Model-In-the-Loop Testing of Control Systems and Path Planner Algorithms for QuadRotor UAVs

Model-In-the-Loop Testing of Control Systems and Path Planner Algorithms for QuadRotor UAVs

Iris David Du Mutel de Pierrepont Franzetti[1], Davide Carminati[2], Matteo Scanavino[2], and Elisa Capello[3]

***Abstract*— Real systems, as Unmanned Aerial Vehicles (UAVs), are usually subject to disturbances and parametric uncertainties, which could compromise the mission accomplishment, considering particularly harsh environments or challenging applications. For this reason, the main idea proposed in this research is the design of the on-board software, as autopilot software candidate, for a multirotor UAV. In detail, the inner loop of the autopilot system is designed with a variable structure control system, based on sliding mode theory, able to handle external disturbances and uncertainties. This controller is compared with a simple Proportional-Integral-Derivative controller. The key aspects of the proposed methodology are the robustness to bounded disturbances and parametric uncertainties of the proposed combination of guidance and control algorithms. A path-following algorithm is designated for the guidance task, which provides the desired waypoints to the control algorithm. Model-in-the-loop simulations have been performed to validate the proposed approaches. Computationally efficient algorithms are proposed, as combination of a robust control system and path planner. Extensive simulations are performed to show the effectiveness of the proposed methodologies, considering both disturbances and uncertainties.**

***Index Terms*— Indoor applications for UAVs, Bezier-based Trajectory Planner, Robust GNC, Model-In-the-Loop Simulations.**

I. Introduction

Indoor Unmanned Aerial Systems (UASs) are widely used in a variety of applications. They can be exploited from industrial scenarios to monitoring and supervising critical operations. Initially developed for military purposes [1], drones have become popular for scientific research also, as for remote sensing and mapping. In particular, the main challenge is related to the GPS-denied scenario, that is emerging nowadays due to the advancement in technology and the flexibility of these platforms. As clearly explained in [2], the majority of the commercial UASs are able to operate in an outdoor environment. The recent development and spreading of Unmanned Aerial Vehicles (UAVs) in both commercial and civil fields is due to the many possible applications and activities in which they can get involved. In the commercial field, they can be used to supervise warehouse lots, advertisement or even delivering products and services. On the other hand, the civil uses are oriented towards no profit-making objectives such as traffic management or pollution tracking in big cities. It has to be noted that the characteristics of drones make them suitable for rescue tasks after catastrophic events or surveillance missions. All these applications constitute just a few of the possible roles drones can play, that not only help humans but most times prevent them from being exposed to hazardous conditions. Even then, since drone industry grows exponentially, social and ethical aspects of their use must be evaluated [3].

The main objective of this research is the evaluation and design of a computational efficient software, which represents the on-board algorithms of a multirotor UAV. The selected algorithms are deployed on the on-board autopilot and tested for indoor applications, combined with a dedicated sensor. To properly evaluate the reliability of the proposed guidance and control scheme, the classical multi-step software verification & validation approach for model-based design has been followed. As described in [4], the first step consists in performing Model-In-the-Loop (MIL) testing, i.e. executing the controller algorithm for non-real-time execution on the same host platform that is used by the modeling environment. The next step will involve the Processor-In-the-Loop (PIL) testing, during which the model does a single calculation iteration. Then, inputs are calculated and passed to a code running on the embedded microprocessor.

The proposed approach combines the waypoint-based guidance strategy proposed in [5], [6] with three tracking-based schemes: (1) a full Proportional-Integral-Derivative (PID) [7] control and (2-3) two combined structures including a Linear Quadratic Regulator (LQR) [8] and a first order Sliding Mode Controller (SMC) [9], to guarantee tracking of desired waypoints. Moreover, a two loop control scheme is proposed to improve the UAV performance with respect to classical approaches, including different control laws for the fast dynamics (i.e. inner loop dynamics). As previously said, a multi-step software approach for model-based design has been followed.

The main parameters of the quadrotor, as moments of inertia or thrust/torque curves, are experimentally identified. Starting from the work of [10], moments of inertia are identified with a compound pendulum method [11], which

[1]Iris David Du Mutel de Pierrepont Franzetti is with Department of Signal Theory and Communications, Polytechnic University of Catalonia, Barcelona, Spain iris.david.du.mutel.de.pierrep@estudiant.upc.edu

[2]Davide Carminati and Matteo Scanavino are with Department of Mechanical and Aerospace Engineering, Politecnico di Torino, Torino, Italy, davide.carminati, matteo.scanavino@polito.it

[3]Elisa Capello is with Department of Mechanical and Aerospace Engineering, Politecnico di Torino, CNR-IEIIT, Torino, Italy, elisa.capello@polito.it

is a model-based approach. This method is an open loop identification method, in which a dedicated sensor is used for the measurements of pendulum physical parameters. Moreover, the thrust and torque fitting curves are identified with a test benchmark [12], as function of the selected motors and propellers.

In our paper, the limited computational power of the on-board systems is simulated considering low update frequency of the controller, including actuation limitations. As described in [13], the sampling speed is one issue to take into account, since the accuracy of the proposed strategy is function of the maximum attainable sample frequency. One key aspect of the proposed strategies is to show the error accuracy, in terms of trajectory tracking, even when limited-capacity hardware is available. In this paper, hardware constraints, in which a reduced update frequency and saturation on the actuation system, are included. The key features of this proposed on-board software are the preliminary validation of computationally-efficient robust control systems, in which a high-level software is used for the definition of them. The controllers are translated in C/C++ language in order to be deployed in the on-board software and tested by flights. This means that the default firmware is updated with the designed control laws. The novelty of this research is twofold: (i) combination of computational efficient algorithms, able to be easily translated to on-board software and deployed on the autopilot board and (ii) model-in-loop simulations are performed, including identification of the quadrotor parameters and state estimation with a dedicated sensor. Moreover, the quadrotor is connected by Wi-Fi with a ground control station, in order to include sensor measurements in the simulations.

The paper is organized as follows. The prototype of the Quadrotor is introduced in Section II, in which the description of the autopilot and dedicated sensors are included. The Simulation environment is deeply described in Section III. All the elements of the complete simulator are detailed in this Section. Section IV includes the path planners, analyzed in this paper. In a similar way, the cascade controllers are in Section V. MIL Simulations are detailed in Section VI. The simulations parameters and the computational effort are included in this section. Finally, conclusions are drawn in Section VII.

II. Real-World Quadrotor Prototype

This work is based on the prototype quadrotor depicted on Figure 1, which was developed at Politecnico di Torino, in the Flight Mechanics Research Group (Dept. of Mechanical and Aerospace Engineering). The quadrotor prototype was designed for sensor integration in GPS-denied applications [14] and propeller characterization in non- conventional environments (i.e. harsh environments) [15].

The parameters, used for the design of the simulation model, are experimentally identified and the exploited software architecture features the main elements of the PX4 Flight Stack. The quadrotor is mainly used for indoor flights

Fig. 1: Quadrotor prototype

in GPS-denied environment and it is equipped with the Pixhawk 2.1 Cube Autopilot (Figure 2a) running the PX4 Flight Stack and a MoCap sensor - the Otus Tracker (Figure 2b) - as a replacement for the GPS sensor.

(a) On-board Autopilot (b) Otus Tracker

Fig. 2: On-board system and dedicated sensor

The autopilot is an ARM Cortex M4 based embedded system running the Nuttx Real-Time operating system that executes the PX4 Flight Stack processes [16]. The PX4 is equipped with an Inertial Measurement Unit (IMU), which includes three redundant accelerometers, gyroscopes and compasses and two redundant barometers. The advantage of using this software framework is the possibility of customization of the control laws using high-level software such, as Simulink. The Wi-Fi communications with the ground station PC are enabled by an on board Raspberry Pi 3B. The communication link and all the devices, used for the communication, are in Figure 3.
The quadrotor external frame is in carbon fiber except for the arms, which are in aluminum alloy. The size and the used convention are shown in Figure 4.

Brushless electrical motors by T-motor MN2212-18 and propellers by HQProp $10'' \times 4.5''$ are used.

III. Simulation Environment

The simulation environment is used for designing, tuning and validating the GNC software developed for the considered UAV. The overall scheme of this environment is shown

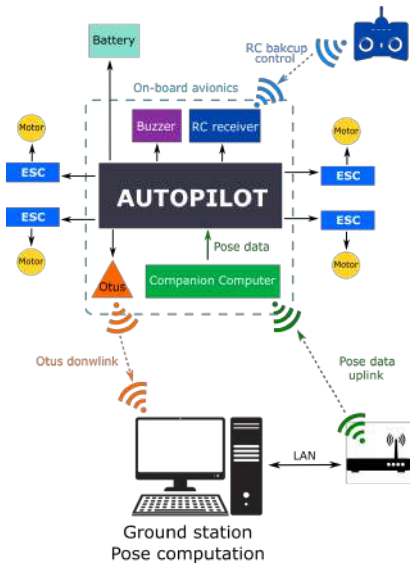


Fig. 3: On-board connection scheme and avionics

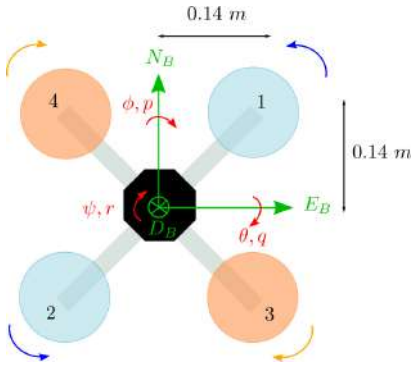


Fig. 4: Main dimensions of the prototype UAV

in Figure 5, in which the main features of the firmware software structure are included. Moreover, the quadrotor main geometric parameters are experimentally determined from the prototype, as we detail later in this Section. Furthermore, this framework allows the *Controller* Simulink block to be easily translated to C/C++ language and deployed on the quadrotor autopilot.

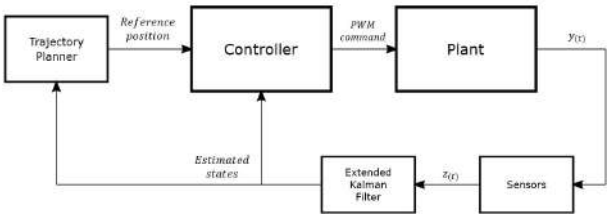


Fig. 5: Simulation block diagram

As a consequence, the signal provided by *Controller* block is the duty cycle of the Pulse Width Modulation (PWM) signal sent to the motors, i.e. the signal sent to the "real" system. The *Plant* block contains the mathematical model of the quadrotor dynamics and the experimental model of the actuators, the *Sensors* block models the uncertainty given by the employed sensors, the *Extended Kalman Filter* block performs data fusion and states estimation, the *Trajectory Planner* block provides the controller the trajectory. The model is implemented in MATLAB®/Simulink®R2019a.

A. Dynamics identification - Plant

The plant block includes the actuators model and the quadrotor equations of motion . The former converts the duty cycle values coming from the *Controller* block into physical quantities compatible with the latter.

1) Quadrotor dynamical model: The dynamical model is based on the well-known equations of motion of a 6 Degrees-of-Freedom body [17], [18]. No external disturbances or aerodynamic forces are considered in the formulation. The state vector is: $\mathbf{x} = \begin{bmatrix} p_N & p_E & h & \phi & \theta & \psi & u & v & w & p & q & r \end{bmatrix}^\intercal$. The nonlinear mathematical model can be written in the form: $\dot{\mathbf{x}} = \mathbf{f}_{(x)}$ where:

$$\mathbf{f}_{(x)} = \begin{bmatrix} \mathbf{R}_{body}^{NED} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \\ p + \tan\theta(q\sin\phi + r\cos\phi) \\ q\cos\phi - r\sin\phi \\ \frac{1}{\cos\theta}(q\sin\phi + r\cos\phi) \\ rv - qw - g\sin\theta + \frac{F_x}{m} \\ -ru + pw + g\sin\phi\cos\theta + \frac{F_y}{m} \\ qu - pv + g\cos\phi\cos\theta + \frac{F_z}{m} \\ q(c_1 r + c_2 p) + c_3\tau_x + c_4\tau_z \\ prc_5 - (p^2 - r^2)c_6 + c_7\tau_y \\ q(c_8 p - c_2 r) + c_4\tau_x + c_9\tau_z \end{bmatrix}, \quad (1)$$

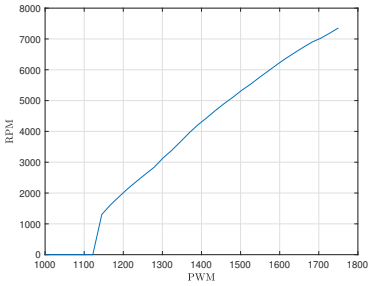
in which $\mathbf{R}_{body}^{NED}$ is the transformation matrix between the body coordinates and the North-East-Down (NED) inertial coordinates. Finally, the coefficients c_i are:

$$c_1 = \frac{J_{xz}(J_x - J_y + J_z)}{J_x J_y - J_{xz}^2} \quad c_2 = \frac{J_{xz}(J_x - J_y + J_z)}{J_x J_y - J_{xz}^2} \quad c_3 = \frac{J_z}{J_x J_y - J_{xz}^2}$$
$$c_4 = \frac{J_{xz}}{J_x J_y - J_{xz}^2} \quad c_5 = \frac{J_z - J_x}{J_y} \quad c_6 = \frac{J_{xz}}{J_y}$$
$$c_7 = \frac{1}{J_y} \quad c_8 = \frac{J_{xz}^2 + J_x(J_x - J_y)}{J_x J_y - J_{xz}^2} \quad c_9 = \frac{J_x}{J_x J_y - J_{xz}^2}$$

.

2) Actuator experimental modeling: A test bench is used to map the duty cycle value provided by the *Controller* block with the corresponding value of thrust and torque generated by motors and propellers. The RCBenchmark Series 1520 Thrust Stand (Figure 6a) [12] is able to record electrical and mechanical values from the motor and propeller set. The relationship between the PWM signal and the angular speed expressed in RPM (Figure 6b) is linear in the region in which propellers actually spin. Figure 7a and Figure 7b show the curve fitting that allows the conversion from the

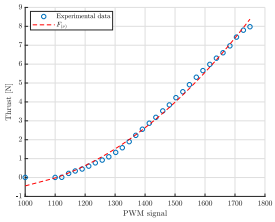
(a) Test bench



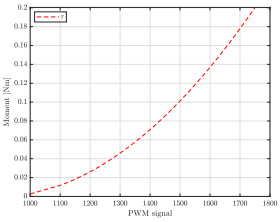
(b) Relationship between duty cycle and angular speed

Fig. 6: Identification benchmark for motors and propellers

duty cycle value into physical values of the input to the dynamical equations.



(a) Relationship between duty cycle and propellers thrust



(b) Relationship between duty cycle and propellers torque

Fig. 7: Experimental evaluation of thrust and torque

B. Sensors

The *Sensors* block reproduces the uncertainties of the on-board equipment, modeled as *white noise* and a bias as reported in [19]. Moreover, it provides as output only the available states (i.e. measures), that are inputs of the Extended Kalman Filter (EKF). The output of this block can be detailed as follows:

$$\mathbf{z}_{(t)} = \mathbf{h}_{(\mathbf{x},t)} = \begin{bmatrix} M_{acc}\begin{bmatrix} rv - qw - g\sin\theta \\ -ru + pw + g\sin\phi\cos\theta \\ qu - pv + g\cos\phi\cos\theta + \frac{F_z}{m} \end{bmatrix} S_{acc} + \Delta_{acc} + \mathbf{v}_{acc} \\ M_{gyro}\begin{bmatrix} p \\ q \\ r \end{bmatrix} S_{gyro} + \Delta_{gyro} + \mathbf{v}_{gyro} \\ -\frac{gh}{RT_0} \\ k_B p_0 e + \Delta_{baro} + v_{baro} \\ M^{pos}_{Otus}\begin{bmatrix} p_N \\ p_E \\ h \end{bmatrix} S^{pos}_{Otus} + \Delta^{pos}_{Otus} + \mathbf{v}^{pos}_{Otus} \\ M^{att}_{otus}\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} S^{att}_{Otus} + \Delta^{att}_{Otus} + \mathbf{v}^{att}_{Otus} \end{bmatrix}. \tag{2}$$

$\Delta_{(*)}$ is the sensor bias, $\mathbf{v}_{(*)}$ is the sensor noise and $M_{(*)}$ and $S_{(*)}$ are the misalignment and scale factor matrices defined as:

$$M_{(*)} = M^T_{(*)} = \begin{bmatrix} 1 & M_{xy} & M_{xz} \\ M_{yx} & 1 & M_{yz} \\ M_{zx} & M_{zy} & 1 \end{bmatrix} \quad S_{(*)} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{bmatrix}$$

C. Extended Kalman Filter

The *Extended Kalman Filter* block performs data fusion of the redundant data provided by the Otus Tracker and state estimation (by EKF and sensors). It is a discrete-time filter, starting from the nonlinear system

$$\begin{cases} \mathbf{x}_{(k+1)} = \mathbf{f}^{DT}_{(\mathbf{x}_{(k)},\mathbf{u}_{(k)})} + \mathbf{v}_1 \\ \mathbf{y}_{(k)} = \mathbf{h}^{noiseless}_{(\mathbf{x}_{(k)})} \\ \mathbf{z}_{(k)} = \mathbf{h}_{(\mathbf{x}_{(k)})} \end{cases}, \tag{3}$$

where $\mathbf{v}_1$ is the process noise with variance V_1. The variance V_1 is determined with a trial and error method, while V_2 - related to the noises in Equation 2 - is determined experimentally exploiting the data collected during test flights. The blocks $F_{(k-1)}$ and $H_{(k)}$ are defined as

$$F_{(k-1)} = \left.\frac{\partial \mathbf{f}^{DT}}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{(k-1|k-1)}} \qquad H_{(k)} = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{(k|k)}}.$$

D. Identification of moments of inertia

In this work, experimental values have been obtained as a result of several tests using a pendulum. A similar procedure as in [10] has been followed, obtaining the equations of motion from the Lagrangian of the system. The moments of inertia can be computed from Equation 4:

$$I = \left(\frac{T}{2\pi}\right)^2 (mg\frac{l_1}{2} + m_1 g(l_1 + d)) - m\frac{l_1^2}{4} - m_1(l_1 + d)^2 - I_{rod}, \tag{4}$$

where m is the mass of the bar, g is the value of acceleration of gravity, l_1 is the distance between the fixed point of the pendulum and the Center of Gravity (CoG) of the bar, m_1 is the mass of the quadrotor, d is the distance between the CoG of the bar and the Center of Mass (CoM) of the quadrotor. We measure the angle of oscillation of the pendulum θ and the period of oscillation T with a dedicated sensor. The scope of this method is to identify I_{rod} and I, which are the moments of inertia of the rod and the quadcopter, respectively. So, the angle of the pendulum is evaluated with the Otus Tracker and the results are recorded through RCbenchmark Tracking Lab interface. Positions and velocities have been also recorded. Ten experiments are carried out for each axis of the quadrotor. In figures 8 and 9 the measurements of yaw and pitch angles during the evaluation of I_{xx} and I_{zz}, respectively, are exposed.

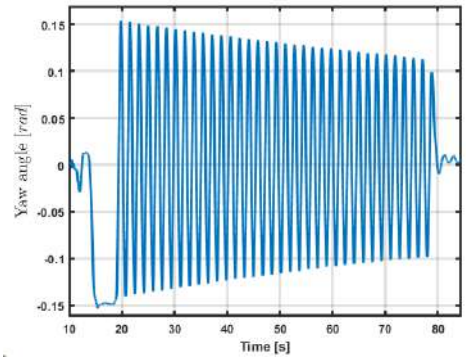


Fig. 8: Yaw angle measurement, for the evaluation of I_{zz}

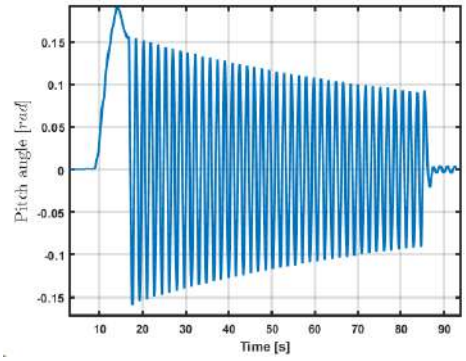


Fig. 9: Pitch angle measurement, for the evaluation of I_{xx}

Results from testing determine that the values of inertia are 0.0331 kgm^2 for I_{xx}, 0.0338 kgm^2 for I_{yy} and 0.0504 kgm^2 for I_{zz}.

IV. Trajectory Planner

A trajectory planner is designed with the aim of generating position and velocity references to be tested with the proposed controllers. For this purpose, a list of waypoints and their corresponding times of arrival must be provided beforehand within the 2D x-y plane.

In this work, two alternatives concerning the shape of the trajectory are exploited: Dubins curves and Bézier curves. For the Dubins planner, circle arcs and straight lines are defined to pass over the target points [20], while the latter allow a more flexible design, choosing the shape and length of the curves [21]. What is sought through the use of curves in the trajectory planning approach is the continuity of the references obtained from it. This can be addressed from two different perspectives: (i) continuity within the same curve or (ii) at the joint of two consecutive curves. The first case corresponds to function continuity, where a function of class C^n with $n \geq 1$ denotes the existence of its partial derivatives of order n, all of them continuous. On the other hand, the continuity in the union of curves is defined as G_i or geometric continuity. It implies the existence and matching of the derivatives of order i at the joining point of two independent curves. A compound curve is G_0 continuous if both sections coincide in the position of the joining point. The continuity degree can be increased as additional restrictions are added up. For planar curves, the maximum curvature degree accessible is G_2 [22].
Depending on the trajectory planning approach, references will be computed differently. Only vertical displacement is addressed identically in both cases. The vertical speed is computed as a trapezoidal profile and the position reference is obtained through its integration.

A. Dubins curves

A Dubins curve is defined as the shortest path between two points with an initial and final heading, respectively [20]. Such curves can be composed by three different fragments: a right-oriented curve, a left-oriented curve or a straight line. This arrangement ensures continuity in position, but this can not be said about curvature. When switching from a straight segment of curvature $\kappa = 0$ to a circular segment with $\kappa = \frac{1}{R}$, with R being the predefined radius of the curved segments, the continuity is lost. If a trajectory is designed using Dubins curves, the heading of the waypoints should be indicated beforehand by the user. The velocity profiles have been computed using the times of arrival together with an imposed trapezoidal profile. In Figure 10, v_{max} is the

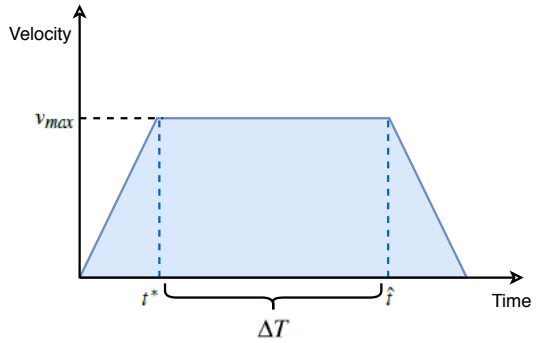


Fig. 10: Trapezoidal velocity profile

maximum value of velocity in a given direction, t^* is the time where the maximum velocity is reached, ΔT is the period in which the velocity is maximum and at $\hat{t}$, the velocity starts decreasing.

The heading reference ψ_{des} is comprised by the values of heading at each point of the trajectory. Then, the turning rate is computed as the discrete derivative of the heading as in Equation 5,

$$\dot{\psi}_k = \frac{\psi_{k+1} - \psi_k}{\delta t}, \tag{5}$$

where δt is the time interval between two consecutive points of the trajectory.

B. Bézier curves

The output of the Bézier planner is a series of points, which guide the quadrotor to the final point. Given start and ending coordinates, a set of intermediate points is arranged to define the curve's length and curvature. This guarantees a

more accurate tracking of the desired points. A Bézier curve of degree n is defined by Equation 6:

$$B(t) = \sum_{i=0}^{n} \binom{n}{i} P_i (1-t)^{n-i} t^i, \tag{6}$$

where $t \in [0, 1]$, being t the parameter of the curve [23]. In this work, cubic Bézier curves have been used in order to guarantee G_1 continuity of the whole path, implying continuity in position and in its first derivative.

A Bézier curve is continuous as it is infinitely differentiable, but, when it comes to joining two consecutive curves, differentiability is highly desirable but not always achieved. In this case, some geometrical constraints can be added to ensure such condition.

Let Q and R be two Bézier curves of the same order n defined by $(Q_0, Q_1, ..., Q_n)$ and $(R_0, R_1, ..., R_n)$, respectively. The continuity in position (i.e, G_0 continuity) is satisfied when condition 7 is fulfilled.

$$R_0 = Q_n. \tag{7}$$

Moreover, to achieve G_1 continuity, we have to ensure $Q(1)' = R(0)'$. Making the appropriate derivation and substitution of the t values, the remaining condition is 8:

$$Q_n - Q_{n-1} = R_1 - R_0. \tag{8}$$

Substituting 7 in 8, results in:

$$R_1 = 2Q_n - Q_{n-1}. \tag{9}$$

This condition is the same as imposing collinearity between the last control point of the previous segment, the link point of the two curves and the first control point of the following segment (see Figure 11).

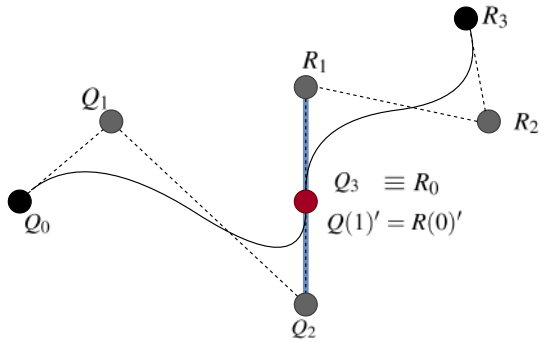


Fig. 11: Collinearity condition in cubic Bézier curves

Once the path has been generated, the velocity profiles are computed as the derivative of the position in time. The yaw angle reference is computed as the derivative of the curve w.r.t the parameter t. Finally, the rate of change of the yaw angle $\dot{\psi}_{des}$ is obtained as the time derivative of the heading.

V. Design of Controllers

In this work, three different approaches have been considered regarding the control of the quadrotor: (1) a full PID control [7] and (2-3) two combined structures including a Linear Quadratic Regulator (LQR) [8] and a first order Sliding Mode Controller [24].

A. PID

The PID controller is implemented using a cascade architecture, in which a slower outer loop and a faster inner loop are present [25]. This structure is described in Figure 12. The position control is handled with outer PD control system

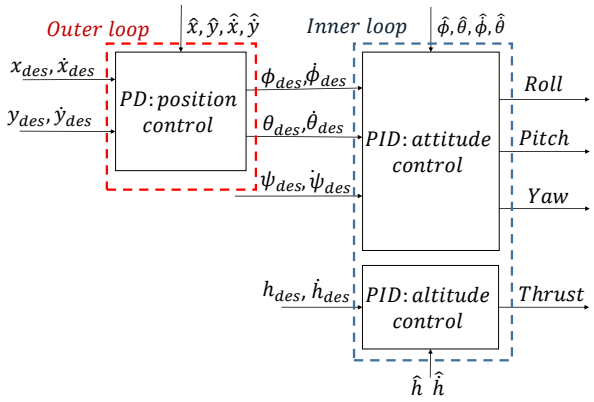


Fig. 12: PID-based cascade controller scheme

and, on the other hand, the attitude and altitude variables are controlled by inner PID control system. Note that, for each variable (i.e. x, y, ϕ, θ, ψ and h), an independent controller has been used.

B. LQR

Considering the state space representation of the linearized model

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned}, \tag{10}$$

where the corresponding values in the matrices A, B, C and D are in Appendix I. As well known, a state feedback policy of the form $u = -K_{LQR} \cdot x$ such that minimizes a cost function is sought. The cost function J corresponds to Equation 11

$$J = \frac{1}{2} \sum_{0}^{n} (x^T Q x + u^T R u). \tag{11}$$

Then, state-feedback matrix is the solution to *Riccati's algebraic equation*

$$A^T S + SA - SBR^{-1} B^T S + Q = 0. \tag{12}$$

Finally, the optimal gain is defined as

$$K = R^{-1} B^T S. \tag{13}$$

Matrices Q and R are usually chosen by Bryson's method [26]. Then, by means of MATLAB's '`lqr`' function as in expression 14, K_{LQR} is obtained.

$$\left[\ K_{LQR},\ S,\ CLP\ \right] = \texttt{lrq}(A,\ B,\ Q,\ R). \tag{14}$$

The LQR control scheme is used for tracking x_{des}, which includes all the states . Besides, a PID controller deals with the altitude control using the same scheme as previously described for the inner loop.

C. SMC

Sliding Mode Control (SMC) [9] is a type of Variable Structure Control (VSC) in which states are constrained on a suitably defined sliding surface. The SMC has good performance, using a limited amount of computational effort. The drawbacks are a high command activity, that leads to a higher energy consumption, and the *chattering* problem [27]. In this work, a First-Order SMC [28] is designed and it is implemented in the faster inner loop, while the outer loop is driven by the same PD position controller described in subsection V-A. The overall control scheme is shown in Figure 13.

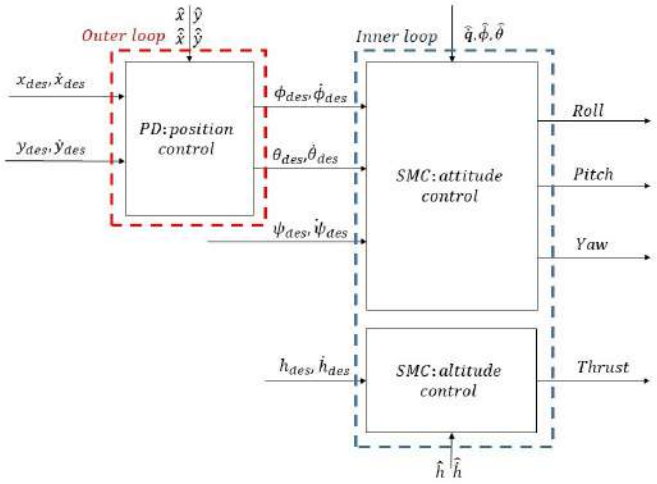


Fig. 13: SMC-based cascade controller scheme

Different sliding surfaces are defined for the variables controlled by SMC. A candidate sliding surface for altitude control is

$$s_{(h)} = \dot{\tilde{h}} - \lambda_{alt}\tilde{h}, \tag{15}$$

where $\tilde{h} = h_{des} - \hat{h}_{NED}$ is the error on the vertical position of the quadrotor. The control law for vertical displacements is

$$F_z = \frac{m}{\cos\theta\cos\phi}(g - \lambda_{alt}\tilde{h}) - Ksign(s_{(h)}), \tag{16}$$

where $\dot{\tilde{h}}$ and $\tilde{h}$ are the errors on the vertical position and velocity in the inertial frame, m the mass of the quadrotor, g the gravitational acceleration and λ_{alt} and K positive parameters to be tuned.

The attitude controller is based on quaternions and Euler angles derivatives since these values are provided by the EKF process in the PX4 Flight Stack. A candidate sliding surface for attitude control is

$$\mathbf{s}_{(\phi,\theta,\psi,\mathfrak{q})} = \begin{bmatrix}\dot{\tilde{\phi}}\\ \dot{\tilde{\theta}}\\ \dot{\tilde{\psi}}\end{bmatrix} + \mathbf{\Lambda}\tilde{\mathbf{q}} = \begin{bmatrix}\dot{\tilde{\phi}}\\ \dot{\tilde{\theta}}\\ \dot{\tilde{\psi}}\end{bmatrix} + \begin{bmatrix}\lambda_{roll} & 0 & 0\\ 0 & \lambda_{pitch} & 0\\ 0 & 0 & \lambda_{yaw}\end{bmatrix}\tilde{\mathbf{q}} \tag{17}$$

and the control law is:

$$\begin{bmatrix}\tau_x\\ \tau_y\\ \tau_z\end{bmatrix} = -\begin{bmatrix}(J_y - J_z)\dot{\psi}\dot{\theta}\\ (J_z - J_x)\dot{\psi}\dot{\phi}\\ (J_x - J_y)\dot{\theta}\dot{\phi}\end{bmatrix} + \mathbf{\Lambda}\begin{bmatrix}J_x & 0 & 0\\ 0 & J_y & 0\\ 0 & 0 & J_z\end{bmatrix}\dot{\tilde{\mathbf{q}}} - \mathbf{K}sign(\mathbf{s}), \tag{18}$$

where $\dot{\tilde{\phi}}$, $\dot{\tilde{\theta}}$ and $\dot{\tilde{\psi}}$ are the derivatives of the error on the Euler angles, $\mathbf{\Lambda}$ and $\mathbf{K}$ positive-definite matrices of the parameters to be tuned and $\tilde{\mathbf{q}}$ is the imaginary part of the quaternion error $\tilde{\mathfrak{q}}$.

VI. Model-in-the-Loop Simulations

In this section, both trajectory planning approaches are tested with the proposed controllers, explained beforehand, within the MATLAB Simulink model of the quadcopter. The control gains are selected for each control system. The Q and R matrices of LQR controller are chosen as two identity matrices. The parameters of the PID controller are listed in Table I, while the SMC parameters are listed in Table II.

Inner Loop		
Channel	Param	Value
Pitch	K_P	6
	K_I	0.05
	K_D	0.3
	$Sat.$	$\pm 3.2Nm$
Roll	K_P	6
	K_I	0.05
	K_D	0.3
	$Sat.$	$\pm 3.2Nm$
Yaw	K_P	0.25
	K_I	0.05
	K_D	0.27
	$Sat.$	$\pm 1Nm$
Thrust	K_P	60
	K_I	70
	K_D	40
	$Sat.$	$32N$

(a) Inner Loop PID parameters

Outer Loop		
Chn	Param	Value
Pos (N)	K_P	0.1
	K_D	0.5
	$Sat.$	$\pm\frac{\pi}{6}rad$
Pos (E)	K_P	0.1
	K_D	0.5
	$Sat.$	$\pm\frac{\pi}{6}rad$

(b) Outer Loop PD parameters

TABLE I: PID parameters

Inner Loop		
Chn	Param	Value
Pitch	λ_θ	20
	k_θ	-0.2
	$Sat.$	$\pm 3.2Nm$
Roll	λ_ϕ	20
	k_ϕ	-0.2
	$Sat.$	$\pm 3.2Nm$
Yaw	λ_ψ	10
	k_ψ	-0.2
	$Sat.$	$\pm 1Nm$
Thrust	λ_T	40
	k_T	2
	$Sat.$	$32N$

(a) Inner Loop

Outer Loop		
Chn	Param	Value
Pos (N)	K_P	0.15
	K_D	0.22
	$Sat.\ \phi, \theta$	$\pm \frac{\pi}{6} rad$
	$Sat.\ \dot{\phi}, \dot{\theta}$	$\pm 1 rad/s$
Pos (E)	K_P	0.15
	K_D	0.22
	$Sat.\ \phi, \theta$	$\pm \frac{\pi}{6} rad$
	$Sat.\ \dot{\phi}, \dot{\theta}$	$\pm 1 rad/s$

(b) Outer Loop PD

TABLE II: SMC parameters

Saturation limits have been imposed to all signals to avoid reaching unfeasible values during simulations.

Fixed-step simulations are performed with an *ode4* solver, and a sample time of 1000 Hz for the quadrotor dynamics. Instead, a fixed-step discrete sample frequency of 125 Hz is considered for all the controller schemes, the EKF and the sensors, in accordance with the PX4 variable refresh rate. Note that, as previously explained, in order to include the limited computational power of the on-board systems, the frequency of all the subsystems is limited. At the beginning of the simulations, the quadrotor is set at zero altitude (on ground), pointing North position.

A. *Infinity-shapped pattern*

This pattern comprises a change in altitude and then a planar trajectory in the shape of an '8'. The resulting patterns and simulated results can be seen in figures 14 and 15:

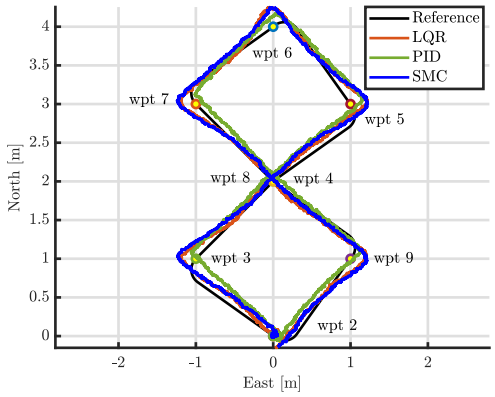


Fig. 14: Dubins planar trajectory simulated with LQR (orange line), PID (green line) and SMC (blue line) controllers

To generate the Dubins-based pattern, a curvature radius of 0.2 m and a stepsize of 1 mm are chosen.

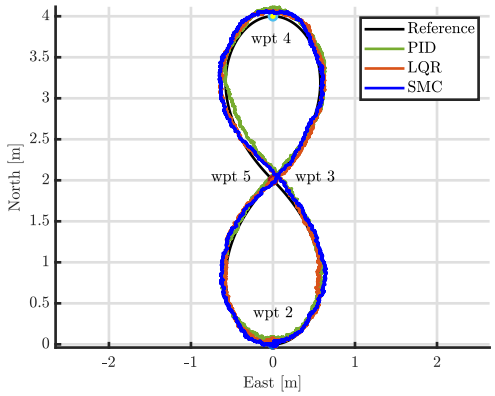


Fig. 15: Bézier planar trajectory simulated with LQR (orange line), PID (green line) and SMC (blue line) controllers

Better results are obtained when the trajectory is generated using Bézier curves. In Figure 14, some overshoot is present in the case of the SMC in the intersection of segments. The LQR has a very similar behaviour, and both controllers exceed the final waypoint while the PID finishes the pattern closer to it. In the Bézier case, the PID shows a greater overshoot than the other two.

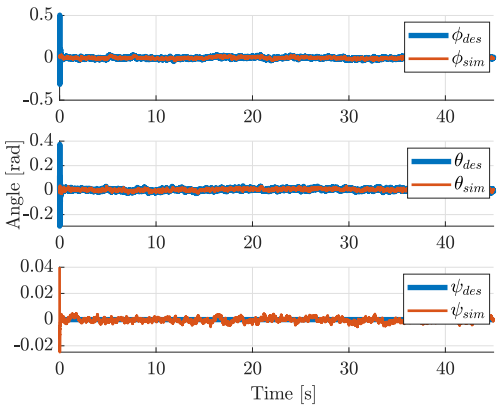


Fig. 16: Bézier Euler angles: a. simulated angles (orange line) and b. reference (blue line) for the SMC controller

In general, the three controllers give good results when following the reference trajectory. However, the SMC presents higher accuracy with respect to the other two solutions. In Figure 16, the attitude response of the model when using the SMC is displayed. No reference in heading is produced for this pattern.

B. *S-shaped pattern with altitude variations*

This pattern contains changes in position, including variations in the altitude halfway through the path, and changes in attitude. Moreover, a reference for the heading is generated. In Figure 17, the position accuracy of the SMC can be appreciated. The final waypoint is reached within the expected time. Unlike with Bézier curves, the quadrotor is not able to fly-by all the waypoints in the trajectory with

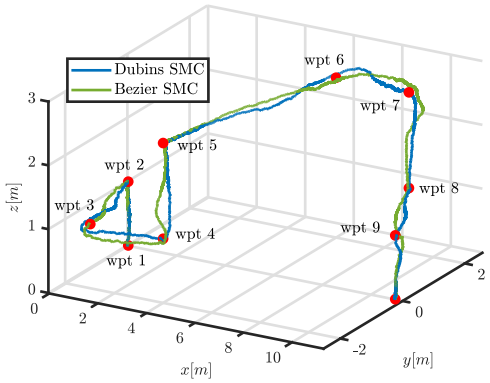


Fig. 17: 3D trajectory simulated with SMC using Bézier (green line) and Dubins (blue line) curves

the Dubins-based trajectory planner. The attitude behaviour obtained for both cases is depicted in Figure 18. Finally, the

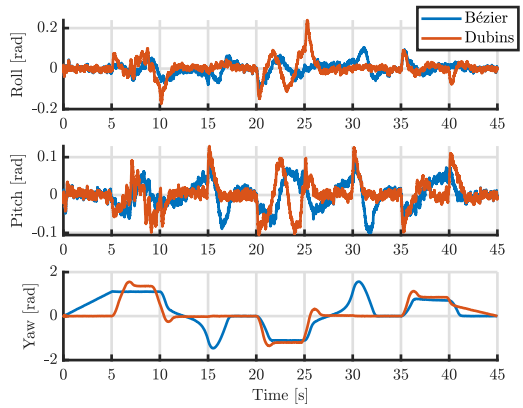


Fig. 18: Euler angles simulated with SMC using Bézier (blue line) and Dubins (orange line) curves trajectory planners

velocity profiles and the results obtained from simulation are presented in figures 19 and 20. For both cases, the vertical velocity reference is identical. The difference is based on the north and east velocity references, where a higher accuracy is achieved with Bézier curves.

VII. Conclusions and Future Work

This paper describes the control and planning approaches considered for a modelled quadrotor in the MATLAB Simulink environment. Moreover, some of the system parameters are experimentally identified, by a model-based approach. The proposed combination of guidance and control algorithms is computational efficient and can be easily translated on a C/C++ code, to be deployed on the on-board autopilot. Inner and outer loop control scheme are considered: (1) the inner loop controls the fast dynamics, guaranteeing robustness and an higher sample time, and (2) the outer loop for the position dynamics. The altitude control loop is separately controlled, in order to obtain very accurate

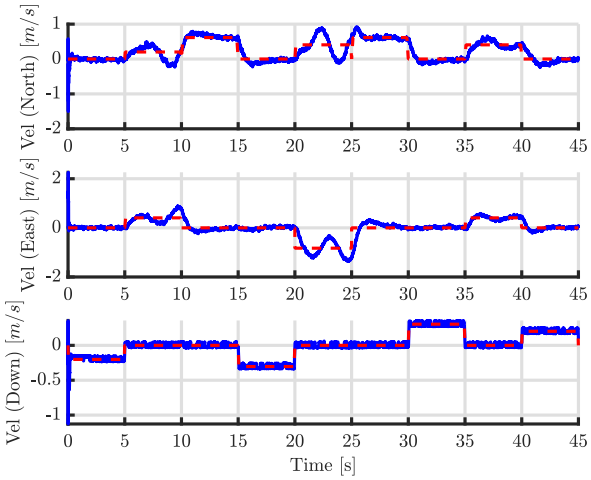


Fig. 19: Velocity profile simulated (blue line) with SMC using Dubins curves references (dashed red line)

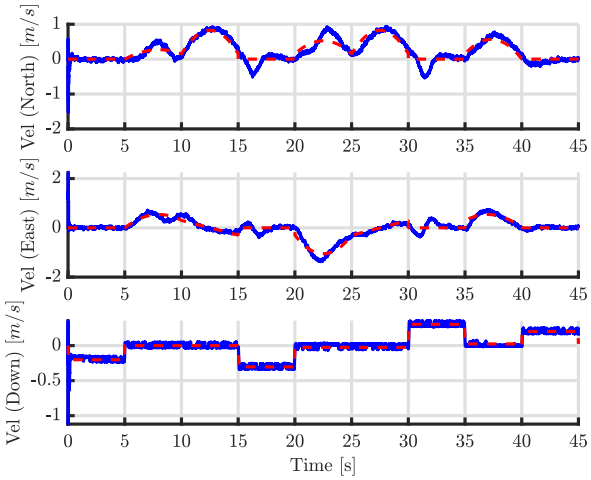


Fig. 20: Velocity profile simulated (blue line) with SMC using Bézier curves (dashed red line)

precision. Extensive simulations are performed to verify the performance of the selected controllers, and the combination with the two approaches for the planner. From the point of view of the obtained performance, the combination for Sliding Mode Controller (SMC) and Bézier trajectory planner show good performance, as depicted by Model-in-the-Loop simulations. We observe that the the reference velocity can strongly affect the performance of the selected on-board software. Note that the main reason of the overshoot on the Dubins-based trajectory planner is due to the tracking of the velocity. In a similar way, tracking a reference heading has an impact on the position accuracy.

As future work, the attitude control system will be improved, introducing disturbances in the definition of the model and the Linear Quadratic Regulator (LQR) will be divided into two separate dynamics. An extension to 3D Dubins and 3D Bézier curves is also a target, including an

higher degree for the Bézier curves.

ACKNOWLEDGEMENT

This research was partially carried out within the framework of the "CREATEFORUAS"project, financially supported by the Italian Ministry for Education, University, and Research, within the PRIN Programme. This work has been partially developed with the contribution of the Politecnico di Torino Interdepartmental Centre for Service Robotics PIC4SeR (https://pic4ser.polito.it).

REFERENCES

[1] A. C. Watts, V. G. Ambrosia, and E. A. Hinkley, "Unmanned aircraft systems in remote sensing and scientific research: Classification and considerations of use," *Remote Sensing*, vol. 4, no. 6, pp. 1671–1692, 2012.

[2] Y. Khosiawan, Y. Park, I. Moon, J. M. Nilakantan, and I. Nielsen, "Task scheduling system for uav operations in indoor environment," *Neural Computing and Applications*, vol. 31, no. 9, pp. 5431–5459, 2019.

[3] R. Luppicini and A. So, "A technoethical review of commercial drone use in the context of governance, ethics, and privacy," *Technology in Society*, vol. 46, pp. 109–119, 2016.

[4] T. Erkkinen and M. Conrad, "Verification, validation, and test with model-based design," SAE Technical Paper, Tech. Rep., 2008.

[5] T. Gindele, S. Brechtel, and R. Dillmann, "A probabilistic model for estimating driver behaviors and vehicle trajectories in traffic environments," in *13th International IEEE Conference on Intelligent Transportation Systems*. IEEE, 2010, pp. 1625–1631.

[6] J.-w. Choi, R. E. Curry, and G. H. Elkaim, "Continuous curvature path generation based on bézier curves for autonomous vehicles." *International Journal of Applied Mathematics*, vol. 40, no. 2, 2010.

[7] P. Wang, Z. Man, Z. Cao, J. Zheng, and Y. Zhao, "Dynamics modelling and linear control of quadcopter," in *2016 International Conference on Advanced Mechatronic Systems (ICAMechS)*, Nov 2016, pp. 498–503.

[8] B. Anjali, A. Vivek, and J. Nandagopal, "Simulation and analysis of integral lqr controller for inner control loop design of a fixed wing micro aerial vehicle (mav)," *Procedia Technology*, vol. 25, pp. 76–83, 2016.

[9] V. I. Utkin, *Sliding modes in optimization and control problems*. Springer Verlag, New York, 1992.

[10] E. Capello, H. Park, B. Tavora, G. Guglieri, and M. Romano, "Modeling and experimental parameter identification of a multicopter via a compound pendulum test rig," *2015 Workshop on Research, Education and Development of Unmanned Aerial Systems, RED-UAS 2015*, pp. 308–317, 2016.

[11] S. Patankar, D. Schinstock, and R. Caplinger, "Application of pendulum method to uav momental ellipsoid estimation," in *6th AIAA Aviation Technology, Integration and Operations Conference (ATIO)*, p. 7820.

[12] "Series 1520 thrust stand." [Online]. Available: https://www.rcbenchmark.com/pages/series-1520

[13] H. Elmali and N. Olgac, "Implementation of sliding mode control with perturbation estimation (smcpe)," *IEEE Transactions on Control Systems Technology*, vol. 4, no. 1, pp. 79–85, Jan 1996.

[14] Y. Li, M. Scanavino, E. Capello, F. Dabbene, G. Guglieri, and A. Vilardi, "A novel distributed architecture for uav indoor navigation," *Transportation research procedia*, vol. 35, pp. 13–22, 2018.

[15] M. Scanavino, A. Vilardi, and G. Guglieri, "An experimental analysis on propeller performance in a climate-controlled facility," *Journal of Intelligent & Robotic Systems*, pp. 1–13.

[16] [Online]. Available: https://px4.io/

[17] B. Etkin and L. Reid, *Dynamics of Flight: Stability and Control*. New York: John Wiley and Sons, 1996.

[18] B. Stevens and F. Lewis, *Aircraft Control and Simulation*. New York: John Wiley and Sons, 2003.

[19] L. Ascorti, "An application of the extended kalman filter to the attitude control of a quadrotor," 2013.

[20] L. E. Dubins, "On Curves of Minimal Length with a Constraint on Average Curvature , and with Prescribed Initial and Terminal Positions and Tangents Author (s): L . E . Dubins Source : American Journal of Mathematics , Vol . 79 , No . 3 (Jul ., 1957), pp . 497-516 Pu," *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[21] H. Prautzsch, W. Boehm, and M. Paluszny, "Bézier- and B-spline techniques," Tech. Rep., 2002.

[22] "Informática gráfica - Pascual González López, Jesús García-Consuegra Bleda - Google Libros."

[23] T. W. Sederberg, "Computer Aided Geometric Design Course Notes," Tech. Rep., 2012. [Online]. Available: https://scholarsarchive.byu.edu/facpubhttps://scholarsarchive.byu.edu/facpub/1

[24] K. Runcharoon and V. Srichatrapimuk, "Sliding mode control of quadrotor," 05 2013, pp. 552–557.

[25] G. Szafranski and R. Czyba, "Different approaches of pid control uav type quadrotor," 2011.

[26] A. Jezierski, J. Mozaryn, and D. Suski, "A comparison of lqr and mpc control algorithms of an inverted pendulum," in *Trends in Advanced Intelligent Control, Optimization and Automation*, W. Mitkowski, J. Kacprzyk, K. Oprzedkiewicz, and P. Skruch, Eds. Cham: Springer International Publishing, 2017, pp. 65–76.

[27] J.-J. E. Slotine, W. Li *et al.*, *Applied nonlinear control*. Prentice hall Englewood Cliffs, NJ, 1991, vol. 199, no. 1.

[28] V. Utkin, "Variable structure systems with sliding modes," *IEEE Transactions on Automatic control*, vol. 22, no. 2, pp. 212–222, 1977.

APPENDIX I
LINEAR MODEL

The matrices corresponding to the linear model of the quadrotor can be seen hereafter:

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \vdots & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ & 0 & -g & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \cdots & & & & & & & \cdots & 0 \end{bmatrix}_{12\times 12} \tag{19}$$

$$B = \begin{bmatrix} 0 & \cdots & & & \cdots & 0 \\ \vdots & & & & & \vdots \\ 0 & \cdots & & & \cdots & 0 \\ \frac{1}{m} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{m} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{m} & 0 & 0 & 0 \\ 0 & 0 & 0 & c_3 & 0 & c_4 \\ 0 & 0 & 0 & 0 & c_7 & 0 \\ 0 & 0 & 0 & c_4 & 0 & c_9 \end{bmatrix}_{12\times 6} \tag{20}$$

Matrices C and D correspond to an identity matrix with the dimensions of A and a zeros matrix with the dimensions of B, respectively.