HHRI report

- Lab 2 -

SIMULATION AND REALISATION OF A PID CONTROLLER

10th September 2024

Students : Killian Raude NX - Jade Therras NX - GROUP 12
Assistants : Giulia Ramella, Jonathan Muheim, Aiden Xu, Mouhamed Zorkot

Contents

1 Basic identification

To assess the dry friction of the paddle from the contact between the metallic cable and the motor screw in both directions,the motor torque was gradually increased/decreased from the graphical interface until motion was observed in the clockwise and counterclockwise directions respectively. A few trials were made in both directions and the torque needed for the paddle to move was noted. Motion in the clockwise direction was observed with a torque of $0.0015[N \cdot m]$ for the paddle to move, and $-0.001[N \cdot m]$ for it to turn the other way.

This torque can be converted in a force by dividing it by the radius of the worm screw yielding :

$$\tau_{motor} = F_{friction} \cdot r_{wormscrew} \qquad F_{friction} = \frac{\tau_{motor}}{0.005m} \tag{1}$$

Giving rise to dry friction forces of $0.3[N]$ and $-0.2[N]$ in the clockwise and counterclockwise direction.

2 Simulation with Simulink

First, the paddle was modeled on Simulink using the sum of momentum equation Fig.1.

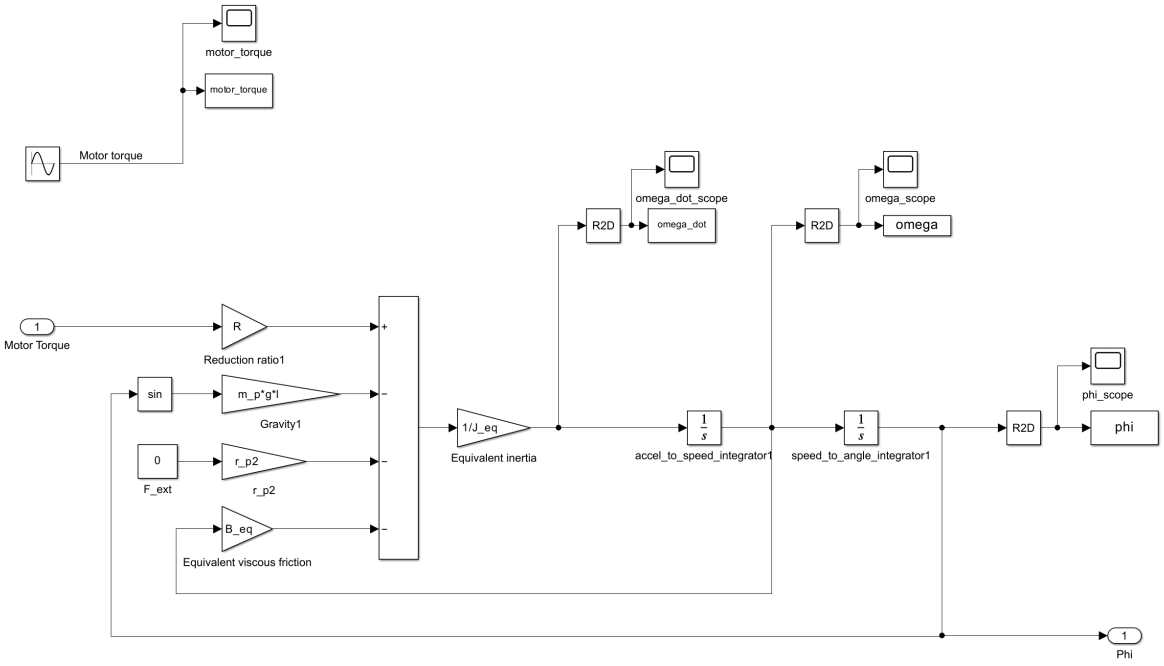


Figure 1: Screenshot of the paddle simulation on Simulink [1]

The equation for the sum of momentum considers the force of the motor, the viscous friction, the gravity and the external force. The dry friction is ignored in this model. The equivalent inertia takes into account the inertia of the rotor, the screw and the paddle. The equivalent viscous friction can be computed from the friction of the paddle and of the motor, taking care to include the reduction ratio of the cable transmission. The equivalent inertia and viscous friction are computed on the paddle side. All notations can be found in the annex.

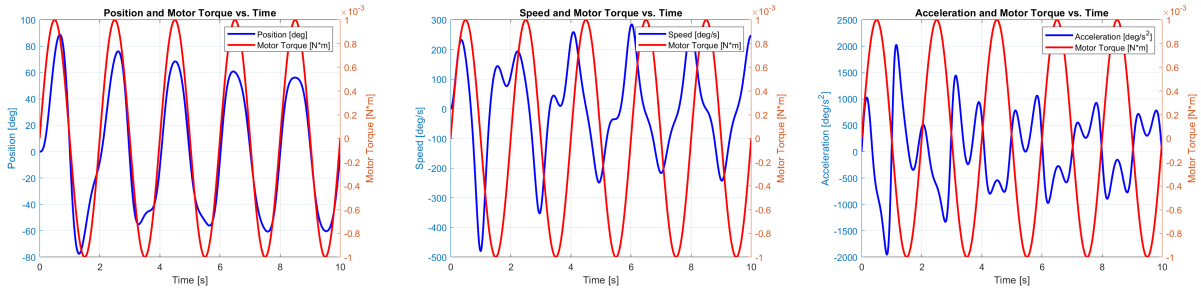
$$\ddot{\Theta} = \frac{1}{J_{eq}}(RF_{motor} - B_{eq}\dot{\Theta} - m_{paddle}glsin(\Theta) - F_{ext}l_{paddle}) \tag{2}$$

where

$$B_{eq} = B_{motor}R^2 + B_{paddle} \tag{3}$$

$$J_{eq} = (J_{rotor} + J_{wormscrew})R^2 + J_{paddle} \tag{4}$$

The behavior of the model can then be observed from the output of the simulation. Here are the results of the simulation of the position, speed and acceleration for an applied sin wave motor torque of amplitude $0.001[Nm]$ and a phase of $\pi[rad]$ (parameters given for the simulation).



Position Speed Acceleration

Figure 2: Simulated behaviour of the paddle (blue) while applying a sin wave motor torque (red) [2]

The main problem of the simulation is that is does not respect the physical constraints of the physical paddle. The range of motion is supposed to be constrained between [30;-45] degrees while the maximum motor torque applicable is supposed to be $0.032[Nm]$. In the simulation the motor torque and the position go beyond that range. The position is also expected to follow the torque but can be seen decreasing gradually.

A PID controller has then also been implemented in the simulation using the default PID block in Simulink Fig.3.

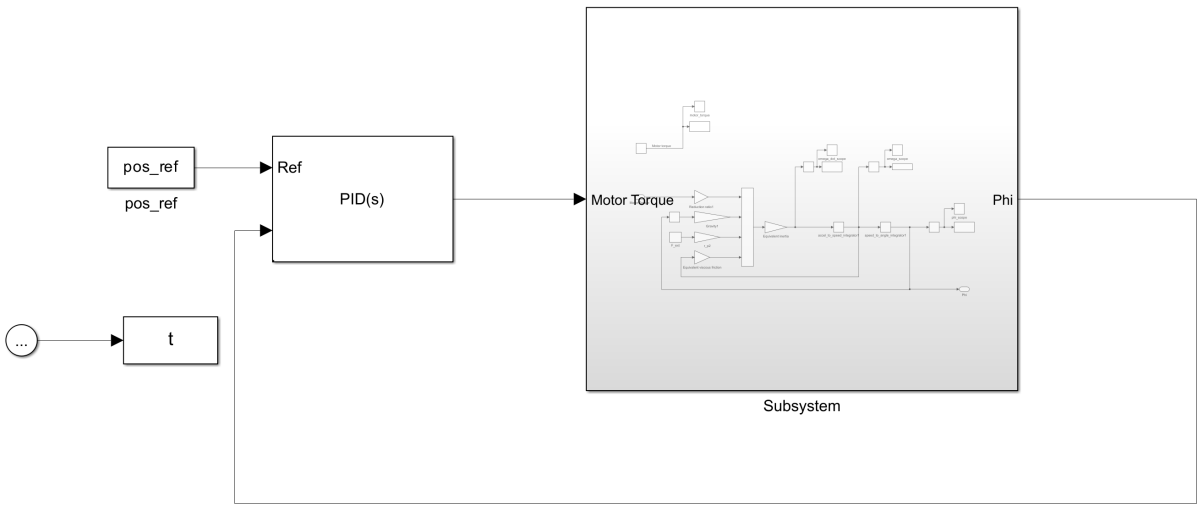


Figure 3: Screenshot of the full simulation on Simulink including the paddle and the PID controller [3]

The PID controller takes a reference position and the actual position of the model to output and feed back the actuated position into the paddle subsystem. The controller is also dependent on 3 parameters K_p, K_i and K_d which can be hand tuned.

K_p change the strength of the correction of the instantaneous error, K_d the strength of the correction of the instantaneous change in error and, K_i the strength of the correction of the cumulative error. Starting with a small K_p and increasing it, the model reaches the target position with an overshoot and oscillations around the targeted value. K_d was then tuned to moderate the overshoot and finally, K_i was used to fine tune the model.

In the simulation 4, the rising time is fast, the overshoot is important and the settling time is high.

3 Position controller implementation on the board firmware

The PID controller was implemented on the board according to the equation :

$$\tau_m = K_p(\Theta_d - \Theta_m) - K_d\dot{\Theta}_m + K_i \int_0^t (\Theta_d - \Theta_m)dt \tag{5}$$

The PID controller starts disabled and can be turned on on the board graphical interface with the possibility to choose between the hall sensor and the incremental encoder sensor for the measure of the paddle position, it is also possible to change the filtering frequency of these signals.

4 Comparison between the simulation and the physical implementation

The PID controller was tested on the paddle using the parameters extracted from the simulation as a first step.

$$K_p : 0.015[Nm/rad]$$
$$K_i : 0.025[s \cdot Nm/rad]$$
$$K_d : 0.001[Nm/rad \cdot s]$$

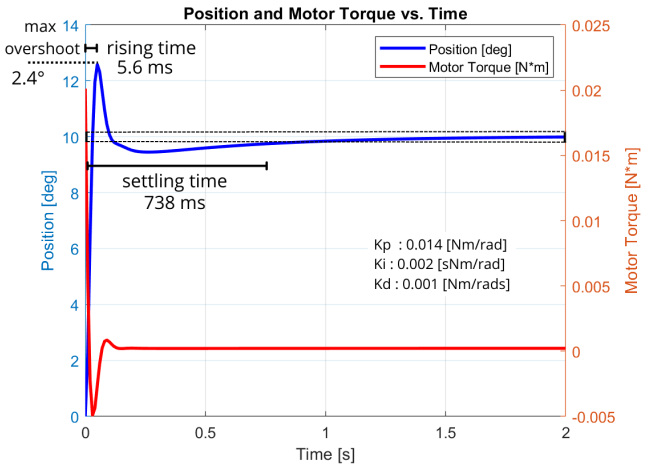


Figure 4: Simulated 10° step response [4]

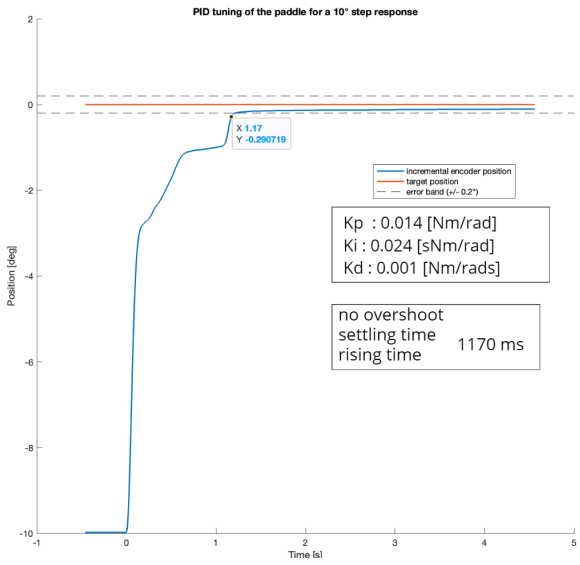


Figure 5: physical 10° step response [5]

With hand-tuned parameters obtained form the simulation and the incremental encoder

In the physical system Fig.5, there is no overshoot, the rising time is slower and the settling time is much longer but it still stabilizes in an acceptable range. This implementation does not meet the requirements for a 10 degree step response (maximum overshoot of 25%, maximum settling time of 350 ms, maximum rise time of 100 ms) so the K parameters were tuned later accordingly. However this step was still very important to understand the effect of each K parameters and to see the difference between the model and the implementation.

The difference between the simulation and the physical paddle can be mainly explained by the lack of modeling of dry friction forces in the simulation. As the dry friction forces were calculated at the beginning, the simulation could be changed to include a friction modeling including Stiction and Stribeck effects to reflect the physical paddle more.

5 PID Hand-tuning with incremental encoder and hall sensor

in this part, the filtering has been done with a cutoff frequency of 20Hz and the tuning has been done using the incremental encoder as it is less noisy.
Starting from the parameters of the simulation, the parameters of the physical implementation were hand tuned. Firstly, K_p has been increased to obtain a faster convergence, secondly K_d has been tuned to reduce the overshoot and manage the oscillations. Finally K_i is used for fine tuning to help improving the performance in the region close to the target value.

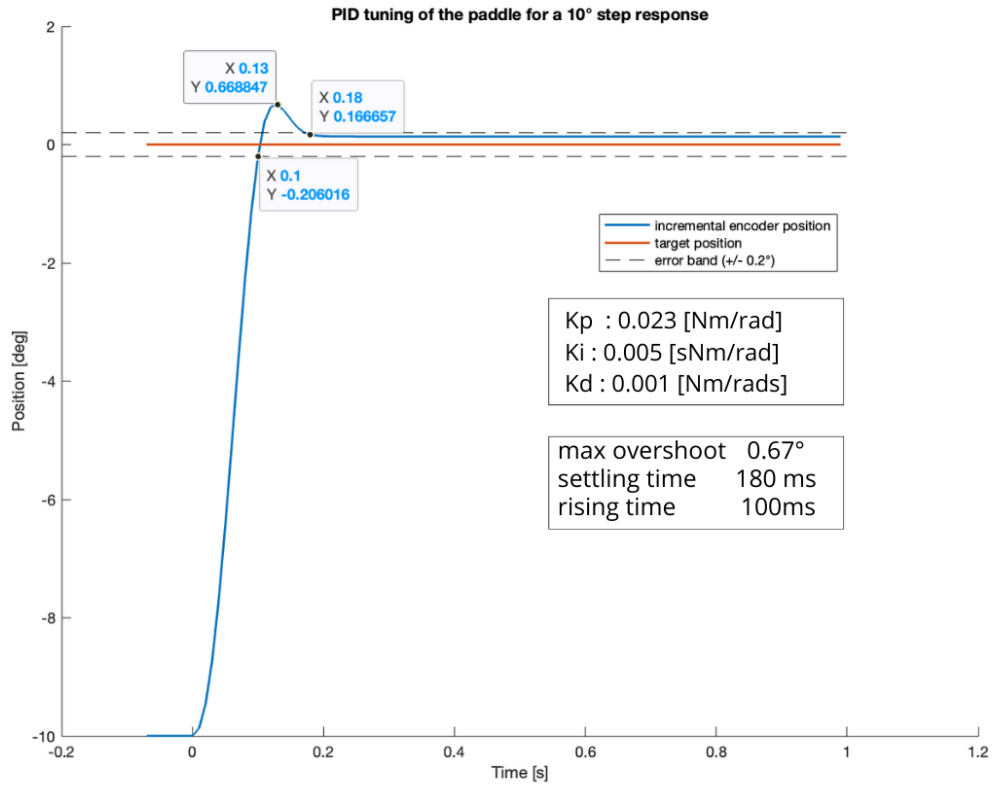
Hand-tuned parameters values can be found in the table 1 in the annex.

With the first set of hand-tuned parameters an optimal behavior for both the incremental encoder sensor and the hall sensor can be obtained. The paddle rise quickly and overshoot and then slowly decreases to the target value. Using the hall sensor the overshoot is higher (as the distance was a little more than 10° the overshoot is still in the required range). The rising time is a little bit smaller and the settling time is similar. [Fig.6]

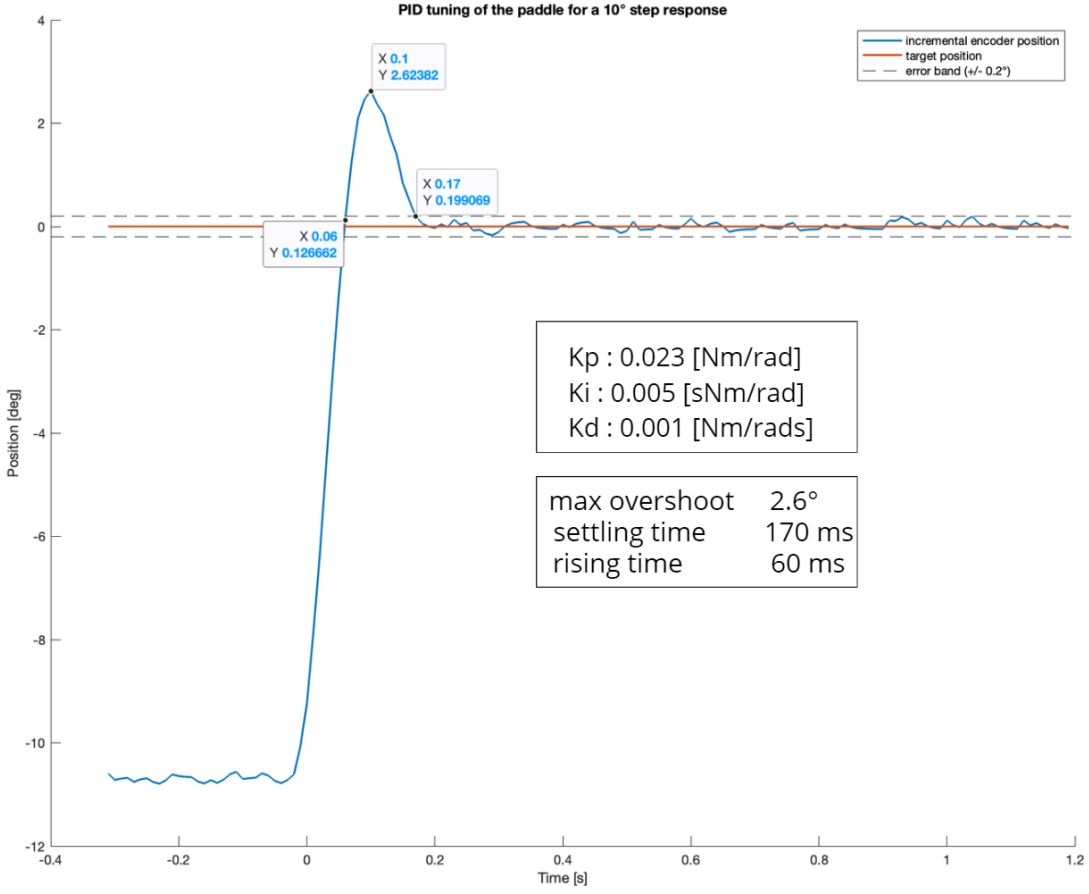
The second set of parameters provide a slower rise with no overshoot. With the same parameters, the hall sensor provide a faster rising and a little overshoot. This set of parameters could be use if overshoot need to be avoided. K_d is in the order of 10 time higher than in the first configuration. K_p is similar. K_i have been increased to faster the convergence close to the targeted value. This example highlight the effect of K_d. [Fig.7]

The last set of parameters explore the effect of K_p and K_i. The K_p is in the range of 2 time the one of the optimal parameters and K_d have been lowered while K_i is higher to manage convergence. In this set up, the paddle describe a huge overshoot and oscillate around the value before settling. If the K_p is too high in comparison to the K_d, the paddle may oscillate without settling or shoot up and become unstable. The performance of the 2 sensor are similar, suggesting that the effect of the noise, the main difference between the two sensor, affect mainly the parameter K_d. [Fig.8]

The main difference between the two sensors of the position is the noise. The hall encoder is much more noisy even after filtering at 20Hz, this mostly affect the K_p as this parameters relies on the derivative on the error which is highly variable when derivative a noise signal. In consequence, using the Hall sensor the paddle have a tendency to overshoot and oscillate more. Note that the hall sensor show a difficulty to follow the paddle when the movement is too fast, increasing the error.

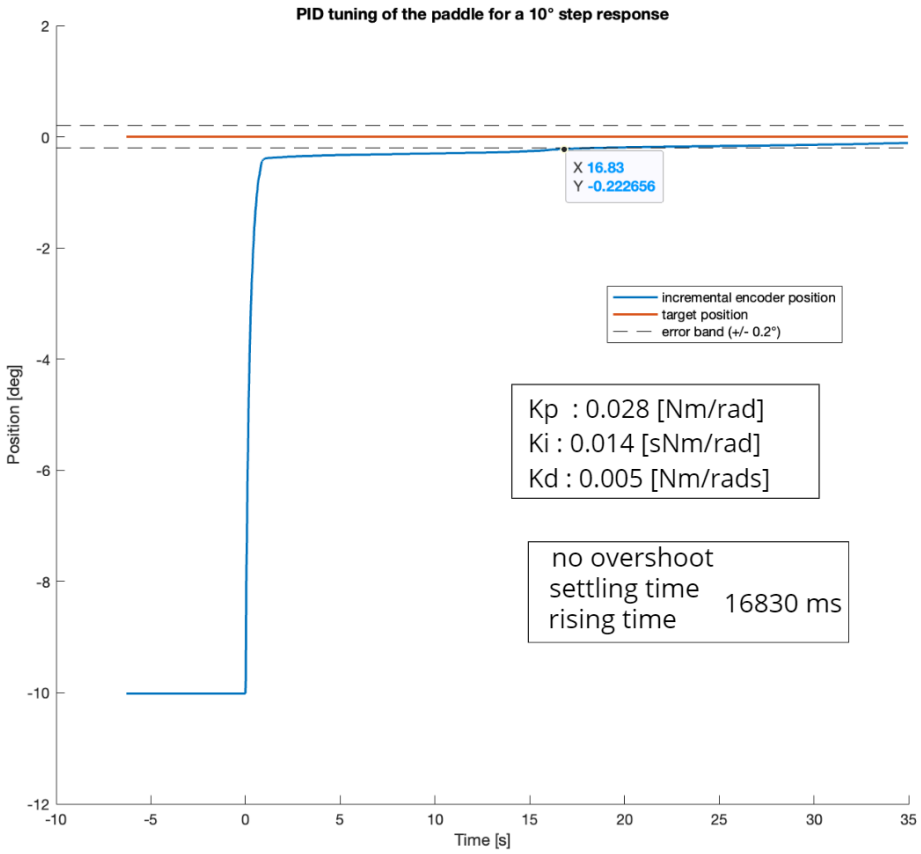


Position of the physical paddle with the incremental encoder sensor

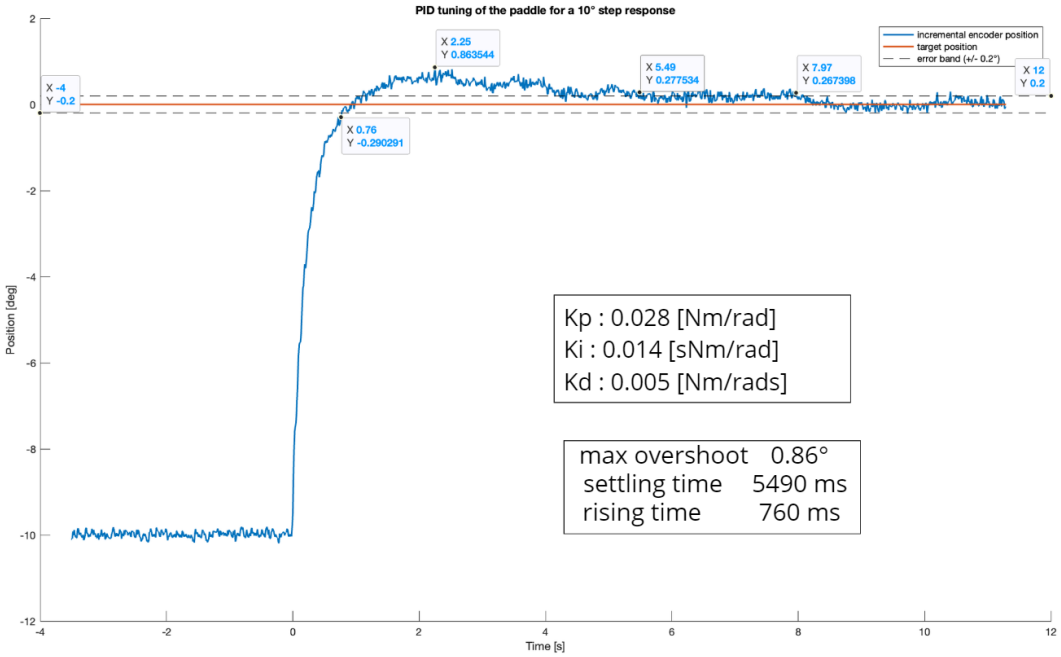


Position of the physical paddle with the Hall sensor

Figure 6: PID tuning for a 10° step response - hand-tuned set of parameters 1 [6]

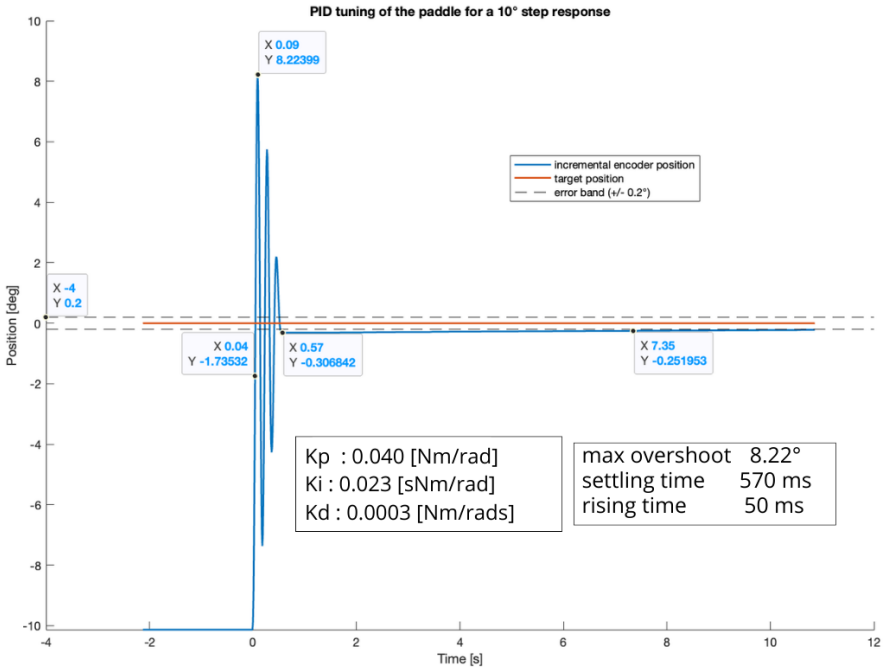


Position of the physical paddle with the incremental encoder sensor

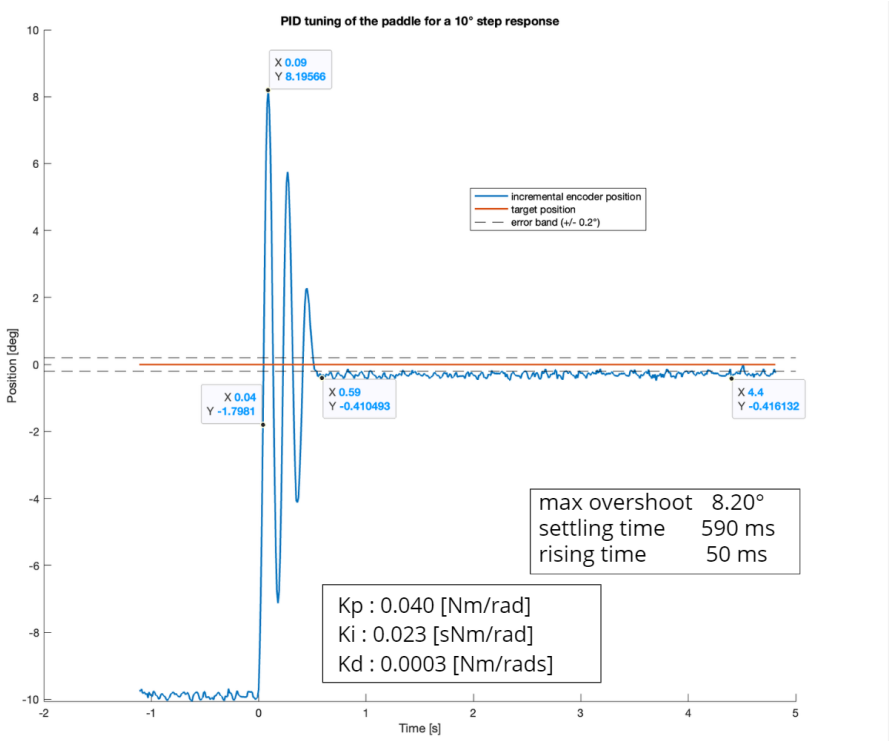


Position of the physical paddle with the Hall sensor

Figure 7: PID tuning for a 10° step response - hand-tuned set of parameters 2 [7]



Position of the physical paddle with the incremental encoder sensor



Position of the physical paddle with the Hall sensor

Figure 8: PID tuning for a 10° step response - hand-tuned set of parameters 3 [8]

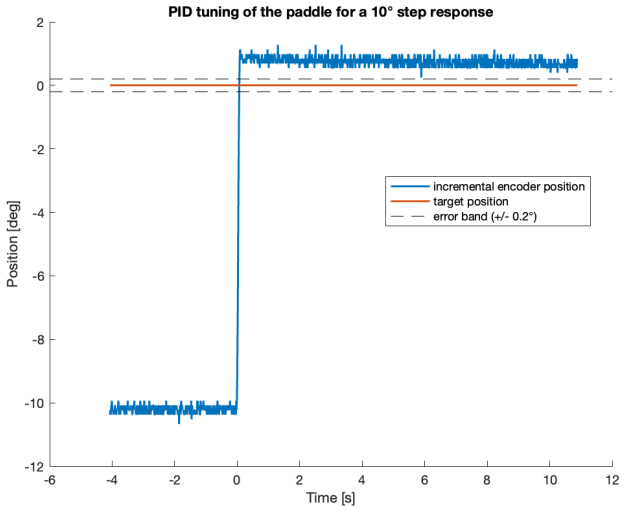
6 Effect of the filter

As explained in the previous part, the main difference between the two sensors is the noise ratio. As show in equ.5 the noisy measure of the position would also impact the integral, and the velocity, affecting the second term of the equation (the derivative). A higher K_d would increase the difference between the two sensor. While the noise on the position is already bad enough on the hall sensor and prevent a good implementation of a PID controller on the physical paddle, it only gets worse as the noise increases during numerical differentiation affecting the noise on the derivative term even more than the position.

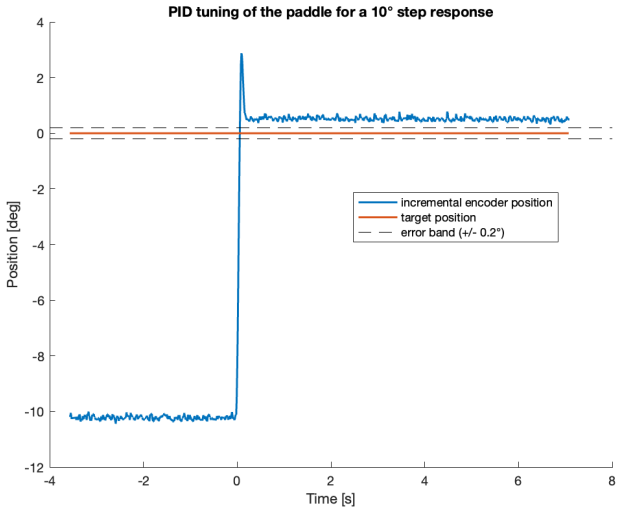
Filtering doesn't really have a effect if we use the incremental encoder sensor as it is already really stable. For the hall sensor, filtering has an effect.

The effect of filtering is tested by taking the best set of parameters and testing different cut off value. The result is shown in the figure 9.
With no filter applied to the hall sensor position, the noise is high. The paddle rise fast, shows no overshoot and decrease slowly to the error band. The model is seen to settle at a higher value and has difficulty to be precise.
With a cut off frequency of 20 Hz and the same parameter, the model show a clear overshoot and settle around the error band.
Interestingly, if the cut-off frequency is too high the paddle oscillate and does not manage to settle at the targeted value, even start to shoot up at every oscillation. It may be because the filter can catch up with the fast rise and only noise stay. A cut off frequency of 20Hz was sufficient to reduce the noise on an acceptable range when the paddle was still.

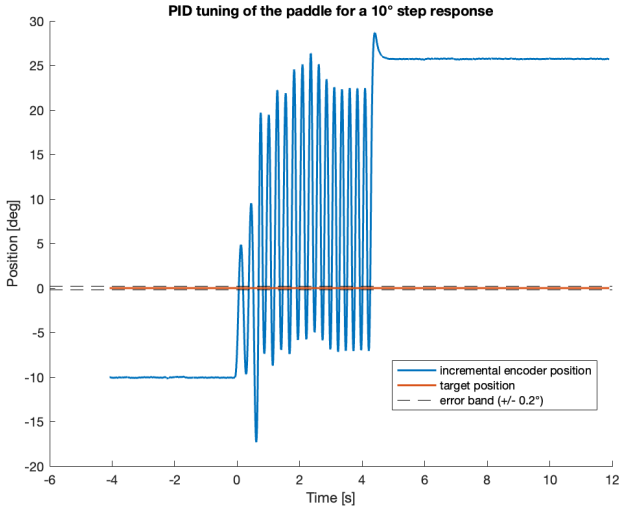
Here, the position was filtered and the result was used to calculate the speed. Filtering the speed may give a more stable result for the second term of the equation and provide better result. However if the position isn't filtered the noise will still affect the two other terms of the equation.



Hall sensor position no filter



Hall sensor position filtered cutoff at 20 Hz



Hall sensor position filtered cutoff at 2 Hz

Figure 9: Effect of different cut off frequency with best set of parameters [9]

7 PID tuning with Ziegler-Nichols method

The parameters of the PID were then tuned using the Ziegler Nichols method. First, K_i and K_d are set to 0 while K_p is slowly increased until the paddle oscillates constantly. The observed K_p for constant oscillation is called K_u and will be the maximum gain, and let's call T_u the period of the oscillation 10

These parameters can be extracted from the physical implementation.

$$T_u = 0.217[s] \quad Ku = 0.026[Nm/rad] \tag{6}$$

The K parameter can then be computed using different equations depending on the precise wanted Z-N behavior.

$$\text{Classical Z-N control : } K_p = 0.6K_u \quad K_i = 1.2K_u/T_u \quad K_d = 0.075K_uT_u \tag{7}$$

$$\text{No overshoot Z-N control : } K_p = 0.2K_u \quad K_i = 0.4K_u/T_u \quad K_d = 0.066K_uT_u \tag{8}$$

$$\text{Some overshoot Z-N control : } K_p = 0.33K_u \quad K_i = 0.66K_u/T_u \quad K_d = 0.11K_uT_u \tag{9}$$

Below are the graphs for the oscillation and the physical implementation of the paddle for the Z-N PID control.

The classical Z-N tuning offers remarkable performance with a very low settling and rising time while the overshoot is slightly bigger than the restrictions 11. The Z-N "no overshoot" conditions does exactly what it's name suggest, while the rising and settling time is higher than the classical condition it does not overshoot which can be desirable for some applications 11.

However the "some overshoot" condition shows a peculiar behavior. While the overshoot is well reduced compared to the classical condition, it's settling time is way too long and shows an oscillating behavior 11.

Overall the Ziegler-Nichols method for PID tuning offers a good technique that can be used as a first step of tuning and then hand refined to restrict the model for specific imposed constraints depending on the wanted behavior.

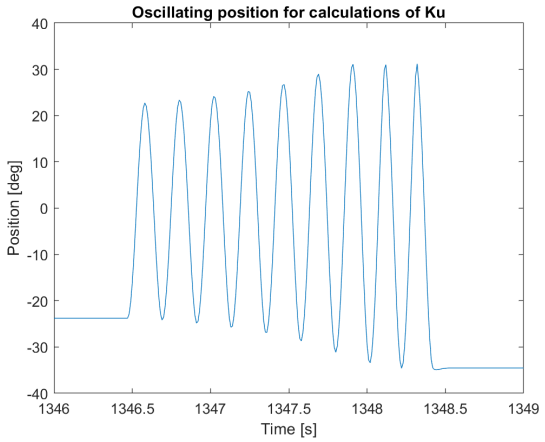
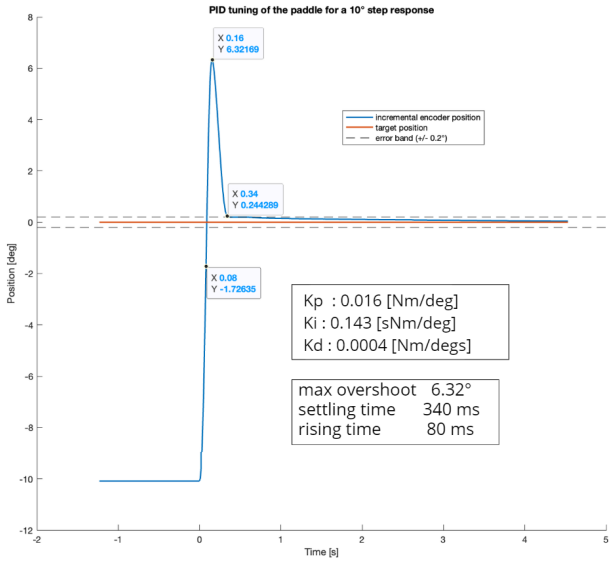
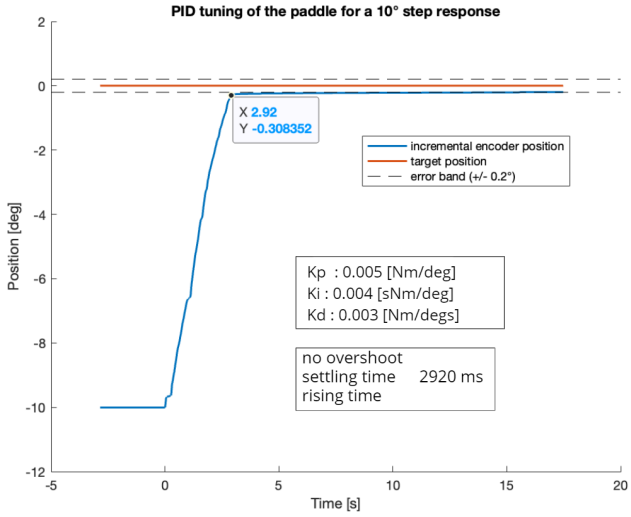


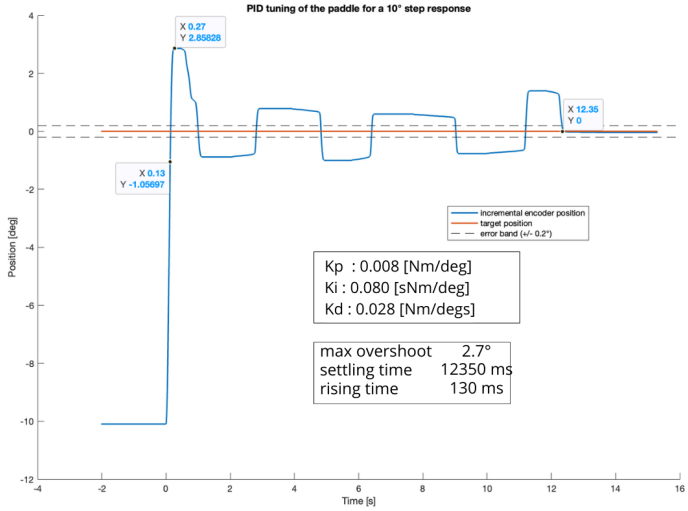
Figure 10: Oscillating position of the Paddle for calculations of K_u and T_u [10]



Classical ZN



No overshoot ZN



Some overshoot ZN

Figure 11: PID tuning for a 10° step response - ZN methods [11]

8 Simulation comparison

The plots of the position can then be compared with the output of the simulation by giving the same set of K parameters that have been either hand tuned of calculated using Ziegler-Nichols tuning method. First, comparing 6 and 12,then 11.1 and 13 the difference between the simulation and the physical implementation is the same. The overshoot and the oscillating time on the simulations are drastically bigger due to the lack of dry friction in the model. Indeed, without a good model of dry friction, no force opposes the movement of the simulated paddle thus making the initial overshoot bigger and the oscillating time longer whereas in the physical paddle, these forces oppose the movement and reduce both of these factors.
This time however, compared to 2, the simulation respects the physical constraints of the paddle as it moves in an acceptable range and the motor torques are physically implementable.

As discussed previously, the simulation would need to take into account both the physical constraints of the model and a good model of friction in order to be more useful when predicting the real physics of the paddle.

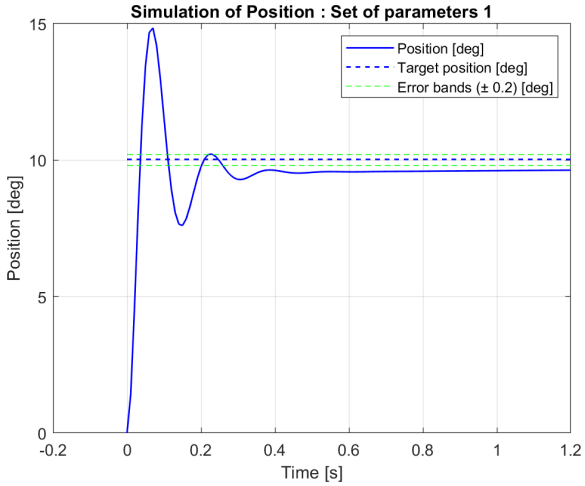


Figure 12: Simulation with PID control for set of parameters 1 [12]

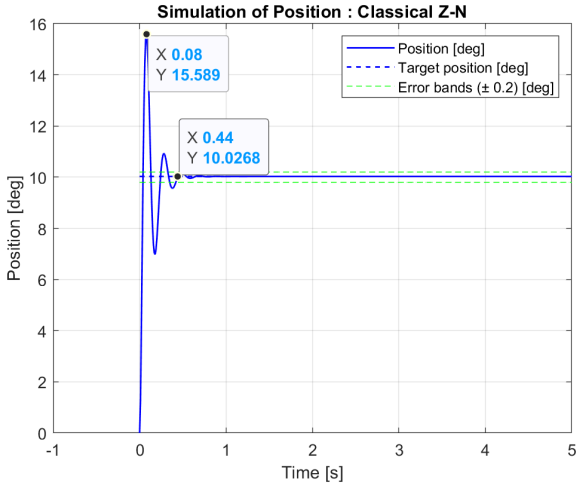


Figure 13: Simulation with PID control for classical Z-N tuning [13]

Comparison between physical implementation and simulation for given sets of parameters

9 Annex

Θ : paddle position $[rad]$
$\dot{\Theta}$: paddle speed $[rad/s]$
$\ddot{\Theta}$: paddle acceleration $[rad/s^2]$
Θ_m : mesured position $[rad]$
Θ_d : desired/targeted position $[rad]$

J_{eq} : Equivalent total inertia, computed on the paddle side $2.80 * 10^{-4}[kg * m^2]$
J_{rotor} : Rotor inertia $1.10 * 10^{-6}[kg * m^2]$
J_{paddle} : Paddle inertia $2.05 * 10^{-7}[kg * m^2]$
$J_{wormscrew}$: Worm screw inertia $0.14 * 10^{-6}[kg * m^2]$

B_{eq} : Equivalent viscous friction coefficient, computed on the paddle side $1.77 * 10^{-4}[Nm/(rad/s)]$
B_{motor} : Viscous friction of the ball bearings $7.90 * 10^{-7}[Nm/(rad/s)]$
B_{paddle} : Paddle ball bearing viscous friction $0[Nm * s/rad])$

m_{paddle} : Paddle mass $0.077[kg]$.
l : Distance between the center of rotation and the center of mass of the paddle $0.025[m]$.
g : Gravity $9.81[N/kg]$.
R : Reduction ratio of the cable transmission 15[]

F_{motor} : Force applyed by the motor $[N]$
F_{ext} : external force $[N]$
τ_m : torque of the motor $[Nm]$

K_p : proportional component of the PID $[Nm/rad]$
K_i : inertial component of the PID$[s \cdot Nm/rad]$
K_d : derivative component of the PID$[Nm/rad \cdot s]$

name	**Kp (e-02)**	**Ki (e-02)**	**Kd (e-02)**
simulation	1.490	2.498	0.100
Hand tuning 1 (optimal)	2.292	0.573	0.043
Hand tuning 2 (slow no overshoot)	2.865	1.432	0.573
Hand tuning 3 (fast oscillation)	4.011	2.292	0.032
ZN classical PID	1.581	14.324	0.043
ZN no overshoot	0.527	0.485	0.382
ZN some overshoot	0.870	8.021	0.063
constant oscillation	2.6	0	0

Table 1: Parameters for the PID [1]

List of Figures

List of Tables