Robot Control Exercise 5: Model Predictive Control

Nicholas Shindler, shindler25@gmail.com

December 10, 2019

1 Water Tank Model

yes, they appear to work consistently.

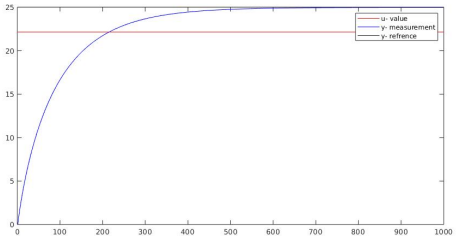

Figure 1: Test $y^r = 25$ starting from $y = 0$

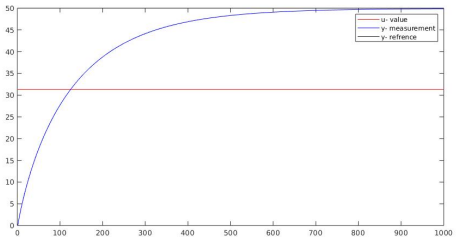

Figure 2: Test $y^r = 50$ starting from $y = 0$

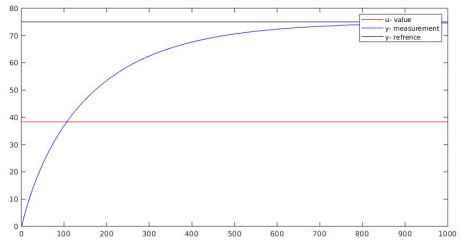

Figure 3: Test $y^r = 75$ starting from $y = 0$

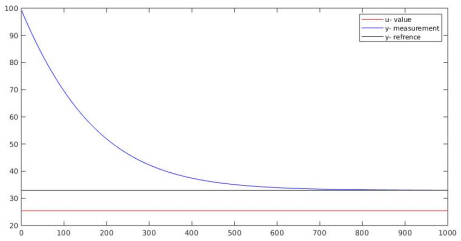

Figure 4: Test $y^r = 33$ starting from $y = 99$

Setting y equal to y^r results in no change.

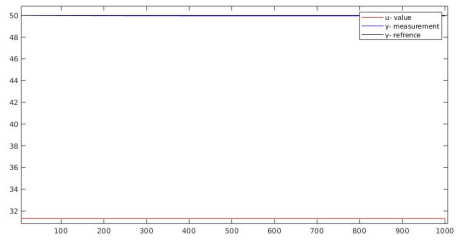

Figure 5: Test $y^r = 50$ starting from $y = 50$

2 The MPC Formulation

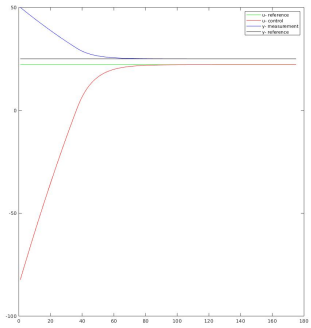

Figure 6: Test $y^r = 25$ starting from $y = 50$

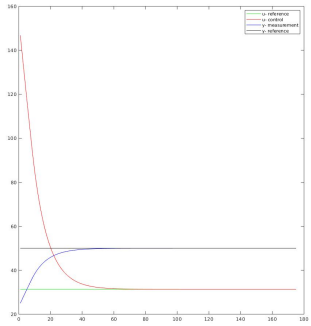

Figure 7: Test $y^r = 50$ starting from $y = 25$

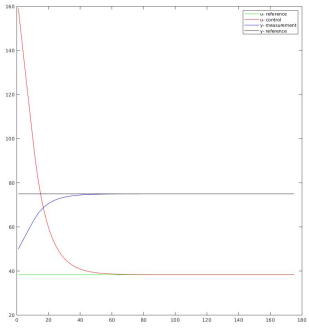

Figure 8: Test $y^r = 75$ starting from $y = 50$

Setting y equal to y^r results in no change.

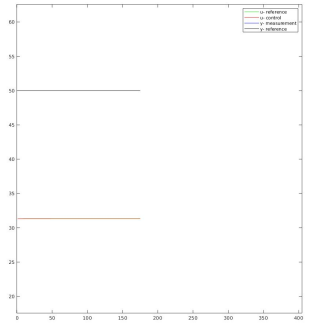

Figure 9: Test $y^r = 50$ starting from $y = 50$

3 MPC reference trajectory

3.1 custom trajectory

creating a trajectory based on a $f(x) = (1 + \exp^{-x})^{-\alpha}$ curve

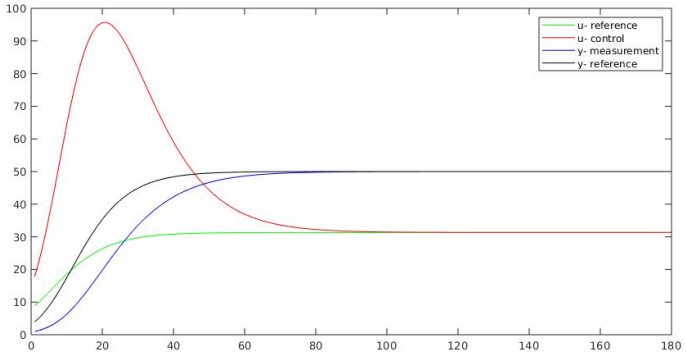

Figure 10: Test $y^r = 50$ starting from $y = 0$

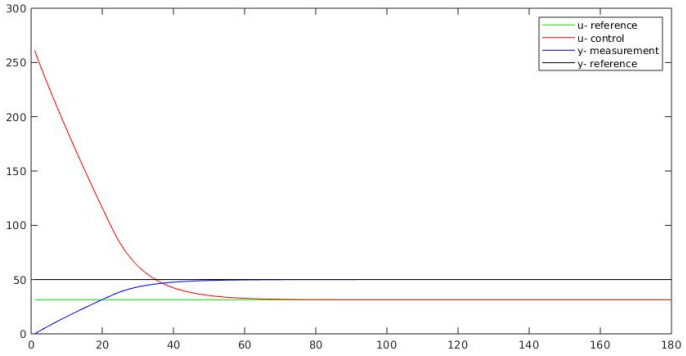

Figure 11: Test $y^r = 50$ starting from $y = 0$

This trajectory allows a much smoother curve, preventing the very high initial control values that are unnecessary as real systems are limited by physical properties and control values over a certain value are meaningless. It performed very well for what I was expecting, given how the example sin function worked.

3.2 On - Off trajectory

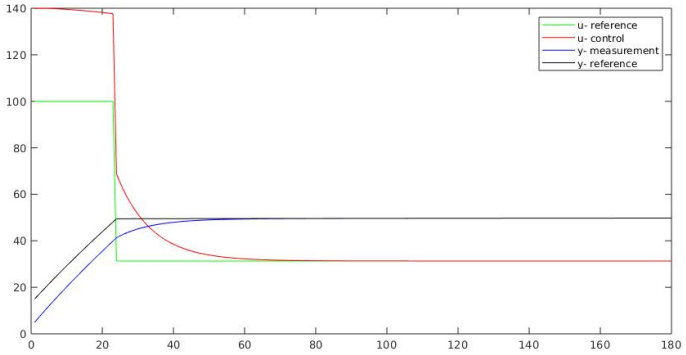

Figure 12: Test $y^r = 50$ starting from $y = 5$

I modified this so that at the reference point the the control would be steady state rather than 0 to prevent oscillation. I also set the reference to start from $y = 15$ rather than $y = 5$ so that the reference and actual would perform slightly differently.

3.3 PID trajectory

This seems like a good solution when you do not know much about the system. however due to the control values it performed very similar to pure on off + steady state. Given that the on off with steady state is largely optimal as a step function this a good result.

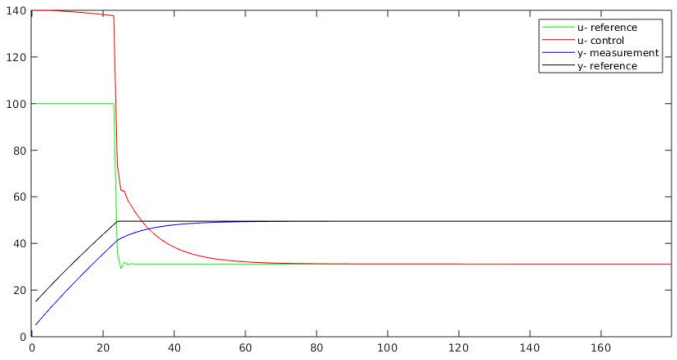

Figure 13: Test $y^r = 50$ starting from $y = 5$

I set the reference to start from $y = 15$ rather than $y = 5$ so that the reference and actual would perform slightly differently.

4 Tune the controller

I changed A from 0.005 to 0.008, a from 0.001 to 0.0021 and γ from 0.001 to 0.01. additionally q is scaled by 6 and r is scaled by 0.3. I then used the trajectory function that I created for the controller.

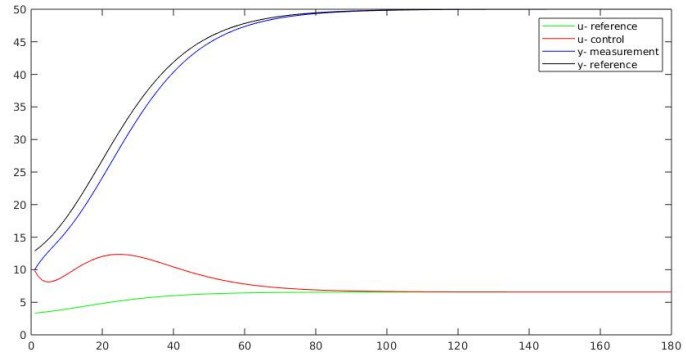

Figure 14: Test $y^r = 50$ starting from $y = 10$

I further updated A from 0.005 to 0.0005, a from 0.001 to 0.0001 and γ from 0.001 to 3 in the controller (with the model containing values as previously stated. additionally q is scaled by 42 and r is scaled by 0.7.

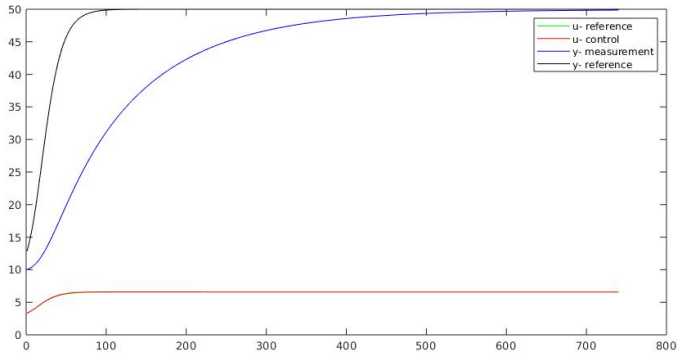

Figure 15: Test $y^r = 50$ starting from $y = 10$

with these changes convergence to the set point was much slower.

I then retesting with the pid controller, which resulted in a much faster gain, though with much more noise.

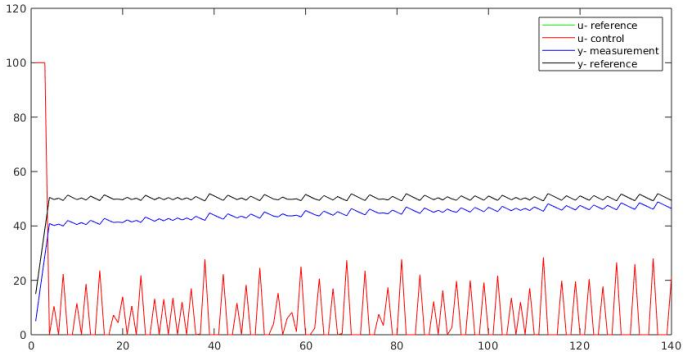

Figure 16: Test $y^r = 50$ starting from $y = 10$

It seemed that the scale values worked well when there was a small error, however at large enough discrepancy between model and controller the tuning seemed

to break down.

5 Constrain the controller

using the initial modification case I constrained to control from 0 to 30 and as a result:

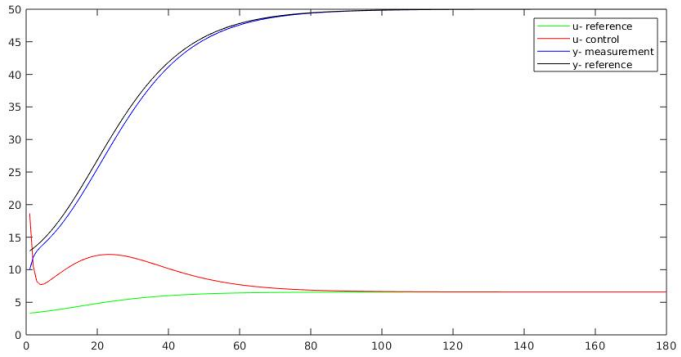

Figure 17: Test $y^r = 50$ starting from $y = 10$

I tested again with the steady state controller:

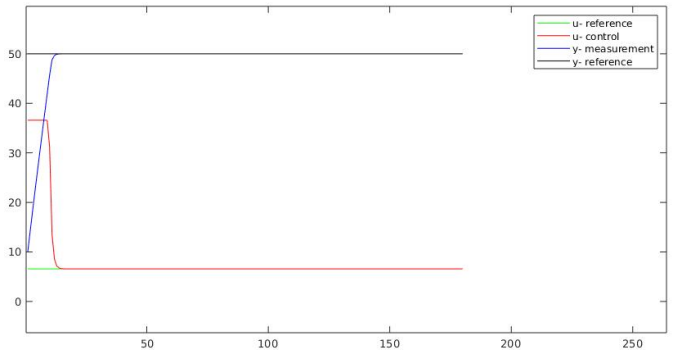

Figure 18: Test $y^r = 50$ starting from $y = 10$

You do not see the impact of the constraints in the first case as the control is not naturally exceeding 30 however it is present in the second graph that without the limit would have a control value of about 200 to start with.