CMN-O – Laboratory – Control of an inverted pendulum

1st Pedro Baptista
IST Number - 96302
pedro.maria.baptista@tecnico.ulisboa.pt

2nd Filipe Vaz
IST Number - 100520
filipe.vaz@tecnico.ulisboa.pt

3rd Ricardo Francisco
IST Number - 103093
ricardo.n.francisco@tecnico.ulisboa.pt

***Abstract*—This project, part of the CMN course, focuses on designing and implementing a state-feedback controller for an inverted pendulum. The controller and a state observer are developed, validated in SIMULINK, and tested on a real system. The study emphasizes the integration of control algorithms with physical dynamics, highlighting their crucial role in system stabilization. A simulation was created, and through successive lab sessions, the simulation was fine-tuned to closely approximate the real system. Additionally, a deadzone compensation controller, a type of nonlinear controller, was incorporated to address specific challenges. The project required students to define the desired controller objectives and systematically pursue these goals through rigorous testing and optimization, both in simulation and on the real system, to achieve the best possible results.**

***Index Terms*—LQR (Linear Quadratic Regulator), LQG (Linear Quadratic Gaussian), Nonlinear Control, State-Feedback Control, Multivariable Systems, Dead zone Compensation**

I. Introduction

This project focuses on analyzing both the real and linearized models of an inverted pendulum, a naturally nonlinear system. The main objectives are to implement a state-feedback controller using the LQR method and design a Kalman filter observer. The controller and observer will be fine-tuned to optimize system performance, ensuring stability and robustness in the real system.

II. Lab session Planning

A. *Q1 - Characterization of the Model*

The model consists of an inverted pendulum that is linearized around the upright position. It has five state variables: the bar's horizontal angle (x_1) and angular rate (x_2), the pendulum's vertical angle (x_3) and angular rate (x_4), and the motor current (x_5). The system's input is the motor voltage, which is limited to an absolute value of 5 volts.

To analyze the system, the eigenvalues are first computed using the following MATLAB script:

```
load('matfiles\IP_MODEL.mat')

% Question 1
fprintf('Q1 - Open-loop system eigenvalues: ');
fprintf('%f ', eig(A).');
fprintf('\n');
```

The eigenvalues of the dynamics matrix are $\{0, -737.3184, -19.3673, -5.7612, 6.9974\}$. Based on these values, it is concluded that the system is unstable due to the presence of a positive eigenvalue. Furthermore, since all eigenvalues are real, the system does not exhibit oscillatory behavior, although this may not necessarily be the case in the real system. Additionally, the presence of an eigenvalue at zero indicates that the system includes an integrator.

B. *Q2 - Characterization of the Open-Loop System in Terms of Controllability*

The rank of the controllability matrix is determined using the following MATLAB script:

```
% Question 2
fprintf('Q2 - Rank of controllability matrix: %f \n'
    , rank(ctrb(A,B)));
```

The computed rank of the controllability matrix is 5, which corresponds to the system's state dimension. Therefore, it is concluded that the system is fully controllable.

C. *Q3 - Characterization of the Open-Loop System in Terms of Observability*

The observability of the system is analyzed using the following MATLAB script:

```
% Question 3
C1 = [0 0 1 0 0];  % C matrix for measuring only the
     pendulum angle (x3)
fprintf('Q3: \n')
fprintf('Observability matrix for case 1 (C = [0 0 1
     0 0]): %f \n', rank(obsv(A, C1)));

C2 = [1 0 0 0 0; 0 0 1 0 0];  % C matrix for
    measuring x1 and x3
fprintf('Observability matrix for case 2 (C = [1 0
     0 0; 0 0 1 0 0]): %f \n', rank(obsv(A, C2)));
```

When only the pendulum angle (x_3) is measured, the output matrix is defined as:

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

In this case, the observability matrix has rank 4, which is lower than the system's state dimension (5), indicating that the system is not fully observable.

When both the pendulum angle (x_3) and the horizontal bar angle (x_1) are measured, the output matrix is given by:

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

With this configuration, the system is fully observable.

D. Q4 - Bode Diagram of the Open-Loop System

The Bode diagram of the system is generated using the following MATLAB script:

```
% Question 4
sys = ss(A,B,C,D);
options = bodeoptions;
options.FreqUnits = 'Hz';
figure(1)
bode(sys, options);
```

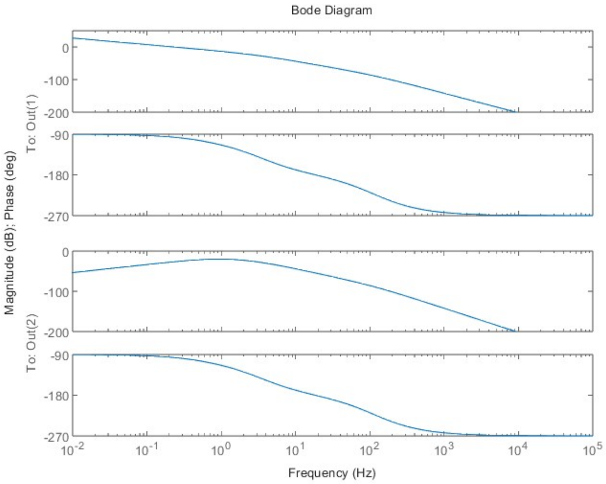


Fig. 1: Open-loop Bode plot

- **Horizontal angle response** (x_1): The system exhibits a cutoff frequency (where the gain drops by -3 dB) at approximately **0.3 Hz**.
- **Pendulum angle response** (x_3): The system maintains a gain consistently lower than **-20 dB**, indicating low sensitivity to frequency variations.
- **Phase characteristics**: Both responses exhibit an initial phase offset of **-90 degrees at the origin**, which aligns with the presence of a pole at the origin. As the frequency increases, the phase decreases further, eventually reaching **-270 degrees**.

From the Bode plot, it can be concluded that the system is generally **insensitive to frequency variations**.

E. Q5 - Computation of the Regulator Gain

Since the system is controllable, the closed-loop poles can be assigned to desired locations. To compute the regulator gains for the infinite-horizon Linear Quadratic Regulator (LQR), the Riccati equation must be solved. This is achieved using MATLAB's `lqr` function, where appropriate weight values are assigned for each state and input. The following MATLAB script demonstrates this process:

```
% Regulator parameters
% State = [alpha(x), alpha rate, Beta(z), Beta rate,
   motor current]
Qr = diag([10,1,10,1,0]); % Weight matrix for state
  variables
Rr = 0.005; % Weight for the input variable
K = lqr(A, B, Qr, Rr); % Calculate feedback gain
```

The computed regulator gain vector for the selected weights is:

$$K = \begin{bmatrix} -44.721360 & -29.961626 & 320.316370 & 50.150446 & 1.2 \end{bmatrix}$$

F. Q6 - Development of a Simulink Block Diagram for System Simulation

To develop a simulation for testing the calculated controller, the Simulink block diagram in figure 2 was formulated:

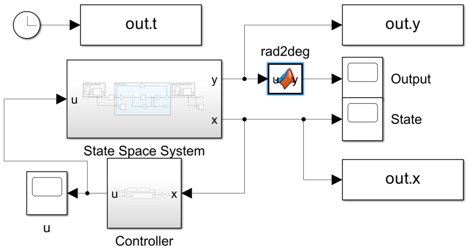


Fig. 2: LQR simulation block diagram

The model outputs variables for plotting the results in a MATLAB script. The MATLAB script to initialize all the variables for the system is shown below, **in addition to the controller macro presented in the previous question**.

```
%% Simulation parameters
x0=[pi/2 0 0.1 0 0].';

% Noise
use_output_noise = 1;
output_noise_power = 10;

use_input_noise = 1;
input_noise_power = 10;

% Deadzone
use_deadzone = 1;
use_deadzone_compensation = 0;

if use_deadzone
    deadzone_lower = -0.38;
    deadzone_higher = 0.25;
else
    deadzone_lower = 0;
    deadzone_higher = 0;
end

T=10; % Time duration of the simulation
out = sim("SimulatorLQRmodel",T);

%% Plotting

% (...)

%% Error
t_steady = out.t > 5;
error = y_degrees(t_steady,:);

RMSE = sqrt(sum(error.^2)/size(error,1))

action = out.u_sat(t_steady,:);

RMSE_action = sqrt(sum(action.^2)/size(action,1))
```

In addition to initializing the system's initial state and controller gains, the macro allows for the addition of white noise to both the system's output and input. It also provides the option to introduce a dead zone and apply the corresponding dead-zone compensation controller. After the simulation, the RMS (Root Mean Square) error is calculated to systematically evaluate the system's performance.

The plotting code is omitted for brevity.

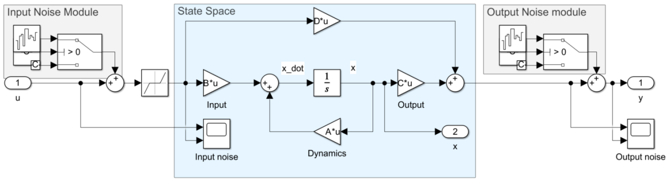


Fig. 3: State-space simulation block diagram

The block diagram illustrated in Figure 3 includes a subsystem that simulates the linear state-space model while providing access to both the states and the output. Additionally, noise modules are incorporated into both the input and output to test the robustness of the closed-loop system. A dead zone is also included in the input to more accurately simulate motor behavior.

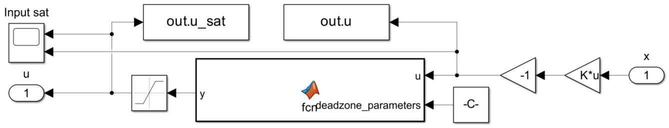


Fig. 4: Controller simulation block diagram

The controller block computes the control action based on the system states (assuming full state accessibility) and incorporates real-world system saturation effects. Additionally, the controller also includes a nonlinear hysteresis component for dead-zone management, as seen in Figure 4.

This simulation was designed to test the LQR gains and includes multiple modules to replicate the real system as accurately as possible. A detailed discussion of these modules, along with an analysis of the simulation results, is provided in Section III.

G. Q7 - Computation of the Estimator Gain

Similarly to the computation of the regulator gain, the estimator gain is determined using the MATLAB `lqe` function. It is important to note that, due to the separation principle, the design of the controller and the observer can be carried out independently. Consequently, adjusting the weights of one does not significantly affect the performance of the other. The optimal gains for the observer are calculated by assigning appropriate weight values, which correspond to the variance of each state and output, as demonstrated in the following MATLAB script:

```
% Estimator parameters
G = eye(size(A)); % Process noise matrix
Qe = eye(size(A)) * 10; % Variance of process errors
Re = eye(2); % Variance of measurement errors
L = lqe(A, G, C, Qe, Re); % Calculate estimator
    gains
```

The computed estimator gain matrix is:

$$L = \begin{bmatrix} 3.2666 & 1.1371 \\ 0.9820 & 11.8725 \\ 1.1371 & 14.2789 \\ 8.0784 & 97.5895 \\ -0.2990 & -3.6109 \end{bmatrix}$$

H. Q8 - Development of a SIMULINK Block Diagram for Controlled System Simulation Using the Designed Controller and Observer

For the simulation with the observer, this model aims to closely replicate the real system by incorporating the necessary observer to estimate the system states. Additionally, a subsystem is included to compare slight different observers state estimates with the actual system states, as shown in Figure 5.

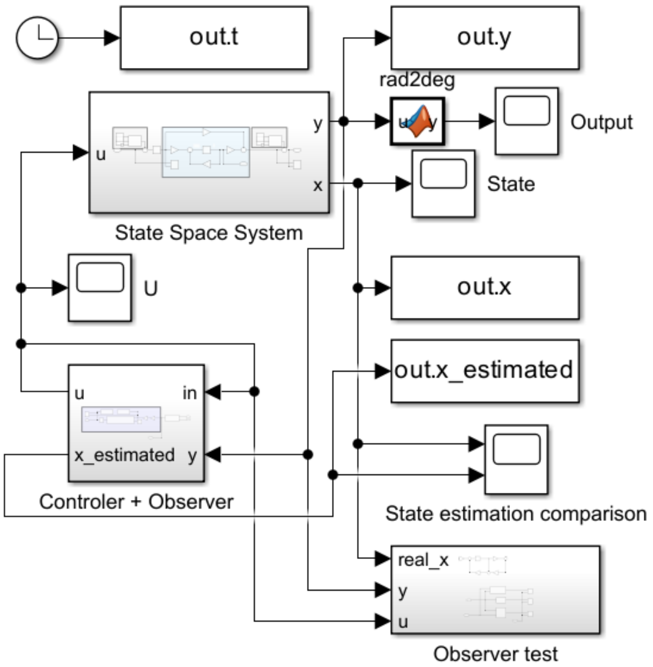


Fig. 5: LQG simulation block diagram

As for the macro, it extends the LQR simulation by incorporating the observer computation step discussed in the previous question.

The combined controller and observer block diagram is shown in Figure 6. In this setup, observer dynamics matrix is given by $A - BK - LC$ (the controller's "A" matrix), the input matrix is L (the controller's "B" matrix), and the output equation is $-K$ (the controller's "C" matrix).

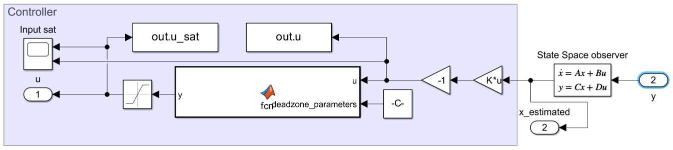


Fig. 6: Simulation LQG controller block diagram

Same for the LQR simulations the details of the implementation and results will be shown in the next section.

III. Lab session Simulation

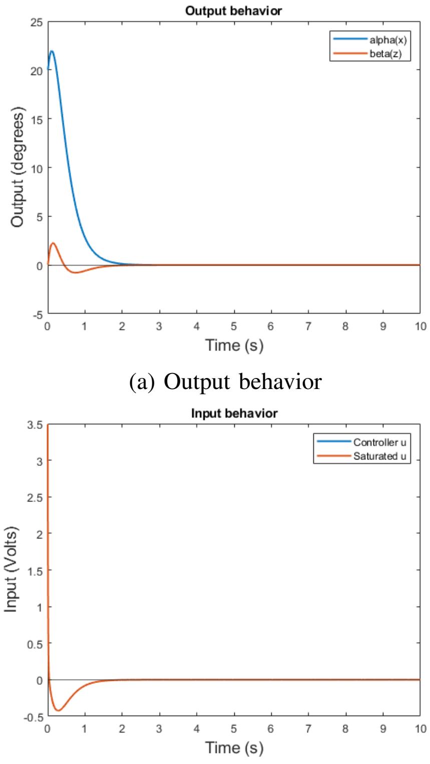
Since time in the lab with the real system was limited, simulations were also conducted at home, but using a Simulink version of the lab pendulum. For this end, the same parameters used in the lab were put under simulation, all with the same Qr and Rr parameters, but with different ones for dead zone consideration, handling and perturbations.

Before starting the simulations, the objective of the control must be defined. The goal is to achieve and maintain equilibrium (state at zero) while ensuring a fast system with a settling time of less than 3 seconds and minimal oscillations. These oscillations should correspond to an RMS error of less than 5 degrees (with respect to state 0), ensuring a highly effective system. Additionally, once these parameters are met, the control input effort should be minimized if possible.

LQR Simulations

To begin the simulations, it is advisable to start with a simple approach. Initially, a basic feedback loop is established between the LQR controller and the linearized system. So using the LQR simulation explained in question 7, the system was put in an non-equilibrium state to observe the controller's performance. After some tuning the requirements are easily met with the following weights: $Qr = [10 \quad 1 \quad 10 \quad 1 \quad 0]^T$ and $Rr = 0.1$. The results can be observed in fig 7

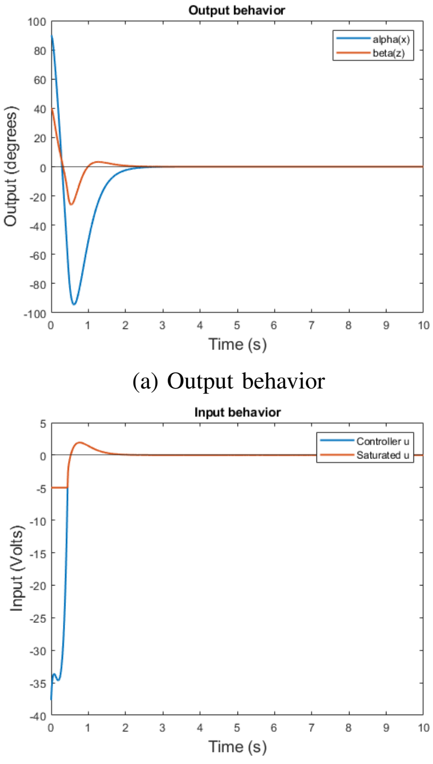
The input weight could be reduced to achieve a faster response. However, for initial states that are farther from equilibrium, the input will saturate, as demonstrated in Figure 8. Therefore, the input weight should only be reduced moderately to avoid excessive saturation.



(a) Output behavior

(b) Input behavior

Fig. 7: Simulation with initial $x_1 = 20\ (degrees)$ and $x_3 = 0$



(a) Output behavior

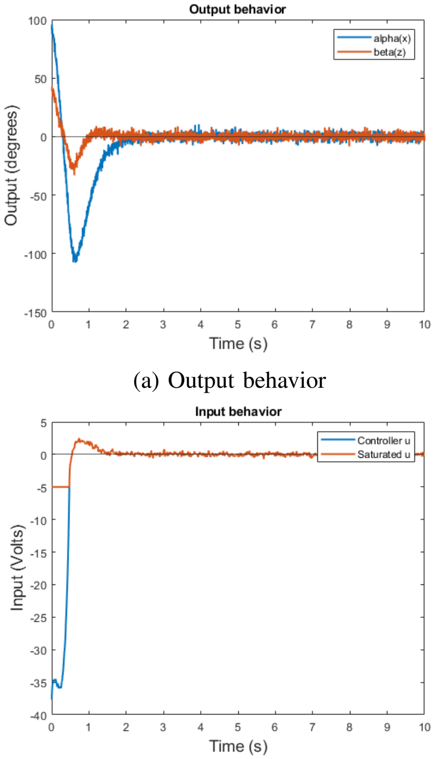
(b) Input behavior

Fig. 8: Simulation with initial $x_1 = 90\ (degrees)$ and $x_3 = 35\ (degrees)$

For both of these cases, the requirements set in the beginning are met.

On a curious note, it was found through iterative simulation that the maximum pendulum angle allowed by this simulation was 40 degrees. Any value above this threshold would render the system unstable due to the distance from the equilibrium point. However, this does not seem to be the case for the horizontal angle, as it can attain larger values.

Now introducing noise to the model seen in figure 9 the system still behaves as expected, and even without a filter, it still fulfills the requirements.



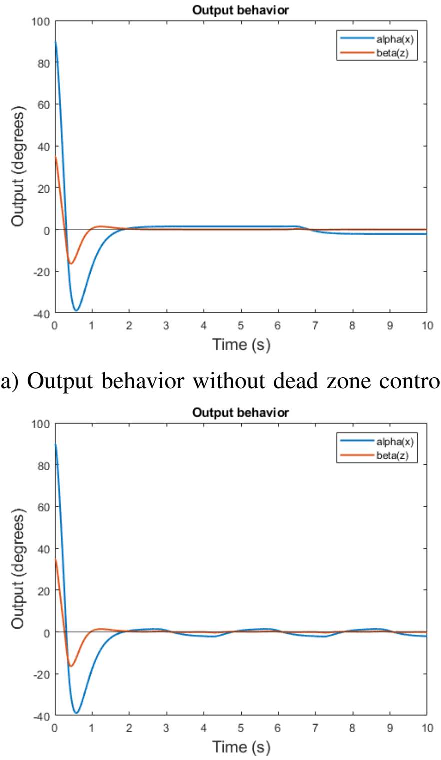
(a) Output behavior

(b) Input behavior

Fig. 9: Simulation with noise with initial $x_1 = 90\ (degrees)$ and $x_3 = 35\ (degrees)$

By that time, a few lab sessions had passed, and the group had already encountered some difficulties with the real system, as will be discussed in Section IV. In that regard, it was decided to apply a dead zone to the input of the simulation, as it is present in the real system. Additionally, a dead zone compensation controller was implemented to ensure that the control input remained outside the dead zone and was therefore always applied. In the real lab the dead zone was roughly calculated and those values were the ones used in the dead zone block in simulink simulation and in the dead zone controller for both the simulation and real system.

To test this new controller, it was first applied to the simple LQR simulation. The comparison can be observed in Figure 10.



(a) Output behavior without dead zone controller

(b) Output behavior with dead zone controller

Fig. 10: Simulation with initial $x_1 = 90\ (degrees)$ and $x_3 = 35\ (degrees)$

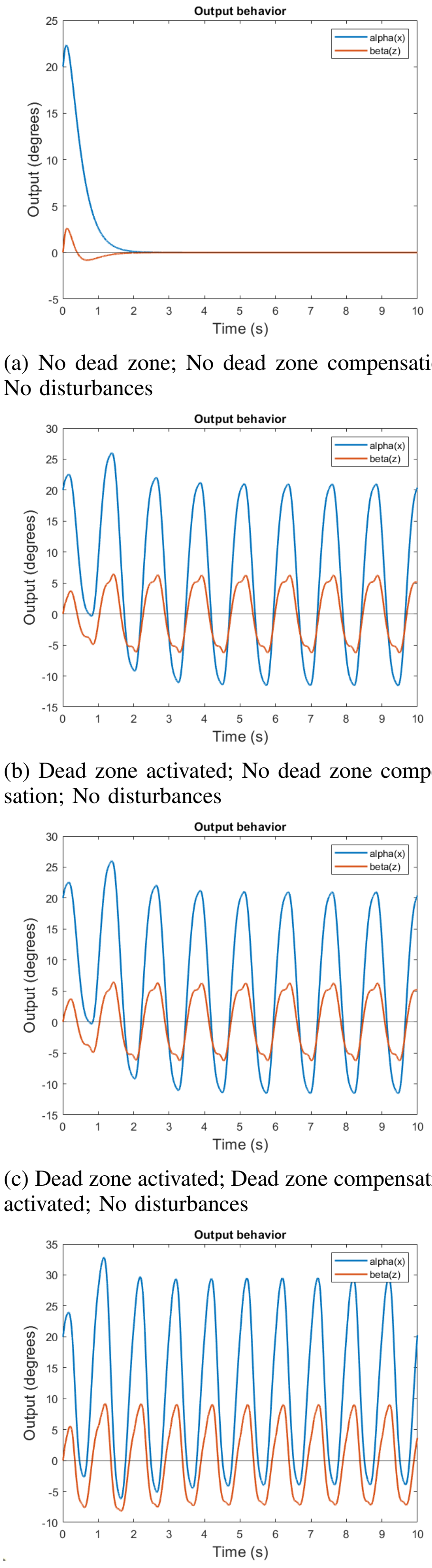
As observed, the dead zone controller provides promising results. It was tested under various configurations in simulation, consistently proving to be more effective. This improvement is evident not only in the graphs but also in the RMS error.

For the simulations shown in the figure, the system without the dead zone controller had RMS errors of 1.7838 for the horizontal angle and 0.0740 for the pendulum angle. With the controller, these errors were reduced to 1.3346 and 0.1398, respectively. When noise was added, the controller demonstrated even greater effectiveness. Under the same configuration but without the controller, the errors were 2.7544 for the horizontal angle and 2.5128 for the pendulum angle, whereas with the controller, they were reduced to 2.5691 and 2.5180.

Although the improvement is not drastic, implementing the controller has no drawbacks apart from a very small increase in input effort. This minimal increase, approximately 0.01 volts per second without noise and 0.2 volts per second with noise, is negligible compared to the overall system performance gains. Furthermore, these benefits may be even more pronounced in the real system.

LQG Computer Simulations

In this case, the perturbations used were sequential and continuous square pulses, one per second, each one with a duration of $0.25s$ and with an amplitude of $0.5V$. Since the system is clearly robust to white noise, which in a real scenario



(a) No dead zone; No dead zone compensation; No disturbances

(b) Dead zone activated; No dead zone compensation; No disturbances

(c) Dead zone activated; Dead zone compensation activated; No disturbances

(d) Periodic disturbances with amplitude of 0.5 V; dead zone and dead zone compensation activated

Fig. 11: Computer simulation results: responses for $Q_r = [10 \quad 1 \quad 10 \quad 0 \quad 0]^T$ and $R_r = 0.1$

could be seen as, for example, sensor noise, wind and other natural perturbations, these square impulses simulate more rough disturbances, such as sudden object crashes against the pendulum.

From **Fig. 11** it is possible to conclude that, without a dead zone in the system, it would converge quickly and practically without any overshoot to the point of equilibrium. On the other hand, when a dead zone is present (similar to the one present in the lab system), the system oscillates as expected, but always around a stable point. Finally, when introducing perturbations, which were the square pulses described earlier, both the horizontal and vertical angles grew in terms of amplitude, but the system managed to remain stabilized. The RMS error for the active dead zone and dead zone compensation and enabled perturbations was 17.3906 for the horizontal angle and 5.9621 for the vertical angle.

IV. Lab session Testing

As in the simulations, the control objective remains the same: achieving and maintaining equilibrium (state at zero) with a settling time under 3 seconds and minimal oscillations. The RMS error should stay below 5 degrees, ensuring effectiveness. Once these criteria are met, control effort should be minimized if possible.

After experimentation with different configurations in the simulation and getting insight on the performance impact of the different design options tested, the controller and observer gains were replaced with the ones computed and subsequently tested on the real system.

Initial tests showed that even though the pendulum managed not to fall, it oscillated around the upright equilibrium point with a relatively large amplitude (roughly 40º). That is, the angle of the horizontal bar (α) rapidly fluctuated between two values, which in turn made the angle of the pendulum (β) fluctuate accordingly, as seen in figure 12.

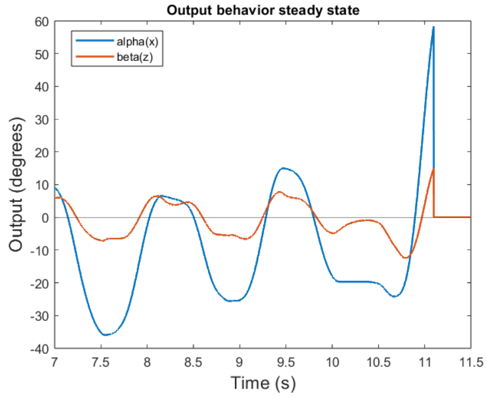


Fig. 12: Lab results: steady state oscillation for $Qr = [10 \quad 1 \quad 10 \quad 0 \quad 0]^T$ and $Rr = 0.1$

Furthermore, when met with a perturbation, the angle of the horizontal bar would perform an instant and large shift to account for it, also observable in figure 12. This indicated that the system was overcompensating for changes in the angle of the pendulum, which resulted in excessively high angular rate in the horizontal angle.

After adjusting the Q_r matrix by decreasing the weight of x_3 (β) and increasing the weights of x_1 (α) and x_5 (motor current), tests revealed that while the previously observed oscillations were reduced, they were still present. At this point, the group realized that due to the existence of the dead zone, some level of oscillation would always persist. Therefore, the primary objective of the controller should be to minimize these oscillations as much as possible.

The next steps were then to implement both a saturation limit and a dead zone for the motor.

The purpose of the saturation is to prevent the motor from outputting voltages that are unreasonably high: the output of the motor only goes up to 5V and the system was attempting to draw higher values; to circumvent this, a limit of 5V was set.

The purpose of the dead zone is to compensate for the motor at low voltages: for values close to zero, the voltage of the motor is incorrectly set to zero; to circumvent this, voltages whose absolute value is lower than $|V_{threshold}|$ are set equal to that (actually there are two threshold values, one positive and the other negative). This threshold was determined using a scope to observe the plot of the angle of the horizontal bar versus the plot of the motor voltage, and estimating the point at which the motor voltage had little to no effect on the angle of the horizontal bar. The calculated voltages for the dead zone were -0.25 V for the beginning and 0.38 V for the end of the dead zone.

The following sessions consisted of an iterative process, alternating between applying simulation-tested values to the real system and observing the results. Necessary adjustments were then made to the simulation to ensure both environments behaved as similarly as possible. This iterative approach continued until the final lab session, always aiming for the best possible results while adhering to the initial project requirements.

Lastly, the simulated system converges to equilibrium relatively quickly; however the oscillations present in the real system are not present in the simulated one.

Only in the last lab session was the group able to achieve more satisfactory results that fulfilled the requirements.

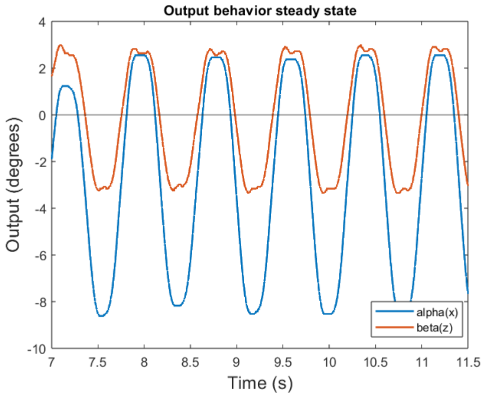
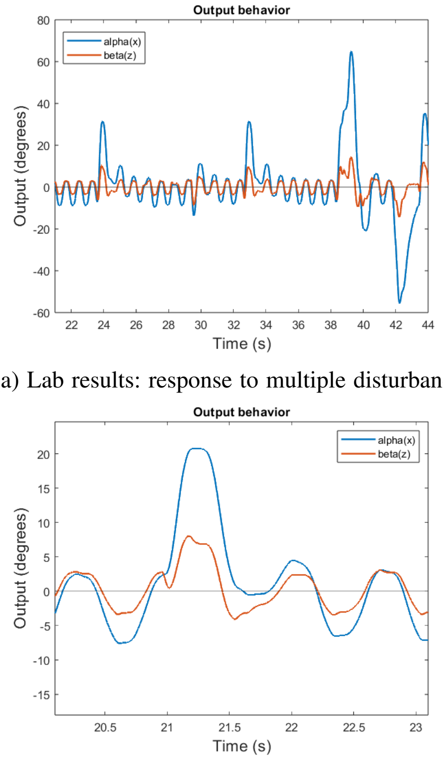


Fig. 13: Lab results: steady-state oscillations for $Q_r = [2000 \quad 0 \quad 50 \quad 0 \quad 0]^T$ and $R_r = 0.05$

From Figure 13, it can be seen that the steady-state oscillations are significantly reduced, complying with the requirements. The RMS errors were 4.1997 for the horizontal angle and 2.25 for the pendulum angle, respectively (both below the threshold of 5). However, there is an observable offset in the horizontal angle, which is present in both the real system and the simulations. Throughout this project, it was not possible to fully mitigate this offset.

For these values, the system also demonstrated a very good response to disturbances, as shown in Figure 14.



(a) Lab results: response to multiple disturbances

(b) Lab results: response to a single disturbance

Fig. 14: Lab results: response to disturbances for $Q_r = [2000 \quad 0 \quad 50 \quad 0 \quad 0]^T$ and $R_r = 0.05$

From Figure 14, the third requirement—a settling time of less than 3 seconds—is also observed to be met. After a sudden increase in the pendulum angle (caused by physically disturbing the system), the system promptly returned to steady-state in approximately 2 seconds.

With these results, all the set requirements were met.

V. Final Computer Simulations

In order to, once again, compare lab performance with system robustness in the simulated environment, final LQG simulations were conducted using the optimal parameters for Qr and Rr found in the latest lab session. This time, the system was subjected to the same simulation criteria, dead zone and perturbation parameters that were set for 11. This yielded the following results:

It is possible to observe that, with these final parameters, the pendulum tends to overshoot more when trying to reach the equilibrium from the initial position. However, it has practically the same settling time as the model with the previous simulated parameters, and is visibly more robust to perturbations. This statement is proven by the RMS error, which was visibly lower for both angles (9.0161 for the horizontal angle and 2.9733 for the vertical angle).

VI. Lab session reviewing

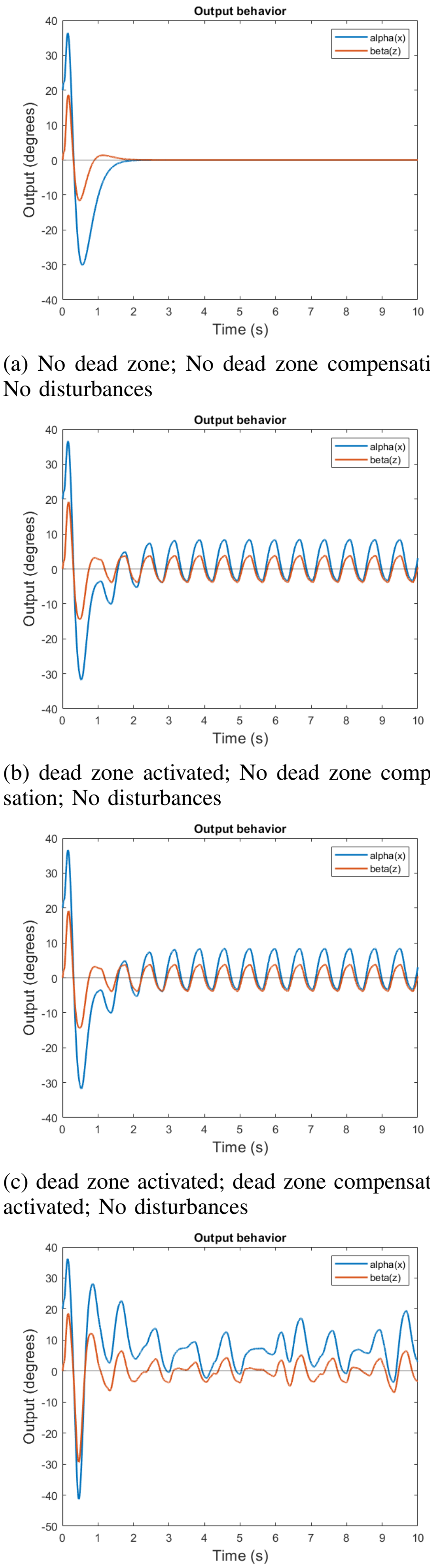
Overall, this project successfully met the control objectives: an RMS error below 5 degrees in steady state and a settling time under 3 seconds. To achieve this, an LQR was implemented using the provided linearized system, along with a dead zone hysteresis controller applied to its output. Since not all states are accessible in the real system, an observer was used to estimate them. In summary, **the LQR utilized weights** $Q_r = [2000 \quad 0 \quad 50 \quad 0 \quad 0]^T$ **and** $R_r = 0.05$, while **the observer had state covariance parameters** $w_e = 10$ **and** $r_v = 1$.

Complementary to real-world testing, simulations were developed for both the LQR and LQG controllers. These simulations enabled efficient testing of new parameter values without requiring constant lab access, which was not always feasible. As a result, they played a fundamental role in fine-tuning the controller and determining the final values that satisfied the project requirements.

Despite meeting the defined parameters, the group observed that oscillations could still be further reduced, as seen in other groups' experiments. Additionally, a steady-state error in the horizontal angle remained unaddressed. These findings present opportunities for future work.

The persistence of oscillations may be attributed to several factors. One major factor is the motor's dead zone, which makes the system inherently oscillatory, while another is the mechanical assembly—specifically, loose screws that were not always properly tightened, negatively impacting performance.

These limitations could be mitigated through several modifications. Implementing an LQI controller could help further reduce oscillations, particularly by correcting the steady-state error in the horizontal angle. Additionally, an MPC controller



(a) No dead zone; No dead zone compensation; No disturbances

(b) dead zone activated; No dead zone compensation; No disturbances

(c) dead zone activated; dead zone compensation activated; No disturbances

(d) Periodic disturbances with amplitude of 0.5 V; dead zone and dead zone compensation activated

Fig. 15: Computer simulation results: responses for $Q_r = [2000 \quad 0 \quad 50 \quad 0 \quad 0]^T$ and $R_r = 0.05$

— either linear or nonlinear — could better handle system constraints and nonlinearities, potentially leading to improved performance, although it might be somewhat excessive. A third improvement could be the use of Bryson's Rule to faster and more accurately estimate almost-optimal Q and R parameters.

Ensuring consistent lab conditions that closely match those of the simulation would also be beneficial. Moreover, implementing a nonlinear system in simulation could further improve its accuracy in representing the real system. While these improvements are beyond the scope of this project, they offer valuable directions for future research. All our work is available in this GitHub repository.

References

[1] Course slides. Available in Fénix, on the course web page.

[2] João Miranda Lemos (2019). Controlo no Espaço de Estados, Chap. 5.

[3] Franklin, Powell and Emami-Naeini. Feedback Control of Dynamic Systems, Pearson/Addison-Wesley (several editions). Chap. 7.

[4] Samuel Balula (2016). Nonlinear control of an inverted pendulum. M. Sc. Thesis, IST, Universidade de Lisboa. Available in Fénix, on the course web page.