UNIVERSITÀ DI BOLOGNA

School of Engineering

Master Degree in Automation Engineering

Optimal Control

Optimal Control of an Aircraft

Professor: **Giuseppe Notarstefano**

Students:

Laura Calzoni - 0001058438
Jacopo Merlo Pich - 0001038025
Francesca Paradiso - 0001037825

Academic year 2022/2023

Abstract

Optimal control of an aircraft, exploiting the Newton's algorithm to compute the optimal trajectory.

Considered a simplified, discretized model of an aircraft, Newton's algorithm has been implemented in order to follow both a step trajectory between two equilibrium points and a smoother sigmoidal trajectory between two different altitudes. Furthermore, trajectory tracking of the optimal trajectory computed has been achieved. At the end, also physical limits were taken into account, restricting the absolute values of the pitch moment and thrust force to be respectively *20 Nm* and *40 N* at maximum. We realized an animation to show the behavior of the airplane in all these cases.

Contents

Introduction

Motivations

Practical implementation of algorithms studied during the Optimal Control M course, in Python language.

Contributions

The development of our project has been guided and supervised by the tutor of the course Lorenzo Sforni and Simone Baroncini.

Chapter 1

Mathematical Model

1.1 Dynamics

The nonlinear dynamics for a simplified aircraft model is:

$$\dot{x} = V\cos\gamma \tag{1.1}$$

$$\dot{z} = -V\sin\gamma \tag{1.2}$$

$$\dot{\theta} = q \tag{1.3}$$

$$m\dot{V} = -D(V,\alpha) - mg\sin\gamma + T\cos\alpha \tag{1.4}$$

$$mV\dot{\gamma} = L(V,\alpha) - mg\cos\gamma + T\sin\alpha \tag{1.5}$$

$$J\dot{q} = M \tag{1.6}$$

Whose states variables are:
x: position of the aircraft along the horizontal axis
z: position of the aircraft along the vertical axis
θ: pitch angle
V: velocity of the aircraft
γ: flight path angle between the velocity and the horizontal plane
q: pitch angular velocity

Moreover, α, L(V,α), D(V,α) are respectively the angle of attack, lift force and drag aerodynamical force, defined as:

$$\alpha = \theta - \gamma$$

$$L(V,\alpha) = \frac{1}{2}\rho V^2 S C_{l\alpha}\alpha$$

$$D(V,\alpha) = \frac{1}{2}\rho V^2 S(C_{d0} + C_{d\alpha}\alpha^2)$$

Where:

$C_{d0} = 0.1716$
$C_{d\alpha} = 2.395$
$C_{l\alpha} = 3.256$

The physical parameter of the system are:

$$m = 12 \text{ kg} \quad g = 9.81 \text{ m/s}^2 \quad S = 0.61 \text{ m}^2 \quad \rho = 1.2 \text{ kg/m}^3 \quad J = 0.24 \text{ kg m}^2$$

Finally, T and M are the control inputs, which respectively represents the thrust force and the pitch moment acting on the airfoil.

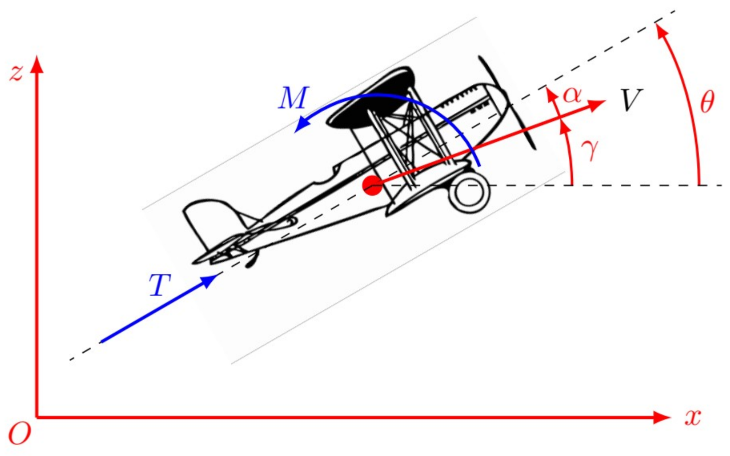


Figure 1.1: *Simplified planar aircraft model*

1.2 Discretization

First of all we discretized the dynamics exploiting forward Euler method, with a time-step *dt* equal to *1e-3*. The function **dynamics()** implements this procedure.

$$x_{t+1} = x_t + dt\,(V_t \cos\gamma_t)$$

$$z_{t+1} = z_t + dt\,(V_t \sin\gamma_t)$$

$$\theta_{t+1} = \theta_t + dt\,(q_t)$$

$$V_{t+1} = V_t + dt\left(-\frac{\rho S}{2m}{V_t}^2\left(C_{d0} + C_{d\alpha}{\alpha_t}^2\right) - g\sin\gamma_t + \frac{T_t}{m}\cos\alpha_t\right)$$

$$\gamma_{t+1} = \gamma_t + dt\left(\frac{\rho S}{2m}V_t C_{l\alpha}\alpha_t - \frac{g}{V_t}\cos\gamma + \frac{T}{mV_t}\sin\alpha_t\right)$$

$$q_{t+1} = q_t + dt\left(\frac{M_t}{J}\right)$$

Chapter 2

Trajectory exploration: gain altitude

The first task was to find the optimal trajectory to move between two equilibrium states at different altitudes. As already mentioned, we exploited the Newton Algorithm to achieved this result.

2.1 Reference curve

We chose as desired curve to follow a step function in z coordinates and a uniform rectilinear motion in x. In particular, we imposed a constant velocity $V_{ref} = 50m/s$, an initial altitude level $z_0 = 5m$ and a final altitude $z_T = 10m$. The angle γ was maintained equal to zero in order to have a straight horizontal motion, as well as the pitch angular velocity. This latter part implies a constant θ, a null M and a null q.

$$x_{ref}(t) = dt * V_{ref} * t$$

$$\begin{cases} z_{ref}(t) = z_0 & \text{for} \qquad t < TT/2 \\ z_{ref}(t) = z_T & \text{for} \qquad t \geq TT/2 \end{cases}$$

$$V_{ref}(t) = V_{ref}$$

$$\gamma_{ref}(t) = 0$$

$$q_{ref}(t) = 0$$

$$M_{ref}(t) = 0$$

The values of the other variables have been found solving the system imposing this pseudo equilibrium condition, i.e. equating (1.4) and (1.5) to zero.

$$\begin{cases} L(V,\alpha) - mg\cos\gamma + T\sin\alpha = 0 \\ -D(V,\alpha) - mg\sin\gamma + T\cos\alpha = 0 \end{cases}$$

This is a non-liner system with two equations in two unknowns. To solve it, we have used the *fsolve* funciton of *SciPy*, that provided the following result:

$$\theta_{ref}(t) = 0.0375 \qquad rad$$

$$T_{eq} = 160.21 \qquad N$$

Since it's often desirable to perform a task using the less input power possible, we decided to set all our desired inputs to zero. As a consequence:

$$T_{ref}(t) = 0$$

The equilibrium value T_{eq} just found, will be used later on for the algorithm initialization.

2.2 Cost Function

In order to find the trajectory that approximates the reference the best, we have to solve an optimization problem and minimize a cost function that has to represent the distance from the desired curve. In our case the cost function $\bar{\ell}\left(\bar{\mathbf{x}}, \bar{\mathbf{u}}\right)$ has a quadratic structure w.r. to the states and the inputs and an additive structure in time.

$$\bar{\ell}\left(\bar{\mathbf{x}}, \bar{\mathbf{u}}\right) = \sum_{t=0}^{T-1} \ell_t\left(\mathbf{x}_t, \mathbf{u}_t\right) + \ell_T\left(\mathbf{x}_T\right)$$

With:

- stage cost

$$\ell_t\left(\mathbf{x}_t, \mathbf{u}_t\right) = \frac{1}{2}\left(\mathbf{x}_t - \mathbf{x}_t^{ref}\right)^T W_x \left(\mathbf{x}_t - \mathbf{x}_t^{ref}\right) + \frac{1}{2}\left(\mathbf{u}_t - \mathbf{u}_t^{ref}\right)^T W_u \left(\mathbf{u}_t - \mathbf{u}_t^{ref}\right)$$

- terminal cost

$$\ell_T\left(\mathbf{x}_T\right) = \frac{1}{2}\left(\mathbf{x}_T - \mathbf{x}_T^{ref}\right)^T W_T \left(\mathbf{x}_T - \mathbf{x}_T^{ref}\right)$$

This objective function is a weighted sum of the norm of the tracking errors, with weight matrices suitably chosen symmetric and positive-semidefinite. We have defined them as diagonal matrices, as follow:

$$Q_t = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 50 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.001 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.001 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.001 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.001 \end{bmatrix} \qquad R_t = \begin{bmatrix} 0.001 & 0 \\ 0 & 0.001 \end{bmatrix}$$

$$Q_T = 5 * Q_t$$

The higher the weights, the more we care to precisely follow the reference of the respective state/input variable. We decided to assign small weights to all the variables but z and x.

In the code, the cost and its components are calculated by the functions **stagecost()**, **termcost()** and **fullcost()**. These also return the cost derivatives that will be part of the discussion later on.

2.3 Newton's method

Newton method is a recursive algorithm for zero finding that can be exploited in optimization problems for finding the zero of the gradient of the cost function, i.e. it's minimum. Practically, the descent direction of the cost can be found minimizing the II order Taylor expansion of the cost function (quadratic approximation of l about x each *k-th* iteration).

2.3.1 Initialization

The algorithm works better and faster if a good guess of the initial trajectory $(\boldsymbol{x^0}, \boldsymbol{u^0})$ is provided. Consequently, we selected as initial condition of our guess (xx_{init}) the value of the reference at *t=0.* Then, the trajectory has been build up as a result of the dynamics, starting from that point and with the control inputs equal to the ones necessary to reach the desired pseudo equilibrium (calculated in section 2.1). More precisely:

- the initial input guess trajectory (at the first iteration) shall coincide with the equilibrium condition

$$\begin{cases} T_{guess}(t) = T_{eq} = 160.21 & \forall t = 0, 1, \dots, T-1 \\ M_{guess}(t) = M_{eq} = 0 & \forall t = 0, 1, \dots, T-1 \end{cases}$$

- $x_0^k = x_0^{des} = xx_{init} \ \ \forall k = 0, 1, \dots, N_{max}$, the starting state point for all k iteration coincides with the starting point of the reference curve,

- $x_{t+1}^0 = f\left(x_t^0, u_t^0\right) \ \ \forall t = 0, 2, \dots, T-1$ the rest of the initial guess state trajectory is obtained integrating the system dynamics in open loop from the starting point and using the initial guess input sequence

2.3.2 Jacobians and Hessians computation

At the beginning of each iteration k, it is expected to evaluate the gradients and the hessians of both the dynamic and cost function, for each instant t of our time span T.

For the objective we have,

- Jacobians

$$\nabla_{x_t}\ell_t\left(x_t, u_t\right) = Q_t\left(x_t - x_t^{des}\right)$$

$$\nabla_{u_t}\ell_t\left(x_t, u_t\right) = R_t\left(u_t - u_t^{des}\right)$$

$$\nabla\ell_T\left(x_t\right) = Q_T\left(x_T - x_T^{des}\right)$$

- Hessians

$$\nabla^2_{x_t x_t}\ell_t\left(x_t, u_t\right) = Q_t$$

$$\nabla^2_{x_t u_t}\ell_t\left(x_t, u_t\right) = 0$$

$$\nabla^2_{u_t u_t}\ell_t\left(x_t, u_t\right) = R_t$$

$$\nabla^2\ell_T\left(x_t\right) = Q_T$$

While, for the dynamics results:

- Jacobians (provided by the function **derivatives()**

$$\nabla_x f_t\left(x_t,u_t\right)=\begin{bmatrix}\nabla_{\mathbf{x}}f_1 & \nabla_{\mathbf{x}}f_2 & \nabla_{\mathbf{x}}f_3 & \nabla_{\mathbf{x}}f_4 & \nabla_{\mathbf{x}}f_5 & \nabla_{\mathbf{x}}f_6\end{bmatrix}$$

$$\nabla_{\mathbf{x}}f_1=\begin{bmatrix}1\\0\\0\\dt\cos\gamma_t\\-dtV_t\sin\gamma_t\\0\end{bmatrix}\qquad \nabla_{\mathbf{x}}f_2=\begin{bmatrix}0\\1\\0\\dt\sin\gamma_t\\dtV_t\cos\gamma_t\\0\end{bmatrix}\qquad \nabla_{\mathbf{x}}f_3=\begin{bmatrix}0\\0\\1\\0\\0\\dt\end{bmatrix}$$

$$\nabla_{\mathbf{x}}f_4=\begin{bmatrix}0\\0\\dt\left(-\frac{\rho SC_{d\alpha}}{m}V_t^2\alpha_t-\frac{T_t}{m}\sin\alpha_t\right)\\1-dt\frac{\rho S}{m}\left(C_{d0}+C_{d\alpha}{\alpha_t}^2\right)V_t\\dt\left(\frac{\rho SC_{d\alpha}}{m}V_t^2\alpha_t-g\cos\gamma_t+\frac{T_t}{m}\sin\alpha_t\right)\\0\end{bmatrix}$$

$$\nabla_{\mathbf{x}}f_5=\begin{bmatrix}0\\0\\dt\left(\frac{\rho SC_{l\alpha}}{2m}V_t+\frac{T_t}{mV_t}\cos\alpha_t\right)\\dt\left(-\frac{\rho SC_{l\alpha}}{2m}\alpha_t+\frac{g}{V_t^2}\cos\gamma_t-\frac{T_t}{mV_t^2}\sin\alpha_t\right)\\1+dt\left(-\frac{\rho SC_{l\alpha}}{2m}V_t+\frac{g}{V_t}\sin\gamma_t-\frac{T_t}{mV_t}\cos\alpha_t\right)\\0\end{bmatrix}\qquad \nabla_{\mathbf{x}}f_6=\begin{bmatrix}0\\0\\0\\0\\0\\1\end{bmatrix}$$

$$\nabla_{\mathbf{u}}f_t=\begin{bmatrix}0&0&0&\frac{dt}{m}\cos\alpha_t&\frac{dt}{mV_t}\sin\alpha_t&0\\0&0&0&0&0&\frac{dt}{J}\end{bmatrix}$$

- Hessians (calculated by the function **hessian()**

The Hessian of our dynamical system is a tensor formed by a collection of layers, each one generated by the 2D Hessian matrix of the corresponding scalar function f_i with respect to the state and the input

Hence, the layers are:

$$Hf_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -dtsin(\gamma_t) & 0 \\ 0 & 0 & 0 & -dtsin(\gamma_t) & -dtV_t sin(\gamma_t) & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$Hf_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & dtcos(\gamma_t) & 0 \\ 0 & 0 & 0 & dtcos(\gamma_t) & -dtV_t sin(\gamma_t) & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$Hf_3 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$Hf_4 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & Hf_4^{3,3} & Hf_4^{3,4} & Hf_4^{3,5} & 0 \\ 0 & 0 & Hf_4^{4,3} & Hf_4^{4,4} & Hf_4^{4,5} & 0 \\ 0 & 0 & Hf_4^{5,3} & Hf_4^{5,4} & Hf_4^{5,5} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

With:

$$
\begin{aligned}
Hf_4^{3,3} &= -\frac{dt}{m}(\rho V_t^2 SC_{d\alpha} - T_t cos(\alpha_t));\\
Hf_4^{3,4} &= -\frac{dt}{m}2\rho V_t SC_{d\alpha}(\alpha_t);\\
Hf_4^{3,5} &= -\frac{dt}{m}(-\rho V_t^2 SC_{d\alpha} + T_t cos(\alpha_t));\\
Hf_4^{4,3} &= -\frac{dt}{m}2\rho SV_t(\alpha_t);\\
Hf_4^{4,4} &= -\frac{dt}{m}\rho S(C_{d0} + C_{d\alpha}(\alpha_t)^2);\\
Hf_4^{4,5} &= \frac{dt}{m}2\rho SV_t C_{d\alpha}(\alpha_t);\\
Hf_4^{5,3} &= -\frac{dt}{m}(\rho V_t^2 SC_{d\alpha} + T_t cos(\alpha_t));\\
Hf_4^{5,4} &= \frac{dt}{m}2\rho V_t SC_{d\alpha}(\alpha_t);\\
Hf_4^{5,5} &= \frac{dt}{m}(-\rho V_t^2 SC_{d\alpha} + mgsin(\gamma_t) - T_t cos(\alpha_t))
\end{aligned}
$$

$$
Hf_5 = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & Hf_5^{3,3} & Hf_5^{3,4} & Hf_5^{3,5} & 0\\
0 & 0 & Hf_5^{4,3} & Hf_5^{4,4} & Hf_5^{4,5} & 0\\
0 & 0 & Hf_5^{5,3} & Hf_5^{5,4} & Hf_5^{5,5} & 0\\
0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
$$

With:

$$
\begin{aligned}
Hf_5^{3,3} &= -\frac{dt}{mV_t}T_t sin(\alpha_t);\\
Hf_5^{3,4} = Hf_5^{4,3} &= \frac{dt}{2m}(\rho SC_{l\alpha} - T_t cos(\alpha_t));\\
Hf_5^{3,5} = Hf_5^{5,3} &= \frac{dt}{mV_t}T_t sin(\alpha_t);\\
Hf_5^{4,4} &= -\frac{2dt}{V_t^3}(gcos(\gamma_t) - \frac{T_t}{m}sin(\alpha_t));\\
Hf_5^{4,5} = Hf_5^{5,4} &= \frac{dt}{m}(-\frac{1}{2}\rho SC_{l\alpha} - mgsin(\gamma_t)V_t^{-2} + T_t cos(\alpha_t)V_t^{-2});\\
Hf_5^{5,5} &= \frac{dt}{mV_t}(gcos(\gamma_t) - sin(\alpha_t))
\end{aligned}
$$

$$Hf_6 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$Hu_1 = Hu_2 = Hu_3 = Hu_4 = Hu_5 = Hu_6 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

The mixed second order derivatives $\nabla^2_{\mathbf{x_t},\mathbf{u_t}} f_t$ have been instead calculated as the gradient with respect to u of the Jacobian $\nabla_{x_t} f_t$. Again, they form a tensor of 6 layers, the i-th one implementing $\nabla^2_{\mathbf{x_t},\mathbf{u_t}} f_{it}$

$$\nabla^2_{\mathbf{x_t},\mathbf{u_t}} f_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \qquad \nabla^2_{\mathbf{x_t},\mathbf{u_t}} f_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\nabla^2_{\mathbf{x_t},\mathbf{u_t}} f_3 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\nabla^2_{\mathbf{x_t},\mathbf{u_t}} f_4 = \begin{pmatrix} 0 & 0 & \frac{dt}{m} sin(\alpha_t) & 0 & -\frac{dt}{m} sin(\alpha_t) & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\nabla^2_{\mathbf{x_t},\mathbf{u_t}} f_5 = \begin{pmatrix} 0 & 0 & \frac{dt}{mV_t} cos(\alpha_t) & \frac{dt}{mV_t^2} sin(\alpha_t) & -\frac{dt}{mV_t} cos(\alpha_t) & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\nabla^2_{\mathbf{x_t},\mathbf{u_t}} f_6 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

All these calculations have been made firstly symbolically, exploiting *Scipy*, and then rewritten replacing inside our parameters values in order to speed-up the code execution.

2.3.3 Second order Newton method

At this point, all the components necessary to construct and solve the affine LQR optimal control problem at each iteration have been calculated. Thus, first of all, the adjoint equation should be solved, backward in time:

$$\lambda_T^k = \nabla \ell_T \left(\mathbf{x}_T^k\right)$$
$$\forall t = T-1, \ldots, 1, 0,$$
$$\lambda_t^k = \nabla_{x_t} f \left(\mathbf{x}_t^k, \mathbf{u}_t^k\right) \lambda_{t+1}^k + \nabla_{x_t} \ell_t \left(\mathbf{x}_t^k, \mathbf{u}_t^k\right)$$

Then $\forall t = 0, 1, \ldots, T-1$ (forward) we computed:

- $\mathbf{a}_t^k = \nabla_{x_t} \ell_t \left(\mathbf{x}_t^k, \mathbf{u}_t^k\right)$
- $\mathbf{b}_t^k = \nabla_{u_t} \ell_t \left(\mathbf{x}_t^k, \mathbf{u}_t^k\right)$
- $\mathbf{q}_T^k = \nabla \ell_t \left(\mathbf{x}_T^k\right)$
- $Q_t^k = \nabla^2_{x_t x_t} \ell_T \left(\mathbf{x}_t^k, \mathbf{u}_t^k\right) + \nabla^2_{x_t x_t} f \left(\mathbf{x}_t^k, \mathbf{u}_t^k\right) \cdot \lambda_{t+1}^k$
- $S_t^k = \nabla^2_{x_t u_t} \ell_T \left(\mathbf{x}_t^k, \mathbf{u}_t^k\right) + \nabla^2_{x_t u_t} f \left(\mathbf{x}_t^k, \mathbf{u}_t^k\right) \cdot \lambda_{t+1}^k$
- $R_t^k = \nabla^2_{u_t u_t} \ell_T \left(\mathbf{x}_t^k, \mathbf{u}_t^k\right) + \nabla^2_{u_t u_t} f \left(\mathbf{x}_t^k, \mathbf{u}_t^k\right) \cdot \lambda_{t+1}^k$
- $Q_T^k = \nabla^2 \ell_T \left(\mathbf{x}_T^k\right)$
- $A_t^k = \nabla_{x_t} f \left(\mathbf{x}_t^k, \mathbf{u}_t^k\right)^T$
- $B_t^k = \nabla_{u_t} f \left(\mathbf{x}_t^k, \mathbf{u}_t^k\right)^T$

For any k iteration the descent direction of our objective function is found as the solution of the following affine LQR optimization problem:

$$\min_{\Delta x_t^k, \Delta u_t^k} \sum_{t=0}^{T-1} \begin{bmatrix} q_t^k \\ r_t^k \end{bmatrix}^T \begin{bmatrix} \Delta x_t^k \\ \Delta u_t^k \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \Delta x_t^k \\ \Delta u_t^k \end{bmatrix}^T \begin{bmatrix} Q_t^k & S_t^{kT} \\ S_t^k & r_t^k \end{bmatrix} \begin{bmatrix} \Delta x_t^k \\ \Delta u_t^k \end{bmatrix}^T +$$

$$+ q_T^{kT} \Delta x_T^k + \frac{1}{2} \Delta x_T^k Q_T^k \Delta x_T^k$$

$$subj.\ to \quad \begin{cases} \Delta x_{t+1}^k = A_t^k \Delta x_t^k + B_t^k \Delta u_t^k & t = 0, 1, \dots, T-1 \\ \Delta x_0 = 0 \end{cases}$$

A solution exists if the necessary conditions of optimality are met:

$$Q_t = Q_t^T \geq 0 \qquad R_t = R_t^T > 0 \qquad Q_t - S_t^T R_t^{-1} S_t \geq 0$$

The part of these matrices that depends on the stage cost does not create problems, since it's something we can define. The addition of the hessians of the dynamics, which are not tunable instead, may make it necessary to carry out a further, subsequent regularization step.

The solution of the aforementioned optimization problem is given by the Riccati equation performed by the augmented state, since in this way it can be reformulated in to a simpler optimal control problem

$$\Delta \tilde{x}_t = \begin{bmatrix} 1 \\ \Delta x_t \end{bmatrix}$$

Start from $p_T^k = q_T$ and $P_T^k = Q_T$, solve backward the difference Riccati equation:

$$p_t = q_t + A_t^T p_{t+1} + A_t^T P_{t+1} c_t - K_t^* \left(R_t + B_t^T P_{t+1} B_t\right) \sigma_t^*$$
$$P_t = Q_t + A_t^T P_{t+1} A_t - K_t^{*T} \left(R_t + B_t^T P_{t+1} B_t\right) K_t^*$$

And then compute for $t \in [T-1,\ T-2, \dots 1, 0]$:

$$K_t = -\left(R_t + B_t^T P_{t+1} B_t\right)^{-1} \left(S_t + B_t^T P_{t+1} A_t\right)$$

$$\sigma_t = -\left(R_t + B_t^T P_{t+1} B_t\right)^{-1} \left(r_t + B_t^T p_{t+1} + B_t^T P_{t+1} c_t\right)$$

The descent direction at the k-th iteration is given by

$$\Delta u_t^k = K_t^k \Delta x_t^k + \sigma_t^k$$

Accordingly the new input sequence is found as

$$u_t^{k+1} = u_t^k + \gamma^k \Delta u_t^k$$

where the stepsize γ is selected based on the Armijo rule. In our code the Δu_t^k is provided by the function **ltv_LQR()**

The corresponding new state trajectory must be computed forward integrating in open loop, $\forall t = 0, 1, \dots, T-1$, with input u_t^{k+1} and initial condition $x_0^{k+1} = x_{init}$

$$x_{t+1}^{k+1} = f(x_t^{k+1}, u_t^{k+1})$$

2.3.4 Hessians regularization

As said before, the optimality conditions cannot be granted in our case, due to the presence of the hessians of the dynamics that may not be semi-positive definite. If this is the case, we need to perform some regularization of the problem. To achieve the regularization we decided to approximate the Newton's method to the first order for the first iterations, computing the weights matrices as:

$$Q_t^k = \nabla^2_{x_t x_t} \ell_T \left(x_t^k, u_t^k \right)$$

$$S_t^k = \nabla^2_{x_t u_t} \ell_T \left(x_t^k, u_t^k \right)$$

$$R_t^k = \nabla^2_{u_t u_t} \ell_T \left(x_t^k, u_t^k \right)$$

In this way, we avoid to add the problematic hessian contributions at the beginning of the algorithm when it is more probable to violate the optimality conditions. The number of iterations during which we perform the regularization strictly depends on the specific task we have to implement.

The introduction of the hessians allows to speed up the convergence of the algorithm.

2.4 Task 1 plots

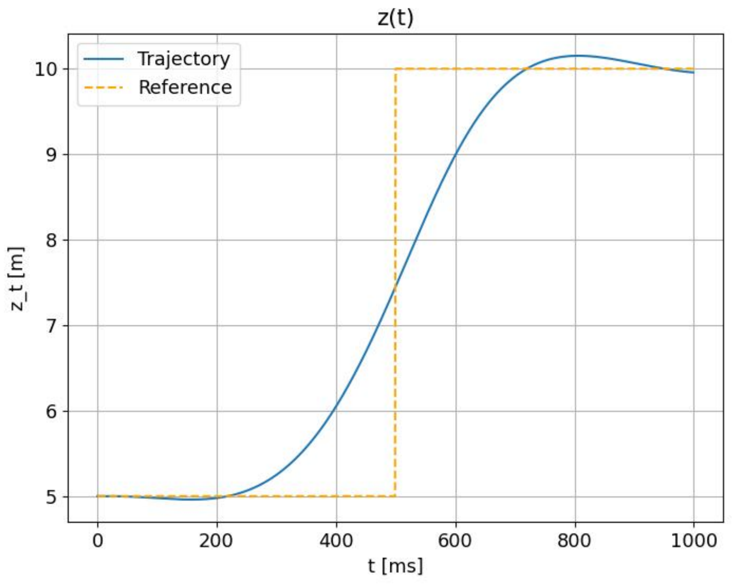


Figure 2.1: *Optimal z(t) computed over a step*

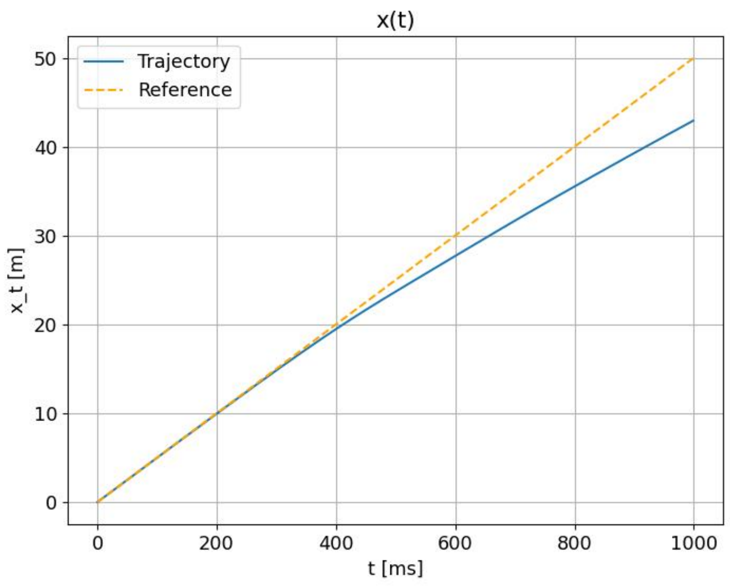


Figure 2.2: *Optimal x(t) computed over a step*

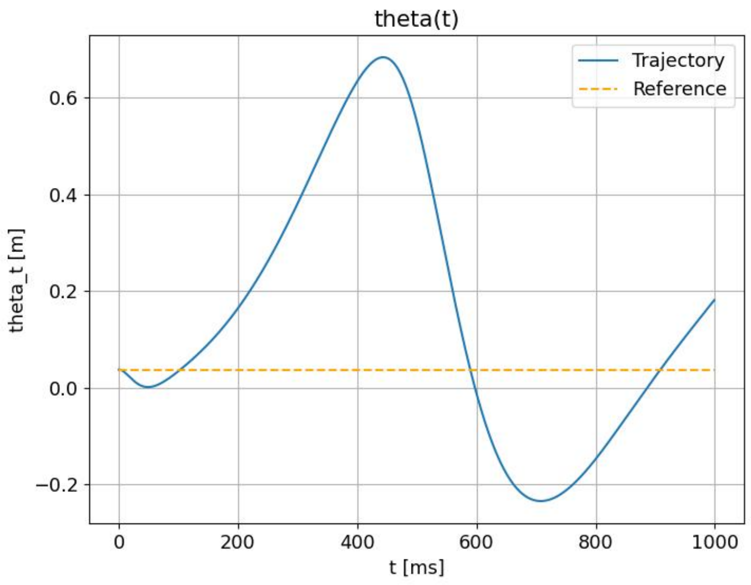


Figure 2.3: *Optimal theta(t) computed over a step*

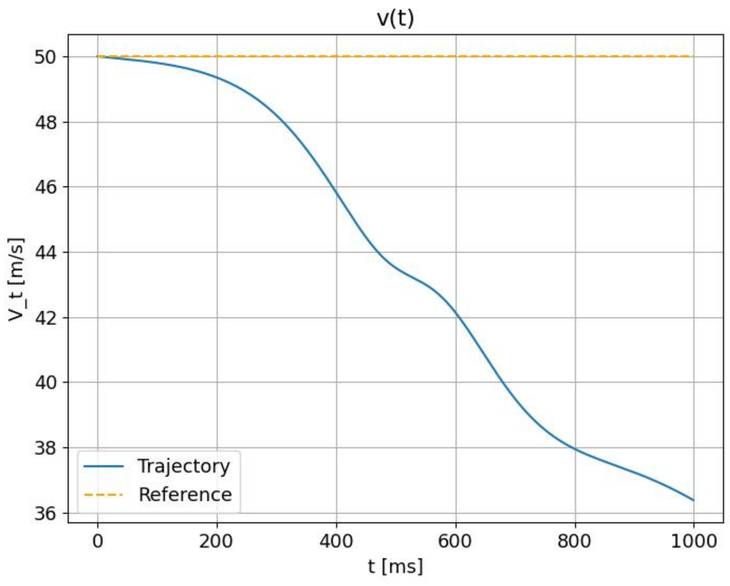


Figure 2.4: *Optimal v(t) computed over a step*

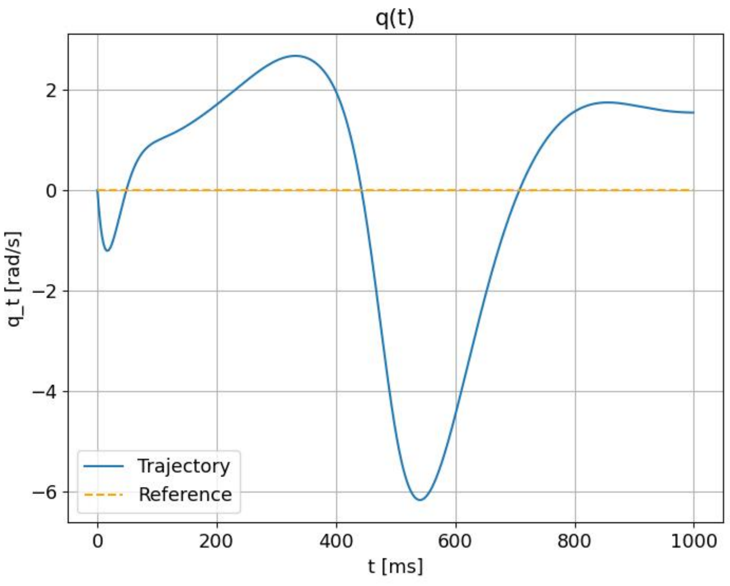


Figure 2.5: *Optimal q(t) computed over a step*

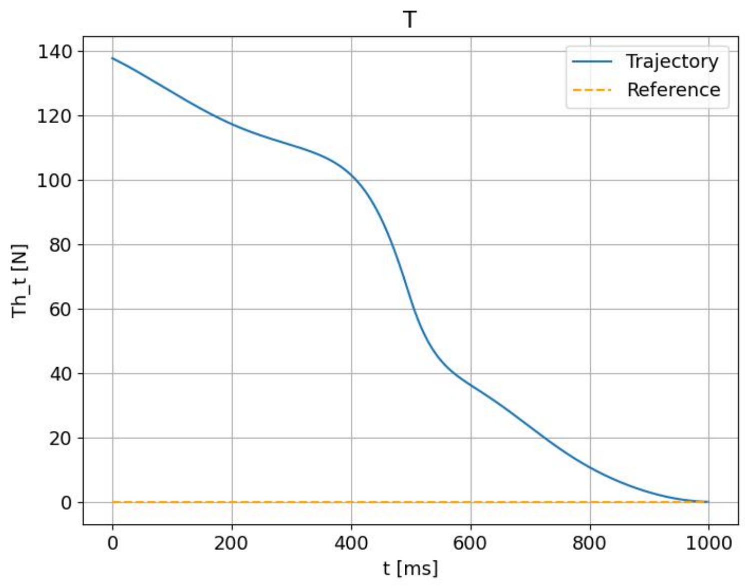


Figure 2.6: *Optimal T(t) computed over a step*

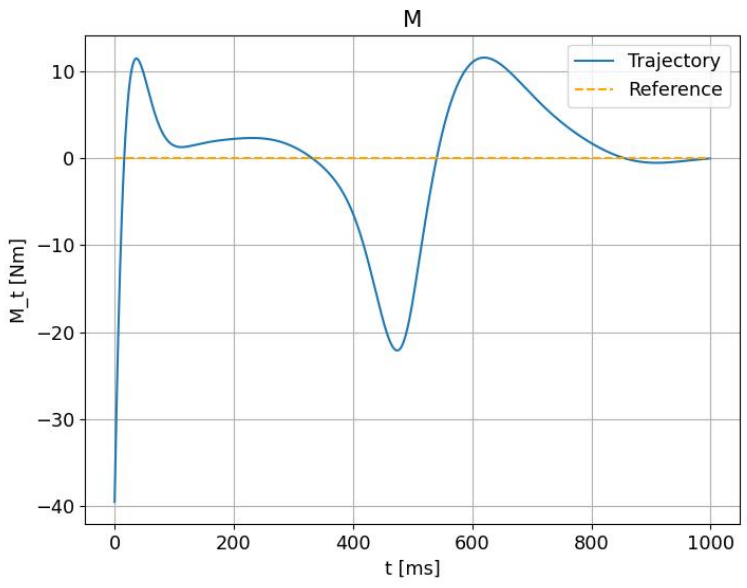


Figure 2.7: *Optimal M(t) computed over a step*

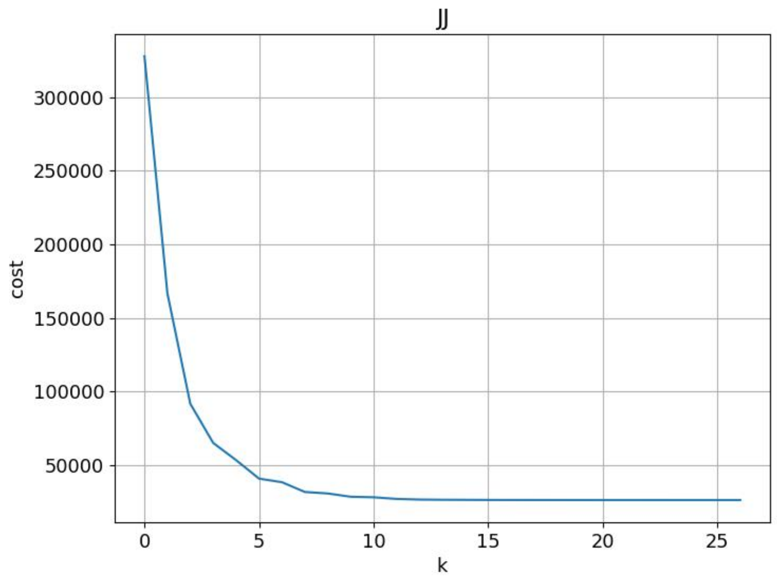


Figure 2.8: *Cost obtained along iterations*

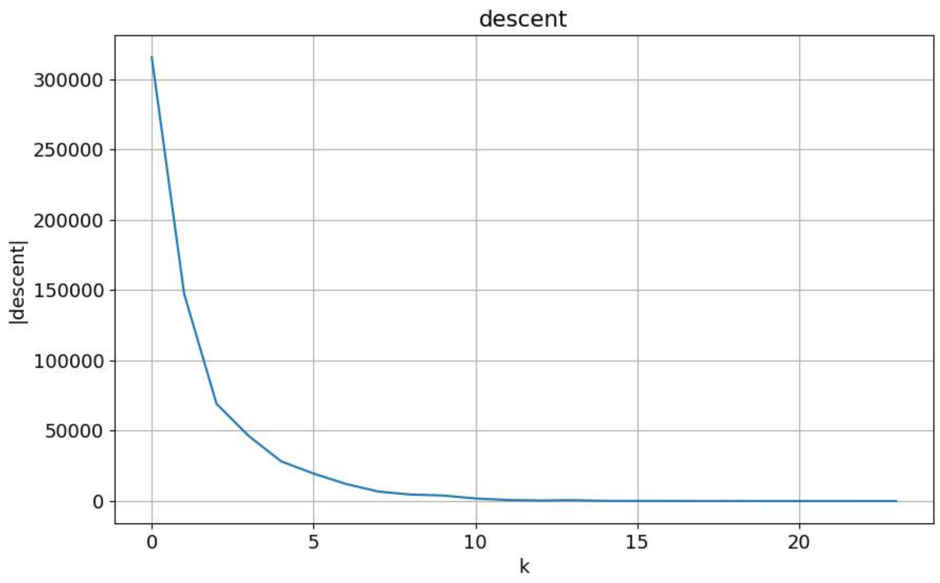


Figure 2.9: *Absolute value of the descent direction obtained along iterations*

We can notice a better chasing of the reference by the states with higher weights. However, z(t) is still very far from its reference since the latter is an unfeasible motion.

Chapter 3

Trajectory optimization: acrobatic flight

As required, we proceeded by defining a smoother reference trajectory to perform a step ahead and again by exploiting the Newton's algorithm for optimal control to compute the new optimal trajectory associated. In the code, we constructed the functions **reference_position()** and **sigmoid_fnc()** for this purpose.
Therefore, this time we chose as desired curve a sigmoid in z coordinates to follow with constant velocity $V_{ref} = 50 m/s$ (same value as before). Like before we wish to perform an ascending motion from *5 m* to *10 m*.

From the dynamics then we obtained (assuming $\dot{\alpha}_t = 0$):

$$\gamma_{ref}(t) = arcsin(\frac{\dot{z}_t}{V_{ref}})$$

$$q_{ref}(t) = \dot{\gamma}_t = \frac{\ddot{z}_t}{V_{ref}\sqrt{1 - (\frac{\dot{z}_t}{V_{ref}})^2}}$$

$$M_{eq}(t) = J(\frac{\dddot{z}_t}{V_{ref}} + \frac{\ddot{z}_t^{\,2}}{V_{ref}} * \frac{\dot{z}_t}{V_{ref}^2} * \frac{1}{1 - (\frac{\dot{z}_t}{V_{ref}})^2})/\sqrt{1 - (\frac{\dot{z}_t}{V_{ref}})^2}$$

Finally, the value of the state variable $\theta_{ref}(t)$ and of the input $T_{eq}(t)$ have been found solving the system with **solver()**:

$$\begin{cases} \frac{1}{2m}(rho(V_t^2)s)C_{l\alpha}\alpha_t - gcos(\gamma_t) + \frac{T_t}{m}sin(\alpha_t) - V_t\dot{\gamma}_t = 0 \\ -\frac{1}{2m}rho(V_t^2)s * (C_{d0} + C_{d\alpha}alpha_t^2)) - gsin(\gamma_t) + \frac{T_t}{m}cos(\alpha_t) = 0 \end{cases}$$

To conclude, since we verified that the angle γ_t remains always small, we approximated the motion along x as a uniform rectilinear

$$x_{ref}(t) = dt * V_{ref} * t$$

Indeed: $x_{t+1} = x_t + dt\,(V_t \cos\gamma_t)$
and for $\gamma_t << 1 \quad \rightarrow \quad \cos\gamma_t \approx 1 \quad \rightarrow \quad x_{t+1} = x_t + dtV_{ref}$

While the reference inputs $M_{ref}(t)$ and $T_{ref}(t)$ were kept to zero, the equilibrium values $M_{eq}(t)$,$T_{eq}(t)$ just calculated in order to obtain the "stationary trajectory" associated to the sigmoid have been used as initial guess for the input.

Once changed the desired behaviours and the initial guess as indicated above, the optimal trajectory can be calculated following the same steps reported in the sections 2.3.2, 2.3.3, 2.3.4 of the previous chapter.

3.1 Task 2 plots

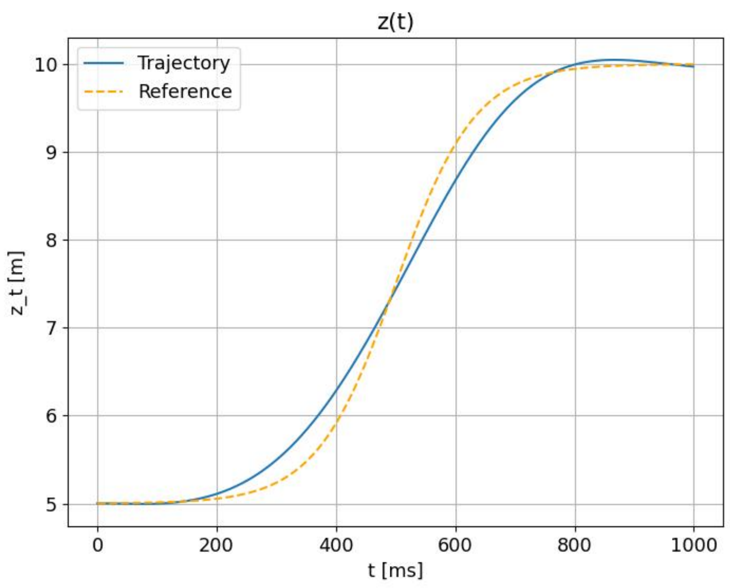


Figure 3.1: *Optimal z(t) computed over a step*

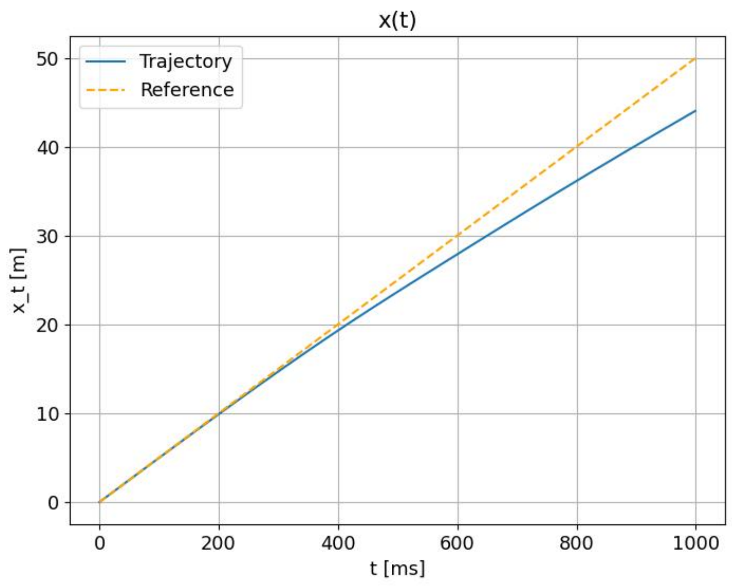


Figure 3.2: *Optimal x(t) computed over a step*

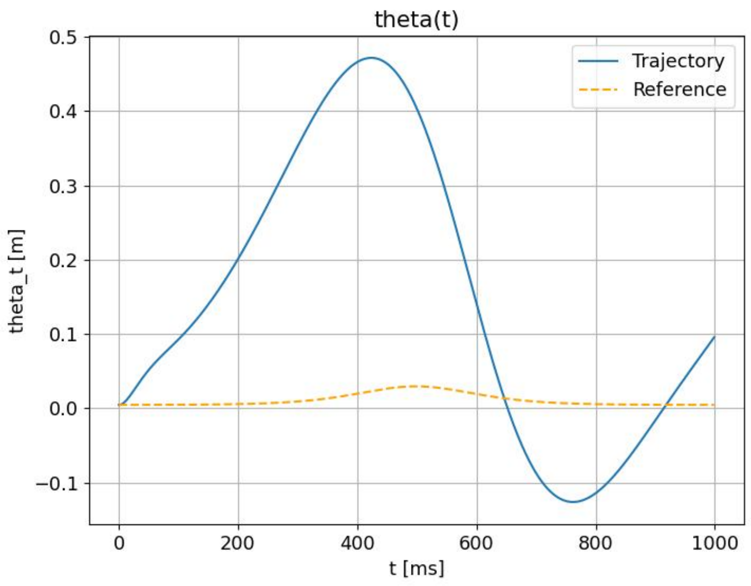


Figure 3.3: *Optimal theta(t) computed over a step*

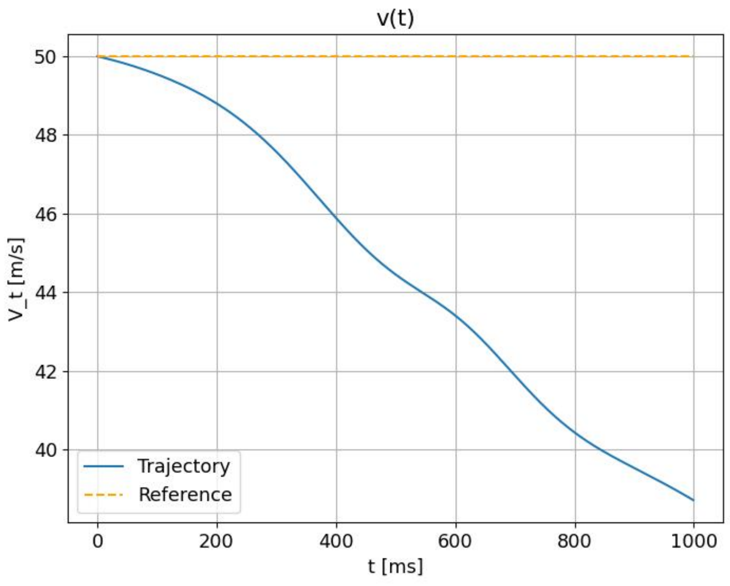


Figure 3.4: *Optimal v(t) computed over a step*

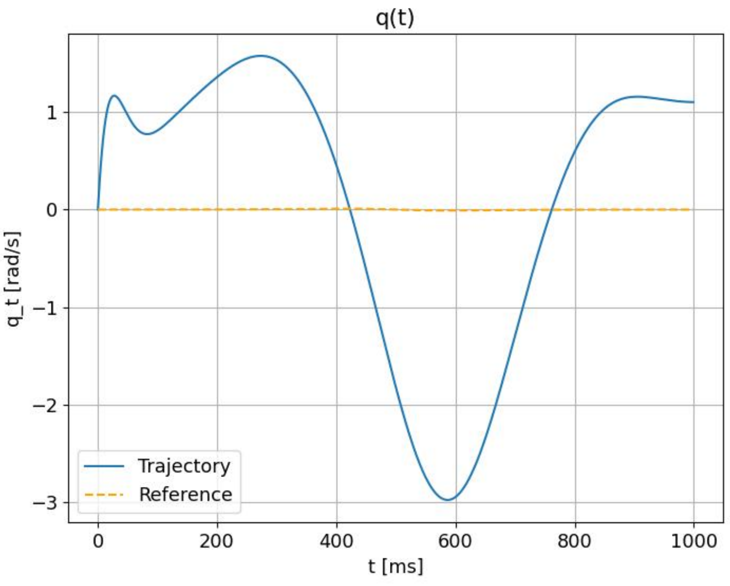


Figure 3.5: *Optimal q(t) computed over a step*

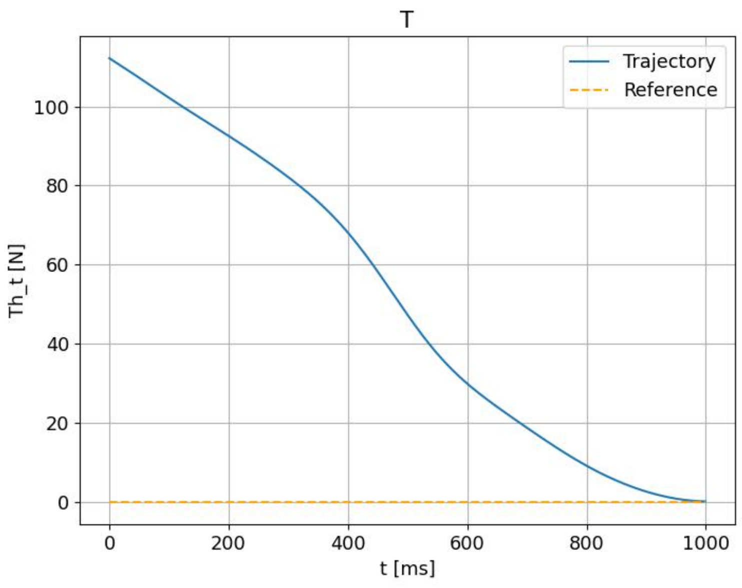


Figure 3.6: *Optimal T(t) computed over a step*

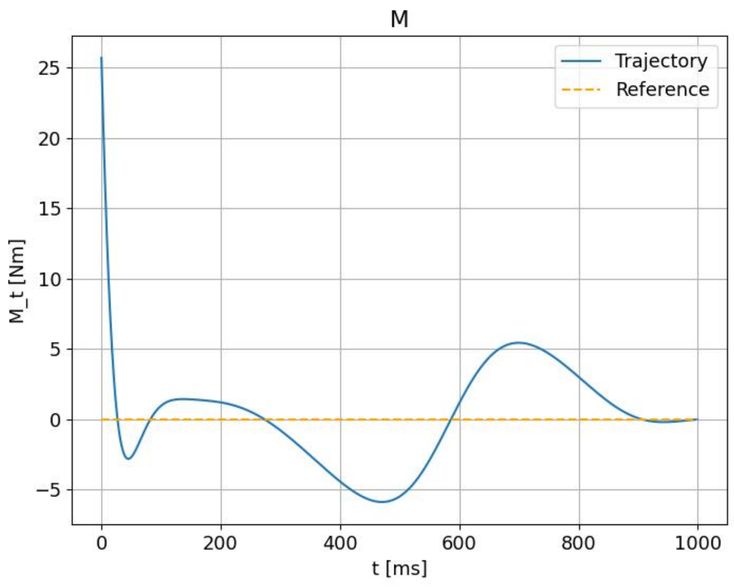


Figure 3.7: *Optimal M(t) computed over a step*

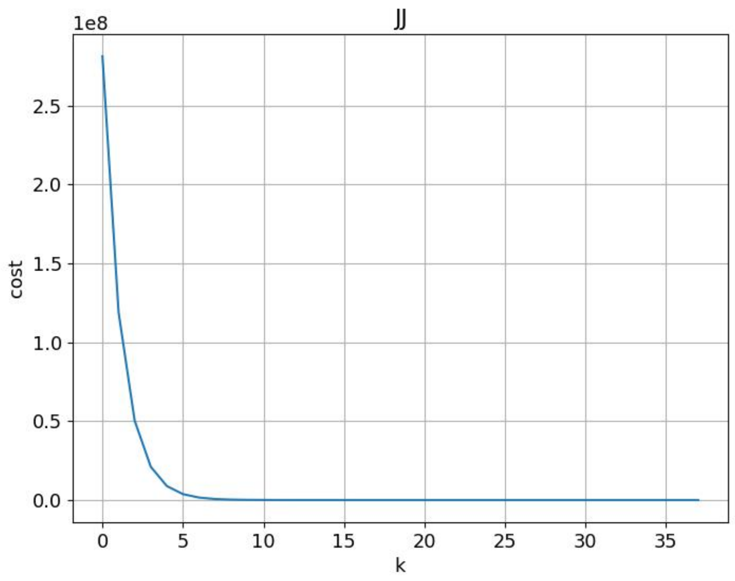


Figure 3.8: *Cost obtained along iterations*

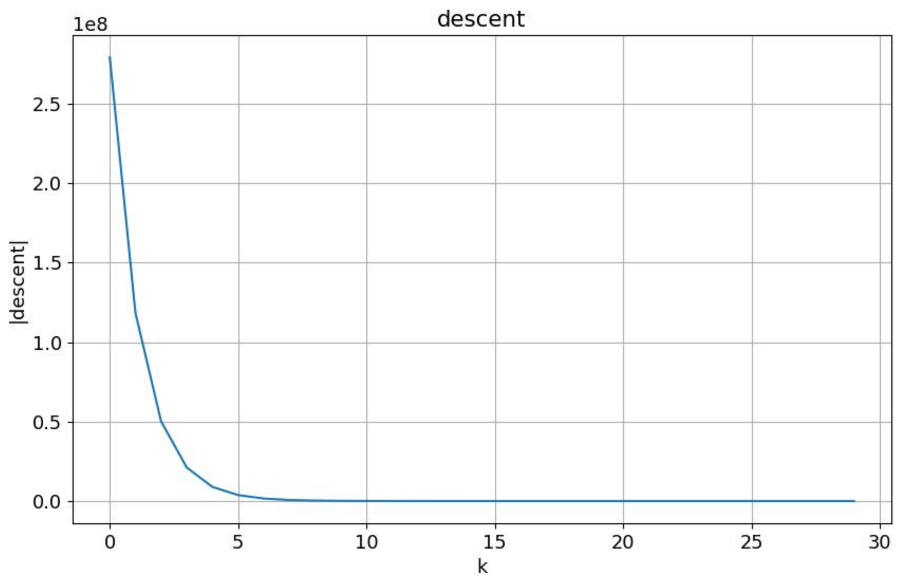
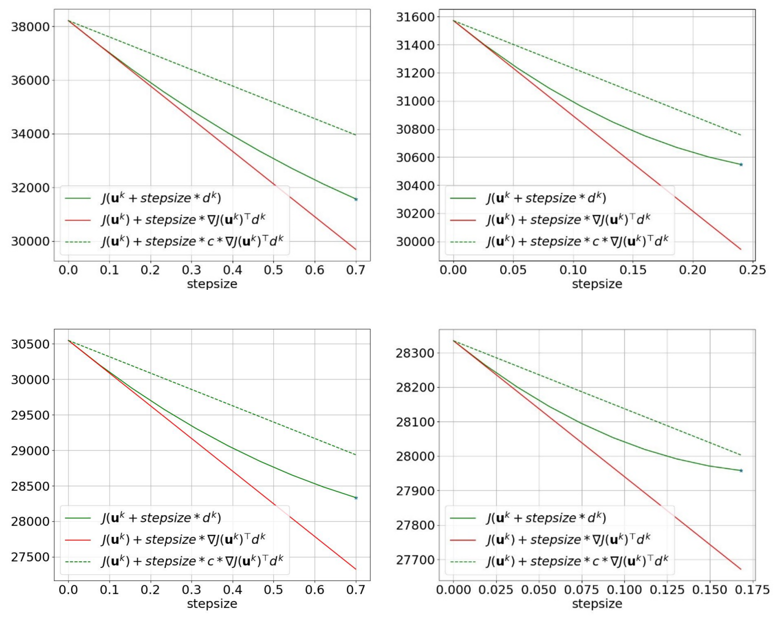


Figure 3.9: *Absolute value of the descent direction obtained along iterations*

As we can see, having a sigmoid reference allows to have a better result in terms of the z state.

This result, as for the previous task, is achieved using Armijo method for the step size selection. Few iterations are reported below:



Chapter 4

Trajectory tracking

For the third task instead it was requested to track the optimal trajectory $(\mathbf{x^{opt}}, \mathbf{u^{opt}})$ solution of optimal control problem in chapter 3.

Consequently, we started approximating the dynamics about $(\mathbf{x^{opt}}, \mathbf{u^{opt}})$, which is feasible by construction, via the linear time-varying system:

$$\Delta x_{t+1} = A_t^{opt} \Delta x_t + B_t^{opt} \Delta u_t$$

where,

$$\begin{aligned} \Delta x_t =& x_t - x_t^{opt} \\ \Delta u_t =& u_t - u_t^{opt} \\ A_t^{opt} =& \nabla_{x_t} f(x_t^{opt}, u_t^{opt})^T \\ B_t^{opt} =& \nabla_{u_t} f(x_t^{opt}, u_t^{opt})^T \end{aligned}$$

for all (x_t^{opt}, u_t^{opt}) with $t =$ *0, ...,* T state-input pairs at time t of trajectory $(\mathbf{x^{opt}}, \mathbf{u^{opt}})$ with lenght T.

Then, in order to calculate the stabilizing feedback controller required to achieve the tracking, we solved the LQR optimal control:

$$\min_{\Delta x_1^k, \ldots \Delta u_{T-1}^k} \frac{1}{2} \sum_{t=0}^{T-1} \Delta x_t^T Q_t^{reg} \Delta x_t + \Delta u_t^T R_t^{reg} \Delta u_t \; + \frac{1}{2} \Delta x_t^T Q_T^{reg} \Delta x_t$$

$$subj.\, to \;\; \Delta x_{t+1}^k = A_t^{opt} \Delta x_t^k + B_t^{opt} \Delta u_t^k \quad for \quad t \in [0, 1, \ldots T-1]$$

The solution gave the optimal sequence of K_t^{reg} feedback gain matrices that must be used to implement the feedback control law:

$$u_t = u_t^{opt} + K_t^{opt} \left(x_t - x_t^{opt} \right)$$

to apply to the original non linear system $\quad x_{t+1} = f(x_t, u_t)$

4.1 Task 3 plots

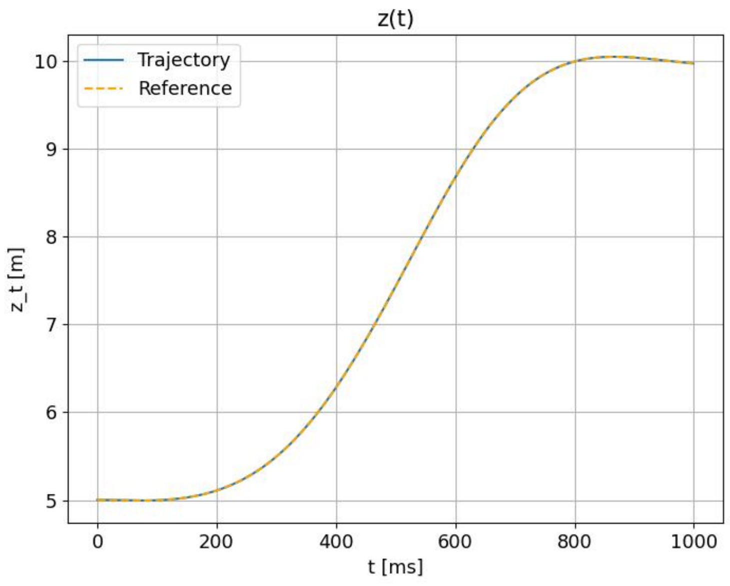


Figure 4.1: *Tracking of optimal z(t)*

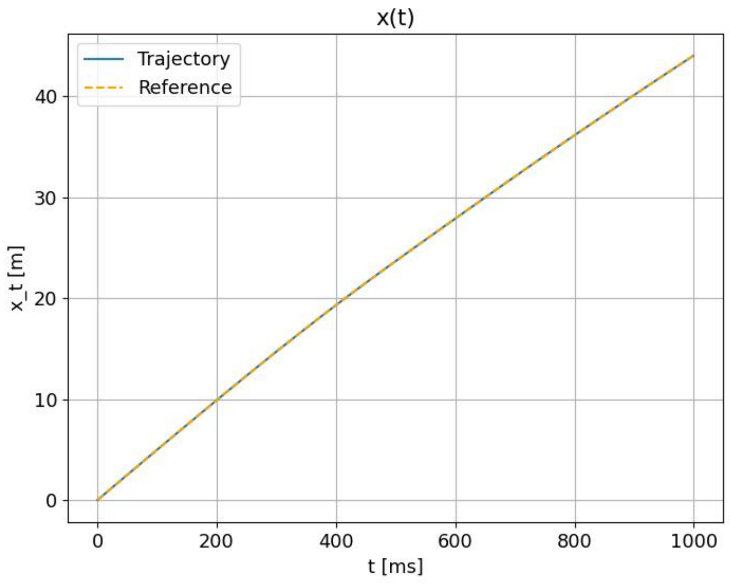


Figure 4.2: *Tracking of optimal x(t)*

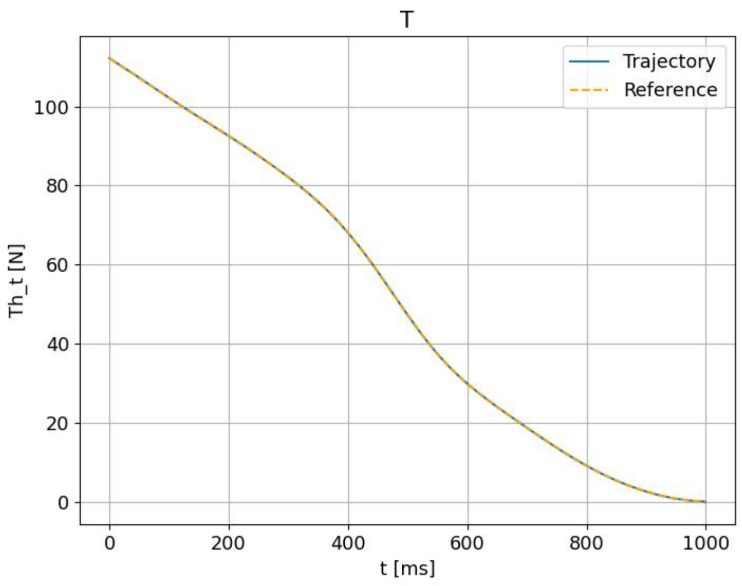


Figure 4.3: *Tracking of optimal T(t)*

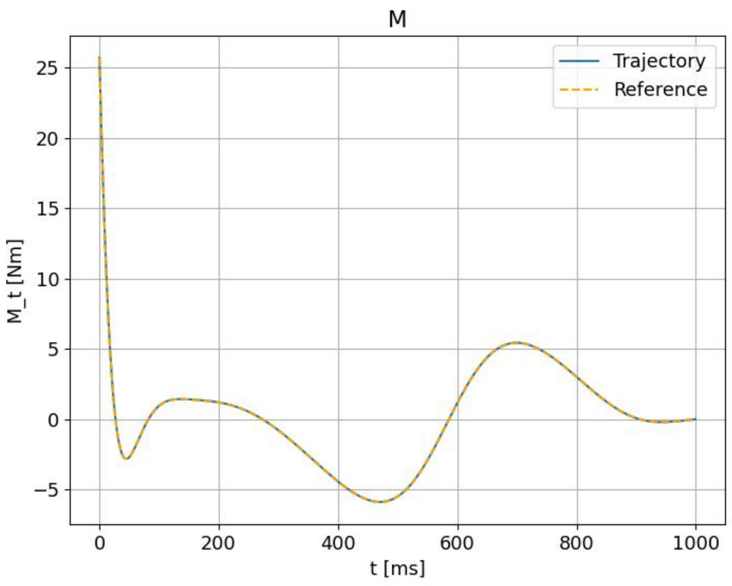


Figure 4.4: *Tracking of optimal M(t)*

Chapter 5

Physical limits

For the last task it was requested to introduce some arbitrary physical limits on the states or inputs. Therefore, we decided to place boundary conditions over both the absolute values of the pitch moment and the thrust force. In particular, from the previous tasks we observed that the maximum values reached by these two control inputs are almost 40 Nm and 60 N respectively. Consequently, we decided to impose as limits 20 Nm for the pitch and 40 N for the thrust.
In order to take into account also these additional constraints in the previous optimal control problems, we implemented the **barrier function method**. Such approach requires to modify the loss function adding a logarithmic term (one for each extra constraint) which has the inequality constraint as the argument and approximates the Indicator function. The optimization problem becomes:

$$min \;\; l(x) = +\epsilon \sum_{j=1}^{r} -log(-g_j(x))$$

with,

$$g_1 : |T| \leq 40$$

$$g_2 : |M| \leq 20$$

The ϵ parameter is initialized at 10^{-1}, then it's shrinked of a factor 10 each five iterations of the Newton's method. In fact, for higher values of ϵ the representation of the inequality constraints in the cost function is less precise, so the bonds result more "relaxed". This condition better suits the first cycles of the optimization problem.
As result, we noticed a relevant decrease in the maximum values of both the inputs and a deterioration of the trajectory.

5.1 Plots with physical limits

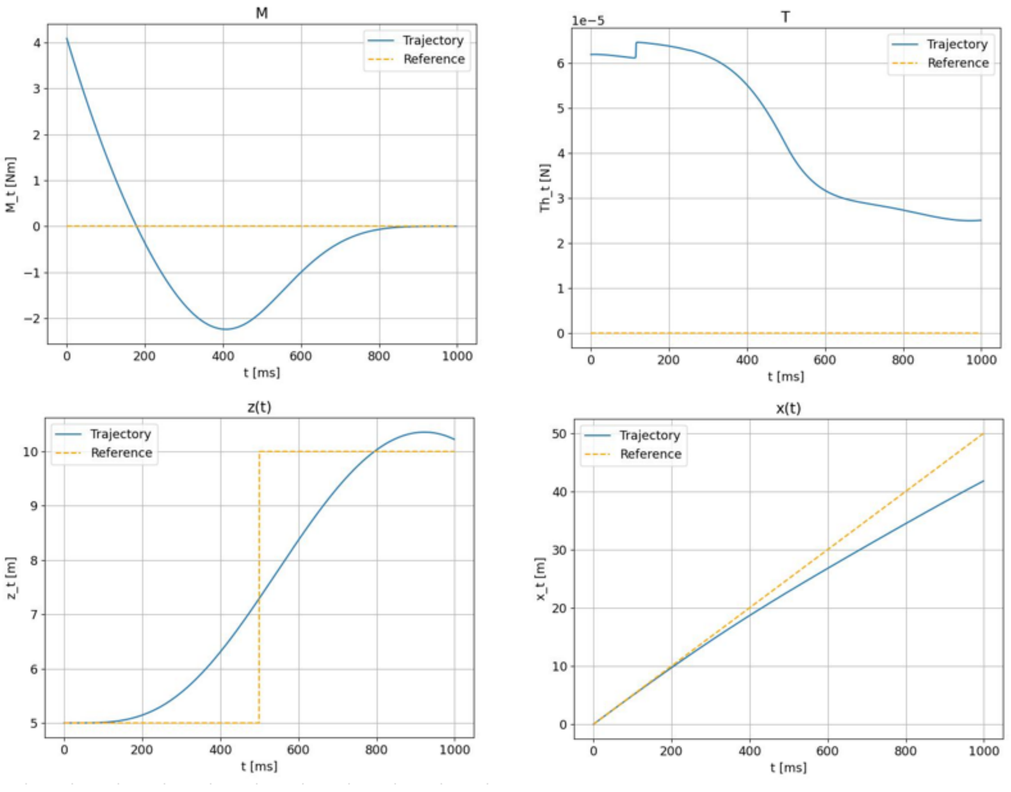


Figure 5.1: *Results for step reference*

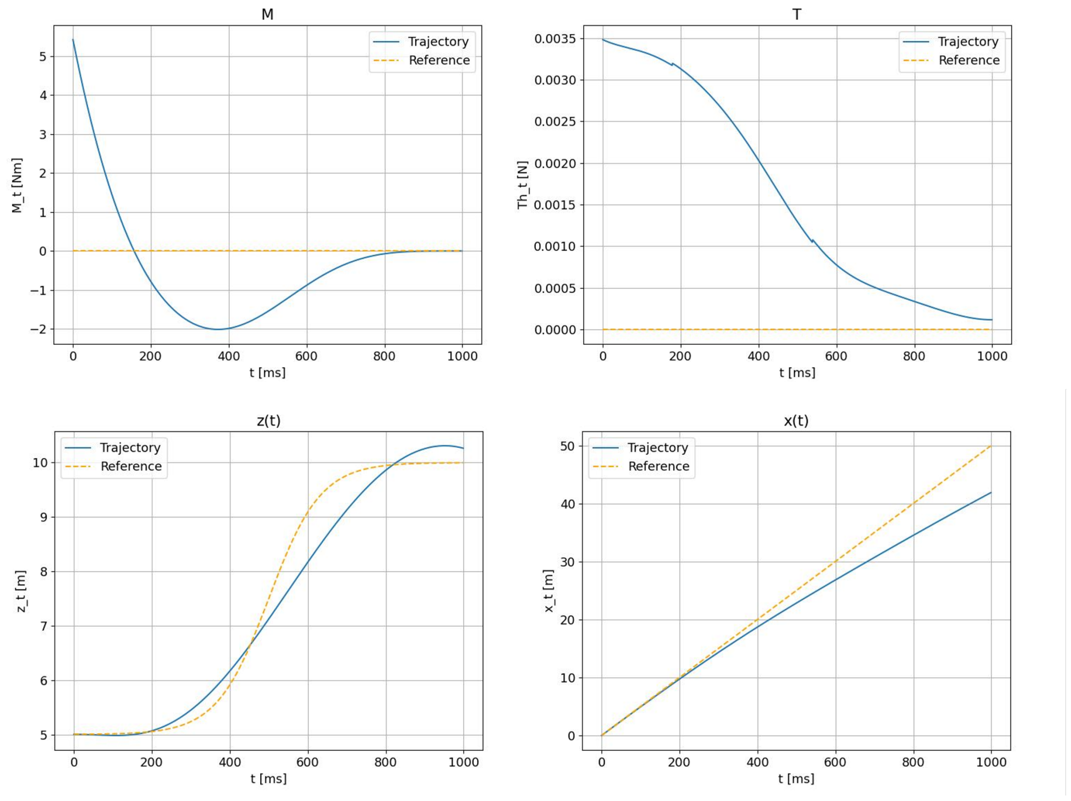


Figure 5.2: *Results for sigmoid reference*

Conclusions

We have achieved the goal of the project, which was to implement the optimal control of an aircraft, exploiting the Newton's algorithm studied during the Optimal Control M course.
The results obtained can be considered satisfying, even if the convergence of the descent direction is achieved after 30 iterations of the algorithm.