Control Engineering

Professor: Enrique Javier Aguayo Lara

Final Project

Group members:

Adriana Ibarra Sánchez

A01229596

Perla Vanessa Jaime Gaytán

A00344428

INDEX

Introduction

The objective of this project was to simulate a production line, which included one system (Automated storage and retrieval system) and four stations (Filling, Heating, Cooling, and Packaging). Each station/system, had certain restrictions that had to be applied by using the concepts learnt through the semester, such as transfer functions, controllers, stability, and observability among others.

Theoretical framework

Second Order System.

A second order system has a particular response over time. There are three different times: delay time (t_d), time of growth (t_r) and peak time (t_p). The delay time is the time that takes the response to reach half of the final value for the first time, the time of growth is the time that required for the response to grow from 10% to 90% and the peak time is the time required for the response to reach the first peak of the overshoot. This different times can also be defined by the Figure 1, eq. 1, eq. 2 and eq. 3.

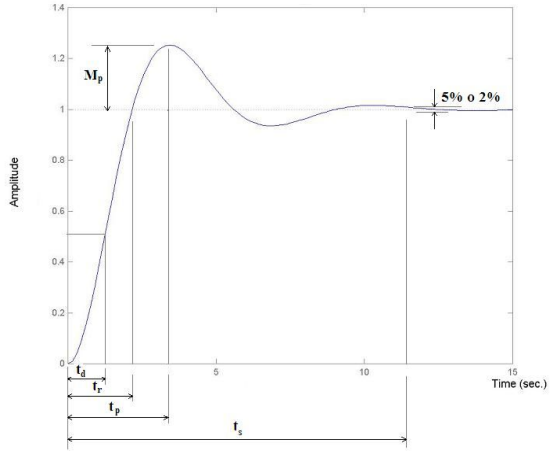

Figure 1. Response time of a second order system.

$$t_r = \frac{1}{\omega_d} tan^{-1}(-\frac{\sqrt{1-\delta^2}}{\delta}) \qquad \text{eq. 1}$$

$$t_p = \frac{\pi}{\omega_d} \qquad \text{eq. 2}$$

$$\omega_d = \omega_n \sqrt{1-\delta^2} \qquad \text{eq. 3}$$

Feedback Control Systems

A system that maintains a prescribed relationship between the output and the reference input by comparing them and using the difference as a means of control is called a feedback control system.

- Closed-Loop: In a closed-loop control system the actuating error signal, which is the difference between the input signal and the feedback signal is fed to the controller so as to reduce the error and bring the output of the system to a desired value.
- Open-Loop: In an open-loop control system the output is neither measured nor fed back for comparison with the input.

An advantage of the closed loop control system is the fact that the use of feedback makes the system response relatively insensitive to external disturbances and internal variations in system parameters. From the point of view of stability, the open-loop control system is easier to build because system stability is not a major problem. On the other hand, stability is a major problem in the closed-loop control system, which may tend to overcorrect errors and thereby can cause oscillations of constant or changing amplitude.

Performance and characterization of dynamic systems

The transfer function of a linear, time-invariant, differential equation system is defined as the ratio of the Laplace transform of the output (response function) to the Laplace transform of the input (driving function) under the assumption that all initial conditions are zero. For a linear, time-invariant system the transfer function G(s) is: $G(s) = \frac{Y(s)}{X(s)}$. Where X(s) is the Laplace transform of the input and Y (s) is the Laplace transform of the output, where we assume that all initial conditions involved are zero.

Controller: An automatic controller compares the actual value of the plant output with the reference input (desired value), determines the deviation, and produces a control signal that will reduce the deviation to zero or to a small value.

Proportional-plus-integral-plus-derivative controllers: The combination of proportional control action, integral control action, and derivative control action is termed proportional-plus-integral-plus-derivative control action.This combined action has the advantages of each of the three individual control actions. The equation of a controller with this combined action is given by the transfer function:

$\frac{U(s)}{E(s)} = K_p(1 + \frac{1}{T_1 s} + T_d s) \rightarrow$ where K, is the proportional gain, T, is the integral time, and T, is the derivative time.

Kp tries to minimize the system error. When the error is large, the control action is large and tends to minimize this error. Increasing the proportional action Kp has the following effects:

- Increases the response speed of the system.

- Decrease the system error in permanent regime.
- Increases system instability.

The derivative of the error is another way of calling the "speed" of the error. Increasing the derivative control constant Kd has the following effects:

- Increases the stability of the controlled system.
- Slow down the system a bit.
- The error in permanent regime will remain the same.

The integral can be seen as the sum or accumulation of the error signal. Increasing the integral action Ki has the following effects:

- Decrease the system error in permanent regime.
- Increases system instability.
- It increases the speed of the system a little

Analysis of Control Systems in State Space: Controllability and Observability

When a system is controllable at time t_0 it means that it is possible to transfer the system from initial state $x(t_0)$ to any other state in a finite interval of time by means of an unconstrained control vector. When a system is observable at time t_0 means that, with the system in $x(t_0)$, it is possible to determine this state by observation of the output over a finite time interval. This two concepts has an extremely importance for the design of a control system in state space. In matlab, we can check whether the system is observable or controllable by the following commands where A, B and C are the matrix of the system in state space.

rank(obs(A,C)) and rank(ctrb(A,B)) the result of this two commands should give the number of the order of the system, if the result is not equal to the order, the system is neither controllable nor observable.

Nyquist Stability Criterion

The Nyquist stability criterion determines whether the system is stable in closed-loop from its frequency response and poles in open-loop. For stability, all the roots of the characteristic equation, all the poles of the closed-loop transfer function, must lie in the left-half of the plane. The Nyquist stability criterion relates the open-loop frequency response to the number of zeros and poles that lie in the right-half of the plane. This criterion is useful in Control Engineering because because the absolute stability in close-loop can be determined graphically from open-loop frequency-response curves, and there is no need for actually determining the closed-loop poles. By analyzing the stability of the linear control system using the Nyquist stability criterion, there are three possibilities that can occur. The first one is when is no encirclement of the -1 + 0j point, this implies that the system can be stable only if there are no poles in the right-half plane. The second one is where there are one or more counter clockwise encirclements of the -1 +0j, in this case, the system is stable if the number of counterclockwise encirclements are the same as the number if poles. An the third one is where is one or more clockwise encirclements of the -1 +j0 point, in this case the system is unstable.

Calculations

DC Motors

AS/RS motors:

There are 3 types of DC motors to help move the bottle from one of the cells into the conveyor. The first, which moves horizontally, has a length of 200 cm. The second, which moves vertically, has length of 250 cm. The last motor helps to move the bottle from one place to another. After the other two motors reach the position to grab the bottle, the placement movement motor grabs the bottle and then, move again to put it into the conveyor. The general function of the motors after the bottles is picked from one of the cells, the car must go to the center point of the storage and move forward 25 cm to place the bottle into the conveyor. After the bottle is in the conveyor down 1 cm, wait 1 second, back 25 cm and the it can be set again to another position.

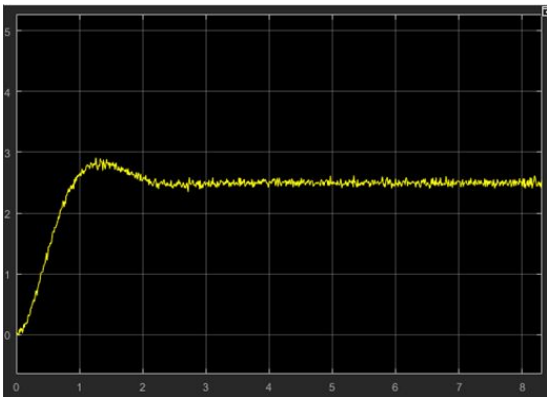

Figure 2. Angular velocity rev/s

To obtain the transfer function on the motor we have the graph on the figure X of the angular velocity of the motor in Open Loop. From this graph we obtain:

$t_p = 1.2\ s \qquad t_r = 0.8\ s$

$t_p = \frac{\pi}{\omega_d} \qquad \omega_d = \frac{\pi}{t_p} = \frac{\pi}{1.2}$

$\omega_d = 2.618$

$t_r = \frac{1}{\omega_d} tan^{-1}(-\frac{\sqrt{1-\delta^2}}{\delta}) \qquad 0.8 = \frac{1}{2.618} tan^{-1}(-\frac{\sqrt{1-\delta^2}}{\delta})$

$\delta = 0.5$

$\omega_d = \omega_n\sqrt{1-\delta^2} \qquad 2.618 = \omega_n\sqrt{1-(0.5)^2}$

$\omega_n = 3.023$

$$G(s) = \frac{k_s\omega_n^2}{s^2+2\delta\omega_n+\omega_n^2} = \frac{k_s(3.023)^2}{s^2+2(0.5)(3.023)s+(3.023)^2}$$

$$G(s) = \frac{9.139k_s}{s^2+3.023s+9.139}$$

We established that the input of the motor was 1 V and the final value will be 2.5 rps, therefore u(t) = 1 and $y_{ss} = 2.5$

$$y_{ss} = \lim_{s\to 0} u(t)G(s) \qquad 2.5 = \lim_{s\to 0} 1(\frac{9.139k_s}{s^2+3.023s+9.139}) = \frac{9.139k_s}{9.139}$$

$k_s = 2.5$

$$G(s) = \frac{9.139(2.5)}{s^2+3.023s+9.139}$$

$$G(s) = \frac{22.846}{s^2+3.023s+9.139}$$

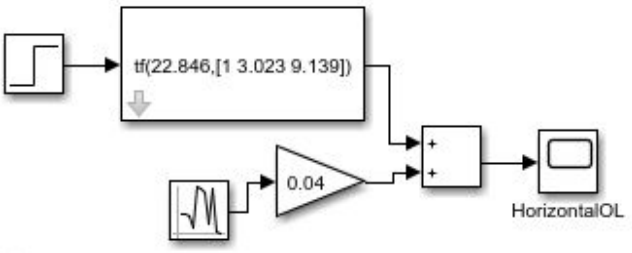

Figure 3. Open Loop System for a DC motor.

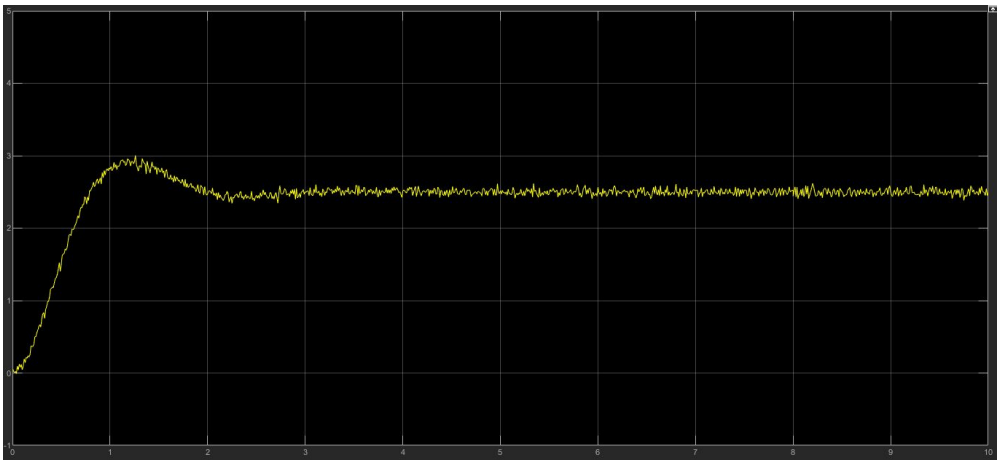

Figure 4. Step response on Open Loop for a DC motor

The three DC motors has the same transfer function. The only thing that changed is the gain in Closed Loop of every DC motor. The input of the system is the position in cm where the motor will arrived and the output will be this same position in cm. In order to convert the rev/s into cm/s we need to multiply by a gain and an integrator.

Integrador = $\frac{1}{s}$

Gain of the plant = k_2

Proportional gain = k_1

$$G(s) = \frac{1}{s}(k_2)\frac{22.846}{s^2+3.023s+9.139}$$

$y_{ss} = \lim_{s\to 0} k_1 u(t) s G(s) \qquad u(t) = \lim_{s\to 0} k_1 u(t) s\left(\frac{1}{s}(k_2)\frac{22.846}{s^2+3.023s+9.139}\right)$

$1 = k_1(k_2)\frac{22.846}{9.139} \qquad k_1(k_2) = \frac{9.139}{22.846}$

$k_1(k_2) = 0.4$

For the horizontal movement motor, 1 rev = 1 cm, therefore $k_2 = 1$ and $k_1 = 0.4$. The reference of this motor it goes first to the desire position, then move to the middle, in this case to 100 cm, after the bottle is picked and then wait until the other motors finished in order to move to another desire position.

For the vertical movement motor, 1 rev = 2 cm, therefore $k_2 = 2$ and $k_1 = 0.2$. The reference of this motor it goes first to the desire position, then move to 0 and wait that the bottle is picked and the horizontal motor reached its middle position. Then it goes to the middle, in this case to 125 cm. After the displacement motor reached 25 cm, this motor moves 1 cm down. After that, it waits until the bottle is on the conveyor in order to go to another desire position.

And for the displacement motor, 1 rev = 0.5 cm, therefore $k_2 = 0.5$ and $k_1 = 0.8$. The reference of this motor it goes first to 40 cm to reach the bottle, then move to 0 and wait until the vertical motor reached 125 cm. Then moves to 25 cm waits that the vertical motor go down 1 cm plus 1 second more and then move to 0 cm to wait another desire position to picked a bottle.

All the motors depends on each other in order to move. The horizontal motor (yellow) moves first from 0 to one desire position from 0 to 200 cm (the length of this motor), in this

case the desire position is 118, and waits there to the other motors to move. After the horizontal motor arrives, the vertical motor (blue) moves from 0 to a desire position from 0 to 250 cm (the length of this motor), in this case the desire position is 56 cm, and waits there. After the vertical motor reaches its desire position, the displacement motor (orange) moves from 0 to its maximum length, waits 2 second to pick up the bottle and then it goes back to 0. After the displacement motor reaches 0, the horizontal motor moves to its medium point, which is 100 cm, and waits there. After the horizontal motor reaches 100 cm, the vertical motor moves to 125 cm. When the horizontal motor reaches 125cm, the displacement motor moves to 25 cm, then the vertical moves -1 cm and then the vertical comes back to 0 cm. After the displacement motor reachers 0, the horizontal motor moves to 0 cm, and after that, the vertical motor also moves to 0 cm.

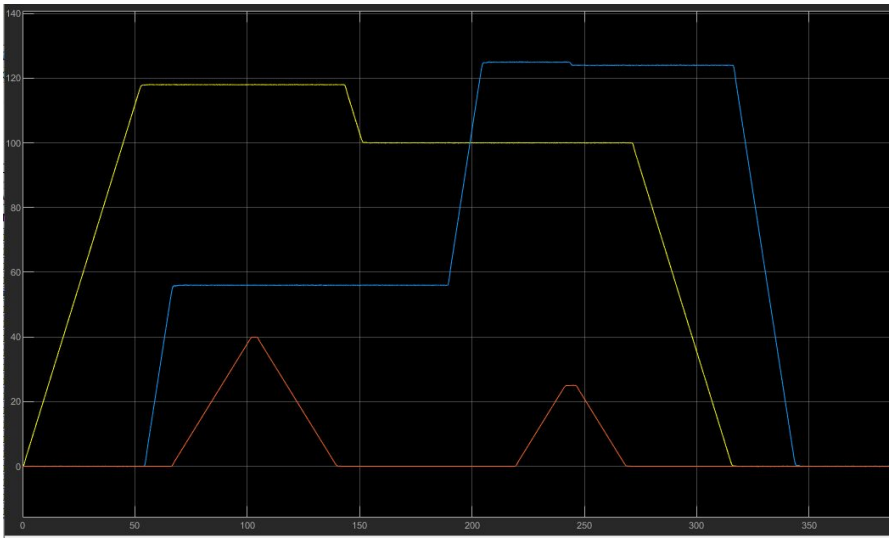

Figure 5. Output of the simulation of three AS/RS motors

The times that the simulation takes is different in every case, since its motor has its velocity an takes different time to reach the desired position. We use the following code on matlab to simulate all the possible cases.

```
a=randi(200);               %Desired horizontal position
b=randi(250);               %Desired vertical position

tHor=round(a/2.25 + 1);             %time that takes horizontal motor to reach a

tVer=tHor + round(b/(2*2.25) + 1);      %time,after tHor, that takes vertical motor to reach b

tPla=tVer + round(40/(0.5*2.25) + 1)+1;     %time, after tVer, that takes placement motor to reach 40 cm
tPla2= tPla + round(40/(0.5*2.25) + 1)+2;   %time after tPla that placement motor take to reach 0 cm again

tHor2= tPla2 + round(100/2.25 + 1)+1;       %time, after tPla2, that horizontal motor takes to reach 100 cm

tVer2= tHor2 +round(125/(2*2.25) + 1)+1;    %time, after tHor2, that vertical motor takes to reach 125 cm

tPla3 = tVer2 + round(25/(0.5*2.25) + 1)+1; %time, after tVer2, that placement motor takes to reach 25 cm

tVer3 = tPla3 + round(1/(2*2.25) + 1)+2;    %time, after tPla3, that vertical motor takes to reach 1 cm + 2s

tF= tVer3 +  round(25/(0.5*2.25) + 1)+2;    %time, after tVer3, that the placement motor take to reach 0 cm

tF2= tF + round(100/2.25 +1 );              %time after tF that horizontal motor takes to reach 0 cm
```

Figure 6. Code to simulate AS/RS motors.

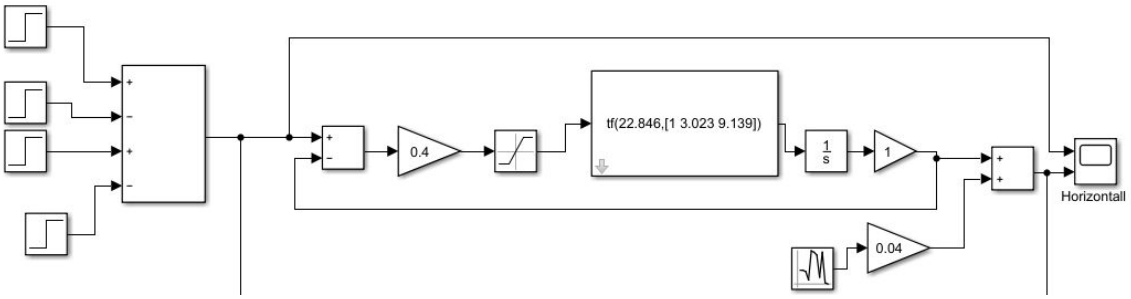

Figure 7. Horizontal motor simulink system.

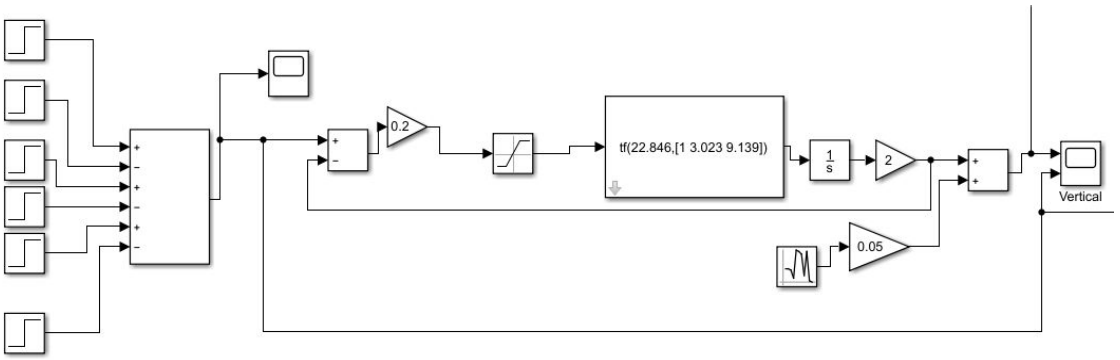

Figure 8. Vertical motor simulink system.

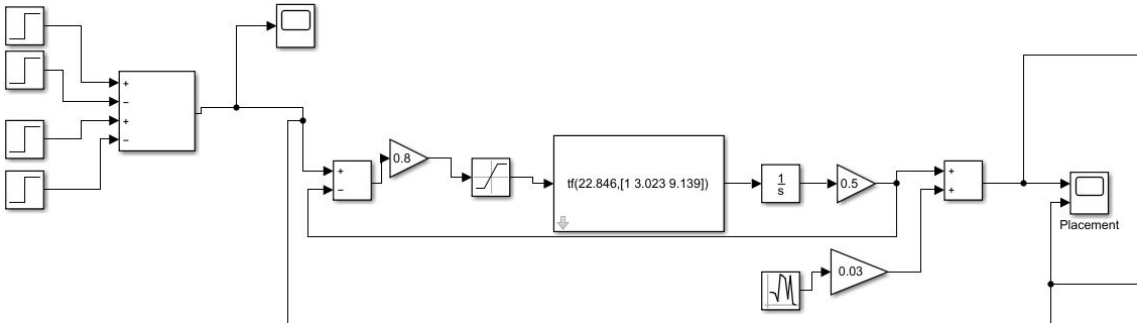

Figure 9. Placement motor simulink system.

Displacer to stations:

There are 2 types of displacers but the two works the same, the only difference is the angular velocity that has each motor. This displacers have a linear guide that can move up to 40 cm. It first moves to 30 cm toward the station, wait for 2 seconds and then move back to its initial position and waits until the station finishes. After each station finish, the motor must move again 30 cm toward the station, wait 2 sec and then move back to its initial position in order to set again.

The type 1, 1 revolution is equal to 1 cm, therefore $k_2 = 1$ and $k_1 = 0.4$. This displacer is used in two stations: Filling and Heating.This motor take 14 seconds approximately to arrive to 30 cm. The goal is maintaining the references 16 seconds in order to let the motor arrive the position and wait 2 seconds more on the station and then come back to its initial position. The Filling station takes no more than 2s, therefore we decided that it does not make sense to wait for 32 seconds (16 to get to 0 and 16 to come back to 30 cm) for a station that is done in less than 2 seconds. Therefore we decided that it only goes back 5 cm and then goes forward another 5 cm to get back to 30 cm again. The Heating Station does not have this problem, since last 60 seconds, so it works as described before.

The type 2, 1 revolution is equal to 1.2 cm, therefore $k_2 = 1.2$ and $k_1 = 0.333$.

This displacer is used in two stations: Cooling and Packaging.This motor takes approximately 10 seconds to arrive 30 cm, therefore the reference should wait 12 seconds. Since the Cooling station takes 40 seconds and the Packaging takes 60 seconds, this stations works as described before.

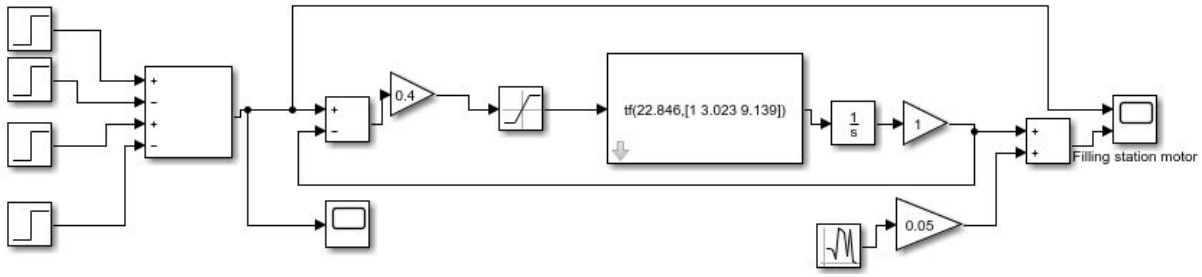

Figure 9. Displacement motor of the filling station simulink system.

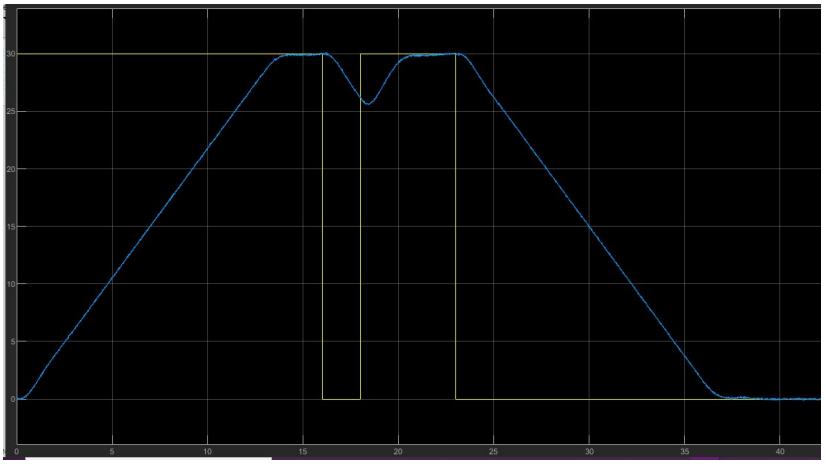

Figure 10. Output of the filling station dc displacement motor.

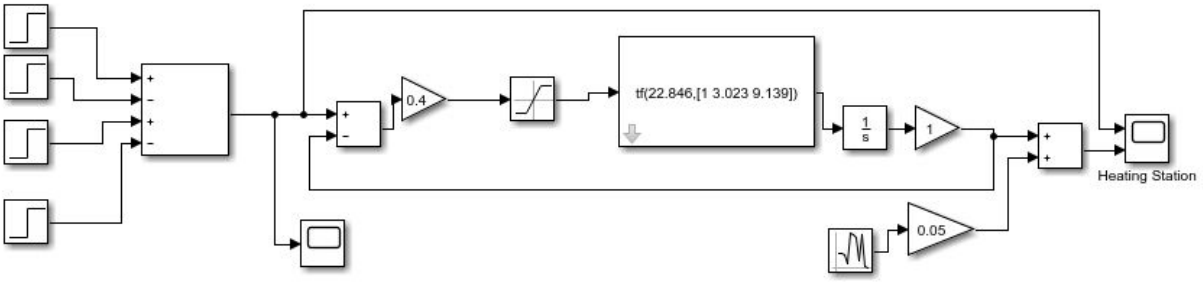

Figure 11. Displacement motor of the heating station simulink system.

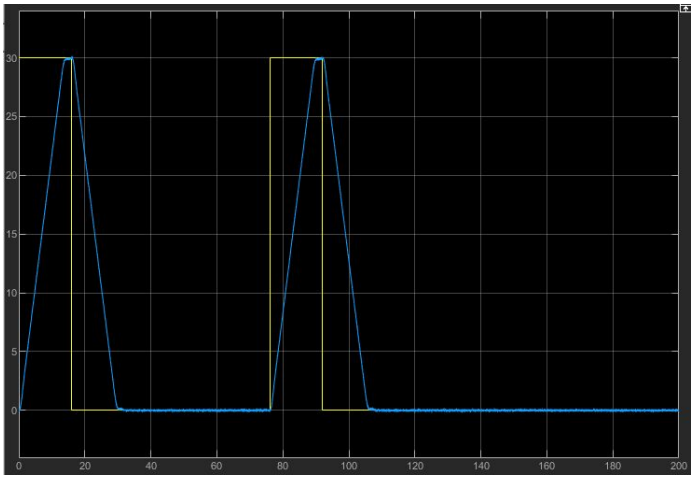

Figure 12. Output of the heating station dc displacement motor.

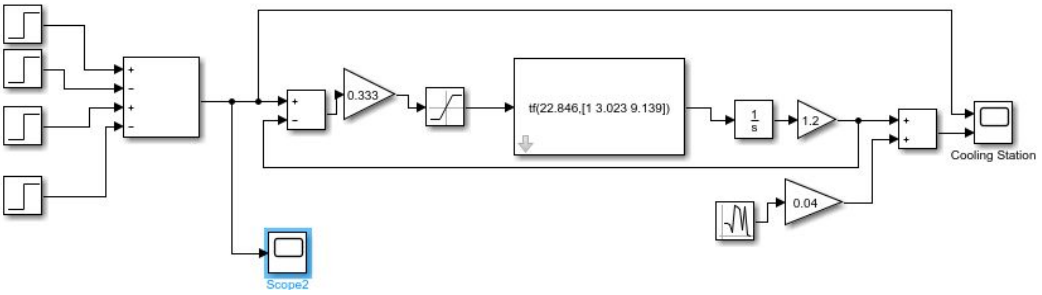

Figure 13. Displacement motor of the cooling station simulink system.

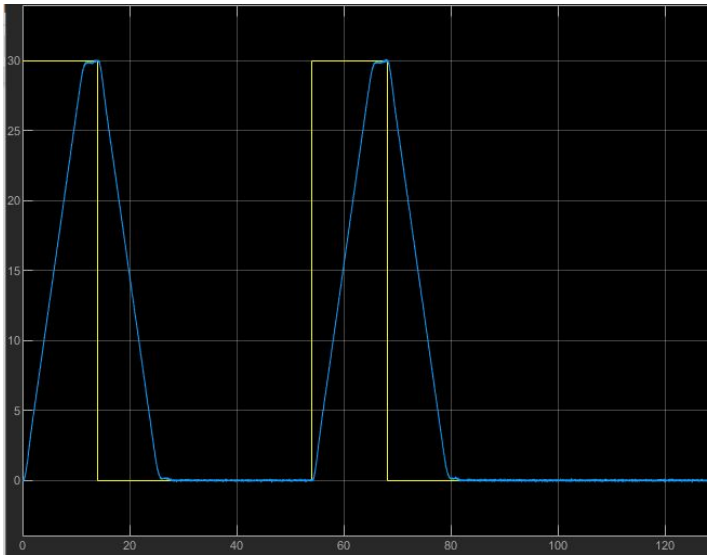

Figure 14. Output of the cooling station dc displacement motor.

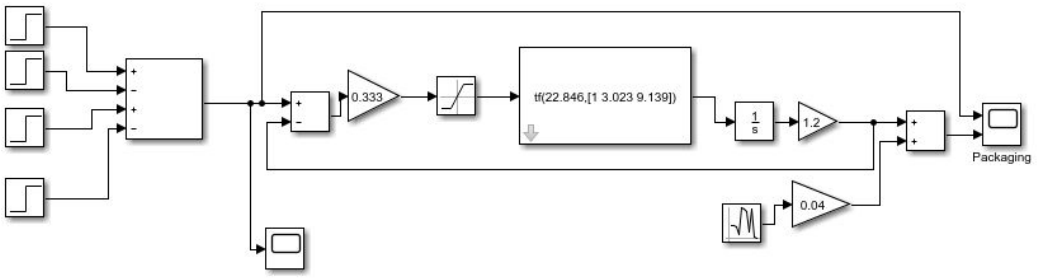

Figure 15. Displacement motor of the packaging station simulink system.

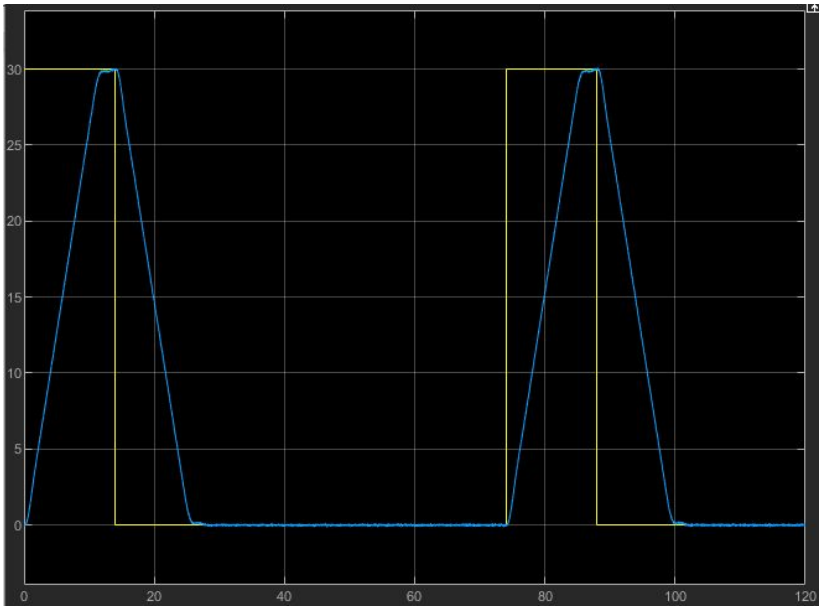

Figure 16. Output of the packaging station dc displacement motor.

```
d= round(30/2.25+1);                    %time to takes 1st disp motor to arrive 30cm
tDisFi=d+2;                             %time on stations
tDisFi2= tDisFi +2;                     %time after the 2 seconds of the station
tDisFi3 =tDisFi2 +round(5/2.25+1)+2;    %time after the 2 seconds on the station

tDisHeat= d + 2;                        %time after disp arrive plus the 2s on the station
tDisHeat2=tDisHeat + 60;                %time after waiting for the station
tDisHeat3= tDisHeat2 + d + 2;           %time after comes back to 30 cm and wait 2 seconds

e =  round(30/(1.2*2.25) + 1);          %time to takes 2nd disp motor to arrive 30 cm

tDisCol = e + 2;                        %time on station
tDisCol2 = tDisCol + 40;                %time after waiting the station
tDisCol3 = tDisCol2 + e + 2;            %time after goes to 30 cm + 2 seconds

tDisPack = e + 2;                       %time on station
tDisPack2=tDisPack + 60;                %time after waiting the station
tDisPack3= tDisPack2 + e + 2;           %time after goes to 30 cm + 2 seconds
```

Figure 17. Code to simulate displacement motors.

Filling Station

In the filling station, we are making sure that a bottle is filled in 1.5 seconds or less. In order to accomplish it, we had to see how much liquid was going to be pumped to the bottle as well as the characteristics of it. The pump maximum capacity is: 15L/min at 100%. Meanwhile, the bottles have a maximum volume of 260 ml, but their filling volume is 245 ml, so we needed to make that in this station, the bottle could get above 245 ml in less than 1.5 seconds. If we convert the pump capacity to ml/s we get 250 ml/s. We can get the transfer function of the plant, by analyzing :

$dV = Q_{in} - Q_{out}$ → because the bottle doesn't has a hole, the formula will look like:

$dV = Q_{in}$ → we can look at this as a simple integrator, making our transfer function: $\frac{1}{s}$

The characterization in open loop of the system will look as figure 18 with the output shown in figure 19, with a final value step of 250 and a noise factor of 0.5.

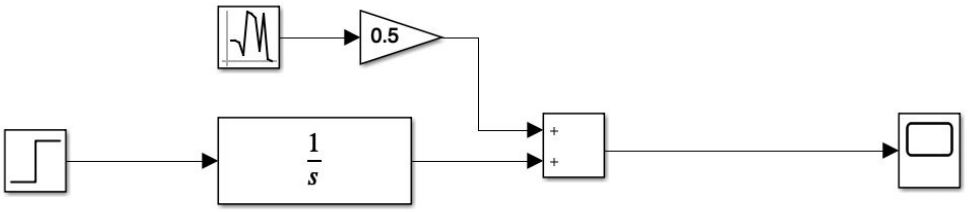

Figure 18. Filling station model in open loop

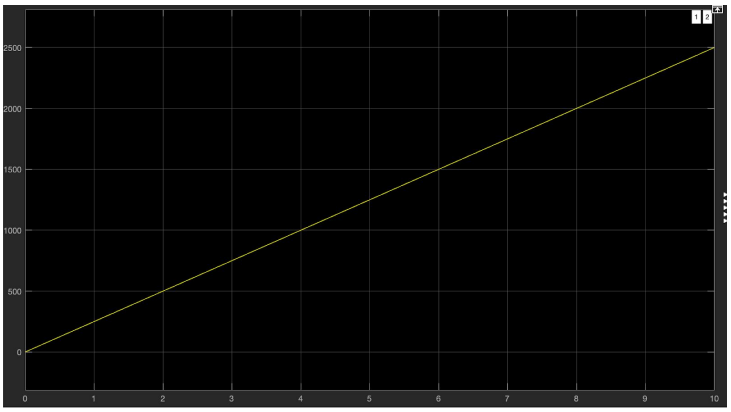

Figure 19. Output of system in open loop

This is a type 1 system, so we only need a proportional controller. We need to fill the bottle in less than 1.5 seconds, so by putting a proportional gain of 5, we can make sure that the system is going to have a quick response, making this desired output possible. The final model and output are shown in figures 20 and 21 with the noise factor included, also a final value step of 250 was added so that the machine doesn't goes above 260 ml or below 245 ml.

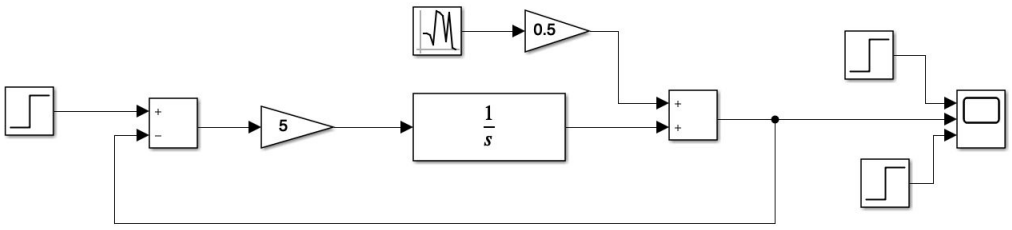

Figure 20. Filling Station Model

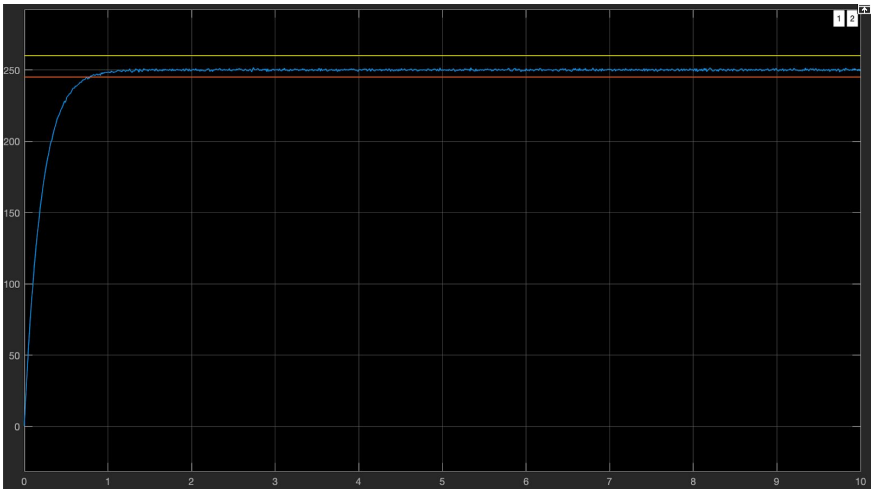

Figure 21. Filling station output

We can see at the output the two main important criterion to follow. This graph shows that the bottle at 1 second has already gotten more than 245 ml, and is staying at a point that is neither above the maximum volume nor at the minimum filling volume.

Heating Station

For this station, we had a reference to follow stated by the professor, which we can see in figure 22. In other words, this is the input for the system.

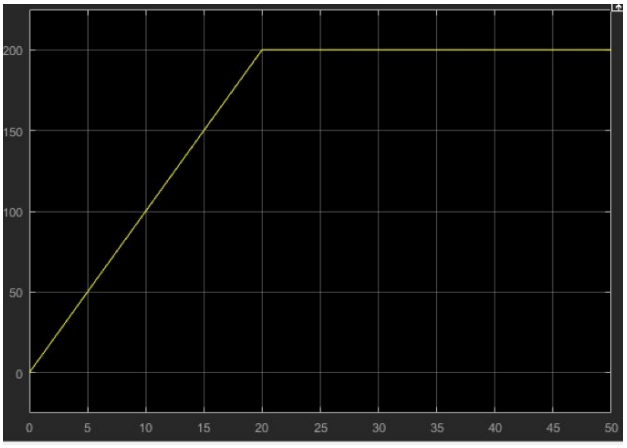

Figure 22. Reference to follow

To accomplish this in Simulink, we calculated the slope of the ramp which was 10. So we put two ramps, the first one starting at 0 and the second one was subtracted, starting at 20.

The other graph established by the professor was the impulse response of the system, seen in figure 23.

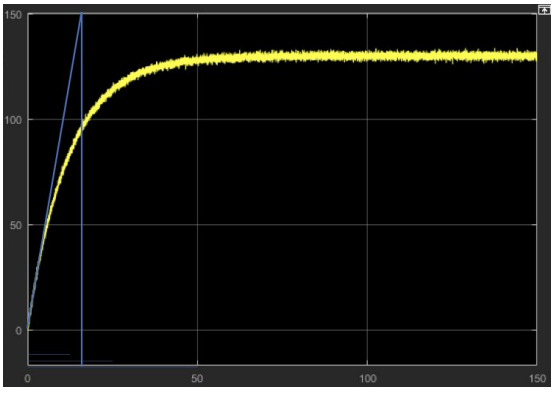

Figure 23. Impulse response

With the following formula, we were able to obtain the transfer function of the first order system given to us.

$G(s) = \frac{k}{Ts+1}$ → By looking at the graph we can estimate that $T = 10$

Therefore: $G(s) = \frac{130}{10s+1}$

The problem stated that the response of the system looked as figure 23 when the heater was at 40% of its capacity, so if we want to calculate what the system will work with 100% of capacity, k would now be: $\frac{130}{0.4} = 325$

Therefore: $G(s) = \frac{325}{10s+1}$

The characterization of the system in open loop and its output with a noise factor of 1 will look as figure 24 and 25.

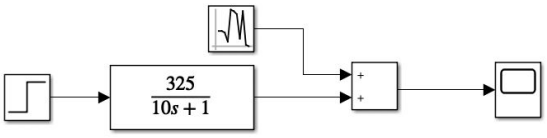

Figure 24. Heating station model in open loop

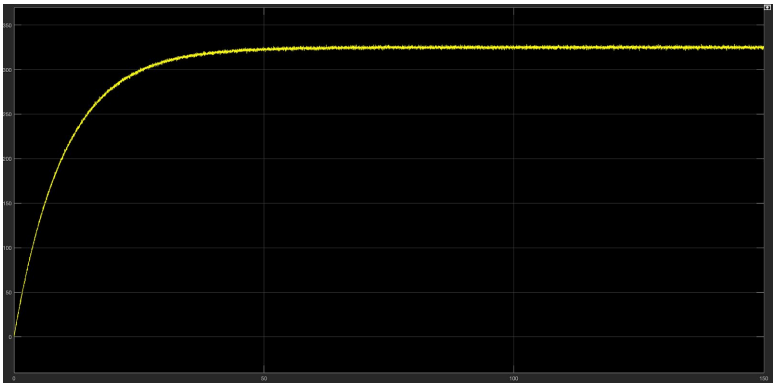

Figure 25. Output of system in open loop

For the controller of this system we used a PI controller, and manually changed these two gains so that we could obtain a graph that didn't oscillate much and that could follow the reference given with minimum error. The calculated gains where:

Proportional gain: 0.05

Integral gain: 0.009

Now, the heating station with the reference, the controller and the plant with our function, will look like figure 26, when the station is at 100% of its capacity. After the transfer function, a noise factor of 1 was added for the final output.

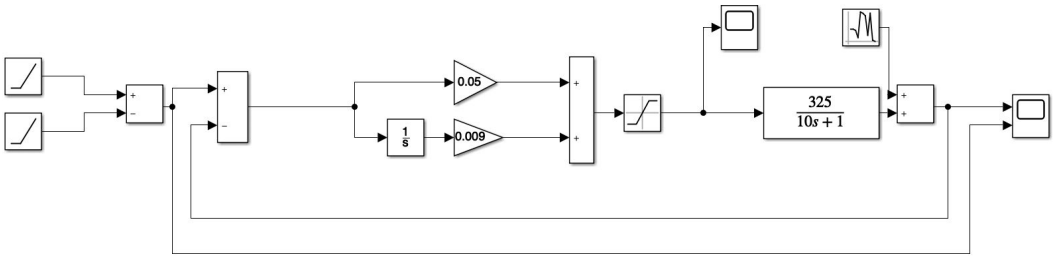

Figure 26. Heating Station Model

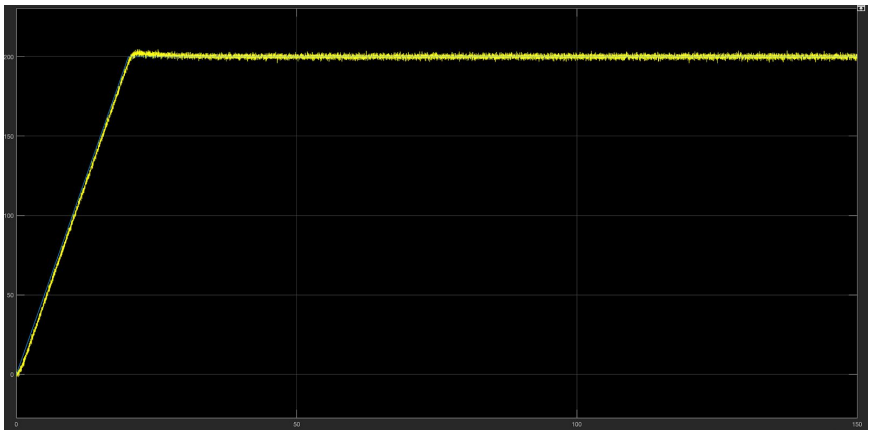

Figure 27. Output of heating station

By looking at the graph we can say that the PI made the system follow the established reference. We can see that at 20 seconds, the temperature elevates to about 203º but after that small peak, it returns to 200º and after it gets to the maximum temperature it stays there for more than 25 seconds as requested.

As we could see in figure 26, a saturator was placed between the controller and the transfer function. This saturator helped to make sure that the plant wasn't giving more than the 100% requested. If we take a look at the input, we can see that the plant is working well and isn't saturated although some noise is disturbing it, so at the end it's not working all the time at a 100%.

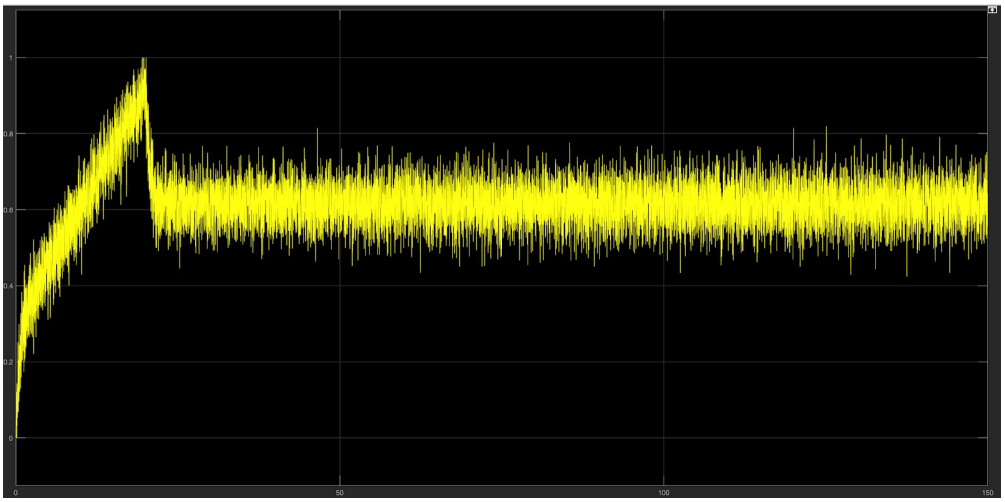

Figure 27. Output of the saturator

Cooling Station

The goal of this station was to cool down the bottle that passed through the previous heating station. As well as in the heating station, for this station we had a reference and an impulse response given by the professor, seen in figures 28 and 29.

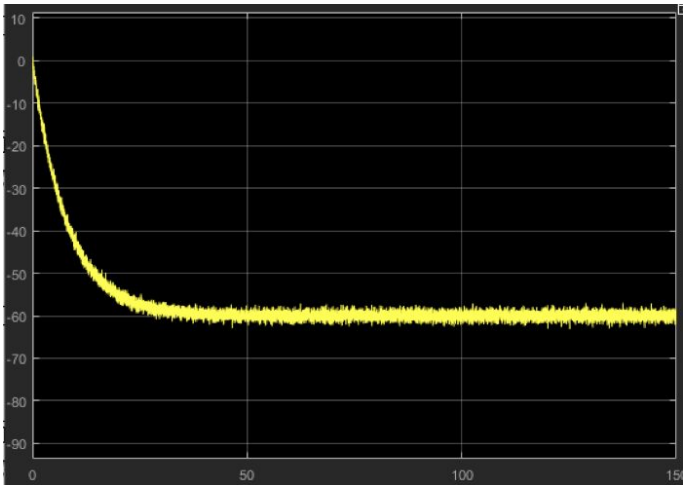

Figure 28. Impulse response

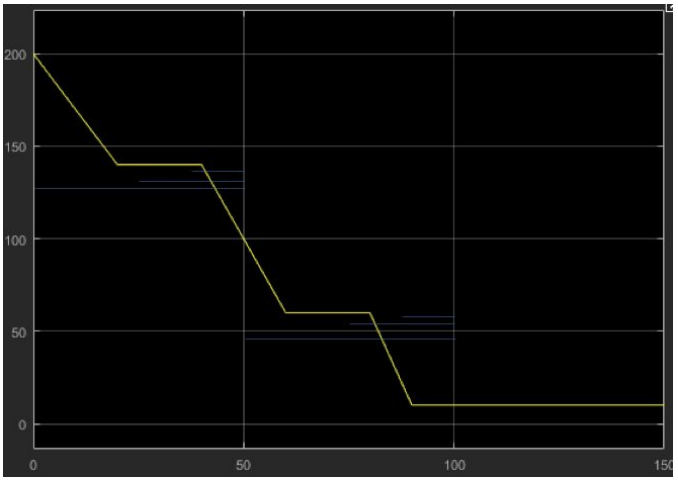

Figure 29. Reference to follow

After calculating the slopes by looking at the figure of the input of the system, we used 6 slopes, with different starting times, and a step to make the graph start at 200º.

With the following formula, we were able to obtain the transfer function of the first order system given to us.

$G(s) = \frac{k}{Ts+1}$ → By looking at the graph we can estimate that $T = 10$

Therefore: $G(s) = \frac{60}{10s+1}$

The problem stated that the response of the system looked as figure 28 when the cooler was at 25% of its capacity, so if we want to calculate what the system will work with 100% of capacity, k would now be: $\frac{60}{0.25} = 240$

Therefore: $G(s) = \frac{240}{10s+1}$

The purpose of the cooling system was to cool down the temperature, so if we wanted to graph this output, we could put a negative sign to the function, nevertheless this would make our system unstable, so to make sure that our transfer function looked good in open loop, we just added a negative gain at the end. The response in open loop and it's output will look as shown in figure 30 and 31.

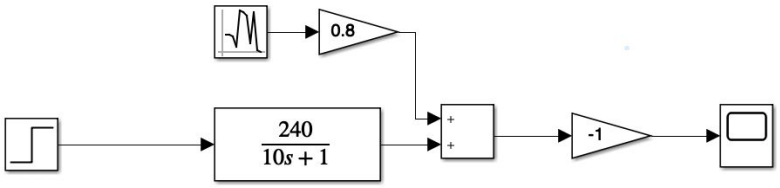

Figure 30. Cooling station model in open loop

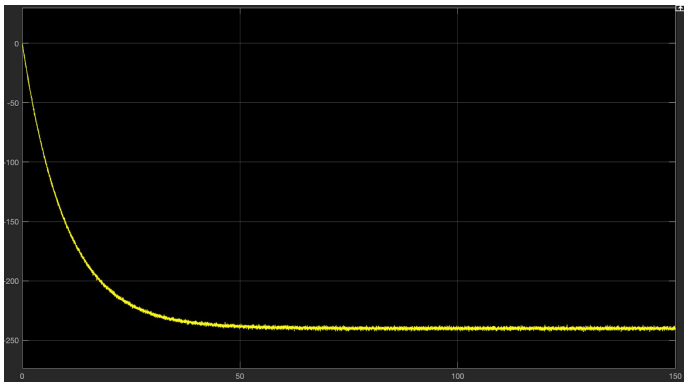

Figure 31. Output of the system in open loop

For the PID of this system we used the algebraic method. We used the Root Locus Editor of Matlab and stated a pole at 0, and two zeros at -0.5 and -0.2, the resulting graph is seen in figure 32.

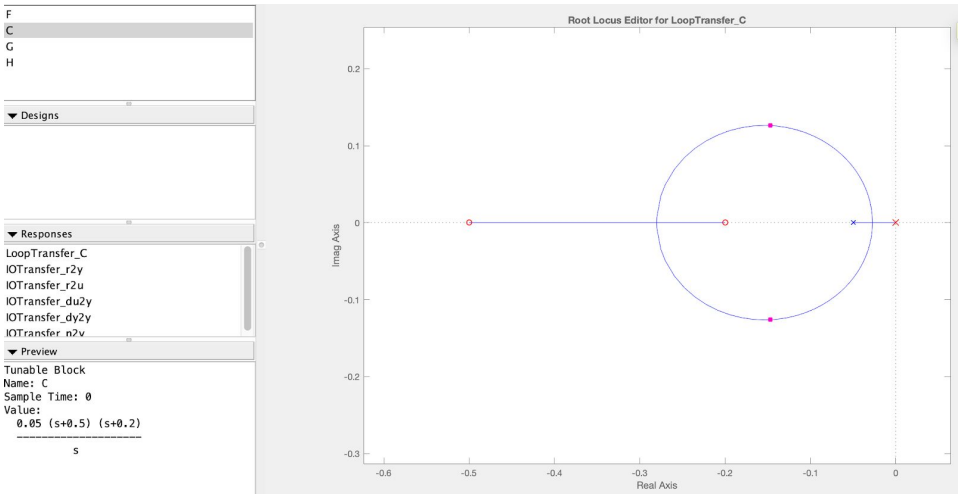

Figure 32. Root Locus Diagram

Now, going back to the requirements of this system, we know that a transfer function for default, starts at 0, so what we did was begin to simulate as if 0 was the 325º that the heating system gives us. What we did to still follow the reference from 0 was to begin the

reference in 0 instead of 200, but keep the slopes that made the system cool down 60º in each one.

Now, that the cooling station has already a reference, a PID controller and the plant with our function, when the station is at 100% of its capacity the model and output will look as figure 33 and 34. A saturator was added before the transfer function to control the percentage in which the machine was working so it didn't go above or below 0-100%. After the transfer function, a noise factor of 1 was added for the final output.

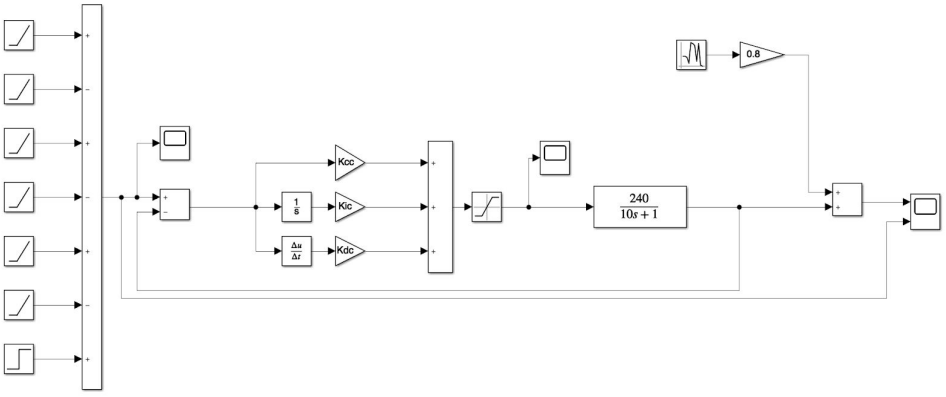

Figure 33. Cooling Station Model

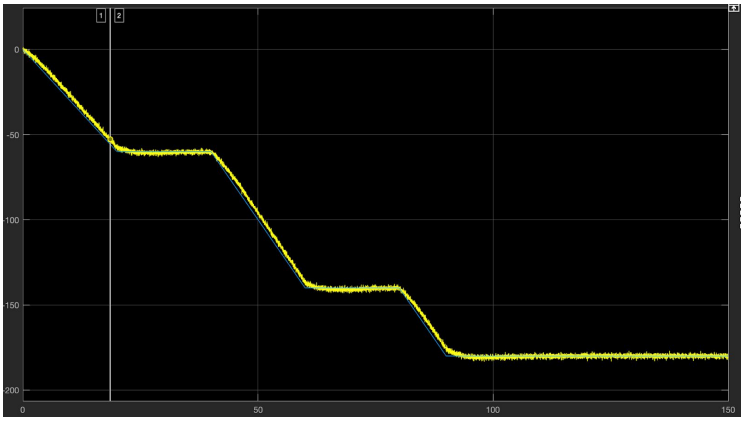

Figure 34. Output of Cooling System

Because the graph showed how the temperature got lower starting from 0, the saturator was set between the limits of [-1 0], and we can see in figure 35 how the output of the saturator keeps the reference with the controller between these limits.

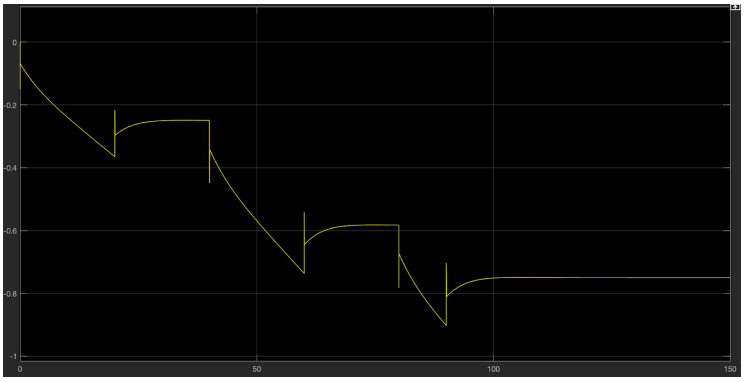

Figure 35. Output of saturator

Packaging Station

In the Packaging station, there are 2 DC motors:

- Rotation movement → It should rotate at 50 RPM
- Vertical Movement → It should follow 5 complete cycles of a sine wave with period equal t0 60 sec and amplitude from min to max equal to 20cm.

For this station we have x1 = Angular Position, x2 = Angular Velocity, and x3= Angular Acceleration.

When we have a transfer function G1, we can represent it in a canonical way as A1, B1, C1 and C2. Being :

```
A =
     0     1     0
     0     0     1
     0     0     0

B =
          0
          0
     0.2000

C1 =
     0     1     0

C2 =
     1     0     0
```

Vertical Movement Motor

In the packaging station we have this vertical motor, which will make the bottle go up and down so that the plastic covers it completely, and it is possible to measure x1, x2 and x3. We now propose a second function G2 with its own polynomials. In this case we used [-10 -11 -12], to get the characteristic polynomial that has as a solution those poles, we use the function of "poly" in Matlab with those proposed poles, getting: [1 33 362 1320]

Our matrix A is already in a canonical form, so matrix Ac is [0 0 0], so A2c with our solution of the recent poles we state A2c as [-1320 -362 -33].

We know that $\dot{x} = Ax + Bu$ and the output is $y = x$.

For making this model, we added three derivatives, each one with a saturator, that was set between -1 and 1, so that it can follow correctly the sine wave.

We now define the errors in the form of $e_1 = x_1 - r$, by putting the subtracting bloques between each of the x's and the derivatives that were put above them. The errors are connected to a mux which will be attached to the gain obtained with help of the proposed poles (A2C). Now the sum of those three signals will make "u". So the model will finally look as figure 36.

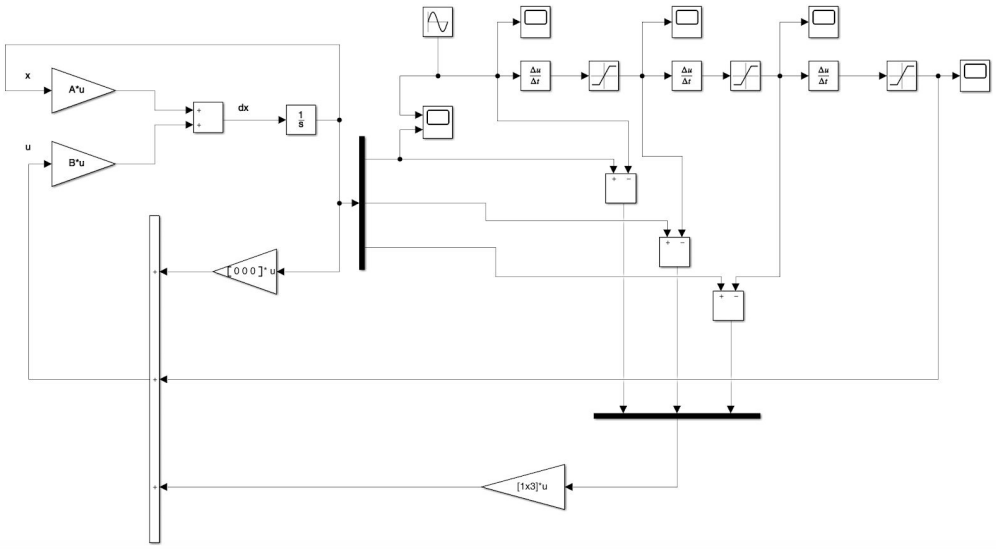

Figure 36. Vertical movement motor model

For the sine wave we state as amplitude the 20 cm, and because that Simulink bloque doesn't use periods as a parameter, we convert it to frequency being it: $\frac{20\pi}{60} = 0.1047$

If we want to check that x1 is following the sine wave reference, we connect a scope between x1 that was obtained as the decomposition of x in its state variable and the reference. We can appreciate in figure 37 that x1 follows the sine reference.

Figure 37. Output of x1

Rotation Movement Motor

This motor will make the bottle rotate so that the plastic covers all its surface, and it is possible to measure x1 only.

As stated on the theoretical framework, an observer is a dynamical system designed to estimate the state x(t) of $\Sigma(A, B, C)$

Unlike the previous motor, we will state our system as $\dot{o} = Ao + Bu + L(y - s)$. Being $s = c \cdot o$. Our observation error will now be $e_0 = x - o$, so if we derive it we will get $\dot{e}_0 = \dot{x} - \dot{o}$, therefore $\dot{e}_0 = Ax + Bu - Ao - Bu - L(y - s)$.

With the “ctrb”, “rank”, and “obsv” functions, we check that the system is controllable and observable. After this we design a feedback gain with systems A, B and poles in closed loop in [-2 -1 -3], with help of the “acker” function of Matlab. For the observation gain, will be

obtained with the transposed form of A and C2. For the new set of poles, we need them to be at least 5 times faster than the previous ones, so we proposed [-9 -10 -11], after this we transpose the whole function.

What we did after establishing those things, we could now move on to the design of the entire rotation movement motor. We have a small design of the system in state space, that looks as figure 38.

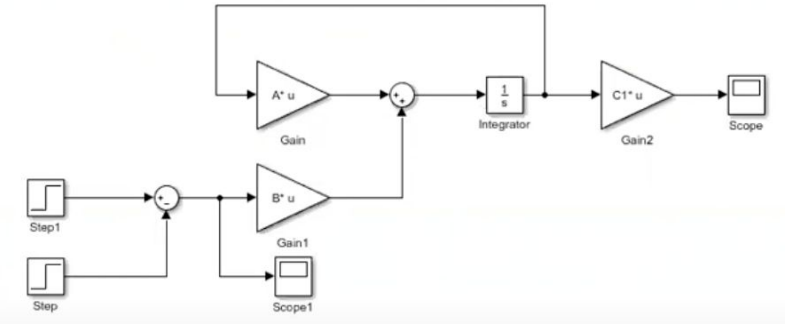

Figure 38. System in state space

We copy this model, because basically our observer is a copy of the system. The observation gain was implemented so that the output of Y and S could be subtracted and inserted into the gain. We have now in the observer, A times o, plus B times u, plus L that multiplies Y - S.

Because we want to know the velocity and not the position we put a ramp with a slope of 50 as the reference of the system, this could be made also with a step and an integrator. Two demuxes were placed for x and o, so that we can estimate each variable. The final model will look as figure 39.

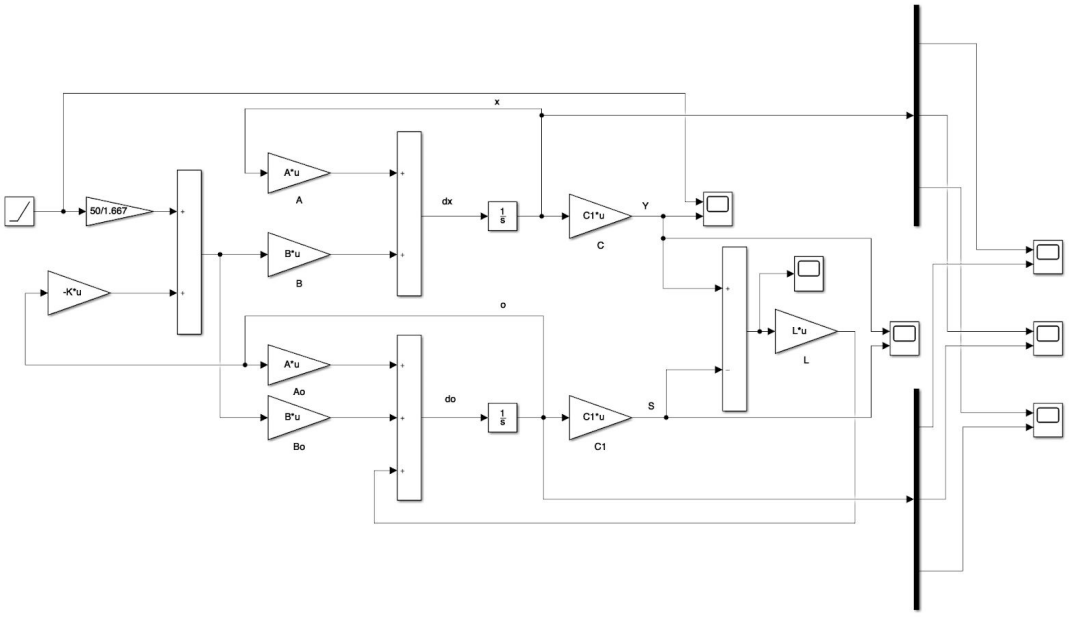

Figure 39. Rotation movement motor model

As we can see in the previous figure, a gain was added after the step unit. The problem stated that x2 had to follow 50 rpm, so for this we measured x2 first without the gain, showing an output as seen in figure 40.

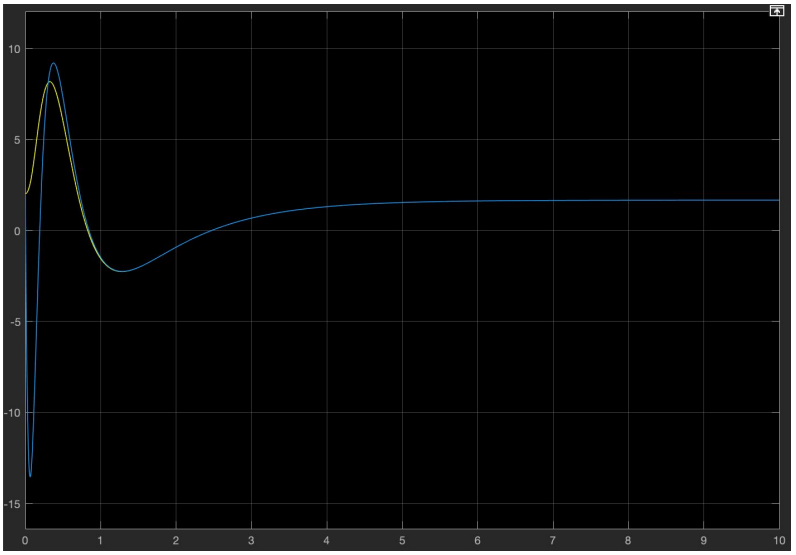

Figure 40. Output of x2

We can see that x2 isn't following the reference of 50, so what we did is establish a gain in which we divide 50 by the final value given in figure x this case 1.667. After establishing this gain, x2 started following the reference. We can see on the following figures 41, 42 and 43, the three x's with the three o's from the observer, and in figure 42, we can also see how x2 is following the 50 rpm.

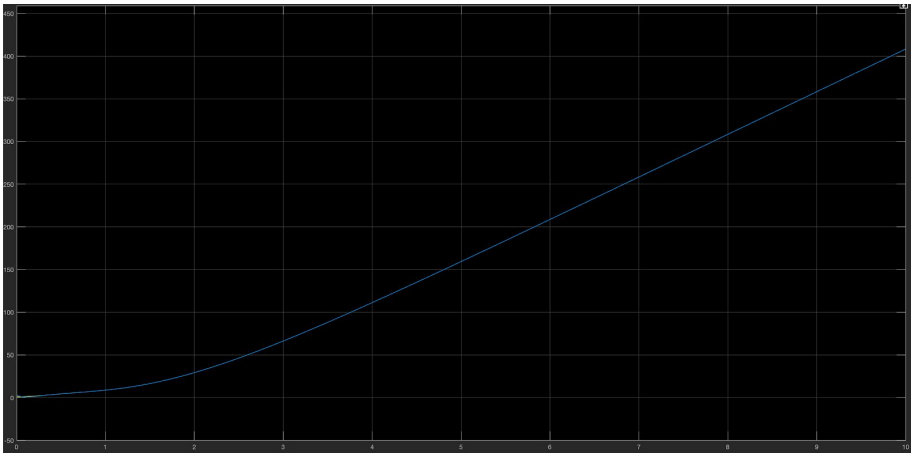

Figure 41. Output of $x_1 \cdot o_1$

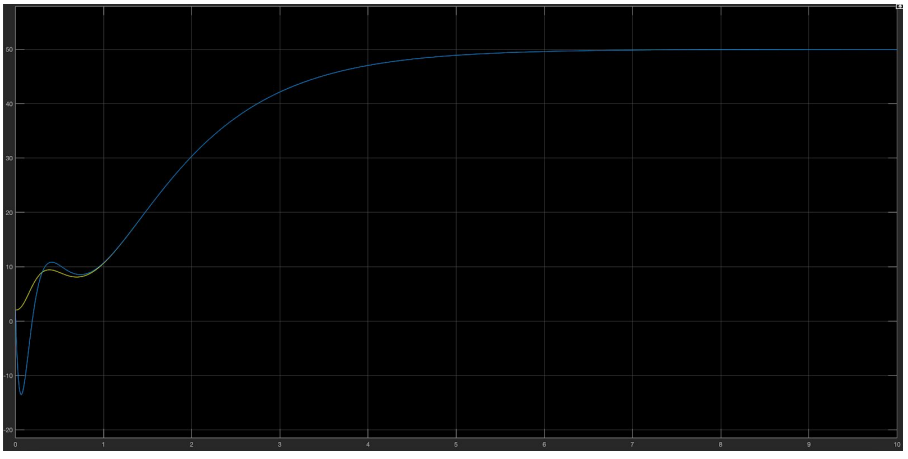

Figure 42. Output of $x_2 \cdot o_2$

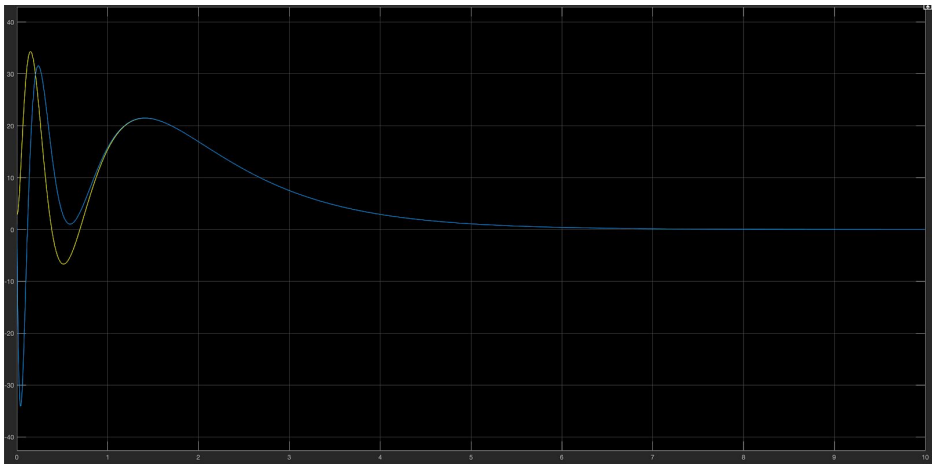

Figure 43. Output of $x_3 \cdot o_3$

Explanation Video

https://youtu.be/NjBbTikllik

Conclusions

After making this project by integrating the majority of the concepts seen in class, we can see how we can use them in real applications. Like regulate systems with a PID controller and looking how each gain affects the system by using a control loop feedback mechanism to control process variables. Or having sense of what an observer is needed for, for example when environments are as complex as they are today, you can't just monitor the known problems. So without an observable system, you can't know what is causing the problem and you don't have a standard starting point to find out what is affecting the system. We think that this project helped us to understand better all the concepts that we saw in class and how to do a correct implementation in real life. Now, we can have a better picture of how all the systems could work in an engineering point of view, why they work as they do, and how external factors could affect for good or for worse these systems.

Bibliography

Ogata, K. (2002). *Modern Control Engineering*. (4th edition). New Jersey: Pearson Education International.