TÉCNICO LISBOA

Numerical Optimal Control

MMAC

Project 1

Authors:

Carlo Andrea Venturini (104958) venturini.carloandrea@gmail.com
Rui Lança dos Santos (90549) rui.manuel.lanca.dos.santos@tecnico.ulisboa.pt

Grupo ♣

2022/2023 – 2º Semestre

Contents

0 Introduction

Throughout our project, we have the classic problem (in control theory and robotics), the inverted pendulum. It involves a rigid pendulum that is mounted on a cart, where the objective is to keep the pendulum balanced in the upright position by moving the cart back and forth, requiring precise control of the cart's position and velocity to maintain the pendulum's equilibrium.

1 Exercise 1

1.1 Lagrangian function and equations of motion

We have two mass couplings in our system that we have to consider, the mass M of the cart, and the mass m of the ball that is attached to the pendulum cable. The position vectors (with 2 components - vertical and horizontal) of these two masses is given by:

$$X_M(t) = (x_{1M}, x_{2M}) = (x, 0)$$

$$X_m(t) = (x_{1m}, x_{2m}) = (x + l \cdot sin\theta, l \cdot cos\theta)$$

the derivative being its velocity over time

$$\frac{dX_M(t)}{dt} = \dot{X_M} = (\dot{x}, 0)$$

$$\frac{dX_m(t)}{dt} = \dot{X_m} = (\dot{x} + \dot{\theta}l \cdot cos\theta, -\dot{\theta}l \cdot sin\theta)$$

and the velocity module:

$$v_m = \sqrt{|v_{1m}|^2 + |v_{1m}|^2} = \sqrt{(\dot{x})^2 + 2\dot{x}\dot{\theta}l \cdot cos(\theta) + (\dot{\theta}l)^2 cos^2(\theta) + (\dot{\theta}l)^2 sin^2(\theta)} \Leftrightarrow$$

$$\Leftrightarrow v_m = \sqrt{(\dot{x})^2 + 2\dot{x}\dot{\theta}l \cdot cos(\theta) + (\dot{\theta}l)^2}$$

$$v_M = \dot{x}$$

Given the Lagrangian function, that is, the function $L(x, \dot{x}, \theta, \dot{\theta})$ given by the difference of the kinetic energy (E_c) of the system minus its potential energy (E_p), given respectively by:

$$E_c = \frac{1}{2}mv_m^2 + \frac{1}{2}Mv_M^2 = \frac{1}{2}(M+m)\dot{x} + m\dot{x}\dot{\theta}l \cdot cos(\theta) + \frac{1}{2}m(\dot{\theta}l)^2$$

$$E_p = mgh_m + Mgh_M = mgx_{2m} + Mgx_{2M} = mgl \cdot cos(\theta)$$

$$L(x, \dot{x}, \theta, \dot{\theta}) = E_c - E_p \Leftrightarrow$$

$$\Leftrightarrow L(x, \dot{x}, \theta, \dot{\theta}) = \frac{1}{2}(M+m)(x')^2 + mlx'\theta' cos(\theta) + \frac{1}{2}ml^2|\theta'|^2 - mgl \cdot cos(\theta) \tag{1}$$

Apply the Lagrange theorem to the function $L(x, \dot{x}, \theta, \dot{\theta})$:

$$\begin{cases} \frac{d}{dt}\left(\frac{\partial L}{\partial \theta'}\right) = \frac{\partial L}{\partial \theta} \\ \frac{d}{dt}\left(\frac{\partial L}{\partial x'}\right) = \frac{\partial L}{\partial x} \end{cases} \tag{2}$$

we get, the following system of equations, given by equation (3):

$$\begin{cases} l\theta'' + x''cos(\theta) = g \cdot sin(\theta) \\ (M+m)x'' + ml\theta''cos(\theta) - ml|\theta'|^2 sin(\theta) = 0 \end{cases} \tag{3}$$

reduce the ODE system to a first-order system, thus defining the **state variables**, $Y = [y_1, y_2, y_3, y_4]$, which will accompany us throughout the entire project, given by the following transformation:

$$\begin{bmatrix} x \\ x' \\ \theta \\ \theta' \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$$

that substituting in the system (3), we end up with the system (4):

$$\begin{cases} \dot{y_1} = y_2 \\ \dot{y_3} = y_4 \\ l\dot{y_4} + \dot{y_2}cos(y_3) = gsin(y_3) \\ (M+m)\dot{y_2} + ml\dot{y_4}cos(y_3) - ml|y_4|^2 sin(y_3) = 0 \end{cases} \tag{4}$$

isolating the variables being derived in time to a vector $[\dot{y_1}, \dot{y_2}, \dot{y_3}, \dot{y_4}]$, in order to obtain the previous system in matrix form:

$$[W](\dot{Y}) = (K) \Leftrightarrow$$

$$\Leftrightarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & cos(y_3) & 0 & l \\ 0 & M+m & 0 & mlcos(y_3) \end{bmatrix} \begin{bmatrix} \dot{y_1} \\ \dot{y_2} \\ \dot{y_3} \\ \dot{y_4} \end{bmatrix} = \begin{bmatrix} y_2 \\ y_4 \\ g \cdot sin(\theta) \\ ml|y_4|^2 sin(y_3) \end{bmatrix}$$

getting the following ODE-IVP (ordinary differential equations with initial value problem) that governs the problem, for a time $t \in [0, T]$:

$$\Rightarrow \dot{Y} = [W]^{-1}(K) = F(t, Y(t)) \Leftrightarrow$$

$$\Leftrightarrow \begin{bmatrix} \dot{y_1} \\ \dot{y_2} \\ \dot{y_3} \\ \dot{y_4} \end{bmatrix} = \begin{bmatrix} y_2 \\ \frac{lm|y_4|^2 * sin(y_3)}{(-m*cos(y_3)^2+M+m)} - \frac{gm*cos(y_3)*sin(y_3)}{(-m*cos(y_3)^2+M+m)} \\ y_4 \\ \frac{gsin(y_3)(M+m)}{-lmcos(y_3)^2+Ml+lm} - \frac{lm|y_4|^2*cos(y_3)*sin(y_3)}{-lm\cdot cos(y_3)^2+M*l+l*m} \end{bmatrix}$$

with initial value condition known a priori:

$$Y(0) = Y_0 \in \mathbb{R}^4$$

1.2 Runge Kutta 4th order for solve the nonlinear problem

We noticed that the function of the dynamics , F, of $\dot{Y} = F(t, Y(t))$ is non-linear of Y, so to solve this system of ODE's numerically we need a method like the 4th order Runge-Kutta (RK-4), which is described by an iterative method, given by the system (5):

$$\begin{cases} y_{k+1} = y_k + h(\beta_1 F1 + \beta_2 F_2 + \beta_3 F_3 + \beta_4 F_4) \\ F_1 = f(t_k + \tau_1 h, y_k) = f_k \\ F_2 = f(t_k + \tau_2 h, y_k + \alpha_{2,1} h F_1) \\ F_3 = f(t_k + \tau_3 h, y_k + \alpha_{3,1} h F_1 + \alpha_{3,2} h F_2) \\ F_4 = f(t_k + \tau_4 h, y_k + \alpha_{4,1} h F_1 + \alpha_{4,2} h F_2 + \alpha_{4,3} h F_3) \end{cases} \tag{5}$$

and the respective constants used for the method are in the Butchers table (1):

τ	α_{ij}				β
0	0				1/6
1/2	1/2	0			1/3
1/2	0	1/2	0		1/3
1/2	0	0.7052	1	0	1/6

Table 1: Butcher's table for RK-4

1.3 Implementation and Results

The model was implemented in a MatLab file named `"exer1.m"`.

It was successfully implemented, we can see the physical behavior of the pendulum and the cart with different types of $\theta = y_3$, with 2 points of stability, that is when the pendulum starts from a situation of equilibrium $\theta = 0$ and $\theta = \pi$, so the physics of the system behaves well as expected.

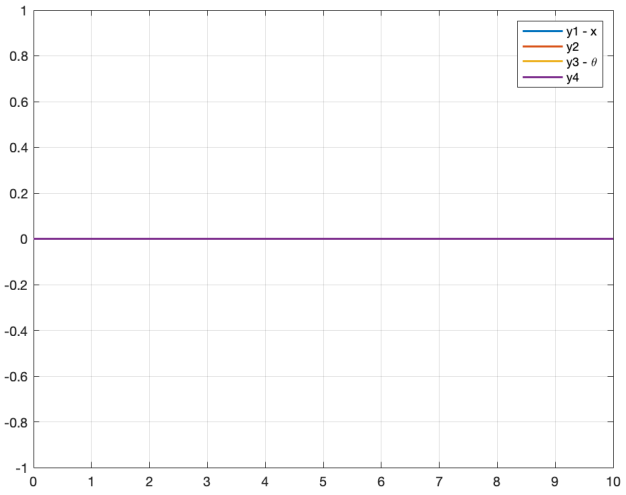


Figure 1: $y_3 = 0$

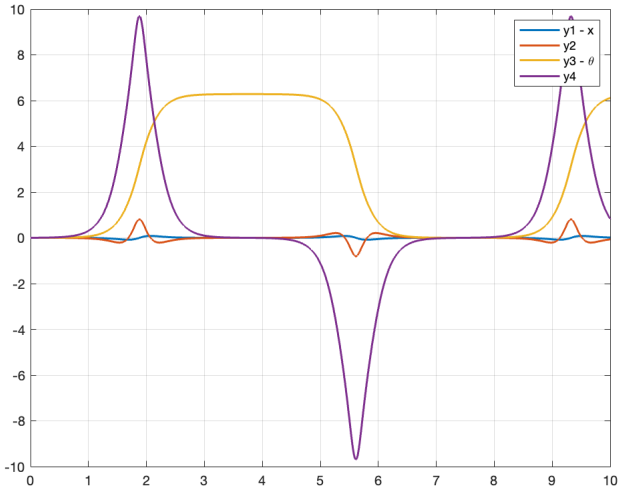


Figure 2: $y_3 = 0.001$

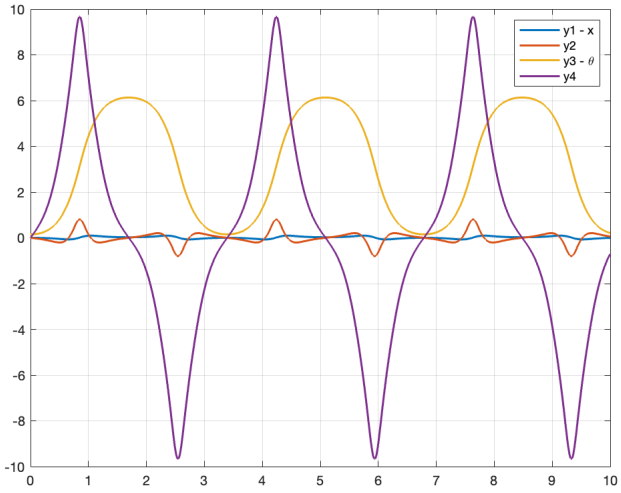


Figure 3: $y_3 = 0.15$

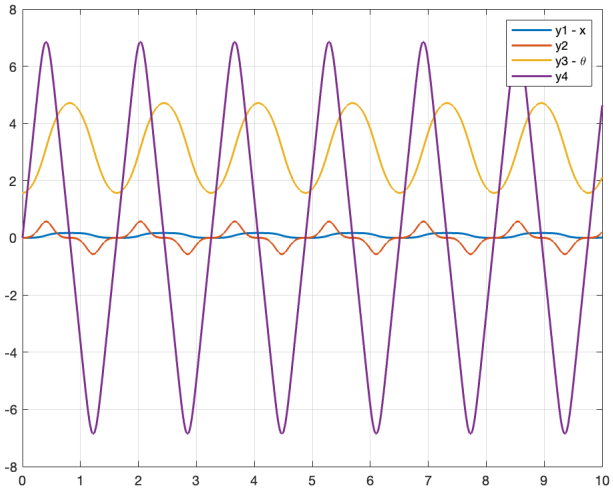


Figure 4: $y_3 = pi/2$

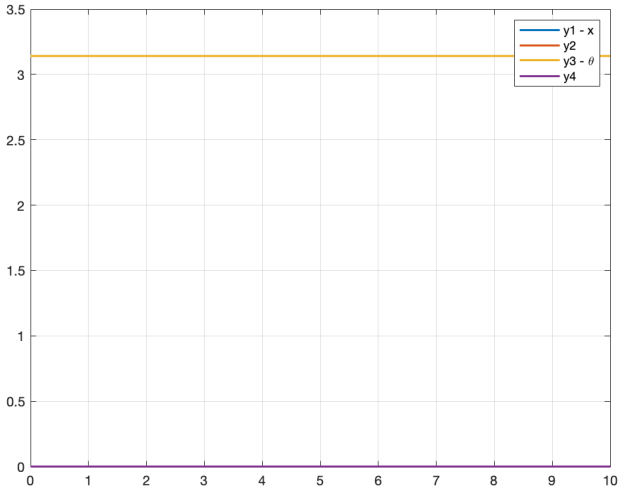


Figure 5: $y_3 = pi$

2 Exercise 2

Next, we can consider the control problem associated with the pendulum. In this case, we assume that our control u is an external force applied in the x direction on the cart. In this case, the equations would be modified using the second law of Newton, and we obtain:

$$\begin{cases} l\theta'' + x''cos(\theta) = g \cdot sin(\theta) \\ (M+m)x'' + ml\theta''cos(\theta) - ml|\theta'|^2 sin(\theta) = u \end{cases} \tag{6}$$

If we now introduce the vector of variables $Y = (y_1, y_2, y_3, y_4)^T = (x, x', \theta, \theta')^T$ the equations (6) can be written as a nonlinear ordinary differential system:

$$\frac{d}{dt}Y = F(Y) + G(Y)u \tag{7}$$

To solve this nonlinear problem, we can use different methods, but ultimately they all involve linearization. Thus, if we linearize the previous system around 0 (since this is our equilibrium point), we can rewrite the system as follows:

$$\frac{d}{dt}Y = F(0) + DF(0)Y + G(0)u \tag{8}$$

2.1 Linearization of the model, around the state zero (for the 4 variables)

The initial reasoning that we have to do is very similar to exercise 1, except that now we are in the presence of a new variable, the control $u(t)$.Therefore, the system resulting from Lagrange's theorem with the 4 variables will be different from the exercise, given by the system of equations reduce the ODE system to a first-order system (9):

$$\begin{cases} \dot{y_1} = y_2 \\ \dot{y_3} = y_4 \\ l\dot{y_4} + \dot{y_2}cos(y_3) = g \cdot sin(y_3) \\ (M+m)\dot{y_2} + ml\dot{y_4}cos(y_3) = ml|y_4|^2 sin(y_3) + u \end{cases} \tag{9}$$

isolating the variables being derived in time to a vector $[\dot{y_1}, \dot{y_2}, \dot{y_3}, \dot{y_4}]$, in order to obtain the previous system in matrix form:

$$[W](\dot{Y}) = (K) \Leftrightarrow$$

$$\Leftrightarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & cos(y_3) & 0 & l \\ 0 & M+m & 0 & mlcos(y_3) \end{bmatrix} \begin{bmatrix} \dot{y_1} \\ \dot{y_2} \\ \dot{y_3} \\ \dot{y_4} \end{bmatrix} = \begin{bmatrix} y_2 \\ y_4 \\ g \cdot sin(\theta) \\ ml|y_4|^2 sin(y_3) + u \end{bmatrix}$$

getting the following ODE-IVP (ordinary differential equations with initial value problem) that governs the problem, for a time $t \in [0, T]$:

$$\Rightarrow \dot{Y} = [W]^{-1}(K) = F(Y) + G(Y)u \Leftrightarrow$$

$$\Leftrightarrow \begin{bmatrix} \dot{y_1} \\ \dot{y_2} \\ \dot{y_3} \\ \dot{y_4} \end{bmatrix} = \begin{bmatrix} y_2 \\ \frac{lm|y_4|^2 * sin(y_3)+u}{(-m*cos(y_3)^2+M+m)} - \frac{gm*cos(y_3)*sin(y_3)}{(-m*cos(y_3)^2+M+m)} \\ y_4 \\ \frac{gsin(y_3)(M+m)}{-lmcos(y_3)^2+Ml+lm} - \frac{cos(y_3)(lm|y_4|^2*sin(y_3)+u)}{-lm\cdot cos(y_3)^2+M*l+l*m} \end{bmatrix}$$

where $F(Y)$ and $G(Y)u$ are vectors given respectively by:

$$F(Y) = \begin{bmatrix} y_2 \\ \frac{lm|y_4|^2*sin(y_3)}{(-m*cos(y_3)^2+M+m)} - \frac{gm*cos(y_3)*sin(y_3)}{(-m*cos(y_3)^2+M+m)} \\ y_4 \\ \frac{gsin(y_3)(M+m)}{-lmcos(y_3)^2+Ml+lm} - \frac{cos(y_3)(lm|y_4|^2*sin(y_3))}{-lm\cdot cos(y_3)^2+M*l+l*m} \end{bmatrix} ; \; G(Y)u = \begin{bmatrix} 0 \\ \frac{1}{(-m*cos(y_3)^2+M+m)} \\ 0 \\ \frac{cos(y_3)}{-lm\cdot cos(y_3)^2+M*l+l*m} \end{bmatrix} u$$

let's calculate the derivative of F with respect to Y, that is $DF(Y) = \nabla_y F(y)$. Finally we get all the requested parameters:

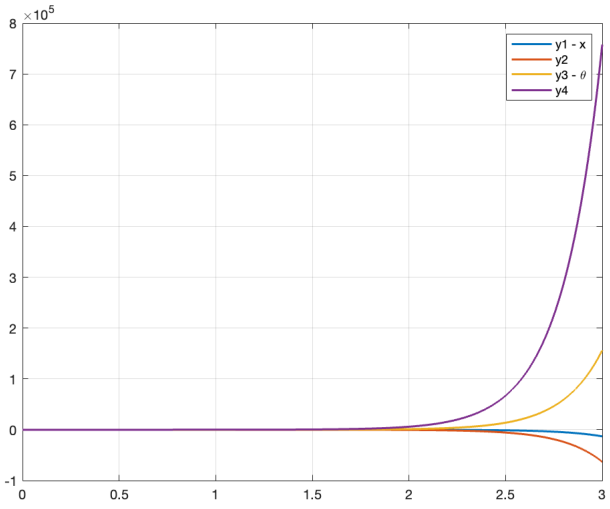
$$A = DF(0) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -(gm)/M & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & (gM+gm)/(Ml) & 0 \end{bmatrix} ; \; F(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} ; \; B = G(0) = \begin{bmatrix} 0 \\ 1/M \\ 0 \\ -1/(Ml) \end{bmatrix}$$

and we are ready to linearize our problem:

$$\dot{Y} = AY + Bu$$

2.2 Experimental simulation

To test the model and be able to make comparisons, we implement the model with null control, $u(t) = 0$,the model was implemented in a MatLab file named `"exer2.m"`.Whose result is present in the following image, with the initial condition $Y = [0; 0; 0.15; 0]$:



we can see that, after a certain instant, the solution (state solution) explodes, because the physical system leaves far from the equilibrium point where the linearization is being done, around $Y = (0, 0, 0, 0)$, not having a realistic physical behavior.

2.3 The valid time range where the linearization it holds

In order to verify this, let's put the linearized version (of this exercise 2) and the non-linearized version (of exercise 1) side by side, and see until when we start to see differences. For both cases, the initial condition of $Y = (0, 0, 0.15, 0)$ was chosen:

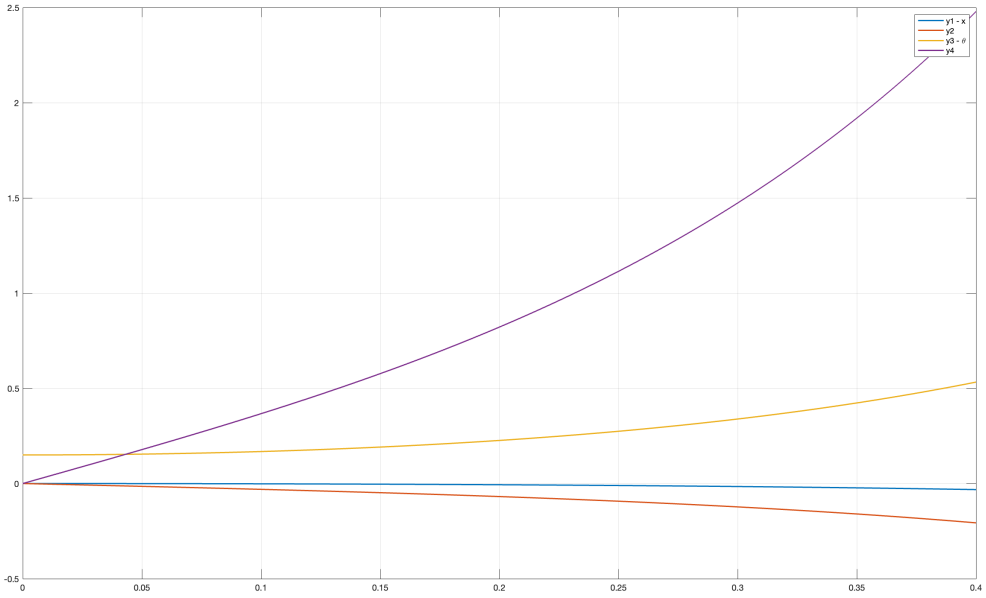


Figure 6: Linearized model

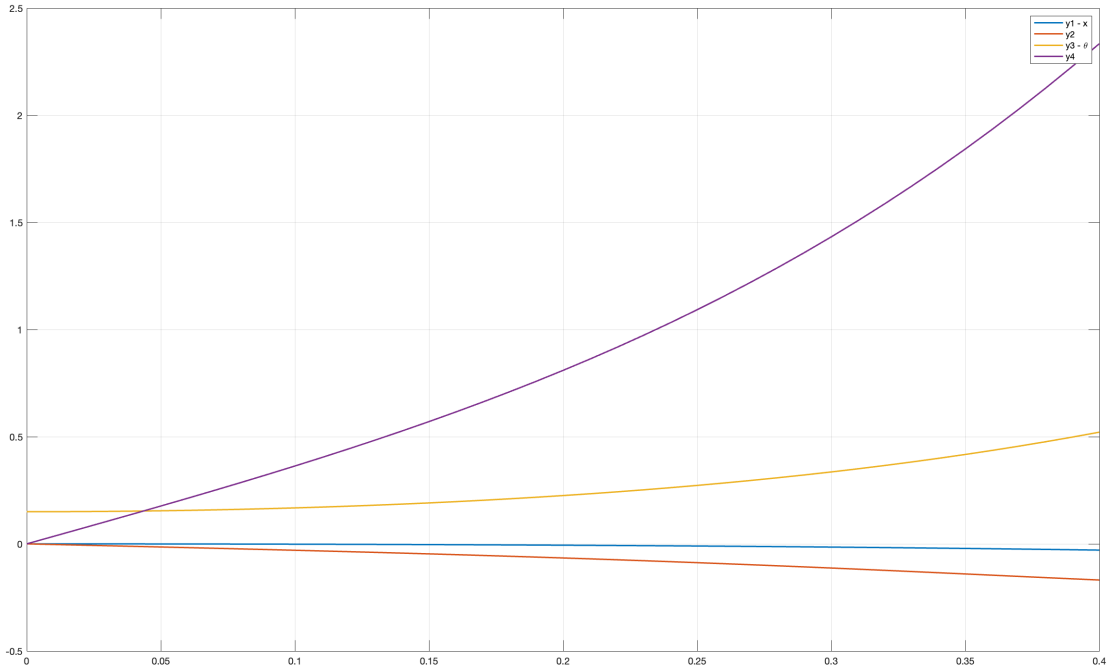


Figure 7: Non-linearized model

by observation, we see that from approximately 0.4 units of time, the models start to diverge, and the linearization starts to make no sense.

3 Exercise 3

If we now consider the goal of controlling u in order to make the angle θ as close to 0 as possible, we can define the optimal control problem:

$$\begin{cases} min\, J(Y,u) \\ \text{s.t. } (Y,u) \text{ solves } \dot{Y} = AY + Bu \end{cases} \tag{10}$$

with

$$J(Y,u) = \frac{\alpha_2}{2}|y_3(T)|^2 + \frac{\alpha_1}{2}\int_0^T |y_3|^2 dt + \frac{\alpha_3}{2}\int_0^T |u|^2 dt \tag{11}$$

We want to solve the quadratic control problem, also referred to as Linear-Quadratic Regulator (LQR).

3.1 Analytical deduction of the adjoint equation and the gradient of cost

First, let's start by rewriting the OCP:

$$min\, \tilde{J}(y(u),u)$$

where the reduced cost functional is given by the following equation:

$$J(u) = \tilde{J}(y(u),u) = \psi(y(T)) + \int_0^T [f(y(s)) + q(u(s))]ds \tag{12}$$

where the inner products are given by:

$$\psi(y(t)) = \frac{1}{2}\langle Sy(t), y(t)\rangle_{\mathbb{R}^N};\ f(y(t)) = \frac{1}{2}\langle Py(t), y(t)\rangle_{\mathbb{R}^N};\ q(u(t)) = \frac{1}{2}\langle Qu(t), u(t)\rangle_{\mathbb{R}^M}$$

the vectors $y(t) \in \mathbb{R}^N$ and $u(t) \in \mathbb{R}^M$ have dimension N and M respectively, and $S, P \in \mathbb{R}^{N\times N}$ are $N \times N$ positive symmetric semidefinite matrices, while $Q \in \mathbb{R}^{M\times M}$ is a $M \times M$ symmetric positive definite matrix, under the usual dynamics:

$$\begin{cases} \dot{y}(t) = Ay(t) + Bu(t)\,;\ 0 < t < T \\ y(0) = \xi \end{cases} \tag{13}$$

for a given fixed $T > 0$, and the class $u \in C_T$ of piecewise continuous admissible controls. The matrices $A \in \mathbb{R}^{N\times N}$ and $B \in \mathbb{R}^{N\times M}$ are the matrices previously calculated by linearizing our problem in exercise 2.

In our particular case we have $N = 4$ and $M = 1$, and the matrices S, P and Q are given by:

$$S = \alpha_2 \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix};\ P = \alpha_1 \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix};\ Q = \alpha_3$$

Let's introduce the multiplier (adjoint state) $p(t) \in \mathbb{R}^N$,in order to introduce one more zero term in the cost equation:

$$J(u) = \psi(y(T)) + \int_0^T [f(y(s)) + q(u(s))]ds = J(u) + \int_0^T \langle Ay(s) + Bu(s) - \dot{y}(s), p(t)\rangle ds \Leftrightarrow$$

$$\Leftrightarrow J(u) = J(u) + \int_0^T [p^T(Ay + Bu) - p^T\dot{y}]\, ds$$

Using integration by parts on the second term, we are left with:

$$\Leftrightarrow J(u) = J(u) + \int_0^T [p^T(Ay + Bu) - \dot{p}^T y]\, ds - [p(t)^T y(t)]|_0^T \Leftrightarrow$$

$$\Leftrightarrow J(u) = J(u) + \int_0^T [p^T(Ay + Bu) - \dot{p}^T y]\, ds - p(T)^T y(T) + p(0)^T y(0) \Leftrightarrow$$

$$\Leftrightarrow J(u) = J(u) + \int_0^T [p^T Ay + p^T Bu - \dot{p}^T y]\, ds - p(T)^T y(T) + p(0)^T y(0) \Leftrightarrow$$

$$\Leftrightarrow J(u) = J(u) + \int_0^T [\langle Ay, p\rangle + \langle Bu, p\rangle - \dot{p}^T y]\, ds - p(T)^T y(T) + p(0)^T y(0) \Leftrightarrow$$

$$\Leftrightarrow J(u) = J(u) + \int_0^T [\langle y, A^T p\rangle + \langle u, B^T p\rangle - \dot{p}^T y]\, ds - p(T)^T y(T) + p(0)^T y(0) \Leftrightarrow$$

$$\Leftrightarrow J(u) = J(u) + \int_0^T [\langle A^T p, y\rangle + \langle B^T p, u\rangle - \dot{p}^T y]\, ds - p(T)^T y(T) + p(0)^T y(0)$$

finally the whole expression becomes:

$$J(u) = \psi(y(T)) + \int_0^T [f(y(s)) + q(u(s))]ds +$$

$$+ \int_0^T [\langle A^T p, y\rangle + \langle B^T p, u\rangle - \dot{p}^T y]\, ds - p(T)^T y(T) + p(0)^T y(0)$$

Proposition: For every $u \in C_T$, the initial value problem (13) has a unique solution given by (14):

$$y(t; \xi, u) = e^{tA}\xi + \int_0^t [e^{(T-s)A} Bu(s)]ds \tag{14}$$

In order to arrive at the gradient of J, we need a notion of the differential (or derivative) of J, for that we will use the Gateaux differential (or Gateaux derivative) which is a generalization of the concept of directional derivative in differential calculus.

Let $v \in C_T$, $\epsilon \neq 0$ and $u_\epsilon = u^* + \epsilon v \in C_T$, and let y_ϵ be the corresponding state (or trajectory) variable. We define the differential of J at u^* along the v direction:

$$DJ(u^*)v = \lim_{\epsilon \to 0} \frac{J(u^* + \epsilon v) - J(u^*)}{\epsilon} = \lim_{\epsilon \to 0} \frac{J(u_\epsilon) - J(u^*)}{\epsilon} \tag{15}$$

then the cost difference:

$$J(u_\epsilon) - J(u^*) = \psi(y_\epsilon(T)) - \psi(y^*(T)) + \int_0^T [f(y_\epsilon) - f(y^*) + q(u_\epsilon) - q(u^*)]ds +$$

$$+ \int_0^T [\langle A^T p, y_\epsilon - y^* \rangle + \langle B^T p, u_\epsilon - u^* \rangle - \dot{p}^T (y_\epsilon - y^*)]\, ds - p(T)^T (y_\epsilon(T) - y^*(T)) + p(0)^T (y_\epsilon(0) - y^*(0))$$

$$\begin{aligned} \Leftrightarrow J(u_\epsilon) - J(u^*) = \psi(y_\epsilon(T)) - \psi(y^*(T)) + \int_0^T [f(y_\epsilon) - f(y^*) + q(u_\epsilon) - q(u^*)]ds + \\ + \int_0^T [\langle A^T p, y_\epsilon - y^* \rangle + \langle B^T p, u_\epsilon - u^* \rangle - \dot{p}^T (y_\epsilon - y^*)]\, ds + \\ - p(T)^T (y_\epsilon(T) - y^*(T)) + p(0)^T (y_\epsilon(0) - y^*(0)) \end{aligned} \tag{16}$$

Let's do a little axillary calculation, of the following difference:

$$u_\epsilon - u^* = u^* + \epsilon v - u_* = \epsilon v$$

$$y_\epsilon - y^* = e^{tA}\xi + \int_0^t [e^{(T-s)A} B(u^* + \epsilon v)] - \left[e^{tA}\xi + \int_0^t [e^{(T-s)A} B u^*] ds \right] = \epsilon \int_0^t [e^{(T-s)A} B v(s)] ds = \epsilon z(t)$$

where

$$z(t) = \int_0^t [e^{(T-s)A} B v(s)] ds; \text{ notice that } z(0) = 0$$

and for $\psi(y(t)) = \frac{1}{2}\langle Sy(t), y(t) \rangle = \frac{1}{2} y^T S y$ their difference:

$$\psi(y_\epsilon(t)) - \psi(y^*(t)) = \frac{1}{2} y_\epsilon^T S y_\epsilon - \frac{1}{2} y^T S y$$

making an analogy with the expression $a^2 - b^2 = (a-b)(a+b) = 2b(a-b)$, we get:

$$\psi(y_\epsilon(t)) - \psi(y^*(t)) = \frac{1}{2} [y_\epsilon(t) - y(t)]^T S [y_\epsilon(t) + y(t)]$$

$$= \frac{1}{2} \left[[y_\epsilon(t) - y(t)]^T S [y_\epsilon(t) - y(t)] + 2 [y_\epsilon(t) - y(t)]^T S y(t) \right] = \epsilon^2 \frac{1}{2} z^T(t) S z(t) + \epsilon z^T(t) S y(t)$$

finally, we can write

$$\psi(y_\epsilon(t)) - \psi(y^*(t)) = \epsilon^2 \psi(z(t)) + \epsilon \langle Sy(t), z(t) \rangle$$

Similarly, for $f(y(t)) = \frac{1}{2}\langle Py(t), y(t) \rangle$ and $q(u(t)) = \frac{1}{2}\langle Qu(t), u(t) \rangle$:

$$f(y_\epsilon(t)) - f(y^*(t)) = \epsilon^2 f(z(t)) + \epsilon \langle Py(t), z(t) \rangle$$

$$q(u_\epsilon(t)) - q(u^*(t)) = \epsilon^2 q(v(t)) + \epsilon \langle Qu(t), v(t) \rangle$$

Inserting the above expressions into (16), we get:

$$J(u_\epsilon) - J(u^*) = \epsilon^2 \psi(z(t)) + \epsilon \langle Sy(t), z(t) \rangle + \int_0^T [\epsilon^2 f(z(t)) + \epsilon \langle Py(t), z(t) \rangle + \epsilon^2 q(v(t)) + \epsilon \langle Qu(t), v(t) \rangle] dt +$$

$$+\int_0^T [\epsilon\langle A^T p(t), z(t)\rangle + \epsilon\langle B^T p(t), v(t)\rangle + \epsilon\dot{p}^T(t)z(t)]dt - \epsilon p(T)^T z(T) + \epsilon p(0)^T z(0)$$

, dividing by $\epsilon \neq 0$ and letting $\epsilon \to 0$, we get:

$$DJ(u^*)v = \langle Sy(t), z(t)\rangle + \int_0^T [\langle Py(t), z(t)\rangle + \langle Qu(t), v(t)\rangle]dt +$$

$$+\int_0^T [\langle A^T(t), z(t)\rangle + \langle B^T p(t), v(t)\rangle + \langle \dot{p}(t), z(t)\rangle]dt - \langle p(T), z(T)\rangle \Leftrightarrow$$

$$\Leftrightarrow DJ(u^*)v = \langle Sy(t) - p(T), z(t)\rangle + \int_0^T [\langle \dot{p}(t) + A^T p(t)Py(t), z(t)\rangle + \langle Qu(t) + B^T p(t), v(t)\rangle]dt$$

get rid of the terms containing $z(t)$ and the expression becomes:

$$DJ(u^*)v = \int_0^T [\langle Qu(t) + B^T p(t), v(t)\rangle]dt;\ \forall u, v \in C_T \tag{17}$$

we now choose $p = p^*$, where p^* is the solution of th adjoint final-problem (18):

$$\begin{cases} \dot{p^*}(t) = -A^T p^*(t) + Py^*(t)\,;\ t \in (0,T) \\ p(T) = Sy^*(T) \end{cases} \tag{18}$$

Since formula (17) expresses the linear functional $v \mapsto DJ(u^*)v$ as an inner product in $L^2(0,T)$, we coherently define the gradient of J at u^* by the formula

$$DJ(u^*)v = \langle \nabla J(u^*), v\rangle_{L^2(0,T)} = \int_0^T \langle \nabla J(u^*), v\rangle dt = \int_0^T [\langle Qu(t) + B^T p(t), v(t)\rangle]dt$$

$$\Rightarrow \nabla J(u^*) = Qu(t) + B^T p(t) \tag{19}$$

3.2 The Steepest descent Algorithm - finding an open loop control

Here we will apply a numerical method to obtain the solution of our OCP, since we have access to the cost functional gradient, given above by equation (19), then we will apply **the gradient descent with constant step size**:

$$u_{k+1} = u_k - \tau\nabla J(u_k);\ k = 0, 1, \ldots \tag{20}$$

where τ is the step-length, and will be used as a constant in this algorithm. We will need to numerically approximate some functions, such as our control function $u(t)$, the state solution $y(t)$ and the adjunct solution $p(t)$:

$$u_k(t) = [u_k^0, u_k^1, \ldots, u_k^n, \ldots, u_k^N]$$

$$y_k(t) = [y_k^0, y_k^1, \ldots, y_k^n, \ldots, y_k^N]$$

$$p_k(t) = [p_k^0, p_k^1, \ldots, p_k^n, \ldots, p_k^N]$$

where k is the iteration number of the descent method ranging from $k = 0$ to $k = K_{\text{final}}$ (until converge, or stop the algorithm), and n is the time component number of each of the vectors, for that, a time discretization must be done, where we divide the interval $[0, T]$ into N sub-intervals:

$$\Delta t = \frac{T}{N};\ t_n = n \cdot \Delta t;\ n = 0, 1, \ldots, N$$

Initially, we start by giving an approximation to the initial control:

$$u_0(t)$$

. It may happen that we already have a priori knowledge of an approximation of a control $u(t)$ that fits our OCP, which will allow us to have a faster convergence of the method. While the stopping conditions of point (V) have not been satisfied, it will be iterated from k=0 until these conditions are met, repeating points (I), (II), (III), (IV) and (V) sequentially:

(I) Determine the state solution $y_k(t)$, using the Backward (Implicit) Euler method:

$$\begin{cases} \dot{y}_k(t) = Ay_k(t) + Bu_k(t) \\ y_k(0) = \xi \end{cases} \Rightarrow \begin{cases} \frac{y_k^n - y_n^{n-1}}{\Delta t} = Ay_k^n + Bu_k^n \\ y_k^0 = \xi \end{cases} \Leftrightarrow$$

$$\Leftrightarrow \begin{cases} y_k^n = (I_N - A)^{-1} \left[\frac{1}{\Delta t} y_k^{n-1} + Bu_k^n\right] \\ y_k^0 = \xi \end{cases}$$

where I_N is an identity matrix of dimension $N \times N$, to solve this IVP ODE we iterate the method from $n = 1$ to $n = N$.

(II) Determine the Adjoint solution $p_k(t)$, using again the Backward (Implicit) Euler method, but Backward time:

$$\begin{cases} \dot{p}_k(t) = -A^T y_k(t) + -Pu_k(t) \\ p_k(T) = Sy_k(T) \end{cases} \Rightarrow \begin{cases} \frac{p_k^{n+1} - p_n^n}{\Delta t} = -A^T p_k^n - Py_k^n \\ p_k^N = Sy_k^N \end{cases} \Leftrightarrow$$

$$\Leftrightarrow \begin{cases} p_k^n = \left(-I_N - A^T\right)^{-1} \left[-\frac{1}{\Delta t} p_k^{n+1} - Py_k^n\right] \\ p_k^N = Sy_k^N \end{cases}$$

to solve this FVP ODE we iterate the method from $n = N - 1$ to $n = 0$.

(III) Calculate the new gradient and update the control, with descent method:

$$\begin{cases} \nabla J(u_k) = Qu_k + B^T p_k \\ u_{k+1} = u_k - \tau \nabla J(u_k) \end{cases}$$

(IV) Calculate the Functional Cost:

$$J_k = J(u_k(t)) = \psi(y_k(T)) + \int_0^T [f(y_k(t)) + q(u_k(t))]dt$$

Whose integrals were approximated by the composite trapezoidal rule with uniform grid:

$$\int_a^b f(x)dx \approx \sum_{n=1}^{N} \frac{f(x_{k-1}) + f(x_k)}{2} \Delta x_k; \text{ where } \Delta x_k = x_k - x_{k-1} = \Delta x = \frac{b-a}{N}$$

$$\Leftrightarrow \int_a^b f(x)dx \approx \frac{\Delta x}{2} \sum_{n=1}^{N} f(x_{k-1}) + f(x_k) =$$

$$= \frac{\Delta x}{2} \left[f(x_0) + 2f(x_1) + 2f(x_2) + \ldots + 2f(x_{N-1}) + f(x_N) \right] \Leftrightarrow$$

$$\Leftrightarrow \int_a^b f(x)dx \approx \Delta x \left[\frac{f(x_0) + f(x_N)}{2} + \sum_{n=1}^{N-1} f(x_k) \right]$$

(V) Check the stopping criterion, and here we have more than one criterion that we can choose, either a maximum number of completed iterations or for example:

$$|u_{k+1} - u_k| < \epsilon$$

$$|J_{k+1} - J_k| < \epsilon$$

$$|\nabla J(u_{k+1})| < \epsilon$$

where ϵ is a constant (the tolerance). In our case we mostly use the 2nd stopping method.

3.3 Plot of the uncontrolled state, the control and the controlled states

The model was implemented in a MatLab file named `"exer3.m"`. The initial condition of $Y = (0, 0, 0.15, 0)$ was used, and for the constants $\alpha_1 = \alpha_2 = 1$ and $\alpha_3 = 10^{-6}$, and a tolerance of $\epsilon = 10^{-7}$ for the stopping criterion.

For the control of $u(t) = 0$, the following result was obtained, shown in the following image:

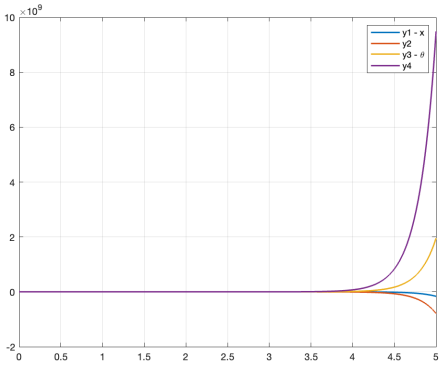


Figure 8: Uncontrolled state's, with $u(t) = 0$

As we can see the model explodes once again in the absence of control with linearization.

For a time between $[0; 5]$, a descent step of $\tau = 10^{-}18$ was used, and we verified that the greater the time interval that we are choosing, the smaller it will have to be the step, otherwise our method does not seem to converge. The result is shown in the following image, converging on $k = 104$ iterations, with a corresponding cost value of $J = 0.0062$.

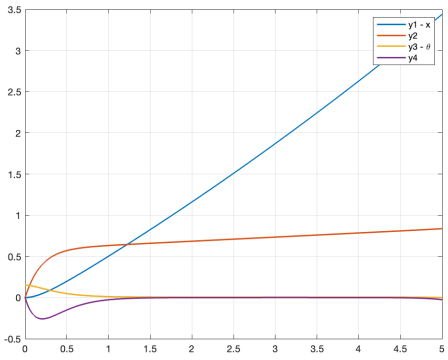


Figure 9: Controlled state's, with $u(t)$ optimized

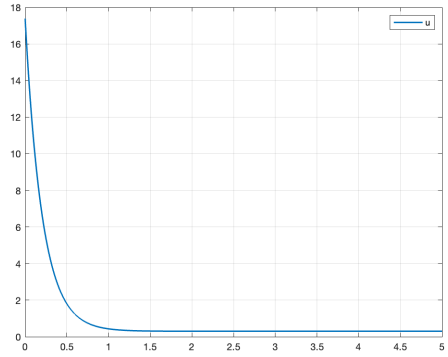


Figure 10: Resultant control $u(t)$

One of the things that we have to notice is that the weights that are present in the minimize equation are only relative to the control $u(t)$, and to the state variable $y_3(t)$, this being one of those that will tend to zero (and consequently $y_4(t)$ is also its derivative), around equilibrium, only y_1 and y_2 are diverging, probably because they are not included in the equation to be minimized.

4 Exercise 4

Consider now the problem of

$$\begin{cases} min\, J(Y,u) \\ \text{s.t. } (Y,u) \text{ solves } \dot{Y} = AY + Bu \end{cases} \tag{21}$$

with

$$J(Y,u) = \frac{\alpha_1}{2}\int_0^T |y_2|^2 + |y_3|^2 + |y_4|^2 dt + \frac{\alpha_2}{2}\int_0^T |u|^2 dt \tag{22}$$

with $\alpha_1 = 1$, $\alpha_2 = 10^{-6}$, the matrices S, P and Q are given by:

$$S = 0;\ P = \alpha_1 \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix};\ Q = \alpha_2$$

4.1 Feedback control of type $u = KY$, and solving the Algebraic Riccati equation

In the special case $T = \infty$ and $\psi = 0$, we want to minimize the cost functional

$$J(u) = \frac{1}{2}\int_0^\infty [Py \cdot y + Qu \cdot u] ds$$

under the given dynamics; this problem is sometimes referred to as time-invariant LQR. Since there is no terminal payoff, there are no constraints on the final value $p(T)$ or, equivalently, on $R(T)$. We can thus seek a constant matrix R that fulfills the Riccati matrix equation, which is

$$RA + A^\top R - RBQ^{-1}B^\top R + P = 0.$$

This equation is also referred to as the algebraic Riccati matrix equation. Since Q is positive definite, we can express it as $Q = H^\top H$, so that the cost function becomes

$$J(u) = \frac{1}{2}\int_0^\infty [|Hy|^2 + Qu \cdot u] dt.$$

Moreover, let us assume that the pair (A, H) is observable. It is well-known that the Riccati equation admits a variety of solutions: it may have no solution at all or, if it does have one, it can admit both real and complex solutions, some of them being Hermitian or symmetric matrices; there can even be infinitely many solutions. Due to the underlying physical problem, however, only non-negative solutions are of interest to us. Therefore, we are mainly concerned with the existence and uniqueness of such a solution.

We also assume that (A, B) is controllable, so that the state can be steered to the origin in finite time and, due to the observability assumption, equilibrium at the origin is the only solution corresponding to the vanishing of the cost function. Hence, the solution of the optimal

control problem converges to the origin, so that $\lim_{t\to\infty} y(t) = 0$. In other words, the optimal control required for moving from any state to the origin can be found by applying a feedback law and letting the system evolve in closed loop. Precisely, the following result holds:

The optimal control is given by:

$$u^* = -Q^{-1}B^T R^* y^* \Leftrightarrow u^* = Ky^*$$

where R^* is the unique (symmetric) positive definite solution of the algebraic Riccati equation, and y^* is the solution of the linear ODE system:

$$\dot{y^*}(t) = \left[A - BQ^{-1}B^T R^*\right] y^*(t), \quad y^*(0) = \xi$$

Moreover, the minimum cost value is given by:

$$J(\tilde{u}) = \frac{1}{2} R \xi^T \xi$$

4.2 Obtaining the Riccati matrix with the matlab solver

To obtain the Riccati matrix, R, with the MatLab solver, the following command was used:

```
[K,R,L] = lqr(A,B,P,Q)
```

with which we obtain the matrix, R, given by:

$$R \approx \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0.5859 & 1.6309 & 0.2955 \\ 0 & 1.6309 & 6.4692 & 0.8300 \\ 0 & 02955 & 0.8300 & 0.1518 \end{bmatrix}$$

in the following values were obtained for K:

$$K = -Q^{-1}B^T R \approx \begin{bmatrix} 0 & -1000 & 5800.72 & 1630.93 \end{bmatrix}$$

After obtaining the matrix R, we solve the following IVP ODE, to discover the state solution $y(t)$, for that we use the RK-4:

$$\begin{cases} \dot{y}(t) = [A - BQ^{-1}B^T R]y(t) \\ y(0) = \xi \end{cases}$$

and finally to take control:

$$u(t) = -Q^{-1}B^T R y(t) = Ky(t)$$

4.2.1 Implementation and results

This model was implemented in the file named `"exer4_matlab.m"`, which resulted in the following results shown in the images below. With the initial condition of $Y = (0, 0, 0.15, 0)$, the cost value obtained was $J = 0.0728$.

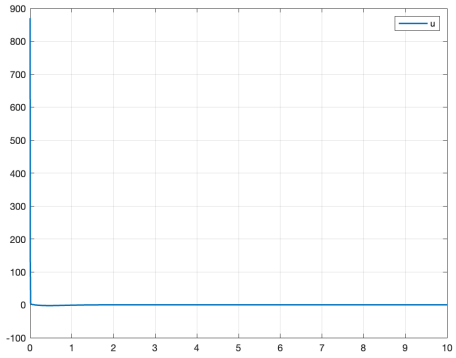


Figure 11: Control $u(t)$

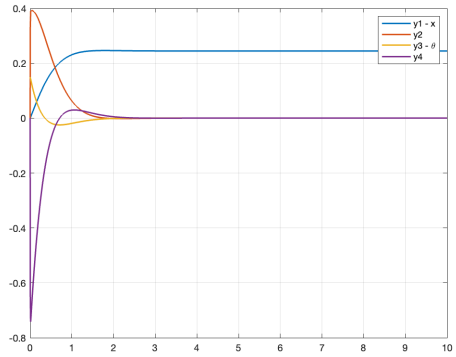


Figure 12: States $y(t)$

4.3 Obtaining the Riccati matrix with Newton's method

It is well known that the stationary Riccati equation:

$$RA + A^T R - RBQ^{-1}B^T R + P = 0$$

can be formulated in the following nonlinear functional operator:

$$F(R) = RA + A^T R - RBQ^{-1}B^T R + P$$

which can be solved by the Newton method, resulting iterative sequence:

$$R^{k+1} = R^k - [F'(R^k)]^{-1}F(R^k)$$

so we need a notion of the derivative of the F operator, so we resort to the frechet derivative. the frechet derivative, will be an operator $A(R)$ if it is linear and bounded in H, and if it satisfies the following limit:

$$\lim_{\|H\|\to 0} \frac{\|F(R+H) - F(R) - A(R)H\|}{\|H\|} = 0$$

Let's start by making the difference:

$$F(R+H)-F(R) = (R+H)A+A^T(R+H)-(R+H)Q^{-1}B^T(R+H)+P-(RA+A^TR-RBQ^{-1}B^TR+P) \Leftrightarrow$$

$$\Leftrightarrow F(R+H)-F(R) = HA+A^TH-HBQ^{-1}B^TR-RBQ^{-1}B^TH-HBQ^{-1}B^TH \Leftrightarrow$$

$$\Leftrightarrow F(R+H)-F(R) = (A-BQ^{-1}B^TR)^TH+H(A-BQ^{-1}B^TR)-HBQ^{-1}B^TH$$

is it true $A(R)H = (A-BQ^{-1}B^TR)^TH+H(A-BQ^{-1}B^TR)$? We will see:

$$\lim_{\|H\|\to 0}\frac{\|F(R+H)-F(R)-A(R)H\|}{\|H\|} = \lim_{\|H\|\to 0}\frac{\left\|HBQ^{-1}B^TH\right\|}{\|H\|}$$

let's consider the following property:

$$\|T\| = \sup_{u\in X}\frac{\|Tu\|}{\|u\|} \Rightarrow \|AB\| \leq \|A\|\|B\|$$

$$\Leftrightarrow \lim_{\|H\|\to 0}\frac{\left\|HBQ^{-1}B^TH\right\|}{\|H\|} \leq \lim_{\|H\|\to 0}\frac{\|H\|^2\left\|BQ^{-1}B^T\right\|}{\|H\|} = \lim_{\|H\|\to 0}\|H\|\left\|BQ^{-1}B^T\right\| \to 0$$

it's easy to see that $A(R)H$ is linear in H, obviously. We now show that it is bounded (in H), that is $\exists C>0 : \|Tu\| \leq C\|u\|$

$$\|A(R)H\| \leq \left\|(A-BQ^{-1}B^TR)^TH+H(A-BQ^{-1}B^TR)\right\| \leq$$

$$\leq \left\|(A-BQ^{-1}B^TR)^T\right\|\|H\|+\|H\|\left\|A-BQ^{-1}B^TR\right\| \leq C\|H\|$$

then we finally conclude, that Frechet's derivative is given by:

$$A(R)H = (A-BQ^{-1}B^TR)^TH+H(A-BQ^{-1}B^TR)$$

Let's make a change of notation, where we will call $F'(R)H = A(R)H$.

The Newton method, where $[F'(R^k)]^{-1}$ is a functional operator that is operating on $F(R^k)$, this is:

$$R^{k+1} = R^k - [F'(R^k)]^{-1}F(R^k)$$

we don't know what the inverse $[F'(R^k)]^{-1}$ looks like, so we'll have to manipulate the expression a bit until we have something we can work with, considering $\Delta R^k = R^{k+1}-R^k$:

$$R^{k+1} = R^k - [F'(R^k)]^{-1}F(R^k) \Leftrightarrow [F'(R^k)](R^{k+1}-R^k) = -F(R^k) \Leftrightarrow$$

$$\Leftrightarrow F'(R^k)\Delta R^k = -F(R^k)$$

where $F'(R^k)$ is a functional operator that is operating on ΔR^k, this is:

$$F'(R^k)\Delta R^k = -F(R^k) \Leftrightarrow (A-BQ^{-1}B^TR^k)^T\Delta R^k + \Delta R^k(A-BQ^{-1}B^TR^k) = -F(R^k)$$

now we just need to find out for each iteration, and newton's method can be given by:

$$R^{k+1} = R^k + \Delta R^k$$

To solve this problem we resort to kronecker products, as follows:

$$AX + XB = F \Leftrightarrow AXI + IXB = F \Leftrightarrow (I^T \otimes A)x + (B^T \otimes I)x = f \Leftrightarrow$$

$$\Leftrightarrow (I^T \otimes A + B^T \otimes I)x = f \Leftrightarrow Mx = f \Leftrightarrow x = M^{-1}f$$

where $M = I^T \otimes A + B^T \otimes I$, the matrix I is the identity matrix, $x = vec\{X\}$ and $f = vec\{F\}$ are the vectorization of matrices X and F, which is done by stacking the columns on top of each other, in order:

$$x = vec\{X\} = [x_{11}, \ldots, x_{N1}, x_{12}, \ldots, x_{N2}, \ldots, \ldots, x_{1N}, \ldots, x_{NN}]$$

so for our problem in concrete:

$$(A - BQ^{-1}B^T R^k)^T \Delta R^k + \Delta R^k (A - BQ^{-1}B^T R^k) = -F(R^k)$$

for simplicity, we will call $D = BQ^{-1}B^T$ (which is symmetric), $R = R^k$ and the value we have to find $X = \Delta R^k$:

$$\Rightarrow (A - DR)^T X + X(A - DR) = -F(R) \Leftrightarrow (A - DR)^T XI + IX(A - DR) = -F(R) \Leftrightarrow$$

$$\Leftrightarrow (I^T \otimes (A - DR)^T)x + ((A - DR)^T \otimes I)x = -f$$

as the matrices are symmetrical $I^T = I$, $R^T = R$ and $D^T = D$, and the distributive property is valid on kronecker's product:

$$\Leftrightarrow (I \otimes (A^T - RD) + (A^T - RD) \otimes I)x = -f \Leftrightarrow$$

$$\Leftrightarrow (I \otimes A^T - I \otimes RD + A^T \otimes I - RD \otimes I)x = -f$$

solving system by finding x, and putting it back in the form of a matrix $N \times N$, at each iteration. And finally update the array R:

$$R^{k+1} = R^k + \Delta R^k$$

Initially, to start Newton's method, we must have an initial guess for the matrix R^0, in addition to being chosen in order to be symmetric, we still have to verify the stability (in Lyapunov's sense) of the matrix $A - BQ^{-1}B^T R^0$.

No stopping method was used in particular, we just stopped the algorithm when it reached a maximum number of iterations chosen by us, for which we understood that the matrix R had already converged.

After obtaining the matrix R, we solve the following IVP ODE, to discover the state solution $y(t)$, for that we use the RK-4:

$$\begin{cases} \dot{y}(t) = [A - BQ^{-1}B^T R]y(t) \\ y(0) = \xi \end{cases}$$

and finally to take control:

$$u(t) = -Q^{-1}B^T R y(t) = Ky(t)$$

4.3.1 Implementation and results

The strategy used to obtain the first initial approximation of the riccati matrix, R^0, and also to make the $A - BQ^{-1}B^TR^0$ stable in the Lyapunov sense, was automatically generate symmetric random matrices, until you get one that fits. Therefore, when we run our code we always get matrices of different riccati, and consequently different controls and control states, and many times with very good results.

The code is implemented in the file `"exer4_riccati_algebric_with_newton.m"`. One of the simulations made, for the initial condition of $Y = (0, 0, 0, 0)$, we obtained a cost of $J = 0.1678$, whose results are shown in the following images:

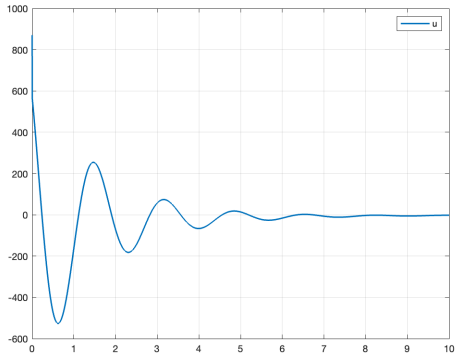


Figure 13: Control $u(t)$

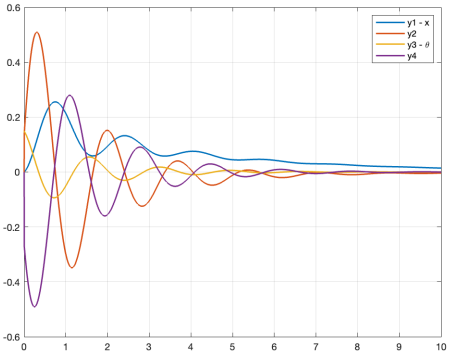


Figure 14: States $y(t)$

5 Exercise 5

Here we want to find an alternative strategy to improve the results obtained in question 3.

5.1 Solving the Riccati equation, with the problem linearized

The method of descent is very slow, also very computationally expensive as it involves a lot of calculations.

To remedy this situation, one of the ideas we had as a possible strategy to improve the results obtained in question 3 involves solving the Riccati equation. The will allow us to have a faster method.

Given the L.Q.R. problem, described in exercise 3, the Differential Riccati equation (D.R.E.) is defined as:

$$\begin{cases} \dot{R}(t) = -R(t)A - A^T R(t) + R(t)BQ^{-1}B^T R(t) - P \\ R(T) = S \end{cases} \tag{23}$$

Assuming that (o^*, y^*) is the only optimal solution of Q.P., characterized by the optimality condition:

$$\nabla J(u^*) = 0 \Leftrightarrow Qu^*(t) + B^T p^*(t) = 0$$

where

$$\begin{cases} \dot{p}(t) = -A^T p(t) - Py(t) \\ P(T) = S \end{cases}$$

then u^* can be given by: $u^*(t) = [A - BQ^{-1}B^T R(t)]y^*(t)$, where $y^*(t)$ is the solution of the linear ODE system:

$$\begin{cases} \dot{y^*}(t) = Ay^*(t) + Bu(t) \\ y^*(0) = \xi \end{cases}$$

There exists a unique symmetric and positive semidefinite solution $R^*(t)$ of the problem (23). Moreover, the minimum cost value is given by:

$$J(u^*) = \frac{1}{2}\xi^T R^*(0)\xi$$

To solve the Riccati equation (23), it could be useful to consider the matrix factorization

$$R(t) = Y(t)X^{-1}(t)$$

Then, $R(t)$ solves the Riccati system (23) if and only if the matrices $X(t)$ and $Y(t)$ solve the following linear ODE system:

$$\begin{cases} \begin{bmatrix} \dot{X}(t) \\ \dot{Y}(t) \end{bmatrix} = \begin{bmatrix} A & -BQ^{-1}B^T \\ -P & -A^T \end{bmatrix} \begin{bmatrix} X(t) \\ Y(t) \end{bmatrix}; \quad t \in [0, T] \\ \text{with } X^{-1}(T)Y(T) = S \end{cases} \tag{24}$$

proof:

Let's start by seeing that from the system we want to prove, we have the following equalities:

$$\dot{X} = AX - BQ^{-1}B^TY$$
$$\dot{Y} = -PX - A^TY$$

let's manipulate the following expression, so that we don't have inverses, and differentiate in time, applying the chain rule:

$$R = YX^{-1} \Leftrightarrow Y = RX$$
$$\Rightarrow \dot{Y} = \dot{R}X + R\dot{X}$$

(I) Of expression $\dot{Y} = \dot{R}X + R\dot{X}$, replace $\dot{R} = -RA - A^TR + RBQ^{-1}B^TR - P$

$$\Rightarrow \dot{Y} = (-RA - A^TR + RBQ^{-1}B^TR - P)X + R\dot{X}$$

now we replace $R = YX^{-1}$:

$$\Rightarrow \dot{Y} = (-YX^{-1}A - A^TYX^{-1} + YX^{-1}BQ^{-1}B^TYX^{-1} - P)X + YX^{-1}\dot{X} \Leftrightarrow$$
$$\Leftrightarrow \dot{Y} = -YX^{-1}AX - A^TY + YX^{-1}BQ^{-1}B^TY - PX + YX^{-1}\dot{X} \Leftrightarrow$$
$$\Leftrightarrow \dot{Y} = (-PX - A^TY) - YX^{-1}AX + YX^{-1}BQ^{-1}B^TY + YX^{-1}\dot{X} \Leftrightarrow$$
$$\Leftrightarrow \dot{Y} = (-PX - A^TY) + YX^{-1}(-AX + BQ^{-1}B^TY + \dot{X}) \Leftrightarrow$$
$$\Leftrightarrow \dot{Y} - \underbrace{(-PX - A^TY)}_{=\dot{Y}} = YX^{-1}(\dot{X} - \underbrace{(AX - BQ^{-1}B^TY)}_{=\dot{X}})$$

(II) Of expression $\dot{Y} = \dot{R}X + R\dot{X}$, replace $\dot{X} = AX - BQ^{-1}B^TY$ and $\dot{Y} = -PX - A^TY$:

$$\Rightarrow -PX - A^TY = \dot{R}X + R(AX - BQ^{-1}B^TY) \Leftrightarrow$$
$$\Leftrightarrow \dot{R}X = (-RA - A^TR + RBQ^{-1}B^TR - P)X$$

Deducing the equation from both sides, we prove (24).

5.1.1 Implementation and Results

We have to notice 3 things to solve the system's ODE:

1st) Problem: in this case we have a FVP-EDO, so we'll have to make some kind of change to resolve it backwards in time, since we are going to implement the RK-4;

$$\begin{cases} y'(t) = f(t, y(t)); \ t \in [0, T] \\ y(T) = \xi \end{cases}$$

Make the following transformation in time: $t(\tau) = T - \tau \Rightarrow \frac{d}{d\tau}t(\tau) = -1$

$$z(\tau) = y(t) = y(T - \tau)$$

$$z'(\tau) = \frac{d}{d\tau}z(\tau) = \frac{dy(t)}{dt} \cdot \frac{dt}{d\tau} = -y'(t) = -f(T-\tau, y(T-\tau)) \Leftrightarrow$$
$$\Leftrightarrow z'(\tau) = -f(T-\tau, z(\tau))$$
$$\begin{cases} z'(\tau) = -f(T-\tau, y(T-\tau)) = -f(T-\tau, z(\tau)); \ \tau \in [0,T] \\ z(0) = \xi \end{cases}$$

the 4th order Runge-Kutta, for IVP-ODE's, iterative method from $k = 1$ till $t = N_t$ (number of discretizations in time), given by the system (forward-time):

$$\begin{cases} F_1 = f(t_k + \tau_1 h, y_k) = f_k \\ F_2 = f(t_k + \tau_2 h, y_k + \alpha_{2,1} h F_1) \\ F_3 = f(t_k + \tau_3 h, y_k + \alpha_{3,1} h F_1 + \alpha_{3,2} h F_2) \\ F_4 = f(t_k + \tau_4 h, y_k + \alpha_{4,1} h F_1 + \alpha_{4,2} h F_2 + \alpha_{4,3} h F_3) \\ y_{k+1} = y_k + h(\beta_1 F1 + \beta_2 F_2 + \beta_3 F_3 + \beta_4 F_4) \end{cases} \tag{25}$$

will now be seen as:

$$\begin{cases} F_1 = -f(T - (t_k + \tau_1 h), z_k) \\ F_2 = -f(T - (t_k + \tau_2 h), z_k + \alpha_{2,1} h F_1) \\ F_3 = -f(T - (t_k + \tau_3 h), z_k + \alpha_{3,1} h F_1 + \alpha_{3,2} h F_2) \\ F_4 = -f(T - (t_k + \tau_4 h), z_k + \alpha_{4,1} h F_1 + \alpha_{4,2} h F_2 + \alpha_{4,3} h F_3) \\ z_{k+1} = z_k + h(\beta_1 F1 + \beta_2 F_2 + \beta_3 F_3 + \beta_4 F_4) \end{cases} \tag{26}$$

so the system will be resolved backwards in time, from $k = 0$ to $k = N_t$ (just like forward-time), except that to be correct we have to "flip" the vector we are iterating, to get the order right, as the first element we calculate corresponds to the end of time.

2st) Problem: The final condition is not very clear, $R(T) =$.To solve this, intuitively we choose $X(T) = I$ and $Y(T) = S$

$$\begin{bmatrix} X(T) \\ Y(T) \end{bmatrix} = \begin{bmatrix} I \\ S \end{bmatrix}$$

3st) Problem: is the dimension of the matrix R, because it is a matrix of dimension $N \times N$ and not a column vector, or a scalar as we are used to, so we will have to come up with a plan to get rid of this problem.

To solve this problem, what we did was to slice the matrix:

$$\begin{bmatrix} X(t) \\ Y(t) \end{bmatrix} = V(t) = [v_1(t), \ldots, v_N(t)]$$

into $v_i(t)$ distinct columns of the ODE system (24), such as columns in the final condition S_i:

$$\begin{bmatrix} X(T) \\ Y(T) \end{bmatrix} = \begin{bmatrix} I \\ S \end{bmatrix} = V(T) = [v_1(T), \ldots, v_N(T)]$$

with $i = 1, \ldots, N$, so we will have N different FVP-EDO's to solve:

$$\begin{cases} v_i(t) = \begin{bmatrix} A & -BQ^{-1}B^T \\ -P & -A^T \end{bmatrix} v_i(t); \;\; t \in [0,T]; \;\; i = 1, \ldots, N \\ v_i(T) \end{cases}$$

and in the end, just append them all, in the right order, and we get the final matrix R (which will actually be a tensor).

After obtaining the matrix R, we solve the following IVP ODE, to discover the state solution $y(t)$, for that we use the RK-4:

$$\begin{cases} \dot{y}(t) = [A - BQ^{-1}B^T R(t)]y(t) \\ y(0) = \xi \end{cases}$$

and finally to take control:

$$u(t) = -Q^{-1}B^T R y(t)$$

5.1.2 Implementation and results

This algorithm was developed in a file `"exer5_riccati_equation.m"`, for the initial condition of $Y = (0, 0, 0.15, 0)$, for which a cost of $J = 8.2063 \times 10^{-4}$, and numerical solutions given in the following images:

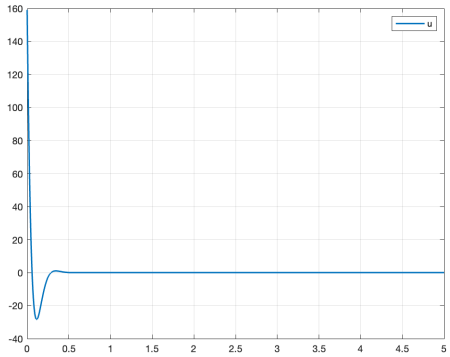


Figure 15: Control $u(t)$

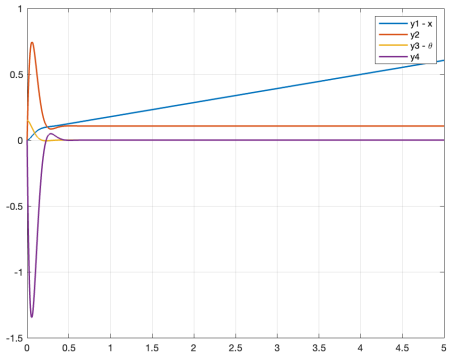


Figure 16: States $y(t)$

5.2 Solution with the non-linearized problem

Another idea we had, to improve our results, was to forget the linearization and apply it in total with the non-linearized model. This choice implies redoing our calculations, as we have to come up with a new expression for the adjunct and a new gradient for our cost.

Where the reduced cost functional is given by the following equation, for a general case:

$$J(u) = \tilde{J}(y(u), u) = \psi(y(T)) + \int_0^T l(y(t), u(t))dt \tag{27}$$

subject to the following dynamics (for the non-linearized model):

$$\begin{cases} \dot{y}(t) = f(y(t), u(t)); \;\; t \in (0, T) \\ y(0) = \xi \end{cases} \tag{28}$$

We have to make an assumption, $y = S(u)$, the solution mapping being $S : C_T \to \mathbb{R}^N$ continuously Frechet-differentiable, where C_T is the set of piecewise continuous functions (P.C.). And f, l and ψ are of class C^1.

Let's introduce the multiplier (adjoint state) $p(t) \in \mathbb{R}^N$,in order to introduce one more zero term in the cost equation:

$$J(u) = \psi(y(T)) + \int_0^T [l(y(t), u(t))]dt = J(u) + \int_0^T \langle f(y(t), u(t)) - \dot{y}(t), p(t)\rangle t \Leftrightarrow$$

$$\Leftrightarrow J(u) = J(u) + \int_0^T p^T(t)[f(u, y) - \dot{y}(t)]\, dt$$

Using integration by parts on the second term, we are left with:

$$\Leftrightarrow J(u) = J(u) + \int_0^T [p^T f + \dot{p}^T y]\, dt - [p^T(t)y(t)]|_0^T \Leftrightarrow$$

$$\Leftrightarrow J(u) = \psi(y(T)) - p(T)^T y(T) + p(0)^T y(0) + \int_0^T [l(y(t), u(t)) + p^T f - \dot{p}^T y]\, dt \Leftrightarrow$$

$$\Leftrightarrow J(u) = \psi(y(T)) - p(T)^T y(T) + p(0)^T y(0) + \int_0^T L(y(u(t)), u(y))\, dt$$

where $L(y(u(t)), u(y)) = l(y(t), u(t)) + p^T(t) f(y, u) + \dot{p}^T(t) y(t)$

Let $v \in C_T$, $\epsilon \neq 0$ and $u_\epsilon = u^* + \epsilon v \in C_T$, and let y_ϵ be the corresponding state (or trajectory) variable. We define the differential of J at u^* along the v direction:

$$DJ(u^*)v = \lim_{\epsilon \to 0} \frac{J(u^* + \epsilon v) - J(u^*)}{\epsilon} = \lim_{\epsilon \to 0} \frac{J(u_\epsilon) - J(u^*)}{\epsilon} \tag{29}$$

$$u_\epsilon = u^* + \epsilon v \Rightarrow y_\epsilon = S(u^* + \epsilon v)$$

now we can apply the chain rule inside the functional (by the properties of the derivative of Frechet and also of Gauteaux):

$$\lim_{\epsilon \to 0} \frac{\psi(y_\epsilon(t)) - \psi(y^*(t))}{\epsilon} = \frac{\partial \psi}{\partial y}\frac{\partial y}{\partial u} + \underbrace{\frac{\partial \psi}{\partial u}}_{=0} = [\nabla_y \psi(t)]^T \nabla_u y(t) v = \langle \nabla_y \psi(t), S'(u) v \rangle$$

$$\lim_{\epsilon \to 0} \frac{p^T(t) y_\epsilon(t) - p^T(t) y^*(t)}{\epsilon} = \lim_{\epsilon \to 0} \frac{p^T(t) y(t)}{\epsilon} = \frac{\partial p^T(t) y(t)}{\partial y}\frac{\partial y}{\partial u} + \underbrace{\frac{\partial g}{\partial u}}_{=0} = p(t) \nabla_u y(t) v = \langle p(t), S'(u) v \rangle$$

Regarding $p^T(0) y(0) = p^T(0)\xi$ as it does not depend on either u or $y = s(u)$ it cancels out with the derivation

$$\lim_{\epsilon \to 0} \frac{L_\epsilon - L^*}{\epsilon} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial u} + \frac{\partial L}{\partial u} =$$

$$= \nabla_y \left\{ l(y(t), u(t)) + p^T(t) f(y, u) + \dot{p}^T(t) y(t) \right\} \nabla_u \{y\} + \nabla_u \left\{ l(y(t), u(t)) + p^T(t) f(y, u) + \dot{p}^T(t) y(t) \right\} =$$

$$= (\nabla_y l(y, u) + [\nabla_y f(y, u)]^T p + \dot{p}) \nabla_u y(t) v + (\nabla_u l(y, u) + [\nabla_u f(y, u)]^T p + 0) v$$

$$\Leftrightarrow DJ(u^*)v = \langle \nabla_y \psi(y(T)) - p(T), S'(u(T)) v \rangle +$$

$$+ \int_0^T [\langle \nabla_y l(y(t), u(t)) + [\nabla_y f(y(t), u(t))]^T p(t) + \dot{p}(t), S'(u(t)) v \rangle] dt +$$

$$+ \int_0^T [\langle \nabla_u l(y(t), u(t)) + [\nabla_u f(y, u)]^T p(t), v(t) \rangle] dt$$

get rid of the terms containing $S'(u(T))$ (the computation of $S'(u(T))$ is expensive) and the expression becomes:

$$DJ(u^*)v = \int_0^T [\langle \nabla_u l(y(t), u(t)) + [\nabla_u f(y(t), u(t))]^T p(t), v(t) \rangle] dt; \ \forall u, v \in C_T \tag{30}$$

we now choose $p = p^*$, where p^* is the solution of th adjoint final-problem (31):

$$\begin{cases} \dot{p^*}(t) = -[\nabla_y f(y(t), u(t))]^T p(t) - \nabla_y l(y(t), u(t)) ; \ t \in (0, T) \\ p(T) = \nabla_y \psi(y(T)) \end{cases} \tag{31}$$

Since formula (30) expresses the linear functional $v \mapsto DJ(u^*)v$ as an inner product in $L^2(0,T)$, we coherently define the gradient of J at u^* by the formula (32):

$$DJ(u^*)v = \langle \nabla J(u^*), v \rangle_{L^2(0,T)} = \int_0^T \langle \nabla J(u^*), v \rangle dtdt$$

$$\Rightarrow \nabla J(u^*) = -[\nabla_y f(y(t), u(t))]^T p(t) - \nabla_y l(y(t), u(t)) \tag{32}$$

5.2.1 Implementation and Results

The implementation here is very similar to the one we did in exercise 3, with the descent algorithm.
start by giving an approximation to the initial control: $u_0(t)$ While the stopping conditions of point (V) have not been satisfied, it will be iterated from k=0 until these conditions are met, repeating points (I), (II), (III), (IV) and (V) sequentially:

(I) Determine the state solution $y_k(t)$, using the RK-4:

$$\begin{cases} \dot{y}_k(t) = f(y_k(t), u_k(t)) \\ y_k(0) = \xi \end{cases}$$

(II) Determine the Adjoint solution $p_k(t)$, using again RK-4 (backward-time):

$$\begin{cases} \dot{p^*}_k(t) = -[\nabla_y f(y_k(t), u_k(t))]^T p_k(t) - \nabla_y l(y_k(t), u_k(t)) \\ p(T) = \nabla_y \psi(y_k(T)) \end{cases}$$

(III) Calculate the new gradient and update the control, with descent method:

$$\begin{cases} \nabla J(u_k) = Qu_k + B^T p_k \\ u_{k+1} = u_k - \tau \nabla J(u_k) \end{cases}$$

(IV) Calculate the Functional Cost:

$$J_k = J(u_k(t)) = \psi(y_k(T)) + \int_0^T [l(y_k(t), u_k(t))] dt$$

Whose integral were approximated by the composite trapezoidal rule with uniform grid

(V) Check the stopping criterion:

$$|J_{k+1} - J_k| < \epsilon$$

where ϵ is a constant (the tolerance).

This algorithm was developed in a file `"exer5_non_linear.m"`, for the initial condition of $Y = (0, 0, 0.15, 0)$, and for the constants $\alpha_1 = \alpha_2 = 1$ and $\alpha_3 = 10^{-6}$, and a tolerance of $\epsilon = 10^{-7}$ for the stopping criterion. with a descent step of $\tau = 10^{-2}$ was used, other values do not seem to be working well, however, compared to exercise 3, although the results are more

reliable and better, the algorithm is really slow, and the larger the interval $[0; T]$, longer will be the waiting time.

A simulation was performed for the time interval $[0; 1]$, for which a cost of $J = 0.0058$, converging in $k = 472$ iterations, and numerical solutions given in the following images:

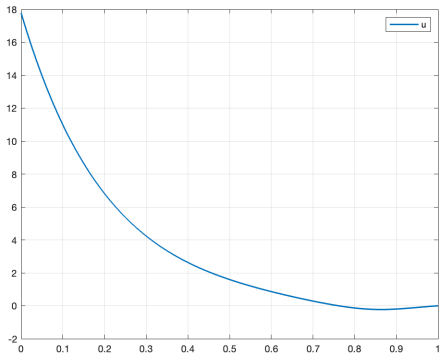


Figure 17: Control $u(t)$

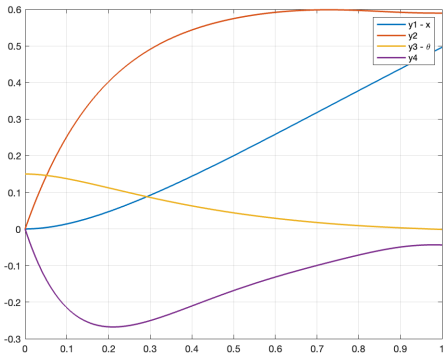


Figure 18: States $y(t)$

References

Apêndices

A 5.2 - Another way to get the gradient

$$J(u) = \psi(y(T)) + \int_0^T [l(u,y)]\,dt + \int_0^T p^T(t)[f(u,y) - \dot{y}(t)]\,dt$$

$$DJ(u^*)v = \psi'(y(T))S'(u(T)) + \int_0^T \frac{\partial l}{\partial y}\frac{\partial y}{\partial u} + \frac{\partial l}{\partial u}\,dt + \int_0^T p^T(t)\left[\frac{\partial f}{\partial y}\frac{\partial y}{\partial u} + \frac{\partial f}{\partial u} - \frac{\partial \dot{y}}{\partial u}\right]\,dt$$

Using integration by parts:

$$\int_0^T -p^T(t)\frac{\partial \dot{y}}{\partial u}\,dt = \int_0^T \dot{p}^T(t)\frac{\partial y}{\partial u}\,dt - \left[p^T(t)\frac{\partial y}{\partial u}\right]_0^T$$

we get:

$$\Rightarrow DJ(u^*)v = \langle \nabla_y \psi(y(T)) - p(T), S'(u(T))v\rangle +$$

$$+ \int_0^T [\langle \nabla_y l(y(t),u(t)) + [\nabla_y f(y(t),u(t))]^T p(t) + \dot{p}(t), S'(u(t))v\rangle]dt +$$

$$+ \int_0^T [\langle \nabla_u l(y(t),u(t)) + [\nabla_u f(y,u)]^T p(t), v(t)\rangle]dt$$