BALL AND BEAM SYSTEM WITH STATIC ERROR AND STATE OBSERVER CONTROL

Christian Ricardo Conchari Cabrera
Mechatronics Engineering Student
Universidad Católica Boliviana "San Pablo"
La Paz, Bolivia
chrisconchari@gmail.com

Hamed Emmerson Quenta Alvarez
Mechatronics Engineering Student
Universidad Católica Boliviana "San Pablo"
La Paz, Bolivia
hamedquenta@gmail.com

***Abstract*—The purpose of this report is to present the results achieved for the final project of the IMT-341 course. We will begin by presenting the modelling and the main characteristics of the plant dynamics corresponding to the Ball and Beam system. After that, the plant dynamics will be modified to make it stable under design parameters, such as maximum overshoot and maximum settling time. For this purpose, methods such as eigenvalue placement and ITAE were implemented. Following this, control techniques will be implemented to mitigate the static error. These are a pre-compensation gain to decrease the steady-state error and the extension of the plant to an additional state for static error control. In the next instance, a state observer will be designed for the plant. Also, a static control for the plant will be implemented with the state observer, in this part, two different methods will be implemented, the first one will be the addition of an additional state to control the static error and a pre-compensation gain, as it has been done so far, and also a model will be implemented where a state observer including the state feedback controller and the static error control will be implemented. Finally, the conclusions corresponding to the project and some important considerations are presented.**

***Index Terms*—Ball and Beam, Eigenvalue Placement, ITAE, State Observer, Static Error.**

I. Introduction

The ball and beam system consists of a long beam with a ball rolling back and forth on top of it that may be tilted by a servo or electric motor. In control theory, it's a common textbook example [1]. The ball and beam system is significant because it is a basic, open-loop unstable system. Even if the beam is limited to be almost horizontal, it will swing to one side or the other without active input, and the ball will roll off the end of the beam. A control system that measures the ball's location and adjusts the beam accordingly must be utilized to steady the ball.

Therefore, this plant has been chosen to implement all the skills acquired during the course Control II - IMT 341. Briefly mentioning that it will be tested to control the plant by applying state feedback techniques, defining the dynamics of the system by eigenvalue placement and similar techniques. The implementation of control techniques for static error, such as pre-compensation gain and integral control using extended state space, will also be proved. Finally, the addition to the plant control of a full order state observer will be tested along with the control techniques mentioned above.

II. Objectives

A. General Objective

- The general objective of this project can be summarized as the application of the knowledge acquired during the IMT - 341 course in a real Ball and Beam plant simulated employing the Mathworks® Simulink and MATLAB software.

B. Specific Objectives

- Verify the controllability and observability of the system.
- Shape the Plant dynamics using Eigenvalue Placement for achieving a Max Overshoot of 4% and a max settling time of 0.7 seconds.
- Shape the Plant dynamics using ITAE control design for a fourth-order plant.
- Design a state observer for the Eigenvalue Placement and ITAE closed-loop Plants.
- Implement a Static Error Control together with a state observer for the Eigenvalue Placement and ITAE closed-loop Plants.
- Implement a state observer which includes a Feedback state and static error controller for the Eigenvalue Placement and ITAE closed-loop Plants.
- Generate the simulations corresponding to each of the design cases proposed in the previous objectives using Simulink software.

III. Plant Model and parameters

Figure 1 shows a diagram of the assigned plant, highlighting certain variables of importance for the modeling of the system and the subsequent development of a control system.

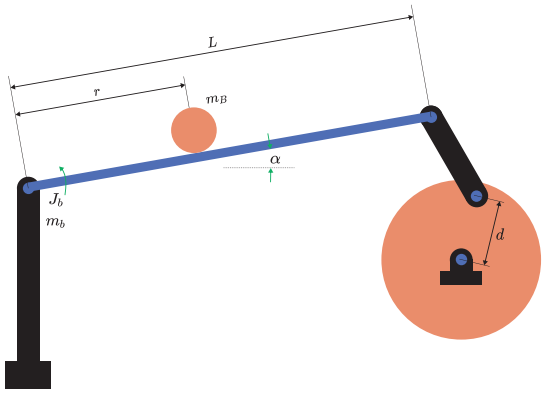


Figure 1: Ball and beam system

A. System Modeling

The states vector is given by:

$$x = \begin{bmatrix} \alpha \\ r \\ \dot{\alpha} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

Note that: $\dot{x_1} = x_3$ and $\dot{x_2} = x_4$
The model of the system is given by the following state-space representation [2]:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{m_b J_b g + m_B^2 g \delta^2}{(J_b + m_B \delta^2)^2} & -\frac{K_g^2 K_i K_m \eta_{total}}{R_m (J_b + m_B \delta^2)} \frac{L^2}{d^2} & 0 \\ -\frac{5g}{7} & 0 & 0 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ \frac{K_g K_i \eta_{total}}{R_m (J_b + m_B \delta^2)} \frac{L}{d} \\ 0 \end{bmatrix}$$

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

$$D = \begin{bmatrix} 0 \end{bmatrix}$$

B. Plant Parameters

The parameters of the given system are shown in table I:

Symbol	Quantity	Value
g	Gravity acceleration	$9.8[m/s^2]$
m_B	Ball mass	$0.064[kg]$
m_b	Beam mass	$0.65[kg]$
R	Ball radius	$0.0254[m]$
L	Beam length	$0.425[m]$
d	Lever length	$0.12[m]$
δ	Equilibrium point	$0.2[m]$
K_m	Back EMF constant	$0.00767[V \cdot s/rad]$
K_i	Torque constant	$0.00767[N \cdot m]$
K_g	Gear ratio	14
R_m	Motor resistance	$2.6[\Omega]$
η_{motor}	Motor efficiency	0.69
$\eta_{gearbox}$	Gearbox efficiency	0.85
η_{total}	Total efficiency	1.54
J_b	Beam moment of inertia	$0.0587[kg \cdot m^2]$

Table I: System parameters

C. Natural Plant Behavior

With the previously defined parameters, the definition of the variables was carried out in the Workspace of the Mathworks® MATLAB software:

```
g = 9.8;              %   Gravity acceleration
m_B = 0.064;          %   Ball mass
m_b = 0.65;           %   Beam mass
R = 0.0254;           %   Ball radius
L = 0.425;            %   Beam Length
d = 0.12;             %   Lever length
delta_2 = 0.2;        %   Equilibrium Point
K_m = 0.00767;        %   Back EMF constant
K_i = 0.00767;        %   Torque Constant
K_g = 14;             %   Gear ratio
R_m = 2.6;            %   Motor resistance
n_motor = 0.69;       %   Motor efficiency
n_gearbox = 0.85;     %   Gearbox efficiency

n_total = n_motor + n_gearbox;  % Total efficiency
J_b = (1/2)*m_b*L^2;    % Beam moment of inertia
```

With the defined parameters, the definition of the state space of the plant was carried out, which was shown in the System Modeling section:

```
a_32 = ((m_B*J_b*g) + m_B^2*g*delta_2^2)/(J_b+...
        m_B*delta_2^2)^2;
a_33 = ((K_g^2*K_i*K_m*n_total)/(R_m*(J_b+...
        m_B*delta_2^2)))*(L^2/d^2);
a_41 = (5*g/7);

% Define the A matrix for SS
A = [0,     0,      1,     0 ;
     0,     0,      0,     1 ;
     0,     -a_32, -a_33, 0;
     -a_41, 0,      0,     0];

b_31 = ((K_g*K_i*n_total)/(R_m*(J_b+m_B*delta_2^2...
        )))*(L/d);

% Define the B matrix for SS
B = [0;
     0;
     b_31;
     0];

% Define the C matrix for SS
C = [0 1 0 0];

% Define the D matrix for SS
D = 0;

%   Define the system order
n = size(A,1);

% Show the plant space state
PWC = ss(A,B,C,D);
PWC
```

Subsequently, the step response was obtained for the plant without control with the following code:

```
step(PWC,20), title('Natural plant behavior')
```

Obtaining the output:

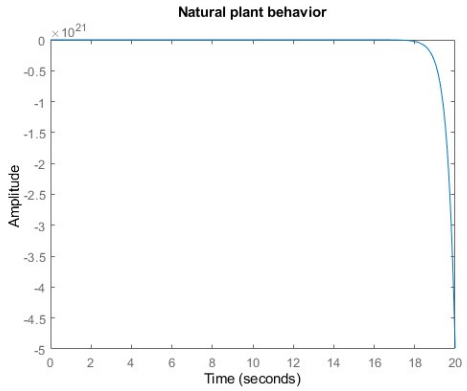


Figure 2: Natural Plant Behavior

As can be seen, initially it is an unstable system, that for an input Step has a response that tends to an indeterminate value.

Proceeding in this workflow, the next thing to do is to determine if our plant is controllable and observable. Below is the code with the Matlab functions used to perform these operations.

D. Controllability

In order to determine the controllability of the plant, the following code was executed:

```
controllable = rank(ctrb(A,B));
if controllable == n
    resp_controllability = 'Controllable';
else
    resp_controllability = 'Uncontrollable';
end
disp(['The system is: ',resp_controllability]);
```

The following output was obtained:

```
The system is: Controllable
```

This is because the controllability matrix obtained has a rank equal to 4, which matches the order of the system.

E. Observability

In a similar way, the following code was executed to determine the observability of the plant.

```
observable = rank(obsv(A,C));
if observable == n
    resp_observability = 'Observable';
else
    resp_observability = 'Unobservable';
end
disp(['The system is: ',resp_observability]);
```

The following output was obtained:

```
The system is: Observable
```

This is because the observability matrix obtained has a rank equal to 4, which once more matches the order of the system.

Now that we know that our plant is controllable we can proceed to implement control methods to modify the dynamics of the system as needed.

IV. Controller design considering Static Error

In this report we will present the design of state feedback controllers by the eigenvalue placement and ITAE method; although additionally in the attached Matlab documents we also tried to implement an LQR controller.

A. Eigenvalue Placement

The plant must be modified according to the following parameters:

$$OS = 4\%$$

$$t_s = 0.7$$

For this, the eigenvalue positioning technique will be used. Now, with these data, it is possible to obtain the **damping ratio** and **undamped natural frequency** by means of the following equations:

$$\xi' = \frac{\left|\ln\left(\frac{OS}{100}\right)\right|}{\sqrt{\pi^2 + \left[\ln\left(\frac{OS}{100}\right)\right]^2}}$$

$$\omega_n = \frac{4}{\xi' t_s}$$

In addition, the standard second-order transfer function must be considered:

$$G(s) = \frac{\omega_n^2}{s^2 + 2\xi\omega_n + \omega_n}$$

With the following eigenvalues:

$$\lambda_{1,2} = -\xi\omega_n \pm \omega_n\sqrt{\xi^2 - 1}$$

The code for the implementation is as follows:

```
% Desired plant dynamics parameters
OS=4;
Ts=0.7;

% Define the dominant poles
E = abs(log(OS/100))/sqrt(pi^2+(log(OS/100))^2);
Wn = 4/(E*Ts);

polescl1_PUC1 = -E*Wn+Wn*sqrt(E^2-1);
polescl2_PUC1 = -E*Wn-Wn*sqrt(E^2-1);
polescl3_PUC1 = real(polescl1_PUC1)*10;
polescl4_PUC1 = polescl3_PUC1-1;
Polescl_PUC1 = [polescl1_PUC1 polescl2_PUC1 ...
                polescl3_PUC1 polescl4_PUC1];

% Place the poles to shape the system dynamic
K1 = place(A,B,Polescl_PUC1);
```

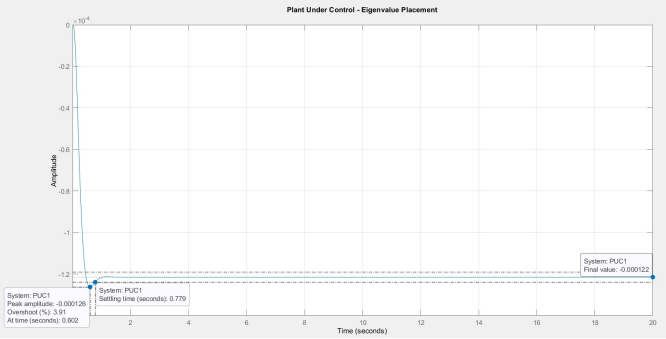


Figure 3: Plant Under Control - Eigenvalue Placement Step Response

Let's see that the figure 3 shows that we have achieved an **Overshoot** of 3.91% and a **Settling Time** of 0.78 seconds, parameters that are within the range of acceptable values for the desired design. On the other hand, let us note that our steady-state output deviates moderately from the expected output for a step-type input, this means a steady-state error, which can be identified as a static error that since the plant is sufficiently known can be corrected without problems employing a pre-compensation gain.

As mentioned above, using a feedback status control does not guarantee a static null error, so it is necessary to implement a gain in order to reduce this error. However, this method may not work in all cases and could increase the static error, this is why it is necessary to increase the order of the system by adding an additional integral part.

```
% Determine the precompensation gain
Kpre1 = inv(-(C-D*K1)*inv(A-B*K1)*B + D);

% Define the extended Space State
Aext = [A zeros(n,1) ; -C 0];
Bext = [B ; -D];
Cext = [C 0];
Dext = 0;
```

```
% Poles placement for extended SS
polescl5_PUC1 = polescl4_PUC1-1;
Polescl_PUC1_ext = [polescl1_PUC1 polescl2_PUC1 ...
        polescl3_PUC1 polescl4_PUC1 polescl5_PUC1];
Kext1 = place(Aext,Bext,Polescl_PUC1_ext);
K1ext = Kext1(1:n);
Kprelext = Kext1(n+1);
```

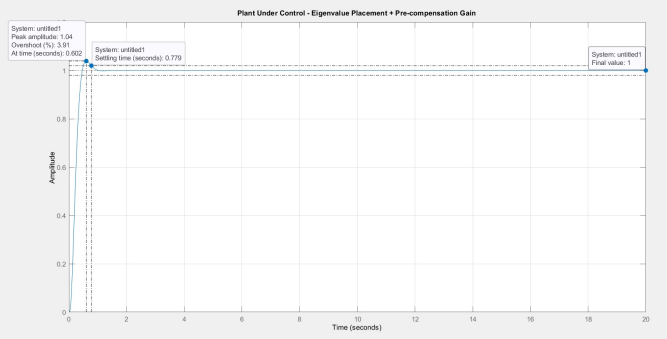


Figure 4: Plant Under Control - Eigenvalue Placement + Pre-compensation Gain Step Response

As can be seen in image 4 by using the pre-compensation gain we were able to stabilize the plant at unity, which would be expected for a step input. And the desired performance of the plant has been maintained.

B. ITAE

When applying this method, an attempt is made to shape by penalizing the error, in the case of the present plant it is the deviation between the step response and steady state value.

$$\int_0^\infty t\,|e(t)|\,dt$$

By minimizing the ITAE function produces a step response with small oscillation and overshoot.
For a Fourth-order system the following characteristic polynomial is given:

$$G(s) = s^4 + 2.1\omega_n s^3 + 3.4\omega_n^2 s^2 + 2.7\omega_n^3 s + \omega_n^4$$

The code implementation is as follows.

```
Polescl_PUC2 = (roots([1 2.1*Wn 3.4*Wn^2 ...
                2.7*Wn^3 Wn^4]))';
K2 = place(A,B,Polescl_PUC2);
```

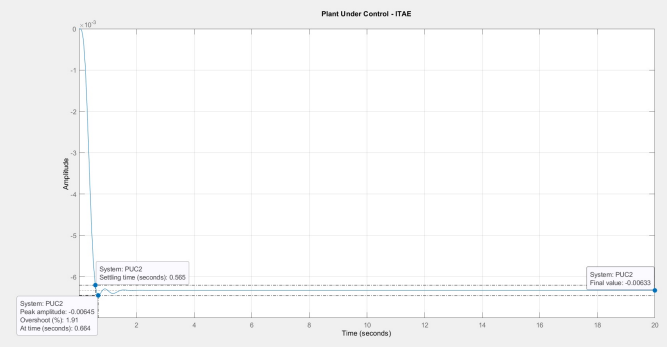


Figure 5: Plant Under Control - ITAE Step Response

As can be seen in the figure 5 we have achieved an **Overshoot** of 1.91% and a **Settling Time** of 0.56 seconds, parameters that are below the limit of the design conditions and represent quite acceptable dynamics. On the other hand, let's see that in the same way, we obtain a steady-state value far from the expected response for an input of the Step type.

The Static Error control design will remain the same as the one calculated for the prior art implementation, what will change will be the placement of the poles.

```
% Determine the precompensation gain
Kpre2 = inv(-(C-D*K2)*inv(A-B*K2)*B + D);
% Poles placement
polescl5_PUC2 = real(Polescl_PUC2(4))-1;
Polescl_PUC2_ext = [Polescl_PUC2(1), ...
    Polescl_PUC2(2), Polescl_PUC2(3), ...
    Polescl_PUC2(4), polescl5_PUC2];
Kext2 = place(Aext,Bext,Polescl_PUC2_ext);
K2ext = Kext2(1:n);
Kpre2ext = Kext2(n+1);

Acl2 = A-B*K2;
PUC2 = ss(Acl2,B,C,D);
PUC2
step(Kpre2*PUC2,20), ...
title('Plant Under Control - ITAE + Static Error')
stepinfo(Kpre2*PUC2)
```

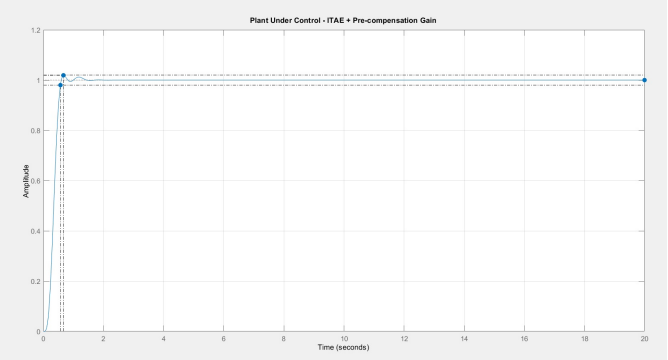


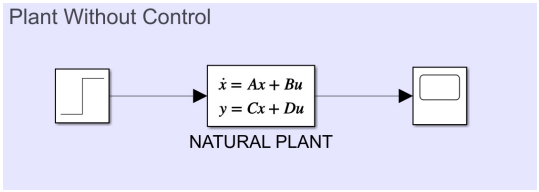
Figure 6: Plant Under Control - ITAE + Pre-compensation Gain Step Response

As we can see in image X the steady-state value has been modified so that our plant is stabilized at unity.

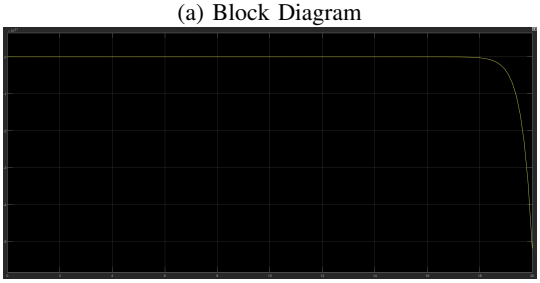
V. Simulation Results Part I

A. Natural Plant Behavior

In the figure 7 we can see the simulation of the plant under a step response with its natural behavior.



(a) Block Diagram



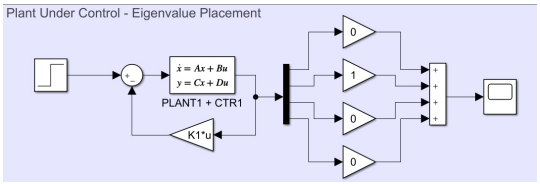
(b) Step Response Scope

Figure 7: Natural Plant Behavior

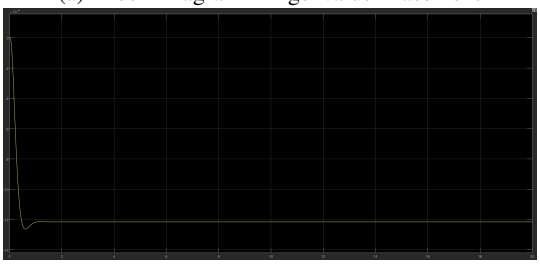
As we can see the response of the system is unstable and tends to an indeterminate value, as shown in the previous section.

B. Plant Under Control + Static Error - Eigenvalue Placement

In the figure 8 we can see the simulation of the plant under control, with the dynamics determined by the technique Eigenvalues placement.



(a) Block Diagram - Eigenvalue Placement

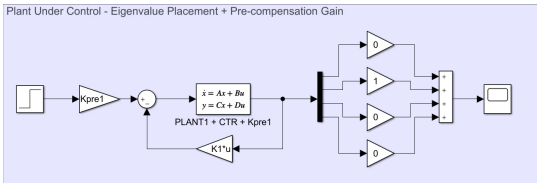


(b) Step Response Scope - Eigenvalue Placement

Figure 8: Plant Under Control - Eigenvalue Placement

Briefly, we can mention that the plant is adequately stabilized, just as it was in the design stage.

In the figure 9 we can see the simulation of the plant under control considering the pre-compensation gain to attenuate the static error.



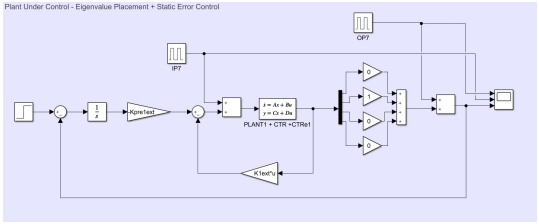
(a) Block Diagram - Eigenvalue Placement + Pre-compensation gain

(b) Step Response Scope - Eigenvalue Placement + Pre-compensation

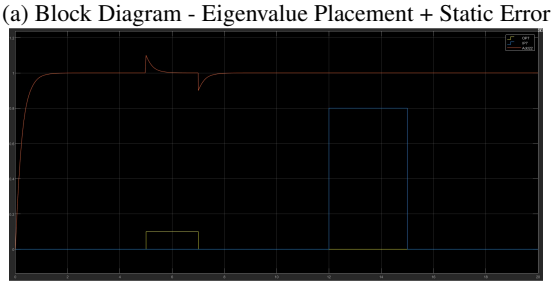
Figure 9: Plant Under Control - Eigenvalue Placement + Pre-compensation

As can be seen the pre-compensation gain reduces the steady state error. Thus, it could be seen that the static error has been compensated.

In the figure 10 we can see the simulation of the plant under control considering the pre-compensation gain and the integral control adding an extra state to our plant, to attenuate the static error and protect it from input-output disturbances and parametric uncertainty.



(a) Block Diagram - Eigenvalue Placement + Static Error



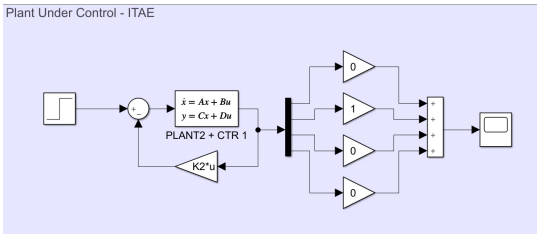
(b) Step Response Scope - Eigenvalue Placement + Static Error

Figure 10: Plant Under Control - Eigenvalue Placement + Static Error

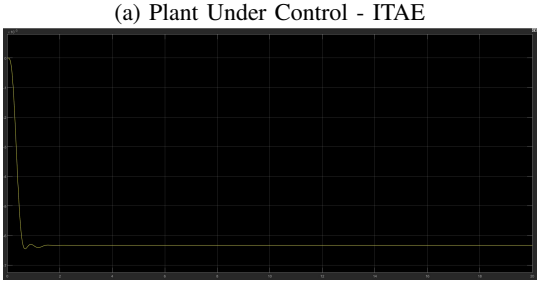
As can be seen in the figure 10b under input and output disturbances, and even under parametric uncertainty, which can be reviewed in the accompanying Simulink, the desired plant dynamics remains the same.

C. Plant Under Control - ITAE

In the figure 11 we can see the simulation of the plant under control, whose dynamics have been defined by the ITAE method for a fourth-order characteristic equation.



(a) Plant Under Control - ITAE

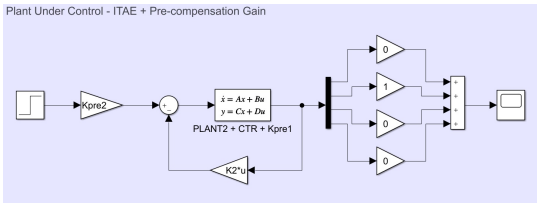


(b) Step Response Scope - ITAE

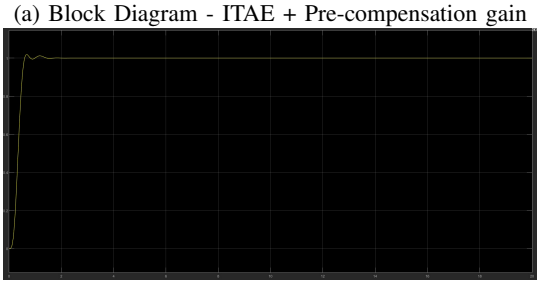
Figure 11: Plant Under Control - ITAE

The behavior of the plant is the same as presented in the previous section, during its design. It can also be seen in 11b the existence of a steady-state error.

In the following figure 12 we can see the simulation of the plant under control considering the pre-compensation gain to attenuate the static error.



(a) Block Diagram - ITAE + Pre-compensation gain

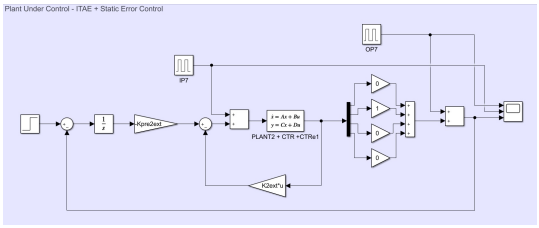


(b) Step Response Scope - ITAE + Pre-compensation gain

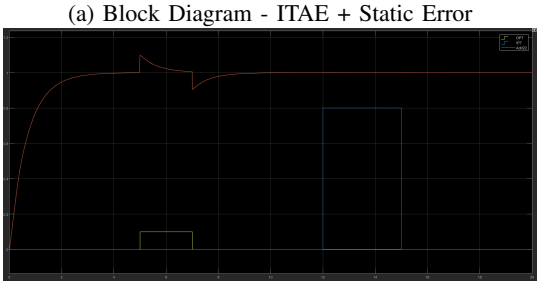
Figure 12: Plant Under Control - ITAE + Pre-compensation gain

In the image 12b can be seen as how the pre-compensation gain reduces the steady state error. Thus, it could be seen that the static error has been compensated.

In the figure 13 we can see the simulation of the plant under control considering the static error to protect the behavior of our plant from input-output disturbances and parametric uncertainty.



(a) Block Diagram - ITAE + Static Error



(b) Step Response Scope - ITAE + Static Error

Figure 13: Plant Under Control - ITAE + Static Error

As we can see in the image 13b under input-output disturbances and parametric uncertainty the plant remains stable.

VI. State Observer Design

The observer design is carried out for the plant, this is implemented in both control techniques, considering the design of poles that are 10 times distant from the plant poles.

A. Eigenvalue Placement

The observer gains are obtained with the poles calculated in the previous step and with the A and C matrices. Then, with the obtained gains, the observer state space is obtained that will be implemented later in the Simulink simulations of the plant.

```
% Define the observer poles 10 times the placed poles
PolesObs1 = Polescl_PUC1(1:n)*10;
% Find the L-values
L1 = place(A',C', PolesObs1)';
% Define the Observer Space State
Aobs1 = A - L1*C;
Bobs1 = [B-L1*D L1];
Cobs1 = eye(n);
Dobs1 = zeros(n,2);
```

1) Observer + States Controller Design: .

```
Aoc1 = A-L1*C-B*K1+L1*D*K1;
Boc1 = [B-L1*D, L1];
Coc1 = -K1;
Doc1 = [1, 0];
```

2) Observer + States Controller Design + Static Error: .

```
Aocc1 = [A-L1*C-B*K1ext+L1*D*K1ext ...
        -B*Kpre1ext+L1*D*Kpre1ext; zeros(1,n) 0];
Bocc1 = [zeros(n,1), L1; 1, -1];
Cocc1 = [-K1ext -Kpre1ext];
Docc1 = zeros(1,2);
```

B. ITAE

In the same way as in the Eigenvalues Placement method, the observer gains and the observer state space are obtained with the following code:

```
% Define the observer poles 10 times the placed poles
PolesObs2 = Polescl_PUC2(1:n)*10;
% Find the L-values
L2 = place(A',C', PolesObs2)';
% Define the Observer Space State
Aobs2 = A - L2*C;
Bobs2 = [B-L2*D L2];
Cobs2 = eye(n);
Dobs2 = zeros(n,2);
```

1) Observer + States Controller Design: .

```
Aoc2 = A-L2*C-B*K2+L2*D*K2;
Boc2 = [B-L2*D, L2];
Coc2 = -K2;
Doc2 = [1, 0];
```

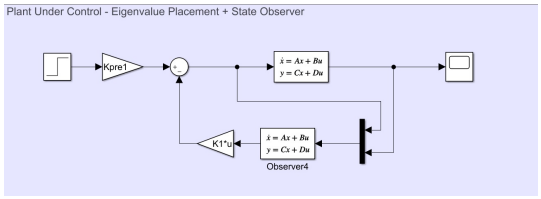
2) Observer + States Controller Design + Static Error: .

```
Aocc2 = [A-L2*C-B*K2ext+L2*D*K2ext ...
        -B*Kpre2ext+L2*D*Kpre2ext; zeros(1,n) 0];
Bocc2 = [zeros(n,1), L2; 1, -1];
Cocc2 = [-K2ext -Kpre2ext];
Docc2 = zeros(1,2);
```

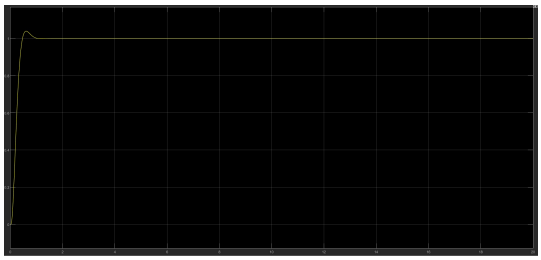
VII. Simulation Results Part II

A. Plant Simulation with State Observers & Static Error - Eigenvalue Placement

The image 14 shows the block diagram of the implementation of a state observer, together with a pre-compensation gain.



(a) Block Diagram - Eigenvalue Placement + State Observer

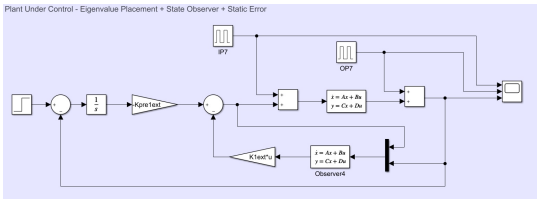


(b) Step Response Scope - Eigenvalue Placement + State Observer Scope

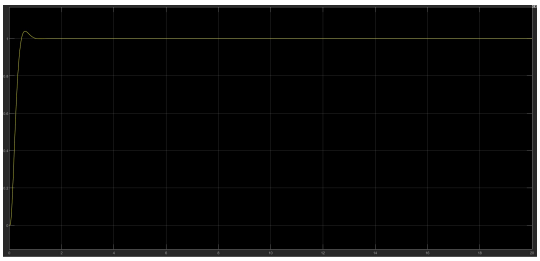
Figure 14: Plant Under Control - Eigenvalue Placement + State Observer

As can be seen in the image 14b, the behavior of the plant has been maintained even when implementing the state observers. Note that a pre-compensation gain has been implemented to reduce the static error.

In order to deal with input and output disturbances, as well as parametric uncertainty, it will be necessary to implement an additional state in addition to the pre-compensation gain, what can be seen in the figure 15.



(a) Block Diagram - Eigenvalue Placement + State Observer + Static Error

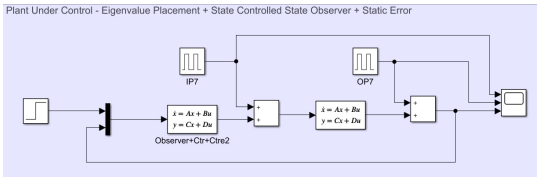


(b) Step Response Scope - Eigenvalue Placement + State Observer

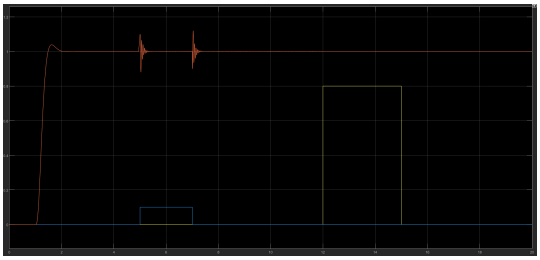
Figure 15: Plant Under Control - Eigenvalue Placement + State Observer + Static Error

As can be seen in the figure 15b parametric uncertainty and disturbances both at the input and output have been properly treated.

In the image 16 another simulation is presented for the implementation of a state observer that also applies closed-loop and static error control.



(a) Block Diagram - Eigenvalue Placement + Static Error & Feedback State Observer Controller



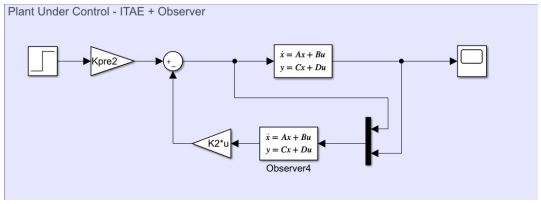
(b) Step Response Scope - Eigenvalue Placement + Static Error & Feedback State Observer Controller

Figure 16: Plant Under Control - Eigenvalue Placement + Static Error & Feedback State Observer Controller

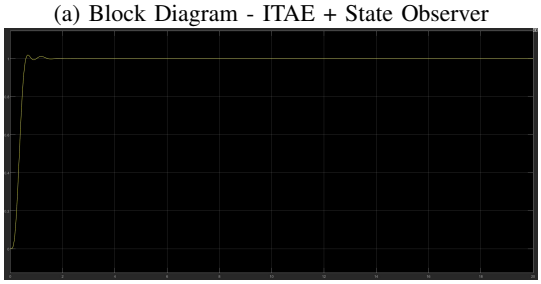
As can be seen in the image 16b, the same results as in the image 16 have been obtained through both implementations. In both cases we were able to maintain the system dynamics protecting our control from static error, internal-external disturbances and parametric uncertainty.

B. Plant Simulation with State Observers & Static Error - ITAE

The image 17 shows the block diagram of the implementation of a state observer, together with a pre-compensation gain for the ITAE design.



(a) Block Diagram - ITAE + State Observer

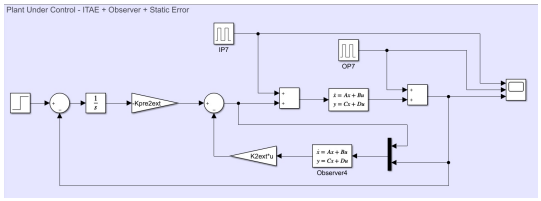


(b) Step Response Scope - ITAE + State Observer Scope

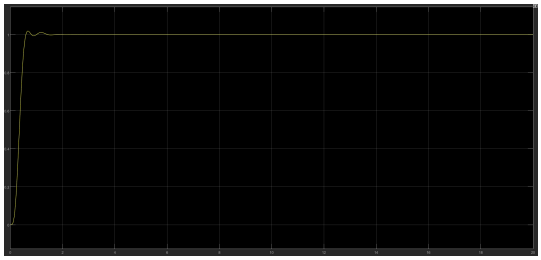
Figure 17: Plant Under Control - ITAE + State Observer

Looking the figure 17b, the plant behavior has been maintained even with the state observers. Note that as with the previous design a pre-compensation gain has been implemented to reduce the static error.

To deal with input-output disturbances, as well as parametric uncertainty, as it has been done for the previous design it will be necessary to implement an additional state in addition to the pre-compensation gain, as in the following figure 18.



(a) Block Diagram - ITAE + State Observer + Static Error

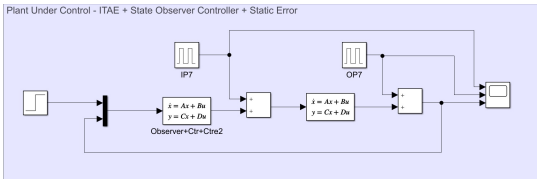


(b) Step Response Scope - ITAE + State Observer

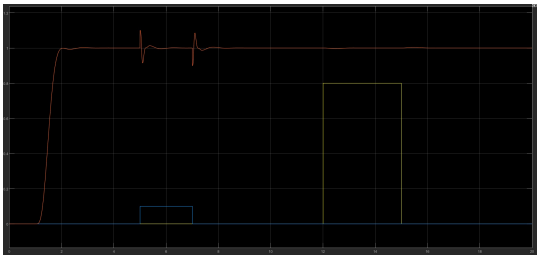
Figure 18: Plant Under Control - ITAE + State Observer + Static Error

In the figure 18b, can be seen that the parametric uncertainty and input-output disturbances both have been properly treated.

In the image 19, as previously done, another simulation is presented for the implementation of a state observer that also applies closed-loop and static error control.



(a) Block Diagram - Eigenvalue Placement + Static Error & Feedback State Observer Controller



(b) Step Response Scope - Eigenvalue Placement + Static Error & Feedback State Observer Controller

Figure 19: Plant Under Control - Eigenvalue Placement + Static Error & Feedback State Observer Controller

As shown with the previous design, in the image 19b, the same results were obtained in the image 19 through both implementations.

VIII. Conclusions

The present project has been carried out to demonstrate the knowledge acquired during the course, in the previous sections it has been shown how all the techniques proposed have been implemented with satisfactory results, being these feedback control designed by eigenvalue placement and ITAE, together with the corresponding static error control, where it has been shown in results as even with input-output disturbances and parametric uncertainty it has been possible to maintain a stable plant. Also have been adequately implemented, plants with state observers, together with static error control. In addition, a plant model has been presented with a state observer capable of state feedback control and static error control.

It is necessary to mention that although this study has been carried out with a wide variety of control methods, this is because the facility of working with a simulation in Simulink allows us to do so. But certainly, if it is about implementing this project in a real physical Ball and Beam plant, we would have to choose to implement a single method that gives us good results. Being that in a real environment a large number of things can fail, the main considerations that we should consider are the energy that the actuator will need and to which states we must give them greater importance.

According to the results obtained with the present work, as a group, we would lean towards the implementation of

the controller design by eigenvalue placement, since this was designed considering a settling time that is quite possible to achieve under a fairly acceptable overshoot. Although under personal appreciations, if we want to work by methods a little more experimental, the implementation of an LQR controller would be a quite viable option, since through this method we can easily manipulate how we penalize the energy that will be imposed on the actuator and penalize the importance that is It will be assigned to each of the states as necessary to achieve a specific system dynamics. Taking this into account in the attached Matlab documents, the design of an LQR controller for the plant of this project was additionally carried out, it should be mentioned that for this implementation it was only necessary to calculate a pre-compensation gain for that way. To reduce the steady-state error, if you want to protect the plant from external disturbances, you could consider implementing an LQG controller, although this could result in making the project more complex to an unnecessary point.

With all the aforementioned, we can conclude the present work by saying that the objectives set have been met, is that the control techniques defined at the beginning have been implemented with satisfactory results

REFERENCES

[1] Wikipedia contributors. (2020, April 2). Ball and beam. Wikipedia. Available: https://en.wikipedia.org/wiki/Ball_and_beam. [Accessed: 17-Jun- 2021]

[2] N. N. A. Aziz, M. I. Yusoff, M. I. Solihin, and R. Akmeliawati, “Two degrees of freedom control of a ball and beam system, ”IOP Conference Series: Materials Science and Engineering, vol. 53, p. 012070, dec 2013. [Online]. Available: https://doi.org/10.1088/1757-899x/53/1/012070