SYST0003-1 - Linear Control System
Project's Final Report

Pierre Hockers, Quentin Lowette, Olivier Moitroux, SirajYounus

Academic year 2018-2019

Table of contents

1 | Motivation and control problem

In a constant evolving world, engineers try to invent new ways of transportation in cities. Some inventions have had few success like the Segways but some more recent one, like the Onewheeler or the Hoverboard try to give a younger spirit to their new way of transportation.

In this work, we will try to design a controller for such unstable machines. The main application is certainly transport but some robots have already been designed in order to give conferences or to serve as guide in offices or museums. Such system can also be used for military application, especially to stabilize missile launchers.

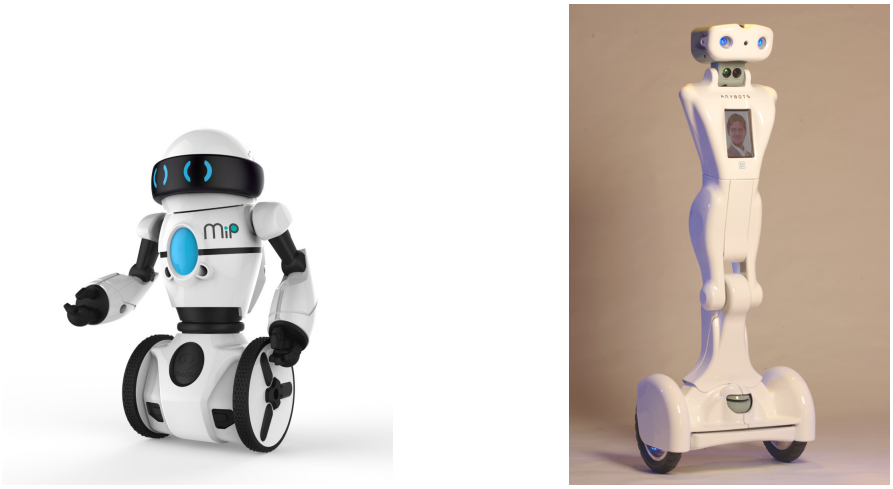

Figure 1.1: Two robots based on an inverted pendulum

1.1 Clarification of the topic

In this work, we present the design and analysis of a motorised inverted pendulum on a cart (with a GoPro camera at the top of the rotating shaft). The goal is to stabilise a vertical platform/frame that has its center of gravity higher than its pivot point, thus creating an unstable system. The micro-controller will have to deal with a several sensors and with the horizontal force applied to the cart in order to maintain the robot steady around a tilt angle as close to zero as possible. We are here focusing on the stabilisation of the pendulum itself, and have no special interest in the horizontal movement of the cart. We still include them in our plot analysis for information only and to analyse variation in the dynamic behaviour of the system when there is a change in the reference.

1.2 Illustrations

The figure 1.2 shows a simplified schematic of the problem. Indeed, most papers modelize the system by a cart with one degree of freedom (translation) with the rotating shaft at its top that adds another degree of freedom (rotation).

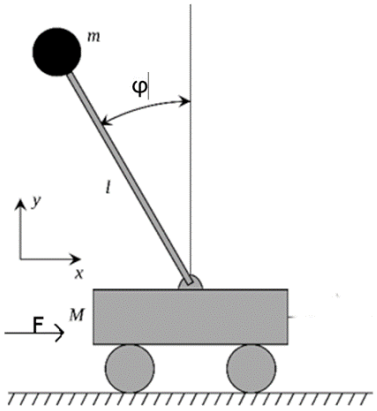

Figure 1.2: Simple representation of an inverted pendulum on cart

1.3 Diagram

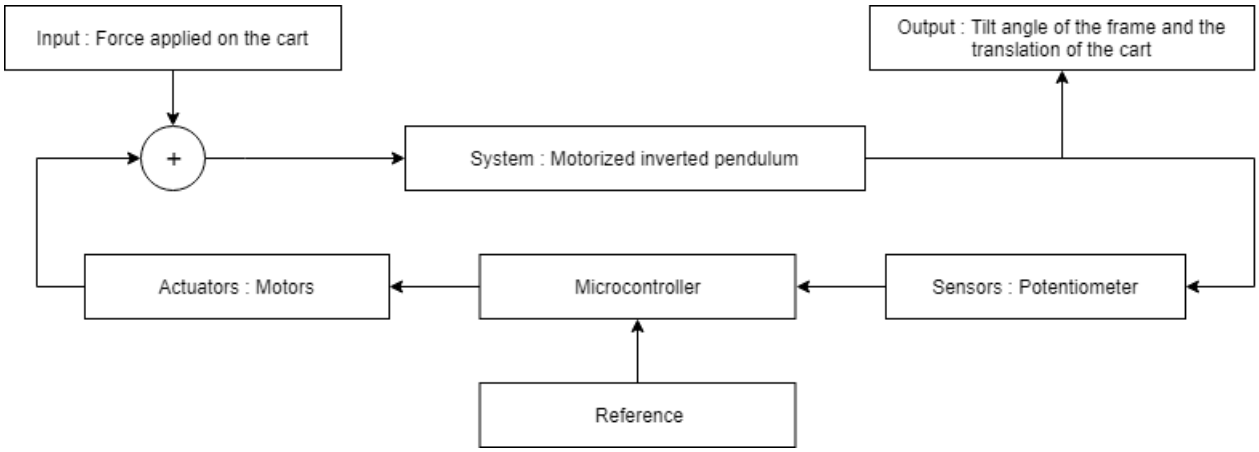

Figure 1.3: Inverted pendulum schematic

1.4 Description of the diagram

- **System to be controlled:** An inverted motorised pendulum with a (GoPro) camera at the top of its head. The position of the head of the pendulum (where the camera is) should be parallel to the ground so that the camera always looks at the horizon.
- **Utility of controller:** Maintain the frame in a perfectly vertical position, thus stabilising the camera.
- **Output:** the angle θ between the shaft and a vertical reference ($\theta = \pi - \varphi$) and the translation x of the cart.
- **Reference:** The angle θ equal to π.
- **Input:** The horizontal force applied to the cart. This force will create a movement in the inverted pendulum used to keep it stable.
- **Sensor:** θ is measured by a potentiometer, and x can be retrieved indirectly by motor feedback (potentiometer).
- **Actuator:** The motors of the cart that will produce the input force.
- **Constraints:** We want to design a relatively "fast" inverted pendulum that won't deviate more than 0.3 radians and that can stabilize itself in around 2.5 seconds or less.
- **State vector:** $[x, \dot{x}, \theta, \dot{\theta}]^T$ where x is the horizontal displacement and θ is the tilt angle of the shaft.

2 | Open loop system

2.1 Schematic of the problem

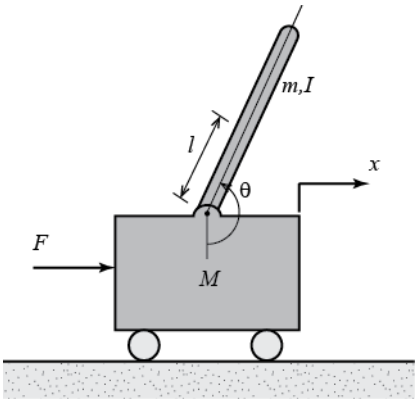

Figure 2.1: 2 degrees of freedom cart with wheels and a pitch-free shaft

2.1.a Variables definition

For our developments, we use the following variables

M	The mass of the cart.
m	The mass of the pendulum.
L_{CM}	The length from the bottom of the shaft to the center of mass of the pendulum (shaft+camera head).
x	The horizontal displacement of the cart.
θ	The angle between the shaft and a fixed vertical reference.
F	The force applied to the cart.
b	The friction coefficient between the ground and the wheels.
I	The moment of inertia of the pendulum.
g	The gravitational acceleration.

2.2 General constraints and assumptions

To design our model, we took into account several constraints:

2.2.a General constraints:

- We give no special importance to the translation x, rotatory movements is the most important thing to compensate.
- There is no slipping between the wheels and the ground
- The maximum recovery tilt angle is set to 0.3 radians $\approx 17°$
- The movement considered is planar and in a straight line (no control system for yaw)
- The robot maintains itself around the vertical position only
- Wheels always stay on the ground.
- The shaft is supposed to be perfectly rigid.

2.2.b Mechanical constraints:

- The robot is in the neighbourhood of the vertical position. It implies that the moment of inertia around the vertical axis can be approximated by a constant.
- In order to calculate the inertia of the pendulum, we assume that the mass is uniformly spread in both parts (the shaft and the camera)
- Inertia of the wheels is neglected
- Damping in the rotation of the shaft is neglected (perfect bearings)

2.2.c Physical properties of the robot

The following table summarises the physical properties of the object :

Symbol	Value	Unit	Description
m_{cam}	0.116	kg	weight of the camera
m_{shaft}	0.1	kg	mass of the shaft
L_{shaft}	0.3	m	length of the shaft
L_{cam}	0.449	m	length of the camera
L_{CM}	0.24	m	length from the bottom of the shaft to the center of mass of the pendulum
m_{pend}	0.216	kg	mass of the total pendulum
I	0.00128	$kg * m^2$	moment of inertia of the pendulum (computed below)
b	0.1	N/m/sec	friction coefficient between the cart and the wheels
g	9.81	$m * s^{-2}$	gravitation constant
M	0.5	kg	mass of the cart

Table 2.1: Physical properties of the entity.

2.3 State space representation

2.3.a Equations of movements

Equation of movement for the cart

The horizontal forces applied on the cart lead to the following equation :

$$F = M\ddot{x} + b\dot{x} + N \tag{2.1}$$

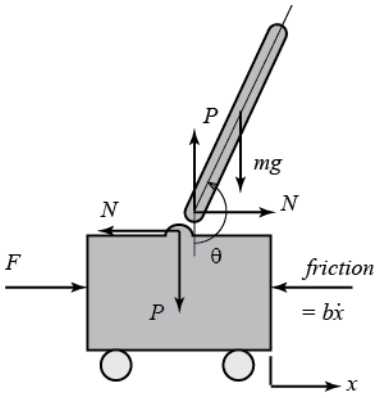

Figure 2.2: Schematic of the pendulum

Where N is the reaction force. By summing the forces in the horizontal direction of the pendulum, we can express N as follows:

$$N = m\ddot{x} + mL_{CM}\ddot{\theta}\cos\theta - mL_{CM}\dot{\theta}^2\sin\theta \tag{2.2}$$

By injecting N in the first equation, we get an equation free of all parasite value, only function of x and θ :

$$F = (M+m)\ddot{x} + b\dot{x} + mL_{CM}\ddot{\theta}\cos\theta - mL_{CM}\dot{\theta}^2\sin\theta \tag{2.3}$$

Equation of movement for the pendulum

Applying the Newton law onto the pendulum (in the vertical direction) yields:

$$P\sin\theta + N\cos\theta = mL_{CM}\ddot{\theta} + m\ddot{x}\cos\theta + mg\sin\theta \tag{2.4}$$

Again, to obtain the equation of movement of the pendulum, we want to eliminate P and N from the above equation. This can be done by summing the moments:

$$(I + mL_{CM}^2)\ddot{\theta} - mglsin\theta = -ml\ddot{x}cos\theta \tag{2.5}$$

Finally, we have our last equation of movement:

$$(I + mL_{CM}^2)\ddot{\theta} + mgL_{CM}\sin\theta = -mL_{CM}\ddot{x}\cos\theta \tag{2.6}$$

Linearization

In this work, we use control design techniques that apply on linear system. Thus, we need to linearize these equations around the vertically upward vertical position ($\varphi = 0$). We will also assume that the system stays in the close neighbourhood of this equilibrium. This assumption seems to be valid with regards to our constraints that we do not want the pendulum to deviate more that 0.3 radians at worse. If we introduce, φ, the deviation of the pendulum from its upward vertical equilibrium, we have

- $\theta = \pi + \varphi$
- $sin(\theta) = cos(\pi + \varphi) \approx -1$
- $cos(\theta) = sin(\pi + \varphi) \approx -\varphi$
- $\dot{\theta}^2 = \dot{\varphi}^2 \approx 0$

The above movement equations, (2.3) and (2.6), finally become:

$$(I + mL_{CM}^2)\ddot{\varphi} - mgL_{CM}\varphi = mL_{CM}\ddot{x} \tag{2.7}$$
$$(M + m)\ddot{x} + b\dot{x} - mL_{CM}\ddot{\varphi} = F \tag{2.8}$$

2.3.b Calculation of the inertia

In the above development, we didn't develop the inertia. In order to make the link between the physical world and the theory, we decided to determine an analytic expression of the inertia so that we could take into account the dimension of the camera and its weight into the code.

Using [3] as reference, we can modelize the inertia of the pendulum this way:

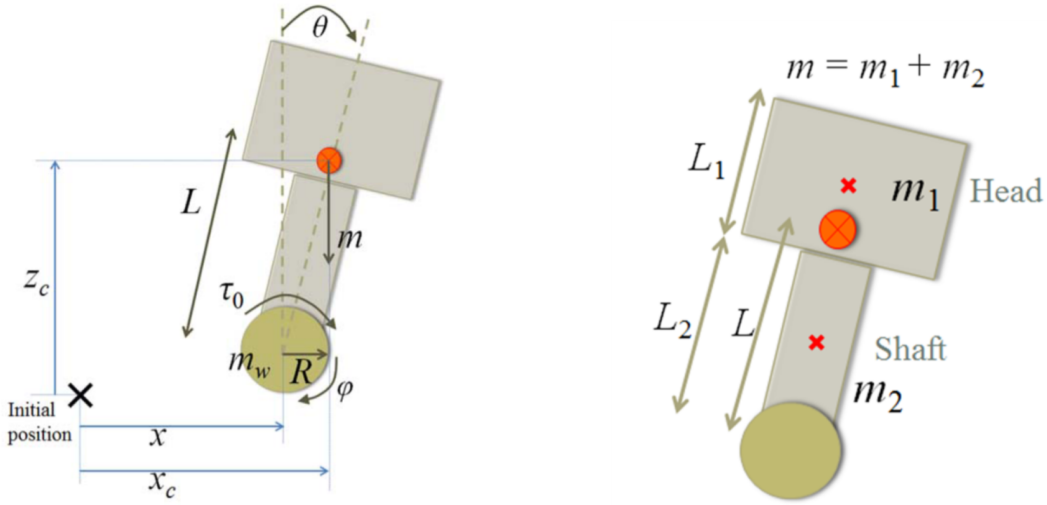

Figure 2.3: Schematic representation of the vertical pendulum sub-system.

This schematic being really close to our model, we will use it to describe our calculus. However, the variables names are slightly different. For clarity purposes, an equivalence table is given in Table 5.1

their variables	our variables
m	m_{pend}
m_1	m_{camera}
m_2	m_{shaft}
L	L_{CM}
L1	L_{camera}
L2	L_{shaft}

Table 2.2: variables names equivalences

Given that the inertia of the robot is the sum of the inertia of the shaft and the head, we can deduce that:

$$I = m_1(\frac{L_1}{2} + L_2)^2 + \frac{1}{12}m_2L_2^2 \tag{2.9}$$

$$I = m_{camera}(\frac{L_{camera}}{2} + L_{shaft})^2 + \frac{1}{12}m_{shaft}L_{shaft}^2 \tag{2.10}$$

These 2 equations are equivalent and represent the inertia of the pendulum in both formalisms.

2.3.c First sketch of the state space representation

Now that we have the modelisation of the inertia and the analytical expressions of the equations of movement, we can isolate $\ddot{x}$ and $\ddot{\theta}$ from (2.7) and (2.8):

$$\ddot{\varphi} = \frac{mL_{CM}}{I + mL_{CM}^2}\ddot{x} + \frac{mgL_{CM}}{I + mL_{CM}^2}\varphi \tag{2.11}$$

$$\ddot{x} = \frac{1}{M+m}F + \frac{mL_{CM}}{M+m}\ddot{\varphi} - \frac{b}{M+m}\dot{x} \tag{2.12}$$

Substituting equations (2.11) in (2.7) and equation (2.12) in (2.8) gives:

$$\ddot{\varphi} = \frac{mgL_{CM}(M+m)}{I(m+M) + MmL_{CM}^2}\varphi + \frac{mL_{CM}b}{I(M+m)MmL_{CM^2}}\dot{x} - \frac{mL_{CM}}{I(M+m) + MmL_{CM}^2}F \tag{2.13}$$

$$\ddot{x} = \frac{-gm^2L_{CM}}{I(M+m) + MmL_{CM}^2}\varphi - \frac{b(I+mL_{CM}^2)}{I(M+m) + MmL_{CM}^2}\dot{x} + \frac{I+mL_{CM}^2}{I(m+M) + MmL_{CM}^2}F \tag{2.14}$$

Let us introduce the input $u_x(t)$ which is the horizontal force F([N]) applied to the cart , the state vector $x(t) = \begin{bmatrix} x(t) & \dot{x}(t) & \varphi(t) & \dot{\varphi}(t) \end{bmatrix}^T$, as well as the output vector $y(t) = \begin{bmatrix} x(t) & \varphi(t) & \end{bmatrix}^T$.
All in all, we obtain the full state space representation:

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu_x(t) \\ y(t) = Cx(t) \end{cases} \tag{2.15}$$

where

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & A_{22} & A_{23} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & A_{42} & A_{43} & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ B_2 \\ 0 \\ B_4 \end{bmatrix}, C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

with $A_{22} = \frac{-(I+m_{pend}L_{CM}^2)b}{I(M+m_{pend})+Mm_{pend}L_{CM}^2}$; $A_{23} = \frac{m_{pend}^2GL_{CM}^2}{I(M+m_{pend})+Mm_{pend}L_{CM}^2}$; $A_{42} = \frac{-m_{pend}L_{CM}b}{I(M+m_{pend})+Mm_{pend}L_{CM}^2}$;
$A_{43} = \frac{m_{pend}GL_{CM}(M+m_{pend})}{I(M+m_{pend})+Mm_{pend}L_{CM}^2}$; $B_2 = \frac{I+m_{pend}L_{CM}^2}{I(M+m_{pend})+Mm_{pend}L_{CM}^2}$; $B_4 = \frac{m_{pend}L_{CM}}{I(M+m_{pend})+Mm_{pend}L_{CM}^2}$.

The C matrix of the full state space is such that the translation x and the angle *theta* are measured. Checking the observability revealed that the system is indeed observable. A more in-depth review of the observability is available in the dedicated section (2.6). We also noticed that observing only θ was not enough to estimate all variables in time but x could be enough. This make quite some sense because none of our developed equations depends on x (see section 2.3.a), and thus, a sensor is needed to measure its state. The system is simply translation-invariant. We also considered a reduced observer in the next section.

2.3.d Reduced state space for pitch only

In order to focus on the control of the tilt (pitch) of the camera, it can be useful to extract from the full state space representation only the elements that influences φ. We don't care of stabilizing the system in a specific x as long as the pendulum is stabilized. In other words, we only care of 3 of the variables of the original state vector. In this way, let us reduce matrices A and B in order to take into account the last remark:

$$A = \begin{bmatrix} \frac{-(I+m_{pend}L_{CM}^2)b}{I(M+m_{pend})+Mm_{pend}L_{CM}^2} & \frac{m_{pend}^2GL_{CM}^2}{I(M+m_{pend})+Mm_{pend}L_{CM}^2} & 0 \\ 0 & 0 & 1 \\ \frac{-m_{pend}L_{CM}b}{I(M+m_{pend})+Mm_{pend}L_{CM}^2} & \frac{m_{pend}GL_{CM}(M+m_{pend})}{I(M+m_{pend})+Mm_{pend}L_{CM}^2} & 0 \end{bmatrix} \tag{2.16}$$

$$B = \begin{bmatrix} \frac{I+m_{pend}L_{CM}^2}{I(M+m_{pend})+Mm_{pend}L_{CM}^2} \\ 0 \\ \frac{m_{pend}L_{CM}}{I(M+m_{pend})+Mm_{pend}L_{CM}^2} \end{bmatrix} \tag{2.17}$$

These new matrices represent the lower-right 3x3 and lower 3x1 matrix of the original matrix A and B respectively. This matrix reduction was possible because x doesn't have any impact on the equation of movements of our system which is translation invariant. If more time was available, we would have considered a reduced order state observer.

2.4 Transfer Function

The transfer function is a link between only one input and one output. Therefore, we have to isolate the two output variables in order to find their respective transfer function.

The Laplace transform for the equation (2.7) and (2.8) are :

$$(I + mL_{CM}^2)\varphi(s)s^2 - mgL_{CM}\varphi s = mL_{CM}X(s)s^2 \tag{2.18}$$

$$(M + m)X(s)s^2 + bX(s)s - mL_{CM}\varphi(s)s^2 = F(s) \tag{2.19}$$

The equation (2.18) gives us

$$X(s) = \left(\frac{I + mL_{CM}^2}{mL_{CM}} - \frac{g}{s^2}\right)\varphi(s) \tag{2.20}$$

after substituting in the the equation (2.19) and simplifying, we get the transfer function :

$$\frac{\varphi(s)}{F(s)} = \frac{\frac{mL_{CM}}{q}s}{s^3 + \frac{b(I+mL_{CM}^2)}{q}s^2 - \frac{(M+m)mgL_{CM}}{q}s - \frac{bmgL_{CM}}{q}} \tag{2.21}$$

with

$q = \big((M + m)(I + mL_{CM}) - (mL_{CM}^2)\big)$

Similarly, we get :

$$\frac{X(s)}{F(s)} = \frac{\frac{(I+mL_{CM}^2)^2 - gmL_{CM}}{q}}{s^4 + \frac{b(I+mL_{CM}^2)}{q}s^3 - \frac{(M+m)mgL_{CM}}{q}s^2 - \frac{bmgL_{CM}}{q}s} \tag{2.22}$$

We can also see if the poles are well on the right side of the plan and therefore if the system is (as strongly expected) unstable. It turns out that the system is indeed observable and controllable and that one pole is in the right-side of the complex plan as can be observed on Figure 2.4. The following graph has been generated with the function `plot_poles` that uses the built-in function `pzplot()`.

The eigenvalues of A are $Eig = [0 \ -0.1396 \ -4.8787 \ 4.8540]$ and the transfer function can be computed automatically in Matlab with `tf` which yields for φ:

$$\frac{\varphi(s)}{F(s)} = \frac{3.374s + 5.702 \times 10^{-18}}{s^3 + 0.1644s^2 - 23.68s - 3.307} \tag{2.23}$$

The independent term at the numerator appears due to numerical round-off and the epsilon machine and can thus be ignored. Indeed, by encoding the the transfer function manually in Matlab and by using the `zpk`, we finally obtain:

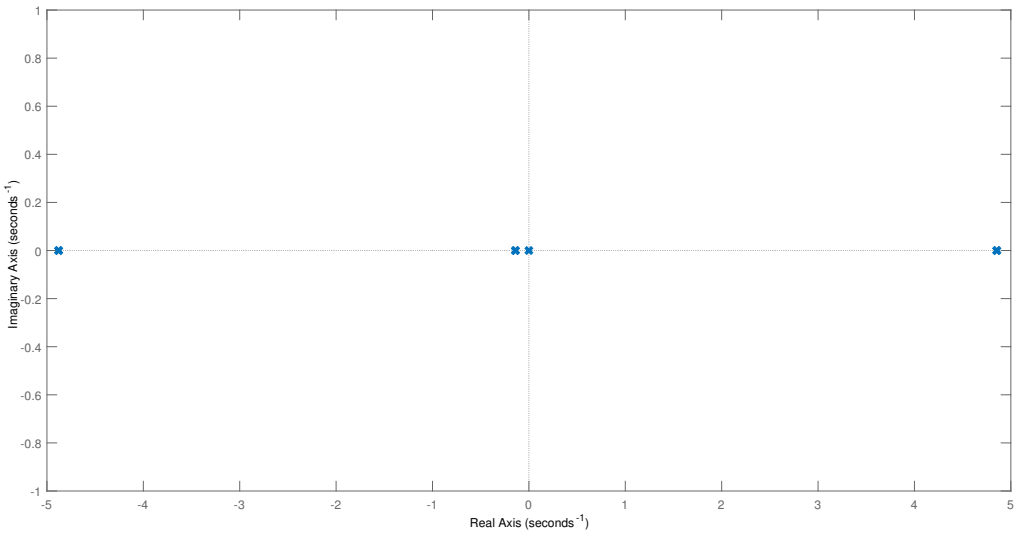

Figure 2.4: Plot of poles and the zero in the s-plane

$$\frac{\varphi(s)}{F(s)} = \frac{3.3745s}{(s-4.854)(s+4.879)(s+0.1396)} \tag{2.24}$$

which, again, highlights three poles and a zero like found and plot previously.

2.5 Simulation of the open-loop system

Systems in which the output has no effect on the controller action are called open-loop system. The response to a step input in open-loop is an unstable system. Figure 2.5 shows how the system react without any controller when subject to an horizontal force disturbance. Unsurprisingly, the system is not stable. In order to ensure the system to be at the equilibrium, one needs to introduce regulators.

The open loop response is generated by using the MatLab function `lsim` (Fig. 2.5). Plotting a step and an impulse response can also be done easily in Maltab by using the functions `step` and `impulse` (Fig. 2.6). We also searched for a way to represent graphically the cart and the pendulum in Matlab in order to better visualize the system. While, the function (credit: Steve Brunton) worked correctly in the open-loop test, it didn't work however for the rest of our code despite some changes. The function relies on parameters that doesn't come from our model and that are very different from us, so it might explain the strange behaviour. Our difference with regards to the inertia is certainly the main reason of problems.

Unsurprisingly, the amplitude of the tilt angle and the cart position grows exponentially and very quickly.

2.6 Observability and controllability

Before designing a regulator, one needs first to determine whether the system is observable and controllable or not. This is easily done by checking that the controllability matrix Wr and the observability matrix Wo are both full rank:

$$W_r = [B \; AB \; \cdots \; A^{n-1}B] \tag{2.25}$$

$$W_0 = [C \; AC \; \cdots \; A^{n-1}C] \tag{2.26}$$

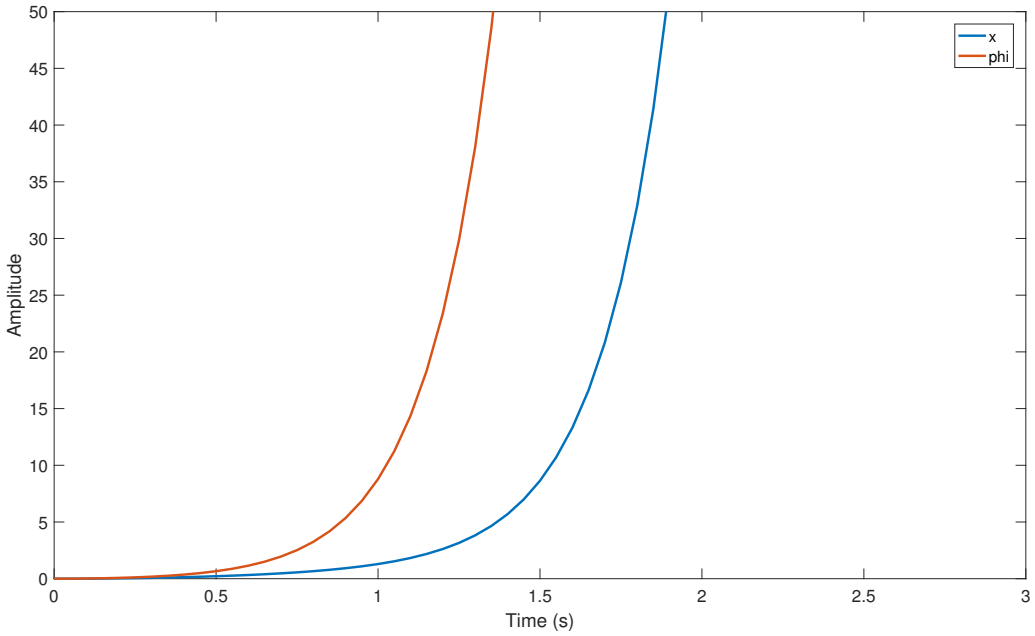

Figure 2.5: Open loop response with an input of 0.2N for 5 seconds.

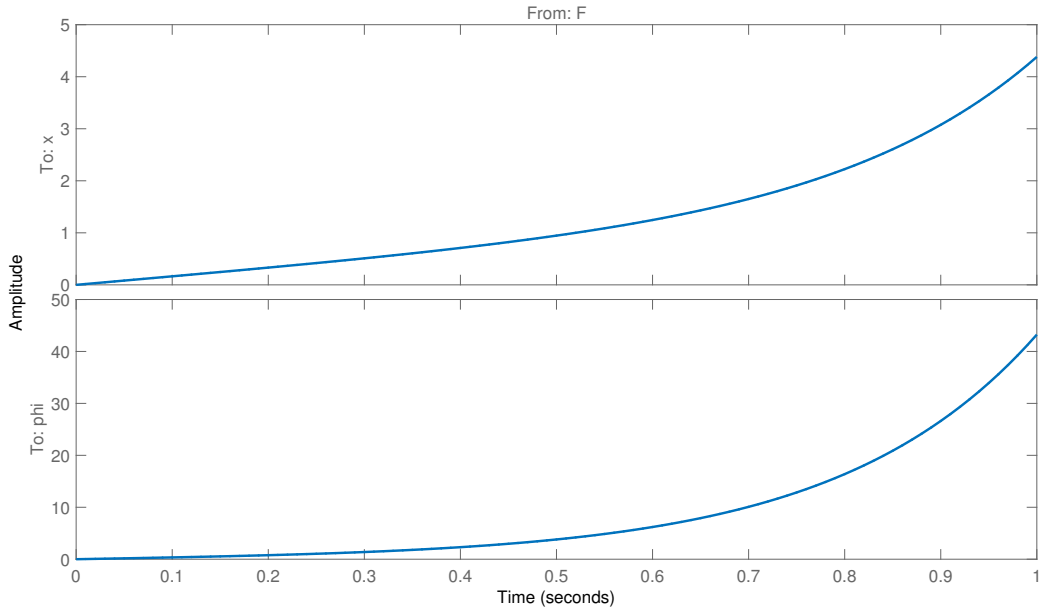

Figure 2.6: Step and impulse response in open-loop.

2.6.a Computation and analysis

To check whether our unstable system is controllable or not, the determinant of the controllability matrix W_r must be computed. However, as the observability matrix is not square, its determinant can't be computed. Instead, we check if the matrix is full rank. We therefore end up with an observable and controllable system

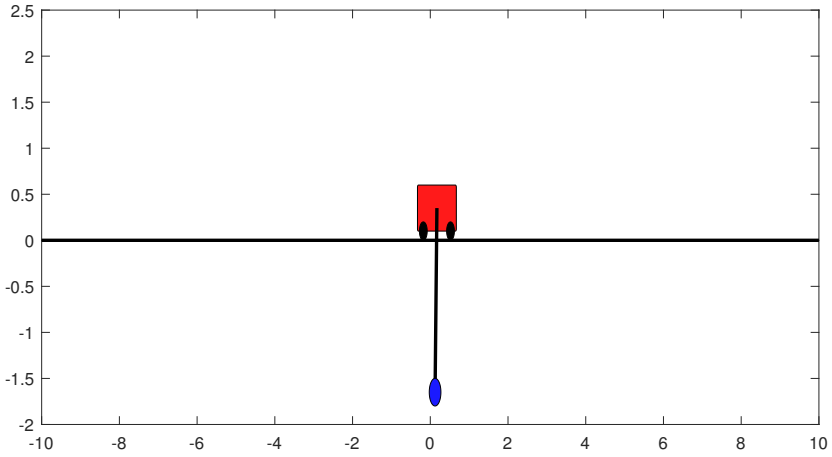

Figure 2.7: Graphical representation of the system when $t \to \infty$

as $det(W_r) = 1.2453e+04 \neq 0$ and $rank(W_o) = 4 = nb_{state\ variables}$.

It should be noted that it is also possible to only consider the reduced expression of the system developed in section 2.3.d. If we consider the full state matrix, the system is not observable if we use a sensor for the angle only. However, it is normally enough to sense the translation x. In the reduced expression, the system is observable whether we measure x, φ, or $\dot{\varphi}$. But the signal to noise ratio may vary depending on the measured state variable. Searching on the internet, we found out that the easiness of observability can be determined by measuring the volume of the observability ellipsoid. To check which variable state yields the best results, we can use `det(gram(sys, 'o'))` in Matlab. We obtained the following results:

	$\dot{x}$	φ	$\dot{\varphi}$
V	0.996	0.137	0.0126

Table 2.3: Volume of the observability ellipsoid. Higher is better.

The conclusion is that, by considering the reduced matrix A, we can observe the system with any of the three above-mentioned state variable but that the x will give us the best results. For this project, we will however stick to the complete state space representation with the measurement of x and φ.

3 | Controller in time domain

3.1 Constraints and simulations specifications

The constraints stay the same as expressed in the previous part of this work.

Constraints

- Maximum acceleration of motors: $5m/s^2$ (600 rpm)
- Maximum deviation of $0.3rad$
- Observer: fast convergence (10 times faster than controller)
- Noise: rejection as much as possible
- Settling time: below 3 seconds

Being in the computer sciences section, we don't need to use real motors and so, didn't look after a specific model. Most stepper motors on the market for 3d-printers turns around 600/800 rpm which is more than enough and a value of $5m/s^2$ seems to be a reasonable margin. The deviation of 0.3 radians is roughly equal to 17°. As we plan to design a fast controller, we expect that the pendulum may deviate a bit more. The settling time of less than 3 seconds is more a challenge as most balancing robot seems to have a larger settling time.

Simulation specifications

- Delays in sensors: $10ms$
- Noise: white noise of power 0.00001
- Load disturbance: 0.2N for 5 seconds (instantaneous kick)
- Reference variation: cart should move 20 cm every 2.2 seconds while still respecting above dynamic constraints

All the applied disturbances ("kicks" in the robot) and changes in reference are steps or square wave due to the nature of the problem.

3.2 State feedback controller

In this part, we use the input $u(t)$ to modify the eigenvalues of A so that we can change the system dynamics.

3.2.a PID controller

In order to make a first rough approach to stabilize our system, we designed in *Simulink* a *PID* controller as can be seen in Figure 3.1. This approach is very convenient due to its easiness of tuning. The corresponding simulation can be observed on figure 3.2. The corresponding vector of gain is $[K_p,\ K_i,\ K_d] = [79.95,\ 125.40,\ 8.24]$.

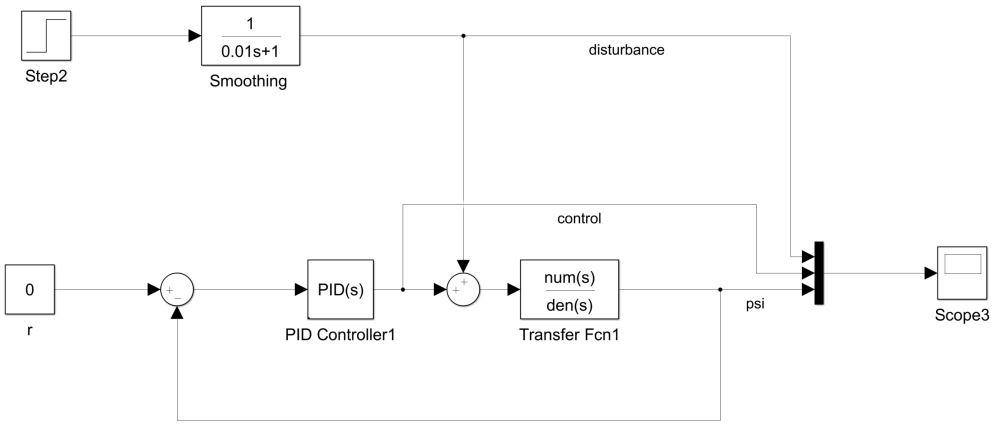

Figure 3.1: PID controller in Simulink

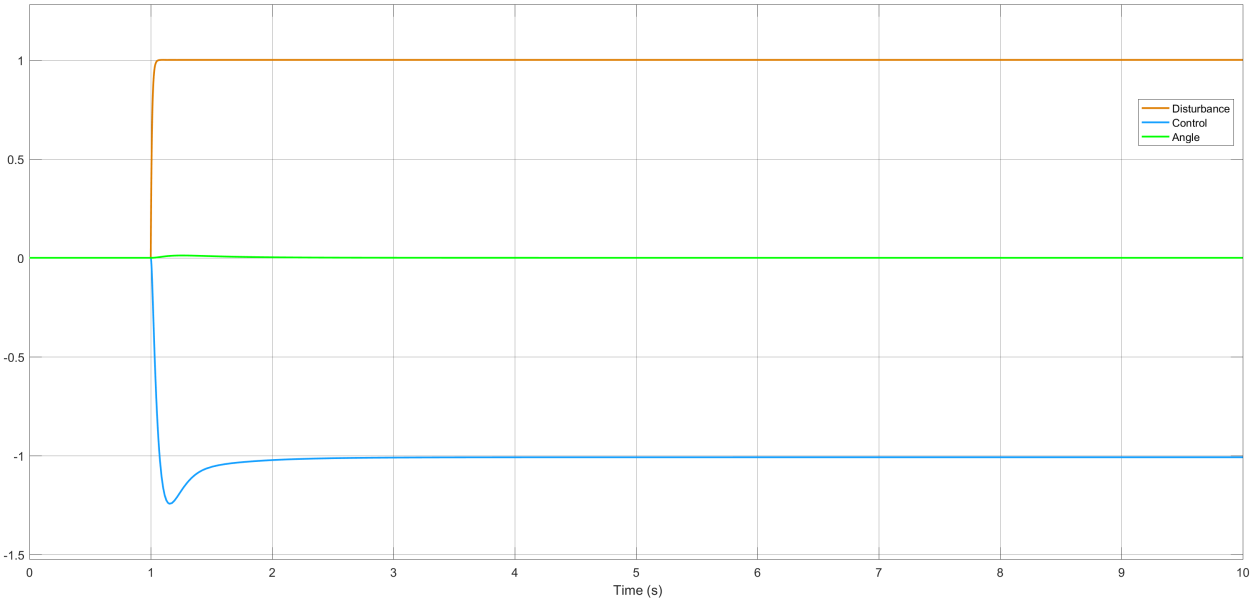

Figure 3.2: Response to disturbance on the system controlled by PID.

3.2.b Pole placement

Tuning the gains

First, we consider that we can measure the value of all states. We can therefore use the proportional state feedback controller (feedback rule):

$$\begin{aligned} u &= -Kx + k_r r \\ &= -k_1 x_1 - k_2 x_2 - k_3 x_3 - k_4 x_4 + k_r r \\ &= -k_1 x_1 - k_2 x_2 - k_3 x_3 - k_4 x_4 \\ &= -[k_1 k_2 k_3 k_4] x(t) \end{aligned} \tag{3.1}$$

where K denotes the gain vector and k_r the gain on the reference. As a reminder, $r = 0$ if we only consider the deviation angle of the shaft.

As we want to track the reference $\varphi = 0$, We have the closed-loop dynamics:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + B(r - Kx(t)) \\ &= (A - BK)x(t) + Br \\ &= A_{cl}x(t) + Br \\ &= A_{cl}x(t) \end{aligned} \tag{3.2}$$

The stability is ensured only if the eigenvalues of the $A - BK$ matrix have a negative real part. Now, we need to pick K so that A_{cl} has the desired properties, e.g.

$$\begin{aligned} A_{cl} &= A - BK \\ &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & A_{22} & A_{23} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & A_{42} & A_{43} & 0 \end{bmatrix} - \begin{bmatrix} 0 \\ B_2 \\ 0 \\ B_4 \end{bmatrix} \begin{bmatrix} k_1 & k_2 & k_3 & k_4 \end{bmatrix} \end{aligned} \tag{3.3}$$

Then,

$$\det(sI - A_{cl}) = f(s) = 0 \tag{3.4}$$

For the desired closed loop system characteristic equation given by:

$$q(\lambda) = (s^2 + 2\zeta\omega_c + \omega_c^2)(s^2 + as + b) \tag{3.5}$$

The pair (ζ, ω_c) are chosen so that the the dominant poles satisfy the required performance and the pair (a,b) are chosen farther in the left half plane.

By tweaking the two parameters ζ and ω_c in the polynomial of the form

$$s^2 + 2\zeta_c\omega_c s + \omega_c^2 = 0 \tag{3.6}$$

, we can tune indirectly our gains by pole placements. Quite arbitrarily we choose to design a controller featuring an overshoot of 8% and a settling time of 0.84 second. Therefore,

$$\begin{cases} \zeta = \frac{len(OS/100)}{\sqrt{\pi^2 + ln(OS/100)}} = 0.627 \\ T_s = 4/(\zeta\omega_c) = 0.84 \Longleftrightarrow \omega_c = 7.59 \end{cases} \tag{3.7}$$

$\zeta = 0.627$ and $\omega_c = 7.59$ can then be used to find the two dominant poles of our system:

$$s_{1,2} = \zeta\omega_c \pm \omega_c\sqrt{(1 - \zeta^2)} = -4.774 \pm 5.9172i \tag{3.8}$$

As a rule of thumb, the third and fourth poles are usually placed between 2 and 10 times deeper (real part) in the s-plane than the dominant poles. All in all, we used the poles $[-4.774 + 5.9172i; -4.774 - 5.9172i; -10.1; -10.2]$ that gave us

$$K = [-180.0737; -65.3276; 199.8242; 40.6161] \tag{3.9}$$

Let's note that the third and fourth poles are not given the same value as the function `place` of Matlab requires unicity in the poles it is given as an input.

Response to a perturbation

The next step is to plot a response to an input of 0.2N for 5 seconds (Fig. 3.3). The deviation is very small and the system stabilizes itself in two seconds which is good with regards to our constraints. Some overshoot persists but it is not a critical flaw at all in our problem statement as we do not desire a rather aggressive controller. By curiosity, we also plot the different states of the system. From Figure 3.4, we can, for instance, observe that the cart stabilize itself as well but is not in x = 0 anymore.

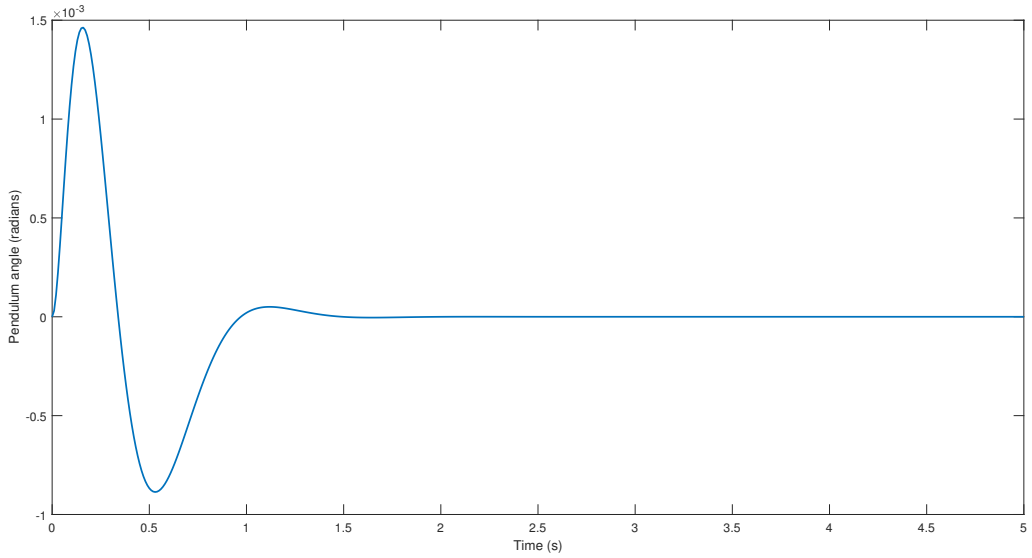

Figure 3.3: Response to a constant input of 0.2N for 5 seconds of controller tuned by pole placement.

The step response and impulse response have also been generated and are available in Figure 3.5.

The instantaneous step response highlights a very small deviation of the angle (0.0732 radians) and a settling time of 1 seconds. The impulse takes more time to be compensated and the deviation of the angle reaches 0.0749 radians.

3.2.c LQR

Tuning the gains

In the first place, we weren't able to use the pole placement technique correctly[1]and used another technique called *LQR* which stands for linear quadritic regulation. We won't enter the details as this technique wasn't reviewed in the couse. In a nutshell, we define a cost function

[1]due to the fact it was not conceivable to derive analytically the characteristic equation and to perform a simple coefficient identification as the polynom was not quadratic but of 4th order.

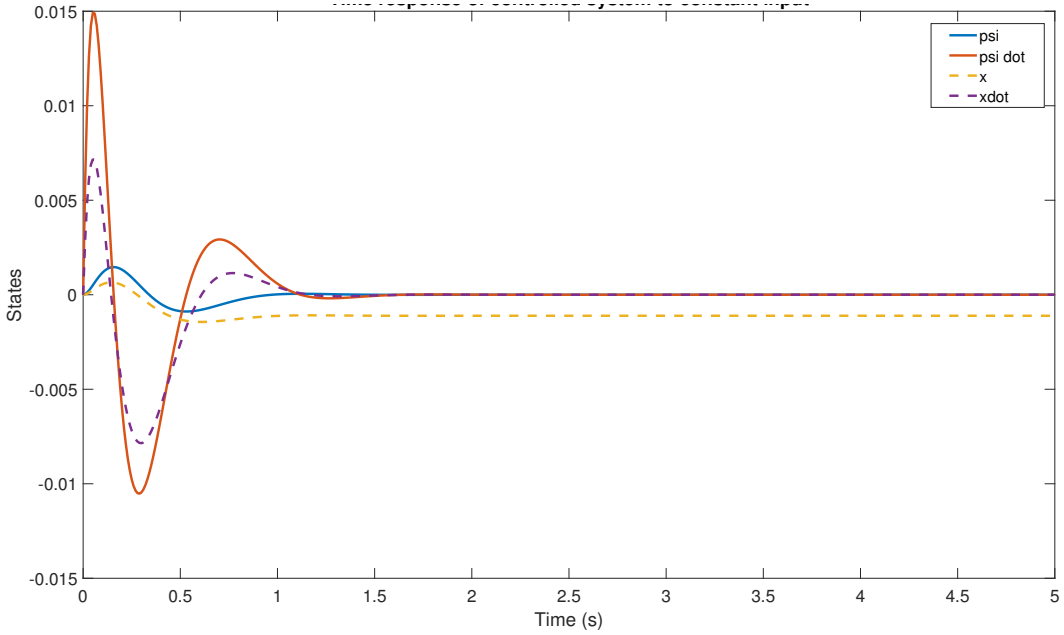

Figure 3.4: Response plot of the states to a constant input of 0.2N for 5 seconds of controller tuned by pole placement.

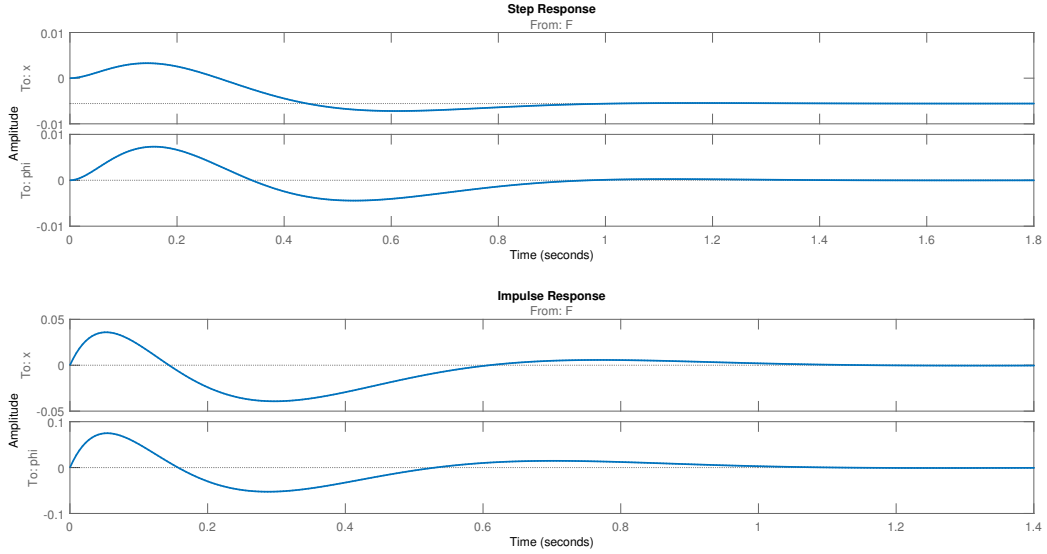

Figure 3.5: Step and impulse response with controller tuned by pole placement.

$$J = \int_0^\infty (x^T Q x + u^T R u) dt \tag{3.10}$$

where Q is a positive semi-definite matrix that can be interpreted as a tuning for how bad the penalty is if the states are not where they should be. R, on the other hand, is a positive definite matrix that can be interpreted as a way of quantifying the cost of the control input (electricity, ...). We first tried with

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 100 & 0 \end{bmatrix} \tag{3.11}$$

and R = 0.001 in order to highly penalize a wrong value of φ or a non-zero angular speed. However, we didn't get good results. We then considered a simpler case with $Q = C' \times C$ to get a sparse matrix Q with the only elements different from zero being the one corresponding to the translation x and the angle φ. It turns out that

$$Q = C^T C = \begin{bmatrix} 5400 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 800 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.12}$$

worked the best. These values were found by making an exhaustive search thanks to a Matlab script. By then using the `lqr` function of Maltab, and plotting the response, we had acceptable performance for a slower and less-demanding system. Indeed, the gain are smaller than the ones obtained via pole placement and, thus may offer an alternative if motors are less performant. The non-zero of our matrix Q represent the weight on the cart's position and the weight on the pendulum's angle. Let's pay attention to the fact that the higher these values, the higher the actuation signal.

Response to a perturbation

The given responses can be observed on Figures 3.6 and 3.7.

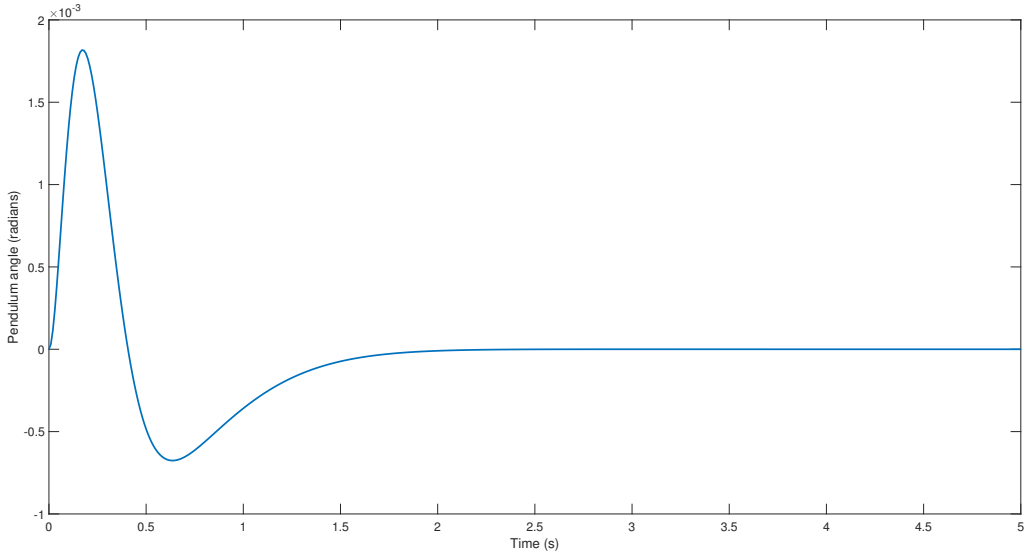

Figure 3.6: Response of the controlled system with LQR for a constant input of 0.2N for 5 seconds.

When observing the response plots, it is unsurprising to see that the gains are lower than the one obtained by pole placement.

$$K_{lqr} = [-73.4847; -43.6178; 133.2959; 28.7986] \tag{3.13}$$

Again unsurprising, gains are higher on the deviation angle and on the angular velocity. The settling time is around the same for LQR but there seems to persist an extremely small slope around 2 seconds. With regards to the step and the impulse response, it is slower to compensate and, as a consequence, the deviation

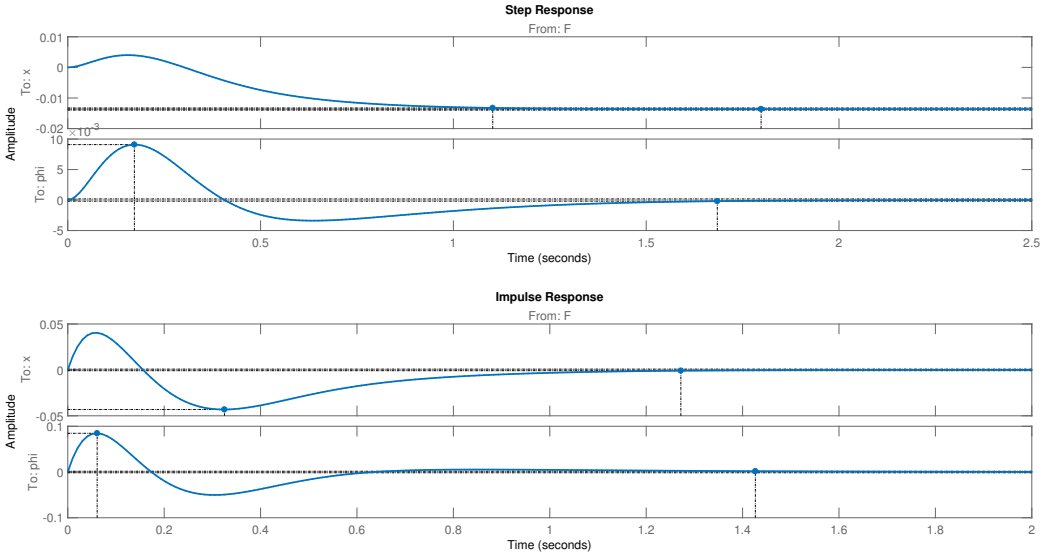

Figure 3.7: Impulse and step response of the controlled system with LQR.

of the angle is larger than the system tuned by pole placement. However, the overshoot is less pronounced proportionally. The peak response of the angle reaches 0.009 radians and 0.0846 radians when the system is applied a step and an impulse respectively.

3.3 Simulink of the state feedback controller (without observer)

All the graphs displayed before were generated in Matlab. However, we also built a Simulink model that can be partially observed on Figure 3.8. The Simulink file is called `closed_loop.slx` and contains two closed-loop models: one without the observer and another one with the observer.

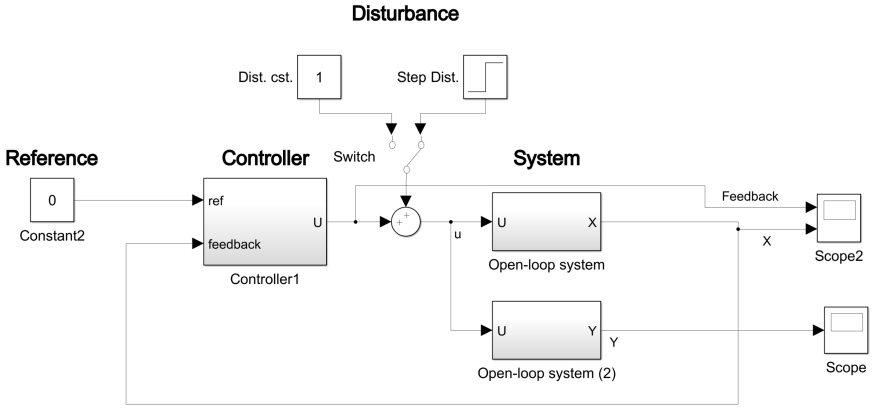

Figure 3.8: State feedback controller in Simulink

The Simulink model can be used to generate the same plot as below. If needed, the feedback signal can also be plot alongside the states or the output easily.

In our first attempt, we model the cart-pendulum system in a more sophisticated way and we model the dc motors as well in order to use a voltage instead of a force as an input. While all this work had to be discarded

during the semester, we had still implemented a Simulink model in the old fashion way by using sums and integrators. Because this step is really tire-some and we proved we could translate equations in the software, this time, we simply used the state space module. A little trick was then used to retrieve the state variables: we used another state space module with our matrices A and B but with a matrix C which is a 4x4 identity matrix. The output of this module gives us the state variables.

3.4 Response to a reference variation

In the assignment, we were asked to analyze the response of the system to a reference variation. However, we thought that changing the reference on the angle wouldn't make much sense (the cart would run forever). For this reason, we develop in this section a way to set the reference of the cart position. That way, we will also be able to analyse how the system (both cart and pendulum this time) reacts to a plausible order in real life. That means that we have to address the steady state error on the cart position. Indeed, Figure 3.4 highlights a little problem with the cart: it does not return to its initial position. In order to proper analyze how the system reacts to a change in reference, we must first address the steady state error by adding a gain on the reference, a.k.a. a precompensator.

By using a gain

$$k_r = \frac{-1}{C(A-BK)^{-}1B} \tag{3.14}$$

, we can reduce the steady state error on the translation.

Sudden change of reference

We then considered the closed loop system, **tuned with pole placement**, added the precompensator and plot the response (Fig. 3.9).

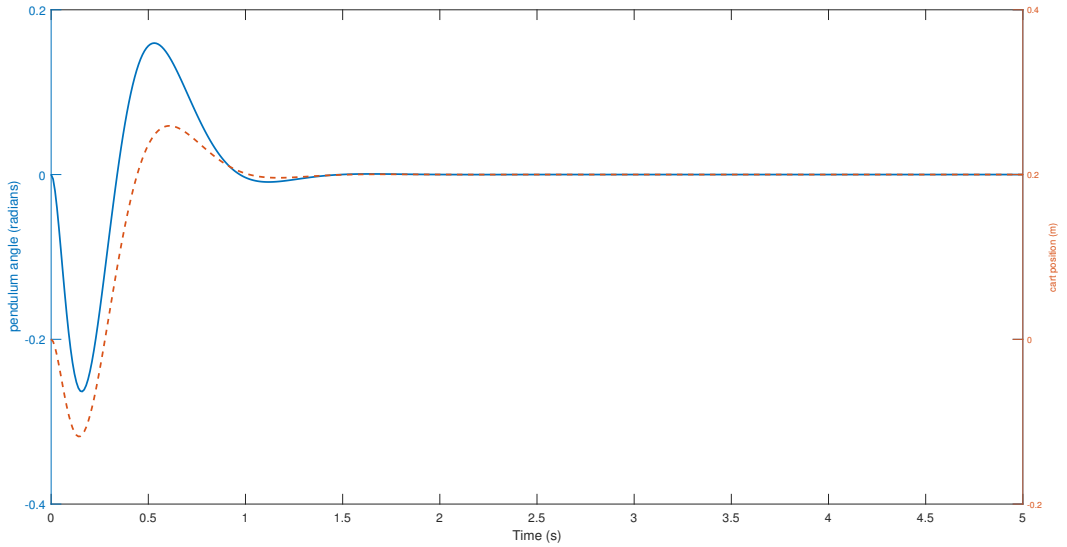

Figure 3.9: Response of the system when the reference on the cart is an impulse of 5 seconds. The cart is asked to move 20 cm.

The Figure 3.9 has been generated by using `lsim` with an abrupt change in the reference of the cart. Actually, the cart was asked to move 20 cm (the impulse lasting 5 seconds as always). On the left, the blue y-axis

represent the scale for the deviation angle. The orange axis[2], on the right, represents the cart position. We observe that the angle started at position 0 and reached its position back after approximately 1.7 seconds. The cart, on the other hand, started from position 0 and reached its final position, $x = 0.2m$ in the same amount of time. The deviation of the angle is much more pronounced this time. Considering the fact we ask the system to move 20 cm in a small amount of time, it is however expected to be quite aggressive. Still, the angle is below our stated constrain of 0.3 radians and the controller stabilizes the system in less than the 3 seconds allowed.

As a bonus, we also put the same plot as Figure 3.9 for the controller tuned with lqr. This figure is available at page 41). This tuning of the controller leads to a deviation of only -0.1335 radian but a larger settling time of almost 2.5 seconds.

Square wave change of reference

The natural extension of our test was to apply a square wave to the system to see whether it was able to handle frequent change in the reference. It would not be hard to do the same simulation with a sine wave but this kind of input is not very relevant in our case.

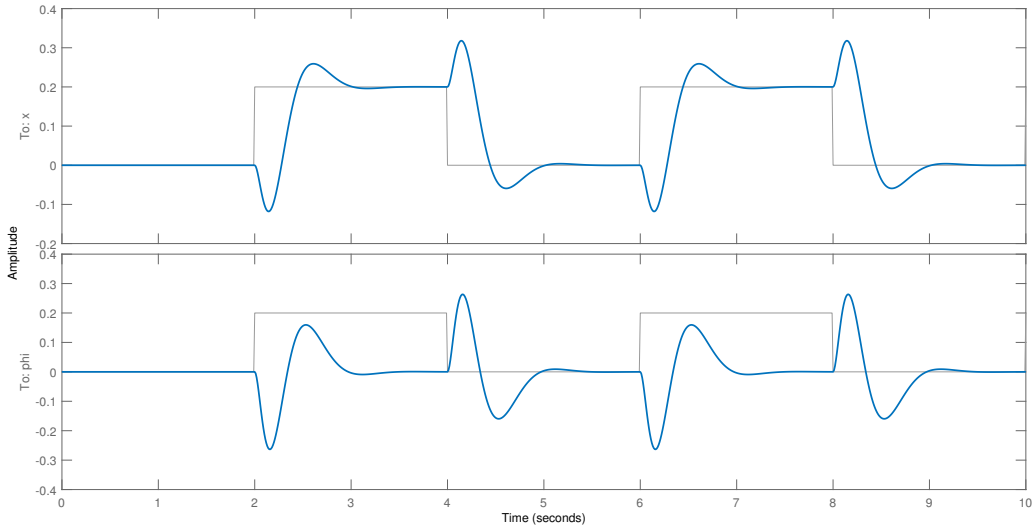

Figure 3.10: Square wave variation of the reference of the closed-loop system (tuned by **pole placement**). The cart is asked to alternate between $x = 0$ and $x = 20$ cm every 2.2s.

Figure 3.10 is a very important plot to visually interpret how the system behaves. The cart, in order to move itself to the goal (x=0.2m), can't simply go in the right direction but needs first to orient the pendulum correctly by going first in the wrong direction. Then, it needs to catch it up. Because, we designed the system to be fast, it needs to overshoot a little bit and go beyond the reference to stabilize back the pendulum. We let the reference to a given value for 2.2 seconds which was enough for the controller to stabilize the system. The deviation of the angle is still under our constraint of 0.3 radians.

we also simulate the change in reference onto the closed-loop system tuned with the less aggressive gains obtained with the LQR method. As expected, the response is a smoother, have less overshoot, but two seconds is really needed to stabilize the pendulum. There is, actually, no overshoot at all for the cart position

[2]Unfortunately, we didn't found a way in Matlab to increase the font size of the axis

and the deviation on the angle only reaches -0.134 radians. All in all, while the response with pole placement is already satisfying, the feedback controller might be a better alternative as cheaper motors can be used.

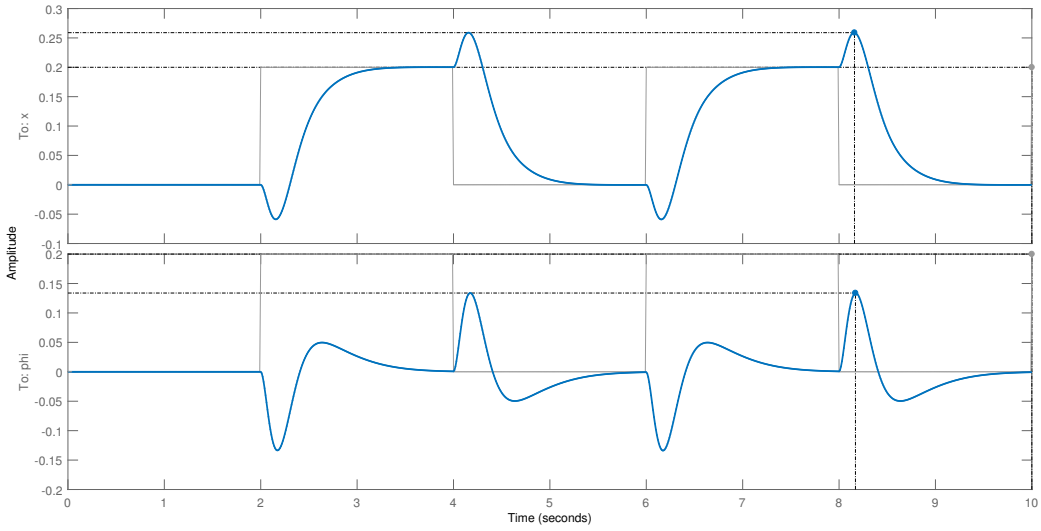

Figure 3.11: Square wave variation of the reference of the closed-loop system (tuned by **lqr**). The cart is asked to alternate between $x = 0$ and $x = 20$ cm every 2.2s.

3.4.a Checking motor saturation

We wanted to limit the acceleration asked to the motors to avoid their saturation. We know that

$$d = \frac{1}{2}at^2 \rightarrow vt = \frac{1}{2}at^2 \tag{3.15}$$

Moreover, we can roughly estimate (no need to be especially precise here) that the motor has to move the cart 0.4m in 0.8 seconds if we look at the steeper curve portion in figure 3.10. That means:

$$\begin{cases} d = 0.4m \\ t = 0.8s \\ d = \frac{1}{2}at^2 \Leftrightarrow a = 1.42m/s^2 \end{cases} \tag{3.16}$$

We can thus consider that we are safe with regards saturation of the motors even with the controller tuned by pole placement.

3.5 Observer

3.5.a Design of the observer

Let's now be more realistic and consider that we don't have access to the true states directly. We now need to estimate them. The observer is a dynamical system characterised by:

$$\frac{d\hat{x}}{dt} = A\hat{x} + Bu \tag{3.17}$$

where $\hat{x}$ denotes the estimation of the state variables. In order to make it faster that the system, we use the estimated output and compare it to the measured output:

$$\frac{d\hat{x}}{dt} = A\hat{x} + Bu + L(y - \hat{y}) \tag{3.18}$$

$$= A\hat{x} + Bu + L(y - C\hat{x}) \tag{3.19}$$

where the new term represent the correction. The estimate error being given by $\frac{de}{dt} = (A - LC)e$ ($e \equiv \tilde{x}$ in the course), we now need to determine the observer gain L thanks to the pole placement technique. Indeed, the error will approach zero ($\hat{x}$ will approach x) if the matrix $(A - LC)e(t)$ is stable. The eigenvalues of $(A - LC)$ will determine the speed of converge. A major guidelines is to design the observer so that it is 10 times faster than the controller but making the estimator too fast can be problematic if the measurement is corrupter by noise. Increasing the observer gain is not a problem with regards to saturation here but there is a trade-off between high-bandwidth observers and low bandwidth ones (less noise sensitive but slower).

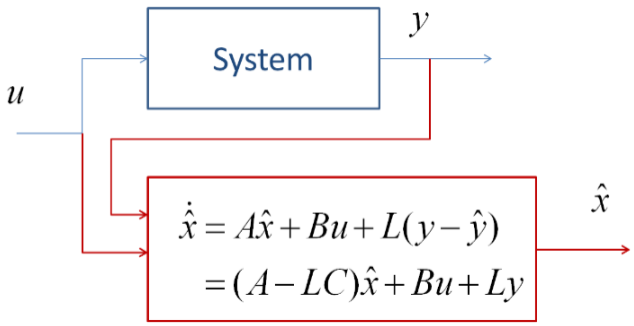

Figure 3.12: State estimation with observer

3.5.b Finding L by scaling the gains of the controller

The poles of our matrix $A_c = A - BK$ are

$$eig(A_c) = \begin{bmatrix} -4.7740 + 5.9172i \\ -4.7740 - 5.9172i \\ -10.1000 + 0.0000i \\ -10.2000 + 0.0000i \end{bmatrix} \tag{3.20}$$

The slowest pole turns out to have a real part equal to -4.7740. Therefore, we can place our new poles at [-47.74; -48.74; -49.74; -50.74]. The fact that the poles are different is just motivated by the implementation of the Matlab function `place` that does not accept redundant values. This yields to the following gains on the state variables:

$$L = 1000. \begin{bmatrix} 0.0982 & -0.0011 \\ 2.4041 & -0.0519 \\ -0.0010 & 0.0986 \\ -0.0556 & 2.4366 \end{bmatrix} \tag{3.21}$$

We use the development done in [7] in order to analyse the system composed of the controller and the observer:

$$\begin{cases} \begin{bmatrix} \frac{dx}{dt} \\ \frac{de}{dt} \end{bmatrix} = \begin{bmatrix} A - BK & BK \\ 0 & A - LC \end{bmatrix} \begin{bmatrix} x \\ \hat{x} \end{bmatrix} + \begin{bmatrix} B \\ O \end{bmatrix} r \\ y = \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} x \\ e \end{bmatrix} \end{cases} \tag{3.22}$$

, where $e = x - \hat{x}$.

The simulation of the response is presented at section 3.5.d. Let's note that we can't use the exact same initial conditions on the estimated states. $x(0)$ is unknown so we cannot choose $\hat{x}(0) = x(0)$.

3.5.c Finding L by regular analytic expression and pole placement

By developing analytically equation (3.19), we can obtain an expression for $\frac{d\hat{x}_i}{dt}$, $\forall i = \{1, 2, 3, 4\}$ that can then be implemented in Simulink with regular integrator, gains and sums blocks. However, developing a 4×4 matrix is always a mess, that's why we stick to a more general observer in Simulink as can be seen in Figure 3.15. By applying the same technique discussed for the controller in section 3.2, we can calculate the characteristic polynomial:

$$det(sI - (A - LC)) \tag{3.23}$$

which would give a polynomial in s. In a 2×2 matrix, making a simple coefficient identification with $s^2 + 2\zeta_o\omega_0 s + \omega_0^2$. would give us a system to solve for l_1 and l_2 given the chosen parameters ζ_0 and ω_0^2. Again, our characteristic polynomial being of the 4^th order, we can first place the two dominant poles as usual and push further the two other poles.

In order to design the observer, we implemented a script to compute several gains L based on different values of η_o and ω_o. A good practise is to consider $\omega_0 = 10\omega_c$, $\zeta_0 = \zeta_c$. This way, we considered an exhaustive search among

$$\zeta_o = [0.25, 0.5, 0.75, 1] \tag{3.24}$$

$$\omega_o = [45, 100, 450] \tag{3.25}$$

The intended goal was to plot the true states and the estimated states of the observer tuned with different parameters on the same plot. Then, we would have chosen the curve that best-matches our needs with regards to the convergence. However, we never succeeded to build such plot despite having tried for a long time. Our first attempt used the system developed in equation 3.22 to plot the evolution of the estimated angle without a success (infinite derivative). For this, we implemented a function called `multi_plot_tune_observer.m`. Other similar developments can be done and are available in the appendix at page 41. Implementing all of them didn't help either. To be sure we were dealing correctly with matrices, we finally opted for another approach by calling in a loop a Simulink model called `observer_tuning.slx` that added some noise and delay. The code that performs this tasks is `multi_plot_tune_simulink_observer.m`. The results, available in the appendix, were rougly the same and so, not conclusive. If no noise is applied, but only a change in the reference and a delay, the figure 6.2 and 6.3 are obtained and all the lines of the estimated states corresponding to different tuning are merged. These plots are available in the appendix page 42.

3.5.d Simulations with observer

Matlab simulation

The figure 3.13 has been generated in Matlab (`responseObserver.m`) by considering the state space of equation 3.22. As can be observed, the response of the system with observer is the same than without it.

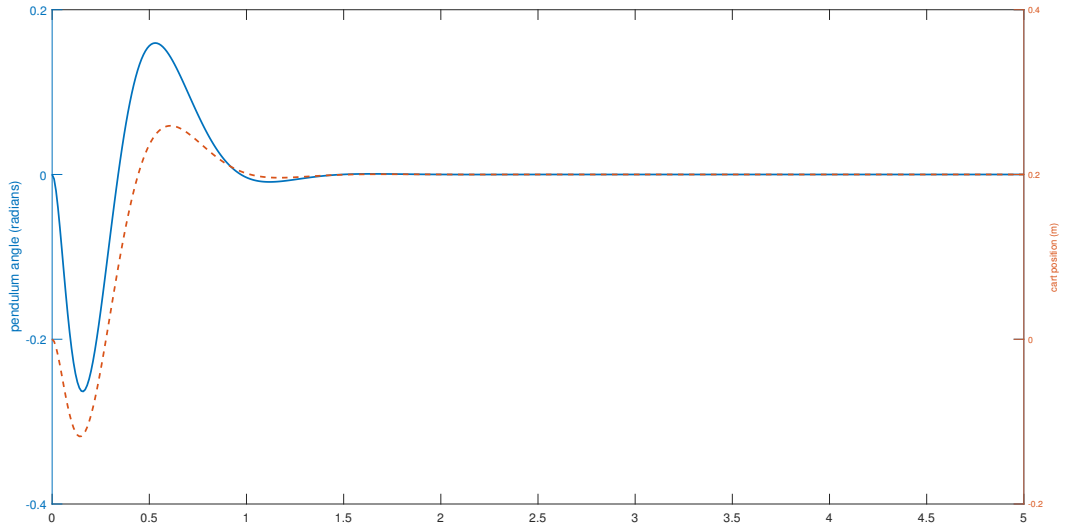

Figure 3.13: Response of the system on a constant input of 0.2 for 5 seconds with observer

Simulink simulation

The observer as well as the full closed-loop system have been implemented in Simulink in the `closed_loop.slx`. The model can of course be used to generate the same plot as above.

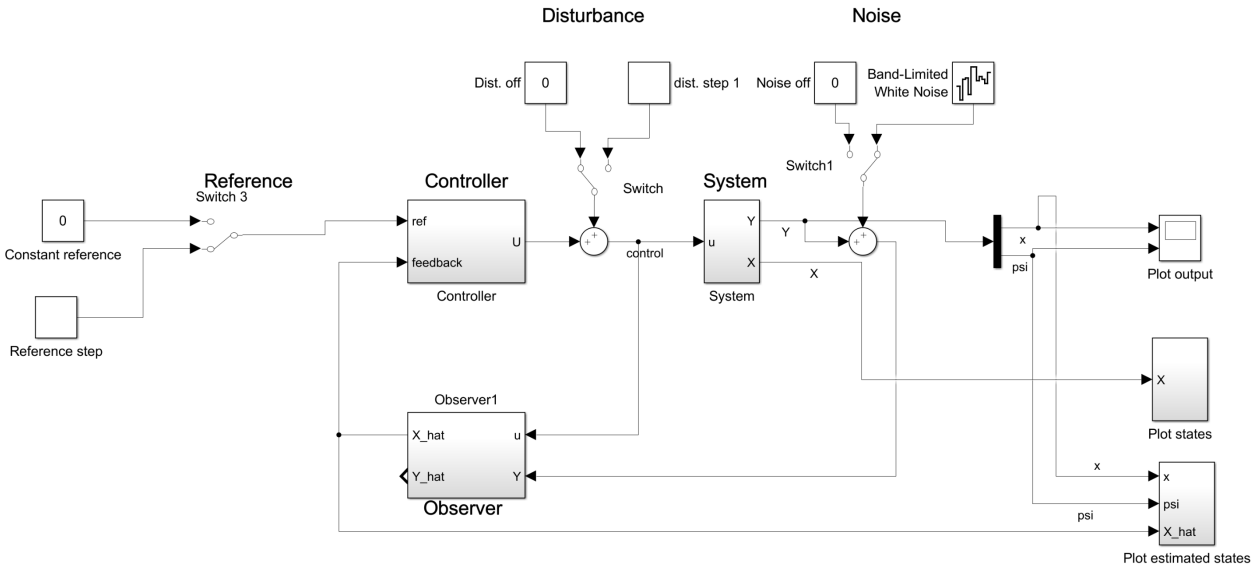

Figure 3.14: State feedback with observer in Simulink[3].

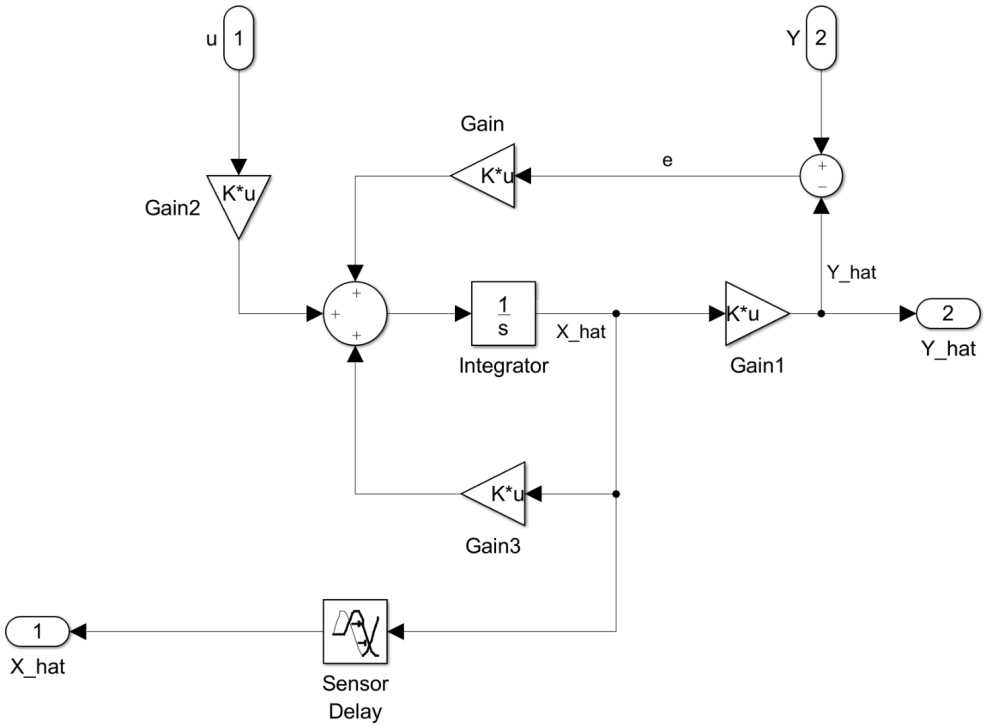

Figure 3.15: Observer block diagram in Simulink

3.5.e Response to a noise measurement

White noise

Adding a white noise of (power parameter = 0.00001) on both output as well as a delay of 10ms seemed to alter quite a lot the performance of our system as can be observed in Figure 3.16. No disturbance (F) or any change in the reference has been done to generate this plot. Like so, the pendulum can oscillate around 4.5° around the vertical. The fact that we used quite high gains to design quite fast controller and observer may explain this lack of robustness.

The last step is to analyze how the system behaves when the cart is asked to move 20 cm. Figure 3.17 show such response.

Unfortunately, the constraint of keeping the deviation angle below 0.3 radians is not met anymore if white noise is applied.

Uniform random noise

When applied a uniform random noise between -0.005 and 0.005, the system behaves already better. When the system is not disturbed or its reference changed, the angle only deviates at worst of -0.021 radians.

When a applied a change in the reference of the cart, the system behaves more or less like without noise, except that that oscillations persists similar to the one shown in figure 3.18.

All in all, we can conclude that the system behaves more or less correctly when the system is applied noise. The observer does a good job of tracking the states thanks to its fast convergence.

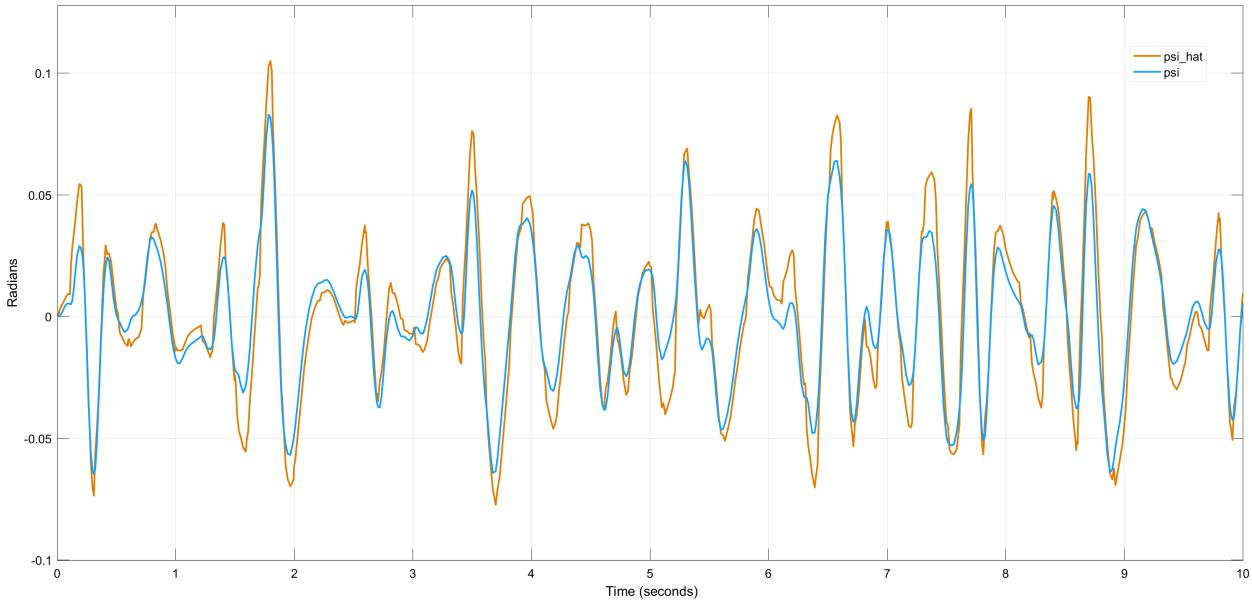

Figure 3.16: Evolution of the angle when the system is subject to white noise.

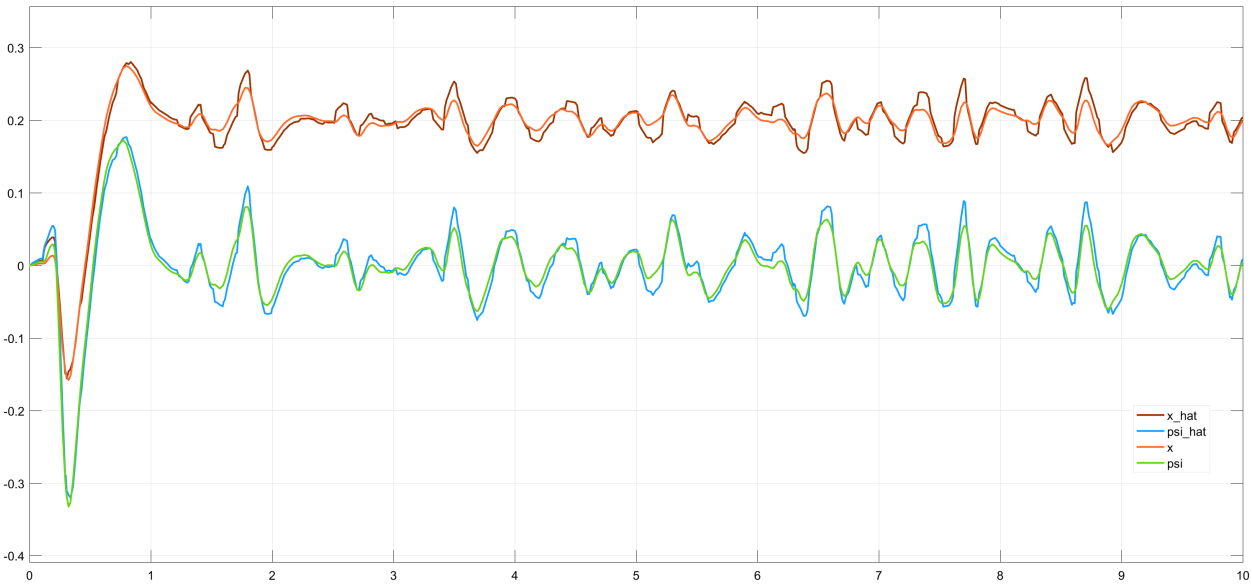

Figure 3.17: Response to a step in reference of 0.2 (cart translate of 20 cm) with observer and subject to noise

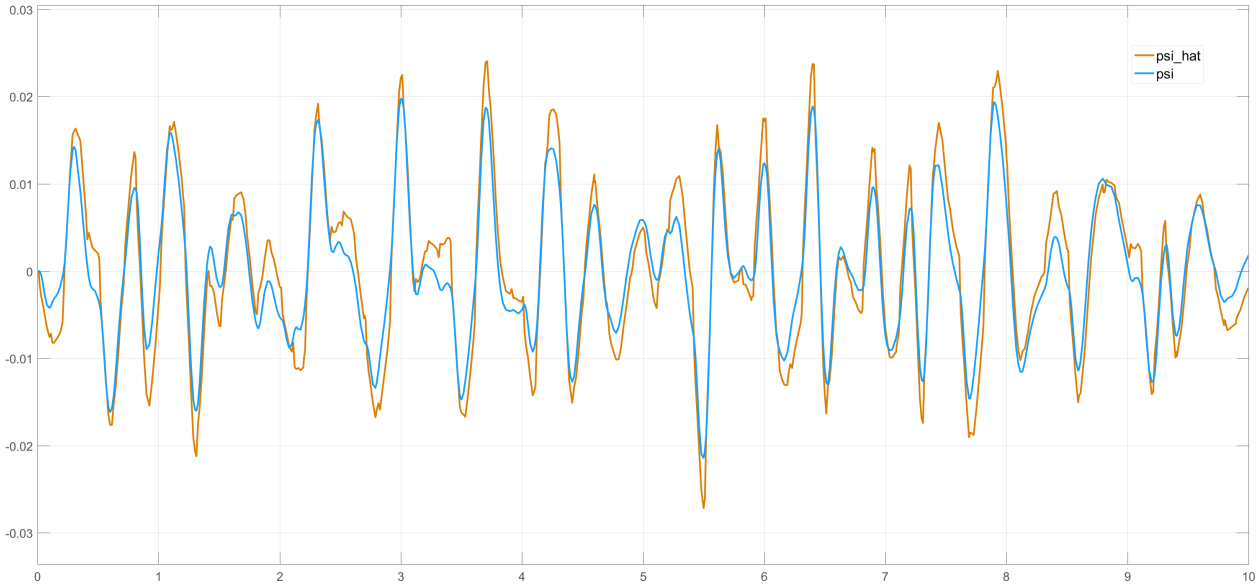

Figure 3.18: Evolution of the angle when the system is subject to random uniform noise in [-0.005, 0.005].

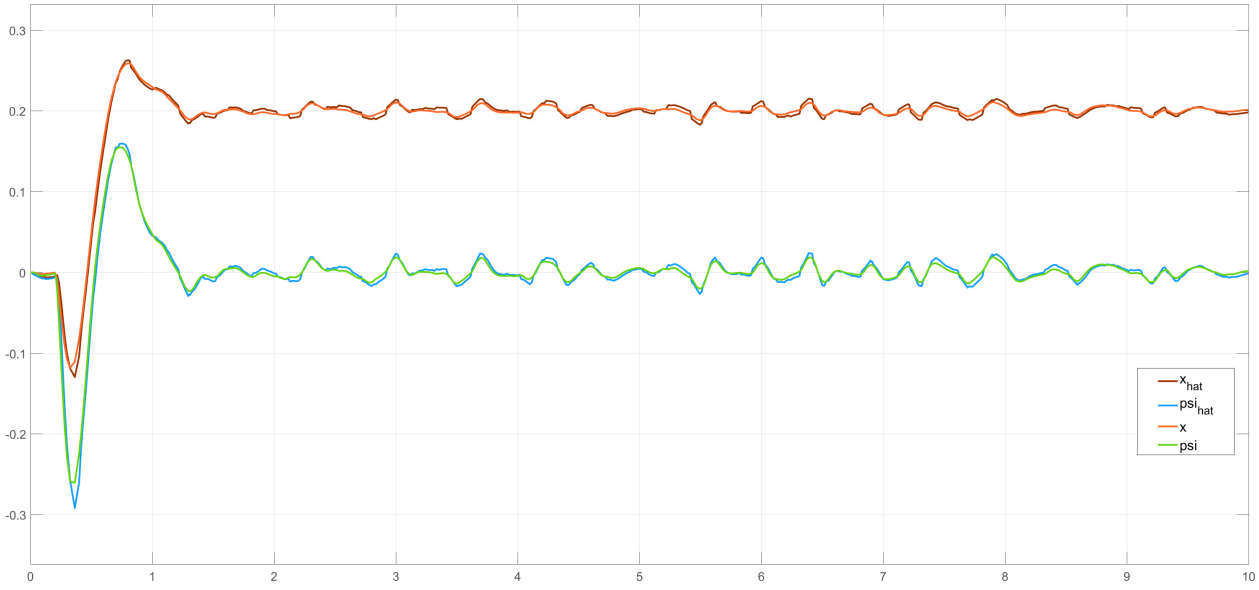

Figure 3.19: Response of the system (tuned by pole placement and with observer) to a change in reference (20 cm) when a random uniform noise in [-0.005, 0.005] is applied.

4 | Frequency analysis

4.1 Transfer Fonction

For designing the controller in frequency domain we need the transfer function of the system.

$$\frac{\phi(s)}{F(s)} = \frac{\frac{mL_{CM}}{q}s}{s^3 + \frac{b(I+mL_{CM}^2)}{q}s^2 - \frac{(M+m)mgL_{CM}}{q}s - \frac{bmgL_{CM}}{q}} \tag{4.1}$$

Since this is the same transfer function (TF) as before, we have our original poles back. As a reminder, here are their values :

4.8540	-0.1396	-4.8787

Table 4.1: Poles of the transfer function of our system

4.2 Bode Plot

Since our system has poles in the right-half complex plane, it is unstable. Thus we cannot analyse it on a Bode diagram.

4.3 Nyquist Plot

A loop transfer function is a function $L(s)$ so that

$$L(s) = P(s)C(s) \tag{4.2}$$

where $P(s)$ is the TF of the process, and $C(s)$ the TF of the controller.

This function is useful to analyse the robustness and stability of the closed loop system. It also allows to tune the controller in order to achieve our expectations.

We will use the Nyquist Plot (the loop transfer function plotted for different complex frequencies) in order to determine the stability of the system. Indeed, the Nyquist criterion states that :

"*If the loop transfer function has no poles in the right half plane, then the closed loop system is stable if and only if the closed contour of the curve has no net encirclements of the critical point s = -1.*"

This Nyquist stability criterion can be written as

$$Z = N + P$$

Where Z is the number of poles of the closed loop system in the right half of the complex plan, N is the net number of clockwise encirclements around the critical point $(-1 + j0)$ or -1 if it is counterclockwise and P the number of poles in the region enclosed by the Nyquist contour.

The Nyquist plot of our open loop system is represented in the Figure 4.1

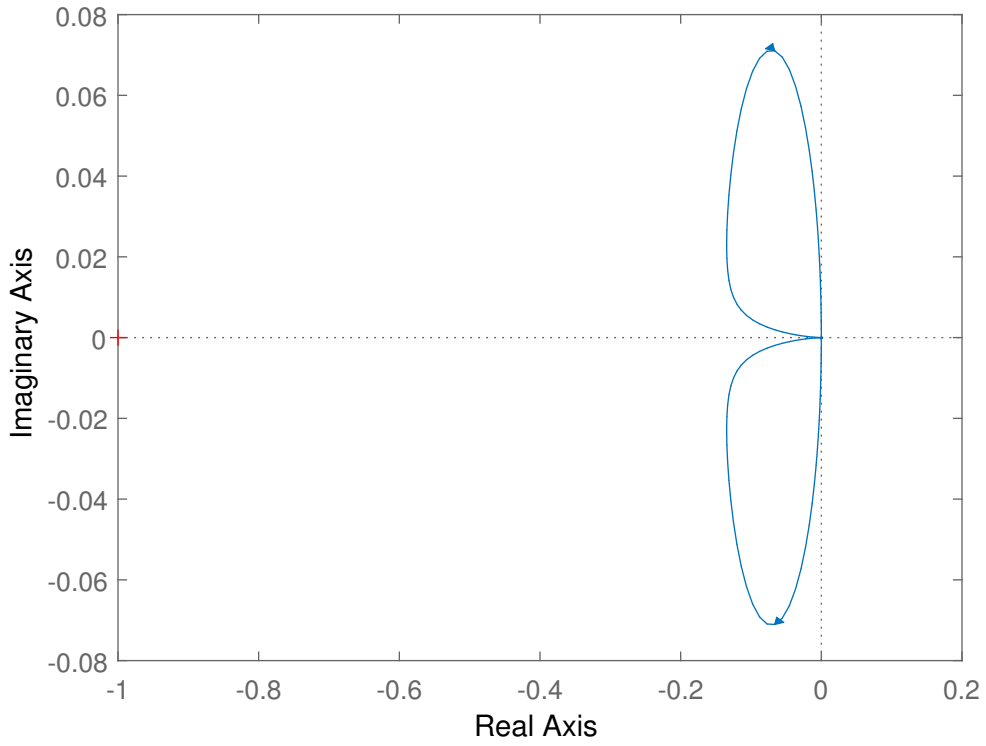

Figure 4.1: Nyquist plot for the open loop system

We can see that there is no circulation around the critical point, therefore $N = 0$. As we know the location of the poles of our system's transfer function, we know that there is 1 pole enclosed by the Nyquist contour (the pole in the right-half complex plane). Thus $P = 1$.

Therefore, we have

$$\begin{aligned} Z &= N + P \\ &= 0 + 1 \\ &= 1 \text{ pole of the closed loop system in the right-half complex plan} \end{aligned} \tag{4.3}$$

As expected, the Nyquist plot confirms that we have an unstable system if we didn't add a controller. So let's design it.

4.4 Design of the controller

In order to stabilize our system, we need to design properly our controller. Mathematically we need to find the function $C(s)$ that stabilizes $L(s)$ according to the Nyquist criterion. To build $C(s)$ we combine several blocks.

$$C(s) = C_1(s) * C_2(s) * C_3(s) * \cdots \tag{4.4}$$

4.4.a Integrator

First of all we want to cancel the zero at the origin. To do so, the first block of our controller needs to be an integrator.

$$C_1(s) = 1/s = C(s)$$

This gives us the following Nyquist plot.

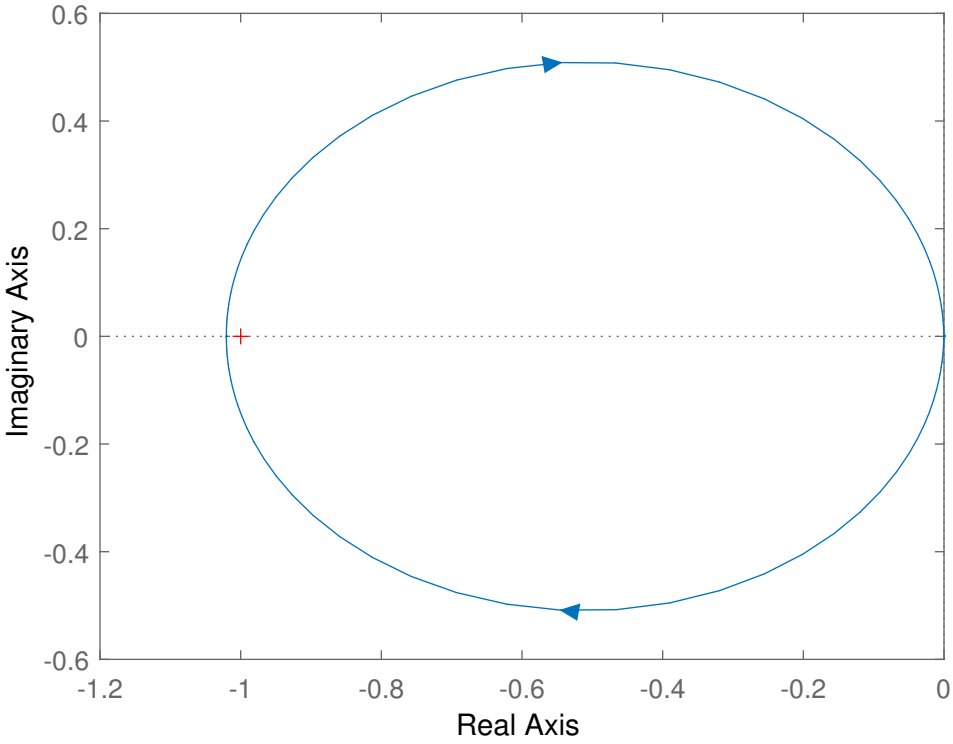

Figure 4.2: Nyquist plot whith an integrator

As we an see in the Figure 4.2, $Z = 2$ now and the system is, thus, still unstable.

4.4.b Addition of zeros

In order to make the system stable, we need to add phase to get a counterclockwise encirclement. For that purpose, we add a zero to our controller. We add it in -1 as it is the default value in Matlab. We obtain the Figure 4.3. Obviously this change doesn't provide enough phase and the encirclement around -1 is still clockwise. So we add another zero in $s = -1$ and we get the Figure 4.4.

Thus we have

$$C_2(s) = (s+1) \quad \text{and} \quad C_3(s) = (s+1)$$

and

$$C(s) = \frac{(s+1)^2}{s}$$

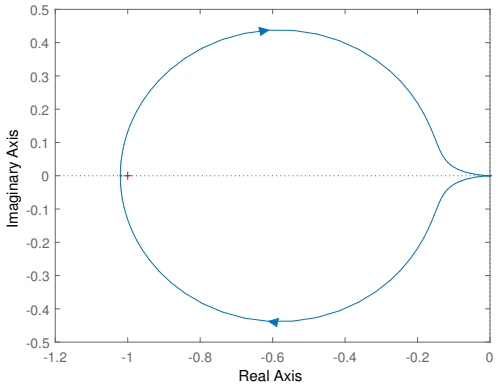

Figure 4.3: With one zero in $s = -1$

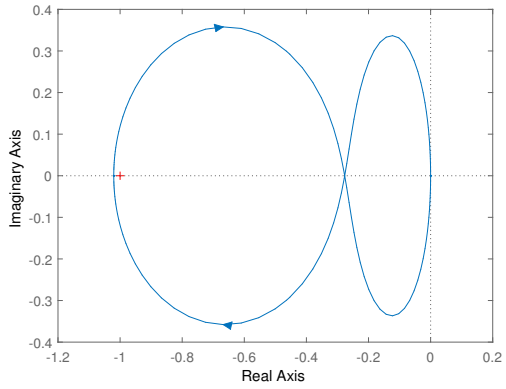

Figure 4.4: With two zero in $s = -1$

In this final Nyquist plot, the -1 point is still encircled clockwise but we have now another loop that is travelled counterclockwise.

4.4.c Addition of gain

In order to stabilize the system, we want that the newly formed loop encircles -1. To achieve that goal, we add some gain in order to increase the magnitude of each points of the Nyquist plot.

The critical point enters the second loop for a gain $\simeq 3.7$. However, at this point, we choose to take a gain $K = 10$. The results of the addition of gain are shown in the Figures 4.5 and 4.6.

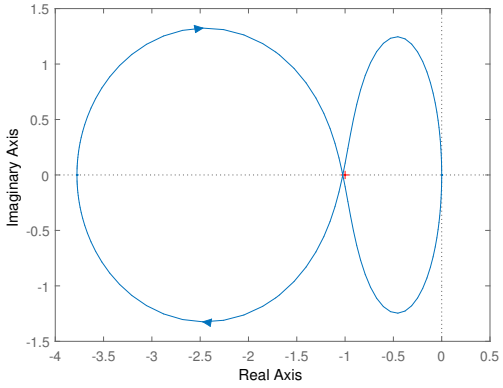

Figure 4.5: With $K = 3.7$

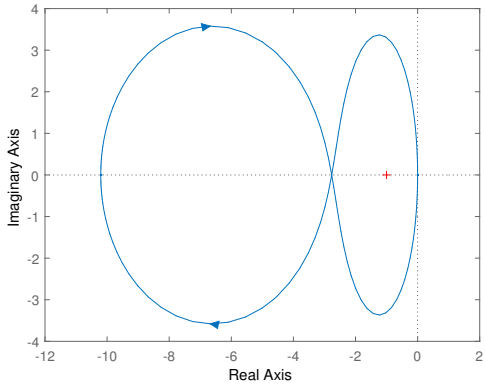

Figure 4.6: With $K = 10$

At the end, we get

$$C(s) = 10\frac{(s+1)^2}{s}$$

With this transfer function for our controller, our system becomes stable. Indeed, if we look at the Nyquist criterion, we have $P = 1$ that didn't change but we have $N = -1$ as the encirclement around -1 is now counterclockwise. This gives us $Z = N + P = 0$ assuring that our closed loop doesn't have poles in the right-half complex plan.

If we plot the impulse response of our closed loop system, we obtain

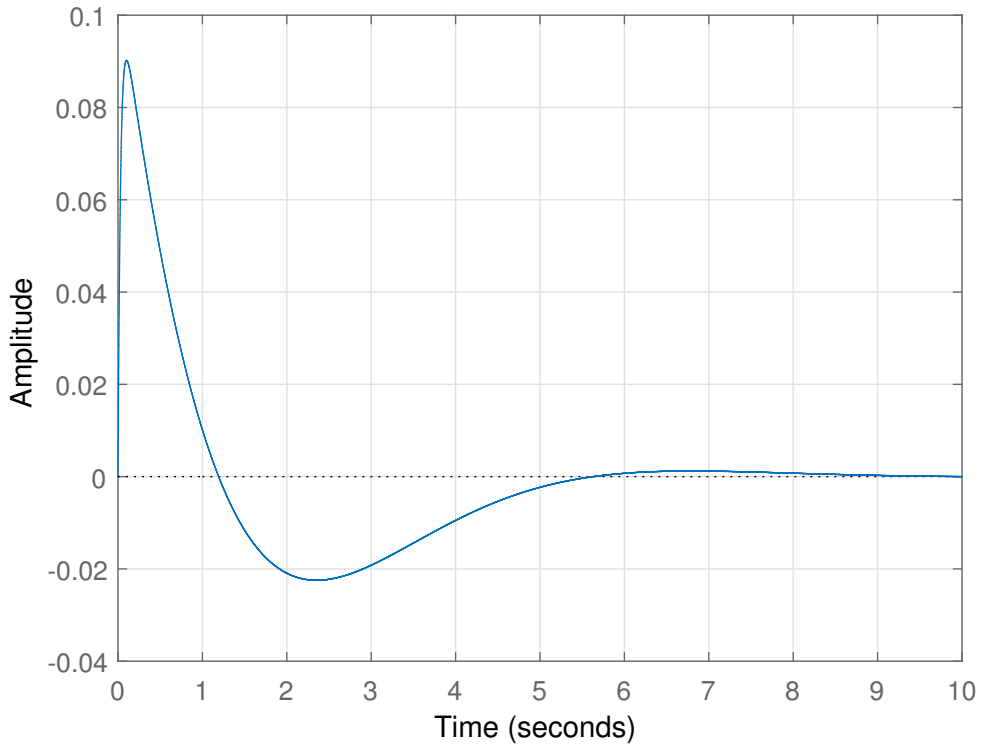

Figure 4.7: Impulse response of the closed loop system

From Figure 4.7, we can see that the angle of the shaft converges, as expected, towards 0. Our system is stable and can handle disturbances.

4.4.d Additional tuning of the controller

However, we can notice in the Figure 4.7 that the system takes more than 5 seconds to stabilize. To correct that, we can tune the gain k and the location of one of the two zeros, pushing it further to the left.

We decide to take

$$C(s) = 20\frac{(s+1)*(s+10)}{s}$$

With that controller we obtain the following impulse response.

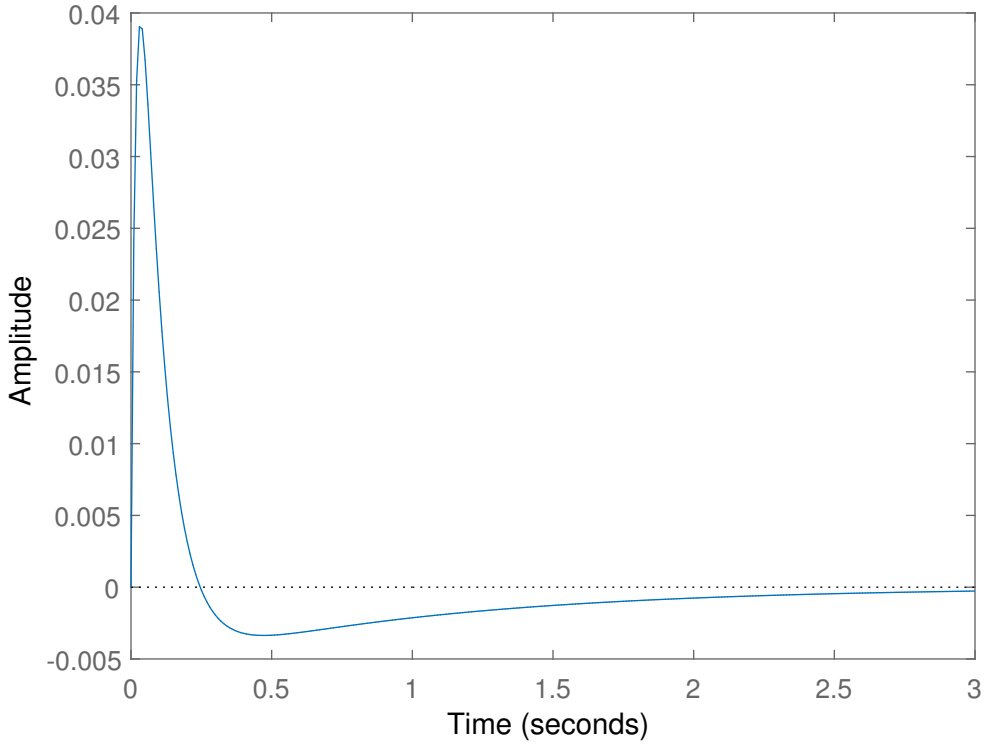

Figure 4.8: Final impulse response of the closed loop system

At first the overshoot can seem pretty big, but if we look at the scale, our overshoot doesn't get higher than 0.040 radiant, which is perfectly good. The stabilisation time is a bit above 2 seconds, which is good considered the really small variation in amplitude it represents.

4.5 Gang Of Four

The whole behaviour of the system can be studied through four transfer functions : the gang of four. Together, they describe the relation between the inputs and the outputs of the system. The first one is the sensitivity transfer function and describe how the noise is seen in the output. This is it's definition :

$$S = \frac{1}{1+L}$$

The second transfer function is the complementary sensitivity function describing how the disturbances influence the control signal. The expression is :

$$T = \frac{L}{1+L}$$

The next function to consider is the load sensitivity transfer function; describing how the disturbances influence the output. Here is its expression :

$$PS = \frac{P}{1+L}$$

Finally,we have the noise sensitivity transfer function which describe the influence of the noise over the control signal. It can be written as :

$$CS = \frac{C}{1+L}$$

As L is a part of these equation, any modification of L after stabilising these equations would mean modifying these equations as well.

The function $C(s)$, found in the previous section, makes our system stable and allow the pendulum to cope correctly with small kicks. However, when we compute them, we realise that some of the functions of the gang of four were still unstable.

Now, the goal is thus to stabilise all four of these equations, as an instability for one of them means that one of our outputs would explode.

Here are the Bode plots of the sensitivity function as well as the complementary sensitivity function :

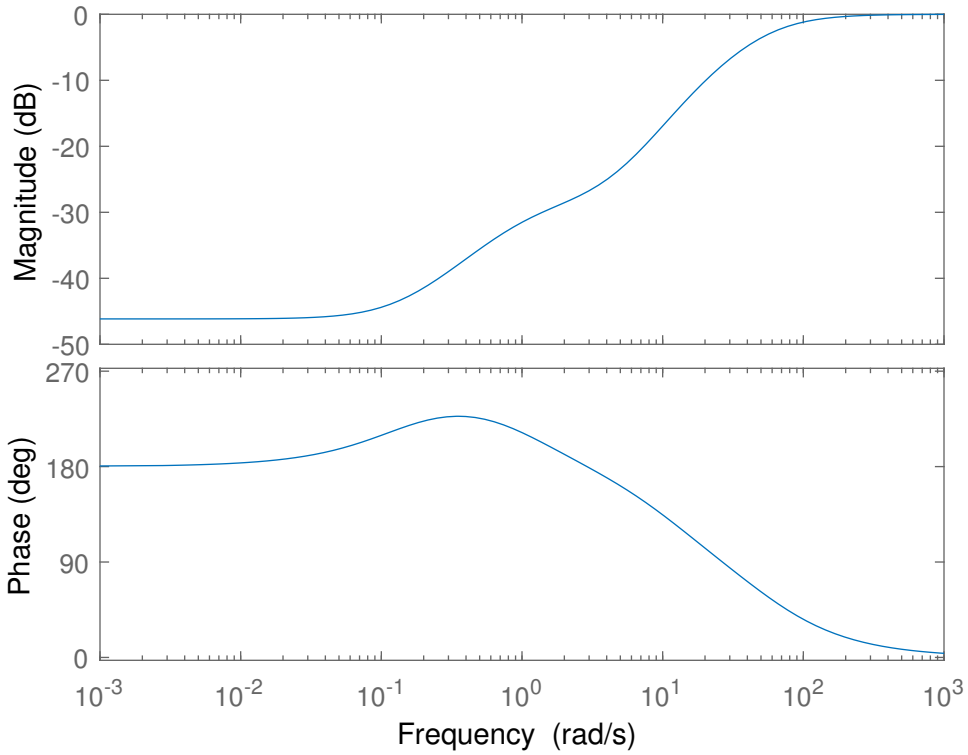

Figure 4.9: Bode plots of the sensitivity function

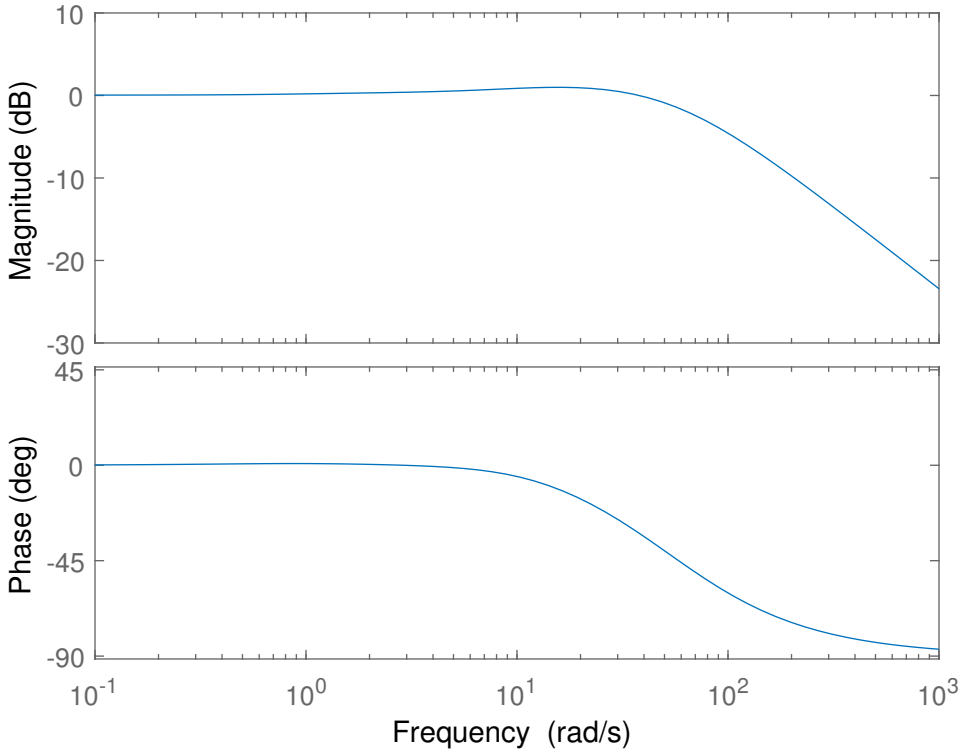

Figure 4.10: Bode plots of the complementary sensitivity function

We can see that the sensitivity function behaves nicely, as the magnitude tends to 0 for high frequencies while the complementary sensitivity function goes down to -30dB, which is good for noise rejection. At the same time the sensitivity function decreases to -50dB for low frequencies, where the complementary sensitivity function is at 0. We should therefore have a good reference tracking.

Unfortunately, our L(s) in its current state was leading to instability for 2 out of four equations of the gang of four : the load sensitivity transfer function and the complementary sensitivity function. The 2 others are fine.

4.6 Noise and delay impact on the system

In order to generate noise and analyze the response of the system, it is more convenient to use a Simulink model. That way, we can plot different response curves for different value of the delay for example. We didn't have the time to play around a lot with it. Due to our C(s), it is impossible to represent the controller with a standard block either. Still, a screenshot of the Simulink, not yet finished, is available in the appendix (page 43).

4.7 Comparison between time and frequency domain

To compare the time domain to the frequency domain we need to look at the Figures 3.7 and 4.8. As a reminder, here are the two graphs :

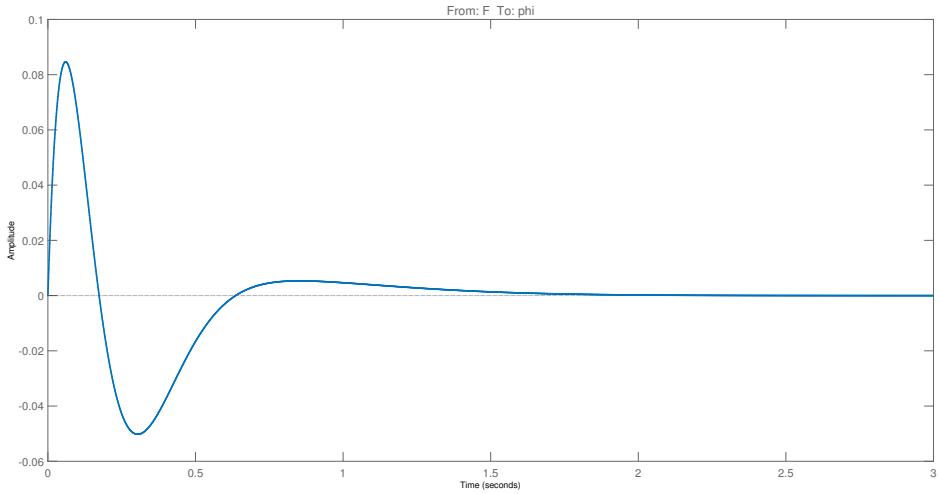

Figure 4.11: Time domain

fig:stable_impulse_final-eps-converted-to.pdf

Figure 4.12: Frequency domain

These graphs show the impulse response of our system. We can see that in the time domain the system converges quickly to the reference ($\approx$ 1s) while in the frequency domain it takes roughly 2s. However, the scale of the overshoot is much smaller for the frequency domain (≈ 0.04) than for the time domain (≈ 0.085).

5 | Conclusions

In this project, we have analysed the behaviour of a inverted pendulum in the time and frequency domain. The time domain implementation required the design of a controller and an observer, while the frequency domain required the design of a controller through the design of the loop transfer function.
In time domain, we successfully designed a controller that can stabilize the pendulum in less than 3 seconds with a deviation from the vertical of less than 0.3 radians in any situation. We also highlighted the good behaviour of the system to a change in the reference of the cart. The observer successfully tracked the state of the system. Only the white noise pushed the system to its limit with a deviation a touch higher than 0.3 radians.
For the frequency domain, we designed a controller through a combination of different blocks fulfilling various roles. The design of these blocks has been driven through the evaluation of Nyquist plots, and then we used the gang of four to conclude about the overall stability. We can see that while both systems have responded to our desired properties, the time domain system has bigger overshoot, compared to the frequency system, for a similar stabilising time. The time domain system also presents more oscillations and a bit of undershoot.

Here is a summary table of the different meaningful results we found in time domain and frequency domain

	Time domain	Frequency domain	Discussion
Stabilisation time (impulse response)	1 sec	3 sec	Very good
Overshoot value (impulse response)	0.085 rad	0.040 rad	Good
Maximum acceleration of the cart	1.42 m/s^2	/	A bit high
Noise resistance	$<$ 0.1 rad (white noise)	/	Ok for t.d.
	$<$ 0.02 rad (uniform noise)	/	
Delay resistance	Very good	/	Very good

Table 5.1: Comparison between time and frequency domain

6 | Appendix

6.1 Bonus plots

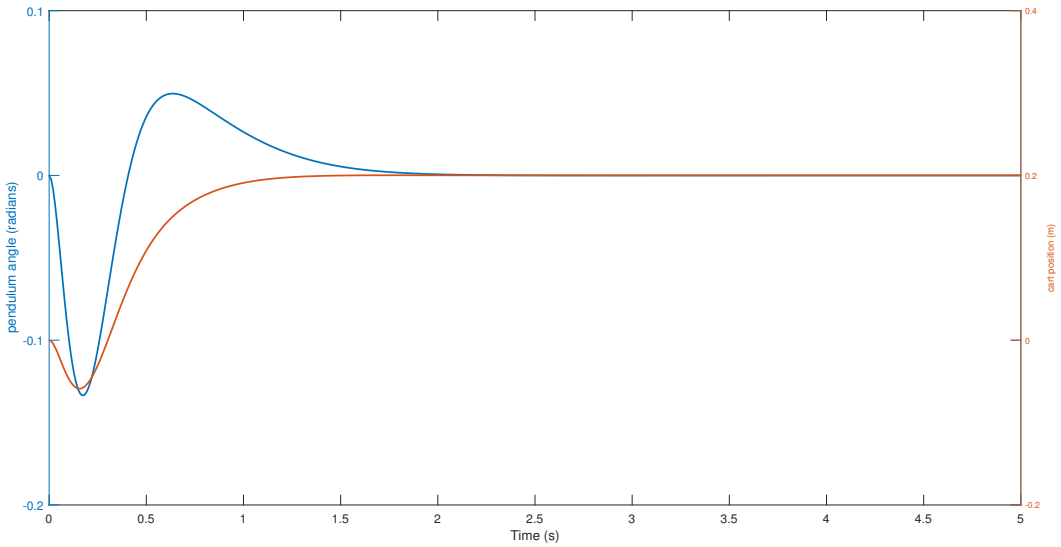

Figure 6.1: Response of the system tuned by lqr to a single impulse change in the reference of the cart position (20cm)

6.2 Observer analysis and design

The combination of the controller and the observer gives:

$$\begin{cases} u = r - K\hat{x} \\ \dot{\hat{x}} = (A - LC)\hat{x} + Bu + Ly \end{cases} \iff \dot{\hat{x}} = (A - BK - LC)\hat{x} + Ly = (A - BK - LC)\hat{x} + LCx \tag{6.1}$$

On the other hand,

$$\dot{x} = Ax + Bu = Ax - BK\hat{x} \tag{6.2}$$

Following the idea of [5, pg18], we stacked (6.1) and (6.2) gives a matrix of the form:

$$\begin{bmatrix} \frac{dx}{dt} \\ \frac{d\hat{x}}{dt} \end{bmatrix} = \begin{bmatrix} A & 0 \\ LC & A - LC \end{bmatrix} \begin{bmatrix} x \\ \hat{x} \end{bmatrix} + \begin{bmatrix} B \\ B \end{bmatrix} u \tag{6.3}$$

.

We implemented such system and simulated it with `lsim` as usual without success. The reference [6, pg4] proposed another approach:

$$\begin{bmatrix} \frac{dx}{dt} \\ \frac{d\hat{x}}{dt} \end{bmatrix} = \begin{bmatrix} A & -BK \\ LC & A - LC - BK \end{bmatrix} \begin{bmatrix} x \\ \hat{x} \end{bmatrix} + \begin{bmatrix} B \\ L \end{bmatrix} r \tag{6.4}$$

6.3 Tuning the observer by pole placement

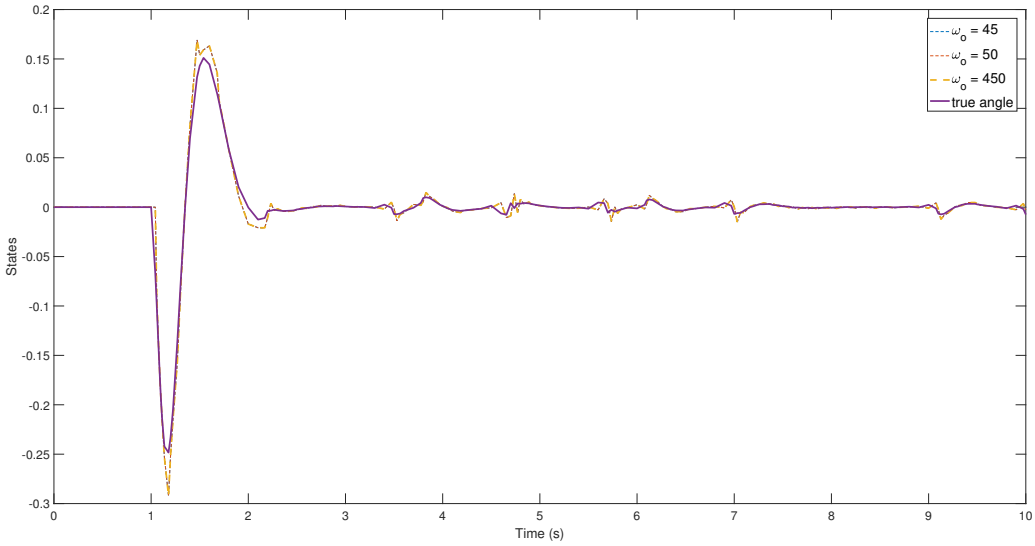

Figure 6.2: Tuning of the observer gain by pole placement for different values of ω_0.

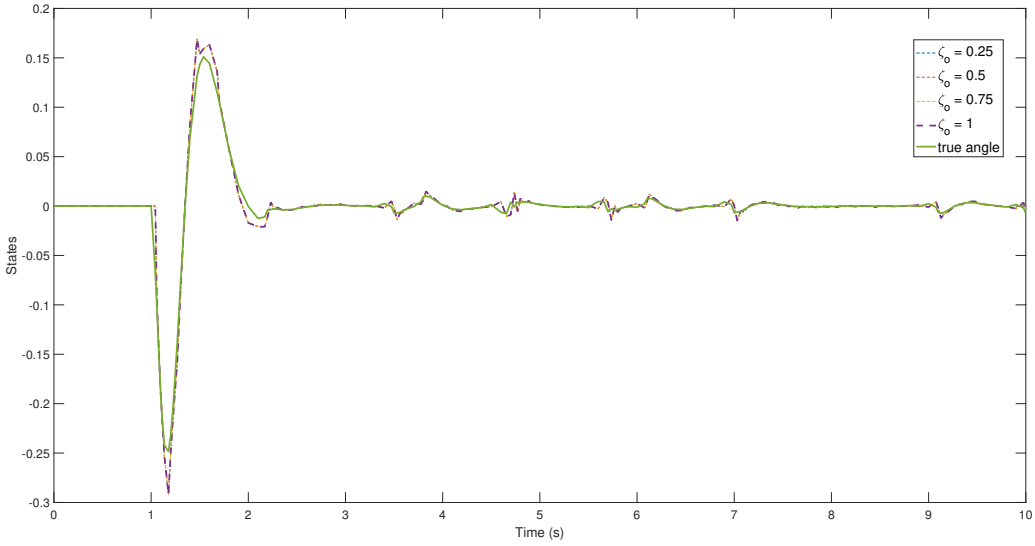

Figure 6.3: Tuning of the observer gain by pole placement for different values of ζ_0.

6.4 Simulink in frequency domain (not finished)

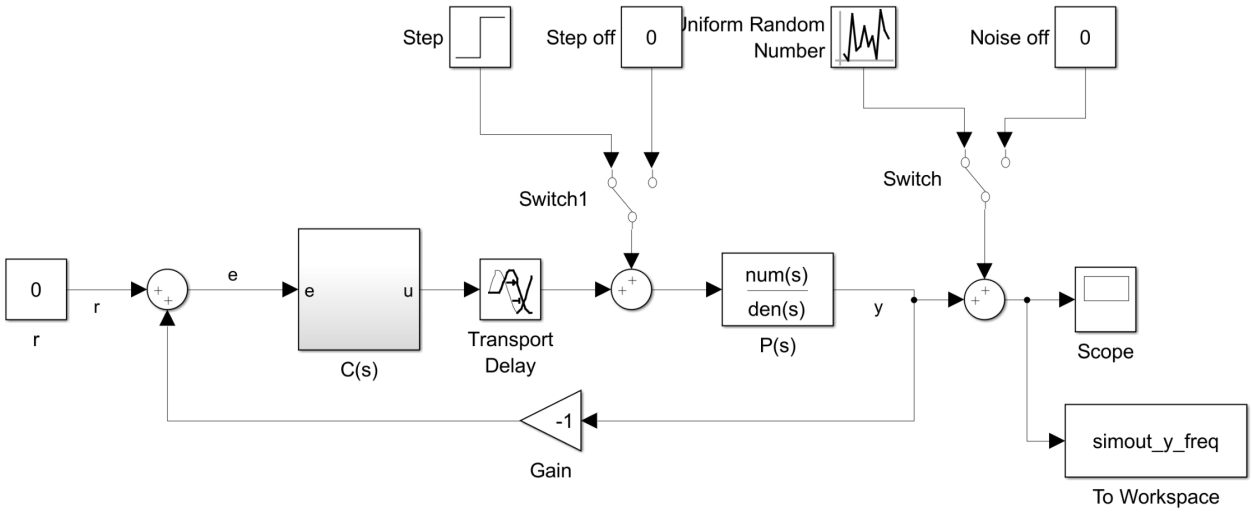

Figure 6.4: Simulink in frequency domain

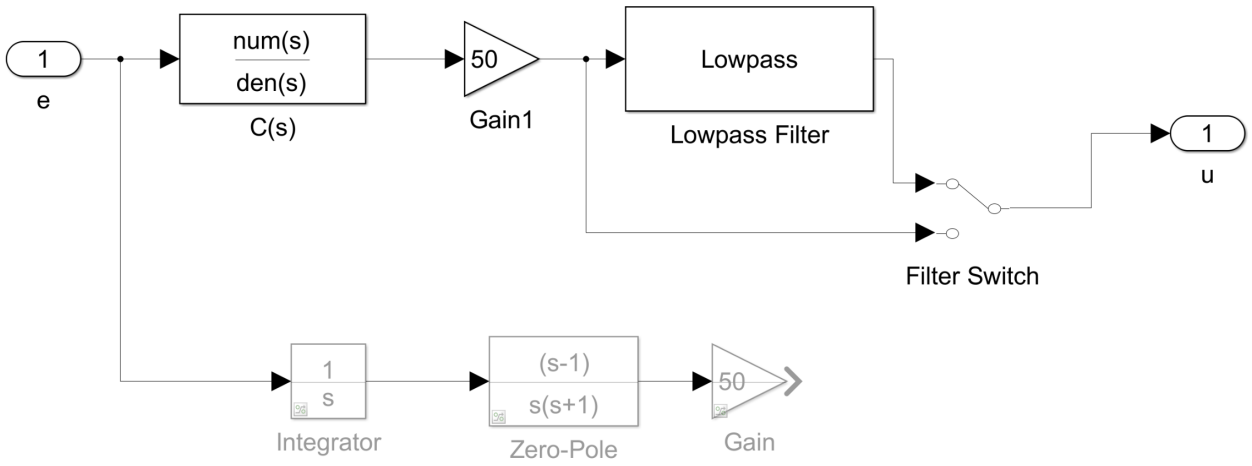

Figure 6.5: Simulink in frequency domain (controller)

7 | Bibliography

- **[1]** MESSABIH Mohamed, ZEKRI Boulanouar, *Modélisation et Commande pour la stabilisation d'un robot pendulaire inversé sur des roues*
- **[2]** Ye Ding, Joshua Gafford, Mie Kunio , *Modeling, Simulation and Fabrication of a Balancing Robot*
- **[3]** KHEMISSAT Abdelmouneim, *Plateforme de prototypage rapide pour la robotique mobile: Application à un robot Auto-balancé*
- **[4]** BENARIBA Hassan , *Commande d'un robot mobile (Mobrob) sur deux roues*
- **[5]** Hari Vasudevan, Aaron M. Dollar, John B. Morrell, *Design for Control of Wheeled Inverted Pendulum Platforms*
- **[5]** Perry Y.Li course
- **[6]** Prof. Khan, *Observer design-separation principle*
- **[7]** F.L. Lewis, *State observer and regulator design*
- **[7]** Bill Messner, Dawn Tilbury, Rick Hill, JD Taylor, *The website "Control Tutorials for MATLAB and Simulink"*