Assignment 03: Optimal Control

Control of a Quadcopter using Single Shooting Algorithm

Romano Pasquale 198102 pasquale.romano-1@studenti.unitn.it

Zanolli Erik 198852 erik.zanolli@studenti.unitn.it

July 19, 2020

1 Introduction

The aim of the assignment was to write a quadcopter physical model and exploit optimal control Single Shooting technique, in order to generate a trajectory that had to be tracked through a Reactive Control algorithm. The task to be performed was to reach the desired position flying through a window. This goal was chosen after considering our knowledge of both the dynamical system and the optimal algorithms, in fact we found it to be a task simple enough to get in touch with optimal control, but it could also be made harder on will adding extra constraints. The whole work has been subdivided in four steps:

Simulation model create a model to simulate the drone in a realistic environment

Control model create a simplified model in order to apply optimal control

Optimal control apply Single Shooting to find an optimal trajectory

Tracking exploit a reactive control to follow the trajectory

Evaluation test the performances of the control

2 Drone physical models

Two different physical models of the drone have been developed. The first model has been written considering the rotor dynamics, taking also into account viscous friction on the blades and linear drag on the body. Besides, torques on the four rotors are applied as input controls and random wind forces are added as disturbances.

In the second model, used in the optimal control,the rotors inertia have been neglected,thus the rotor torques impact directly the thrust forces.The control inputs are re-defined using the lift force and the torques along x, y and z axis of the drone.

m	0.468 kg	drone mass
l	0.225 m	drone arms length
k	2.98e-6	rotor lift coefficient
b	1.140e-7	blade drag coefficient
I_{rotor}	3.357e-5 $kg \cdot m^2$	rotor inertia
I_{xx}	4.85e-3 $kg \cdot m^2$	x-axis inertia
I_{yy}	4.85e-3 $kg \cdot m^2$	y-axis inertia
I_{zz}	8.802e-3 $kg \cdot m^2$	z-axis inertia
a	0.25	linear drag coefficient

Table 1: Parameters used in the models (inertia are given in moving frame

2.1 Simulation model

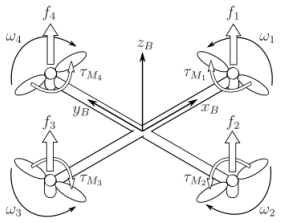


Figure 1: Drone reference frame, torques and angular velocities

The drone reference frame has been defined with respect to absolute reference frame using the three Euler angles: ψ yaw,ϕ roll and θ pitch.

Torques τ_{Mi} and lift forces f_i generated by the rotors are modeled as:

$$\tau_{Mi} = b\omega_i^2 + I_M\dot{\omega}_i$$
$$f_i = k\omega_i^2$$

where: $\omega_i =$ rotors angular velocity

Therefore the Newton equations of motion, in the absolute reference frame are:

$$\ddot{X} = -g + \frac{1}{m}F_t - \frac{1}{m}a\dot{X} + \frac{1}{m}F_w$$

where: X = position in absolute reference frame

g = gravity acceleration

F_t = total thrust in absolute reference frame

F_w = wind force reference frame

a = linear-drag coefficient

The Euler equation in the absolute reference frame are

$$\ddot{\eta} = -\boldsymbol{J}^{-1}(\tau_{\boldsymbol{B}} - \boldsymbol{C}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}})\dot{\eta})$$

where: η = angular position in absolute reference frame

$\boldsymbol{J}$ = jacobian Matrix

$\tau_{\boldsymbol{B}}$ = torques along drone axis

$\boldsymbol{C}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}})$ = coriolis term

2.2 Control model

As mentioned above, in the control model, rotors inertia are neglected and since,in general,a quadcopter can control up to four degrees of freedom,the thrust forces f_i of the rotors have been remapped to the four controlled DoF .

$$\begin{cases} f_z = f_1 + f_2 + f_3 + f_4 \\ \tau_x = l(f_3 - f_1) \\ \tau_y = l(f_4 - f_2) \\ \tau_z = \frac{b}{k}(f_2 + f_4 - f_1 - f_3) \end{cases}$$

where f_z is the total thrust force along z axis and τ_x, τ_y, τ_z are respectively the torques along x,y and z axis in the reference frame of the drone.
The states used in this model are:

$$\bar{\boldsymbol{x}} = [x, y, z, \phi, \theta, \psi, u, v, w, p, q, r] \in \mathbb{R}^{12}$$

where u, v, w are the translational velocities and p, q, r the angular velocities in the drone reference frame. The controls are:

$$\bar{\boldsymbol{u}} = [f_z, \tau_x, \tau_y, \tau_z] \in \mathbb{R}^4$$

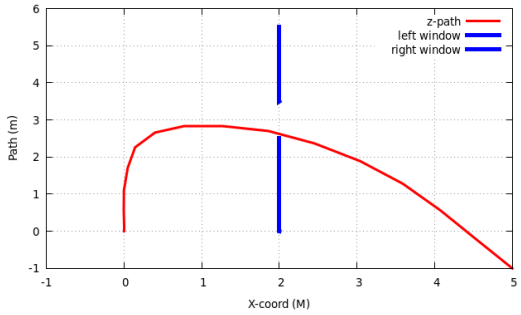
Thus it has been calculated the dynamics of the system obtaining an equation of the form:

$$\dot{\bar{\boldsymbol{x}}} = \boldsymbol{f_D}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}})$$

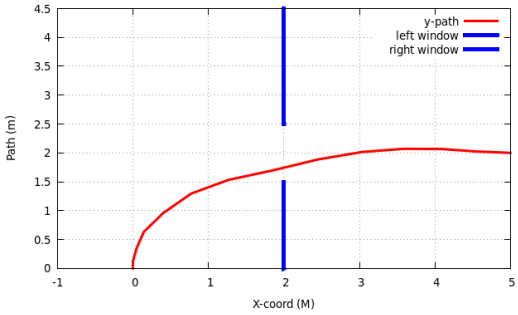
where $f_D(\bar{x}, \bar{u})$ is the non linear function that describes the dynamics of the system. From this function, the partial derivatives $\frac{\partial \boldsymbol{f_D}}{\partial \bar{\boldsymbol{x}}}$ and $\frac{\partial \boldsymbol{f_D}}{\partial \bar{\boldsymbol{u}}}$ are obtained in order to feed them to the optimal control algorithm.

3 Optimal Control

For the choice of optimization algorithm we decided for the single shooting algorithm. The main advantages are the fact that it's easier to implement the cost function and it can manage non-linear constraints:this comes in handy in making the drone fly through the window. We initially thought to implement DDP because, thanks to its gains, tracking optimized trajectory can be more straightforward than using an additional reactive control. Nevertheless,we thought that generating a good trajectory to track without constraints could be more cumbersome than tracking it once found.
Hereafter some of the the optimized trajectories found by the algorithm

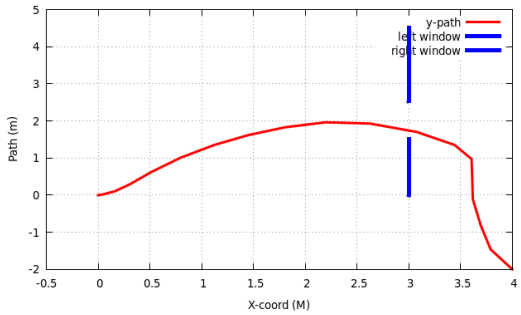


(a) Path along y

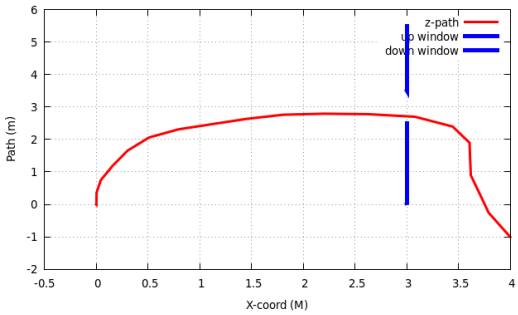


(b) Path along z

Figure 2: Path with 100 iteration with final position [6,-1,2] and window in [2,2,3]

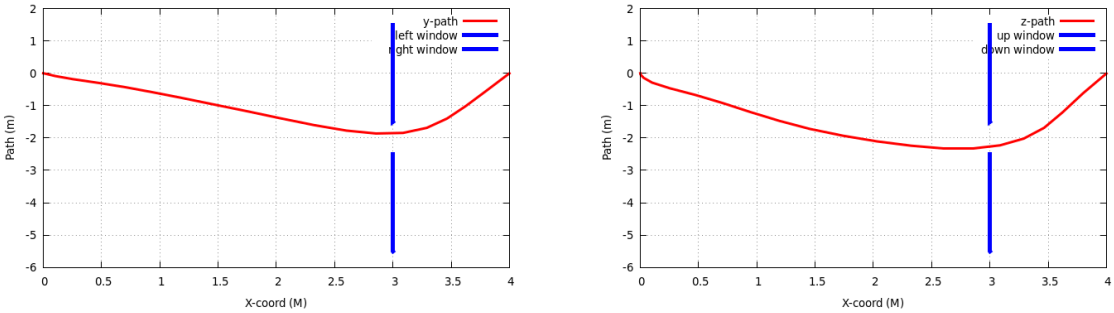


(a) Path along y



(b) Path along z

Figure 3: Path with 100 iteration with final position [6,-2,-1] and window in [3,2,3]



(a) Path along y (b) Path along z

Figure 4: Path with 50 iteration with final position [4,0,0] and window in [3,-2,-2]

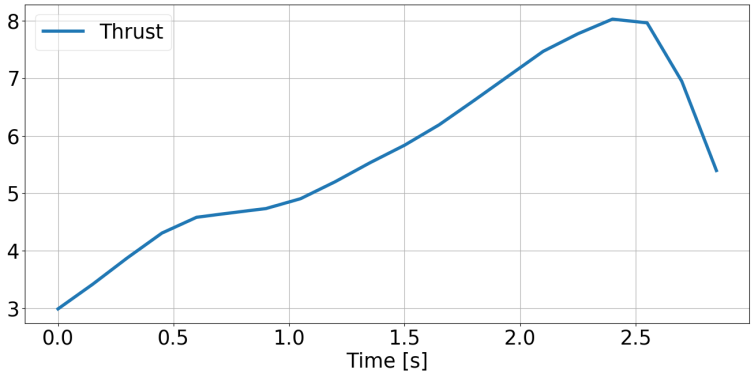


Figure 5: Thrust [N] during simulation Fig.4

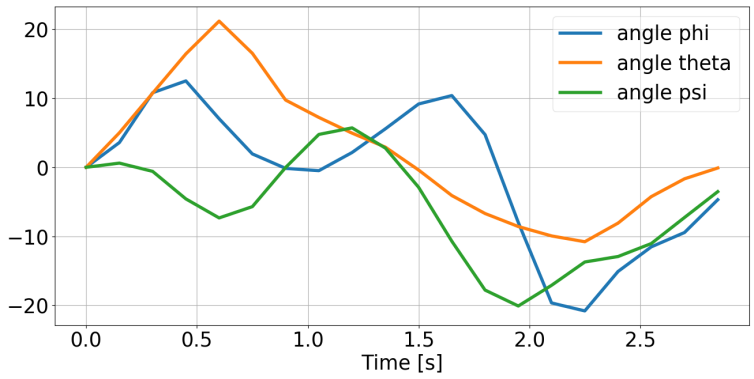


Figure 6: Angle [deg] during simulation Fig.4

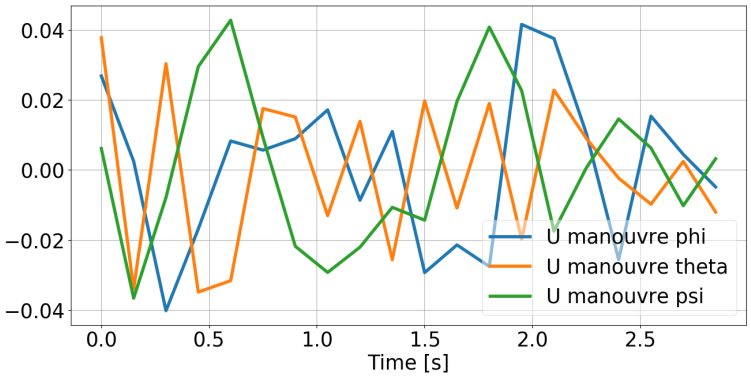


Figure 7: Torques [Nm] along drone axis during simulation Fig.4

The result obtained strongly depend on the initial input guess. As a rule of thumb using as guess the thrust force able to sustain the drone weight improves the results. We also found that the time horizon is also critical, since it has to be adjusted depending on the position of obstacles and desired final position.Moreover, the bounds on the reachable angles are extremely important for the robustness of the algorithm, since the model has a singularity at an angle of 90 degrees. We also added a bound on the maximal thrust and torques, both for realistic results and robustness.

4 Conclusions

Unfortunately making the optimal control algorithm work has been an harder task than we expected, so we didn't manage to implement the tracking algorithm and consequently evaluating its performances. Possible improvement can be make effectively working the calculated Jacobian of constraint for the solver. Then the implementation of TSID on the simulation model and its performance evaluation can give a good feedback on the validity and usefulness of the control model. Another possible improvement could be in the model description, adopting quaternions instead of Euler angles in order to have the capability of doing more complex manoeuvres. Next possible step can be linked to interaction of drones with other object, maybe simulating the carrying of weights.

To see some animations of the results check the figure folder. Github