Nonlinear Model Predictive Control of a Human-sized Quadrotor

Andrea Zanelli[1], Greg Horn[2], Gianluca Frison[1] and Moritz Diehl[1]

Abstract— **This paper discusses the design, implementation and deployment of an attitude controller for a quadrotor based on nonlinear model predictive control on a low-power embedded system equipped with a Cortex A9 CPU running at 800 MHz. Due to the limited computational power of the available hardware, a modified interior-point solver for the so-called partially tightened Real-Time Iteration is used. The algorithm splits the prediction horizon in two sections. A Riccati-like recursion is exploited that relies on a single linearization of the complementarity conditions per sampling-time for the terminal section. In this way, it is possible to achieve a speedup of a factor 3 with respect to a standard real-time iteration formulation for the application under consideration. Simulation results that show the improvement in performance obtained by using NMPC over standard control techniques are discussed and experimental results using the proposed implementation are presented.**

I. INTRODUCTION

Nonlinear model predictive control (NMPC) is an optimization-based control technique that allows one to directly address several sources of complexity in the process of designing a controller. In particular, it provides a framework to design a controller in the presence of nonlinear dynamics and constraints which can lead to significant performance improvements with respect to classical control solutions. However, these advantages come at the cost of a considerably increased computational effort needed to compute the control law: at every sampling time, a nonlinear nonconvex optimization problem has to be solved that can result in prohibitive computation times, especially on low-power embedded systems.

For this reason, NMPC has been historically used in applications with relatively slow dynamics, where enough time is available to compute the control inputs to be applied to the plant [16]. In recent years, due to the considerable progress made in algorithms and software implementations and thanks to the increasing available computation power on embedded platforms, applications with shorter sampling times could be realized. In [10] NMPC is used to control a diesel engine, while in [4] it is applied to a gasoline two-stage turbo charger. In both applications computation times in the order of tens of milliseconds could be met.

This research was supported by the EU via ERC-HIGHWIND (259 166), FP7-ITN-TEMPO (607 957) and H2020-ITN-AWESCO (642 682), by the Federal Ministry for Economic Affairs and Energy (BMWi) via eco4wind and DyConPV, and by DFG via Research Unit FOR 2401.

[1]Andrea Zanelli, Gianluca Frison and Moritz Diehl are with the University of Freiburg, Department of Microsystems Engineering (IMTEK), Georges-Koehler-Allee 102, 79110 Freiburg, Germany - andrea.zanelli, gianluca.frison, moritz.diehl @imtek.uni-freiburg.de

[2]Greg Horn is with Kitty Hawk Corporation, Mountain View, CA, USA - greg@kittyhawk.aero

A. Background and Contribution

Unmanned aerial vehicles (UAVs) are finding their way into several application fields such as inspection, surveillance and rescuing and are drawing considerable interest in the control engineering community. Furthermore, a few applications have emerged in which quadrotor- and multirotor-like systems are used as personal air vehicles [2], [3]. Due to the highly nonlinear dynamics exhibited by quadrotors, the performance of linearization-based controllers can be affected when the vehicles are operated far from the linearization point. Moreover, with classical control solutions, it is non-trivial to deal with constraints on states and inputs, when they are present. Although such approaches have been successfully applied to the problem of controlling the attitude of quadrotors (see [7], [18] to cite only few applications), NMPC could in principle provide a more direct approach in treating nonlinearity and constraints. In [13] an NMPC attitude controller for a multicopter that operates on the rotation group $\mathbf{SO}(3)$ using the Real-Time Iteration (RTI) scheme [8] is proposed.

In this paper, NMPC is used to design an attitude controller for a human-sized quadrotor equipped with a low-power embedded processor running at 800 MHz. The designed controller is required to be able to compute the control action within 10 ms, while sparing enough CPU time for the other routines running on the embedded platform to be performed (e.g telemetry, logging, low-level controller, sensing and estimation, fault detection, etc). In order to meet the required execution time a so-called partially tightened formulation [20] is used that allows one to speed up the computations by exploiting a modified Riccati-based interior-point solver. The optimal control formulation (OCP) utilizes dynamics in quaternion form and a nonlinear least-squares cost that enables direct tracking of references in the Euler angles space.

The main purpose of the paper is, rather than building on top of state-of-the-art control techniques for UAVs, to show that recent advances in algorithms and software implementations enable one to reduce the computational burden associated with optimization-based control techniques. In particular, applications can be tackled where short sampling times need to be met on resource constrained hardware. The paper is structured as follows: in Section II preliminary concepts on NMPC and the RTI scheme are described. Section III introduces the algorithm used in the controller. Sections IV and V present simulation and experimental results respectively and in Section VI conclusions and outlook are discussed.

II. Preliminaries

A. Nonlinear Model Predictive Control

In order to use NMPC to control a system, one has to be able to efficiently solve nonlinear nonconvex optimal control problems. In this paper, the following nonlinear least-squares formulation will be taken into account:

$$\begin{aligned} \min_{\substack{x_0,\cdots,x_N \\ u_0,\cdots,u_{N-1}}} \quad & \frac{1}{2}\sum_{i=0}^{N-1} \|\eta(x_i,u_i)\|^2_W + \frac{1}{2}\|\eta_N(x_N)\|^2_{W_N} \\ \text{s.t.} \quad & x_0 - \bar{x}_0 = 0 \\ & x_{i+1} = f(x_i,u_i), \quad i = 0,\cdots,N-1 \\ & g(x_i,u_i) \le 0, \qquad i = 0,\cdots,N-1 \\ & g_N(x_N) \le 0, \end{aligned} \tag{1}$$

where $x_i \in \mathbb{R}^{n_x}$ and $u_i \in \mathbb{R}^{n_u}$ are the states and inputs of the system respectively and f, g, g_N, η and η_N are twice continuously differentiable functions. The nonlinear residual functions η and η_N are weighted by W, $W_N \succ 0$ respectively and the initial state of the system is denoted by $\bar{x}_0$.

In order to efficiently solve (1), among other approaches [14], SQP-based methods can be used that rely on the solution of a series of quadratic programs (QPs) that locally approximate the original nonlinear program (NLP). Together with several other algorithmic ingredients and under mild assumptions [14], it can be shown that the generated iterates converge to a local minimum of the NLP.

In this paper a method based on SQP and a modified version of the so-called Real-Time Iteration (RTI), which will be described in the following section, will be used.

B. Real-Time Iteration

Since the computation times associated with the solution of an NLP can be rather high, the RTI scheme can be used which relies on a single SQP iteration per sampling time. In this way, the NLP needs to be linearized only once and a single QP needs to be solved, leading to considerably reduced computation times. After linearization, a QP of the following form is obtained:

$$\begin{aligned} \min_{\substack{x_0,\cdots,x_N \\ u_0,\cdots,u_{N-1}}} \quad & \sum_{i=0}^{N-1} l(x_i,u_i) + l_N(x_N) \\ \text{s.t.} \quad & x_0 - \bar{x}_0 = 0 \\ & x_{i+1} - A_i x_i - B_i u_i - c_i = 0, \; i = 0,\cdots,N-1 \\ & d_i + G_i^x x_i + G_i^u u_i \le 0, \qquad i = 0,\cdots,N-1 \\ & d_i + G_N^x x_N \le 0, \end{aligned} \tag{2}$$

where

$$\begin{aligned} l(x_i,u_i) &:= \frac{1}{2}\left(x_i^T Q_i x_i + u_i^T R_i u_i\right) + q_i^T x_i + r_i^T u_i \\ l_N(x_N) &:= \frac{1}{2} x_N^T Q_N x_N + q_N^T x_N. \end{aligned} \tag{3}$$

The linearized dynamics and constraints appearing in (2) use the quantities:

$$\begin{aligned} A_i &:= \nabla_x f(x_i^k,u_i^k)^T, \quad B_i := \nabla_u f(x_i^k,u_i^k)^T \\ G_i^x &:= \nabla_x g(x_i^k,u_i^k)^T, \quad G_i^u := \nabla_u g(x_i^k,u_i^k)^T \\ G_N^x &:= \nabla_x g_N(x_N^k)^T \end{aligned} \tag{4}$$

and

$$\begin{aligned} c_i &:= f(x_i^k,u_i^k) - A_i x_i^k - B_i u_i^k \\ d_i &:= g(x_i^k,u_i^k) - G_i^x x_i^k - G_i^u u_i^k, \end{aligned} \tag{5}$$

where quantities with the k superscript denote states and inputs obtained at the previous SQP iteration. The quadratic cost approximation uses the exact gradients

$$\begin{aligned} q_i &:= \nabla_x \eta(x_i^k,u_i^k)\, \eta(x_i^k,u_i^k) \\ r_i &:= \nabla_u \eta(x_i^k,u_i^k)\, \eta(x_i^k,u_i^k) \\ q_N &:= \nabla_x \eta_N(x_N^k)\, \eta_N(x_N^k) \end{aligned} \tag{6}$$

and the Gauss-Newton Hessian with

$$\begin{aligned} Q_i &:= \nabla_x \eta(x_i^k,u_i^k)\, W^x\, \nabla_x \eta(x_i^k,u_i^k)^T \\ R_i &:= \nabla_u \eta(x_i^k,u_i^k)\, W^u\, \nabla_u \eta(x_i^k,u_i^k)^T \\ Q_N &:= \nabla_x \eta(x_N^k)\, W_N\, \nabla_x \eta(x_N^k)^T, \end{aligned} \tag{7}$$

where W^x and W^u denote the Hessian blocks in W associated with states and inputs respectively.

Remark: notice that cost cross-terms coupling states and inputs have been neglected in the QP formulation for simplicity of notation, since the OCP used (which will be described in the next section) has decoupled residuals which do not jointly depend on both states and inputs.

III. Problem Formulation and Implementation

A. Model and Optimal Control Formulation

For the simulations shown in Section IV, the following model [6] will be used:

$$\dot{q} = \frac{1}{2} S^T \Omega, \quad \dot{\Omega} = J^{-1}(T - \Omega \times J\Omega), \tag{8}$$

where q and Ω describe the orientation of the quadrotor expressed in quaternion representation and its angular velocity respectively and

$$S := \begin{bmatrix} -q_1 & q_0 & q_3 & -q_2 \\ -q_2 & -q_3 & q_0 & q_1 \\ -q_3 & q_2 & -q_1 & q_0 \end{bmatrix}. \tag{9}$$

It is assumed that angular velocities of the propellers ω can be tracked instantaneously, hence they are considered as inputs to the system. Moreover, the angular momentum contribution of the propellers is ignored in order to simplify the model. The matrix J denotes the inertia matrix of the vehicle, while the torques applied to the system are described by $T := [T_1\ T_2\ T_3]^T$, with

$$\begin{aligned} T_1 &:= \frac{AC_l\rho(\omega_2^2 - \omega_4^2)}{2}, \quad T_2 := \frac{AC_l\rho(\omega_1^2 - \omega_3^2)}{2} \\ T_3 &:= \frac{AC_d\rho(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2)}{2}, \end{aligned}$$

where ρ is the air density, C_d and C_l are the drag and lift coefficients and A is the area of the propellers. The values of the parameters appearing in the model are listed in Table I.

Parameter	Value	Description
ρ	$1.225\,\mathrm{kg/m^3}$	air density
A	$0.1\,\mathrm{m^2}$	propeller area
C_l	0.125	lift coefficient
C_d	0.075	drag coefficient
m	$10\,\mathrm{Kg}$	quadrotor mass
g	$9.81\,\mathrm{m/s^2}$	gravitational acceleration
$J_1 = J_2 = 4 \cdot J_3$	$0.25\,\mathrm{Kg \cdot m/s^2}$	moments of inertia

TABLE I: quadrotor model - values of the parameter used for the simulation results in Section IV. Notice that these values are fictitious. They have been used for simulation purpose and do not correspond to the parameters of the physical system.

B. Partially Tightened Real-Time Iterations

In order to reduce the computational burden associated with the solution of the QP subproblems (2), the so-called partially tightened RTI scheme proposed in [20] is used. The main idea behind the algorithm is to replace the constraints in the terminal section of the prediction horizon with barrier terms and to perform Newton-type iterations that require the solution of a reduced QP of the form

$$\begin{aligned} \min_{\substack{x_0,\cdots,x_M \\ u_0,\cdots,u_{M-1}}} \quad & \sum_{i=0}^{M-1} l(x_i,u_i) + \psi(x_M) \\ \text{s.t.} \quad & x_0 - \hat{x}_0 = 0 \\ & x_{i+1} - A_i x_i - B_i u_i - c_i = 0,\ i = 0,\cdots,M-1 \\ & d_i + G_i^x x_i + G_i^u u_i \le 0, \qquad i = 0,\cdots,M-1, \end{aligned} \tag{10}$$

with quadratic terminal cost for stage M

$$\psi(x_M) := \frac{1}{2} x_M^T P_M(\mathcal{D}) x_M + p_M(\mathcal{D})^T x_M, \tag{11}$$

where the variables associated with stages $i > M$ (and the inputs for stage $i = M$) have been eliminated using a structure-exploiting Riccati recursion. Notice that in (11) $\mathcal{D}$ (for "data") has been introduced to stress the dependency of $P_M(\mathcal{D})$ and $p_M(\mathcal{D})$ on quantities associated with stages M to N. Effectively, at each iteration, the modified RTI scheme solves a nonlinear root-finding problem where the only source of nonlinearity lies in the complementarity conditions for stages $i = 0$ to $M-1$. This fact can be exploited in order to efficiently eliminate the variables associated with the terminal section of the prediction horizon which yields the positive definite terminal Hessian $P_M(\mathcal{D})$ and terminal gradient $p_M(\mathcal{D})$ in the reduced QP (10).

The description of the details of the algorithm and its implementation goes beyond the scope of the paper and the interested reader is referred to [20] where numerical results and a sketch of a stability proof are provided. In the next section, numerical simulations will be shown that compare the computation times and the closed loop performance of different attitude controllers among which the above described partially tightened and the standard RTI scheme.

IV. Simulation Results

In this section, the performance of different control strategies will be assessed in simulation. In particular, a proportional-derivative controller (PD), a linear-quadratic regulator (LQR) and different variants of an NMPC-based scheme will be taken into account.

A. PD Controller

The first controller taken into account is a PD acting separately on the three Euler coordinates and using a fixed torque allocation as described in [7]. The main idea consists in defining the torque applied to the vehicle using an a priori fixed parametrization that relies on the observation that torques along the three axis can be obtained, loosely speaking, by adjusting the propeller speeds in a "differential" fashion:

$$\begin{aligned} \tau_1 &= \frac{AC_l\rho}{2J_1}(\omega_2^2 - \omega_4^2), \quad \tau_2 = \frac{AC_l\rho}{2J_2}(\omega_3^2 - \omega_1^2) \\ \tau_3 &= \frac{AC_d\rho}{2J_3}(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2). \end{aligned} \tag{12}$$

Additionally, the equation $F_r = (\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2)\frac{AC_l\rho}{2}$ is used to specify the desired total thrust.

Remark: notice that the torque parametrization used completely neglects the inertial terms in the dynamics which depend on the angular velocities of the vehicle. However, for small angular velocities, it can be expected to provide a reasonable approximation of the actual torques.

The chosen approximate parametrization allows one to design three fully decoupled PD controllers that control the attitude on a separate axis each:

$$\tau = K_p e + K_d \dot{e}, \tag{13}$$

where e and $\dot{e}$ are estimates of the roll, pitch and yaw errors and their derivatives respectively and K_p and K_d are the proportional and derivative diagonal matrix gains. Once the desired torque vector τ has been computed according to (13), the squared rotor speeds ω_s can be efficiently computed by solving the following linear system $\frac{A\rho}{2} M \omega_s = t$, where

$$M := \begin{bmatrix} 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \; t := \begin{bmatrix} \frac{\tau_1 J_1}{C_l} & \frac{\tau_2 J_2}{C_l} & \frac{\tau_3 J_3}{C_d} & \frac{gm}{C_l} \end{bmatrix}^T.$$

Notice that, the matrix $\frac{A\rho}{2} M$ is constant and it can be pre-factorized offline in order to speed up the solution of the linear system. The actual rotor speeds can be finally obtained by computing the element-wise square root of ω_s: $\omega_i = \sqrt{\omega_{s,i}}, \quad i = 1,\cdots,4$.

B. LQR Controller

The second controller that will be taken into account is based on a reduced space LQR. Since the attitude dynamics in quaternion coordinates are not controllable, the dynamics will be first projected onto a controllable subspace as proposed in [18]. In particular, using the fact

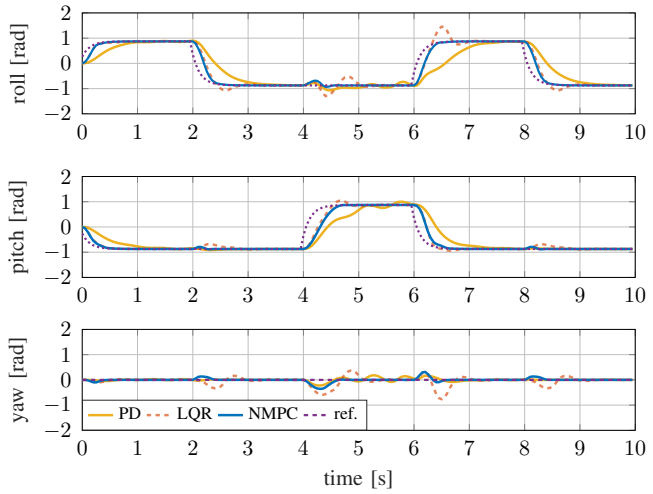

Fig. 1: Attitude tracking simulation results comparing different control strategies - closed-loop state trajectories: PD in solid yellow, LQR in dashed red and converged-NMPC in solid blue. The NMPC controller achieves a more accurate tracking of the reference attitude.

	max CPU time [ms]	avg CPU time [ms]	subopt. [%]
`Ipopt`	131.40	43.40	-
RTI	1.04	0.52	3.69
pt-RTI	0.22	0.18	17.67

TABLE II: maximum and average CPU time in ms and relative closed-loop suboptimality with respect to converged NMPC of the RTI and partially tightened RTI (pt-RTI) schemes. Using the pt-RTI scheme a speedup of about a factor 5 can be achieved with a moderate increase in suboptimality.

that $q_0 = \sqrt{1 - q_1^2 - q_2^2 - q_3^2}$, the first component of the quaternion vector can be eliminated yielding a differential equation which, together with the angular velocity dynamics in (8), will be used to design the LQR static gain. To this end, the dynamics are linearized around the hovering steady state and input $(\bar{x}, \bar{u})$ and discretized using an explicit RK4 integration scheme:

$$x_{k+1} - \bar{x} = A(x_k - \bar{x}) + B(u_k - \bar{u}). \tag{14}$$

At this point, a discrete-time LQR controller can be designed by solving the discrete time algebraic Riccati equation

$$A^T P A - P - (A^T P B)(B^T P B + R)^{-1}(B^T P A) + Q = 0,$$

which provides a static state feedback $u = Kx + \bar{u}$ with

$$K = (B^T P B + R)^{-1}(B^T P A), \tag{15}$$

which, once an estimate of the state of the system has been computed, allows one to readily compute the input to be applied to the system.

C. NMPC Controller

Three different variants of NMPC-based controllers will be compared in simulation. All of them use a nonlinear least-squares formulation with residual functions

$$\eta(x,u) := \begin{bmatrix} \alpha(x) - \alpha_r \\ \beta(x) - \beta_r \\ \gamma(x) - \gamma_r \\ x - x_r \\ u - u_r \end{bmatrix}, \quad \eta_N(x) := \begin{bmatrix} \alpha(x) - \alpha_r \\ \beta(x) - \beta_r \\ \gamma(x) - \gamma_r \\ x - x_r \end{bmatrix}, \tag{16}$$

where α, β and γ define the attitude of the quadrotor in Euler angles (roll, pitch and yaw) as functions of q. The quantities in (16) with the r subscript denote the desired references associated with each residual output.

1) Converged NMPC scheme: the problem formulation in (1) has been implemented using `CasADi` [5] with a prediction horizon of $T = 1.0\,\text{s}$ and $N = 20$ shooting nodes, discretizing the dynamics using the explicit RK4 integration scheme. The obtained OCPs are solved using the interior-point solver `Ipopt` [17]. The `Ipopt` interface available in `CasADi` is used where the possibility of *just-in-time* compiling function evaluations is exploited in order to speed up the computations. The linear system solver `ma57` [9] is used linked against a single threaded build of the high-performance BLAS implementation `OpenBLAS` [15]. Although real-time implementations of nonlinear interior-point methods are present in the literature [19], the computation times obtained when solving the OCPs to a local minimum are often longer than the ones obtained with approximate schemes like the RTI. However, the closed loop trajectories obtained with this approach will be used as a reference to assess the suboptimality associated with the approximate schemes described below.

2) Standard RTI scheme: in order to reduce the computation times associated with the solution of the OCPs, the RTI scheme has been implemented using the software package `acados` [1]. The same number of shooting nodes and the same tuning has been used for the implementation. The chosen QP solver is `hpmpc` [12] which relies on the hardware-tailored linear algebra package `BLASFEO` [11].

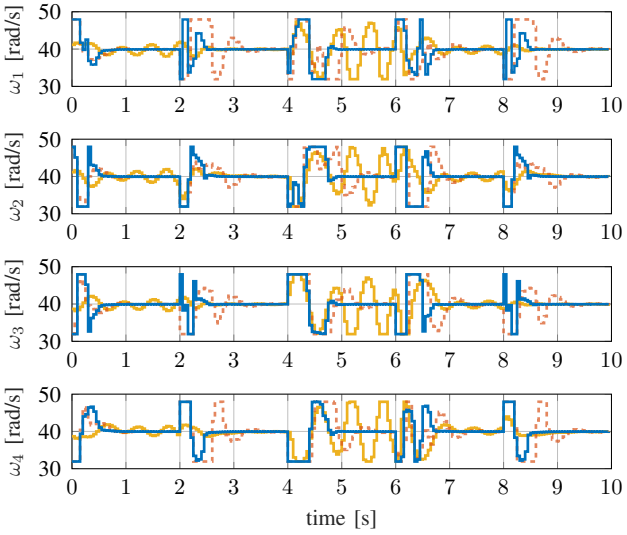

Fig. 2: Attitude tracking simulation results comparing different control strategies - closed-loop input trajectories.

3) Partially Tightened RTI scheme: finally, the partially tightened RTI scheme (pt-RTI) described in Section II has been implemented in `acados` using an untightened horizon of $M = 2$ stages and an overall horizon of $N = 20$. The same tuning used for the previous two schemes is used and a fixed barrier value $\tau = 10$ is used. As for the standard RTI scheme, the solver `hpmpc` will be used to solve the reduced QP (10) as well as to perform the Riccati-based elimination for the terminal section of the horizon described in [20].
The code in the following simulations is set up to use the ANSI C implementation `BLASFEO RF` [11] in order to better resemble the CPU load distribution between different routines (e.g. linearization and QP solution) expected on the embedded hardware.

D. Comparison

The controllers described above are used in the following to track a periodic attitude reference. For the NMPC formulations the following weights are chosen:

$$\begin{aligned} W &= \text{blkdiag}(5 \cdot 10^2 \cdot \mathbf{I}_3,\, 1 \cdot 10^{-3} \cdot \mathbf{I}_{11}) \\ W_N &= \text{blkdiag}(5 \cdot 10^2 \cdot \mathbf{I}_3,\, 1 \cdot 10^{-3} \cdot \mathbf{I}_7). \end{aligned}$$

In order to tune the PD controller, the parametrization $K_p = \kappa_p \cdot \mathbf{I}_3$ and $K_d = \kappa_d \cdot \mathbf{I}_3$, with $\kappa_p \in [1, 60]$ and $\kappa_d \in [1, 20]$ has been chosen. After discretizing each parameter interval into 100 equidistant values, a simulation has been run for each combination of values (κ_p, κ_d) and the squared deviation from the reference trajectories in the Euler space has been taken into account as a performance metric. The values $\kappa_p = 23$ and $\kappa_d = 9$ have been chosen, which provide a reasonable trade-off between deviation from the reference trajectory and chattering of the input trajectories. The LQR controller could be in principle tuned by exploiting a linearization of the transformation from Euler to the controllable quaternion subspace. In this way, the control policy obtained would be locally equivalent to the NMPC one. However, possibly due to nonlinearity and to the presence of constraints, the simulations showed the necessity to detune the controller in order to achieve acceptable performance. To this end, the weighting matrices have been chosen as follows:

$$Q = \text{blkdiag}(1 \cdot 10^2 \cdot \mathbf{I}_3,\, 1 \cdot 10^{-3} \cdot \mathbf{I}_3),\; R = 1 \cdot 10^{-3} \cdot \mathbf{I}_4.$$

A simulation with sampling time $T_s = 0.05\,\text{s}$ is performed where the input bounds are imposed on the propeller velocities: $\omega_{\text{ss}} - \Delta\omega_{\max} \leq \omega_i \leq \omega_{\text{ss}} + \Delta\omega_{\max}$, with $\Delta\omega_{\max} := 8\,\text{rad/s}$ and where $\omega_{\text{ss}} := 39.939\,\text{rad/s}$ denotes the steady-state input associated with a mass of 10 Kg (although the controller only regulates the vehicle's attitude, it is meant to be used in a cascaded architecture, where also position is controlled).
The closed loop trajectories obtained with PD, LQR and NMPC controllers are reported in Figure 1 and 2. Only trajectories obtained with `Ipopt` are shown for the sake of clarity, as the results obtained with the RTI and pt-RTI schemes do not differ much from the ones obtained using converged NMPC. For the two approximate schemes, computation times and closed-loop suboptimality are reported in Table II. From Figure 1 and 2, it can be seen that the converged NMPC controller performs better than the other two control schemes in the sense that smaller overshoots and faster response to references changes can be achieved. This might be due to the fact that nonlinearity and the presence of constraints can degrade the performance of the PD and LQR controllers for large reference changes like the ones used in the benchmark.

Fig. 3: Human-sized quadrotor equipped with a low-power Xilinx Zynq SoC with a dual-core ARM Cortex A9 running at 800 MHz: snapshot from the experiment video (https://www.youtube.com/watch?v=-dsezQa7nzk&feature=youtu.be).

V. Experimental Results

The controller based on the pt-RTI scheme has been deployed to the on-board embedded hardware of the quadrotor which features a Xilinx Zynq system-on-chip with dual-core Cortex A9 clocked at 800 MHz. Notice that, although the instruction set available on such a CPU provides vectorized instructions, they are only available in single precision. Hence, the `GENERIC` implementation `BLASFEO` package [11] has been used, which exploits a panel-major format, but does not make explicit usage of vectorized instructions.
For the embedded implementation, a horizon $T = 1.0\,\text{s}$ is used with $N = 10$ shooting nodes and untightened horizon $M = 2$. In order to achieve faster response to changes in the reference and disturbance rejection, and to improve the convergence of the pt-RTI scheme, the controller is run at a sampling $T_s = 10\,\text{ms}$. Notice that, although to the knowledge of the authors a formal stability proof for this setup does not exist, the "over-sampled" implementation of NMPC schemes is rather common among practitioners.
Similarly to what obtained in simulations, using the pt-RTI scheme with $M = 2$ gives rise to a considerable speedup reducing the average computation times from about 6 ms for the standard RTI to about 2 ms. In this way, enough computational time can be spared to carry out other tasks such as telemetry, logging and executing lower level controllers without approaching high CPU loads which might lead to faults.
Figures 4 and 5 show the attitude and actuators trajectories obtained during a test flight. Notice that, since no position control has been implemented for the test, the attitude reference during the test flight is provided by a human pilot who is making sure that the vehicle is hovering safely above the ground.

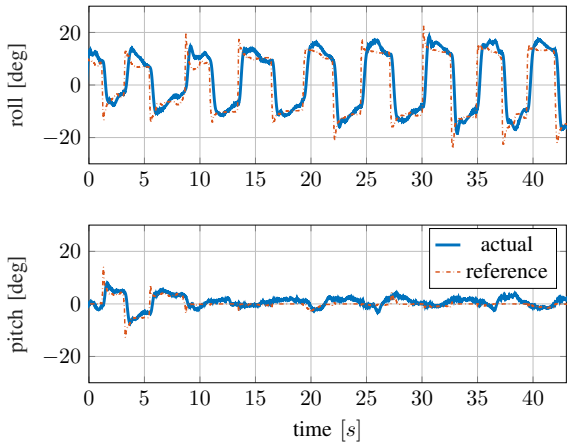

Fig. 4: Experimental results - attitude in Euler angles.

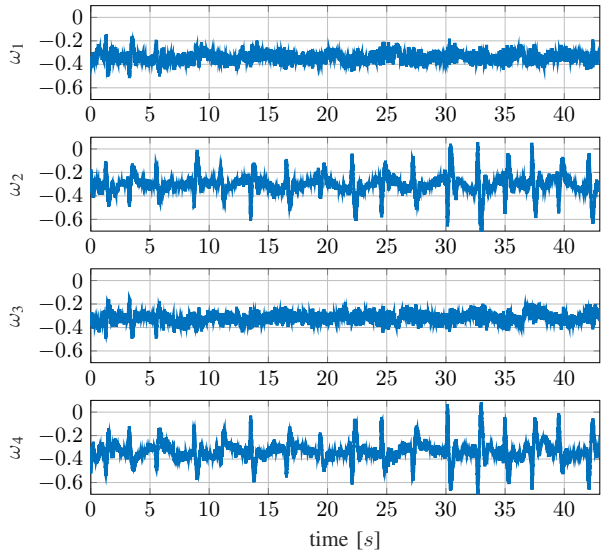

Fig. 5: Experimental results - actuators.

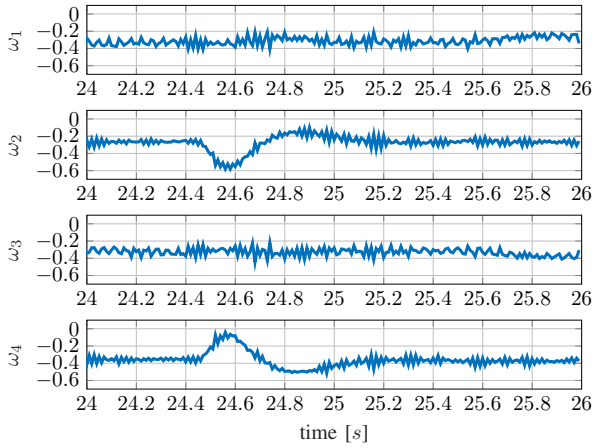

Fig. 6: Experimental results - actuators (zoom in).

VI. Conclusions

In this paper, the design, implementation and deployment of an NMPC attitude controller for a human-sized quadrotor has been presented. An approximate partially tightened formulation is used that allows one to reduce the computation times. Simulation results are discussed where it is shown that considerable speedups can be achieved with a moderate increase in suboptimality with respect to standard approaches. The NMPC controller is deployed to the on-board computer of the vehicle and flight test results are reported and discussed.

References

[1] acados. https://github.com/acados/acados.

[2] Ehang Inc. Guangzhou, China. http://www.ehang.com.

[3] Kitty Hawk, Mountain View, CA, USA. https://kittyhawk.aero.

[4] T. Albin, D. Ritter, N. Liberda, R. Quirynen, and M. Diehl. In-vehicle realization of nonlinear MPC for gasoline two-stage turbocharging airpath control. *IEEE Transactions on Control Systems Technology*, pages 1–13, 2017.

[5] J. Andersson, J. Akesson, and M. Diehl. CasADi – a symbolic package for automatic differentiation and optimal control. In *Recent Advances in Algorithmic Differentiation*, volume 87 of *Lecture Notes in Computational Science and Engineering*, pages 297–307. Springer, 2012.

[6] P. Betsch and R. Siebert. Rigid body dynamics in terms of quaternions: Hamiltonian formulation and conserving numerical integration. *International Journal for Numerical Methods in Engineering*, 79(4):444–473, 2009.

[7] S. Bouabdallah, A. Noth, and R. Siegwart. Pid vs lq control techniques applied to an indoor micro quadrotor. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, Oct 2004.

[8] M. Diehl, H. G. Bock, J. P. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer. Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *Journal of Process Control*, 12(4):577–585, 2002.

[9] I. Duff. Ma57—a code for the solution of sparse symmetric definite and indefinite systems. *ACM Transactions on Mathematical Software*, 30(2):118–144, June 2004.

[10] H. J. Ferreau, P. Ortner, P. Langthaler, L. del Re, and M. Diehl. Predictive control of a real-world diesel engine using an extended online active set strategy. *Annual Reviews in Control*, 31(2):293–301, 2007.

[11] G. Frison, D. Kouzoupis, T. Sartor, A. Zanelli, and M. Diehl. BLASFEO: Basic linear algebra subroutines for embedded optimization. *arXiv:1704.02457*, 2017.

[12] G. Frison, H. B. Sorensen, B. Dammann, and J. B. Jørgensen. High-performance small-scale solvers for linear model predictive control. In *Proceedings of the European Control Conference (ECC)*, pages 128–133, June 2014.

[13] M. Kamel, K. Alexis, M. Achtelik, and R. Siegwart. Fast nonlinear model predictive control for multicopter attitude tracking on so(3). In *IEEE Conference on Control Applications*, Sidney, Australia, Sept 2015.

[14] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, 2 edition, 2006.

[15] OpenBLAS. OpenBLAS: An optimized BLAS library. http://www.openblas.net/, 2011.

[16] S.J. Qin and T.A. Badgwell. An overview of nonlinear model predictive control applications. In F. Allgöwer and A. Zheng, editors, *Nonlinear Predictive Control*, volume 26 of *Progress in Systems Theory*, pages 370–392, Basel Boston Berlin, 2000. Birkhäuser.

[17] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[18] Y. Yang. Spacecraft attitude determination and control: Quaternion based method. *Annual Reviews in Control*, 36(2):198–219, Dec 2012.

[19] A. Zanelli, A. Domahidi, J. L. Jerez, and M. Morari. FORCES NLP: An efficient implementation of interior-point methods for multistage nonlinear nonconvex programs. *International Journal of Control*, 2017.

[20] A. Zanelli, R. Quirynen, G. Frison, and M. Diehl. A partially tightened real-time iteration scheme for nonlinear model predictive control. In *Proceedings of 56th IEEE Conference on Decision and Control*, Melbourne, Australia, December 2017.