present details and experiments of an implementation of LBMPC that runs in real time on-board a quadrotor UAV with limited computing performance and memory. Here, we outline a control architecture that uses a modified extended Kalman filter (EKF) to perform state estimates and learn updated model parameters. LBMPC formulates the control problem as the solution of a convex optimization problem.

Experiments show LBMPC has similar computational requirements to linear MPC, but can improve performance by allowing the models used to be updated on-line. The experiments demonstrate learning updates including the "ground effect" (Leishman, 2006) (increased aerodynamic lift when the UAV is operating close to the ground). LBMPC provides robustness against mis-learning; that is, even if the learning algorithm is poorly designed or tuned, the formulation provides safety. To demonstrate the precision control possible using LBMPC, we program the quadrotor to catch balls.

Organization This chapter is organized as follows. We begin in Sec. 4.2 by providing a description of the theory of LBMPC and extensions thereof required for its application to quadrotor control. In Sec. 4.3 we detail the design of the LBMPC controller, including the particular form of learning we used. In Sec. 4.4 we describe some implementation details. In Sec. 4.5, we describe the experimental setup and results. Finally in Sec. 4.6 we offer some concluding remarks.

4.2 Theory

The LBMPC technique is described in detail in (Aswani et al., 2012a); here we recapitulate the most important features, and in particular describe extensions that are specific to our application of LBMPC to the control of the quadrotor. We start by briefly describing the model of the quadrotor we used.

4.2.1 Quadrotor Vehicle Model

To design an LBMPC controller for the quadrotor, we require a linear (or affine) time-invariant discrete-time nominal model which, with considerations for a bounded uncertainty or modeling error, will be the model used for verifying constraint satisfaction (i.e. safety) in the optimization central to LBMPC. The model we used was,

$$x^+ = Ax + Bu + k + h(x,u) \tag{4.1}$$

$$y = Cx + \epsilon \tag{4.2}$$

where the matrices A, B, C are as defined in (2.16) and (2.17). The term $h(x,u)$ represents the unmodeled dynamics of the system. Thus the "nominal" dynamics state update (the case in which $h \equiv 0$) is,

$$x^+ = F_{\mathcal{N}}(x,u) := Ax + Bu + k. \tag{4.3}$$

4.2.2 Overview of learning-based model predictive control theory

At the heart of the LBMPC control scheme is the on-line solution of a convex optimization problem—specifically, a quadratic program (QP). At timestep m, we solve the QP,

$$\min_{c_{\cdot},\theta} \quad \|\tilde{x}_{m+N} - \bar{x}_s\|_P^2 + \sum_{j=0}^{N-1} \|\tilde{x}_{m+j} - \bar{x}_s\|_Q^2 + \|\check{u}_{m+j} - \bar{u}_s\|_R^2 \tag{4.4}$$

$$\text{s.t.} \quad \tilde{x}_m = \bar{x}_m = \hat{x}_m \tag{4.5}$$

$$\tilde{x}_{m+i} = (A+F)\,\tilde{x}_{m+i-1} + (B+H)\,\check{u}_{m+i-1} + k + z \tag{4.6}$$

$$\bar{x}_{m+i} = A\bar{x}_{m+i-1} + B\check{u}_{m+i-1} + k \tag{4.7}$$

$$\check{u}_{m+i-1} = K\bar{x}_{m+i-1} + c_{m+i-1} \tag{4.8}$$

$$\bar{x}_{m+i} \in \mathcal{X} \tag{4.9}$$

$$\check{u}_{m+i-1} \in \mathcal{U} \tag{4.10}$$

$$\bar{x}_{m+1} \in \mathcal{X} \ominus \mathcal{D} \tag{4.11}$$

$$(\bar{x}_{m+1}, \xi) \in \omega \tag{4.12}$$

for $i \in \{1, \ldots, N\}$ where N is the number of steps forward in time over which the optimization is performed (i.e., the "horizon"). The different notions of the state are indicated by marks on the symbol; hence x (no marks) indicates the true state, $\hat{x}$ the estimated state, $\tilde{x}$ the predicted state incorporating the oracle, and $\bar{x}$ the predicted state using the nominal model. The desired state is x_s, and u_s is the steady-state control that would maintain the state at x_s, i.e. u_s solves $(A + F - I)\, x_s + (B + H)\, u_s + k + z = 0$ and this solution can be obtained explicitly by means of the Moore-Penrose pseudo-inverse, i.e.

$$u_s = (B + H)^{\dagger} \left[-\, (A + F - I)\, x_s - k - z \right].$$

K is a constant feedback gain, serving to limit the effects of model uncertainty (Chisci, Rossiter, and Zappa, 2001). Note that it differs from the feedback gain $\bar{K}$ used in the definition of the invariant set Ω, and we comment on this difference in the following subsection. The polyhedral sets $\mathcal{X}$ and $\mathcal{U}$ are bounded and convex; they encode the allowable states and inputs, respectively. These are typically expressed as sets defined by half-space inequalities. For example, $\mathcal{X} = \{x \mid F_x x \leq h_x\}$. Note that, owing to the boundedness of β, $\mathcal{X}$, and $\mathcal{U}$, the oracle is also bounded: $\mathcal{O}_m(x, u) \in \mathcal{D}$ for some bounded, convex polytope $\mathcal{D}$. The set ω is an approximation of the maximal output admissible disturbance invariant set and $\xi \in \mathbb{R}^3$ is a parametrization of points that can be feasibly tracked with a linear controller (see Sec. 4.2.3).

The solution $\{c_i^*\}_{i=m}^{m+N-1}$ to this QP encodes the optimal—with respect to minimization of the cost function in (4.4)—sequence of controls to apply to the system over the next N steps based on the current parameter and state estimates. The actual controls are the $\check{u}$'s, (4.8) used to determine predicted oracle states $\tilde{x}$ (4.6) used in the cost function and predicted nominal model states $\bar{x}$ (4.7) used for constraint satisfaction.

The key output of the QP is only the first control of the sequence of N controls, $\check{u}_m = K\bar{x}_m + c_m^*$. This is the control that is actually applied to the system; at the next iteration through the control loop, the QP is solved once again with new state estimates and new oracle dynamics based on updated F, H, z matrices.

4.2.3 Extensions of LBMPC for quadrotors

LBMPC is based on a linear MPC scheme for tracking (Limon et al., 2008) that can be robustified using tube-MPC (Chisci, Rossiter, and Zappa, 2001; Langson et al., 2004; Limon et al., 2010), and LBMPC reduces conservativeness by making the initial condition be fixed rather than be an optimization parameter (cf. Langson et al. (2004); Limon et al. (2010)). We further modify LBMPC to reduce conservativeness by assuming that noise enters only into the first time step of the model prediction (noise still enters into all time steps of the true system); the advantage of this is greatly enlarged regions of feasibility for the optimization problem that defines the LBMPC. This formulation of LBMPC has the same robust constraint satisfaction properties of the original form of LBMPC–the results in (Aswani et al., 2012a) trivially extend to this case.

Feasible set point tracking

Here we introduce some concepts related to feasible set point tracking for a constrained linear affine system subject to disturbances. The steady-state output y_s corresponds to some state x_s and input u_s, and it can be characterized (Limon et al., 2008, 2010) by the set of solutions to

$$\begin{bmatrix} A - I & B & 0 \\ C & 0 & I \end{bmatrix} \begin{bmatrix} x_s \\ u_s \\ y_s \end{bmatrix} = \begin{bmatrix} -k \\ 0 \end{bmatrix}.$$

These solutions form a set of affine subspaces that can be parametrized–similarly to (Limon et al., 2008, 2010; Aswani et al., 2012a)–as $x_s = \Lambda\xi + x_0$, $u_s = \Psi\xi + u_0$, $y_s = \Pi\xi + y_0$, where $\xi \in \mathbb{R}^3$; Λ, Ψ, Π are full column-rank matrices with suitable dimensions, and x_0, u_0, y_0 are (possibly zero) vectors.

The maximal output admissible disturbance invariant set $\Omega \subseteq \mathcal{X} \times \mathbb{R}^3$ was defined in (Kolmanovsky and Gilbert, 1998). It is a set of points such that any trajectory of the system with initial condition chosen from this set remains within the set for any sequence of bounded disturbances, while satisfying constraints on the state and input. Consider any point $(x, \xi) \in \Omega$, where the ξ component parametrizes points that can feasibly be tracked using a linear controller. If $\bar{K}$ is a nominal feedback gain such that $\left(A + B\bar{K}\right)$ is Schur stable, then the set Ω satisfies: i) disturbance invariance

$$\left(Ax + B\bar{K}\left(x - (\Lambda\xi + x_0)\right) + B\left(\Psi\xi + u_0\right) + k, \xi\right) \oplus (\mathcal{D}, 0) \subseteq \Omega; \tag{4.13}$$

and ii) constraint satisfaction

$$\Omega \subseteq \left\{(x, \xi) \mid x \in \mathcal{X}; \Lambda\xi + x_0 \in \mathcal{X}; \bar{K}\left(x - (\Lambda\xi + x_0)\right) + (\Psi\xi + u_0) \in \mathcal{U}; \Psi\xi + u_0 \in \mathcal{U}\right\}.$$

Invariant set computations for the quadrotor

Robust constraint satisfaction is the property that following the control law provided by LBMPC will never lead to a situation in which a constraint is violated at some point in the future. It holds for both the original and modified LBMPC, provided that the set Ω can be computed; however, we are unable to compute this set for the quadrotor. Because of these difficulties, we use an approximation of Ω that displays good empirical performance and robustness.

Methods for computing Ω (Gilbert and Tan, 1991; Rakovic and Baric, 2010) start with an initial polytopic approximation of the set, and they refine the approximation by successively adding linear constraints to the polytope. The Ω for the quadrotor may be comprised of a very large number of such constraints, and this may explain why we could not compute Ω–we may not have allowed the algorithms enough time to terminate. However, using a complex Ω in the LBMPC formulation would increase the computation time for the controller so as to render it un-implementable in real-time. the second possible explanation for why we cannot compute Ω is that it does not exist for the parameters of our quadrotor, but this would be reflective of the overly-conservative nature of this type of robustness rather than a statement about the controllability of the system.

To overcome these difficulties, we use an outer approximation of the set Ω. The idea is to start by rewriting the disturbance invariance condition (4.13) as

$$\left(Ax + B\bar{K}\left(x - (\Lambda\xi + x_0)\right) + B\left(\Psi\xi + u_0\right) + k, \xi\right) \subseteq \Omega \ominus (\mathcal{D}, 0), \tag{4.14}$$

and then relax it to

$$\left(Ax + B\bar{K}\left(x - (\Lambda\xi + x_0)\right) + B\left(\Psi\xi + u_0\right) + k, \xi\right) \subseteq \mathcal{X} \ominus (\mathcal{D}, 0). \tag{4.15}$$

The set of points that satisfy (4.14) are a subset of the points that satisfy (4.15), because $\Omega \subseteq \mathcal{X}$ by construction.

Because the constraints $\mathcal{X}, \mathcal{U}$ and disturbance bounds $\mathcal{D}$ are compact, convex polytopes, we can exactly represent the outer approximation of Ω as the points that satisfy a set of linear inequalities. Let $F_P x \leq h_P$, $F_x x \leq h_x$, and $F_u u \leq h_u$ be the inequality representations of the polytopes $\mathcal{X} \ominus \mathcal{D}$, $\mathcal{X}$, and $\mathcal{U}$ respectively. The approximation ω is then given by a polytope, which we can express as the set of points (x, ξ) that satisfy the linear inequalities

$$\begin{bmatrix} F_P\left(A + B\bar{K}\right) & F_P B\left(\Psi - \bar{K}\Lambda\right) \\ F_x & 0 \\ 0 & F_x\Lambda \\ F_u\bar{K} & F_u\left(\Psi - \bar{K}\Lambda\right) \\ 0 & F_u\Psi \end{bmatrix} \begin{bmatrix} x \\ \xi \end{bmatrix} \leq \begin{bmatrix} h_p - F_P\left(k + B\left(\bar{K}\bar{x}_0 - \bar{u}_0\right)\right) \\ h_x \\ h_x - F_x\bar{x}_0 \\ h_u - F_u\left(\bar{K}\bar{x}_0 - \bar{u}_0\right) \\ h_u - F_u\bar{u}_0 \end{bmatrix}. \tag{4.16}$$

Feedback gains

Because the functions in the optimization problem are continuous and the constraints are linear, this implies continuity of the value function (Aswani et al., 2012a); this is in fact one type of robustness (Grimm, 2004). Input-to-state stability (ISS) can be shown if the oracle is bounded and this scheme can be proven to be convergent for the nominal model (Aswani et al., 2012a), but we have been unable to prove this. The reason for this is that existing proof techniques apply to the case where $\bar{K} \equiv K$, but that is not true here. However, our empirical observation of an implementation on the quadrotor is that this scheme is effectively ISS; the quadrotor tracks a steady point within a root-mean-square error of about 2 cm in each positional direction.

The reason for having $\bar{K} \neq K$ is that it leads to better empirical performance on the quadrotor. The intuition for why this is the case is as follows. The smaller the gain $\bar{K}$ is, the larger the set Ω will be. The advantage of a larger Ω is a larger region of feasibility of the LBMPC, and so performance and robustness will be better. On the other hand, having a large gain K leads to fast convergence and tracking for the quadrotor, but it would lead to a very small set Ω. Thus, we use different values for these two gains.

We compute these gains by determining the feedback gain for an LQR controller for the nominal system, given a state cost weighting matrix Q and input cost weighting matrix R. The difference is that we scale the input cost weighting matrix R when computing $\bar{K}$ (feedback gain for terminal set computations) by a factor of 12.5 as compared to when we compute K (nominal feedback gain). Since the inputs are penalized more heavily in the computation of $\bar{K}$, this results in this gain being smaller than K.

4.3 Control System Design

In this section we describe the design of the quadrotor controller incorporating the LBMPC scheme. The overall control architecture is composed of (i) estimation of the vehicle state and learning of the unmodeled dynamics, and (ii) an optimization-based procedure for performing closed-loop control. Both are model-based: The state estimate uses a model of the system to make predictions of the current state based on the past state and input, and the optimization problem uses a model to evaluate the result of prospective control policies over a finite planning horizon, and thus select the best policy based on the corresponding costs. Fig. 4.1 shows a schematic that illustrates the overall system setup and interconnections between the major components.

4.3.1 Vehicle State Estimation and Learning

A natural approach for joint state and parameter estimation is to use an extended Kalman filter (EKF), and we use a modified EKF (Ljung, 1979) known sometimes as a dual EKF that has improved convergence conditions when the system is linear in the state for fixed parameters and vice versa. The reason for its improved convergence is the use of a Luenberger observer to estimate the state and an EKF to estimate the parameters, instead of an EKF for both the state and parameters.

In the quadrotor, the parameters of the oracle correspond to a linearization of the unmodeled dynamics about operating points that vary as the quadrotor moves through the state space; consequently, the parameters will change. The EKF of (Ljung, 1979) requires the addition of a noise term in order to handle this.

Let m be the current timestep. We assume a linear, time-varying oracle (Aswani et al., 2012a) $\mathcal{O}_m : \mathbb{R}^{10} \times \mathbb{R}^3 \to \mathbb{R}^{10}$, parametrized by a vector of parameters $\beta \in \mathbb{R}^{12}$, of the form

$$\mathcal{O}_m(x, u) = F(\beta)x + H(\beta)u + z(\beta), \tag{4.17}$$

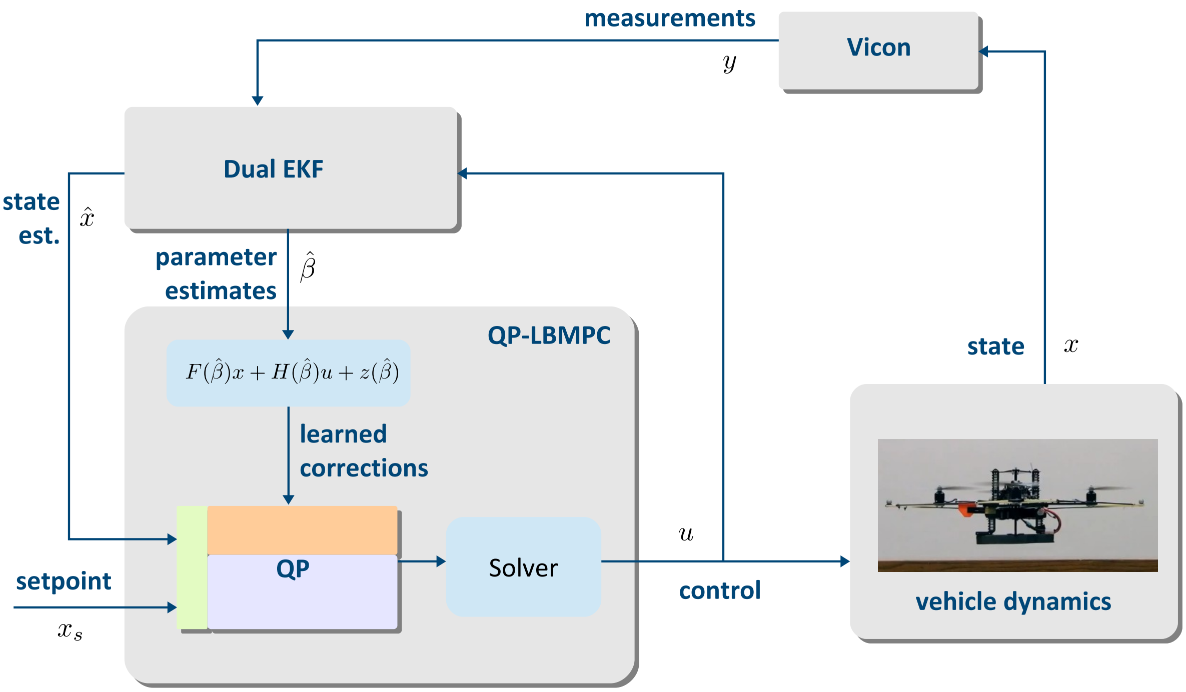


Figure 4.1: System diagram showing QP-LBMPC for the quadrotor.

in which F, H, and z are linear in the components of β. The exact form of $F(\beta)$, $H(\beta)$, and $z(\beta)$ is informed by our physical understanding of the system, as detailed in Chapter 2. For clarity, we show these in the following equations

$$F(\beta) = \begin{bmatrix} 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&\beta_1&\beta_2&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&\beta_4&\beta_5&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0 \end{bmatrix}; H(\beta) = \begin{bmatrix} 0&0&0\\ 0&0&0\\ 0&0&0\\ \beta_3&0&0\\ 0&0&0\\ 0&0&0\\ 0&0&0\\ 0&\beta_6&0\\ 0&0&0\\ 0&0&\beta_7 \end{bmatrix}; z(\beta) = \begin{bmatrix} 0\\ \beta_8\\ 0\\ \beta_9\\ 0\\ \beta_{10}\\ 0\\ \beta_{11}\\ \beta_{12}\\ 80\beta_{12} \end{bmatrix}.$$

Note that the last component of $z(\beta)$ is due to the relationship between the two components of k_z (see 2.11) for our timestep of 0.025 s.

In what follows, we simply write F, H, and z, dropping the explicit dependency on the parameters β. The parameters are constrained such that $\beta_{\min,i} \le \beta_i \le \beta_{\max,i}$, $i = 1, \ldots, 12$. The state update equation under the learned dynamics is then,

$$\begin{aligned} x^+ = F_{\mathcal{O}}(x,u) &:= F_{\mathcal{N}}(x,u) + \mathcal{O}_m(x,u) \\ &= (A+F)\,x + (B+H)\,u + k + z. \end{aligned} \tag{4.18}$$

The parameters $\beta = \{\beta_1, \ldots, \beta_{12}\}$, can be thought of as “adjustments” to certain entries of the nominal dynamics matrices. Estimates $\hat{\beta}$ of the parameters are determined jointly with estimates $\hat{x}$ of the state. We assume that the parameters evolve according to $\beta^+ = \beta + \mu$ where μ is noise. The modified EKF is governed by the set of update equations,

$$\begin{aligned}
\hat{x}^+ &= (A+F)\,x + (B+H)\,u + k + z + \hat{K}\zeta \\
P_2^+ &= (A+F)P_2 + MP_3 - \hat{K}\Xi L^T \\
P_3^+ &= P_3 - L\Xi L^T - \delta P_3 P_3^T + \Upsilon \\
\hat{\beta}^+ &= \text{bound}(\hat{\beta} + L\zeta)
\end{aligned}$$

Where $L := P_2^T C^T \Xi^{-1}$ and $M := \frac{\partial}{\partial\beta}\,(F\hat{x} + Hu + z)$. Here, $\zeta = y - C\hat{x}$ is the measurement innovation. The matrix $\hat{K}$ is a feedback matrix chosen such that $A + F(\beta) - \hat{K}C$ is Schur (asymptotically exponentially; all eigenvalues strictly within the unit circle) stable for all β within bounds. This can be verified using the Bauer-Fike theorem (Bauer and Fike, 1960), for example. The matrices $P_2 \in \mathbb{R}^{10\times 12}$, $P_3 \in \mathbb{R}^{12\times 12}$, $\Xi \in \mathbb{R}^{5\times 5}$, and $\Upsilon \in \mathbb{R}^{12\times 12}$ are the cross-covariance between state and parameter estimates, and the covariances of the parameter estimates, the measurement noise, and the parameter noise, respectively. The tuning parameter $\delta > 0$ improves the numerics. The bound function clips each parameter to be within the specified limits. Table 4.1 shows the chosen values of the various dual EKF design parameters (not to be confused with the model parameters β being estimated). Note that in most cases the upper and lower bounds on each parameter are symmetric, with the exception of the parameter β_7, which the learning is allowed to vary in the negative direction 10 times more than in the positive direction. This represents encoding our expectation from helicopter physics that the ground effect will only ever add to the effective lift per unit input (recall that "up" is actually in the $-\mathbf{x}_3$ direction) and not subtract from it.

Observability considerations Suppose that z were not sparse and that every entry was a parameter, i.e. $z = [\beta_8, \ldots, \beta_{17}]$. Results in (Aswani et al., 2012a), for instance, show that it is impossible in this system to simultaneously identify the parameters of this z while computing state estimates $\hat{x}$ based on noisy measurements y. The central issue is one of observability. If we augment the state to (x, β), then being able to identify all the parameters β and estimate the states x is equivalent to the observability of the system

$$\begin{bmatrix} x^+ \\ \beta^+ \end{bmatrix} = \begin{bmatrix} (A+F)\,x + (B+H)\,u + k + z \\ \beta \end{bmatrix}.$$

These equations are jointly nonlinear in (x, β), even though they are linear in x for fixed β and vice versa. In our application to the quadrotor, we used standard techniques (Albertini and D'Alessandro, 2002) to verify the joint observability of the unmeasured states and the parameters to be identified by the oracle.

4.3.2 LBMPC parameters

Here we detail the key parameters used to generate the LBMPC controllers, including the prediction horizon N, discretization timestep δt, feedback gains (nominal K and terminal set $\bar{K}$), state and input box constraints used to derive the state and input constraint polytopes $\mathcal{X}$ and $\mathcal{U}$ respectively, weighting matrices Q, R, P used in the cost function, and model uncertainty bounds. We tuned the controller somewhat differently for different experiments. Parameters that were the same in all experiments are shown in Table 4.2. The parameters used for the "ball-catching" experiment are shown in Table 4.3, and the parameters we used in the other experiments are shown in Table 4.4.

4.4 Implementation

In this section we describe some considerations of implementing LBMPC for on-board use on the quadrotor. The QP optimization central to the LBMPC technique can be numerically solved using a

Symbol	$\in$	Value	Description
$\hat{K}$	$\mathbb{R}^{10\times 5}$	$\begin{bmatrix} 1.35 & 0.0031 & 0 & 0 & 0 \\ 18 & 0.2453 & 0 & 0 & 0 \\ 0 & 1.1236 & 0 & 0 & 0 \\ 0 & 16.8285 & 0 & 0 & 0 \\ 0 & 0 & 1.35 & 0.0031 & 0 \\ 0 & 0 & 18 & 0.2453 & 0 \\ 0 & 0 & 0 & 1.1236 & 0 \\ 0 & 0 & 0 & 16.8285 & 0 \\ 0 & 0 & 0 & 0 & 0.3 \\ 0 & 0 & 0 & 0 & 7.3 \end{bmatrix}$	innovation feedback gain in state update
Ξ	$\mathbb{R}^{5\times 5}$	$\mathbf{diag}\left\{0.01, 0.01, 0.01, 0.01, 0.0075\right\}$	covariance matrix of measurement noise
δ	$\mathbb{R}_{>0}$	10^{-12}	tuning parameter
$\bar{\beta}$	$\mathbb{R}^{12}$	$(0.0899, 0.0488, 0.0496, 0.0899, 0.0488, 0.0496,$ $0.0003, 0.025, 0.025, 0.025, 0.0003)^T$	maximum values of components of parameter vector β
$\underline{\beta}$	$\mathbb{R}^{12}$	$-\bar{\beta}$ (except for $\underline{\beta}_7 = -10\bar{\beta}_7$)	minimum values of components of parameter vector β
$P_2(0)$	$\mathbb{R}^{10\times 12}$	$\mathbf{0}_{10\times 12}$	initial cross-covariance matrix between state and parameter estimates
$P_3(0)$	$\mathbb{R}^{12\times 12}$	$\mathbf{blkdiag}\left\{P_I, P_{II}\right\}$ where $P_I = 10^{-3}\cdot$ $\mathbf{diag}\left\{0.1798, 0.1953, 0.0992,\right.$ $\left.0.1798, 0.1953, 0.0992, 0.0262\right\},$ $P_{II} = 10^{-7}\cdot$ $\mathbf{diag}\left\{0.25, 0.25, 0.25, 0.25, 0.0031\right\}$	initial covariance matrix of parameter estimates
Υ	$\mathbb{R}^{12\times 12}$	$0.1P_3(0)$	covariance matrix of parameter noise

Table 4.1: Dual EKF design parameter values

variety of algorithms. Recent so-called “fast MPC” techniques Ferreau, Bock, and Diehl (2008); Wang and Boyd (2010); Zeilinger, Jones, and Morari (2011) seek to leverage this sparsity. In Chapter 5 we explore this direction, but in the present chapter we describe results we have obtained using LSSOL (Gill et al., 1986). LSSOL is a dense, two-phase active-set solver implemented in FORTRAN 77. Because LSSOL is a dense solver, reducing the number of decision variables in the optimization problem will improve computation speed. This can be done by making substitutions for the variables $\overline{x}$, $\check{u}$, and $\tilde{x}$. For the quadrotor model, when using a horizon of $N = 15$, this leads to a reduction in the number of decision variables from 363 to 33. For example, the variables $\overline{x}$ can be removed by noting that

$$\overline{x}_{m+i} = \left(A + B\bar{K}\right)^i \hat{x}_m + \sum_{j=0}^{i-1} \left(A + B\bar{K}\right)^{i-j} \left(Bc_{m+i-1} + k\right),$$

and similar reductions can be used to remove $\tilde{x}$ and $\check{u}$.

Another opportunity to improve solution speed lies in the form of the optimization. LSSOL can solve the problem when it is formulated as a constrained least squares problem (in the LSSOL documentation

Symbol(s)	Value	Description, units
N	15	prediction horizon
δt	0.025	discretization timestep [s]
R	$\mathbf{diag}\{8,8,0.8\}$	input cost weights (for QP cost function and nominal feedback gain computation)
$\bar{R}$	$12.5 \cdot R_{LQR}$	input cost weights (for terminal set feedback gain computation)
$(\underline{x}_1, \underline{x}_2, \underline{x}_3)$	$(-2, -3, -3)$	state constraints: minimum position [m]
$(\bar{x}_1, \bar{x}_2, \bar{x}_3)$	$(2, 3, 3)$	state constraints: maximum position [m]
$(\bar{\dot{x}}_1, \bar{\dot{x}}_2, \bar{\dot{x}}_3)$	$(10, 10, 5)$	state constraints: maximum velocity [m/s]
$(\lvert\bar{\theta}\rvert_1, \lvert\bar{\theta}\rvert_2)$	$\left(\frac{\pi}{2}, \frac{\pi}{2}\right)$	state constraints: maximum absolute pitch, roll angle [rad]
$\left(\bar{\dot{\theta}}_1, \bar{\dot{\theta}}_2\right)$	$(4\pi, 4\pi)$	state constraints: maximum absolute pitch, roll angle rate [rad/s]
$(\bar{u}_1, \bar{u}_2, \bar{u}_3)$	$\left(35\frac{\pi}{180}, 35\frac{\pi}{180}, 18\right)$	input constraints: maximum commanded pitch, roll angle; max. thrust [rad, rad, N]
$(\underline{u}_1, \underline{u}_2, \underline{u}_3)$	$\left(-35\frac{\pi}{180}, -35\frac{\pi}{180}, 0\right)$	input constraints: minimum commanded pitch, roll angle; min. thrust [rad, rad, N]
$\bar{x}_{\mathcal{D}}$	$(0.01, 0.1, 0.01, 0.1, 0.01,$ $0.01, 0.1, 0.01, 0.1, 0.01)$	state uncertainty (used to derive oracle bounding polytope $\mathcal{D}$) [m, m/s, rad, rad/s, m, m/s, rad, rad/s, m, m/s]

Table 4.2: LBMPC parameters (common across all experiments)

this is termed a problem of type LS1)

$$\begin{aligned} &\min_{\mu} \|W_1\mu - v_1\|_2^2 \\ &\text{s.t.} b_L \leq U\mu \leq b_U. \end{aligned}$$

Empirical tests show that this formulation requires 2/3 of the computation time on the on-board computer of the quadrotor as compared to solving the standard formulation of a QP.

4.5 Experimental Results

In this section, we describe the results of several experiments that illustrate different aspects of the system performance, with particular emphasis on the benefits of LBMPC over standard linear MPC. We refer readers to Appendix A for details on our quadrotor testbed.

First, we describe how LBMPC is able to learn updates to the model that can be directly related to a known aerodynamic phenomenon (the ground effect). We then show how LBMPC can improve the transient response characteristics, relative to standard linear MPC. Next, we describe an experiment in which the learning was deliberately poorly tuned to illustrate robustness to 'mis-learning'. Finally, to demonstrate the use of LBMPC for fast and accurate control in the context of an overall robotic task, we control the quadrotor using LBMPC to catch balls that are tossed by hand.

Symbol(s)	Value	Description, units
Q	$\mathbf{diag}\{20,1,0.01,0.01,20,1,0.01,0.01,20,1\}$	state cost weights (for QP cost function and nominal and terminal set feedback gain computations)
K	$-\begin{bmatrix} 1.4939 & 0 & 0 \\ 1.0482 & 0 & 0 \\ 0.3444 & 0 & 0 \\ 0.1101 & 0 & 0 \\ 0 & 1.4939 & 0 \\ 0 & 1.0482 & 0 \\ 0 & 0.3444 & 0 \\ 0 & 0.1101 & 0 \\ 0 & 0 & 4.8305 \\ 0 & 0 & 3.8689 \end{bmatrix}$	nominal feedback policy gain
$\bar{K}$	$-\begin{bmatrix} 0.4352 & 0 & 0 \\ 0.4860 & 0 & 0 \\ 0.1164 & 0 & 0 \\ 0.0537 & 0 & 0 \\ 0 & 0.4352 & 0 \\ 0 & 0.4860 & 0 \\ 0 & 0.1164 & 0 \\ 0 & 0.0537 & 0 \\ 0 & 0 & 1.3893 \\ 0 & 0 & 2.0164 \end{bmatrix}$	feedback gain for terminal set computation
P	$10^3 \begin{bmatrix} 1.0735 & 0.2001 & 0.1172 & 0.0090 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.2001 & 0.0614 & 0.0370 & 0.0032 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.1172 & 0.0370 & 0.0247 & 0.0019 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.0090 & 0.0032 & 0.0019 & 0.0003 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.0735 & 0.2001 & 0.1172 & 0.0090 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.2001 & 0.0614 & 0.0370 & 0.0032 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.1172 & 0.0370 & 0.0247 & 0.0019 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.0090 & 0.0032 & 0.0019 & 0.0003 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0113 & 0.1618 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1618 & 0.0465 \end{bmatrix}$ terminal cost weight in QP	

Table 4.3: LBMPC parameters (ball catching experiment)

4.5.1 Learning the ground effect

The ground effect is a well known aerodynamic effect in which the vehicle is subject to additional lift when in the vicinity of the ground. In helicopters, ground effect typically has a non-negligible impact on lift force when the main rotor is within 2 rotor diameters of the ground (Leishman, 2006). This effect has also been noted in other quadrotors (Waslander et al., 2005; Bouabdallah and Siegwart, 2007; Guenard, Hamel, and Eck, 2006).

In this experiment, the quadrotor was commanded to hover at a specified height, out of the ground effect, and after some time (at approx. 249 s on the plot), the altitude command was changed to correspond to a ground clearance of 3 cm. At this height, the plane of the rotors is approximately 0.19 m from ground, or about $3/4$ of one rotor diameter. In the parametrization used, β_7 is the learned change in the input mapping for the thrust input, with the nominal value being the $(10, 3)$ entry of B. As shown in Fig. Fig. 4.2, the parameter estimate quickly (within approximately 1 s) adjusts to reflect an increase in the total thrust per unit thrust command (ratio of β_7 to $B_{10,3}$). A clear increase in effective thrust per input thrust is seen when the quadrotor is in the vicinity of the ground; approximately 6% more thrust per unit command is observed. When the command is returned to the original value, β_7 reverts correspondingly, within about 2 s.

For the same experiment but with standard linear MPC (nominal model only, no learning), the quadrotor is not able to hover at the commanded distance above the ground, because the actual effective thrust is significantly greater than what the nominal model predicts. Thus, when flying with standard linear MPC, it is not possible to perform a “soft landing”—one has to manually cut power to the propellers and let the quadrotor fall the remaining distance.

4.5.2 Decreased overshoot in step response

In this experiment, we investigated the effects of LBMPC on the transient response of the quadrotor to changes in hover setpoint. The quadrotor was commanded to initially hover at $x_1 = -1\,\mathrm{m}$. The setpoint was repeatedly changed to $x_1 = 1\,\mathrm{m}$ and then back to $x_1 = -1\,\mathrm{m}$ after a delay of $3.5\,\mathrm{s}$. We performed this test with both linear MPC (using only the nominal model) and with LBMPC. Fig. Fig. 4.3 shows a comparison of the x_1-axis position of the quadrotor during this maneuver between

Symbol(s)	Value	Description, units
Q	$\mathbf{diag}\{60,1,0.01,0.01,60,1,0.01,0.01,60,1\}$	state cost weights (for QP cost function and nominal and terminal set feedback gain computations)
K	$-\begin{bmatrix} 2.5490 & 0 & 0 \\ 1.4175 & 0 & 0 \\ 0.5319 & 0 & 0 \\ 0.1384 & 0 & 0 \\ 0 & 2.5490 & 0 \\ 0 & 1.4175 & 0 \\ 0 & 0.5319 & 0 \\ 0 & 0.1384 & 0 \\ 0 & 0 & 8.2824 \\ 0 & 0 & 4.9807 \end{bmatrix}$	nominal feedback policy gain
$\bar{K}$	$-\begin{bmatrix} 0.7475 & 0 & 0 \\ 0.6541 & 0 & 0 \\ 0.1771 & 0 & 0 \\ 0.0698 & 0 & 0 \\ 0 & 0.7475 & 0 \\ 0 & 0.6541 & 0 \\ 0 & 0.1771 & 0 \\ 0 & 0.0698 & 0 \\ 0 & 0 & 2.3930 \\ 0 & 0 & 2.6330 \end{bmatrix}$	feedback gain for terminal set computation
P	$\begin{bmatrix} 450.9 & 97.84 & 56.11 & 5.204 & 0 & 0 & 0 & 0 & 0 & 0 \\ 97.84 & 39.72 & 23.04 & 2.372 & 0 & 0 & 0 & 0 & 0 & 0 \\ 56.11 & 23.04 & 15.77 & 1.255 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5.204 & 2.372 & 1.255 & 0.2299 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 450.9 & 97.84 & 56.11 & 5.204 & 0 & 0 \\ 0 & 0 & 0 & 0 & 97.84 & 39.72 & 23.04 & 2.372 & 0 & 0 \\ 0 & 0 & 0 & 0 & 56.11 & 23.04 & 15.77 & 1.255 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5.204 & 2.372 & 1.255 & 0.2299 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 455.6 & 93.73 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 93.73 & 37.65 \end{bmatrix}$	terminal cost weight in QP

Table 4.4: LBMPC parameters (experiments other than ball catching)

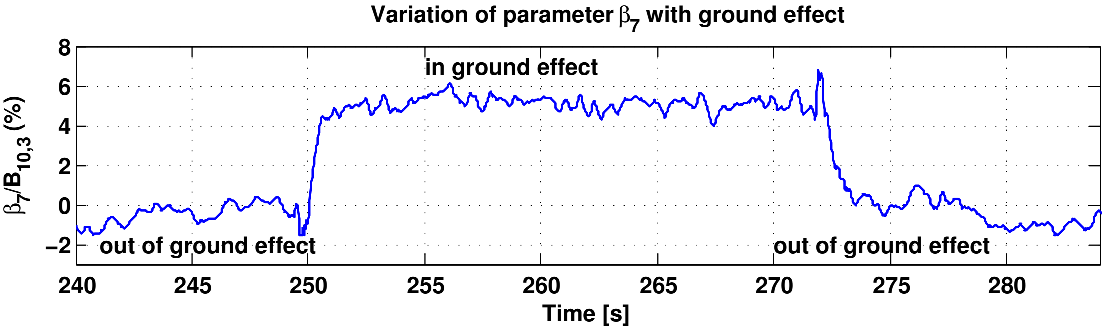


Figure 4.2: Variation of thrust input mapping $(B+H)_{10,3}/B_{10,3}$ vs. time.

linear MPC and LBMPC. The LBMPC response exhibits considerably less overshoot (62% less in the $x_1 = -1\,\mathrm{m}$ maneuver shown in Fig. 4.3) than the linear MPC response. In addition, we observed that the LBMPC response characteristics would improve with repeated maneuvers; this is expected given that the model parameters continue to be refined with each maneuver. We also observed a greater decrease in overshoot when successive step maneuvers were more closely spaced in time. This reflects the fact that the parameter adjustments learned during the transient flight are important in improving the stopping characteristics, and it suggests that a possible avenue for improvement is to introduce a velocity-dependent drag term in the dynamics model. This example demonstrates the type of performance improvement that is possible with LBMPC and a well-behaved oracle.

4.5.3 Robustness to "incorrect learning"

In this experiment, we deliberately caused the dual EKF to be prone to mis-estimate the model parameters by grossly increasing the noise process covariance Υ. We allowed the quadrotor to hover at a height above the ground of 0.85 m using linear MPC (without learning updates; F, H, z all zero), and enabled learning. After some maneuvering, the parameter estimates diverged, hitting their bounding limits. At this time, the quadrotor's altitude dropped sharply, but the quadrotor did not contact the

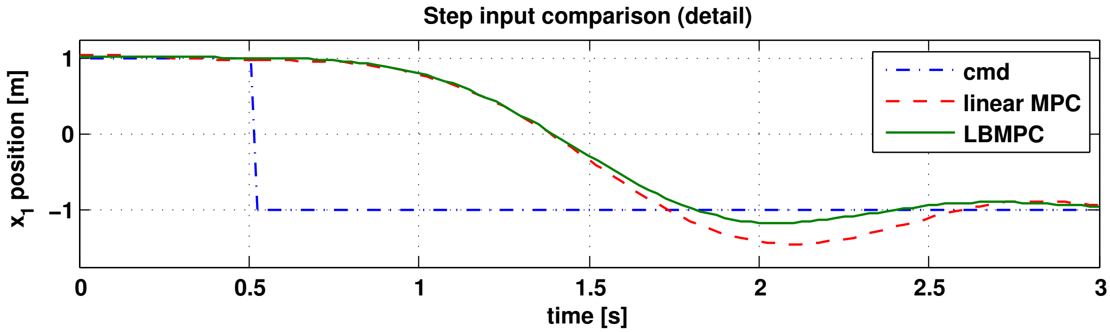


Figure 4.3: Step response for linear MPC with nominal model and LBMPC with learned model. The reference command is the dotted blue line. The LBMPC response here is from the 4th step command after enabling learning.

ground, and ended up in a stable hover approximately 0.1 above the ground (see Fig. 4.4). The optimization found a feasible solution throughout, and this demonstrates that even when the oracle degrades the learned model with respect to the nominal one, the system can remain safe and stable.

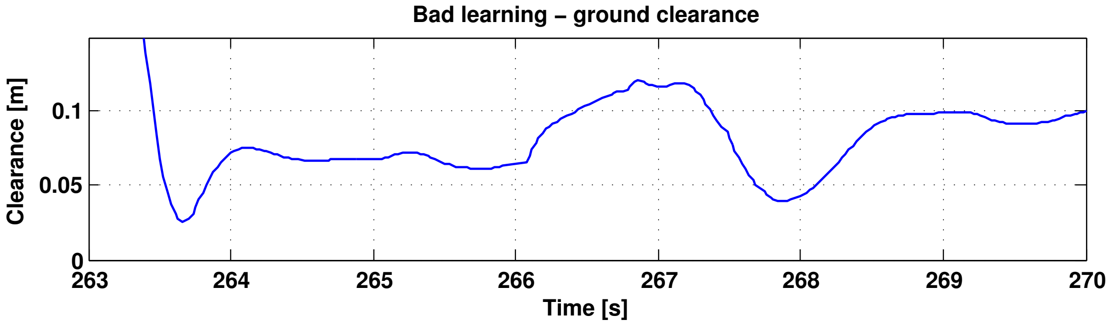


Figure 4.4: Safety is maintained even if parameter learning goes awry.

4.5.4 Precise maneuvering: ball catching

In this experiment, we tested the dynamic performance of the quadrotor using LBMPC on the challenging robotic demonstration task of catching a ball thrown by a human, before it hits the ground, when the ball has an *a priori* unknown trajectory. The task is similar to that in some recent work at ETH Zürich (Muller, Lupashin, and D'Andrea, 2011; Hehn and Andrea, 2012), although the control approach here is considerably different.

We equipped the quadrotor with a simple plastic cup, with a circular opening of radius 0.065 m, directly above the main body. The quadrotor is programmed to hover in place at a fixed altitude of 0.5 m above the ground. A command is issued to ready the quadrotor to catch the ball. Next, the ball, which has a mass of 6 g and a diameter of 33 mm, is tossed towards the robot by hand.

We designed an estimation and prediction scheme to estimate the ball's state (taken as its position and velocity in the inertial frame) and use this estimate to predict where the ball's trajectory will intercept the horizontal plane in which the quadrotor is hovering, to derive a position to command the quadrotor to. Details of the ball model, state estimation and prediction can be found in Appendix B.

The ball catching task is challenging because the quadrotor must arrive quickly and accurately at the location where the ball is predicted to be. Given the constraints of the experiment room, even for a ball thrown high the quadrotor has roughly 1 second from the time that the initial $\hat{x}_c$ are available to when the ball actually crosses the plane. The estimates of x_c must be accurate enough from the beginning that that the quadrotor is not commanded initially in the wrong direction, thus losing ground when the estimate later improves. Furthermore, when the quadrotor is accelerating, the vehicle is tilted and so the effective "catch zone" for the ball is reduced compared to when the quadrotor is stationary; this favors an approach in which the quadrotor can reach the destination and stabilize quickly.

Fig. 4.5 shows data from a typical experiment. We were able to achieve a very high rate of successful catches–over 90%. Fig. 4.6 is a photograph taken during one experiment, when the quadrotor is about to catch a ball thrown by a human. The vast majority of misses were also very close, within one or two ball diameters of the edge of the cup. We elected not to perform a more well-controlled study of success rates because this would require developing a repeatable ball-throwing device. At this stage, we believe that it would be more interesting to investigate a more detailed nonlinear model for the ball's dynamics. Indeed we observed the effects of the Magnus force, which caused a noticeable curvature in the ball's path. We attempted to throw the ball in a similar fashion each time, but some variable amount of spin (usually topspin from the underhanded throw) is induced on each throw. Since a number of important details differ from those in the aforementioned related work at ETH Zürich, a direct quantitative comparison would not be meaningful—however we note that qualitatively the success rates of our approach and those reported in (Muller, Lupashin, and D'Andrea, 2011) on their respective tasks are similar.

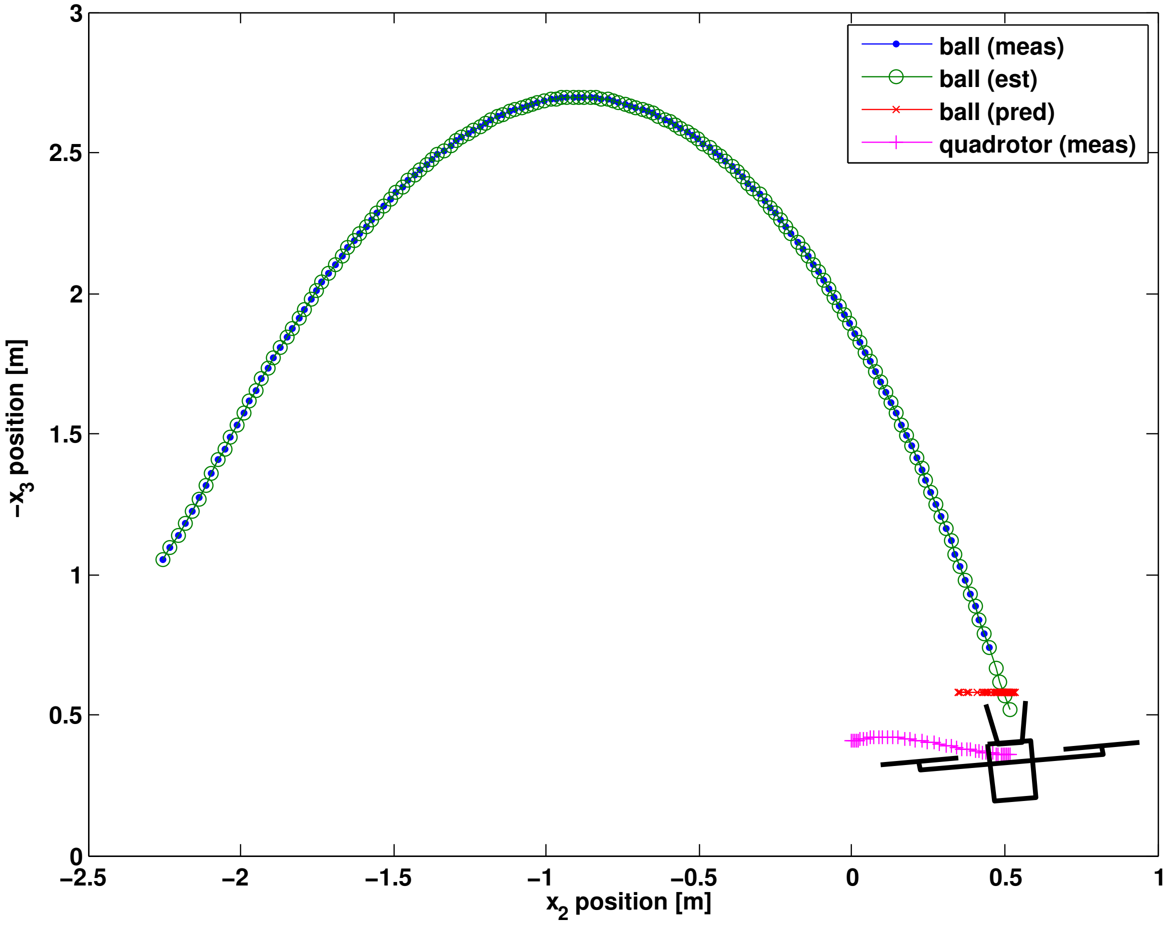


Figure 4.5: Measurements and EKF estimates of the ball's position throughout its trajectory, the estimated final position of the ball, and the trajectory of the quadrotor body frame $\mathbf{F}_B$ are shown.

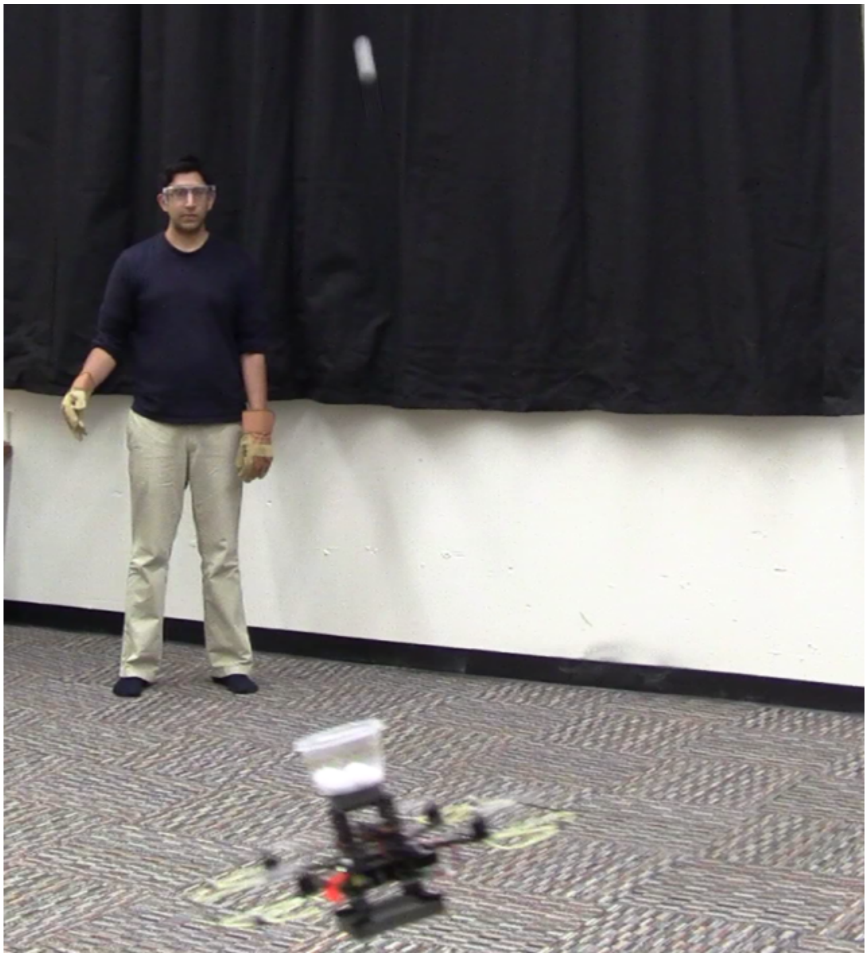


Figure 4.6: "Ball catching" experiment. The quadrotor, controlled using LBMPC, is about to catch a ball. Video from the experiments can be viewed on-line: `http://hybrid.eecs.berkeley.edu/~bouffard/research.html`.

4.6 Conclusions

We have described the design and implementation of a modified LBMPC controller for a quadrotor helicopter. The controller we implemented runs entirely on-board the quadrotor. We have shown the results of experiments that demonstrate some of the performance improvements that LBMPC can enable, as compared to standard MPC. These include robustness to mis-learning, improved dynamic response, and ability to learn physically interpretable effects (ground effect). We also showed the successful use of the quadrotor with the on-board LBMPC controller in a challenging robotic task in which the quadrotor catches balls thrown by a human.

Future work will examine whether the special structure of the MPC problem could enable improvements in computation time. Some initial work in this direction is described in Chapter 5. Also, the issues inherent in computation of the invariant set with polytopes which led to our current implementation using an overapproximation of the set should be addressed. One possible avenue would be to employ and ellipsoidal underapproximation and solve the resulting quadratically-constrained quadratic program (QCQP) (Zeilinger, 2011).

Chapter 5

Exploiting Structure for Fast LBMPC Solutions

The content of this chapter has in part been included in a conference submission currently under review (Zhang et al., 2013).

5.1 Introduction

A variety of methods exists for solving a QP-MPC. Explicit MPC (Bemporad et al., 2002; Tondel, Johansen, and Bemporad, 2001; Mariethoz, Domahidi, and Morari, 2009) computes a lookup table that gives the optimal control as a function of the initial states. It is well-suited for situations where the QP-MPC is time-invariant. However, the number of entries in the table can grow exponentially with horizon, input, and state dimensions. An alternative approach is to use a QP optimization solver (e.g., (Gill et al., 1986; Gill, Murray, and Saunders, 2002; Ferreau, Bock, and Diehl, 2008)) at each time step. Specialized solvers that can exploit the sparsity of QP-MPC have computational complexity that scales linearly in the prediction horizon of the QP-MPC (Wang and Boyd, 2010; Rao, Wright, and Rawlings, 1998), as opposed to non-sparse solvers that scale cubicly. Other approaches include automatic code generation using problem-tailored solvers (Mattingley and Boyd, 2012; Domahidi et al., 2012), and combining explicit and online MPC (Zeilinger, Jones, and Morari, 2011).

If the system dynamics are linear, the state and input constraints are polyhedral, the cost function is quadratic, and the statistical method provides linear model updates, then LBMPC can be described by a QP (QP-LBMPC). These conditions are common in many engineering problems. This paper describes an optimization solver that is specific to such QP-LBMPC problems. We show in (Zhang et al., 2013) that QP-LBMPC has sparsity structure similar to QP-MPC. Hence, sparse solvers can be designed that computationally scale well when solving LBMPC. The proposed primal-dual infeasible start interior point method (PD IIPM) based on Mehrotra's predictor-corrector scheme (Mehrotra, 1992) was implemented in C++, and named LBmpcIPM. LBmpcIPM exploits the sparsity structure of QP-LBMPC, and computationally scales well in the length of the prediction horizon. Theoretical results in (Zhang et al., 2013) are supported by simulations that empirically confirm that the solving time of our solver does indeed scale linearly in the prediction horizon. Furthermore, we present experimental results on a quadrotor helicopter testbed, where we show the robustness and scaling properties of the solver.

In this chapter we will not go into any detail on the PD IIPM scheme itself or implementation details thereof but rather will focus on the simulation and experimental results since the author of this report contributed more directly to those.

5.2 Experimental and simulation results

In this section, we empirically compare LBmpcIPM with two dense active set solvers (LSSOL v1.05-4 and qpOASES v3.0beta) (Gill et al., 1986; Ferreau, Bock, and Diehl, 2008). We begin with simulations on a model of a quadrotor helicopter. Simulations show that the computation time of the sparse PD IIPM does indeed scale linearly in the prediction horizon N. This is in contrast to the dense active set solvers whose computation times are well-known (Wang and Boyd, 2010; Rao, Wright, and Rawlings, 1998) to scale cubicly in the horizon length N. These simulations are followed by experiments on a quadrotor helicopter, in which we empirically compare three solvers. We ran these solvers in real time using the computer on-board the quadrotor helicopter which is slow in comparison to a desktop computer.

The quadrotor model that we used is the same one described previously in Sec. 4.2.

5.2.1 Computational scaling in horizon length

We conducted a simulation in which the quadrotor was commanded to move from a height of 2 m above the ground to a height of 1 m. The reason that LBMPC can be useful in similar scenarios is that complex aerodynamic behavior leads to a change in the thrust of the helicopter when it approaches the ground. LBMPC allows the designer to explicitly specify that the model can change in this manner and then leverage statistical tools to learn and compensate for the effect of this phenomenon, as we showed earlier in Chapter 4. We used nonzero values of $F(\beta)$, $H(\beta)$ (see 4.17) to represent learning that has taken place and has identified changes in the helicopter physics.

We ran the simulations on three different computers of varying computational power and architecture. The horizon size N was ranged from 5 steps to 240 steps, which represents 0.125 s to 6 s of horizon time because of the 0.025 s sampling period of the model. Specifications for the three computers are shown in Table 5.1. The third computer is a small form factor computer-on-module (CoM) which runs onboard the quadrotors used in our laboratory testbed (see Sec. A.1).

Table 5.2 lists the simulation results, and Fig. 5.1 shows the same information in plot form, with linear trendlines. For a given CPU the solution time is strongly linear in the prediction horizon N. This is what is expected for a sparse interior point solver that exploits the special structure of an MPC problem, and stands in contrast to dense solvers that scale cubicly. When the helicopter is in flight, its processor must handle additional overhead due to processes like measurement communication and file storage. And so the solve times reported for the Atom CPU represent a lower bound on the solve times for when the helicopter is in flight.

Computer	Model	CPU	Cache	RAM	Linux
i7	Lenovo T410 laptop	2.67 GHz	4 MB	8 GB	64-bit
Core2	Dell Precision 390 desktop	2.4 GHz	4 MB	2 GB	32-bit
Atom	AscTec Atomboard	1.6 GHz	512 kB	1 GB	32-bit

Table 5.1: Different computers on which we tested the LBmpcIPM solver.

N	i7	Core2	Atom
5	1.1	2.0	9.8
10	2.2	4.0	20.1
15	3.2	6.0	30.3
30	6.3	12.1	60.7
60	12.4	24.5	121.9
120	26.0	49.9	245.7
240	54.1	105.4	491.7

Table 5.2: Average time [ms] to solve problem for one timestep.

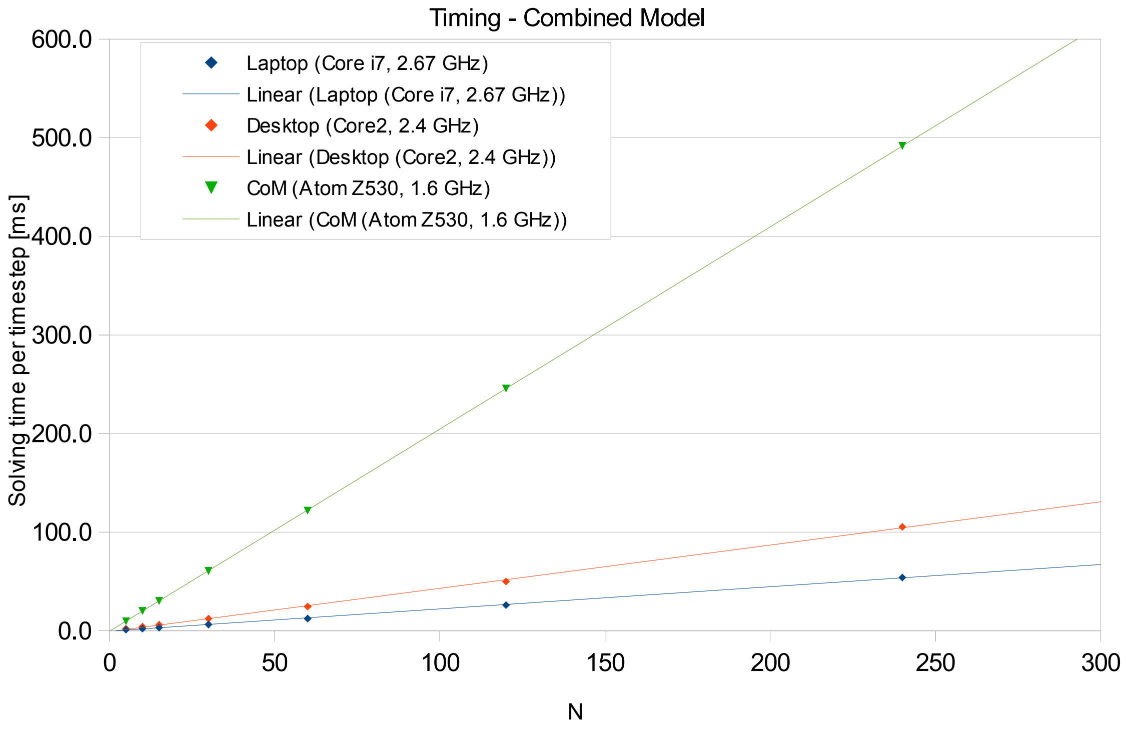


Figure 5.1: Plot of average solving time [ms] vs. horizon length N. Trendlines emphasize the strong linearity of the solving time.

5.2.2 Experimental comparison

LBMPC controllers implemented using LBmpcIPM, LSSOL, and qpOASES were compared on a quadrotor helicopter testbed. This experiment is an interesting comparison of the different solvers because (i) the processor onboard the helicopter is slow in comparison to a desktop computer, (ii) the optimization problem to compute the control must be computed within 25 ms to enable the real time control, and (iii) the quadrotor has constraints placed on its state and inputs that correspond to physical constraints such as not crashing into the ground. We used a horizon of $N = 5$ because this was the largest horizon in which all the solvers could reliably terminate their computations during the 25 ms sampling period for computing the control value. Note that this is in contrast to the horizon of $N = 15$ that was used with the LSSOL solver in previous experiments applying LBMPC to the quadrotor (Chapter 4). This means that the benefits of the linearly-scaling computational complexity are not apparent in these experiments, nor are the benefits of a longer MPC horizon. However, it is worth noting that the current on-board quadrotor computer (which dates from 2009) could be replaced with another of similar size and power requirements. It would be compatible with our quadrotor platform (Bachrach et al., 2012), yet with about an order of magnitude better performance (Meier et al., 2012), which we can predict should enable, for LBmpcIPM, horizons around $N = 30$. An experiment was conducted in which the helicopter was commanded to, starting from a stable hover condition, go left 1 m and then go right 1 m. This was repeated 10 times in quick succession. The same learning described in Chapter 4 was enabled for this experiment.

A plot of a representative step input in which the quadrotor was commanded to go from left to right is shown in Fig. 5.2. The position of the helicopter when using the LSSOL solver is shown in solid blue, the difference between the trajectories of the LSSOL and LBmpcIPM solvers is shown in dashed red, and the difference between the trajectories of the LSSOL and qpOASES solvers is shown in dash-dotted green. We used the trajectory of the LSSOL solver as the reference trajectory, because this was the solver used for our previous experiments in Chapter 4. As can be seen in the plots, the difference in trajectories is within 6.5 cm for LBmpcIPM and 9.8 cm for qpOASES. These differences are within a range that would be expected even between runs of the same trajectory using the same solver due to complex aerodynamic fluctuations that occur during a flight. They indicate that the different solvers are giving the same performance.

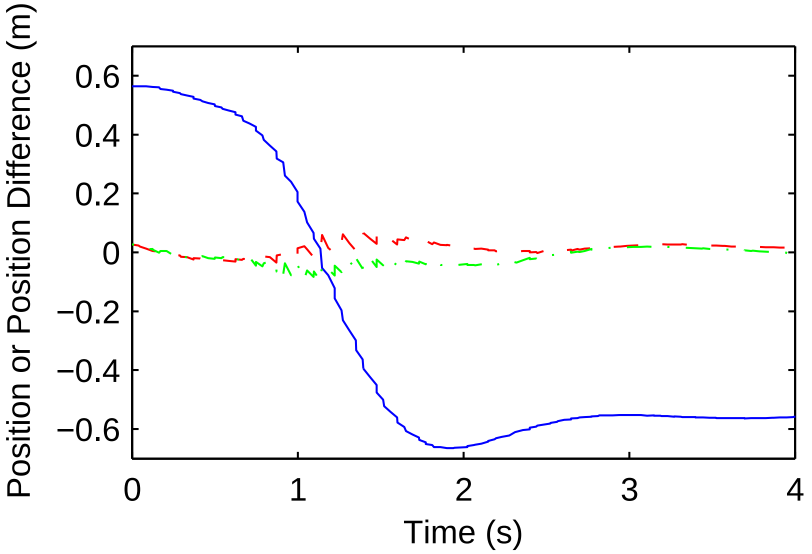
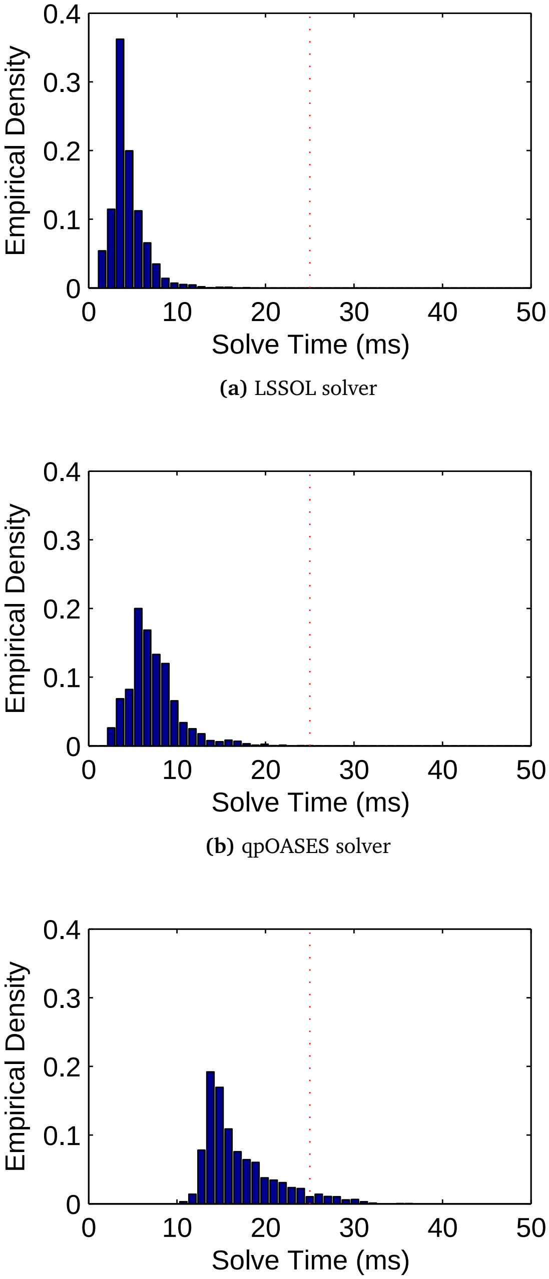


Figure 5.2: The step response trajectory of the quadrotor helicopter flown using LBMPC solved with LSSOL (solid blue), the difference between the trajectories of the helicopter when flown with the LSSOL versus the LBmpcIPM solver (dashed red), and the difference between the trajectories of the helicopter when flown with the LSSOL versus the qpOASES solver (dash-dotted green).

Histograms that show the empirical densities of the solve times for the three solvers can be seen in Fig. 5.3. The two dense active set solvers have a lower variance of solve times, and this lower variance is important because it means that the solver is able to finish its computations under the time limit imposed by the 25 ms sampling period. In contrast, the LBmpcIPM method has a higher variance and so the solve times do exceed the 25 ms limit a small percentage of the time. Lastly, we note that the LBmpcIPM solver is able to provide good quality solutions. The norm of the primal residue is 1.4×10^{-7} on average, with a minimum of 3.5×10^{-11} and a maximum of 4.8×10^{-6}. Similarly, the norm of the dual residue is 3.1×10^{-8} on average, with a minimum of 1.7×10^{-11} and a maximum of 1.8×10^{-6}.

5.3 Conclusions

We have shown that a sparse PD IIPM can efficiently solve QP-LBMPC problems. We carried out both numerical simulations and experiments onboard a quadrator helicopter. Results from these experiments and simulations indicate QP-LBMPC problems can be solved robustly, and that the computational complexity scales linearly in the prediction horizon, making QP-LBMPC attractive for systems with large horizons. The LBMPC problem has additional structure because of the similarity between the dynamics of the learned (with states $\tilde{x}$) and nominal model (with states $\bar{x}$), and this structure is not typical in linear MPC problems. It may be possible to leverage this structure to provide improvements in the solve time.



(a) LSSOL solver

(b) qpOASES solver

(c) LBmpcIPM solver

Figure 5.3: Empirical densities of solve times on quadrotor helicopter for different optimization algorithms are shown. The vertical dashed red line at 25ms indicates the threshold beyond which greater solve times are too slow to be able to provide real time control.

Chapter 6

Conclusions

6.1 Summary

In this report we have described how we applied two types of MPC, explicit MPC and learning based MPC (LBMPC), to control of the flight of a quadrotor helicopter.

We designed controllers based on the explicit MPC technique, which leverages the special form of MPC control laws to effectively pre-compute the solutions to the MPC optimization problem, such that on-line controller implementation is effectively little more than a look-up table and can run very quickly even given the limited computing capability on-board the quadrotor.

We also designed an LBMPC-based controller. LBMPC allows the advantages of the MPC technique, particularly the direct encoding of design requirements as cost function and constraints in an optimization problem, to be combined with the potential performance benefits that can be obtained by applying statistical learning techniques on-line. The latter techniques, used alone, could result in a compromise of safety properties but LBMPC reconciles this by treating learning as an update to a given nominal model for which uncertainty bounds are known. As a result, even in the case in which the learning fails, safety (as encoded in the MPC constraints) is maintained. We described in detail our design and implementation of an LBMPC controller for the quadrotor, and the experiments we performed to demonstrate its effectiveness.

For both the explicit MPC and LBMPC controllers, a model of the quadrotor's dynamics is required. We have described our development of such a model, based on a linearized (near hover conditions) double-integrator, and an empirical identification of the parameters of this model. In the explicit MPC controller, this model is taken effectively as a truth model, though the control synthesis does allow for 'state noise', to represent modeling uncertainties. In the LBMPC controller this model represents the 'nominal' model.

Finally, we have shown that a sparse PD IIPM can efficiently solve QP-LBMPC problems, using a new solver developed specifically for such problems. We carried out both numerical simulations and experiments onboard a quadrator helicopter. Results from these experiments and simulations indicate QP-LBMPC problems can be solved robustly, and that the computational complexity scales linearly in the prediction horizon, making QP-LBMPC attractive for systems with large horizons.

6.2 Future Work

There are several potential directions for future investigations, in terms of the modeling and system identification, explicit MPC controllers, and LBMPC controllers.

The modeling we performed is fairly rudimentary and limits the regime of operation of the quadrotor. It could not be used, for example, for extremely aggressive or aerobatic maneuvers. This would require either a nonlinear model, or a series of linear models based on linearized operating points. Nonlinear models could include terms to account for the ground effect and other aerodynamic factors (Hoffmann, Waslander, and Tomlin, 2009).

In part because of the slowness of the design process owing to the computation time for even relatively short horizon explicit MPC controllers, we were not able to iterate many times on the tuning of these controllers. We therefore suspect that our tuning of these controllers could be improved. They would of course also benefit from any improvement in system identification.

Combining ideas from switched PWA control of quadrotors (Alexis, Nikolakopoulos, and Tzes, 2011) with LBMPC would one possible natural direction. Note that for the LBMPC technique the optimization problem must be solved on-line because of the oracle updates to the cost function–an explicit (precomputed) scheme such at that used in (Alexis, Nikolakopoulos, and Tzes, 2011) would not be applicable.

The LBMPC problem has additional structure because of the similarity between the dynamics of the learned (with states $\tilde{x}$) and nominal model (with states $\bar{x}$), and this structure is not typical in linear MPC problems. It may be possible to leverage this structure to provide improvements in the solve time beyond those already obtained (at least for longer horizon lengths) with our sparse PD IIPM solver.

Finally, since we were not able to compute the maximal output admissable disturbance invariant set because of the dimensionality of our state space and instead used an overapproximation, the safety properties of LBMPC were verified empirically but not theoretically. Obtaining theoretical guarantees would require an underapproximation of this set, which may require the use of non-polytopic techniques. For example, we could investigate the use of ellipsoidal approximations; the main difference would be the requirement to solve a QCQP rather than a QP.

Appendix A

Experimental testbed description

A.1 Overview

Our experimental testbed consists of the following major components:

- AscTec Pelican quadrotor helicopter
- Vicon MX motion capture system
- supporting hardware and software
- safety systems

Fig. A.1 shows a diagram of our experimental testbed, including the main components and their data flow interconnections. We describe each major system component in the following sections.

A.2 Pelican quadrotor

The main element of the system is a quadrotor UAV, based on the popular Pelican vehicle from Ascending Technologies (AscTec for short)[1]. Fig. A.2 shows an image of a Pelican quadrotor equipped with several additional sensors; in the work in this report, we used no additional on-board sensors as our focus was on modeling and control. Several research groups are now using this vehicle for various types of experiments. Our Pelican is equipped with the optional on-board computer which includes a 1.6 GHz Intel Atom N270 CPU, 1 GB of RAM, an 8 GB solid state (micro-SD card) disk, and wi-fi communications. The MPC controllers described in this report run as Robot Operating System (ROS, (2011)) "nodelets" within a system based on a software framework for quadrotor operations developed in the Hybrid Systems Lab (Bouffard, 2011).

A.2.1 Control System

The Pelican is supplied with a proprietary controller (the "AutoPilot") for attitude angles and thrust. According to AscTec, this controller runs at 1 kHz on one of two ARM7 chips (the "low-level" (LL) chip) located on the AutoPilot board. There is another "high-level" (HL) ARM7 chip that does not figure in the configuration used in these experiments–we are effectively using it to forward commands to, and telemetry from, the LL chip[2]. Our MPC controller operates on the Atom CPU, and communicates with the LL controller over a serial port link, via the HL controller. The LL controller is given setpoints for roll, pitch, yaw rate, and thrust over this link at 40 Hz. The same link also provides telemetry data at various rates.

[1] http://www.asctec.de/

[2] For this we currently use the `asctec_mav_framework` ROS stack (http://www.ros.org/wiki/asctec_mav_framework); in prior versions we communicated directly with the LL controller using the `asctec_drivers` stack (http://www.ros.org/wiki/asctec_drivers).

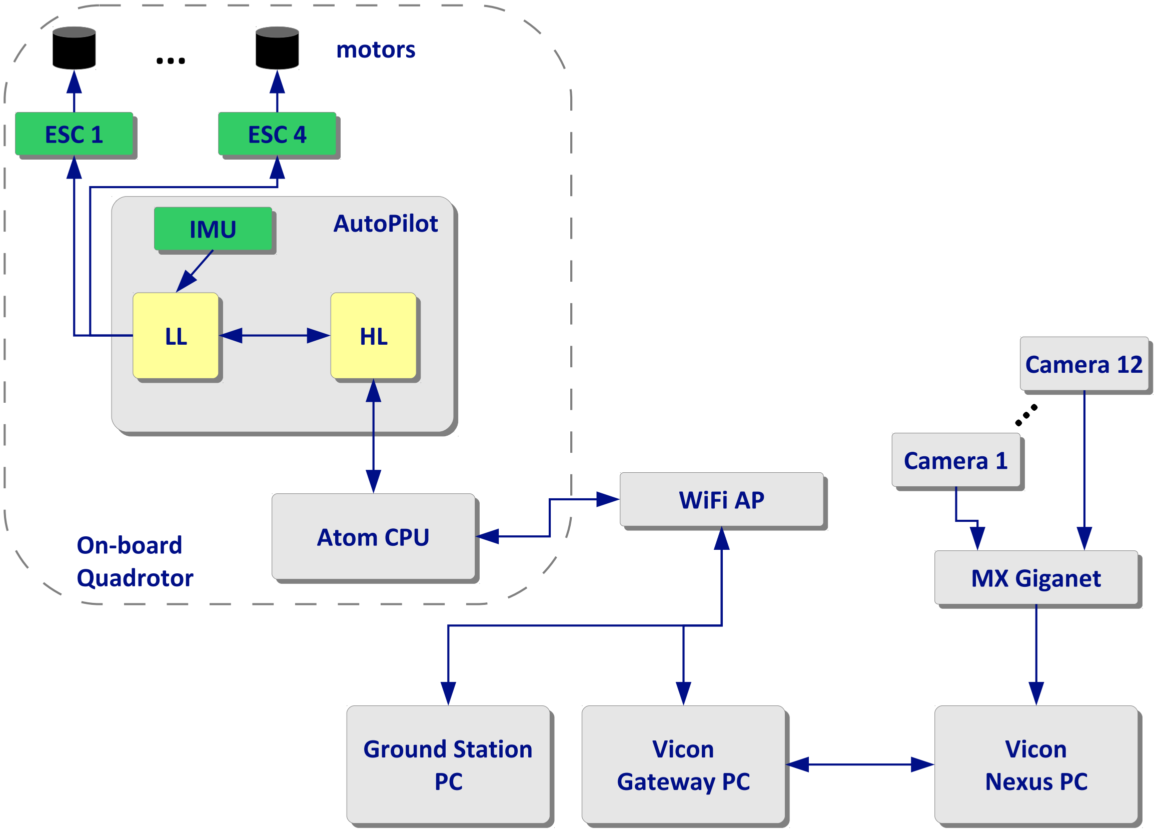


Figure A.1: High-level system diagram showing the main components involved in the control of the quadrotor in our experimental setup. The items within the dashed line are all on-board the quadrotor as it flies. Our controllers run on the Atom CPU.

Note that while the Pelican is provided with a 3D magnetometer (3DMAG) which can be used by the AutoPilot to derive absolute yaw angle, in practice for indoor operation magnetic sources such as re-bar cause the 3DMAG to have significant drift. By disconnecting the 3DMAG, the AutoPilot defaults to controlling only the yaw rate, directly sensed by its yaw gyro. Over time, this can result in yaw position drift, but this is corrected in our setup with a linear (PID) controller using Vicon data to measure the current yaw and estimate yaw rate. For the purposes of this project we assume this controller maintains yaw at a fixed angle–in practice the controller indeed keeps yaw very steady.

A.3 Motion capture system

Experiments are conducted in a laboratory environment equipped with a Vicon[3] MX motion capture system. The system consists of an array of 12 Vicon T40 cameras arranged on custom rails near the ceiling of a room dedicated to aerial robotics experiments[4], two Vicon Giganet MX multiplexing units, a control PC running Vicon Nexus application software, and a gateway PC running software that provides the Vicon based measurements to the rest of the system.

[3] http://www.vicon.com

[4] The explicit MPC work described in Chapter 3 was performed using an earlier incarnation of the system; at that time only 8 cameras were used, covering a smaller overall flight volume in a room that served as both laboratory and graduate student office space.

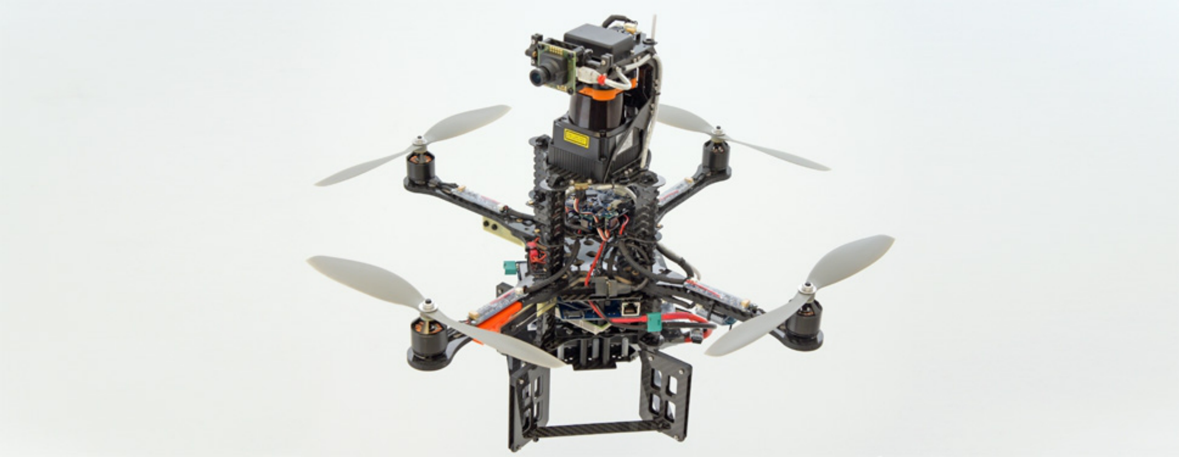


Figure A.2: Ascending Technologies *Pelican* quadrotor (photo courtesy Ascending Technologies)

The Vicon system provides the full rigid-body position and orientation of multiple rigid bodies and/or reflective markers at a rate of 120 Hz. To use these measurements as input to our estimation algorithms, we wrote software[5] employing the Vicon DataStream software development kit (SDK) that provided the measurements to the software on-board the quadrotor via a ROS topic.

We use this same system to obtain measurements of the 3D position of the ball in the "ball catching" experiment.

A.4 Safety systems

The physical safety of researchers is always of greatest importance in any experimental work, and experiments with flying robots are no exception. We take several common-sense precautions when flying the quadrotors in the indoor lab:

- Floor-to-ceiling netting separates the area where the researchers control the quadrotor, and any spectators are required to remain behind the netting during flight experiments. Researchers also stay behind this netting during experiments unless necessary for the experiment at hand (e.g. the "ball catching" experiment described in this report).
- If a researcher needs to be in the flight volume during experiments, s/he wears heavy gloves to protect hands.
- Everyone (both researchers and any spectators present) in the flying lab wears eye protection.
- Whenever flight experiments are underway, a sign is posted on the outside of the flying lab door which clearly indicates this fact and asks visitors to knock before entering (the door opens to the flying-volume side of the protective net).
- Whenever the quadrotor is "live" (rotors engaged), a researcher is ready with a remote control to take over flight of the quadrotor if need be.

Besides these common-sense, low-tech safety precautions we are also in the process of adding a more sophisticated one. We have discovered that the Pelican AutoPilot has a failure mode in which it may not be possible to issue a remote command to shut down the rotors and this has on occasion required us to

[5] Our original software was called `vicon_mocap` and released as BSD-licensed open source (http://www.ros.org/wiki/vicon_mocap). It was later forked by researchers at ETH Zürich, extended to allow multiple bodies and renamed `vicon_bridge` (http://www.ros.org/wiki/vicon_bridge). Subsequently we adopted use of this forked version.

manually disconnect the battery while the rotors were running at maximum speed, a dangerous prospect to say the least. We are thus testing a remote “kill switch” we have developed which incorporates a watchdog principle: A receiver unit on the quadrotor allows power to flow from the battery only if it continuously receives a keep-alive beacon from a transmitter unit. The transmitter unit is also equipped with a button that allows a researcher to command an immediate power cut at any time.

Appendix B

Modeling and estimation for a ball in free flight

A key part of the ball catching experiment is predicting the ball's future trajectory based on an estimate of its current state. To that end, we modeled the ball and used this model in an estimator to predict the ball trajectory. The following sections describe first the model, and then the estimator used.

B.1 Modeling

The trajectory of the ball in free flight is governed by the action of gravity, drag due to air resistance, the Magnus effect, buoyancy, and added (or "virtual") mass (Andersson, 1988; Yingshi et al., 2010). The Magnus effect induces a force perpendicular to both the velocity and the spin axis of the ball, thus causing its trajectory to curve. This force appears to have an effect on our trajectory predictions, based on study of the ball's trajectory. However, a nonlinear EKF estimate that incorporated the Magnus effect did not converge fast enough to provide accurate estimates, and subsequently we used a more straightforward Luenberger observer—which has been proven to converge (Yingshi et al., 2010)—with a model that neglects Magnus. The buoyancy force and the "added mass" on the other hand, are both small enough to neglect.

The gravity force is $F_G = mg \cdot \mathbf{x}_3$, and drag acts in a direction opposite to that of the ball's instantaneous velocity vector $V := (\dot{x}_{b,1}, \dot{x}_{b,2}, \dot{x}_{b,3})^T$ and has magnitude proportional to the square of the ball's velocity, thus $F_D = -\frac{1}{2}\rho C_D \|V\| V$, where C_D is a drag coefficient that is typically determined empirically and ρ is the density of the air.

Let $x_b = (x_{b,1}, \dot{x}_{b,1}, x_{b,2}, \dot{x}_{b,2}, x_{b,3}, \dot{x}_{b,3})^T \in \mathbb{R}^6$ represent the state (position and velocity) of the ball expressed in $\mathbf{F}_I$. The discrete-time (time step δt_b) dynamics update is

$$x_b^+ = F_b(x_b) = \begin{bmatrix} A_b & 0 & 0 \\ 0 & A_b & 0 \\ 0 & 0 & A_b \end{bmatrix} x_b + \begin{bmatrix} 0 \\ \delta t_b F_{D,1}(x_b) \\ 0 \\ \delta t_b F_{D,2}(x_b) \\ \frac{\delta t_b^2}{2} g \\ \delta t_b \left(g + F_{D,3}(x_b)\right) \end{bmatrix} \tag{B.1}$$

where $A_b = \left[\begin{smallmatrix} 1 & \delta t_b \\ 0 & 1 \end{smallmatrix}\right]$, i.e., a discretized double integrator in each axis. Note that we neglect the small contribution of the drag force to the position update. We measure only the position of the ball; the

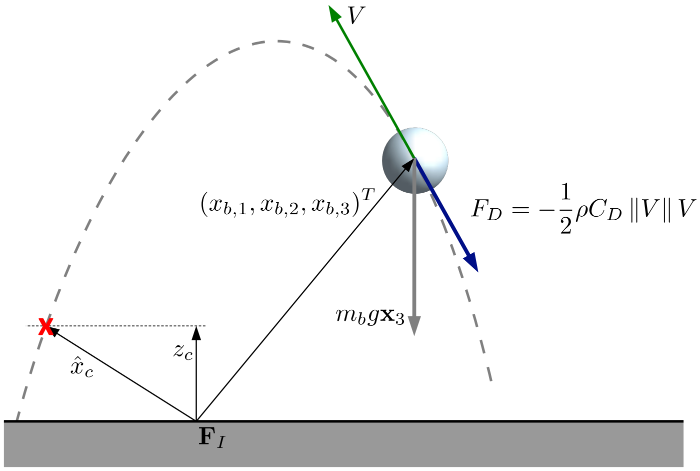

V
$(x_{b,1}, x_{b,2}, x_{b,3})^T$
$F_D = -\frac{1}{2}\rho C_D \|V\| V$
$m_b g \mathbf{x}_3$
z_c
$\hat{x}_c$
$\mathbf{F}_I$

Figure B.1: Diagram of the ball in flight. The ball's trajectory is shown by the dashed line. $\mathbf{F}_I$ is the inertial frame; $(x_{b,1}, x_{b,2}, x_{b,3})^T$ is the position vector of the ball in this frame. The ball's instantaneous velocity vector V is shown in green; drag F_D (blue arrow) acts opposite V and the force of gravity $m_b g \mathbf{x}_3$ acts downwards. The height z_c represents the altitude at which we attempt to catch the ball (based on the quadrotor's altitude), and $\hat{x}_c$ is the predicted intersection of the ball's trajectory and the plane defined by z_c.

output equation is

$$\begin{aligned} y_b &= C_b x_b \\ &= \begin{bmatrix} c_b & 0 & 0 \\ 0 & c_b & 0 \\ 0 & 0 & c_b \end{bmatrix} x_b \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} x_b. \end{aligned}$$

B.2 Estimation and prediction

B.2.1 State estimation

The ball is covered with reflective tape so that the Vicon system can track it. The measurements of the ball's position are fed into a Luenberger observer that uses a nonlinear model incorporating a quadratic drag term for the state prediction step, and a linear correction step. We initialize the observer's velocity estimate using a finite difference estimate from two successive measurements to speed up the observer's convergence.

We found that a simple Luenberger observer, with nonlinear state update based on B.1 and linear corrections

$$\hat{x}_b^+ = \hat{x}_b + K_b \left(y_b - C_b \hat{x}_b\right),$$

performed adequately for the task. Table B.1 shows the values used for the parameters in the ball model and state estimator. The value of $K_b = \mathbf{blkdiag}\left\{k_b, k_b, k_b\right\}$ corresponds blocks k_b which are the gain matrix of a time-invariant Kalman filter for a discrete-time double integrator in each axis. That is, $k_b = \left(c_b^T X c_b + R\right)^{-1} c_b^T X A_b$, where X is the solution to the discrete algebraic Riccati equation,

$$A_b^T X A_b - X - A_b^T X c_b \left(c_b^T X c_b + R\right)^{-1} c_b^T X A_b + Q = 0,$$

and where the weighting matrices were selected as $Q = \mathbf{diag}\left\{0.0001, 0.25\right\}$, $R = 0.0001$.

Symbol(s)	Value	Description
ρC_D	0.09	Air density times drag coefficient
K_b	$\begin{bmatrix} 0.9488 & 0 & 0 \\ 25.7982 & 0 & 0 \\ 0 & 0.9488 & 0 \\ 0 & 25.7982 & 0 \\ 0 & 0 & 0.9488 \\ 0 & 0 & 25.7982 \end{bmatrix}$	Linear correction gain matrix

Table B.1: Parameters for ball state estimation.

B.2.2 Prediction

Once 20 initial measurements have been processed (0.16 s), the state estimate is used to propagate the dynamics model forward to estimate the point $\hat{x}_c$ where the ball's trajectory will intersect the plane in which the quadrotor is hovering. The quadrotor's reference command is then set to $\hat{x}_c$, and it continues to track updates to $\hat{x}_c$ as these are generated in subsequent timesteps.

References

2011. "ROS (Robot Operating System)." http://www.ros.org. Willow Garage.

Abbeel, P., A. Coates, and A. Y. Ng. 2010. "Autonomous Helicopter Aerobatics through Apprenticeship Learning." *Intl. J. of Robotics Res.* 29 (13):1608–1639. URL http://ijr.sagepub.com/cgi/doi/10.1177/0278364910371999.

Albertini, Francesca and D. D'Alessandro. 2002. "Observability and forward-backward observability of discrete-time nonlinear systems." *Mathematics of Control, Signals, and Systems* 15 (4):275–290. URL http://www.springerlink.com/index/W2CFGP3GUX3PBU87.pdf.

Alexis, Kostas, George Nikolakopoulos, and Anthony Tzes. 2010a. "Constrained Optimal Attitude Control of a Quadrotor Helicopter subject to Wind-Gusts : Experimental Studies." In *American Control Conference (ACC)*, 2. Baltimore, MD, USA, 4451–4455.

———. 2010b. "Design and experimental verification of a Constrained Finite Time Optimal control scheme for the attitude control of a Quadrotor Helicopter subject to wind gusts." In *2010 IEEE International Conference on Robotics and Automation*, 2. IEEE, 1636–1641. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5509412.

———. 2011. "Switching model predictive attitude control for a quadrotor helicopter subject to atmospheric disturbances." *Control Engineering Practice* 19 (10):1195–1207. URL http://linkinghub.elsevier.com/retrieve/pii/S0967066111001262.

———. 2012. "On Trajectory Tracking Model Predictive Control of an Unmanned Quadrotor Helicopter Subject to Aerodynamic Disturbances." *Asian Journal of Control* :n/a–n/aURL http://doi.wiley.com/10.1002/asjc.587.

Alexis, Kostas, Christos Papachristos, George Nikolakopoulos, and Anthony Tzes. 2011. "Model predictive quadrotor indoor position control." In *2011 19th Mediterranean Conference on Control & Automation (MED)*. Ieee, 1247–1252. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5983144.

Andersson, Russell L. 1988. *A Robot Ping-Pong Player: Experiments in Real-Time Intelligent Control*.

Åström, Karl Johan and Björn Wittenmark. 1994. *Adaptive Control*. Prentice-Hall, 2nd ed.

Aswani, Anil, Patrick Bouffard, and Claire Tomlin. 2012. "Extensions of Learning-Based Model Predictive Control for Real-Time Application to a Quadrotor Helicopter." In *American Control Conference (ACC)*. Montreal, Canada.

Aswani, Anil, Humberto Gonzalez, S. Shankar Sastry, and Claire Tomlin. 2012a. "Provably Safe and Robust Learning-Based Model Predictive Control." *Automatica* :(to appear)URL http://arxiv.org/abs/1107.2487.

Aswani, Anil, Neal Master, Jay Taneja, David Culler, and Claire Tomlin. 2012b. "Reducing Transient and Steady State Electricity Consumption in HVAC Using Learning-Based Model-Predictive Control." *Proceedings of the IEEE* 100 (1):240–253. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5985456http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5985456.

Austen, Ian. 2011. "Libyan Rebels Reportedly Used Tiny Canadian Surveillance Drone." *The New York Times* :A11URL http://www.nytimes.com/2011/08/25/world/africa/25canada.html.

Bachrach, A., S. Prentice, R. He, P. Henry, a. S. Huang, M. Krainin, D. Maturana, D. Fox, and N. Roy. 2012. “Estimation, planning, and mapping for autonomous flight using an RGB-D camera in GPS-denied environments.” *The International Journal of Robotics Research* 31 (11):1320–1343. URL http://ijr.sagepub.com/cgi/doi/10.1177/0278364912455256.

Bauer, F L and C T Fike. 1960. “Norms and Exclusion Theorems.” *Numerische Mathematik* 2:137–141.

Bemporad, Alberto, Manfred Morari, Vivek Dua, and Efstratios N Pistikopoulos. 2002. “The explicit linear quadratic regulator for constrained systems.” *Automatica* 38 (1):3–20. URL http://linkinghub.elsevier.com/retrieve/pii/S0005109801001741.

Borrelli, F, A Bemporad, and M Morari. 2012. *Predictive Control for linear and hybrid systems*. (in preparation). URL http://www.mpc.berkeley.edu/mpc-course-material.

Bouabdallah, Samir and Roland Siegwart. 2007. “Full control of a quadrotor.” In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Sys.*, 1. IEEE, 153–158. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4399042.

Bouffard, P. 2011. “starmac-ros-pkg ROS repository.” http://www.ros.org/wiki/starmac-ros-pkg.

Bouffard, Patrick, Anil Aswani, and Claire Tomlin. 2012. “Learning-Based Model Predictive Control on a Quadrotor: Onboard Implementation and Experimental Results.” In *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*. Saint Paul, Minnesota, USA.

Bouffard, Patrick and Cameron Rose. 2011. “Zero-Offset MPC (δu Formulation) for a Quadrotor UAV.” {(UC Berkeley ME290J Project Report)}.

Boyd, Stephen and Lieven Vandenberghe. 2009. *Convex Optimization*. Cambridge University Press. URL http://www.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf.

Brockers, Roland, Patrick Bouffard, Jeremy Ma, Larry Matthies, and Claire Tomlin. 2011. “Autonomous landing and ingress of micro-air-vehicles in urban environments based on monocular vision.” In *Proceedings of SPIE*, vol. 8031. 803111–803111–12. URL http://link.aip.org/link/?PSISDG/8031/803111/1http://link.aip.org/link/PSISDG/v8031/i1/p803111/s1&Agg=doi.

Burri, Michael, Janosch Nikolic, Christoph Hurzeler, Gilles Caprari, and Roland Siegwart. 2012. “Aerial Service Robots for Visual Inspection of Thermal Power Plant Boiler Systems.” In *2nd Intl. Conf. on Applied Robotics for the Power Industry*. Zurich, Switzerland.

Chisci, L, JA Rossiter, and G Zappa. 2001. “Systems with persistent disturbances: predictive control with restricted constraints.” *Automatica* 37 (7):1019–1028. URL http://www.sciencedirect.com/science/article/pii/S0005109801000516.

Cutler, Mark. 2012. *Design and Control of an Autonomous Variable-Pitch Quadrotor Helicopter*. M.s. thesis, MIT.

Ding, Jerry, Eugene Li, Haomiao Huang, and Claire J Tomlin. 2011. “Reachability-based Synthesis of Feedback Policies for Motion Planning Under Bounded Disturbances.” In *IEEE International Conference on Robotics and Automation*. Shanghai, China, 2160–2165.

Domahidi, Alexander, Aldo U Zgraggen, Melanie N Zeilinger, Manfred Morari, and Colin N Jones. 2012. “Efficient Interior Point Methods for Multistage Problems Arising in Receding Horizon Control.” In *IEEE Conference on Decision and Control*. (to appear).

Ferreau, H.J., HG Bock, and M Diehl. 2008. “An online active set strategy to overcome the limitations of explicit MPC.” *International Journal of Robust and Nonlinear Control* 18 (8):816–830. URL http://onlinelibrary.wiley.com/doi/10.1002/rnc.1251/abstract.

Gilbert, E.G. and K.T. Tan. 1991. “Linear systems with state and control constraints: the theory and application of maximal output admissible sets.” *IEEE Transactions on Automatic Control* 36 (9):1008–1020. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=83532.

Gill, Philip E., Sven J. Hammarling, Walter Murray, Michael A. Saunders, and Margaret H. Wright. 1986. “User’s Guide for LSSOL (Version 1.0).”

Gill, Philip E., Walter Murray, and Michael a. Saunders. 2002. “SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization.” *SIAM Journal on Optimization* 12 (4):979–1006. URL http://epubs.siam.org/doi/abs/10.1137/S1052623499350013.

Gillula, Jeremy H and Claire J Tomlin. 2011. “Guaranteed Safe Online Learning of a Bounded System.” In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Sys. (IROS)*. San Francisco, CA.

Grimm, G. 2004. “Examples when nonlinear model predictive control is nonrobust*1.” *Automatica* 40 (10):1729–1738. URL http://linkinghub.elsevier.com/retrieve/pii/S0005109804001402.

Guenard, Nicolas, Tarek Hamel, and Laurent Eck. 2006. “Control laws for the tele operation of an unmanned aerial vehicle known as an x4-flyer.” In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE, 3249–3254. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4058899.

Hehn, Markus and Raffaello D Andrea. 2012. “Real-Time Trajectory Generation for Interception Maneuvers with Quadrocopters.” In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. URL http://www.idsc.ethz.ch/people/staff/hehn-m/hehn_dandrea_interception.pdf.

Hoffmann, G., D.G. Rajnarayan, S.L. Waslander, D. Dostal, J.S. Jang, and C.J. Tomlin. 2004. “The Stanford testbed of autonomous rotorcraft for multi agent control (STARMAC).” In *The 23rd Digital Avionics Systems Conference*. Salt Lake City, UT, USA: Ieee, 12.E.4–121–10. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1390847.

Hoffmann, Gabriel and Claire Tomlin. 2008. “Mobile Sensor Network Control using Mutual Information Methods and Particle Filters.” *IEEE Transactions on Automatic Control* URL http://hoffmann.stanford.edu/papers/infoControl.pdf.

Hoffmann, Gabriel M., Haomiao Huang, Steven L. Waslander, and Claire J. Tomlin. 2011. “Precision flight control for a multi-vehicle quadrotor helicopter testbed.” *Control Engineering Practice* 19 (9):1023–1036. URL http://linkinghub.elsevier.com/retrieve/pii/S0967066111000712.

Hoffmann, G.M., Haomiao Huang, S.L. Waslander, and C.J. Tomlin. 2007. “Quadrotor helicopter flight dynamics and control: Theory and experiment.” In *Proc. AIAA Guidance, Navigation, and Control Conference*. Hilton Head, SC: Citeseer, 2007–6461. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.90.4126&amp;rep=rep1&amp;type=pdf.

Hoffmann, G.M., S.L. Waslander, and C.J. Tomlin. 2006. “Distributed cooperative search using information-theoretic costs for particle filters, with quadrotor applications.” In *Proc. AIAA Guidance, Navigation, and Control Conf. and Exhibit*. Keystone, Colorado: Citeseer. URL http://hybrid.eecs.berkeley.edu/pubs/Hoffmann_AIAA_GNC06.pdfhttp://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.137.5661&amp;rep=rep1&amp;type=pdf.

———. 2009. “Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering.” *2009 IEEE International Conference on Robotics and Automation* :3277–3282URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5152561.

How, Jonathan P., Brett Bethke, Adrian Frank, Daniel Dale, and John Vian. 2008. “Real-time indoor autonomous vehicle test environment.” *IEEE Control Systems Magazine* 28 (2):51–64. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4472379.

Huang, A.S., Abraham Bachrach, Peter Henry, Michael Krainin, D. Maturana, Dieter Fox, and Nick Roy. 2011a. “Visual Odometry and Mapping for Autonomous Flight Using an RGB-D Camera.” In *15th Intl. Symp. on Robotics Research*. Flagstaff, AZ, USA. URL http://www.isrr-2011.org/ISRR-2011/Program_files/Papers/Huang-ISRR-2011.pdf.

Huang, Haomiao, Jerry Ding, Wei Zhang, and C.J. Tomlin. 2011b. “A differential game approach to planning in adversarial scenarios: A case study on capture-the-flag.” In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 1451–1456. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5980264.

Kolmanovsky, I. and E.G. Gilbert. 1998. "Theory and computation of disturbance invariant sets for discrete-time linear systems." *Mathematical Problems in Engineering* 4 (4):317–363. URL http://www.emis.ams.org/journals/HOA/MPE/Volume4_4/367.pdf.

Kvasnica, M, P Grieder, and M Baotic. 2004. "Multi-Parametric Toolbox (MPT)." URL http://control.ee.ethz.ch/~mpt/.

———. 2006. *[Users' Guide for] Multi-Parametric Toolbox (MPT)*. URL http://control.ee.ethz.ch/~mpt/downloads/MPTmanual.pdf.

Langson, W, I. Chryssochoos, S.V. Raković, and D. Q. Mayne. 2004. "Robust model predictive control using tubes." *Automatica* 40 (1):125–133. URL http://linkinghub.elsevier.com/retrieve/pii/S0005109803002838.

Leishman, J. G. 2006. *Principles of helicopter aerodynamics*. Cambridge University Press.

Limon, D., I. Alvarado, T. Alamo, and E.F. Camacho. 2008. "MPC for tracking piecewise constant references for constrained linear systems." *Automatica* 44 (9):2382–2387. URL http://linkinghub.elsevier.com/retrieve/pii/S0005109808001106.

———. 2010. "Robust tube-based MPC for tracking of constrained linear systems with additive disturbances." *Journal of Process Control* 20 (3):248–260. URL http://linkinghub.elsevier.com/retrieve/pii/S0959152409002169.

Ljung, L. 1979. "Asymptotic behavior of the extended Kalman filter as a parameter estimator for linear systems." *IEEE Trans. on Autom. Control* 24 (1):36–50. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1101943.

Lopes, Renato Vilela, Geovany Ara, and Yoshiyuki Ishihara. 2011. "Model Predictive Control applied to tracking and attitude stabilization of a VTOL quadrotor aircraft." In *21st International Congress of Mechanical Engineering*, 2008. Natal, RN, Brazil.

Lupashin, Sergei, Angela Schollig, Markus Hehn, and Raffaello D'Andrea. 2011. "The Flying Machine Arena as of 2010." In *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2970–2971. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5980308.

Maeder, Urban, Francesco Borrelli, and Manfred Morari. 2009. "Linear offset-free Model Predictive Control." *Automatica* 45 (10):2214–2222. URL http://linkinghub.elsevier.com/retrieve/pii/S0005109809002969.

Mariethoz, S., A. Domahidi, and M. Morari. 2009. "Sensorless explicit model predictive control of permanent magnet synchronous motors." In *Electric Machines and Drives Conference, 2009. IEMDC'09. IEEE International*. 1250–1257.

Mattingley, Jacob and Stephen Boyd. 2012. "CVXGEN: a code generator for embedded convex optimization." *Optimization and Engineering* 13 (1):1–27. URL http://www.springerlink.com/index/10.1007/s11081-011-9176-9.

Mehrotra, Sanjay. 1992. "On the implementation of a primal-dual interior point method." *SIAM Journal on optimization* 2 (4):575–601. URL http://epubs.siam.org/doi/abs/10.1137/0802028.

Meier, Lorenz, Petri Tanskanen, Lionel Heng, Gim Hee Lee, Friedrich Fraundorfer, and Marc Pollefeys. 2012. "PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision." *Autonomous Robots* 33 (1-2):21–39. URL http://www.springerlink.com/index/10.1007/s10514-012-9281-4.

Michael, N., Daniel Mellinger, Quentin Lindsey, and V. Kumar. 2010. "The GRASP multiple micro-UAV testbed." *Robotics & Automation Magazine, IEEE* 17 (3):56–65. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5569026.

Michael, Nathan, Shaojie Shen, Kartik Mohta, Yash Mulgaonkar, Vijay Kumar, Keiji Nagatani, Yoshito Okada, Seiga Kiribayashi, Kazuki Otake, Kazuya Yoshida, Kazunori Ohno, Eijiro Takeuchi, and Satoshi Tadokoro. 2012. "Collaborative mapping of an earthquake-damaged building via ground and aerial robots." *Journal of Field Robotics* 29 (5):832–841. URL http://doi.wiley.com/10.1002/rob.21436.

Muller, Mark, Sergei Lupashin, and Raffaello D'Andrea. 2011. "Quadrocopter ball juggling." In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 5113–5120. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6048148http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6048148.

Raffo, GV and MG Ortega. 2008. "MPC with Nonlinear H-infinity Control for Path Tracking of a Quad-Rotor Helicopter." In *Proceedings of the 17th World Congress*. 8564–8569. URL http://www.nt.ntnu.no/users/skoge/prost/proceedings/ifac2008/data/papers/2578.pdf.

Rakovic, Sasa V and Miroslav Baric. 2010. "Parameterized Robust Control Invariant Sets for Linear Systems: Theoretical Advances and Computational Remarks." *IEEE Transactions on Automatic Control* 55 (7):1599–1614. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5406128.

Rao, CV, SJ Wright, and JB Rawlings. 1998. "Application of interior-point methods to model predictive control." *Journal of optimization theory and Applications* 99 (3):723–757. URL http://www.springerlink.com/index/k5u61174vh53k777.pdf.

Richalet, J, A Rault, J L Testud, and J Papon. 1976. "Algorithmic control of industrial processes." In *Proceedings of the 4th IFAC symposium on identification and system parameter estimation*. URSS September, Tbilisi,, 1119–1167.

Richalet, J., A Rault, J.L. Testud, and J. Papon. 1978. "Model predictive heuristic control." *Automatica* 14 (5):413–428. URL http://linkinghub.elsevier.com/retrieve/pii/0005109878900018.

Sastry, S. Shankar and Marc Bodson. 1994. *Adaptive Control: Stability, Convergence, and Robustness*. Prentice-Hall.

Schwager, Mac, Brian J. Julian, Michael Angermann, and Daniela Rus. 2011. "Eyes in the Sky: Decentralized Control for the Deployment of Robotic Camera Networks." *Proc. IEEE* 99 (9):1541 – 1561. URL http://people.csail.mit.edu/schwager/MyPapers/SchwagerProcIEEE11Cameras.pdf.

Tedrake, R., I. R. Manchester, M. Tobenkin, and J. W. Roberts. 2010. "LQR-trees: Feedback Motion Planning via Sums-of-Squares Verification." *Intl. J. of Robotics Res.* 29 (8):1038–1052. URL http://ijr.sagepub.com/cgi/doi/10.1177/0278364910369189.

Tondel, P., T.A. Johansen, and A. Bemporad. 2001. "An algorithm for multi-parametric quadratic programming and explicit MPC solutions." In *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*. 1199–1204.

Vitus, Michael. 2012. *Stochastic Control Via Chance Constrained Optimization and its Application to Unmanned Aerial Vehicles*. Ph.d. thesis, Stanford.

Vitus, M.P., Vijay Pradeep, G. Hoffmann, S.L. Waslander, and C.J. Tomlin. 2008. "Tunnel-milp: Path planning with sequential convex polytopes." In *AIAA Guidance, Navigation, and Control Conference*. Honolulu, Hawaii, USA. URL http://hoffmann.stanford.edu/papers/GNC08_TunnelMILP.pdf.

Wang, Y. and S. Boyd. 2010. "Fast model predictive control using online optimization." *Control Systems Technology, IEEE Transactions on* 18 (2):267–278. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5153127.

Waslander, SL, GM Hoffmann, J.S. Jang, and C.J. Tomlin. 2005. "Multi-agent quadrotor testbed control design: Integral sliding mode vs. reinforcement learning." In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Sys.* IEEE, 3712–3717. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1545025.

Yingshi, Wang, Sun Lei, Liu Jingtai, Yang Qi, Zhou Lu, and He Shan. 2010. "A novel trajectory prediction approach for table-tennis robot based on nonlinear output feedback observer." In *Robotics and Biomimetics (ROBIO), 2010 IEEE International Conference on*. IEEE, 1136–1141. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5723488.

Zeilinger, Melanie N. 2011. *Real-time Model Predictive Control*. Ph.d. thesis, ETH Zurich. URL http://e-collection.library.ethz.ch/eserv/eth:3071/eth-3071-02.pdf.

Zeilinger, Melanie Nicole, Colin Neil Jones, and Manfred Morari. 2011. "Real-Time Suboptimal Model Predictive Control Using a Combination of Explicit MPC and Online Optimization." *IEEE Transactions on Automatic Control* 56 (7):1524–1534. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5701768http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5701768.

Zhang, Xiaojing, Anil Aswani, Patrick Bouffard, and Claire Tomlin. 2013. "Experiments Using Sparse Interior Point Solver for Learning-Based MPC with Linear Models." In *submitted to European Control Conference*.

List of Symbols

3DMAG	three-dimensional magnetometer
CFTOC	constrained finite-time optimal control
EKF	extended Kalman filter
HL	high-level (processor on AscTec AutoPilot board)
ISS	input-to-state stability
LBMPC	learning-based model predictive control
LL	low-level (processor on AscTec AutoPilot board)
LP	linear program
LTI	linear, time-invariant
mp-QP	multi-parametric quadratic program
MPC	model predictive control
MPT	Multi-Parametric Toolbox
PD	Proportional-Derivative
PD IIPM	primal-dual infeasible start interior point method
PID	proportional-integral-derivative
PWA	piecewise-affine
PWA	piecewise-affine
QCQP	quadratically-constrained quadratic program
QP	quadratic program
SDK	software development kit
SISO	single-input single-output