Model Predictive Control
Checkpoint 9
Date: 10 November 2020

This checkpoint is about nonlinear MPC using the Casadi package in Python programming. Casadi allows for simpler coding and provides an effective solver for the nonlinear programming in MPC, provided that we respect its predefined syntax.
The nominal nonlinear system whose states we need to drive to zero using MPC is defined according to the following state-space representation:

$$\Sigma_{nominal}: \begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = (1 - x_1^2) \cdot x_2 - x_1 + u \end{cases} \tag{1}$$

with the constraints described in the course. The given codes were for open-loop trajectory of the mentioned system, where the optimization is performed within the prediction horizon giving us the optimal control and state trajectory.
The results of open-loop MPC on the **nominal** system are:

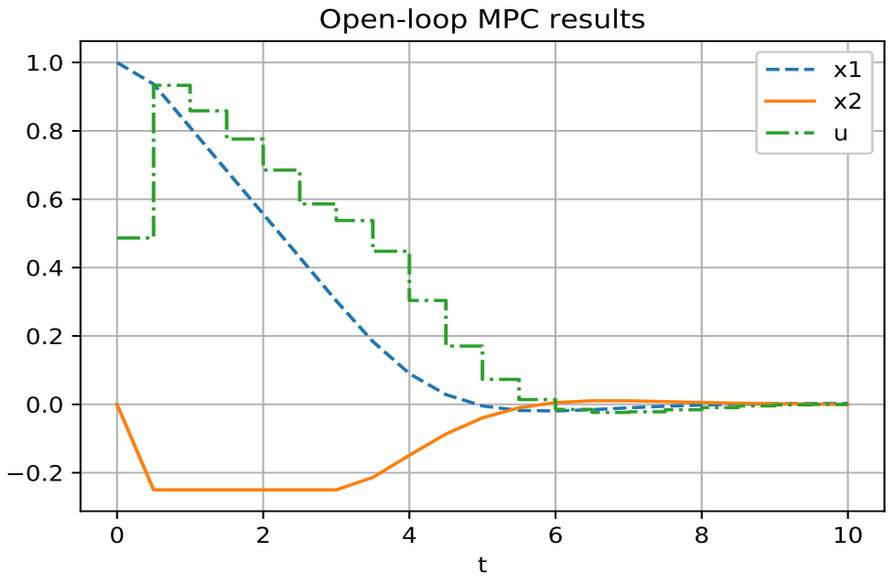

Figure 1: Response of the open-loop system.

Note that with the open-loop system, we cannot simulate its response beyond the prediction horizon `N` = 20 samples. The sampling period is 0.5 second which gives a simulation time of 10 seconds.
Now, we need to close the loop. The following changes are made:

- Introduction of a simulation horizon `Nsim` larger the the prediction horizon `N` (40 samples). This `Nsim` is used in the simulation loop where we get the response of the closed-loop system. Note that the optimization still has to go with `N` because we are optimizing the cost function within the prediction horizon.
- The new state `x_opt` is updated after each iteration and it then becomes one of the inputs of the next iteration, because it appears in the equality constraint of the new optimization problem.
- The control input is now only the first sample of the optimal control signal calculated using nonlinear programming denoted `w_opt[0]`.
- The variable `u_opt` now denotes the real control signal. This means after each time we calculate the new sample of the control input, it is stocked in this variable.
- (Only in the case of time-varying constraints, not this case) We need to implement the new constraints in the loop, before each time we call the optimization problem. The optimization is then performed with the new constraints.

The results of the closed-loop controlled **nominal** system are as follows:

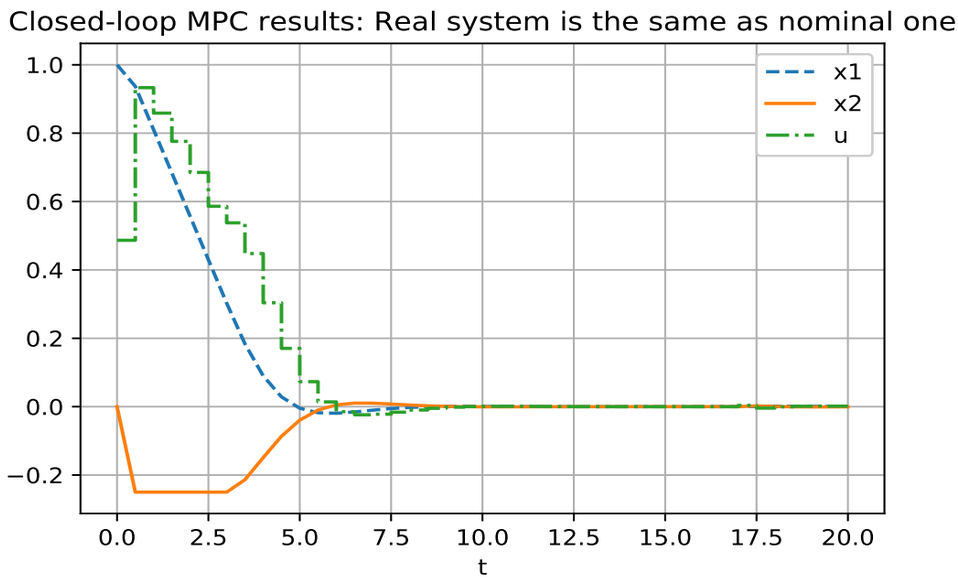

Figure 2: Response of the closed-loop system: Real system is the same as nominal one.

The results are the same as the open-loop one as we have exactly the same system and constraints. Note that now we can simulate the controlled system beyond the prediction horizon `N`, up to the simulation horizon `Nsim`. We are able to stabilize the system and drive the states to 0 while respecting all constraints.
Now, I would like to experiment with the case where the real system is to some extent different from the nominal one, so as to represent the system modelling uncertainty:

$$\Sigma_{real} : \begin{cases} \dot{x}_1 = 0.9 \cdot x_2 \\ \dot{x}_2 = (1 - x_1^2) \cdot x_2 - 1.1 \cdot x_1 + u \end{cases} \tag{2}$$

Note that the optimization problem is done with the nominal system (line 48 in my code) while the response simulation is done with the real system (line 72).
The results of the closed-loop controlled **real** system are as follows:

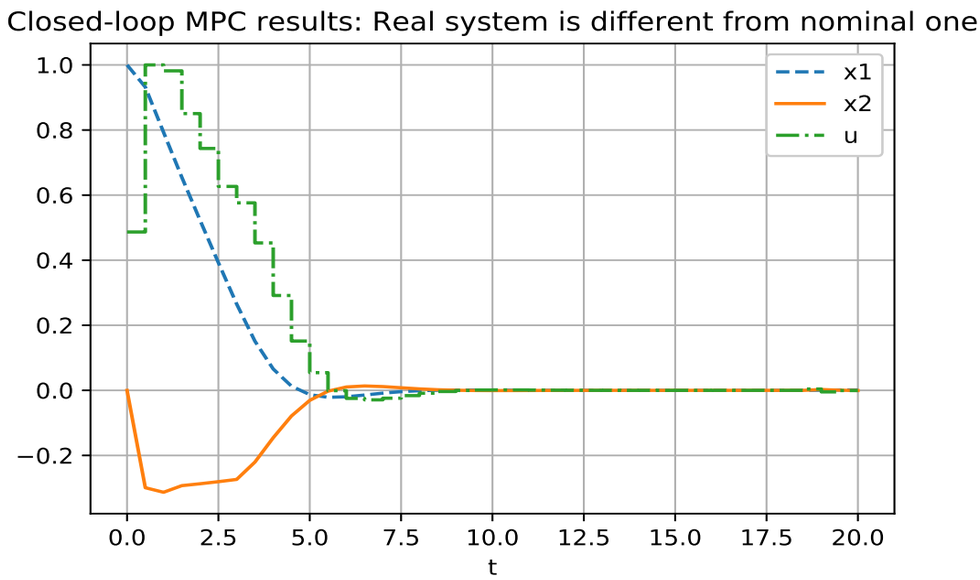

Figure 3: Response of the closed-loop system: Real system is different from nominal one.

We see that the response now is different from that of the nominal system, but the system is still stabilized. Our MPC strategy is robust enough under this uncertainty.
Through this checkpoint, I understood how to use the Casadi package for nonlinear MPC, which simplifies the codes and economize resources better compared to traditional coding.