NOTES ON MIXED-INITIATIVE CONTROL

An optimization-based approach

Written By

Bárbara Barros Carlos

DIAG Robotics Lab
Sapienza Università di Roma

July, 2020

Chapter 1

Adaptive Zone NMPC with Trajectory-Dependent Penalties for Mixed-Initiative Control

1.1 Main Idea

I try to explain in simple words without formulas the main idea of the framework. the nomenclature is completely tentative.

1.1.1 Problem setting

Robot

The robot is represented by a certain dynamical system and a set of 'dynamical constraints' on the system variables.

Working conditions

Certain given conditions are given and represent the minimum requirements that the robot has to obey to operate 'healthy', let us call them the 'working conditions'. These can contain both safety rules (e.g., maximum kinetic energy, minimum distances from obstacles etc.) and/or task related rules (e.g., accuracy of end-effector positioning, orientation, etc.)

Fully autonomous controller and cost

An automatic receding horizon controller is given that is able to ensure that the robot satisfies both dynamical constraints and the working conditions. We call this the 'fully-autonomous controller' and the corresponding cost function is called the 'fully-autonomous cost'.

Human inputs

The human is given the possibility to specify desired values for some of the system variables, either directly for some of the variable or in an implicit way (specifying the desired value of a function of the variables), these are called 'human inputs'.

Objective

Let the robot obey as much as possible to the human inputs while maintaining the working conditions and the dynamical constraints.

1.1.2 Our proposed solution

The objective is twofold, from one side we want the human input to be obeyed as much as possible and from the other side we want to keep valid the working conditions and dynamical constraints all the time. We know that the latter objective would be fulfilled if we let the robot be always controlled by the fully-autonomous controller. However this alone would not ensure the first objective.

Predicted healthiness index

At each time instant we simulate what would be the system behavior starting from the current measured state under the effect of the fully autonomous controller within the controller time horizon. From the simulated behavior we compute how far the robot is from violating the working conditions within the horizon. A certain continuous function between 0 and 1, named the 'predicted healthiness index', is then computed in a way that its value is 0 if the violation is too close and 1 if it is far enough. The idea is that the closer the index to 0 the more troublesome will be to keep the system within the working condition in the next horizon, even if full control is given to the fully autonomous controller (and human inputs are completely disregarded that moment on). Viceversa, the closer the index to 1 the easier the job of the fully autonomous controller will be in case it takes full control of the system.

Actual robot controller

We let the robot be controlled by a receding horizon controller which is a copy of the full autonomous controller a part from a major difference, its cost function, named the 'actual robot cost', is a convex combination between the fully autonomous cost and the 'human input cost'. The latter is a cost term which penalizes the divergence from the human inputs. The convex combination is driven by the Predicted healthiness index, when the index tends to 0 the actual robot cost tends to the fully-autonomous cost, when the index tends to 1 the actual robot cost tends to the human input cost.

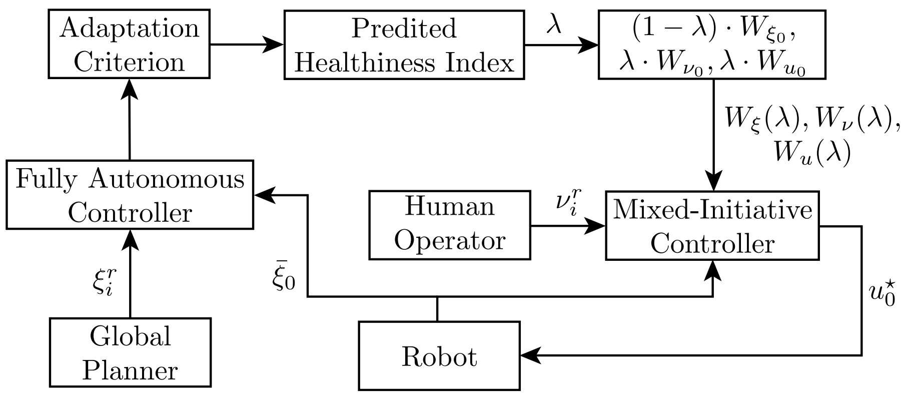
1.2 Introduction

This chapter presents a novel zone AF: is this 'zone' a term used in the literature or is a novel term? nonlinear model predictive control (NMPC) for mixed-inititiative control. A zone NMPC relies on a dynamic cost function that updates its state and control penalty parameters in real-time according to predicted trajectories. The criteria for adapting the parameters are based on zone-excursion functions that determine how the control authority between the human and the robot must be mixed at each sampling instant.

The proposed mixed-initiative control algorithm is evaluated considering a semi-autonomous task, in which a quadrotor must be steered by a human operator through a pre-established navigation plan. Three different levels of piloting skill in humans are regarded, ranging from inexperienced to highly experienced. The results indicate that the approach operates efficiently and safely, even in scenarios of weak piloting skills, reducing the risks of crashes.

1.3 Problem Formulation

AF: one thing is the problem and a different one is the solution, this section seems to talk about the solution, not the problem. A section about the problem should be added,


Adaptation Criterion
Predited Healthiness Index
λ
$(1-\lambda)\cdot W_{\xi_0}$, $\lambda \cdot W_{\nu_0}, \lambda \cdot W_{u_0}$
$W_\xi(\lambda), W_\nu(\lambda), W_u(\lambda)$
Fully Autonomous Controller
Human Operator
ν_i^r
Mixed-Initiative Controller
ξ_i^r
$\bar{\xi}_0$
$u_0^\star$
Global Planner
Robot

AF: a block diagram must be added and referred to.

The base controller of this work is a zone NMPC with trajectory-dependent penalty parameters. The zone NMPC model takes into account the robot dynamics plus part of the robot dynamics that the human is able to control, e.g. attitude and height AF: to be checked). The controller is driven by the predicted trajectories from another NMPC (namely, low-level NMPC), which considers just the dynamics of the robot. At each time instant, two constrained optimization problems are solved numerically, yielding two different feedback control policies.

In this section, we present the respective prediction models for each NMPC, as well as the nonlinear programs (NLP) arising in each NMPC formulation.

1.3.1 Prediction Models

Let $\{\mathcal{B}\}$ be the body-fixed frame, located at the center of mass (CoM) of a quadrotor, aligned with the North-West-Up (NWU) inertial frame $\{\mathcal{I}\}$. In order to implement the low-level NMPC, we consider a nonlinear dynamic model of the form

$$\dot{\xi} = f_1(\xi, u).$$

We then define the state vector, considering only the dynamics of the robot:

$$\xi := (p, \gamma, v_b, \omega, \Omega)^T \in \mathbb{R}^{16},$$

where $p := (x, y, z)$ is the position vector in $\{\mathcal{I}\}$, $\gamma := (\phi, \theta, \psi)$ are the Euler angles for orientation, $v_b := (v_x, v_y, v_z)$ is the linear velocities vector in $\{\mathcal{B}\}$, $\omega := (\omega_x, \omega_y, \omega_z)$ is the vector of angular rates, and finally $\Omega := (\Omega_1, \Omega_2, \Omega_3, \Omega_4)$ is the vector containing the rotation speed of the propellers. Hence, we regard the following quadrotor model:

$$
\begin{aligned}
\dot{p} &= R v_b \\
\dot{\gamma} &= T\omega \\
\dot{v}_b &= \frac{1}{m} F - R^T G - \omega \times v_b \\
\dot{\omega} &= J^{-1}(M - \omega \times J\omega) \\
\dot{\Omega} &= \frac{\tau}{J_m},
\end{aligned}
$$

AF: what about drag of the rotor in the motor equation? where the mass of the quadrotor is $m \in \mathbb{R}^+$, and $G := (0, 0, mg)^T$ with g being the gravitational acceleration. The robot inertia matrix with respect to its CoM and expressed in $\{\mathcal{B}\}$ is given by $J := \mathrm{diag}(J_{xx}, J_{yy}, J_{zz}) \in \mathbb{R}^{3\times 3}$. The rotation matrix from $\{\mathcal{B}\}$ to $\{\mathcal{I}\}$ is expressed as $R \in SO(3)$. The matrix $T : \mathbb{R}^3 \to \mathbb{R}^{3\times 3}$ represents the relation between the instantaneous rates of change of the Euler angles and the instantaneous components of ω. AF: define ALL the symbols introduced, here and after The total external forces and moments applied to the CoM of quadrotor in $\{\mathcal{B}\}$ are defined as

$$
F := \sum_{i=1}^{4} C_T \Omega_i^2 \mathbf{1}_z, \quad M := (M_x, M_y, M_z)^T \tag{1.1}
$$

with

$$
\begin{aligned}
M_x &= C_T \cdot l(-\Omega_1^2 - \Omega_2^2 + \Omega_3^2 + \Omega_4^2), \\
M_y &= C_T \cdot l(-\Omega_1^2 + \Omega_2^2 + \Omega_3^2 - \Omega_4^2), \\
M_z &= C_D(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2),
\end{aligned}
$$

where C_T is the thrust coefficient, C_D is the drag coefficient, and l is half of the distance between motors. When considering a medium-size quadrotor, the rotor inertia J_m is not negligible as the radius of the rotor's axle is not small. Therefore, we consider the rotor torques as the control inputs of this system

$$
u := (\tau_1, \tau_2, \tau_3, \tau_4)^T \in \mathbb{R}^4. \tag{1.2}
$$

Moreover, for the zone NMPC let us consider the following nonlinear dynamic model:

$$
\dot{\sigma} = f_2(\sigma, u).
$$

This model is composed of the robot dynamics plus part of the robot dynamics that the human is able to control. For a quadrotor, the latter is essentially composed of the rotational dynamics and the translational dynamics in z (which is directly linked to the collective speed of the propellers). Thus, we can formally define the part of the dynamics controlled by the human as:

$$
\nu := (\gamma, \omega, \Omega)^T \in \mathbb{R}^{10}.
$$

Then, the state vector of the zone NMPC can be characterized as follows:

$$
\sigma := (p, \gamma, v_b, \omega, \Omega, \nu)^T \in \mathbb{R}^{26},
$$

while the control inputs are the rotor torques, likewise regarded in (1.2).

AF: this is confusing. Some of the states seem to appear twice. I would rather say that in this controller there are some additional exogenous inputs provided by the human which are the desired orientation and desired total thrust and give to these quantities a different name. In any case these quantities should not appear in the state of the robot, in the sense that they do not affect the robot dynamics directly, but rather in the objective function, i.e., in the controlled (closed loop) dynamics, in the same way the desired trajectory appear.

1.3.2 Nonlinear Programs

We regard the nonlinear program ($\mathbf{NLP}_1$) corresponding to the constrained optimization problem of the low-level NMPC as follows:

$$\begin{aligned}
\min_{\substack{\xi_0,\ldots,\xi_N,\\ u_0,\ldots,u_{N-1}}} \quad & \frac{1}{2}\sum_{i=0}^{N-1} \|\eta(\xi_i,u_i)-\bar{\eta}\|_W^2 + \frac{1}{2}\|\eta_N(\xi_N)-\bar{\eta}_N\|_{W_N}^2 \\
\text{s.t.} \quad & \xi_0 = \bar{\xi}_0, \\
& \xi_{i+1} - F_1(\xi_i,u_i) = 0, \qquad i = 0,\ldots,N-1, \\
& \Omega_i \in \mathbb{E}, \qquad i = 0,\ldots,N-1, \\
& u_i \in \mathbb{U}, \qquad i = 0,\ldots,N-1,
\end{aligned}$$

AF: shouldn't the desired state $\bar{\eta}$ depend on i since it is time varying? Also please make it clear from the beginning, in the problem setting, that this is a trajectory that is passed to the system by an external planner in order to execute the task. where $\xi \in \mathbb{R}^{n_\xi}$ and $u \in \mathbb{R}^{n_u}$ denote the state and input trajectories of the discrete-time robot system whose dynamics are described by $F_1 : \mathbb{R}^{n_\xi} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_\xi}$. The residuals in the stage and terminal least-squares terms are denoted by $\eta : \mathbb{R}^{n_\xi} \times \mathbb{R}^{n_u} \to \mathbb{R}$ and $\eta_N : \mathbb{R}^{n_\xi} \to \mathbb{R}$, and will be penalized by the symmetric positive-definite matrices W and W_N, respectively. The output variables $\bar{\eta} : \mathbb{R}^{n_\xi} \times \mathbb{R}^{n_u} \to \mathbb{R}$ and $\bar{\eta}_N : \mathbb{R}^{n_\xi} \to \mathbb{R}$ represent the time-varying references passed to the automatic controller. We denote the admissible sets of u and Ω as $\mathbb{U} := [\underline{u}, \bar{u}]$ and $\mathbb{E} := [\underline{\Omega}, \bar{\Omega}]$, where the bounds $\underline{u}$, $\bar{u}$, $\underline{\Omega}$, and $\bar{\Omega}$ are given. Finally, N and $\bar{\xi}_0$ denote the horizon length and the current state of the system, respectively.

AF: Another general comment I have is that I am not sure that making the difference between Euler angles is mathematically correct, anyway let's go back to this later.

Similarly, the nonlinear program ($\mathbf{NLP}_2$) that corresponds to the constrained optimization problem of the zone NMPC is:

$$
\begin{aligned}
\min_{\substack{\sigma_0,\ldots,\sigma_N,\\ u_0,\ldots,u_{N-1}}} \quad & \frac{1}{2}\sum_{i=0}^{N-1} \|\eta(\sigma_i,u_i)-\tilde{\eta}\|^2_{W(\epsilon_i^\star)} + \frac{1}{2}\|\eta_N(\sigma_N)-\tilde{\eta}_N\|^2_{W_N(\epsilon_i^\star)} \\
\text{s.t.} \quad & \sigma_0 = \bar{\sigma}_0, \\
& \sigma_{i+1} - F_2(\sigma_i,u_i) = 0, \quad && i = 0,\ldots,N-1, \\
& \Omega_i \in \mathbb{E}, && i = 0,\ldots,N-1, \\
& u_i \in \mathbb{U}, && i = 0,\ldots,N-1,
\end{aligned}
$$

AF: OK, now I think understand better why you are elongating the state vector ξ in η repeating some its entries again, I guess this is in order to make the difference in the cost function and then the weighting matrix will take care of the mixing, giving more importance to ξ part of η or to the repeated entries. I think this should be better explained. I think that it is probably better to separate the two parts of the norms, the trajectory following and the human-following one, otherwise it remains hidden in the formulation for the reader and it could be misunderstood. I think it is better to not introduce η as an extended/repeated ξ but to actually reuse ξ in this second optimization problem together with a portion of ξ which you can call ξ_h and is the human 'output' to be regulated in order to match the human 'commands' and to show them separately in this optimization. In this way everything becomes more clear in my view. Another way to do so is to immediately define block-diag structure of the weighting matrix and to define the vector η as the stack of ξ and ξ_h so that is more clear.

AF: another thing that is confusing at this point is how come the human can give desired propeller rates for each propeller at the same time, together with the desired attitude? This is too complex for a human. What the human specifies is actually the total thrust, which is a function of the propeller rates but it does not specify them completely, there are several DoF left. We have to immediately say how we go from the desired thrust to all the propeller rates. I guess this partially explained in the appendix, but on the one side the explanation given in appendix is not completely clear to me and on the other side I think it should be immediately explained, not delayed to the appendix.

where $\sigma \in \mathbb{R}^{n_\sigma}$ and $u \in \mathbb{R}^{n_u}$ denote the state and input trajectories of the discrete-time robot-human system whose dynamics are described by $F_2 : \mathbb{R}^{n_\sigma} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_\sigma}$. The residuals in the stage and terminal least-squares terms are denoted by $\eta : \mathbb{R}^{n_\sigma} \times \mathbb{R}^{n_u} \to \mathbb{R}$ and $\eta_N : \mathbb{R}^{n_\sigma} \to \mathbb{R}$, and will be penalized by the time-varying matrices $W(\epsilon_i^\star), W_N(\epsilon_i^\star) \succ 0$, respectively. The output variables $\tilde{\eta} : \mathbb{R}^{n_\sigma} \times \mathbb{R}^{n_u} \to \mathbb{R}$ and $\tilde{\eta}_N : \mathbb{R}^{n_\sigma} \to \mathbb{R}$ represent the time-varying references coming from both global planner (offline) and human inputs (online). Notice that $W(\epsilon_i^\star)$ and $W_N(\epsilon_i^\star)$ depend on the predicted Euclidean distance ($\epsilon_i^\star$), which we will introduce in the next section, using the prediction model in the low-level NMPC. Accounting for reciprocal feasibility between controllers, the remaining constraints in ($\mathbf{NLP}_2$) are kept the same as in ($\mathbf{NLP}_1$).

1.4 Parameter Adaptation of Trajectory-Dependent NMPC

In this section, the proposed adaptation criteria are presented.

1.4.1 Adaptation Criteria

A natural question to ask is whether improved shared control performance can be achieved by further adapting parameters $W(\epsilon_i^\star)$ and $W_N(\epsilon_i^\star)$, which have a clear and direct relationship with the zone NMPC behavior – assigning smaller values to the part corresponding to the automatic controller in $W(\epsilon_i^\star)$ and $W_N(\epsilon_i^\star)$, and bigger values to the part corresponding to the human inputs, will prioritize the human actions over the automatic controller, and *vice versa*. We implement this idea by designing two adaptation criteria that are capable of achieving an enhanced performance in terms of sharing the control between the human operator and the automatic controller, reducing the risk of crashes. To that end, we first introduce some necessary definitions for both adaptation criteria.

AF: here, or actually before, in the problem setting, we need to make it clear that the main tasks consists in following a position trajectory. Also we need to make clear that going to far from the planned position is dangerous because the robot could collide with the obstacles. This is also why we focus on the position error in order to decide the mixing.

Definition 1.1 *(Predicted Euclidean Distance). Let us define the distance function $\epsilon : \mathbb{R}^3 \to \mathbb{R}$ for an arbitrary point $p \in \mathbb{R}^3$ with respect to a reference point $p^r \in \mathbb{R}^3$ as*

$$\epsilon = \|p^r - p\|_2.$$

Let us then consider $p_i^\star := \{p_0^\star, \dots, p_N^\star\}$ to be the optimal, predicted positions of the robot from the solution of ($\mathbf{NLP}_1$)*. Suppose $\epsilon_i^\star(p) = \epsilon_i^\star$. Thus, the predicted Euclidean distance (PED) is*

$$\epsilon_i^\star = \|p_i^r - p_i^\star\|_2, \qquad i = 0, \dots, N, \tag{1.3}$$

where p_i^r is the reference trajectory of the robot.

AF: instead of just 'predicted' I would call this the 'human-off' distance, or something similar. Basically this is the distance that one would have if the human is taken completely out of the loop.

Definition 1.2 *(Admissible Euclidean Distance). Consider $a \in \mathbb{R}^3$ as the maximum admissible error, in meters, with respect to the instantaneous position of the robot. We can define the admissible Euclidean distance (AED) such as:*

$$\epsilon_a = \|(p_0^r + a) - p_0^r\|_2. \tag{1.4}$$

AF: isn't this just $\|a\|_2$?

The AED is a fixed offset along the entire trajectory, thereby one can choose any point in the reference to calculate its value, e.g. the point at stage $N = 0$ was chosen AF: isn't N the number of prediction steps?.

The zone NMPC penalizes PED excursions from a carefully designed target zone with lower and upper bounds $\check{\epsilon}$ and $\hat{\epsilon}$, respectively. AF: why to put a

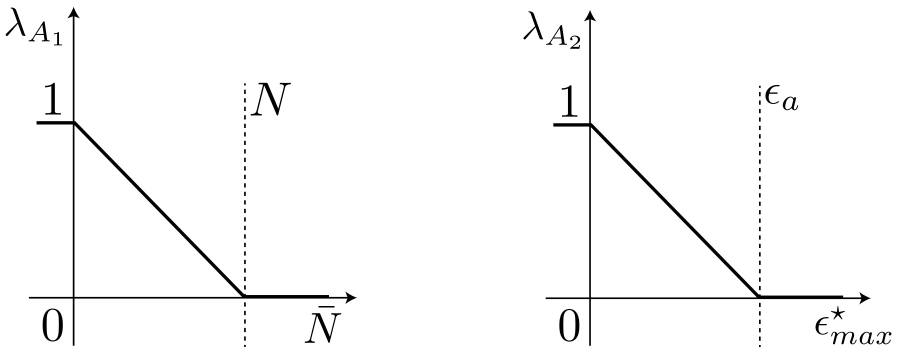


FIGURE 1.1. An illustration of the zone-excursion functions defined in (1.5) and (1.6) for adaptation criteria (A_1) and (A_2), respectively.

lower bound? isn't an error equal to 0 the perfect case that we aim to?At each time step, a zone-excursion function assigns the values of $W(\epsilon^\star_i)$ and $W_N(\epsilon^\star_i)$ based on an adaptation criterion that determines who, between the human and the robot, must have control authority at the moment. AF: rather: how the control authority should be distributed among the human and the fully automatic controller. To implement this principle, we regard two adaptation criteria and their respective zone-excursion functions:

(A_1) *settling stage*: given a horizon with N stages, , we select the last stage for which $\epsilon^\star_i \in [\check{\epsilon}, \hat{\epsilon}]$. AF: define this number more formally, assuming that in the last stage the position is within the bounds then it is the minimum i such that the position is in the bounds for i all all the subsequent stages. Otherwise it is just N. AF: say that this number represents the 'time' that it would take for the fully automatic controller to bring back the position error within the bound. This stage is henceforth termed "active shooting node", $\bar{N}$ AF I would rather call it the 'recovery time' or 'recovery stage'. The zone-excursion function $\lambda_{A_1} : \mathbb{W} \to \mathbb{R}$ is defined as:

$$\lambda_{A_1} := \{\bar{N} : (N - \bar{N})/N \in [0,1]\}. \tag{1.5}$$

(A_2) *maximum PED in the horizon*: given a horizon with N stages, we select the maximum PED within the horizon, $\epsilon^\star_{max}$, and apply the zone-excursion function $\lambda_{A_2} : \mathbb{R} \to \mathbb{R}$ characterized as:

$$\lambda_{A_2} := \{\epsilon^\star_{max} : (\epsilon_a - \epsilon^\star_{max})/\epsilon_a \in [0,1]\}. \tag{1.6}$$

Moreover, assume that $W_R \in \mathbb{R}^{n_\xi \times n_\xi}, W_H \in \mathbb{R}^{n_\nu \times n_\nu}$ and $W_C \in \mathbb{R}^{n_u \times n_u}$ are the weight matrices analogous, respectively, to the contribution of the robot, human, and control input in the cost function of (**NLP**$_2$). AF: the control inputs seem not appear in the cost function. These matrices contain the initial setting over which the zone-excursion functions modulate the control authority. Thus, we denote the structure of $W(\epsilon^\star_i)$ and $W_N(\epsilon^\star_i)$ for both adaptation criteria[1] as being:

[1]where the underscript $\bullet$ is a wildcard that specifies the zone-excursion function used for building $W(\epsilon^\star_i)$ and $W_N(\epsilon^\star_i)$.

$$
\begin{aligned}
W(\epsilon_i^\star) &= \text{blkdiag}((1-\lambda_\bullet)\cdot W_R, \lambda_\bullet \cdot W_H, \lambda_\bullet \cdot W_C) \\
W_N(\epsilon_i^\star) &= \text{blkdiag}((1-\lambda_\bullet)\cdot W_R, \lambda_\bullet \cdot W_H).
\end{aligned}
\tag{1.7}
$$

An illustration of each zone-excursion function is provided in Fig. 1.1.

1.5 Numerical Simulations

In this section, we first provide a detailed description of the navigation task and the implementation. Further we present the simulation results and the computational burden associated with the proposed algorithm.

1.5.1 Semi-Autonomous Navigation Task

The scenario under consideration is inspired by a persistent navigation task in which a person modifies the reference that is autonomously tracked by a robot, rather than piloting the robot itself. The human is assisted by our mixed-initiative control algorithm which allows deformation in the trajectory followed by the robot.

In practice, we would like to use the path calculated by a path planner as reference to ($\mathbf{NLP}_1$) and the part corresponding to the automatic controller in ($\mathbf{NLP}_2$). A path is only a sequence of points on the map, typically without heading (orientation of the robot), velocity information, and time indexing. These need to be added before we can use the provided path as reference trajectory in the aforementioned NLPs, which can still be kinodynamically infeasible. When a high-quality reference trajectory (e.g. dynamically feasible, close to a global optimal, etc.) is readily available, it should be used. This can be employed to initialize and thereby drastically speed-up subsequent optimization solutions, and improve the convergence behavior of the algorithm. Nevertheless, a path planner (even a coarse one) is often not available and may be expensive to design or time-consuming to run. Thus, our approach includes a very simple initialization scheme.

Global Reference: We assume that the time-varying reference in position p^d to be tracked is composed of a simple cartesian helical trajectory, generated using MATLAB. A radius of $r = 0.3$ m is imposed, initial height of $h_0 = 0.35$ m, height increments of $\Delta h = 0.005$ m at each time step, total duration of $t_f = 6$ s, and $N = 400$ time steps. Moreover, we regulate for $\gamma^d := (0,0,0)^T$ and $\Omega^d := 0.06546 \cdot \mathbf{1_4}$, while $v_b^d := (0,0,0)^T$ and $\omega^d := (0,0,0)^T$.

Human Guidance: We simulate human actions by means of a reactive controller. Different levels of piloting skills are achieved through three different sets of gain matrices, which significantly improve the tracking performance of the controller. We regard the following control law:

$$
\begin{aligned}
\mathcal{U}_1 &= \mathbf{f}_d^T \mathbf{R}_d \mathbf{1_z} \\
\mathcal{U}_2 &= -\mathbf{K}_R \mathbf{e}_R - \mathbf{K}_\omega \mathbf{e}_\omega
\end{aligned}
\tag{1.8}
$$

with the respective position, velocity, rotation and angular velocity errors

$$
\mathbf{e}_p = p - p^d, \quad \mathbf{e}_v = \dot{p} - \dot{p}^d, \quad \mathbf{e}_R = \frac{1}{2}(\mathbf{R}_d^T\mathbf{R} - \mathbf{R}\mathbf{R}_d)^\vee, \quad \mathbf{e}_\omega = \omega - \mathbf{R}\mathbf{R}_d^T\omega^d,
$$

and the following vector

$$\mathbf{f}_d = mg\mathbf{1_z} - \mathbf{K}_p\mathbf{e}_p - \mathbf{K}_v\mathbf{e}_v - \mathbf{K}_I\int_{t_0}^{t_s}\mathbf{e}_p d\tau,$$

where $\bullet^\vee : SO(3) \to \mathbb{R}^3$ is the the *vee* operator, and $\mathbf{R} \in SO(3)$ represents the rotation matrix from $\{\mathcal{B}\}$ to $\{\mathcal{I}\}$. The reference attitude trajectory is denoted by $\mathbf{R}_d \in SO(3)$, while $\mathbf{K}_p, \mathbf{K}_v, \mathbf{K}_I, \mathbf{K}_R$ and $\mathbf{K}_\omega$ are positive definite gain matrices.

The control law in (1.8) provides the nominal input required to track the full-pose trajectory $q = (p^d, R_d) : [t_0, t_f] \to SE(3)$, retrieved from the global reference. The matrix $\mathbf{R}_d$ and the nominal input can be transformed into desired attitude and speed of the propellers (see Appendix B.1), which are passed to ($\mathbf{NLP}_2$) as a reference trajectory. In this way, as long as the adaptive criterion is met, the human inputs will retain the control authority and modify the reference tracked by the robot.

The levels of piloting skills considered are inexperienced (**L0**), intermediate (**L1**) and experienced (**L2**).

1.5.2 Implementation Details

Nonlinear Optimization via Real-Time Iteration Scheme: In real-time NMPC applications, ($\mathbf{NLP}_1$) and ($\mathbf{NLP}_2$) need to be solved at each sampling instant under the available computation time. To that end, we use a numerical strategy based on *sequential quadratic programming* (SQP) that relies on the solution of a limited number of quadratic program (QP) subproblems, the so-called real-time iteration (RTI) scheme. More precisely, in this strategy only a single linearization and QP solve are carried out per sampling instant, leading to an approximate feedback control policy. An important ingredient in the RTI scheme is to keep the initial state as a constrained decision variable, which is often referred to as *initial value embedding.* This allows one to divide computations into a preparation and a feedback phase, where the former is typically more expensive. In this work, we will be using the implementation of the RTI method by means of the high-performance software package `acados`.

The *direct multiple shooting* approach is used in order to obtain ($\mathbf{NLP}_1$) and ($\mathbf{NLP}_2$). The state trajectories from stage 0 to N are approximated using a numerical integration scheme, in this case an explicit Runge Kutta 4th order (ERK4), while the input trajectories are parametrized as piecewise constant. We make use of `acados` Python template-based interface to generate the libraries that implement ($\mathbf{NLP}_1$) and ($\mathbf{NLP}_2$), which are then wrapped by our mixed-initiative control algorithm, written in Python, which executes Algorithm 1.

Structure-exploiting QP Solver and Condensing Approach: The QPs arising in both NMPC formulations are addressed using the high-performance interior-point method (`HPIPM`) solver that is built on top of the linear algebra package `BLASFEO`. This Riccati-based solver implements an efficient method for the solution of linear-quadratic (LQ) control problem, that is a special instance of equality constrained quadratic program. The corresponding Karush-Kuhn-Tucker (KKT) system is sparse and structured, and this structure is exploited

Algorithm 1: zone NMPC for mixed-initiative control

```
Input  : States $\bar{\xi}_{0(k)}, \bar{\sigma}_{0(k)}$ and references $\bar{\eta}_{(k)}, \tilde{\eta}_{(k)}$ at iteration $k$.
Output: Solution $(\sigma_i, u_i)_k$ for $(\mathbf{NLP}_2)_k$ at iteration $k$.
Data: maximum admissible error $a = (0.05, 0.05, 0.1)$ m, criterion,
      simulation time $t_f = 6$ s, sampling time $t_s = 0.015$ s.
begin
    $k \longleftarrow 0$;
    Calculate $\epsilon_a$ in (1.4);
    $\check{\epsilon} \longleftarrow 0$;
    $\hat{\epsilon} \longleftarrow \epsilon_a$;
    for $k \longleftarrow 0$ to $t_f/t_s$ do
        Update $\bar{\eta}_{(k)}$;
        Solve $(\mathbf{NLP}_1)_k$ for $\bar{\xi}_{0(k)}$;
        for $i \longleftarrow 0$ to $N$ do
            Calculate $\epsilon^{\star}_{i(k)}$ in (1.3);
        if criterion $= (A_1)$ then
            for $i \longleftarrow N$ to $0$ do
                if $\epsilon^{\star}_{i(k)} < \hat{\epsilon}$ then
                    $\bar{N}_{i(k)} \longleftarrow$ NaN;
                else
                    $\bar{N}_{i(k)} \longleftarrow i$;
            $\bar{N}_{(k)} \longleftarrow$ last non-NaN element of $\bar{N}_{i(k)}$;
            Evaluate zone-excursion function $\lambda_{A_1(k)}$ in (1.5);
        if criterion $= (A_2)$ then
            for $i \longleftarrow N$ to $0$ do
                if $\epsilon^{\star}_{i(k)} > \hat{\epsilon}$ then
                    $\epsilon^{\star}_{i(k)} \longleftarrow \hat{\epsilon}$;
                $\epsilon^{\star}_{i,max(k)} \longleftarrow \epsilon^{\star}_{i(k)}$;
            $\epsilon^{\star}_{max(k)} \longleftarrow \max(\epsilon^{\star}_{i,max(k)})$;
            Evaluate zone-excursion function $\lambda_{A_2(k)}$ in (1.6);
        Build $W(\epsilon^{\star}_{i(k)})$ and $W_N(\epsilon^{\star}_{i(k)})$ in (1.7);
        Calculate $\mathcal{U}_{1(k)}$ and $\mathcal{U}_{2(k)}$ in (1.8);
        Simulate human commands based on $\mathbf{R}_{d(k)}$, $\mathcal{U}_{1(k)}$ and $\mathcal{U}_{2(k)}$;
        Update $\tilde{\eta}_{(k)}$;
        Solve $(\mathbf{NLP}_2)_k$ for $\bar{\sigma}_{0(k)}$;
        return $(\sigma_i, u_i)_k$
```

by HPIPM which grants a reduction in the number of flops. As for the linear algebra library, the major difference between existing high-performance implementations of BLAS– and LAPACK–like routines for embedded applications and BLASFEO is that the last is optimized for small to medium scale matrices. The X64_INTEL_HASWELL implementation of BLASFEO package was used, which exploits a set of vectorized instructions for the target CPU.

Additionally, for the NMPC solutions, we used partial condensing `HPIPM`-based technique. This approach reformulates the large and sparse Hessians of ($\mathbf{NLP}_1$) and ($\mathbf{NLP}_2$) into small and dense ones, which have a more suitable form for the QP solver.

Parametrization of Mixed-Initiative Control Algorithm: To what concerns the parametrization of both NLPs, we assume a sampling time of $t_s = 15$ ms, and $N = 50$ shooting nodes. The values of the parameters appearing in the model are listed in Table 1.1. Constraints for this system include bounds on speed $\underline{\Omega} = 0$, $\bar{\Omega} = 0.09$ kHz, and control $\underline{u} = 0$, $\bar{u} = 0.1285$ kNm. The weight matrices of the low-level NMPC are

$$\begin{aligned} W &= \text{blkdiag}(250 \cdot \mathbf{I_2}, 300, 30 \cdot \mathbf{I_3}, 5 \cdot \mathbf{I_6}, 30 \cdot \mathbf{I_4}, 70 \cdot \mathbf{I_4}), \\ W_N &= \text{blkdiag}(250 \cdot \mathbf{I_2}, 300, 30 \cdot \mathbf{I_3}, 5 \cdot \mathbf{I_6}, 30 \cdot \mathbf{I_4}). \end{aligned}$$

Similarly, the weight matrices for the initial setting of the zone NMPC are

$$\begin{aligned} W_R &= \text{blkdiag}(250 \cdot \mathbf{I_2}, 300, 30 \cdot \mathbf{I_3}, 5 \cdot \mathbf{I_6}, 30 \cdot \mathbf{I_4}), \\ W_H &= \text{blkdiag}(30 \cdot \mathbf{I_3}, 5 \cdot \mathbf{I_3}, 30 \cdot \mathbf{I_4}), \\ W_C &= 70 \cdot \mathbf{I_4}. \end{aligned}$$

Notice that the weights assigned to the human contribution have the same values as their corresponding part in the robot contribution (i.e. Euler angles, angular rates and speed of the propellers). This is done in order to preserve equality of modulation-range.

Finally, the maximum deformation conceded to the human with respect to the reference of the robot is $a := (0.05, 0.05, 0.1)$ m, causing the bounds on the target zone to be $\check{\epsilon} = 0, \hat{\epsilon} = 0.122$ m.

Table 1.1 Quadrotor model - values of the parameter used for the simulation results in this section.

Parameter	**Value**	**Description**
g	9.8066 m/s^2	gravitational acceleration
m	1.04 kg	total mass
l	0.23 m	arm length
J_{xx}	0.010 kg $\cdot$ m^2	inertia moment around x
J_{yy}	0.010 kg $\cdot$ m^2	inertia moment around y
J_{zz}	0.070 kg $\cdot$ m^2	inertia moment around z
C_D	10.0 Nm/kHz	drag coefficient
C_T	595.0 N/kHz	thrust coefficient
J_m	0.8 kg $\cdot$ m^2	rotor inertia

1.5.3 Performance Analysis

In the simulations we assess the effectiveness of the mixed-initiative control algorithm in terms of whether or not human operators are afforded the freedom that is required to modify the reference tracked by the robot, regardless of the level of piloting skill. Additionally, we would like to measure the performance of the navigation task for the two adaptation criteria presented. Figure 1.2–1.7 show the results for six different simulations in which Algorithm 1 was used. Independently of the criterion used to compute $\lambda_\bullet$, the values $\lambda_\bullet = 0.0, \lambda_\bullet \in \mathbb{R} = (0.0, 1.0), \lambda_\bullet = 1.0$ indicate *fully automatic control*, *shared control* and *fully human control*, respectively.

The graphs depict the evolution of both zone-excursion functions $\lambda_\bullet$ for which the insets show the corresponding unfolding of either $\bar{N}$ or $\epsilon^\star_{max}$. It is also shown the predicted Euclidean distances $\epsilon^\star_i$ analogous to the time interval of the inset. The light blue and pink colors in the graphs for criterion (A_1) indicate the predicted Euclidian distances that are within and exceed the target zone, respectively, e.g. $\bar{N} = 0$ and $\bar{N} \in \mathbb{W} = [1, 50]$. In contrast, for criterion (A_2) pink corresponds to $\epsilon^\star_{max} \in \mathbb{R} = (0.0, 1.0)$.

When $\bar{N} \in \mathbb{W} = [1, 50]$, i.e. PED exceeds the target zone, then $0 < \lambda_{A_1} \leq 1$, resulting in a reduction in the human control authority while $\lim_{\bar{N} \to 50} \lambda_{A_1}(\bar{N})$, according to (1.7). In general, the insets of λ_{A_1} show that, when active, the zone NMPC starts producing a precipitous fall in the human control authority. However, due to the high level of "trust in the human" embedded in criterion (A_1), this off-peak is short-lived. Because of the settling stage-weighting, the active shooting node is gradually decreased, with respect to previous NMPC solutions, right after the main λ_{A_1} off-peak. This decrease corresponds to the transition from the shared control phase, in which $\epsilon^\star_i$ exceeds the target zone, to the phase fully controlled by the human, in which $\epsilon^\star_i$ undulates nonchalantly within $[\check{\epsilon}, \hat{\epsilon}]$, accruing no extra penalization, as $\bar{N}$ remains equal to zero.

On the other hand, when using criterion (A_2), the zone NMPC persistently opts for stark shared control throughout the task. The aim of zone-excursion function λ_{A_2} is to aggressively reduce human control authority only an increasing in the error margin arises, indicating that this is a criterion with a low level of "trust in the human". Maximum PED-weighting constantly attenuates human control authority compared with using the fixed nominal weights and, for this reason, is inherently safer as it is "active" at each prediction step.

The simulation results have shown that the proposed mixed-initiative control algorithm is able to provide to the human operator the freedom to modify the reference tracked by the robot, no matter the level of piloting skill. Proposed zone NMPC with criterion (A_1) yields an oscillatory, albeit slightly more sluggish, return to the target zone. Instead, proposed zone NMPC with criterion (A_2) does not exceed the target zone whatsoever. As maximum PED-weighting ends up being continuously "active", it is advantageous for more challenging situations, for instance, with an inexperienced pilot, capable of reducing the human workload and also providing higher performance benefits.

Inexperienced (**L0**)*:* Closed-loop results for mixed-initiative control algorithm.

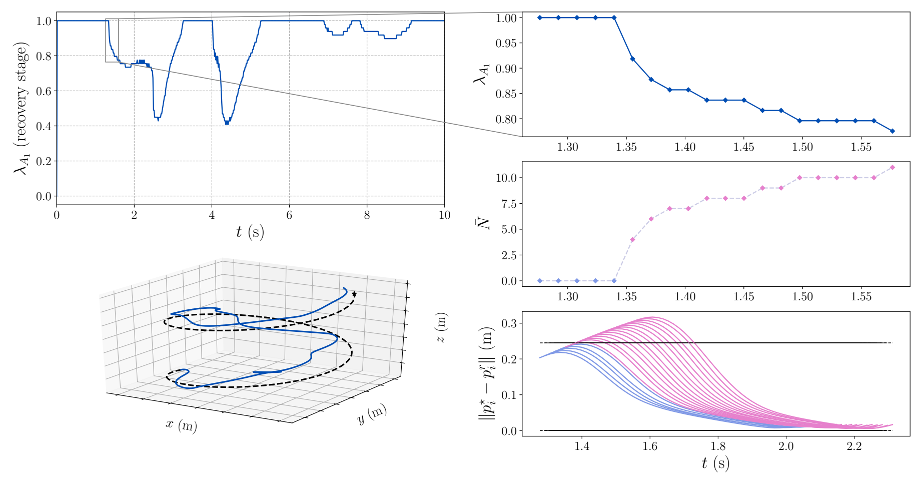


FIGURE 1.2. Using adaptation criterion (A_1).

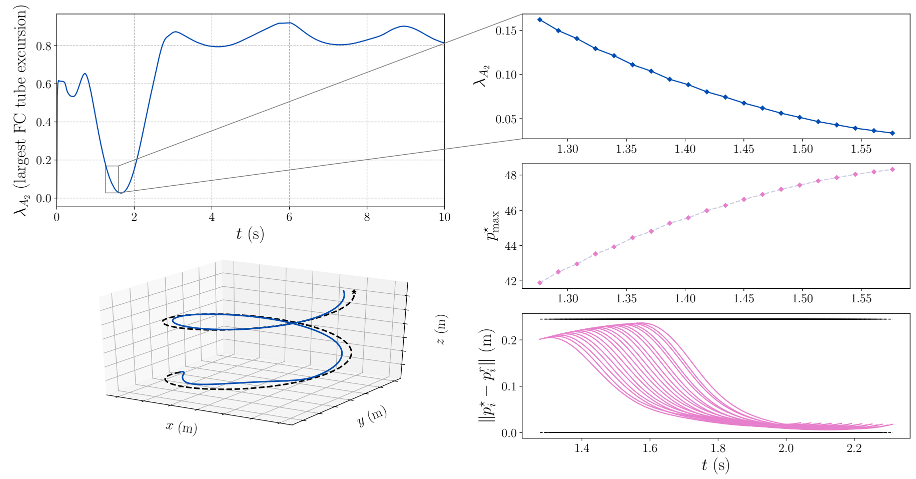


FIGURE 1.3. Using adaptation criterion (A_2).

Intermediate (**L1**)*:* Closed-loop results for mixed-initiative control algorithm.

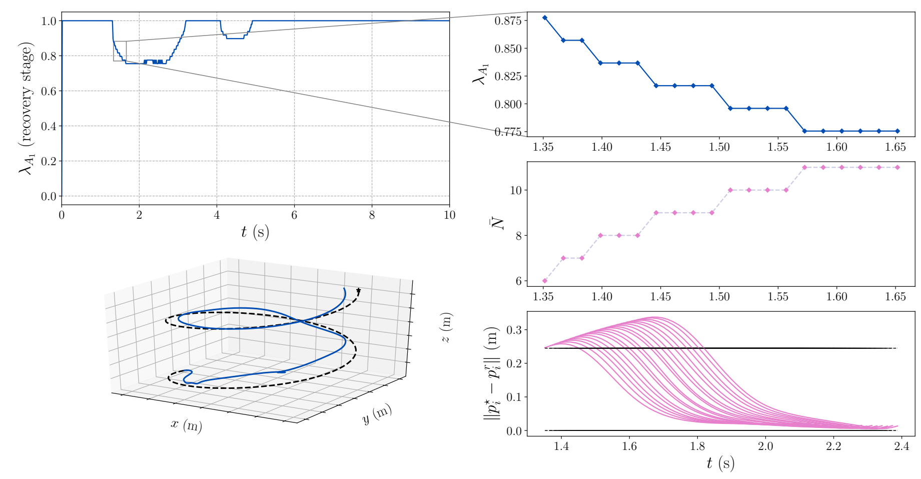


FIGURE 1.4. Using adaptation criterion (A_1).

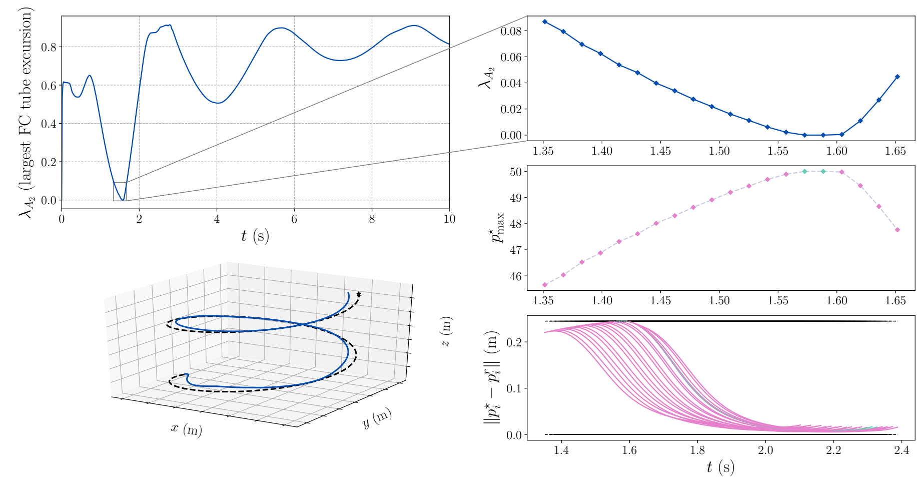


FIGURE 1.5. Using adaptation criterion (A_2).

Experienced (**L2**)*:* Closed-loop results for mixed-initiative control algorithm.

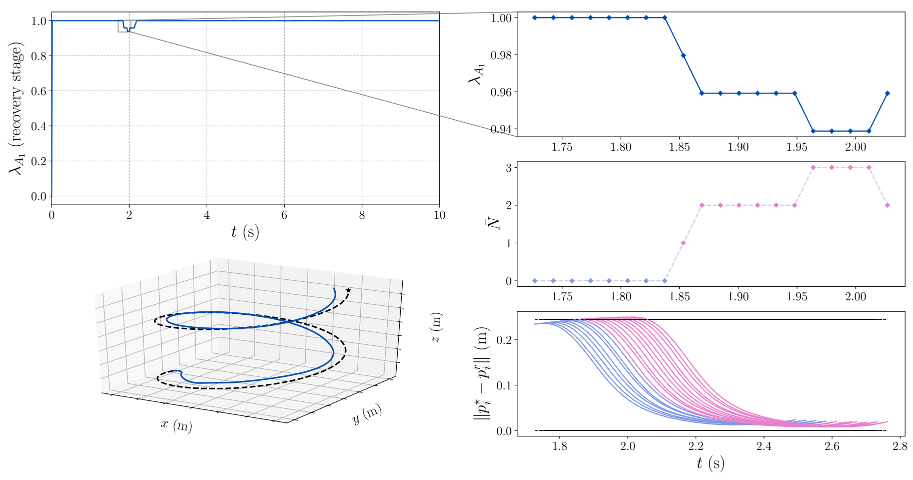


FIGURE 1.6. Using adaptation criterion (A_1).

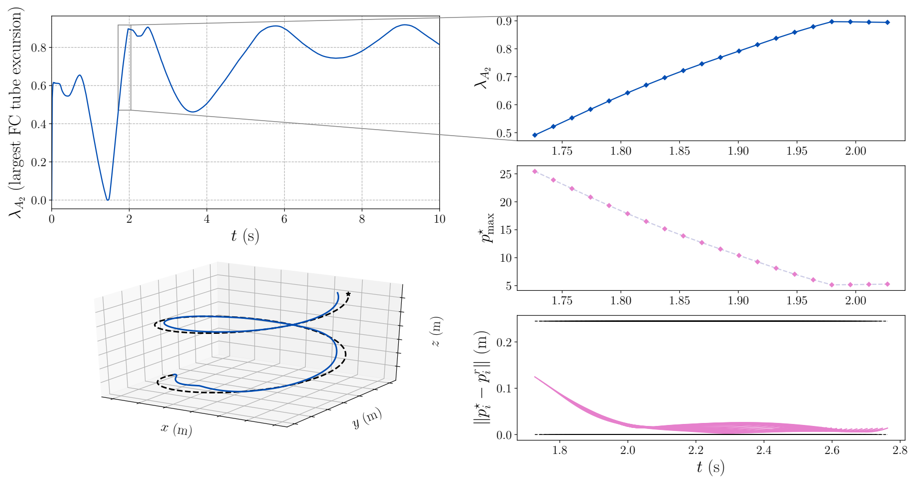


FIGURE 1.7. Using adaptation criterion (A_2).

1.5.4 Computational Burden

Computation times reported are from an Intel Core i5 @2.6 GHz running macOS Catalina. Whether the sparse, condensed or partially condensed formulation is more appropriate for a certain problem depends mainly on the horizon length and the ratio between number of states and controls. In this particular problem, we note that the partially condensed QP is more beneficial from a computational point of view. `HPIPM` is significantly faster in solving the partially condensed QP arising in each NMPC formulation compared to the corresponding dense QP (see Table 1.2 and 1.3). The dense solver `qpOASES` was also tested, but did not return a solution for any of the attempts.

In the condensed formulation, all state deviations are eliminated via the continuity constraints of ($\mathbf{NLP}_1$) and ($\mathbf{NLP}_2$) leading to a smaller but dense QP. Conversely, partial condensing is a condensing strategy that is in-between the sparse and the fully condensed formulations. In this strategy, a state component is retained as an optimization variable at each stage of the partially condensed QP. Given the dimensions of the optimal control problems (OCP) involved in our mixed-initiative control algorithm, partial condensing allows to better exploit hardware throughput.

Table 1.2 Average computation times in [ms] for partial condensing.

		`HPIPM`					
		L0		**L1**		**L2**	
	Condensing approach	(A_1)	(A_2)	(A_1)	(A_2)	(A_1)	(A_2)
LL NMPC[2]	*partial condensing*	4.92	4.51	4.53	4.04	4.05	3.88
zone NMPC	*partial condensing*	10.09	9.74	7.58	7.62	6.83	7.51

Table 1.3 Average computation times in [ms] for condensing.

		`HPIPM`					
		L0		**L1**		**L2**	
	Condensing approach	(A_1)	(A_2)	(A_1)	(A_2)	(A_1)	(A_2)
LL NMPC	*condensing*	9.67	8.43	8.53	7.30	7.78	7.15
zone NMPC	*condensing*	14.07	13.60	10.47	10.74	9.43	10.08

[2]shorthand for low-level NMPC.

Appendix B

APPENDIX

B.1 Generation of Human Commands

Given the reference attitude $\mathbf{R}_d$ and the nominal input $(\mathcal{U}_1, \mathcal{U}_2)$, it is possible to determine the desired attitude $(\phi_h^d, \theta_h^d, \psi_h^d)$ and speed of the propellers $(\Omega_{h,[1,4]}^d)$ that correspond to the human inputs. We will define the following parametrization of the rotation matrix:

$$\mathbf{R}_d = \begin{pmatrix} cos\theta cos\psi & -cos\phi sin\psi + sin\phi sin\theta cos\psi & sin\phi sin\psi + cos\phi cos\theta cos\psi \\ cos\theta sin\psi & cos\phi sin\psi + sin\phi sin\theta sin\psi & -sin\theta cos\psi + cos\phi sin\theta sin\psi \\ -sin\theta & sin\phi cos\theta & cos\phi cos\theta \end{pmatrix}.$$

Based on that, one can compute the desired attitude by doing:

$$\phi_h^d = \text{atan2}\left(\frac{\mathbf{R}_{d,32}}{\mathbf{R}_{d,33}}\right), \quad \theta_h^d = -\text{asin}\left(\mathbf{R}_{d,31}\right), \quad \psi_h^d = \text{atan2}\left(\frac{\mathbf{R}_{d,21}}{\mathbf{R}_{d,11}}\right).$$

Moreover, to compute the desired speed of the propellers we assume control decoupling. First, we linearize the forces and moments in (1.3.1) and rewrite them in the following matrix format:

$$\begin{pmatrix} F_z \\ M_x \\ M_y \\ M_z \end{pmatrix} = 2\Omega_e \underbrace{\begin{pmatrix} C_T & C_T & C_T & C_T \\ -C_T l & -C_T l & C_T l & C_T l \\ -C_T l & C_T l & C_T l & -C_T l \\ -C_D & C_D & -C_D & C_D \end{pmatrix}}_{\mathbf{\Gamma}} \begin{pmatrix} \Delta\Omega_1 \\ \Delta\Omega_2 \\ \Delta\Omega_3 \\ \Delta\Omega_4 \end{pmatrix}, \tag{B.9}$$

where Ω_e is the required speed of each rotor in order to maintain the hover position, and the prefix Δ indicates the result of a linearization process.

Second, if matrix $\mathbf{\Gamma}$ is invertible, then the lines are linearly independent, meaning that the forces and moments acting on the CoM of the quadrotor are independently from each other. The inverse relation of (B.9) is given by

$$\begin{pmatrix} \Delta\Omega_1 \\ \Delta\Omega_2 \\ \Delta\Omega_3 \\ \Delta\Omega_4 \end{pmatrix} = \frac{1}{2\Omega_e} \underbrace{\begin{pmatrix} 1/(4C_T) & -1/(4C_T l) & -1/(4C_T l) & -1/(4C_D) \\ 1/(4C_T) & -1/(4C_T l) & 1/(4C_T l) & 1/(4C_D) \\ 1/(4C_T) & 1/(4C_T l) & 1/(4C_T l) & -1/(4C_D) \\ 1/(4C_T) & 1/(4C_T l) & -1/(4C_T l) & 1/(4C_D) \end{pmatrix}}_{\mathbf{\Gamma}^{-1}} \begin{pmatrix} F_z \\ M_x \\ M_y \\ M_z \end{pmatrix}. \tag{B.10}$$

The inverse mapping (B.10) dictates how each of the forces acting on the CoM of the quadrotor contributes to the rotation speed of each propeller. We can use this fact to establish the relation between the nominal input of the reactive controller and the desired speed of the propellers, which are further passed on to $(\mathbf{NLP}_2)$. Consequently, we have that

$$\begin{pmatrix} \Omega^d_{h,1} \\ \Omega^d_{h,2} \\ \Omega^d_{h,3} \\ \Omega^d_{h,4} \end{pmatrix} = \frac{1}{2\Omega_e}\mathbf{\Gamma}^{-1}\begin{pmatrix} \mathcal{U}_{1,z} \\ \mathcal{U}_2 \end{pmatrix},$$

where $\mathcal{U}_{1,z}$ is the z component of vector $\mathcal{U}_1$.