UNIVERSITÀ DI BOLOGNA

School of Engineering

Master Degree in Automation Engineering

Optimal Control

Aircraft

Professor: **Giuseppe Notarstefano**

Students:
Mohamed Aboraya
Marco Safwat Anwar Ghaly

Academic year 2022/2023

Abstract

The present document study a strategy of an optimal control algorithm for a simplified model of an aircraft whereas the dynamics present in this model are non linear. So in the light of this, the Newton's algorithm was exploited together with the LQR algorithm in order to find the optimal trajectory and to track it whenever given. The results revealed that by exploiting the Newton's method, we could find an optimal trajectory even for a reference that is not trivial. Furthermore, by using the LQR algorithm we could track the trajectory given with positive outcome even if the initial state is perturbed. Finally we prepared an animation to help visulizing the behaviour of the aircraft.

Contents

Introduction

Motivations

This project aims to develop an optimal control strategy for a simplified aircraft model with non linear dynamics. It is required that the aircraft changes altitude having a reference that in a first temptation is simple and then trying to make it more complicated. To achieve such a behaviour, Newton's algorithm was exploited in order to reach the optimal trajectory having the aircraft's dynamics satisfied. Moreover, the LQR algorithm is used to track the optimal trajectory got from the acrobatic behaviour having perturbation on the initial states. At last, the results are shown using plots and animation.

Task 0 - Problem setup

Definition of the states:

$$x_0 = x \qquad x_1 = z \qquad x_2 = v \qquad x_3 = \theta \qquad x_4 = \dot{\theta} = q \qquad x_5 = \gamma$$
$$u_0 = T \qquad u_1 = M$$

Discretization:

Dynamics:

$$\begin{aligned}
\dot{x} &= V\cos(\gamma) \\
\dot{z} &= V\sin(\gamma) \\
\dot{\theta} &= q \\
m\dot{V} &= -D - mg\sin(\gamma) + T\cos(\alpha) \\
mV\dot{\gamma} &= L - mg\cos(\gamma) + T\sin(\alpha) \\
J\dot{q} &= M
\end{aligned}$$

Forward Euler Method:

$$x_{t+1} = x_t + \delta \dot{x}_t$$

Discretization of the equations:

The dynamics functions after Discretization are as follow

$$\begin{bmatrix} x_{0,t+1} \\ x_{1,t+1} \\ x_{2,t+1} \\ x_{3,t+1} \\ x_{4,t+1} \\ x_{5,t+1} \end{bmatrix} = \begin{bmatrix} x_{0,t} \\ x_{1,t} \\ x_{2,t} \\ x_{3,t} \\ x_{4,t} \\ x_{5,t} \end{bmatrix} + \delta \begin{bmatrix} \dot{x}_{2,t}\cos(x_{5,t}) \\ \dot{x}_{2,t}\sin(x_{5,t}) \\ -\frac{1}{m} * (D_t - mg\sin(x_{5,t}) + u_0\cos(\alpha_t)) \\ x_{4,t} \\ \frac{u_{1,t}}{J} \\ \frac{1}{mx_{2,t}} * (L_t - mg\cos(x_{5,t}) + u_{0,t}\sin(\alpha_t)) \end{bmatrix}$$

where D and L are respectively the lift and drag aerodynamical forces, defined as:

$$\alpha = \theta - \gamma$$
$$D = \frac{1}{2}\rho V^2 S(C_d 0 + C_{d\alpha}\alpha^2)$$
$$L = \frac{1}{2}\rho V^2 S C_{l\alpha}\alpha$$

Gradients:

Remark: We got the first and the second gradients using the Symbolic math toolbox in matlab.

$$\nabla_x = \begin{bmatrix} 1 & 0 & f_{0,2} & 0 & 0 & f_{0,5} \\ 0 & 1 & f_{1,2} & 0 & 0 & f_{1,5} \\ 0 & 0 & f_{2_2} & f_{2,3} & 0 & f_{2,5} \\ 0 & 0 & 0 & 1 & \delta & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & f_{5,2} & f_{5,3} & 0 & f_{5,5} \end{bmatrix}$$

where:

$$f_{0,2} = \delta \cos(x_5)$$
$$f_{0,5} = -\delta x_2 \sin(x_5)$$
$$f_{1,2} = -\delta \sin(x_5)$$
$$f_{1,5} = -\delta x_2 \cos(x_5)$$
$$f_{2,2} = 1 - \frac{(S\delta\rho x_2(C_{d0} + C_{da}(x_3 - x_5^2)))}{m}$$
$$f_{2,3} = -\delta\left(\frac{C_{d\alpha} S\rho x_2^2(2x_3 - 2x_5)}{2} + \frac{u_0 \sin(x_3 - x_5)}{m})\right)$$
$$f_{2,5} = \frac{\delta}{m}\left(\frac{C_{d\alpha} S\rho(2x_3 - 2x_5)x_2^2}{2} + u_0 \sin(x_3 - x_5) - gm\cos(x_5)\right)$$
$$f_{5,2} = \frac{C_{l\alpha} S\delta\rho(x_3 - x_5)}{m} - \frac{\delta}{mx_2^2}\left(\frac{C_{l\alpha} S\rho(x_3 - x_5)x_2^2}{2} + u_0 \sin(x_3 - x_5) - gm\cos(x_5)\right)$$
$$f_{5,3} = \frac{\delta}{mx_2}\left(\frac{C_{l\alpha S\rho x_2^2}}{2} + u_0 \cos(x_3 - x_5)\right)$$
$$f_{5,5} = 1 - \frac{\delta}{mx_2}\left(\frac{C_{l\alpha S\rho x_2^2}}{2} + u_0 \cos(x_3 - x_5) - gm\sin(x_5)\right)$$

$$\nabla_u = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{\delta\cos(x_3-x_5)}{m} & 0 \\ 0 & 0 \\ 0 & \frac{\delta}{J} \\ \frac{\delta\sin(x_3-x_5)}{mx_2} & 0 \end{bmatrix}$$

Task 1 - Trajectory exploration: gain altitude

The trajectory exploration is done such that the simplified model of the aircraft gains an altitude. In other words, the dynamical system performs a transition from an equilibrium point at a certain altitude to another equilibrium point at a higher altitude. These equilibrium points are chosen such that this system is at dynamical balance and as if the aircraft was following a quasi-static trajectory before reaching each one of the two equilibria
The optimization setup is an unconstrained optimal control problem (in discrete-time) as defined in the following,

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^{nT},\mathbf{u}\in\mathbb{R}^{mT}} \quad & \sum_{t=0}^{T-1} \ell_t(x_t,u_t) + \ell_T(x_T) \\ \text{subj.to} \quad & x_{t+1} = f_t(x_t,u_t)) \qquad t = 0,...,T-1 \end{aligned}$$

with given initial condition $x_0 = x_{init} \in \mathbb{R}^{\ltimes}$, where $\ell : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is the so called stage cost while $\ell_T : \mathbb{R}^n \to \mathbb{R}$ is the terminal cost.
Assumptions are that the functions $l_t(.,.), l_T(.), f_t(.,.)$ are of class C^2

And we used the shooting method to make a reduced optimal control problem with the following cost function.

$$\begin{aligned} J(u) &:= \sum_{t=0}^{T-1} \ell(\Phi(\mathbf{u}), u_t) + \ell(\Phi(\mathbf{u})) \\ &= \ell(\Phi(\mathbf{u}, \mathbf{u}) \end{aligned}$$

Thus, the optimal control problem can be recast into the so-called reduced version given by

$$\min_{\mathbf{u}\in\mathbb{R}^{mT}} \quad J(\mathbf{u})$$

which is an unconstrained problem with cost function $J : \mathbb{R}^{mT} \to \mathbb{R}$ where the optimization variable is only the input sequence $\mathbf{u} \in \mathbb{R}^{mT}$.

The Newton's method applied to the reduced optimization problem reads as:

$$\mathbf{u}^{k+1} = \mathbf{u}^k \ - \gamma^k \Delta\mathbf{u}^k$$

where

$$\Delta\mathbf{u}^k = arg \min_{\Delta\mathbf{u}} \nabla J(\mathbf{u}^k)^\top \Delta\mathbf{u} + \frac{1}{2}\Delta\mathbf{u}^\top \nabla^2 \nabla J(\mathbf{u}^k)^\top \nabla\mathbf{u}$$

So the steps used for applying the above results are as follows:

- Step 1: Descent direction calculation

For each iter k we do the following,

Evaluate $\nabla_{xt} f_t(x_t^k, u_t^k)$, $\nabla_{ut} f_t(x_t^k, u_t^k)$, $\nabla_{xt}\ell_t(x_t^k, u_t^k)$, $\nabla_{ut}\ell_t(x_t^k, u_t^k)$, $\nabla \ell_T(x_T^k)$

Solve backwards the co-state equation, with $\lambda_T^k = \nabla \ell_T(x_T^k)$

$$\lambda_t^k = \nabla_{xt} f_t(x_t^k, u_t^k)\lambda_{t+1}^k + \nabla_{xt}\ell_t(x_t^k, u_t^k) \qquad t = T-1, ..., 1$$

Compute for all t=0,...,T-1

$$Q_t^k := \nabla^2_{xtxt}\ell_t(x_t^k, u_t^k) + \nabla^2_{xtxt} f_t(x_t^k, u_t^k).\lambda_{t+1}^k, \qquad R_t^k := \nabla^2_{utut}\ell_t(x_t^k, u_t^k) + \nabla^2_{utut} f_t(x_t^k, u_t^k).\lambda_{t+1}^k$$

$$S_t^k := \nabla^2_{xtut}\ell_t(x_t^k, u_t^k) + \nabla^2_{xtut} f_t(x_t^k, u_t^k).\lambda_{t+1}^k$$

$Q_T^k := \nabla^2_{xTxT}\ell_T(x_T^k)$.

But at the first 8 iterations we used the approximated hessian to compute these matrices to introduce some stability at the beginning of the execution of the algorithm

$$\tilde{Q}_t^k := \nabla^2_{xtxt}\ell_t(x_t^k, u_t^k) \qquad \tilde{R}_t^k := \nabla^2_{utut}\ell_t(x_t^k, u_t^k)$$
$$\tilde{S}_t^k := \nabla^2_{xtut}\ell_t(x_t^k, u_t^k) \qquad \tilde{Q}_T^k := \nabla^2_{xtxt}\ell_T(x_T^k)$$

Lastely, the computation of the descent direction depends on solving the following affine LQR problem.

$$\min_{\substack{\Delta x_1,...,\Delta x_T \\ \Delta u_0,...,\Delta u_{T-1}}} \sum_{t=0}^{T-1} \begin{bmatrix} q_t \\ r_t \end{bmatrix}^\top \begin{bmatrix} \Delta x_t \\ \Delta u_t \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \Delta x_t \\ \Delta u_t \end{bmatrix}^\top \begin{bmatrix} Q_t & S_t^\top \\ S_t & R_t \end{bmatrix} \begin{bmatrix} \Delta x_t \\ \Delta u_t \end{bmatrix}^\top + q_T^\top x_T + \frac{1}{2}\Delta x_T^\top Q_T \Delta x_T$$

$$\text{subj.to} \qquad \Delta x_{t+1} = A_t \Delta x_t + B_t \Delta u_t + c_t \qquad t = 0, ..., T-1$$

where:

- $Q_t \in \mathbb{R}^{n_x \times n_x}$ and $Q_t \in Q_t^\top \geq 0$ for all t=0,...,T
- $R_t \in \mathbb{R}^{n_u \times n_u}$ and $R_t \in R_t^\top$>0 for all t=0,...,T
- $S_t \in \mathbb{R}^{n_u \times n_x}$ such that the problem is convex.

And we solved it by augmenting the state as follows,

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

So we can rewrite the cost and system matrices as

$$\tilde{Q}_t := \begin{bmatrix} 0 & q_t^\top \\ q_t & Q_t \end{bmatrix} \quad \tilde{S}_t := \begin{bmatrix} r_t & S_t \end{bmatrix} \quad \tilde{R}_t := R_t \quad \tilde{A}_t := \begin{bmatrix} 1 & 0 \\ c_t & A_t \end{bmatrix} \quad \tilde{B}_t := \begin{bmatrix} 0 \\ B_t \end{bmatrix}$$

Then, we solve the associated LQR problem

$$\begin{aligned} \min_{\substack{\Delta\tilde{x}_1,\ldots,\Delta\tilde{x}_T \\ \Delta u_0,\ldots,\Delta u_{T-1}}} \quad & \sum_{t=0}^{T-1} \frac{1}{2} \begin{bmatrix} \Delta\tilde{x}_t \\ \Delta u_t \end{bmatrix}^\top \begin{bmatrix} \tilde{Q}_t & \tilde{S}_t^\top \\ \tilde{S}_t & \tilde{R}_t \end{bmatrix} \begin{bmatrix} \Delta\tilde{x}_t \\ \Delta u_t \end{bmatrix}^\top + \frac{1}{2}\Delta\tilde{x}_T^\top Q_T \Delta\tilde{x}_T \\ \text{subj.to} \quad & \Delta\tilde{x}_{t+1} = \tilde{A}_t \Delta x_t + \tilde{B}_t \Delta u_t \qquad t = 0, \ldots, T-1 \\ & \Delta\tilde{x}_0 = x_{init} \end{aligned}$$

Then, the optimal solution of the problem reads

$$\Delta u_t^* = K_t^* \Delta x_t^* + \sigma_t^* \qquad t = 0, \ldots, T-1$$

$$\Delta x_{t+1}^* = A_t \Delta x_t^* + B_t \Delta u_t^* \qquad t = 0, \ldots, T-1$$

where

$$\begin{aligned} K_t^* &= -(R_t + B_t^\top P_{t+1} B_t)^{-1}(S_t + B_t^\top P_{t+1} A_t) \\ \sigma_t^* &= -(R_t + B_t^\top P_{t+1} B_t)^{-1}(r_t + B_t^\top P_{t+1} + B_t^\top P_{t+1} c_t) \\ p_t &= q_t + A_t^\top p_{t+1} + A_t^\top P_{t+1} c_t + K_t^{*\top}(R_t + B_t^\top P_{t+1} B_t)\sigma_t^* \\ P_t &= Q_t + A_t^\top P_{t+1} A_t - K_t^{*\top}(R_t + B_t^\top P_{t+1} B_t) K_t^* \end{aligned}$$

with $p_T = q_T$ and $P_T = Q_T$

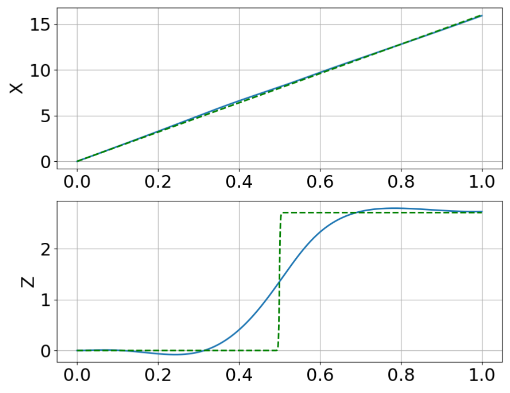
- Step 2: the update of the input sequence becomes as follows,

$$u_t^{k+1} = u_t^k + \gamma^k (K_t^k (x_t^{k+1} - x_t^k) + \sigma_t^k)$$

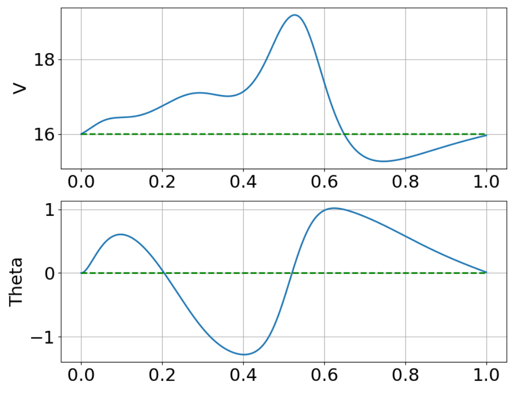
- Step 3: Then, the update of the new state trajectory depends on forward integration (Open-loop)

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

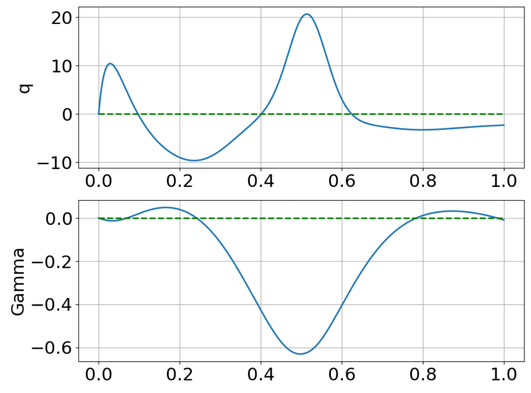
The results are shown in Figure 1.



(a) x z states



(b) v θ states



(c) q γ states

Figure 1: States of task 1

In the above figure it can be seen in figure 1a that both states X, and Z are followed in a very smooth and natural way without spikes, even though the reference trajectory in the Z-direction is a very sharp step function. The X-direction is followed exactly as it's simply a linear function in time and also because the final distance in X-direction allows to make this smooth behavior in the Z-direction.

In figure 1b The pitch angle at the start and end point is zeros as commanded by the reference trajectory. But, in between it crosses the zero line twice which shows that control input is trying to use aerodynamic forces in the acceleration and the deceleration to change the altitude in the time span requested by the reference trajectory. It can also be noticed that the velocity decreases after time 0.6 seconds to decrease the lift force on the aircraft and make it go lower after a small overshoot in the z-direction.

In figure 1c demonstrates the plot of both the angular velocity of the aircraft and the flight path angle. The effect of the angular velocity switching between positive and negative values can be seen later in the aircraft angle. And it is saturated on a certain value after the time 0.6 seconds to make the angle of the aircraft reach the final point on time in a linear fashion. On the plot for the flight path angle Γ, there is a negative peak in the middle which corresponds to the switching between altitudes in the reference trajectory for the z-direction.

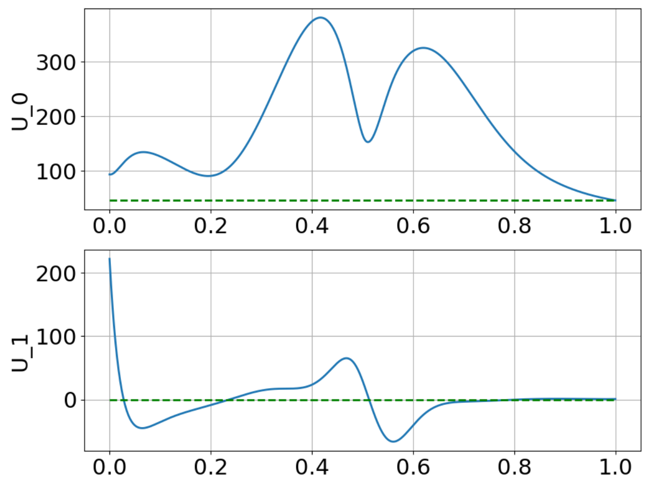
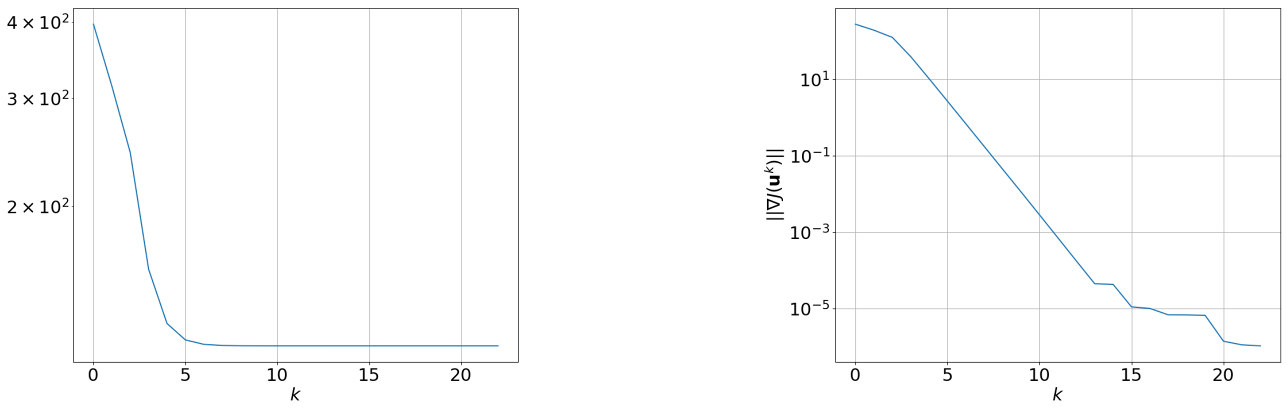


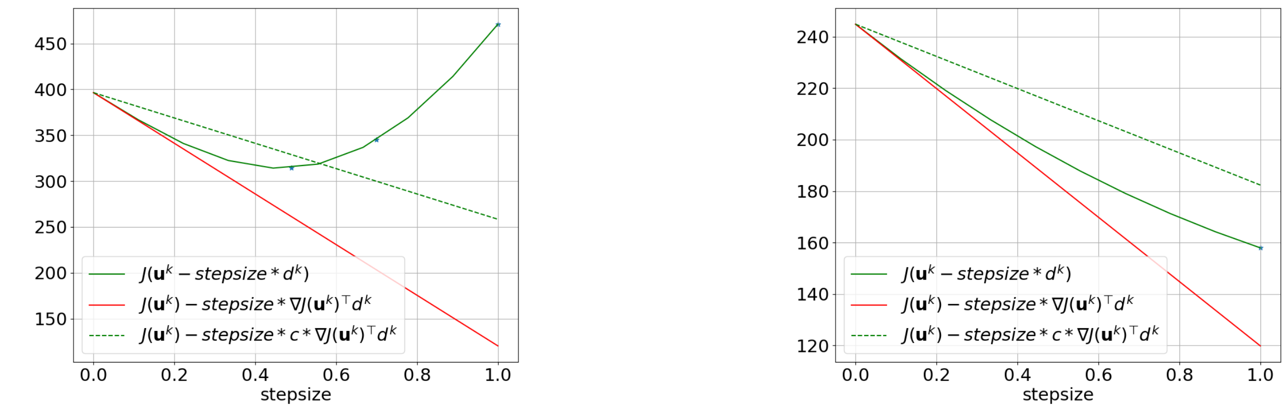
Figure 2: Inputs of task 1

The input for task 1 that is demonstrated in 2 shows that very high moment u_1 at the beginning of the simulation but it's impulse decay fast and starts to oscillate, this indicates that the starting point for this trajectory is not the best choice for starting to follow this trajectory. And also it can be noticed that the moment starts to saturates and changes a little as we look toward the end of the trajectory, because the dynamics is well-suited to reach the final point. And as for the thrust u_0 it increases or decreases slowly as we head toward or away from the center, this the effect of acceleration or deceleration ,respectively, in the z-direction, And it increases in the middle to gain the required altitude.



(a) cost function (b) descent direction

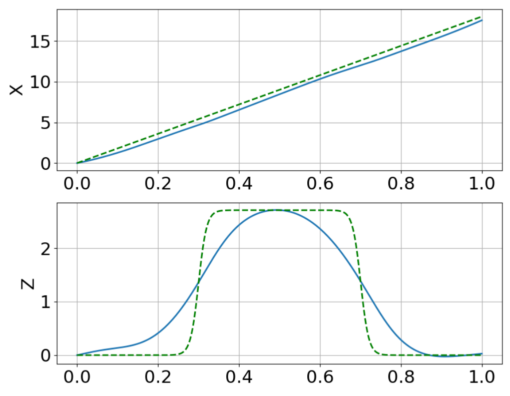
Figure 3: cost and descent of task 1



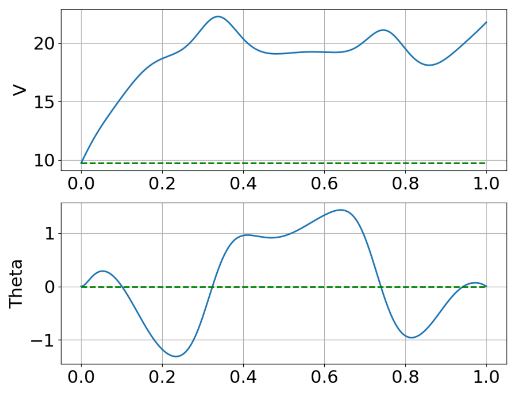
(a) armijo iter 1 (b) armijio iter 3

Figure 4: Armijo

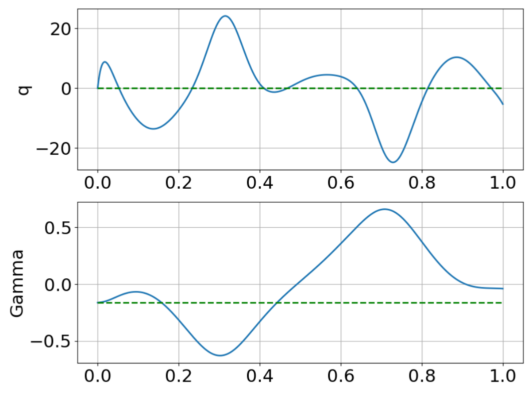
Task 2 - Trajectory optimization: acrobatic flight



(a) x z states



(b) v θ states



(c) q γ states

Figure 5: States of task 2

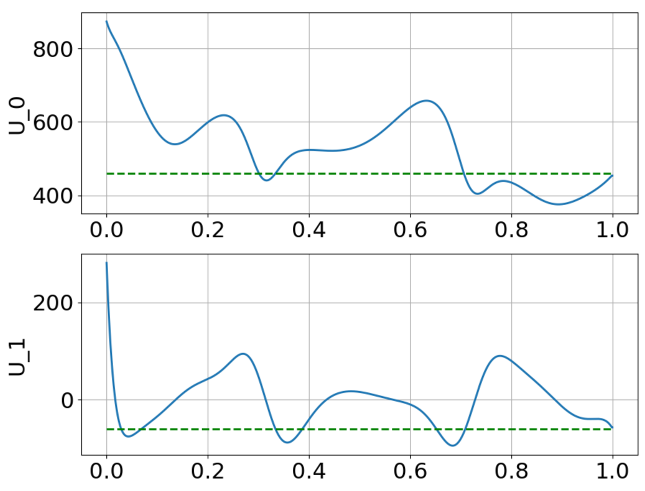
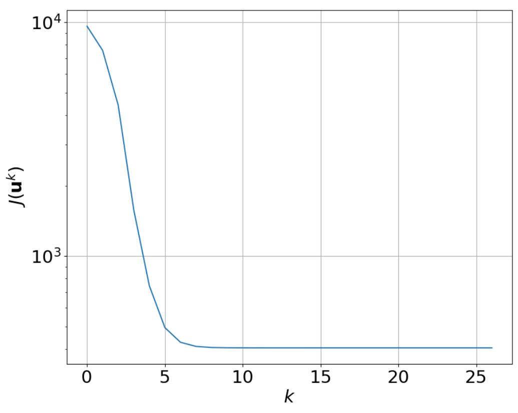
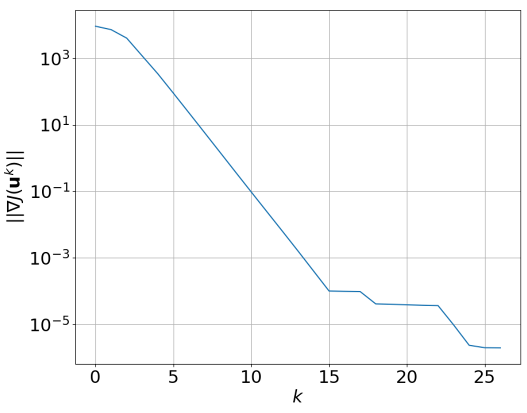


Figure 6: Inputs of task 2



(a) cost function



(b) descent direction

Figure 7: cost and descent of task 2

The figures for the 7, show similar behaviour to that of the task 1 but with more increasing or decreasing peaks, that corresponding to the more involved geometric reference trajectory, which requires the maneuver that is demonstrated in the plot for the z-direction.

Remark: Task 1 and Task 2 are mainely based on the course handouts available at the following link 1.

Task 3 - Trajectory tracking

In this task we will linearize the system about the optimal trajectory obtained form tak 2 and, by exploiting the LQR algorithm, an optimal feedback controller will be defined in order to track this trajectory.

As a matter of fact we need to solve the following LQ problem:

$$\begin{aligned}
&\min_{\substack{\Delta x_1,\dots,\Delta x_T\\ \Delta u_0,\dots,\Delta u_{T-1}}} && \sum_{t=0}^{T-1} \Delta x_t{}^\top Q_t \Delta x_t + \Delta u_t{}^\top R_t \Delta u_t + \Delta x_T{}^\top Q_T \Delta x_T \\
&\text{subj.to} && \Delta x_{t+1} = A_t^* \Delta x_t + B_t^* \Delta u_t \qquad t = 0,\dots,T-1 \\
& && x_0 = 0
\end{aligned}$$

where A_t^* ,B_t^* represent the linearization of the (nonlinear) system about the optimal trajectory.

The idea behind the solution is that we could track the optimal trajectory $\left(\mathbf{x}^{opt}, \mathbf{u}^{opt}\right)$ via a (stabilizing) feedback LQ controller.

Steps towards the solution:

Step 1: linearizing the system
Here we approximate the dynamics about the trajectory $\left(\mathbf{x}^{opt}, \mathbf{u}^{opt}\right)$. We know it is a feasible trajectory because it is the optimal trajecotry got from the controller of task 2.

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

where $\left(\mathbf{x}^{opt}, \mathbf{u}^{opt}\right)$ respectively are defined as

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

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

Step 2: Calculate the LQ optimal controller
We would compute the feedback gain by solving the optimal control problem

$$\begin{aligned}
\min_{\substack{\Delta x_1,\dots,\Delta x_T \\ \Delta u_0,\dots,\Delta u_{T-1}}} \quad & \sum_{t=0}^{T-1} \Delta x_t{}^\top Q_t^{reg} \Delta x_t + \Delta u_t{}^\top R_t^{reg} \Delta u_t + \Delta x_T{}^\top Q_T^{reg} \Delta x_T \\
\text{subj.to} \quad & \Delta x_{t+1} = A_t^{opt} \Delta x_t + B_t^{opt} \Delta u_t \qquad t = 0, ..., T-1 \\
& \Delta x_0 = given \qquad \text{acting as perturbation on the first state}
\end{aligned}$$

for some cost matrices $Q_t^{reg} \geq 0 \in \mathbb{R}^{n\times n}$, $R_t^{reg}{>}0 \in \mathbb{R}^{n\times m}$ and $Q_T^{reg} \geq 0 \in \mathbb{R}^{n\times n}$ (degree of freedom).
In order to solve this problem, we implemented the same fuction used in the Newton's algorithim changing the inputs since the problem is slightly different.
The differences between this problem and the LQR problem encountered in the Newton's algorithm are:

- here we don't have the affine terms in the cost function $\rightarrow$ no need for the augmented state.
- no S matrix since we don't have cross terms.
- the Δx_0 is not equal to zero.

N.B. $\Delta x_0 \neq 0$ because otherwise, the controller imported from the task 2 is going to be the controller we are looking for but we want to test our feedback controller on a perturbed trajecotry and this will be simulated by having this condition.
Using that function we do the following
Set $P_T = Q_T^{reg}$ and backward iterate t=T-1,...,0:

$$P_t = Q_t^r eg + A_t^{opt,\top} P_{t+1} A_t^{opt} - (A_t^{opt,\top} P_{t+1} B_t^{opt})(R_t^{reg} + B_t^{opt,\top} P_{t+1} B_t^{opt})(B_t^{opt,\top} P_{t+1} A_t^{opt})$$

and define for all t=0,...,T-1, the feedback gain $K_t^{reg} \in \mathbb{R}^{m\times n}$

$$K_t^{reg} := -(R_t^{reg} + B_t^{opt,\top} P_{t+1} B_t^{opt})^{-1} (B_t^{opt,\top} P_{t+1} A_t^{opt})$$

Step 3: track the generated (optimal) trajectory
We can apply the feedback controller designed on the linearization to the nonlinear system to track $\left(\mathbf{x}^{opt}, \mathbf{u}^{opt}\right)$

Namely, for alla t=0,...,T, we apply

$$u_t = u_t^{opt} + K_t^{reg}(x_t - x_t^{opt})$$

$$x_t = f(x_t, u_t)$$

with x_0 given.
The results are shown in Figure 8.

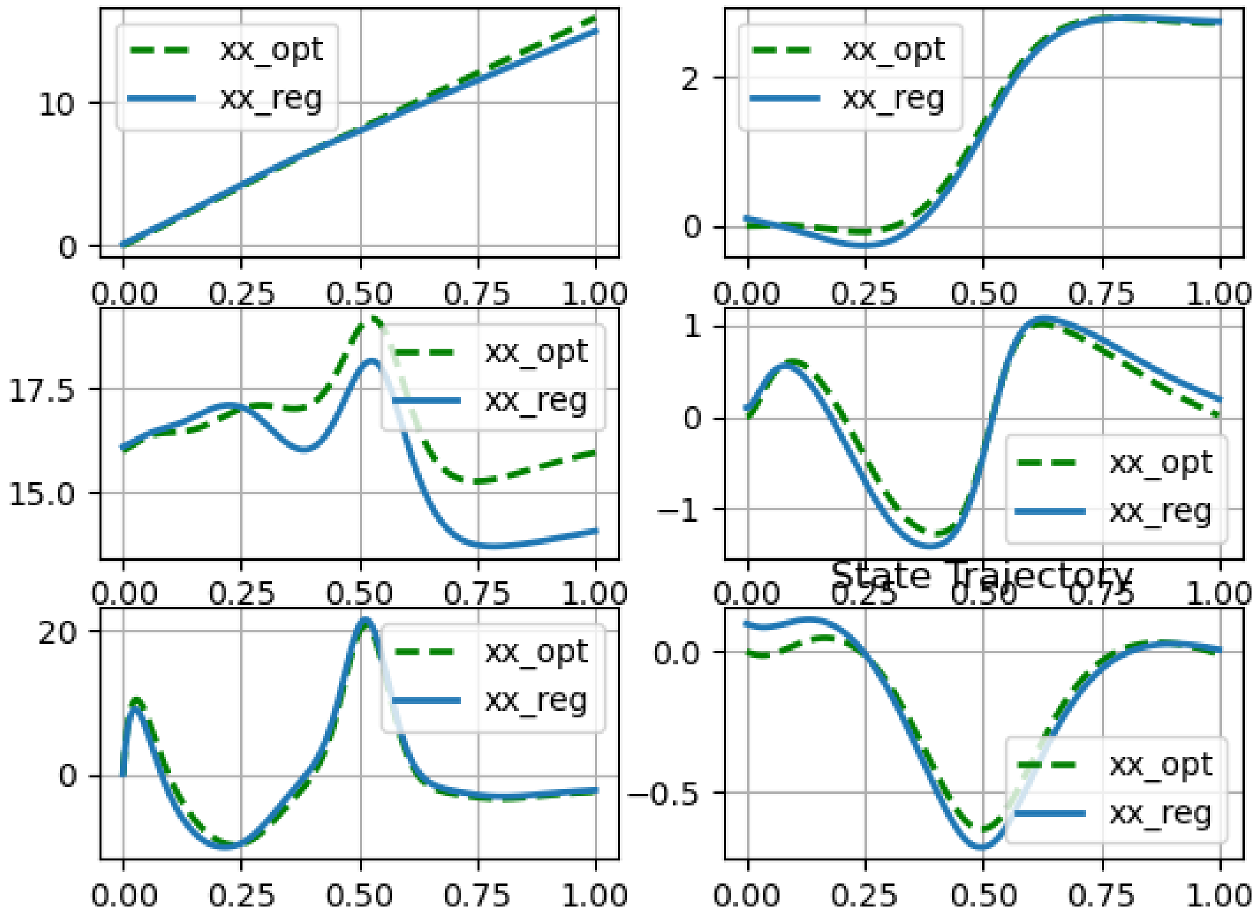


Figure 8: LQR plot tracking

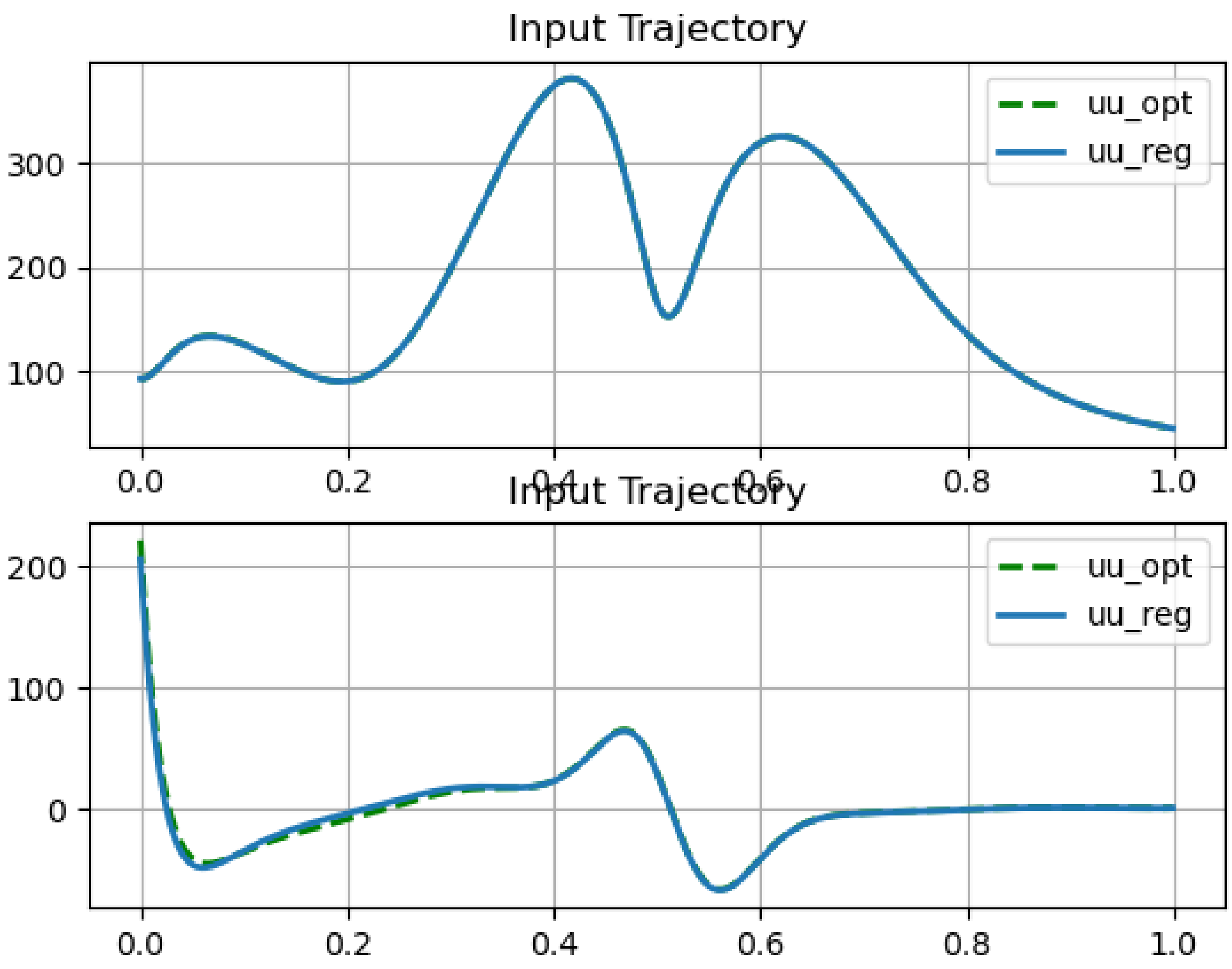
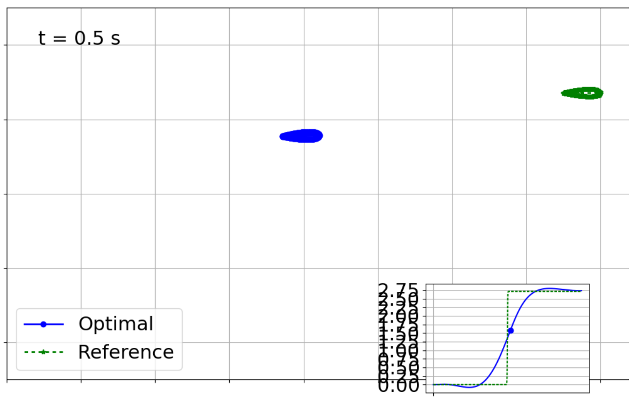


Figure 9: LQR inputs tracking

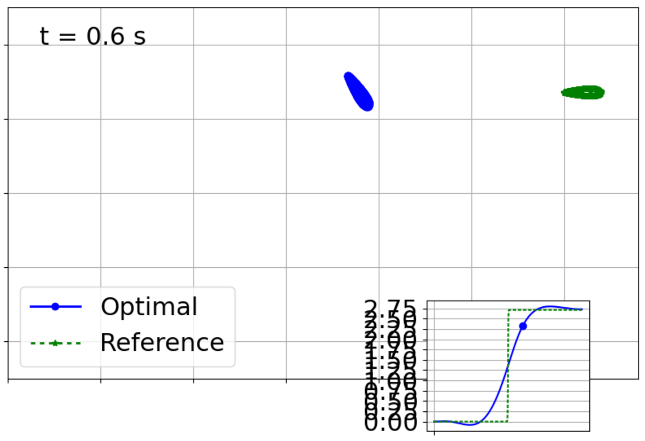
From figure 9 it can be noticed that we have some deviation in the first part of the inputs as it is the result of the perturbation of the initial state of the system.
Remark: This task is mainely based on the course handouts available at the following link 2.

Task 4 - Animation

To make the animation we created a geometrical shape that is popular in aviation which is airfoil to represent the aircraft wing. the airfoil is symmetrical and it's profile depends on the equations of NACA0015. At each update of the animation there is different position and orientation of the aircraft and this update is done using homogeneous transformation matrix that its parameters depend on the geometric states of the aircraft (x, z, θ).

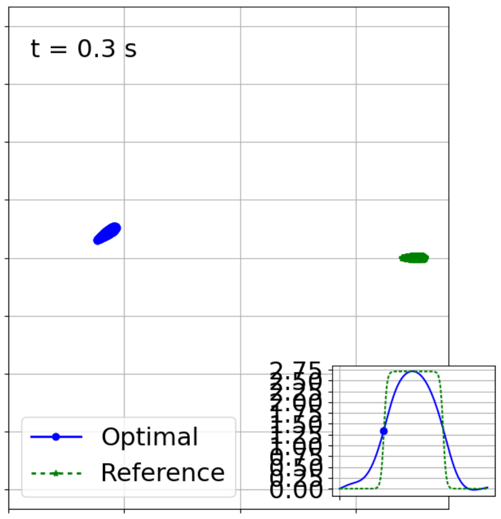


(a) Step Animation $t = 0.5$

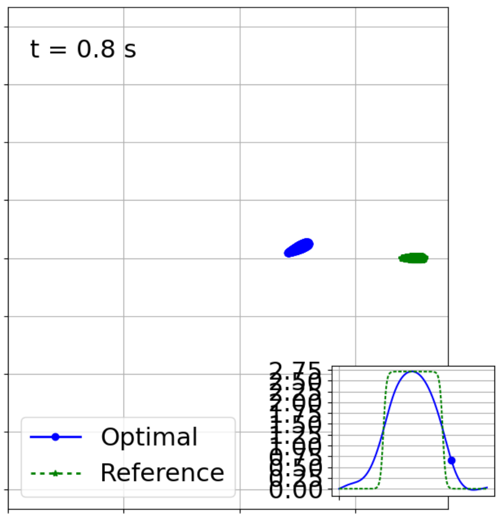


(b) Step Animation $t = 0.6$

Figure 10: Step Animation



(a) Acrobatic Animation $t = 0.3$



(b) Acrobatic Animation $t = 0.8$

Figure 11: acrobatic Animation

For the full animation please refer to the .gif files in the figures folder.

Remark: For more info, please visit the following site 3.

Conclusions

In this project we used Newton method to compute an (discrete) open-loop input that drives the non-linear dynamics of a simplified aircraft in a way that make the states trajectory satisfy specific geometrical reference trajectory, that is designed to make the aircraft perform some maneuvers. The discretization was done as shown in Task 0 section using forward Euler method.
Looking in task 1 to find that we used the shooting technique to reduce the optimal control problem. While in this method the descent direction requires the calculation of a new optimization problem, we formulated the problem such that it is suitable to be solved as an affine LQR with discrete-time Riccati-equation, although it can be solved as SQP (Sequential Quadratic Programing). We also considered using armijio stepsize to ensure the stability of the algorithm as it progresses. Lastely, in this task and the following one we used the approximated hessian in the first 8 iteration, to avoid numerical unstability.
In task 2 we took the same consideration as in task one with minor changes. We used a more complicated reference trajectory, that requires harder geometrical requirements. And for that we changed hyper-parameters like the weights in the cost function to make the dynamics converge in a more robust way. So in this case the difference in trajectories between taks 1 and task 2 requires different hyper-parameters.Also in Task 4 the animation was performed using a simulation of the update of the profile of the airfoil NACA0015, that represets aircraft wing. In task 3 the trajectory tracking was done using discrete-time riccati equation to compute the feedback gain (discrete-time LQR).And the plots in that section show how robust this algorithm to small perturbations in the initial states.
Future work requires to add some constraints on the input or the states, and analyse the behaviour of the algorithm to satisfy further constraints. It can be done using barrier function with scheduled weight(epsilon) to ensure both stability and convergence.

References

1. *https : //virtuale.unibo.it/pluginfile.php/*1501987*/mod_resource/content/*0*/ OPTCON_newton_optimal_control_*2022_12_18*.pdf*

2. *https : //virtuale.unibo.it/pluginfile.php/*1484622*/mod_resource/content/*0*/ OPTCON_optimal_control_design_*2022_11_28*.pdf*

3. *https : //en.wikipedia.org/wiki/NACA_airfoil*