Bellato Filippo 21614 - Outmani Ayoub 21620

Control theory report

Structure of the python code

- **package_LAB.py**: python package where we implemented our functions
- **package_LAB.ipynb**: jupyterlab file where we show the functions we implemented
- **Simulation_OLP.ipynb**: jupyterlab file where we used the drawing functions to draw the different models of the open loop.
- **Simulation_CLP_PID_FF.ipynb**: jupyterlab file where we simulated all our experiments
- **TCLab_CLP_PID_FF.ipynb**: jupyterlab file where we implemented all our experiments on the TCLab
- **Stability_Margins.ipynb**: jupyterlab file where we calculated the stability margins of the process

Objective

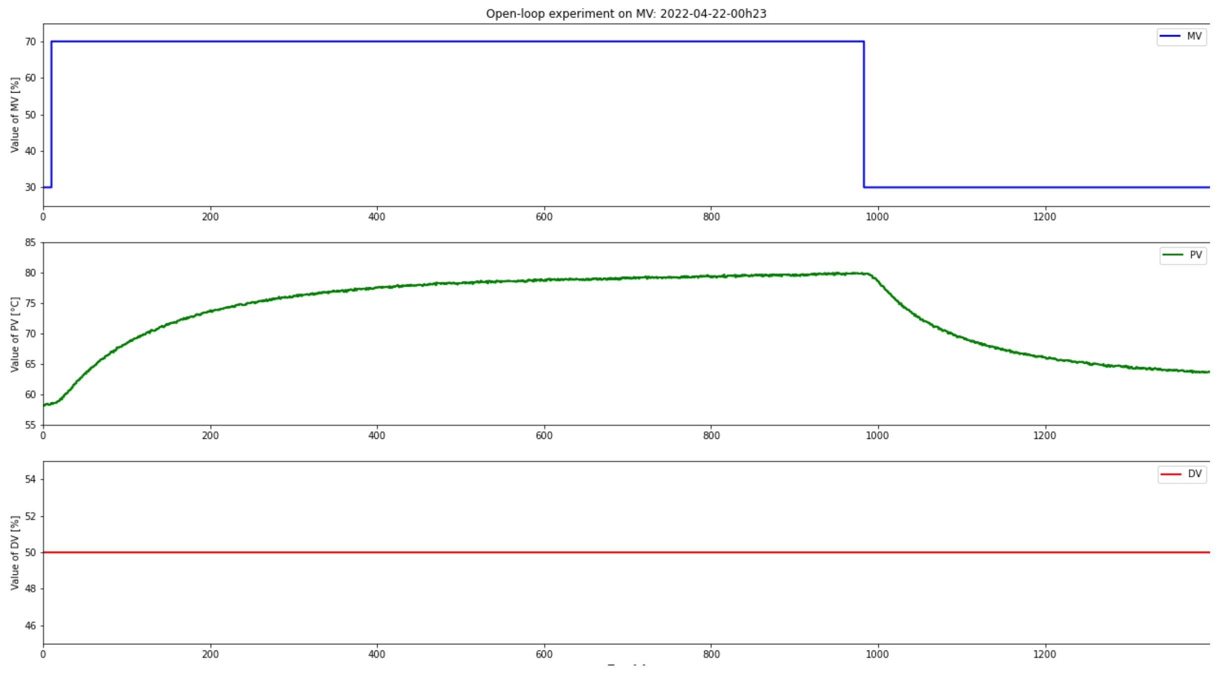
The objective of this report is to illustrate and discuss the results obtained in the laboratory with TCLab.
In this laboratory we had the opportunity to apply what we learned in class regarding process behavior and process control through simulations using the anaconda environment where we implemented a PID function. The second step was to test these simulations with a concrete example of process behavior with disturbance. We did this thanks to TCLab, a device with two engines that can be controlled to heat up a metallic bar while measuring the temperature. We were able to use one engine as the process(P(s)) and the second engine as a disturbance (D(s)). This allowed us to test our PID function in real-time while having the results displayed on screen.

Step response from MV to PV

The first step of controlling a process is its knowledge. There are many aspects in knowing a process behavior which are: physics behind the process, nonlinearities, disturbances and constraints. These aspects influence both the structure and type of control that we want to apply and the ultimate goal of the control: disturbance rejection or set point tracking.
In this laboratory we focused our attention on all these aspects except the physical knowledge of the process, that for our study case could be ignored.
Since TCLab isn't a linear system we proceed by deciding a working point, around which the system can be approximated with a linear model. We decided to set the working point at 50%(for both MV and DV) that gave us a PV of around 70°C. The constraints of MV(specific for TCLab) was the engine power, which could only go from 0%(engine off) to 100%(engine at max power). The upper constraint on PV was however not tested, as to not risk damaging the chip. It is obvious that the lower bound on PV was the room temperature(around 23°C at the time of the experiments). Finally, the disturbances applied on the process(usually wind or

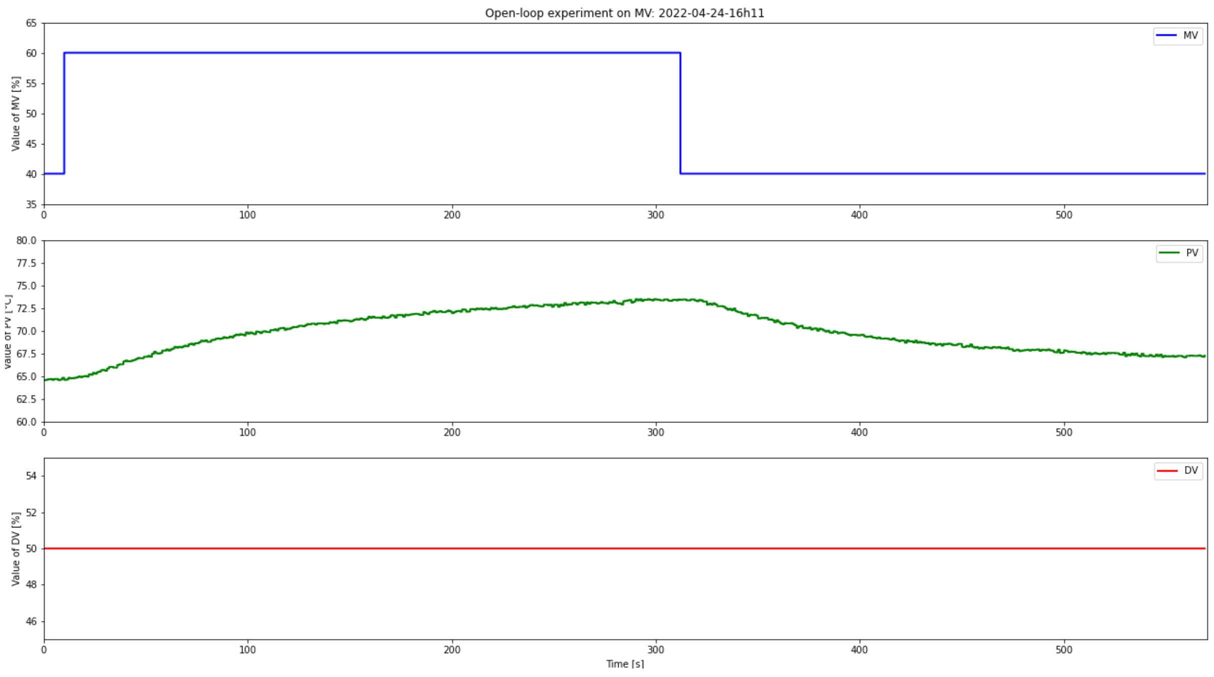
sun) were minimized by creating a plastic cover that was put on top of the TCLab at the time of the experiments, so that even an involuntary blow of air in during the experiment wouldn't alter the behavior of the process. This decision impacted our system behavior, as the time of stabilization and cooling down was dilated since the heat couldn't dissipate as quickly. This can be seen very clearly in the next paragraphs.

We began by plugging the TCLab on our computer and by running the program provided to us. This program would bring the TCLab close to its working point and then apply a step on MV. In the example of a delta of 20 around the working point, after stabilization, MV would go from 30%(50 - 20) up to 70%(50 + 20) with a step.

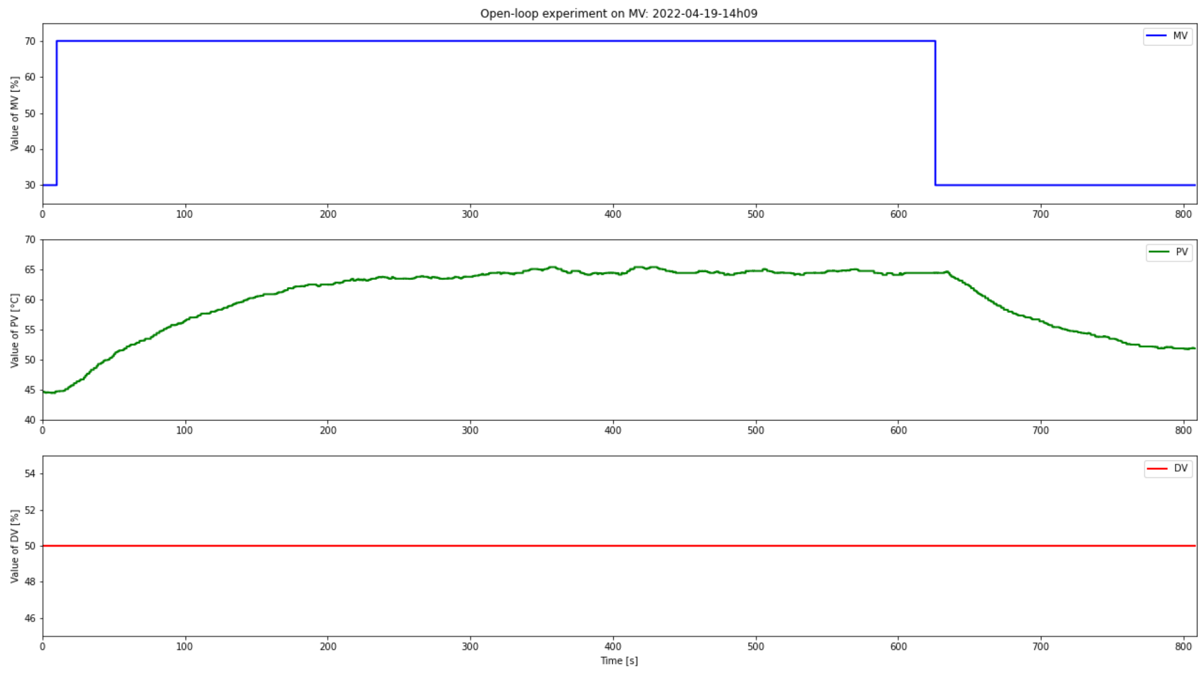


(MV from 30 to 70)

We decided to try a delta of 10. The results were similar but since the scale of the y axis was smaller, the noise in the measure appears more evident.

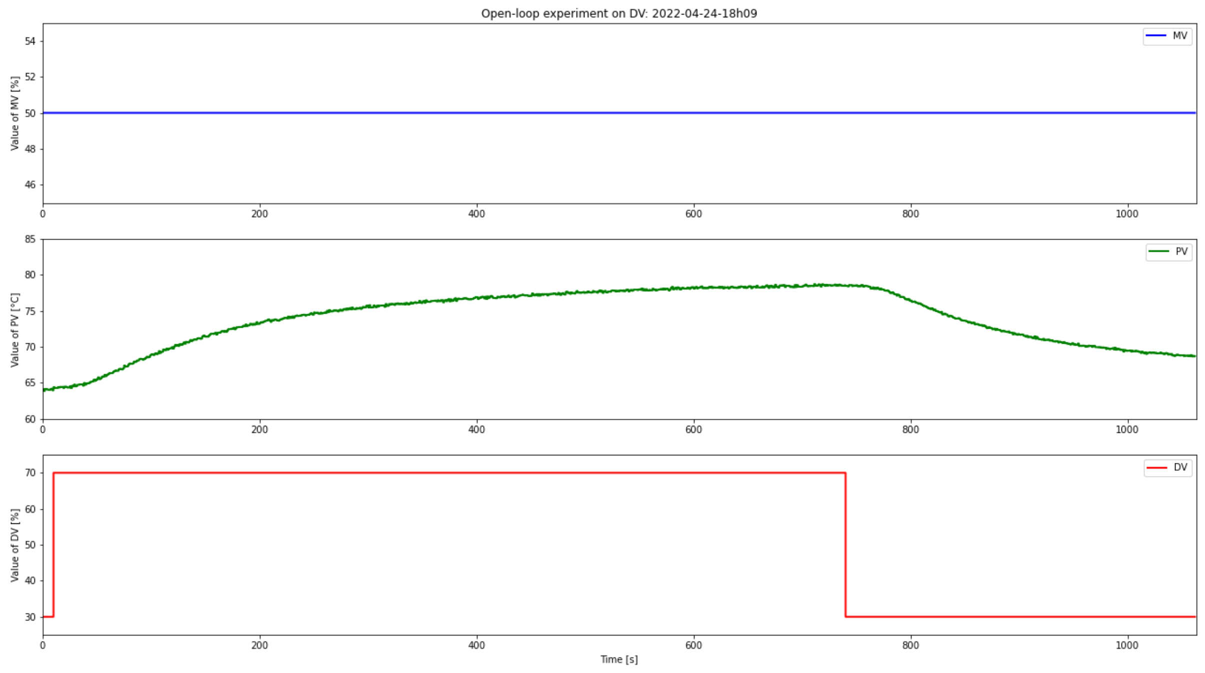


(MV from 40 to 60)
The graphs describe the process when subjected to a step input. From this model we were able to calculate its parameters and compute P(s).
For reference, the image below shows the graph of the process without the plastic cover. It is very obvious that the disturbance on the TCLab had a very big impact, especially in the settling time.

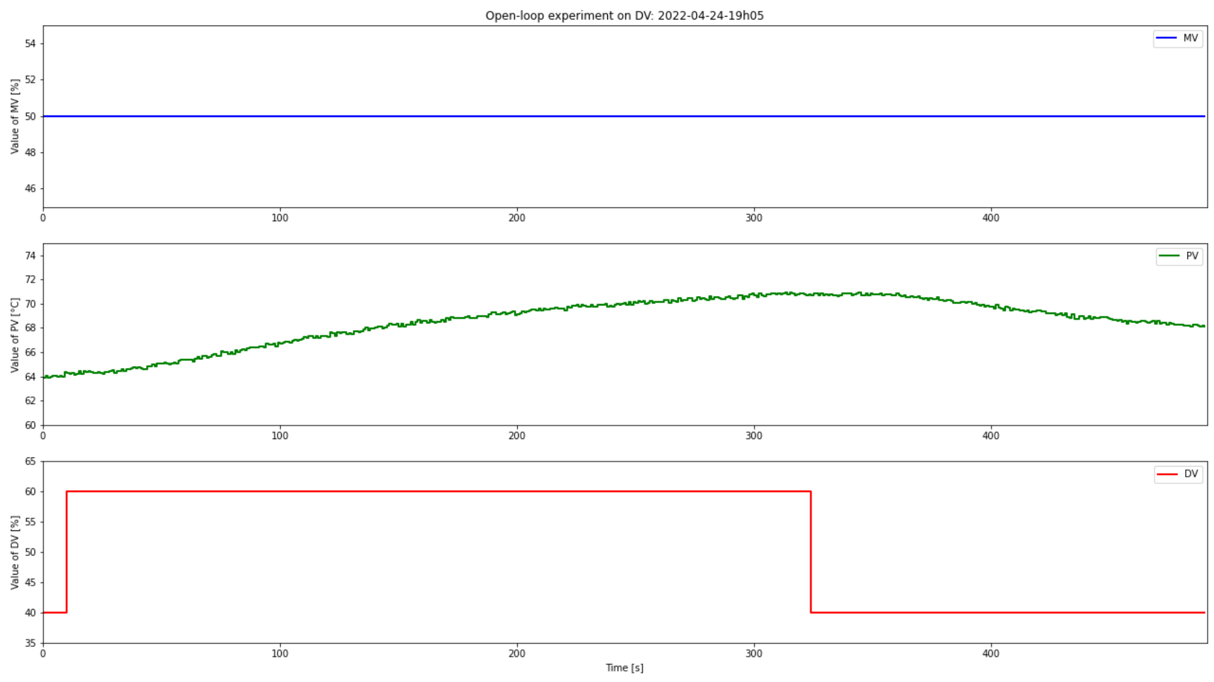


(MV from 30 to 70 without plastic cover on the TCLab)

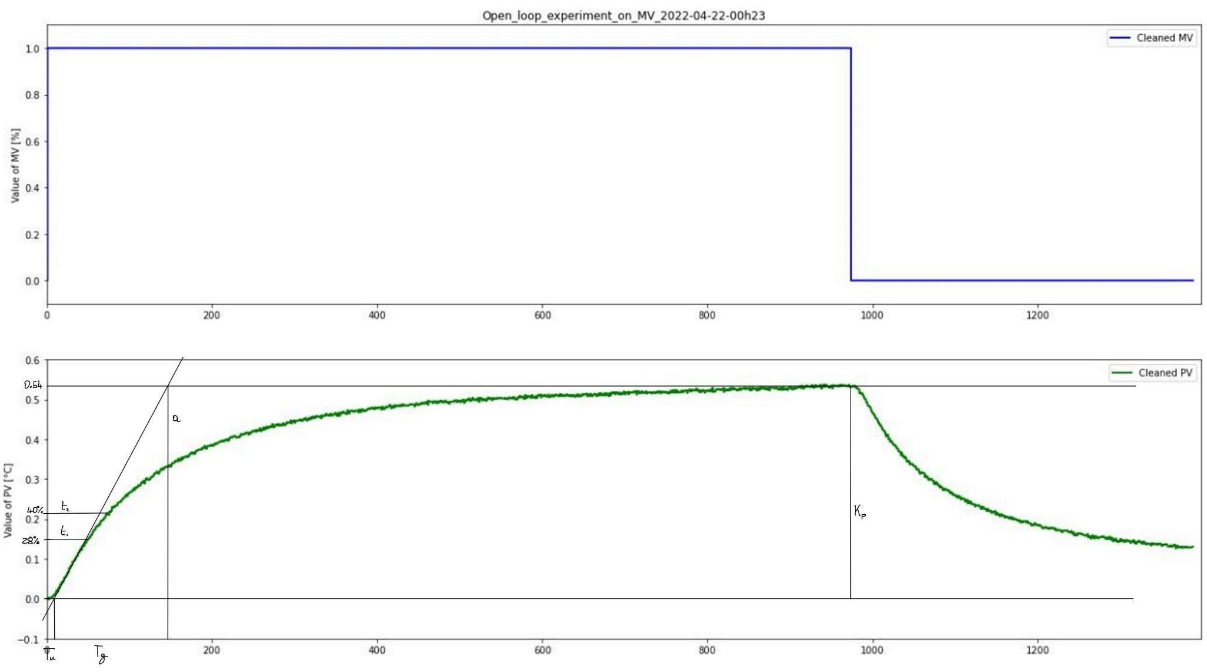
After this, we did the same thing but with DV changing around the set point. Also in this case we later used the parameters to compute D(s).



(DV from 30 to 70)



(DV from 40 to 60)
We calculated the parameters to compute D(s) with the SOPDT identification tool provided to us. (Kp = 0.39, T1 = 225.66, T2 = 8.13, theta = 1.12)
The second step of this first part was to calculate the parameters of the step response on MV to compute different models using different methods: Broida, van der Grinten, and Strejc. We first cleaned the data using the function provided to us. This allowed us to have a MV with a step of 1 and PV going from 0 to 100. We then drew the tangent on the curve and calculated all the parameters necessary, specified in the slides.



We measured the parameters with a ruler to obtain:
a=1,7 cm
Kp=4,6 cm
Tu=0,15 cm

Tg=2,4 cm
T1=0,9 cm
T2=1,3 cm
We then scaled these parameters to match the scale of the cleaned graph and obtained:
a=0,2
Kp=0,54
Tu=8,47
Tg=135,6
T1=50,85
T2=73,45
We then computed the 4 different graphs using the 4 different methods.

Broida model: first order plus delay

$$P_1(s) = 0.54 \cdot \frac{1}{135.6s+1} \cdot e^{-8.47s}$$

$$P_2(s) = 0.54 \cdot \frac{1}{124.3s+1} \cdot e^{-10.17s}$$

van der Grinten: second order plus delay

$$P_3(s) = 0.54 \cdot \frac{1}{(55.4s+1)(40s+1)} \cdot e^{4s}$$

The computation of this model didn't work since the delay was found negative, which is absurd since time cannot be negative.
Strejc model: first order with delay

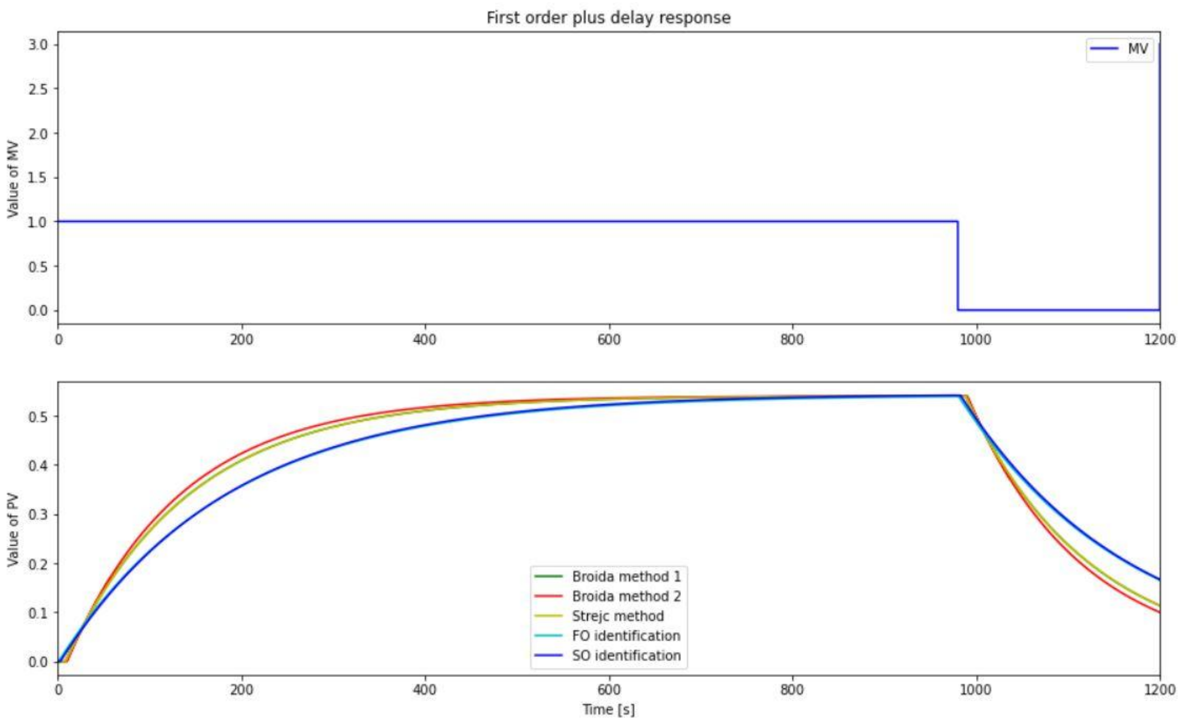
$$P_4(s) = 0.54 \cdot \frac{1}{(135{,}6s+1)} \cdot e^{-8.47s}$$

First order identification model:

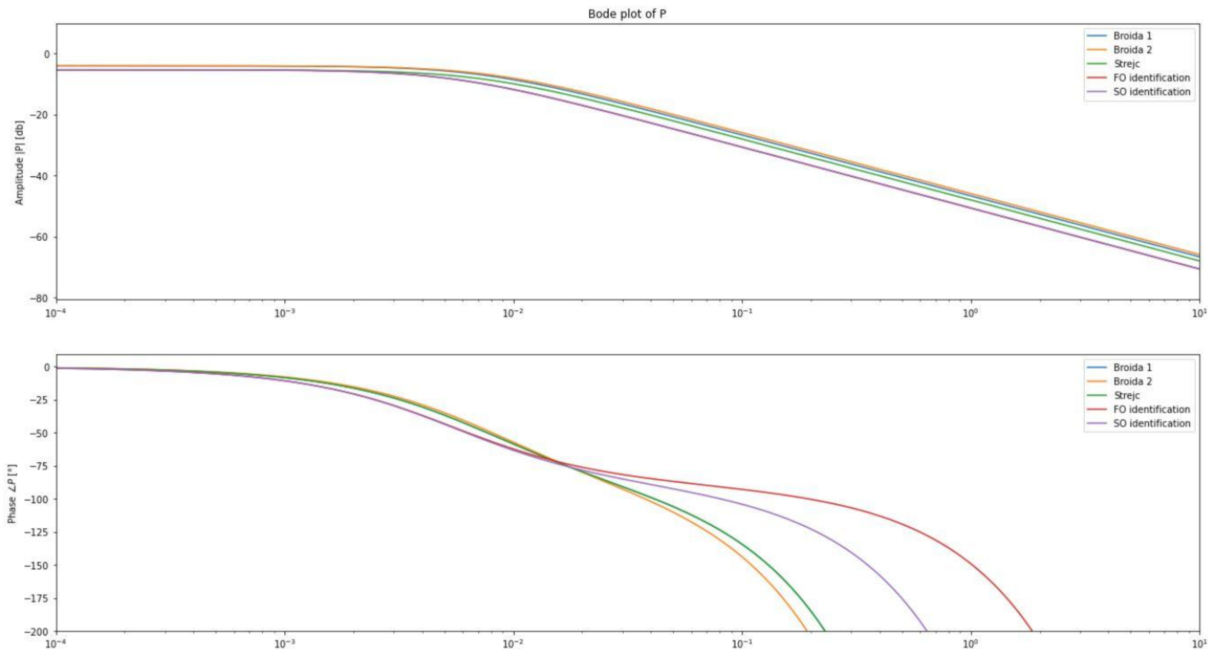
$$P_5(s) = 0.54 \cdot \frac{1}{(185s+1)} \cdot e^{-1s}$$

Second order identification model:

$$P_6(s) = 0.54 \cdot \frac{1}{(184s+1)(0.00006s+1)} \cdot e^{-3s}$$



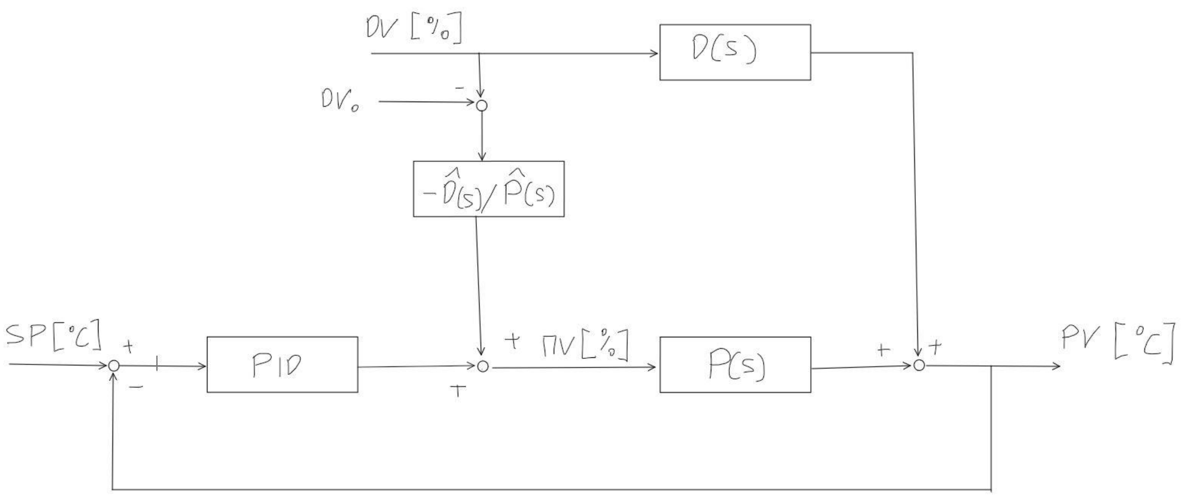
In the graphs Broida method 1 and Strejc method give exactly the same result. The same can be said for FO identification and SO identification. This is because TLag2 of the SO identification is so small that it takes action at very high frequencies.



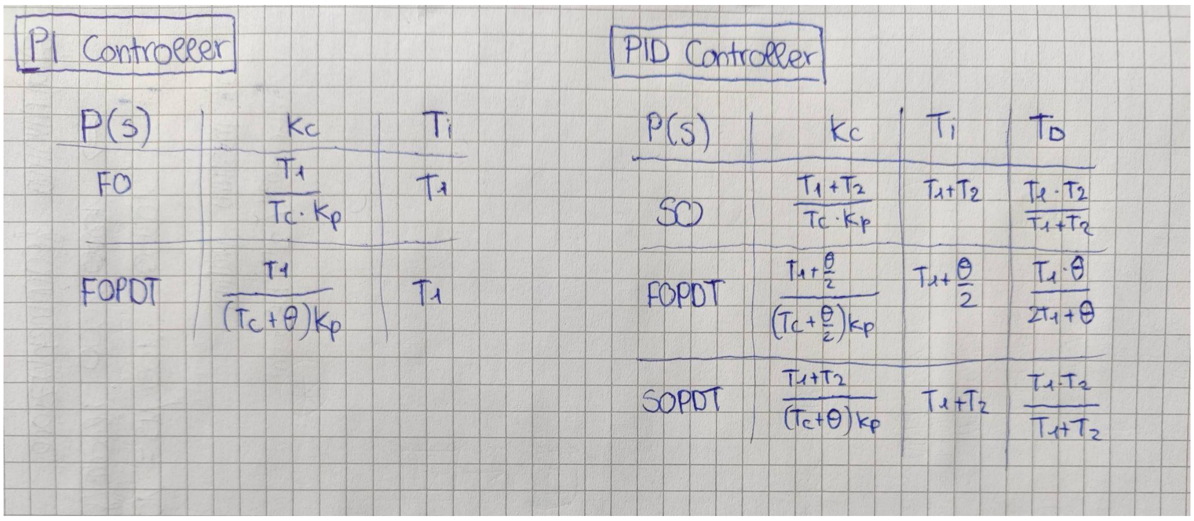
As can be seen from this bode diagram, the SO identification is very close to the other first order models in low frequency. The second pole of the SO identification is in fact at around 10^4.

PID controller

The goal of this second part of the laboratory was to create a PID controller, tune it, and use it in a real time simulation. First of all, we drew a model of the system that we were going to simulate. It consists of a process(P(s) which parameters we already calculated in the previous section), a disturbance(D(s) which parameters we also calculated in the previous section), our PID controller function and a feedforward mechanism to help with disturbance rejection.



To calculate the most appropriate PID time constants for our process we created an IMC tuning function which uses formulas drawn from an IMC tuning table. The result was obtained depending on the type of process we are dealing with(first or second order, with or without delay). The computation also depends if we implement a PID or PI controller.



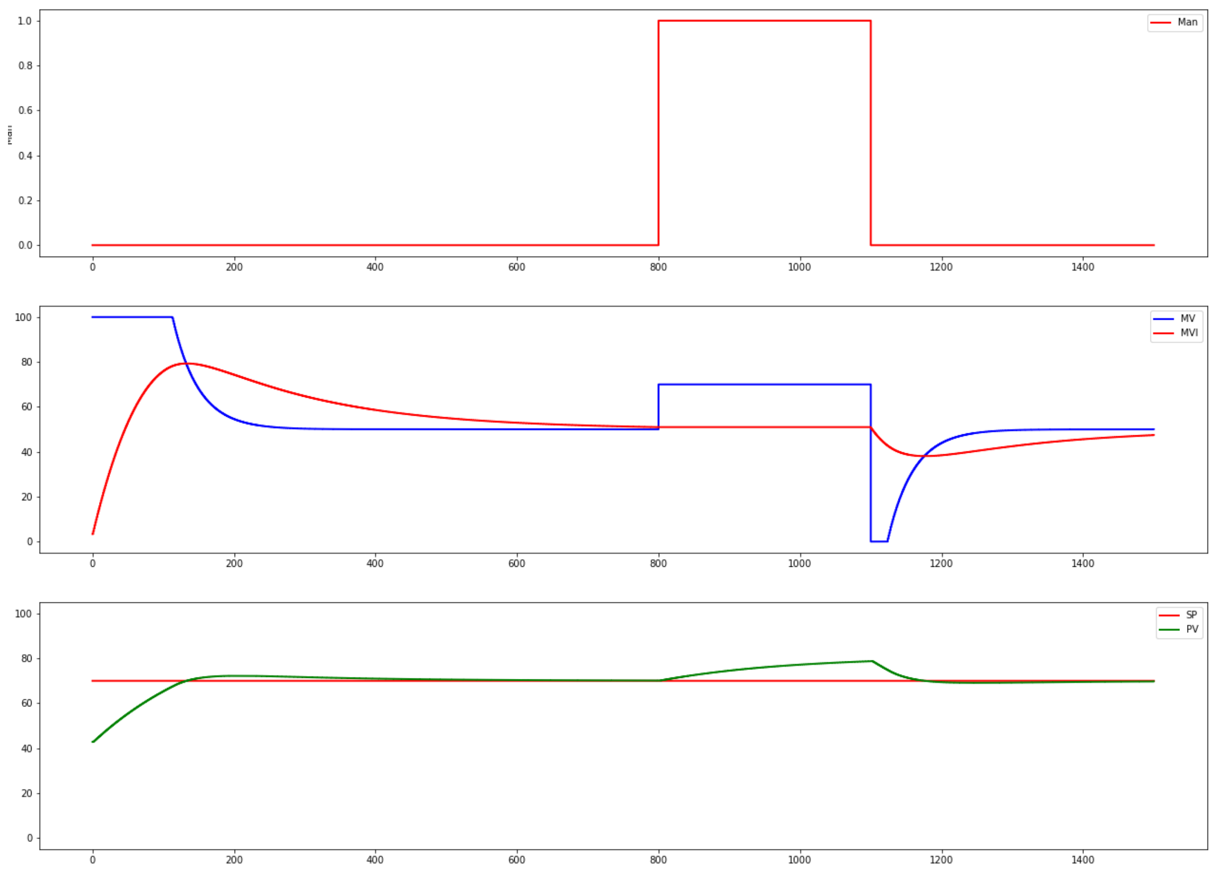
PI Controller

P(s)	Kc	Ti
FO	$\frac{T_1}{T_c \cdot K_p}$	T_1
FOPDT	$\frac{T_1}{(T_c+\theta)K_p}$	T_1

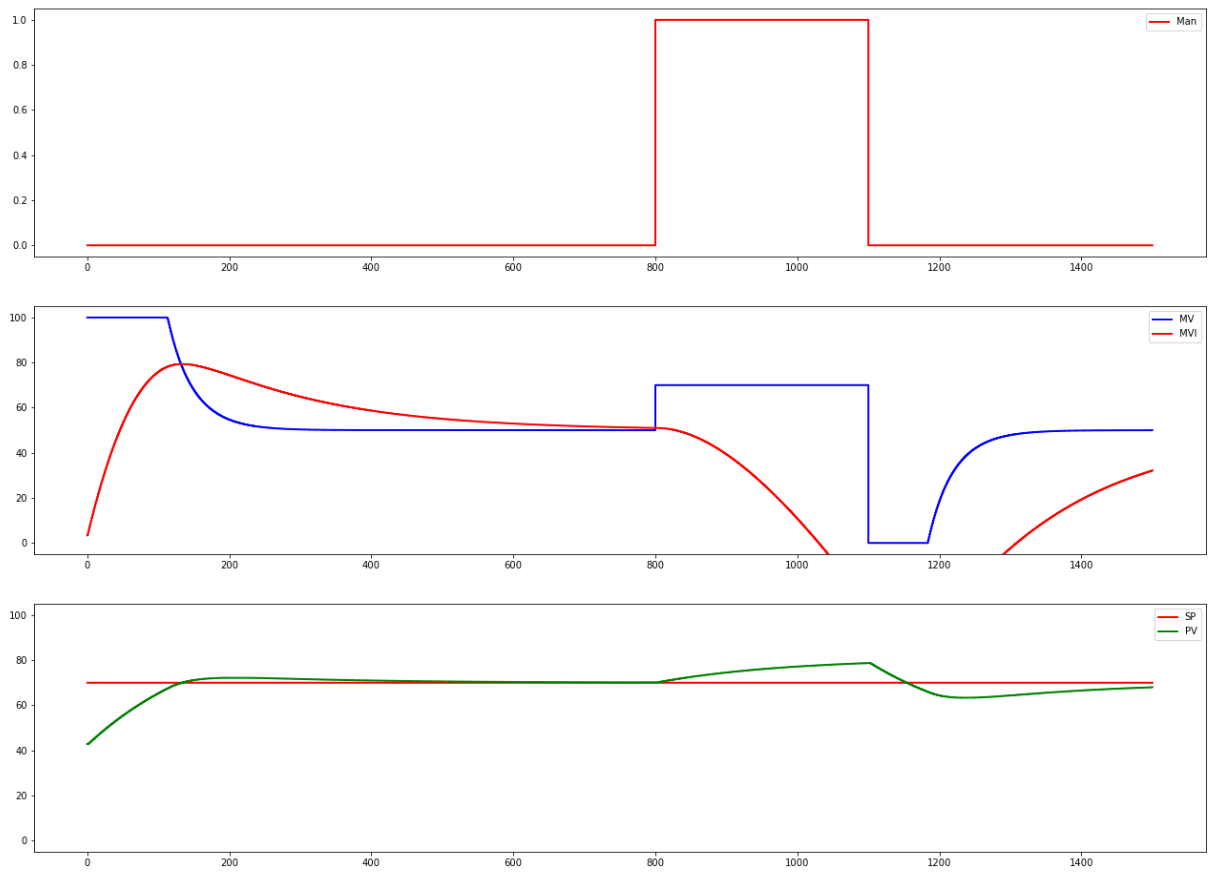
PID Controller

P(s)	Kc	Ti	TD
SO	$\frac{T_1+T_2}{T_c \cdot K_p}$	T_1+T_2	$\frac{T_1 \cdot T_2}{T_1+T_2}$
FOPDT	$\frac{T_1+\frac{\theta}{2}}{(T_c+\frac{\theta}{2})K_p}$	$T_1+\frac{\theta}{2}$	$\frac{T_1 \cdot \theta}{2T_1+\theta}$
SOPDT	$\frac{T_1+T_2}{(T_c+\theta)K_p}$	T_1+T_2	$\frac{T_1 \cdot T_2}{T_1+T_2}$

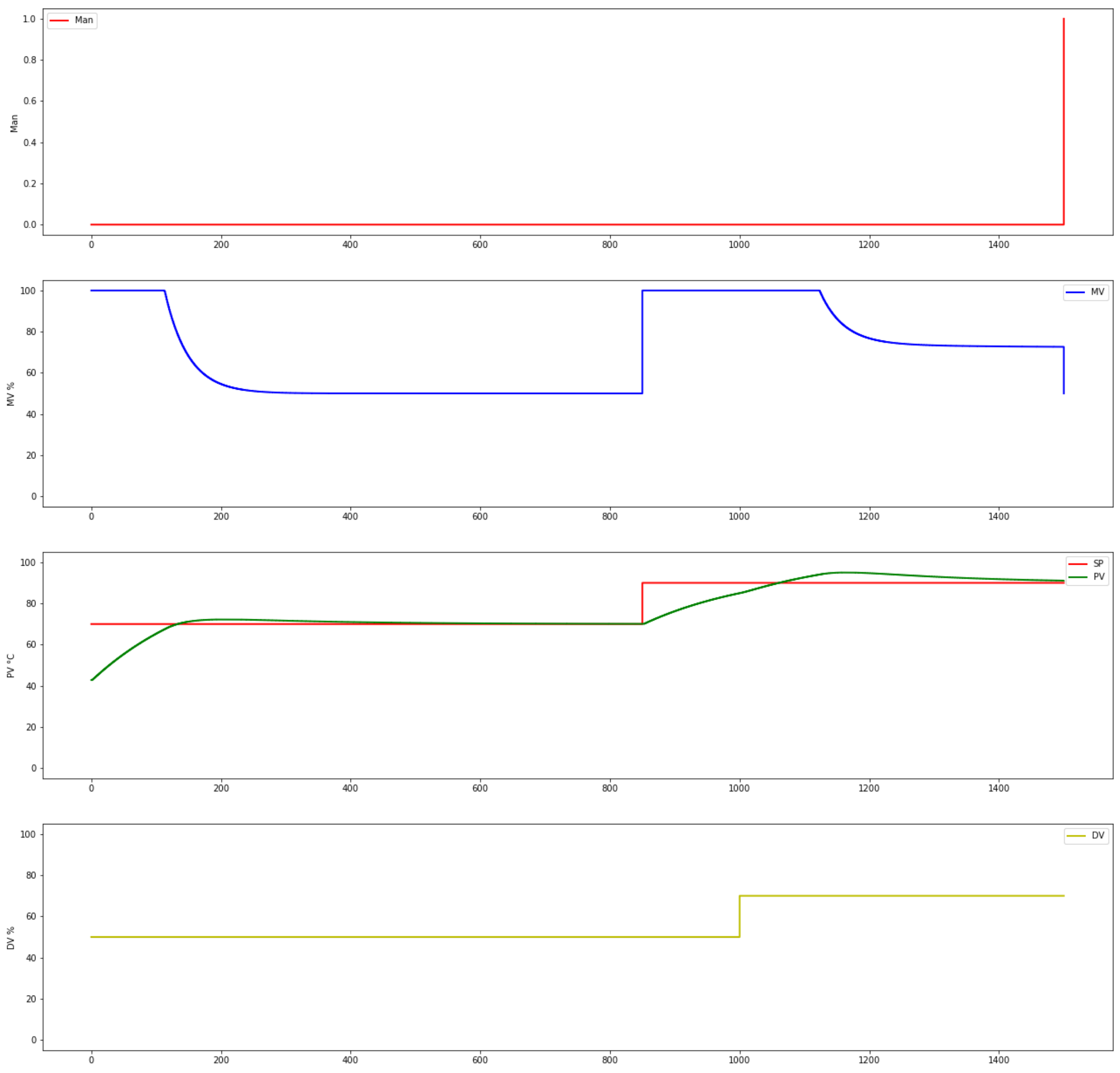
The PID function was created using the formulas provided in the slides, where each action(proportional, integral and derivative) has its own discrete time equation. The PID also has a saturation mechanism that takes action when the output is greater than the upper bound(or lower than the lower bound) of 100(0 for lower bound). It has a feedforward mechanism that allows the feedforward, computed in -D(s)/P(s), to contribute to the change in MV and reject disturbance. There is a manual mode that allows the user to specify and fix an MV. For this reason, an anti wind-up mechanism is in place, that makes sure that the error doesn't get integrated when the PID is in manual mode. This is done by leaving MVI constant while the PID is in manual mode.



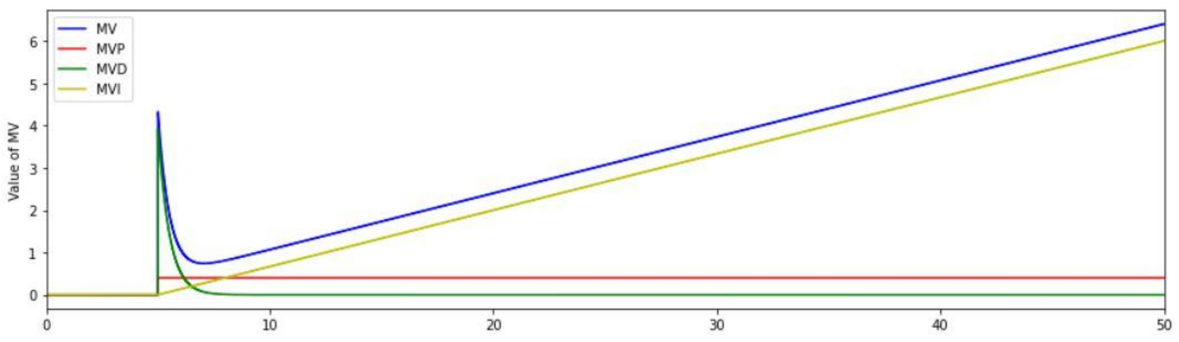
This graph shows the anti wind up mechanism in place. The controller is in automatic mode, then from 800 seconds to 1100 seconds the MV is manually changed to 70%. On the second graph we can see the value of the integral action in MV in a red line. As soon as the controller goes into manual mode, MVI stops increasing so as to not wind up.



In this example we purposely turned off the anti wind up mechanism, to see its effect on the system. It is clear then when manual mode is activated, the error SP-PV keeps getting integrated by MVI resulting in an increasing value. When the PID is in automatic mode again, the integral action still has a very high value, resulting in a big overshoot. The system takes a longer time to reach steady state compared to the correct implementation of an anti wind up mechanism.



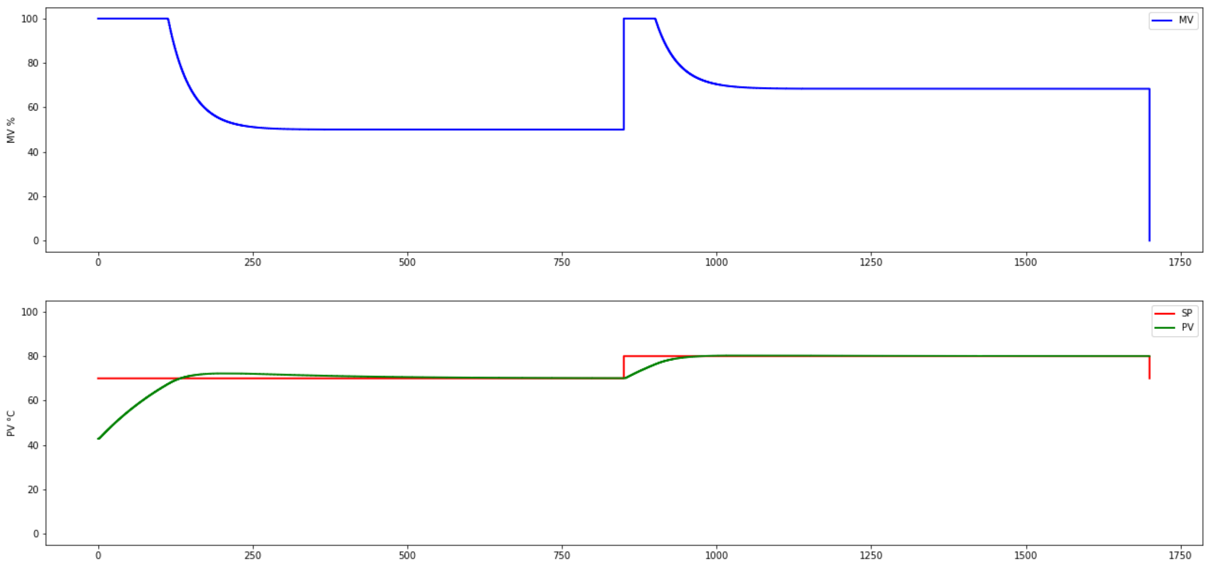
This graph shows an example of saturation. To achieve perfect control MV should be able to reach infinitely large values. This is clearly not possible, for physical reasons. In our case, TCLab MV can go from 0% to a maximum of 100%. This case shows that when MV should have a higher value than 100% the system automatically detects saturation and assigns to MV the maximum value possible: 100%.



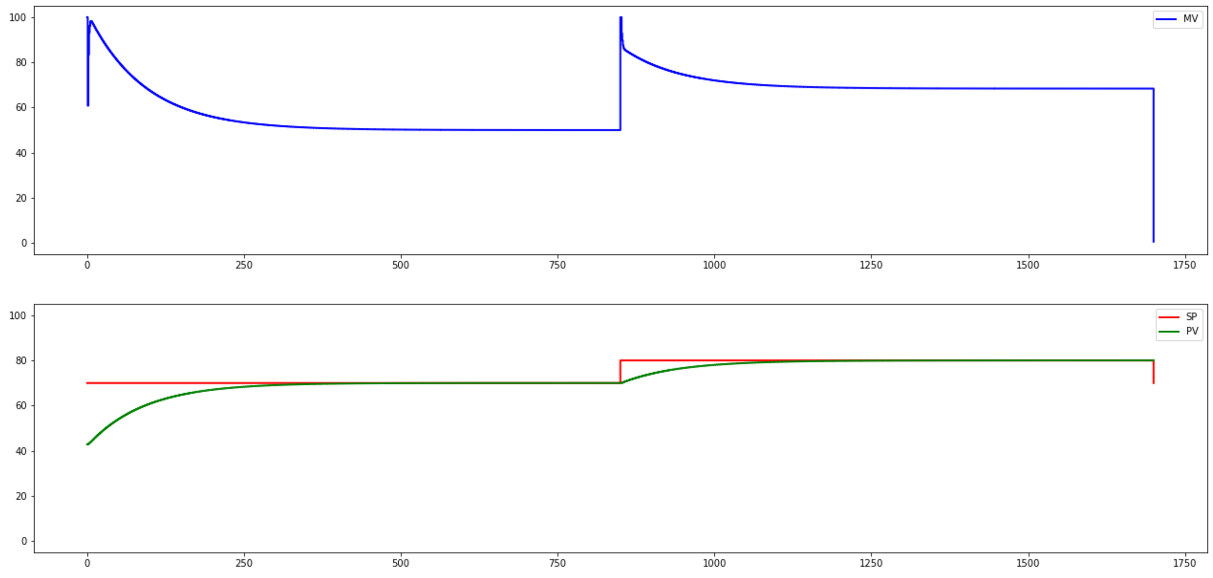
This is the response of the PID to a step input. The three actions are shown in the graph.

Closed loop response to a set point change

The first thing we tried was a closed loop response to a set point change. We tried a simulation of 1700 seconds where SP makes a first step from 0 to PV0(70°C). After 850 seconds the system will be stable in the setpoint and SP will make another step to PV0 + 10. After another 850 seconds the system is stable again.



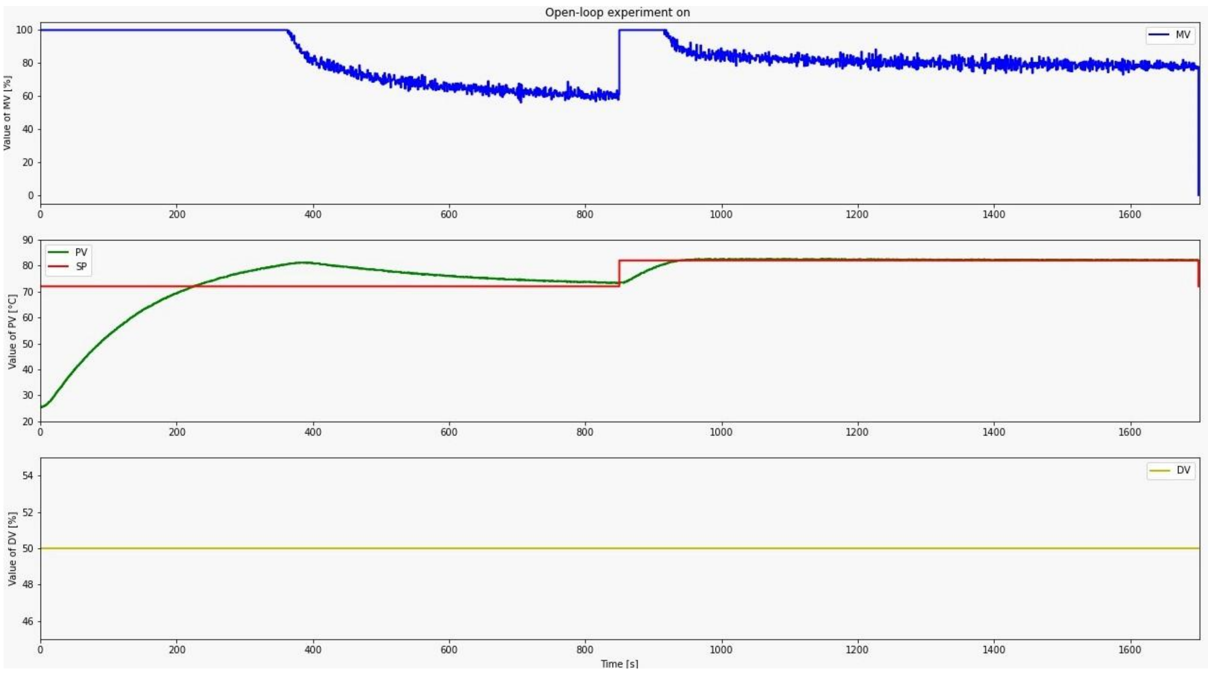
(simulation of set point change in closed loop gamma = 0.2)
The PID appears to behave correctly, with an acceptable rise time and a small overshoot. The MV also behaves as expected and respects the upper and lower bound imposed by the TCLab.



(simulation of set point change in closed loop gamma = 0.5)
With a higher gamma the controller is a lot less aggressive. In this case there is no overshoot and the system reaches steady state faster. This is obvious if we look at how the parameters for the controller are calculated. If we take Kc for example in a first order with delay for a PID:

$$Kc = \frac{T_1 + \frac{\theta}{2}}{\gamma \cdot T_{olp} + \frac{\theta}{2}} \cdot \frac{1}{K_p}$$

The more gamma increases, the more Kc will decrease.

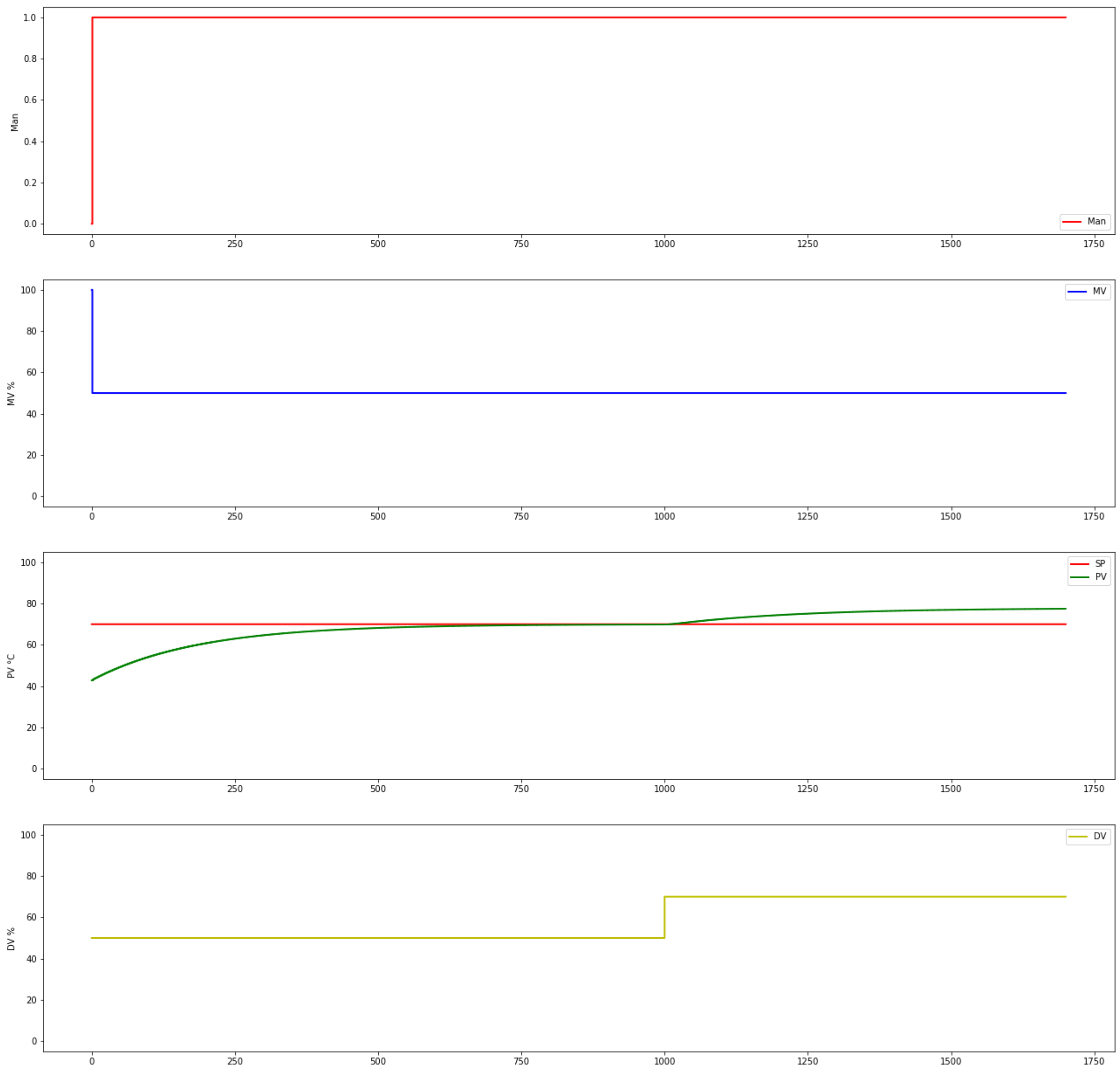


(experimentation on TCLab gamma = 0.2)

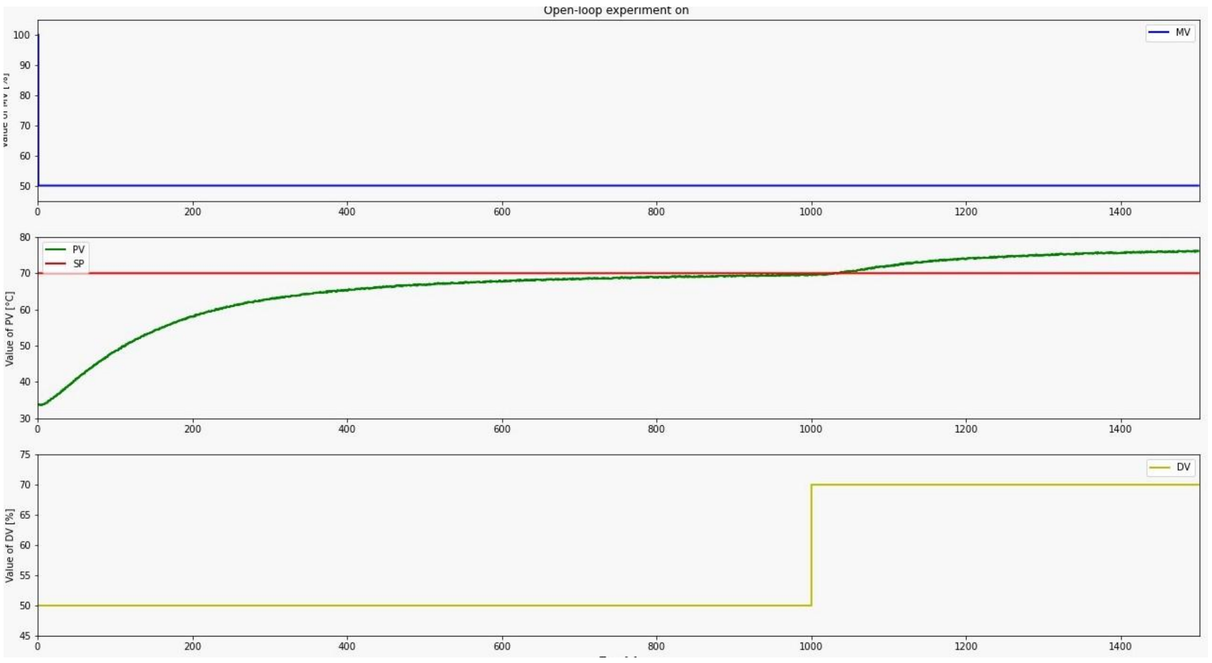
The experimentation resembles the simulations very much. The overshoot is more visible. This is because of the cap that we put on top of the TCLab, where it takes a longer time to cool down. We found the compromise acceptable as it allows us to minimize the noise. You can see on the MV graph that noise is visible but the trace of the curve is very evident and is very similar to the one in the simulation.

Response to DV : No FF and controller in manual mode

In this case, we will leave the PID in manual mode, with MV set to 50%. We will let the system stabilize and then we will apply a change in DV and observe its impact on the system.



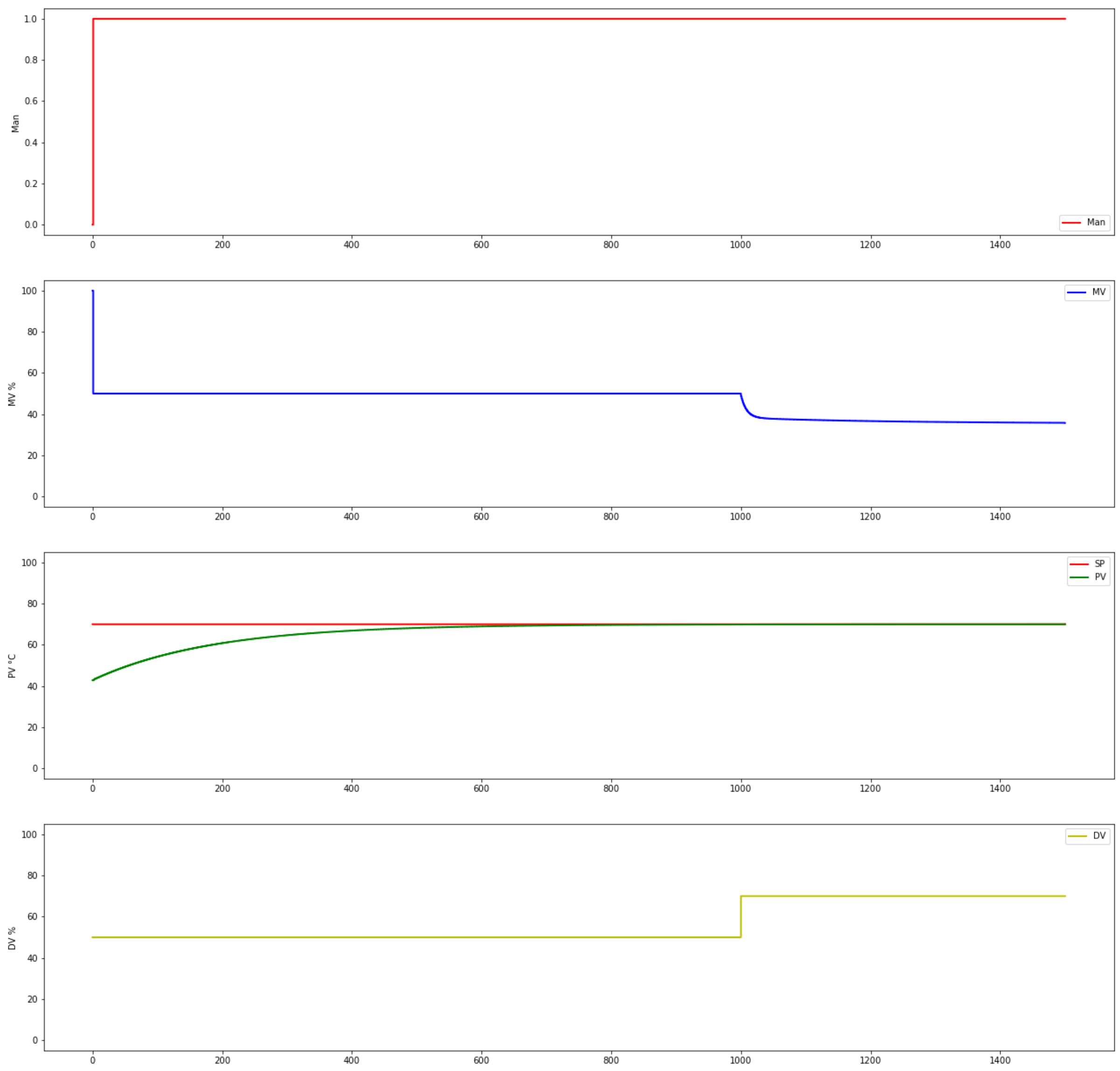
The system behaves correctly. It stabilizes around the set point and then increases slowly as the disturbance is applied.



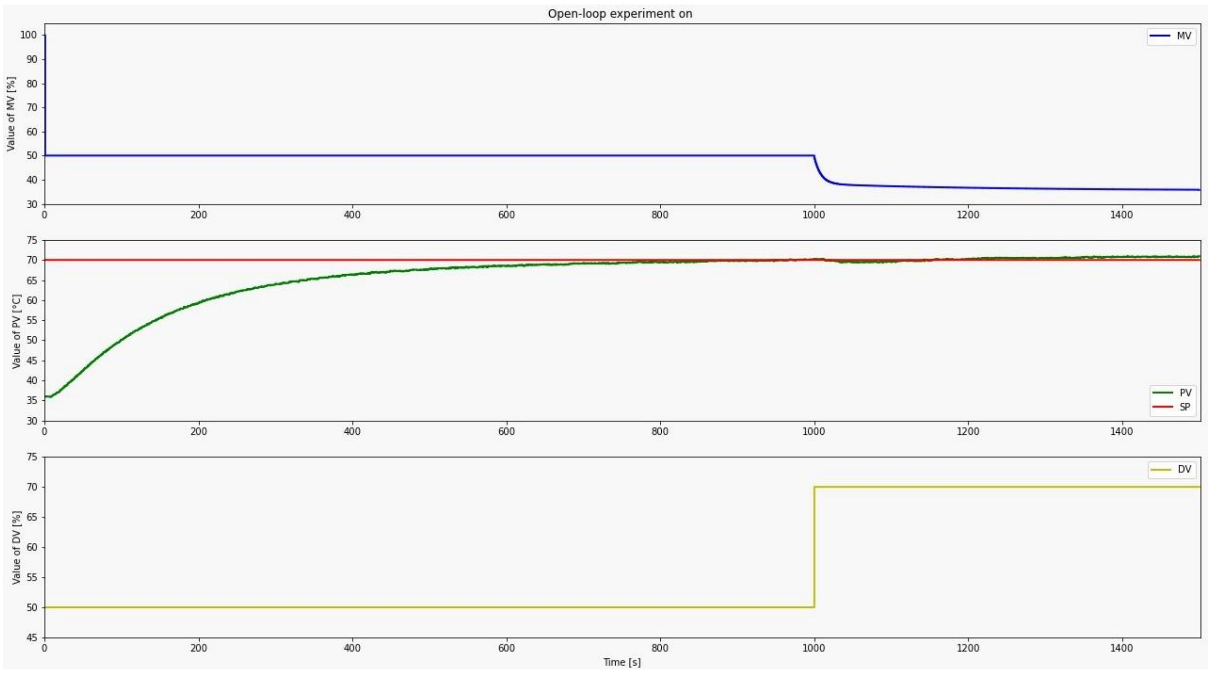
Also the experiment on the TCLab gives similar results. The PV reaches a temperature of 70°C which is the working point when MV = DV = 50%. After the step on DV the temperature starts rising again.

Response to DV : FF and controller in manual mode

The third experiment is with the PID still in manual mode but with the feedforward activated. The feedforward will detect a change in DV and immediately react to fix MV and compensate for the disturbance. In this case the PID will be in manual mode with MV fixed to 50%.



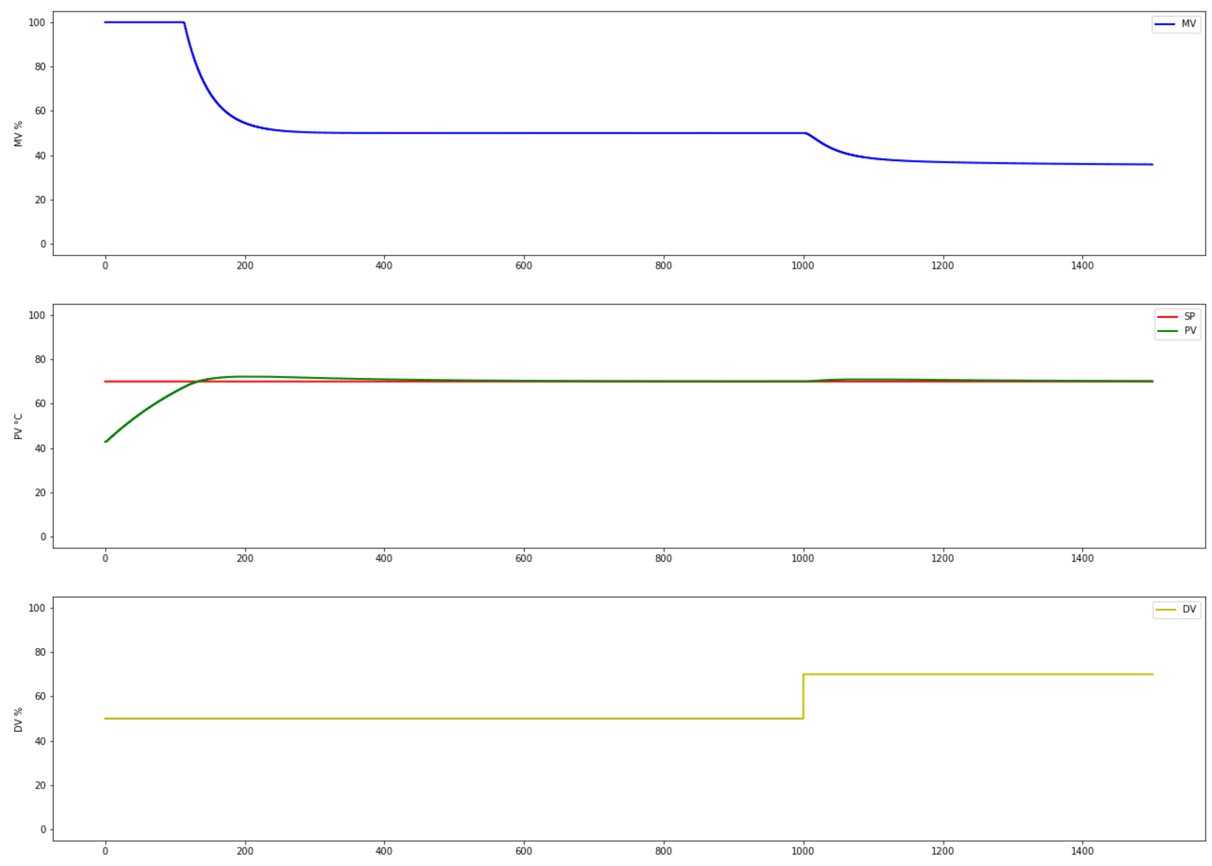
The results of the simulation show that the feedforward is working correctly. As soon as the DV changes, the feedforward takes action to compensate MV. The change in PV is so little that it is not noticeable, which means the feedforward is implemented correctly.



The experiment on the TCLab gives very similar results. When the disturbance hits, the feedforward correctly takes action and modifies the MV. The PV doesn't change, which means that the feedforward is correctly implemented.

Response to DV : No FF and controller in automatic mode

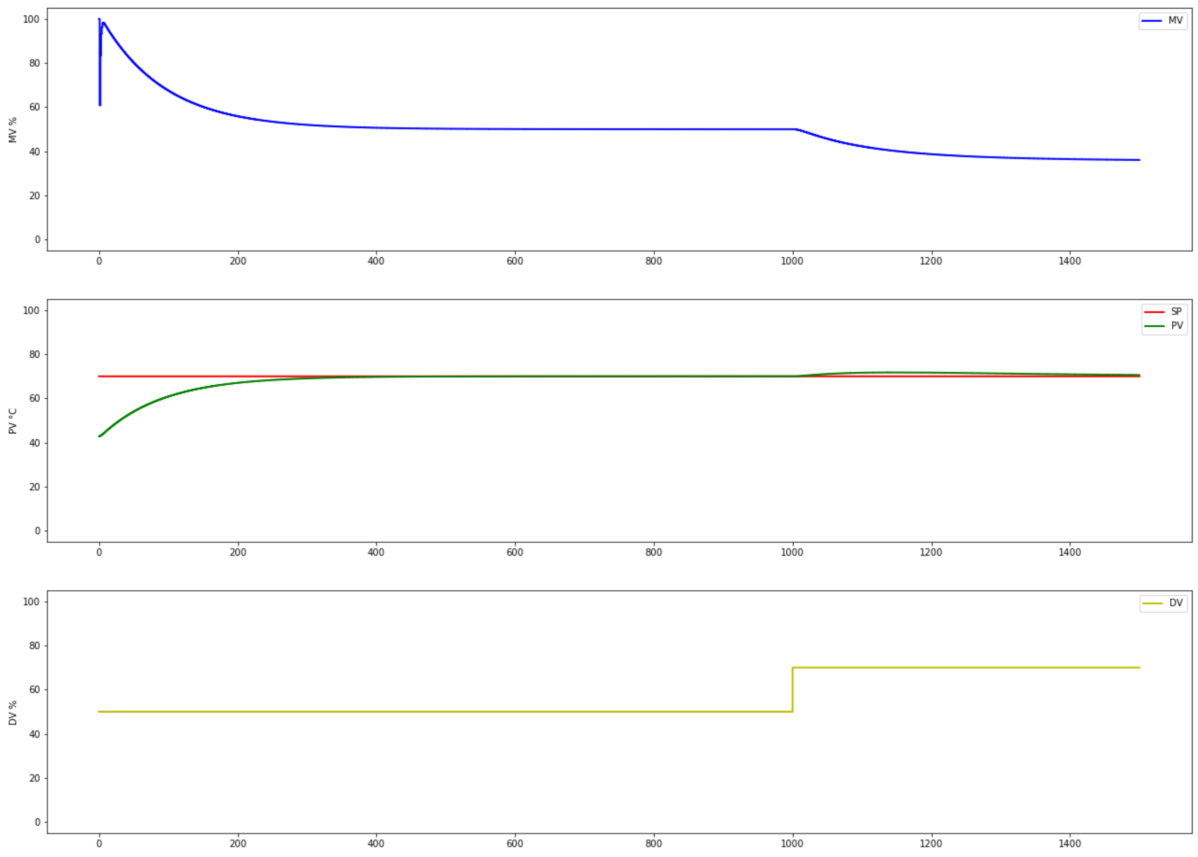
In this experiment we deactivated the feedforward and let the PID take action on the process.



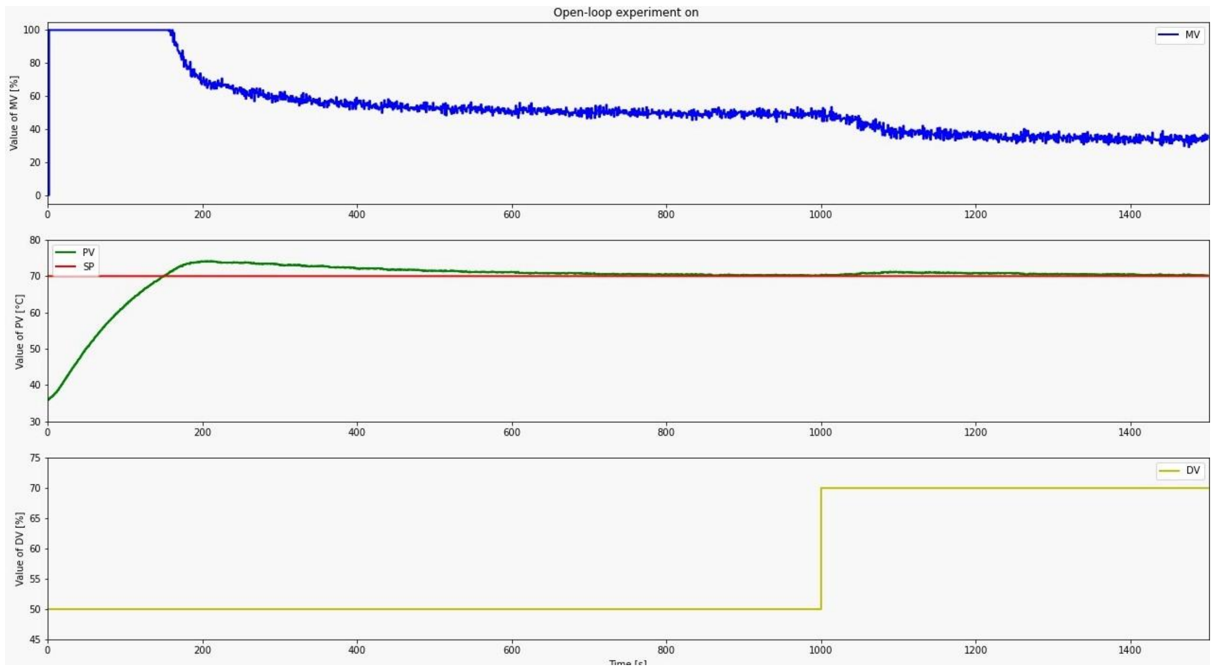
(Automatic mode no FF gamma = 0.2)

Also in this case the PID behaves correctly and takes action to correct the disturbance. We can also see that the time of response is a bit slower than the feedforward. At 1000 seconds the little deviation from SP is clearly visible. The time to steady state after the disturbance is however acceptable.

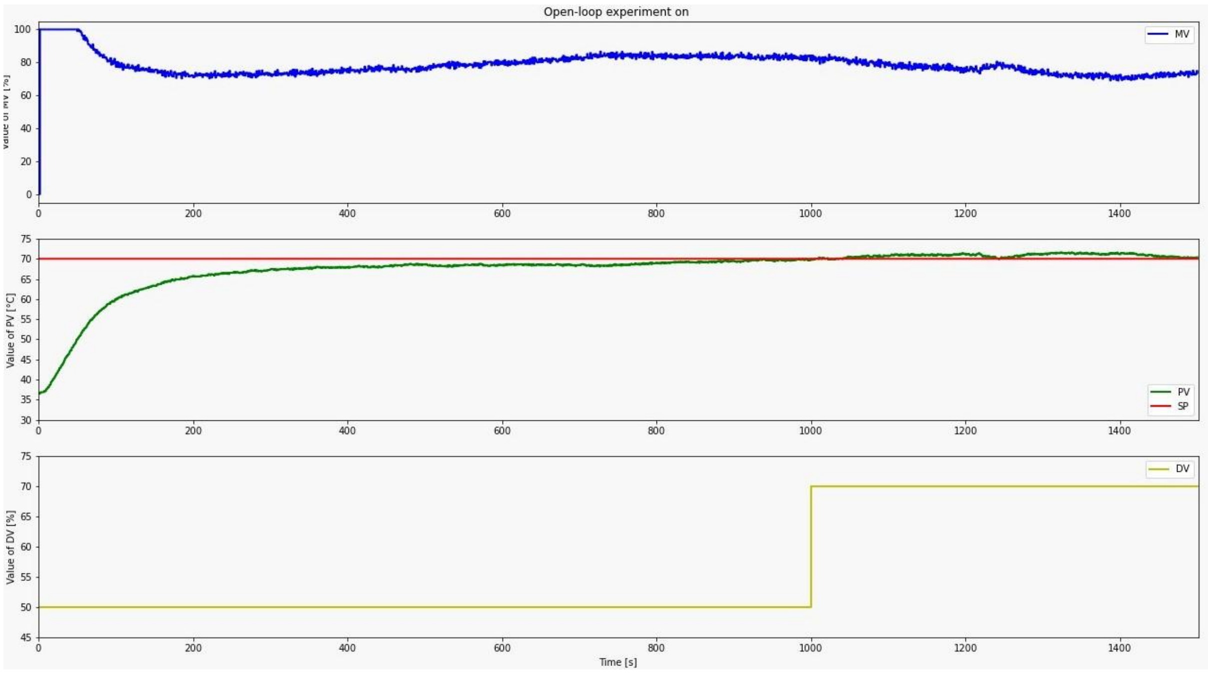
Also in this case we tried to change gamma, and as expected, the more we increased it, the less aggressive the PID became.



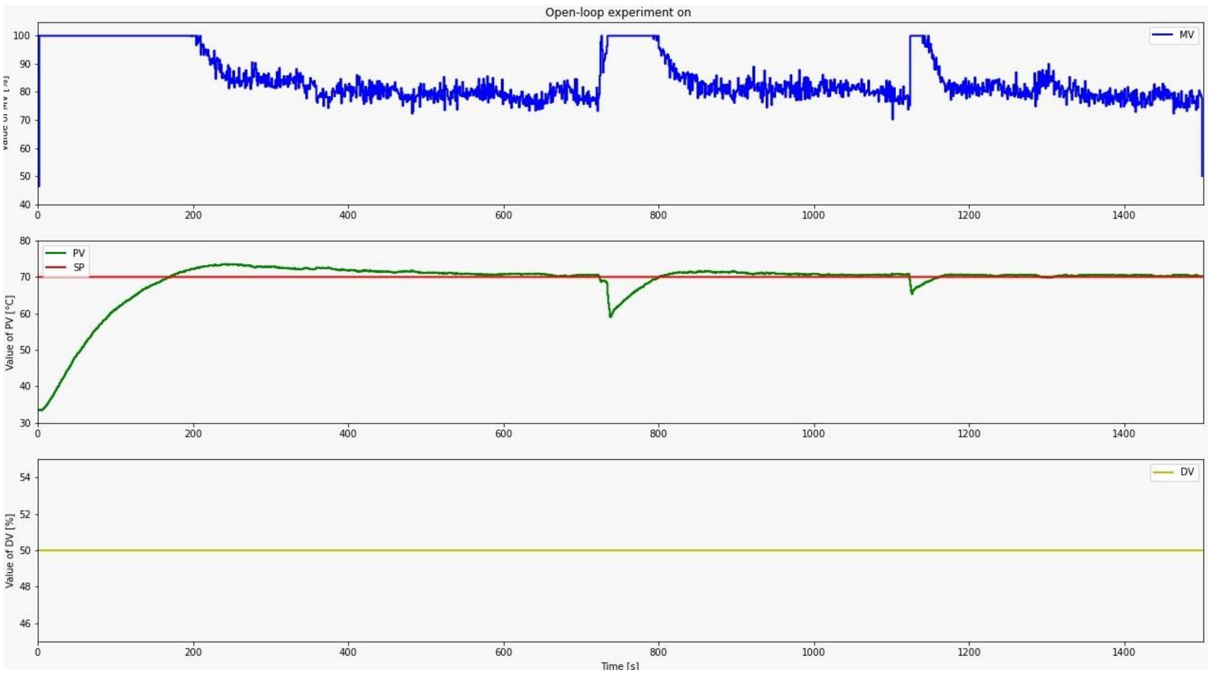
(Automatic mode no FF gamma = 0.5)
It is very clear in the MV graph that the action taken from the PID is a lot less aggressive. This results in PV taking a longer time to reach SP after the disturbance.



The experiment on the TCLab follows the prediction indicated by the simulation. The response to the disturbance is a bit slower than the feedforward but still very acceptable.



In this case we did the same experiment with a gamma = 0.5. Once again we can clearly see that with a higher value of gamma the PID is less aggressive. In this case there is no overshoot. The cost is a slower time of response. Depending on the type of control we want to apply, a less aggressive controller could be the best solution.



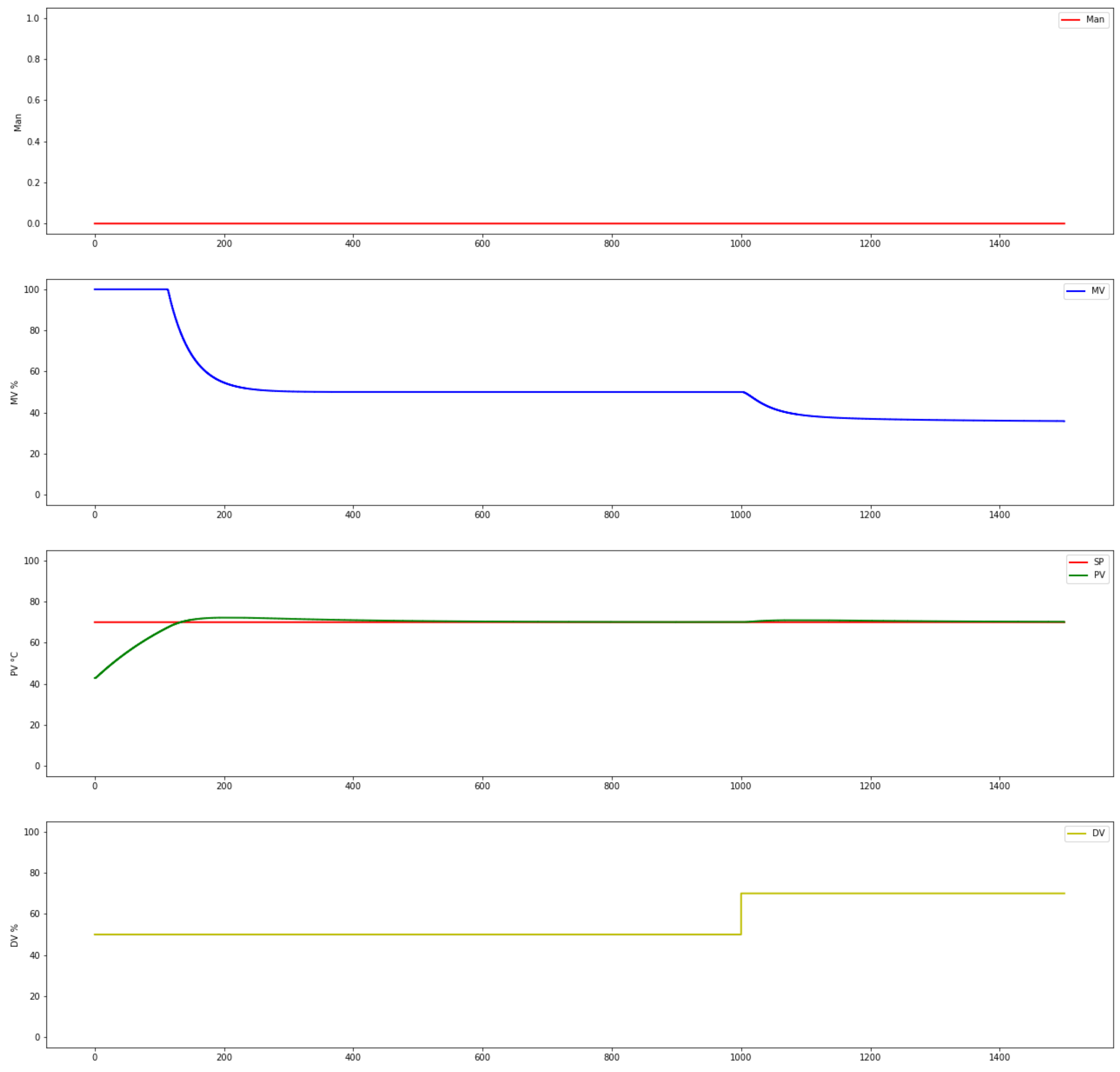
In this last experiment of the section we analyzed the response of the PID to a disturbance coming from an external source. At 750 seconds and 1100 seconds we blew some air directly in the temperature sensor and observed the results. We performed this experiment without the protective cap on the TCLab. Two things can be observed:

1- The noise on MV is a lot more visible than the other times. This is because, without the cap, the sensor is exposed to the room environment and the temperature is likely to have some unexpected drops and rises. This is why, instant by instant, the change on MV needs to be more strong, to compensate for these temperature changes.

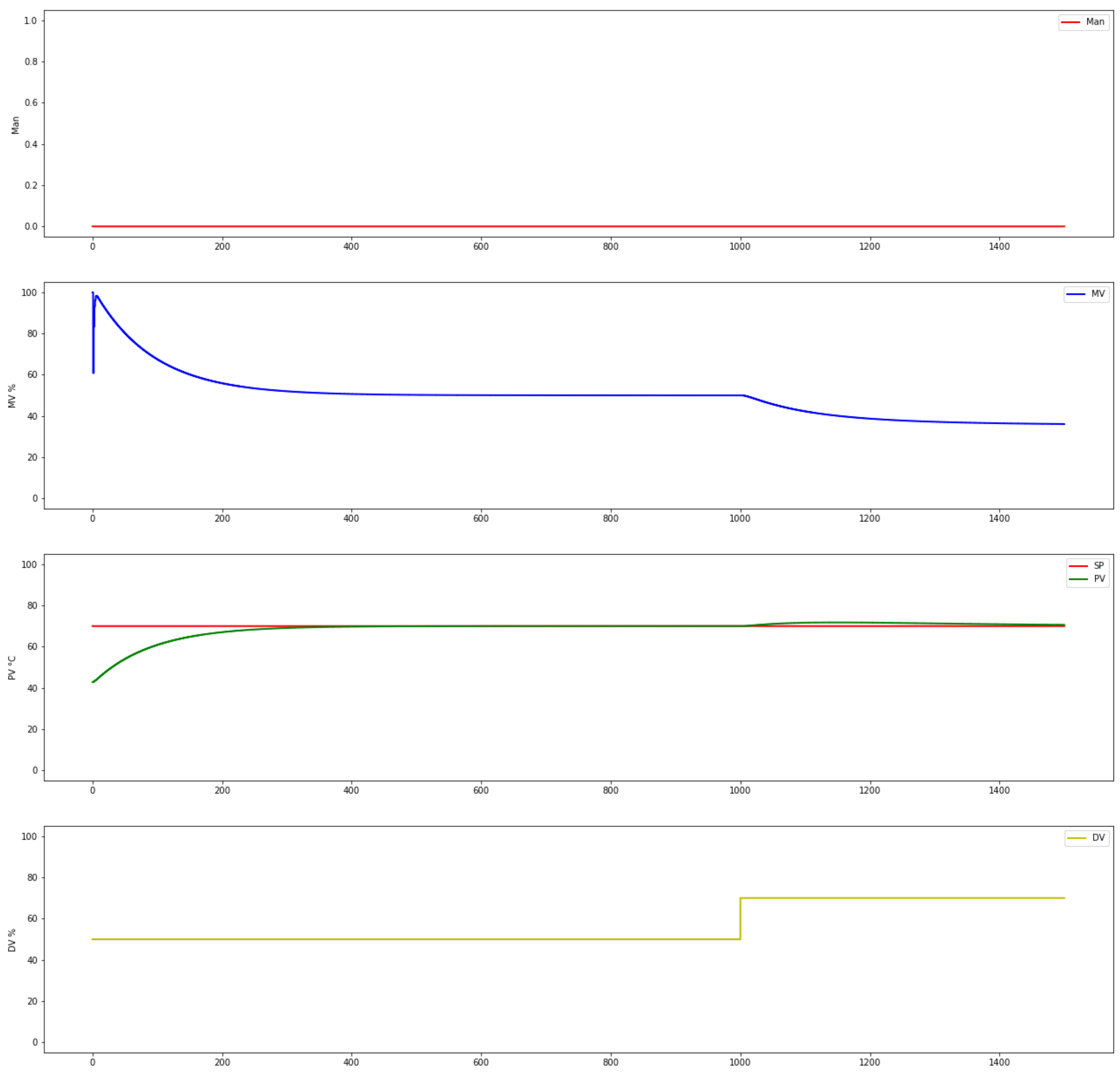
2- The PID responds correctly to external disturbance. As soon as the feedback detects the temperature drop (caused by the blow) it takes immediate action to correct it, and raises MV to 100%. In both cases the return to SP is fast, with some overshoot. The same thing applies here, we could increase gamma to eliminate overshoot at the cost of a slower response time.

Response to DV : FF and controller in automatic mode

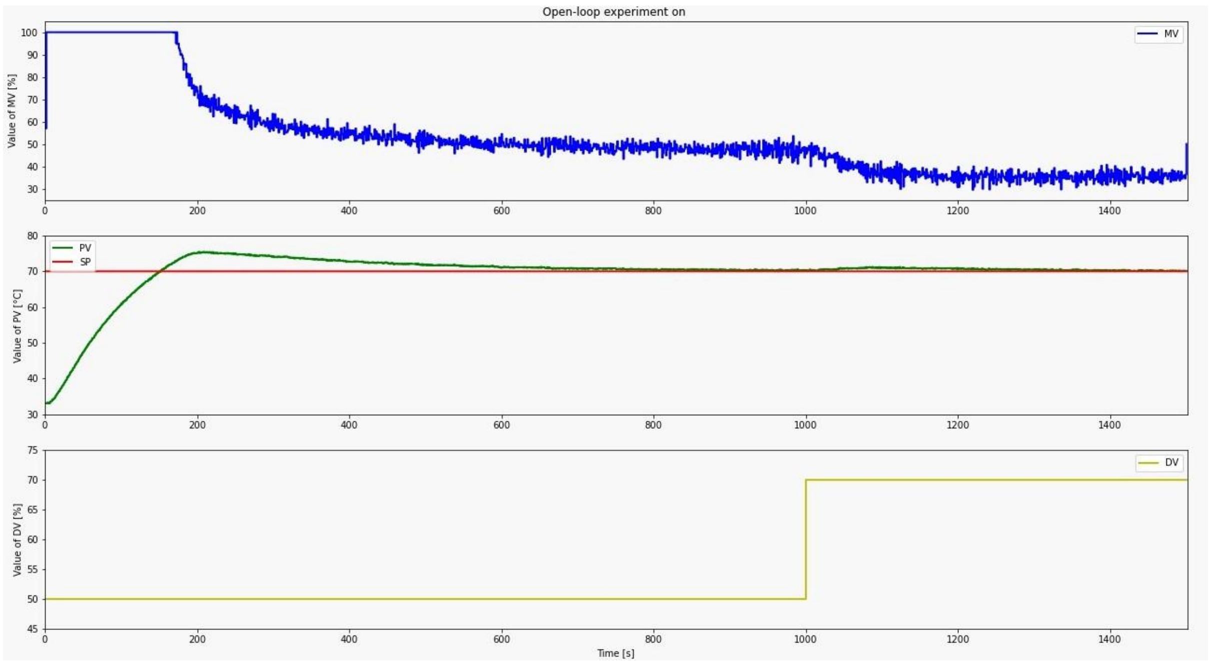
The last experiment consists of a disturbance with PID in automatic mode and the feedforward activated.



(PID automatic FF gamma = 0.2)
The system performed better than the previous experiment but slightly worse than the PID in manual mode with feedforward activated. It is however preferable to activate both the PID in automatic mode and the feedforward mechanism. Feedforward allows for a very fast and accurate response to disturbance. If the disturbance is not measured or it comes from an unexpected source the feedforward won’t be able to correct it. This is why having a PID activated at the same time is a good practice.



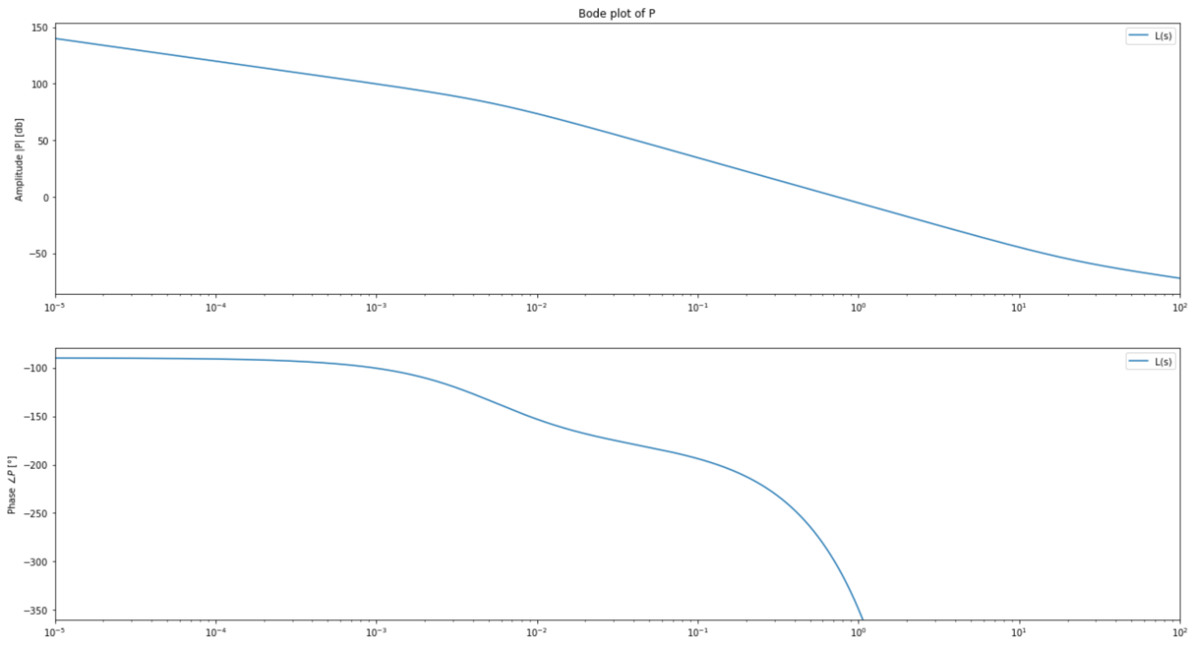
(PID automatic FF gamma = 0.5)
Again, we can see that an increased value in gamma corresponds to no overshoot at the cost of a slower response time.

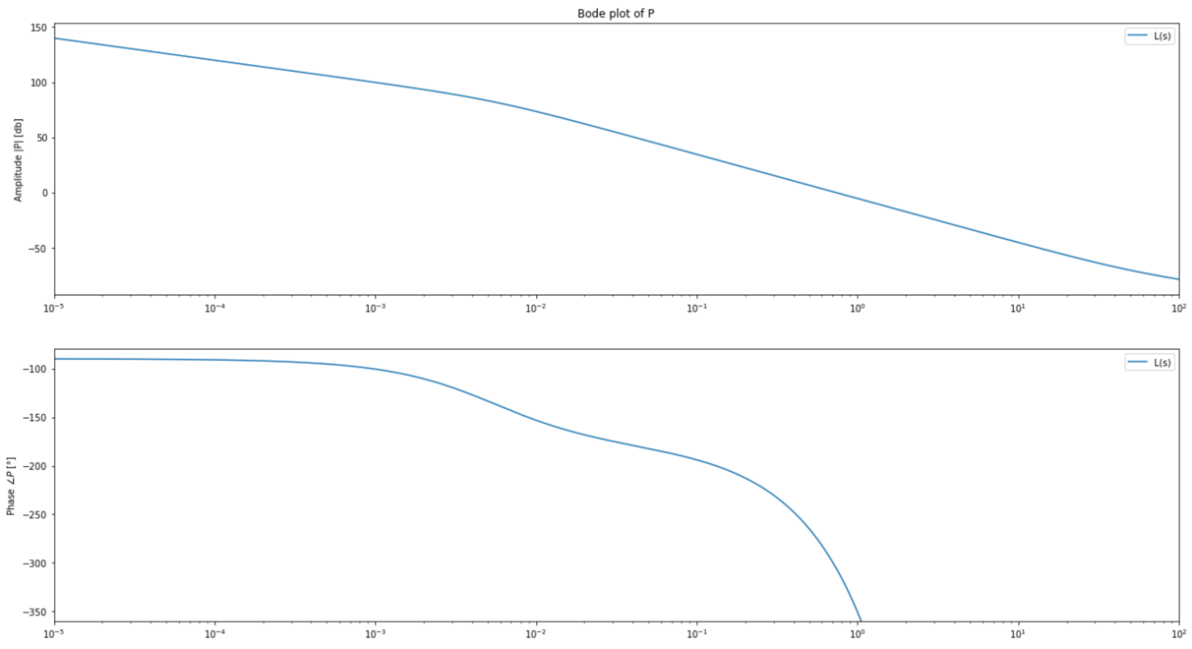


The experiment on the TCLab follows the predictions of the simulations.

Margins

We then calculated the bode plot for the two loop functions (L(s) = P(s) * C(s)) used in our testing (one with gamma=0.2 and the other one with gamma=0.5). We used the data from the bode plot two calculate the stability margins of the function. With our results we can say that between the two functions there isn’t really a stability difference and they both are quite stable since we can still add a gain of about -50 dB or a delay of about -125° without making our system unstable.





in order: gamma=0.2 and gamma=0.5