Third optional course project for nonlinear systems and control

Riccardo Ghetti

June 9, 2025

1 Task 1

We have that our objective function and constraints are:

$$
\begin{aligned}
\min_{x,u} \quad & J(x,u) = \sum_{i=0}^{N-1} \left\{ w_1 [x_2(i) - x_r]^2 + w_2 [u(i+1) - u(i)]^2 \right\} + w_3 (x_2(N) - x_r)^2 \\
\text{s.t.} \quad & x(i+1) = \text{CSTR}(x(i), u(i)) \quad \forall i = 0, \ldots, N-1 \\
& x(0) = x_0 \\
& 0 \leq x_1(i) \leq 1 \quad \forall i = 0, \ldots, N \\
& 0 \leq x_2(i) \leq 0.66 \quad \forall i = 0, \ldots, N \\
& 0.1 \leq u(i) \leq 2 \quad \forall i = 1, \ldots, N
\end{aligned}
$$

2 Task 2

Here we can see the plot of the evolution of the system and the control signals

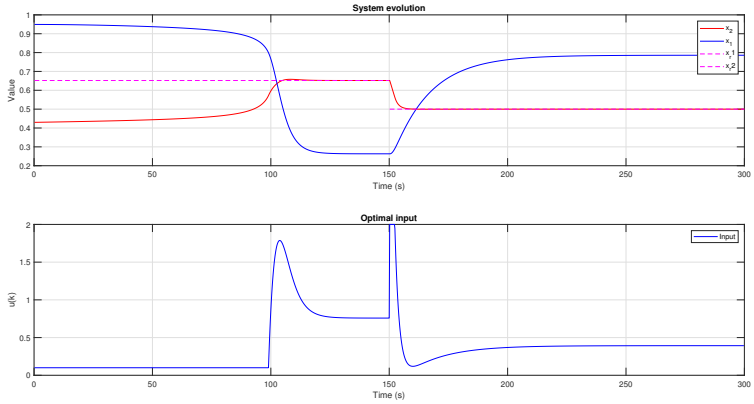

Figure 1: The system response with N=10

We can clearly see the tracking has been achieved.

3 Task 3

Let's plot the input and output together for the different values of N:

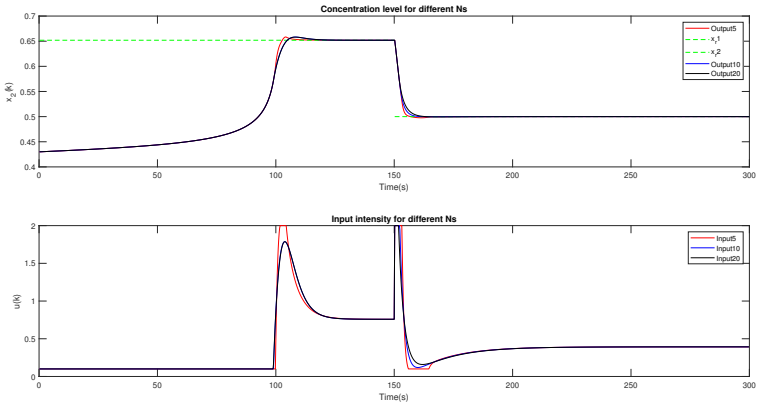

Figure 2: The solutions to the different NMPCs

3.1 Comparison

There is no big difference between the system evolutions, but it seems that for N=5, the controller chooses a more aggresive input, saturating it at around t=100s and rapidly collapses to its minimum in t=150 s. The curves are smoother for bigger horizons penalizing more input variations than tracking. The tracking goal is achieved almost at the same time for all three of them.

3.2 Computational time

For N=5, the average solving time over five runs was 55,32682 seconds, for N=10 it was 138.2936 seconds and for N=20, 829.6113 seconds. Matlab stays most of the time in the optimizer which seems to scale badly with the number of variables even if they increase linearly in N.

4 Task 4

Let's compare the diagrams:

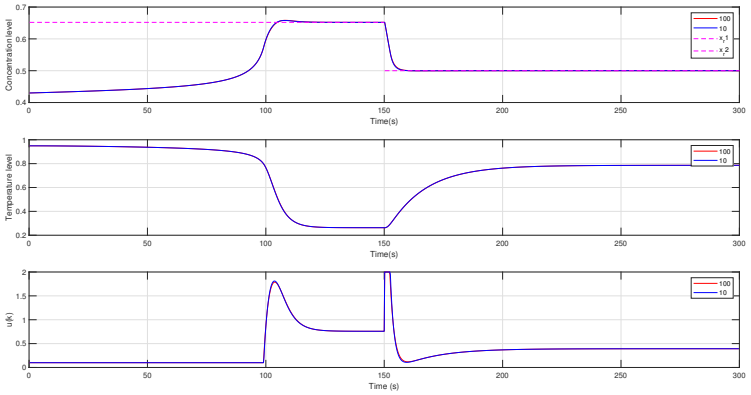

Figure 3: Comparing the states and input

We see almost no difference between the behaviour of the system. It seems that as a terminal weight 10 is enough to force the system to actually track the reference. A notable difference though, is the computational time: for w3=100 it was 113.4339 seconds, but for w3=10 it was more than triple, i.e. 367.1807 seconds. It looks like the previous solution is still optimal, but it takes the solver more time to reach it.

5 Task 5

Comparing the different controllers:

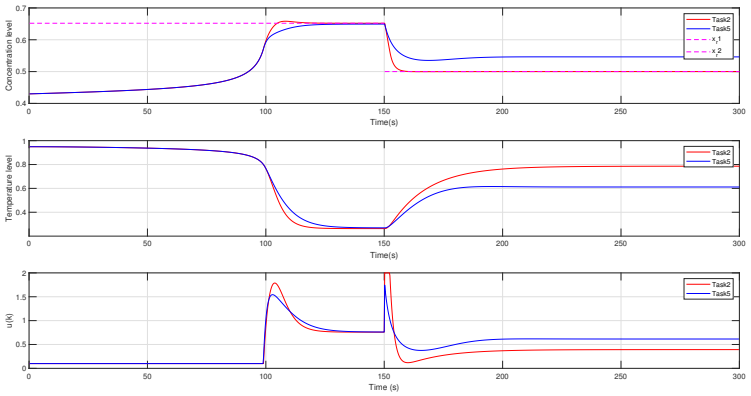

Figure 4: Comparing the different objectives

It seems that the other controller is very similar in the first half of the simulation achieving good tracking, undershooting the reference rather than overshooting it. However when the reference changes, it initially sees more value in picking a lower control action and then actually applies a bigger one which doesn't even achieve tracking. It could be that since we are dealing with absolute values of the input and not variations of it, applying bigger controls weighs a lot more than previously, obstaculating the tracking goal. In fact, from the simulations and the system equations, we can notice that lowering the concentration requires bigger inputs, a fact that could influence a lot the new objective function. Since the first reference is higher and the system naturally evolves to higher concentration levels, the input in the first half doesn't need to be high, allowing the controller to achieve it. However in the second half, after nearing the reference for a bit, the program sees more value in keeping the inputs low rather than doing tracking.

6 Task 6

Here is the comparison of the diagrams with measurement noise when solving the optimization problem:

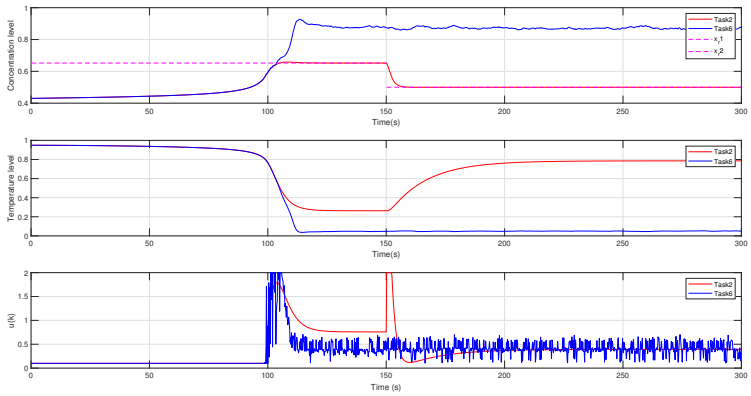

Figure 5: The behaviour of the system with noise

The result is that the controller is completely useless: it keeps oscillating almost every iteration and never achieves tracking. Most important of all, it exceeds the given bounds on concentration! Most of the optimization iterations result infeasible because of that when the system starts to destabilize, resulting in the controller being completely unable to steer the system, particularly after the second reference change. It seems that noise has a very big impact on the controller!
Now we run again the simulation with N=5:

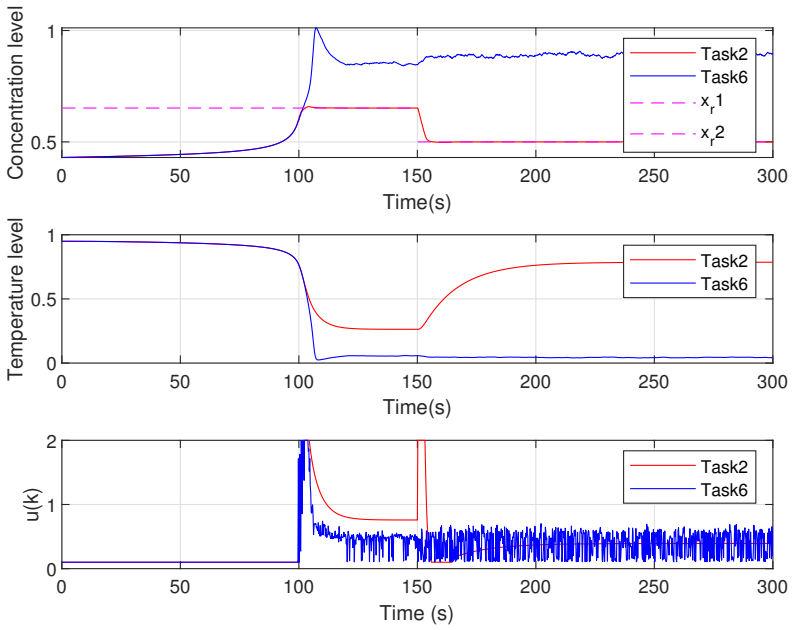

Figure 6: Shortening the horizon

The result is even worse: concentration levels get higher than 1! As for the rest, the behaviour of the controller seems even more erratic and oscillating. It seems that even shortening the horizon doesn't mitigate the mismatch between the noisy predictions and the actual evolution of the system. The problem of this task is that the first reference is very near the constraint $x_2 \leq 0.66$ and even a slight noise can make the problem completely unfeasible causing the optimizer to reach nonsense solutions.