ELEC-E8101 Group project: Lab B report Group 11

HOUILLE Florent, MONTE SASTRE Enrique, SOULARD Yoann, WANG Tong

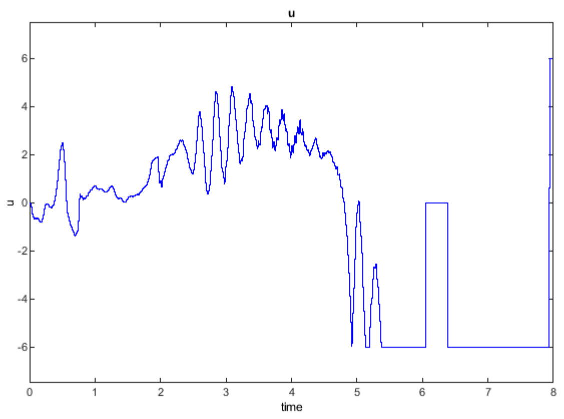
November 20, 2019

Reporting of Task 5.1

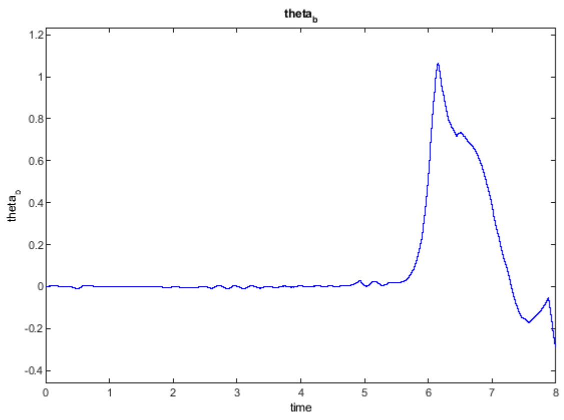
We implemented our PID with the values we obtained in lab A. Unfortunately, the sampling time (42ms) was not working at all. The robot's wheels were just stuck, making a nearly imperciptible back and forth movement. If the sampling time would have been too high, the system would have been unstable and the command would have maxed the limit out right at the beginning. So we didn't find a good reason to explain this behaviour. We tried to bend the robot to force the wheels in one particular direction, but still they are alterning quickly between forward and backward movement. So instead we used the minimum sampling time (5 ms) and tuned the PID's parameter to finally obtain :

$$K_p = -250; K_i = -500; K_d = -0.6$$

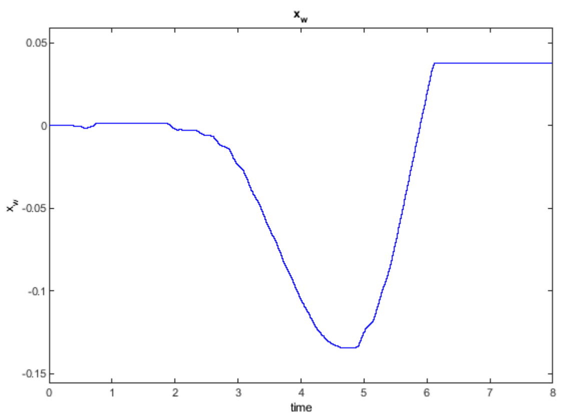
Actually these parameters were difficult to tune because the robot's behaviour is highly depend of its initial angle. For instance sometimes it could stay up for more than 5 seconds and other times it fell in only 1 second with the exact same parameters set. We decreased K_p to get a smoother behaviour around the null angle (i.e. improve stability, prevent jitters). We decreased integral term to prevent overshoot (once the robot reached the null angle, it was falling the other side). We increased derivative term so that the robot is better "aware" that it is going to fall down and can act before it's too late. Here is a result we recorded with Simulink :



(a) $u(t)$



(b) $\theta_b(t)$



(c) $x_w(t)$

Figure 1: Simulation results (the robot falls after 5.5 seconds)

The command goes at its maximum value at $t = 5s$. It corresponds to the time when the robot stops moving (i.e. x_w is stabilized), before going the other way. We think this brutal command is the reason why the robot fell down. One can also notice that the command is oscillating. This might be due to high frequency noises coming from the sensors and then transmitted to the PID's derivative action (we didn't use any filter).

Reporting of Task 5.2

5.2.1 Here are the controllability $\mathcal{C}$ and observability $\mathcal{O}$ matrices in parametric and numeric form (according to lab A state-space):

$$\mathcal{C} = \begin{bmatrix} B & AB & AAB \end{bmatrix} = 1.10^5 \begin{bmatrix} 0 & 0 & 4.10^{-4} & \approx 0 \\ 4.10^{-4} & \approx 0 & -0.3087 & -1.44.10^{-2} \\ 0 & 0 & -1.6.10^{-3} & \approx 0 \\ -1.6.10^{-3} & \approx 0 & 1.3216 & 6.15.10^{-2} \end{bmatrix}$$

$$\mathcal{O} = \begin{bmatrix} C \\ AC \\ AAC \\ AAAC \end{bmatrix} = 1.10^6 \begin{bmatrix} 0 & 0 & \approx 0 & 0 \\ 0 & 0 & 0 & \approx 0 \\ 0 & 3.3.10^{-3} & 1.10^{-4} & -1.10^{-4} \\ 0 & -2.7943 & -0.0262 & 0.0587 \end{bmatrix}$$

$$\Delta\mathcal{C} = -1.10^{-5} \quad \Delta\mathcal{O} = 0$$

$\mathcal{C}$ matrice is full-rank (4), therefore the system is controllable. $\mathcal{O}$ matrice is not full-rank, therefore the system is non-observable.

5.2.2 a) Our system is non-observable. The system is unobservable if there is at least one state we can't observe. When we represented the system state space form, C matrix had 3 null columns (out of 4). That's why our system is unobservable. This can be due to some missing physical sensors.

5.2.2 b) In order to have an observable system, we just need to make the component x_w observable. Once x_w and θ_b are observable, we can then derive other states $\dot{x}_w$ and $\dot{\theta}_b$. We can not add a sensor to physcally measure x_w but one can get it from the wheels' angular speed and diameter.

5.2.2 c) We just added a component to C, therefore we will have two outputs :

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Reporting of Task 5.3

5.3.1 Here are the eigen values of the matrix A before any modification (some are approximated) :

$$p_1 = 0 \qquad p_2 = -843.4 \qquad p_3 = -5.6 \qquad p_4 = 5.7$$

One can see that one pole is an integrator ($p_1 = 0$), another is unstable ($p_4 > 0$), another is stable but slow (p_3) and the last is stable and fast, we can keep it as is (p_2). First, we want all the poles to be stable (< 0).Then, we want to approximate the system as being second order. Therefore, we can choose two dominant poles such that two other poles can be negligated. One can say that a ratio of 6 is enough to declare a pole as negligeable. Therefore, we can take :

$$p_1 = -2 \qquad p_2 = -843.4 \qquad p_3 = -12 \qquad p_4 = -3$$

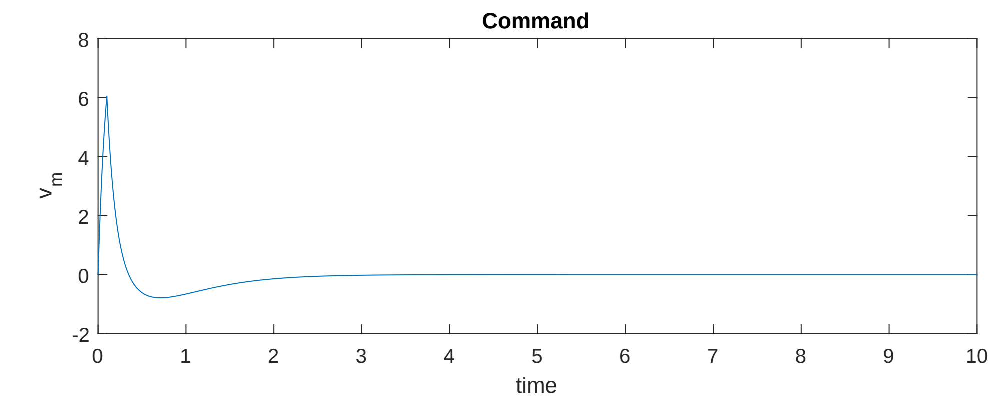
The ratio between poles is the only thing that matters, but we also took poles with small magnitude in order to have a realistic controller (i.e. p_1 and p_4 are slow enough so that they won't require the controller to provide too much energy). So now p_2 and p_3 can be negligated compared to the slowest pole p_1. We end up with the following poles:

$$p_1 = -2 \qquad p_2 = -3$$

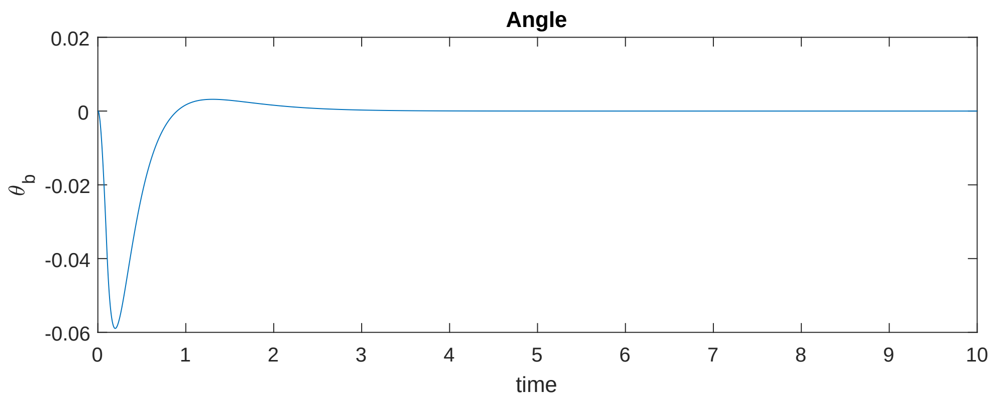
5.3.2 We want to find a gain matrix K such that eigen values of $A - BK$ are equals to the poles previsouly chosen. For that, one can use the Matlab function *place* to which we provide mtrices A,B and the chosen poles vector. We obtain the following values:

$$K = \begin{bmatrix} -47.5 & -64.7 & -103.4 & -15.2 \end{bmatrix}$$

5.3.3 We then implement this SS controller in Simulink and obtain the following result:



(a) $u(t)$



(b) $\theta_b(t)$

Figure 2: Simulation results with first SS controller

One can notice that the system is stable (the robot doesn't fall). Moreover, the command is realistic as the voltage does not go beyond 6 Volts (batteries full capacity is 7.2V).

Reporting of Task 5.4

5.4.1 Here is the full observer L matrix:

$$L = \begin{bmatrix} 2.93 & 0.82 \\ -0.58 & 0.87 \\ 2.22 & 12.11 \\ 10.78 & 55.59 \end{bmatrix}$$

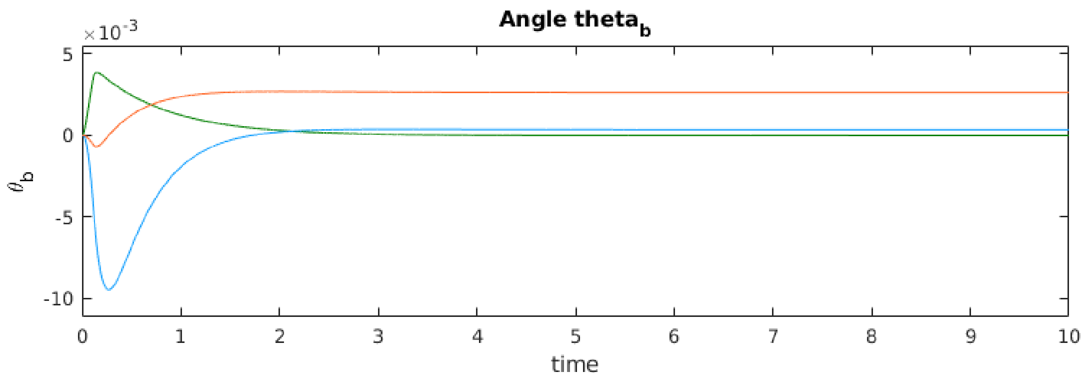
Unfortunately, we were not able to compute the reduced order estimator. The reason is that we found some problems following the guide with the notations, so finally when trying to compute M_1, M_2 etc, it was not possible because of matrix dimension problems in multiplications. However, the code of what we tried is attached. First thing we did is to select a V such that the z obtained is an identity matrix. V should be a 3x4 matrix in order to have an invertible T. Therefore :

$$V = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

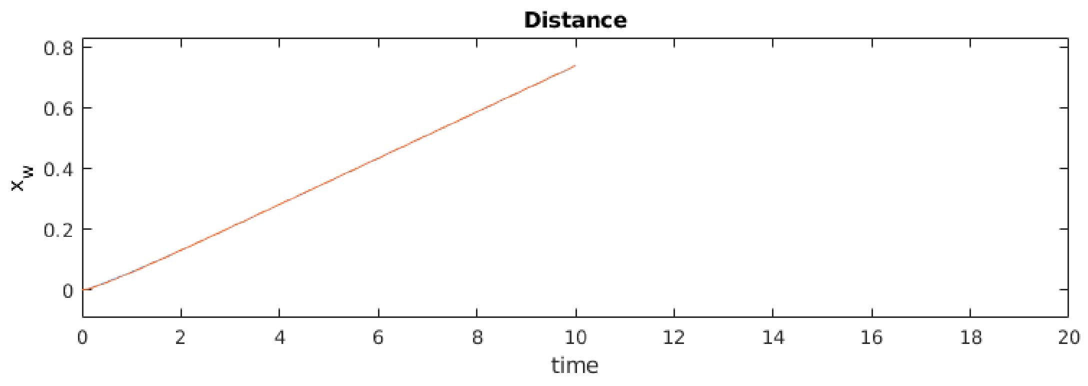
Therefore, future calculations will be easier with z as an identity matrix.

Later, we just tried to follow the steps (and understand them) in the guideline and when we reached the M_1 parameter calculation it was not possible to compute it because of matrix dimension error in multiplication. We suspect the problem to be a wrong choice in our matrices but we tried to read again and again the guideline and despite this we lost ourselves in the variables notations.

5.4.2 Here are the simulations (the reduced observer parameters come from teacher's values):



(a) $\theta_b(t)$ in green, $\widehat{\theta_b}^{full}(t)$ in blue, $\widehat{\theta_b}^{reduced}(t)$ in red



(b) $x_w(t)$ in green, $\widehat{x_w}^{full}(t)$ in blue, $\widehat{x_w}^{reduced}(t)$ in red

Figure 3: Simulation results with full and reduced observers

We can observe that θ_b with full Luenberger observer goes to opposite direction compared to true θ_b. We think this might be due to some negative gain in our L matrix. Furthermore, we can also observe that our robot is going away to an infinite position to stay balanced. The reduced order observer has got a more important error than the full one. This is logical as it is, by definition, less accurate. The fact that it has not been computed with our system's values may also increase the final error.

5.4.3 By measurements with cursors, we can compute:

$$\left|\theta_b - \widehat{\theta_b}^{full}\right| = (3.86 + 6.485) \cdot 10^{-3} = 10,345 \cdot 10^{-3} rad$$

The absolute error of angle is quite small.

We can also compute :

$$\left|x_w - \widehat{x_w}^{full}\right| \approx 0m$$

Here, there is no difference between both positions of the robot. We can conclude that the system is alaways compensating same error sign on θ_b so he is going to same direction to compensate this one. That's why our robot is going to an infinite position.

Reporting of Task 5.5

5.5.1 We formed a state-space system with the matrices A,B,C and D. Then we discretized this ss into a discrete one with a sampling time of $h = 5ms$ and the zero-holder method using Matlab's c2d function.

5.5.2 Then we map all poles chosen previously from s- to z-domain applying the equivalence formula $z_i = e^{p_i h}$.

Lab B conclusion

During this lab we tested our Lab A's PID on the robot and obtained quite good result for this kind of controller. We tried to improve it before designing a more advanced state-space controller by pole placement method. It worked on simulation but we didn't try it on the robot. Then we tried to design a state observer. We obtained the full order one, but unfortunately didn't manage to compute the reduced order one. We should work deeper on this topic for the exam... A lack of time prevented us from doing some simulations and tests over the robot.