Model Predictive Control for Renewable Energy Systems
University of Freiburg – Summer semester 2023

Exercise 5: Nonlinear Model Predictive Control

Dr. Lilli Frison, Jochem De Schutter, Prof. Dr. Moritz Diehl

Let us consider again the inverted pendulum with nonlinear dynamics

$$\dot{x} = f(x,u) = \begin{bmatrix} x_2 \\ \sin x_1 - 0.1 x_2 + u \cos x_2 \end{bmatrix} . \tag{1}$$

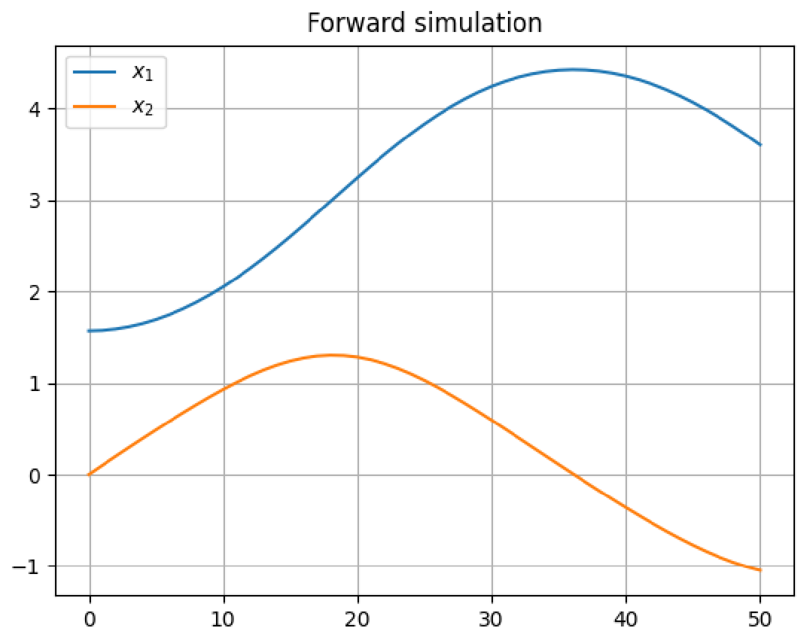
with state vector $x = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^\top = \begin{bmatrix} \theta & \dot{\theta} \end{bmatrix}^\top \in \mathbb{R}^2$ and $u \in \mathbb{R}$. The variables $\theta, \dot{\theta}$ represent the angle deviation and speed w.r.t. the top position, while the control variable u is a horizontal force applied at the tip of the pendulum.
The goal of this exercise is to develop a nonlinear MPC controller that is able to swing up the pendulum from a stable downward position to an upright position and zero angular speed.

1. **Numerical integration.** First, we need to implement and discretize the system dynamics using numerical integration methods.

   (a) Implement a simulator for the system dynamics with an explicit Runge-Kutta 4 integrator with a sampling time of $T_s = 0.1$ s and with 10 intermediate RK4 steps in one sampling interval. You can use the `CasADi integrator` class.

   (b) Simulate the system forward from the initial state for $u(t) = 0$ and $x_0 = (\pi/2, 0)$ and plot the state evolution for a time horizon of $T = 5$ s.

   The forward simulation results in a damped swinging motion around $\bar{\theta} = \pi$:



2. **Optimal control formulations.** Consider the MPC scheme based on the discrete-time optimal control problem

$$\begin{aligned} \underset{\substack{x_0,\dots,x_N \\ u_0,\dots,u_{N-1}}}{\text{minimize}} \quad & 10 x_N^\top x_N + \frac{1}{2}\sum_{k=0}^{N-1} x_k^\top x_k + 2u_k^2 \\ \text{subject to} \quad & x_0 = \hat{x}_0, \\ & x_{k+1} = F(x_k, u_k), \quad k = 0, \dots, N-1 \end{aligned} \tag{2}$$

   with $N = 50$ and $\hat{x}_0 = (\pi, 0)$ as an initial state.

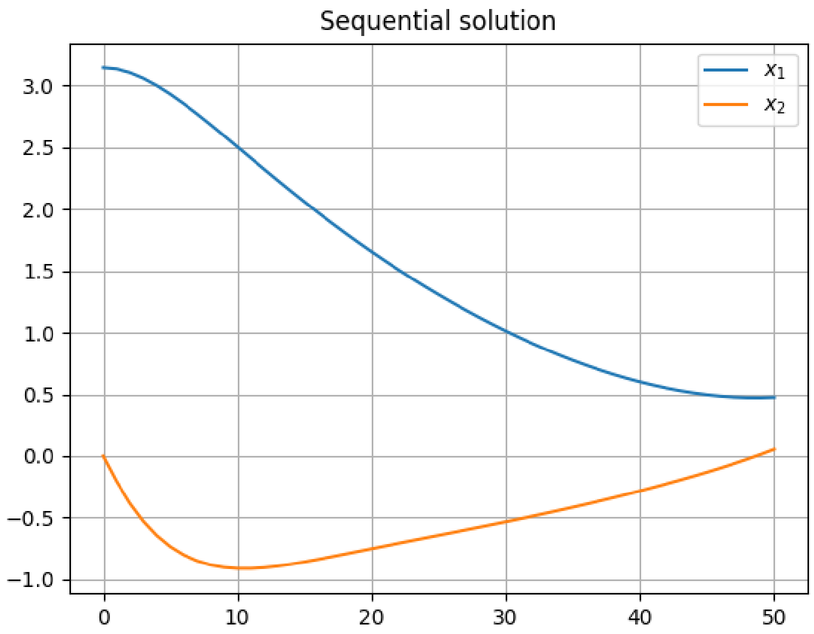
   (a) Implement the optimal control problem using the *sequential* approach (single shooting). The state variables are eliminated using a forward simulation such that the more compact optimization problem

$$\underset{U}{\text{minimize}} \quad \Phi(U, \hat{x}_0) \tag{3}$$

is obtained, with $U = (u_0, \ldots, u_{N-1})$. Solve the problem using `IPOPT` and plot the resulting state and control trajectories in two separate plots.

The solution is a swing-up motion, but it does not reach the upright position ($\bar{\theta} = 0$) within the prediction horizon. To retrieve the states from the optimal solution, a forward simulation with the optimal controls is necessary:

$$x_{k+1} = F(x_k, u_k^*) \quad x_0 = \hat{x}_0 \tag{4}$$

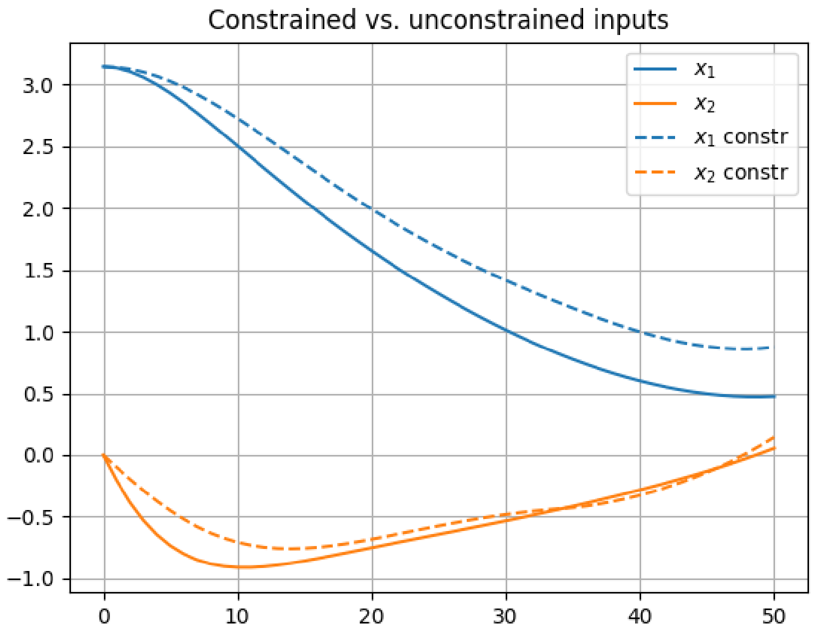


(b) Implement the optimal control problem using the *simultaneous* approach (multiple shooting). Solve the problem using `IPOPT` and make sure that you obtain the same result as with the sequential approach.
The simultaneous solution is identical to that of the sequential problem.

(c) Re-implement the OCP with the input constraints

$$-1 \leq u_k \leq 1, \; \forall k = 0, \ldots, N-1 \,, \tag{5}$$

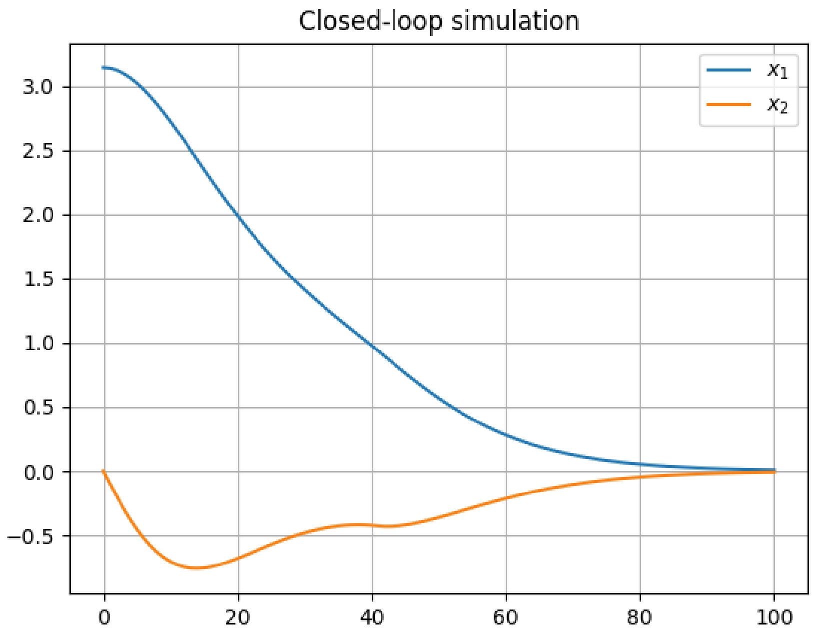
solve the modified problem with `IPOPT` and plot the obtained trajectories.
At the end of the prediction horizon, the constrained solution is further away from the upright position than in the unconstrained case.



3. **Algorithms**. We will now investigate a special algorithm called the "*Real Time Iteration*", which is based on the idea of only performing a single Sequential Quadratic Programming (SQP) step per sampling instant. In this scheme, the closed-loop system and the optimization solver converge *simultaneously*.

   (a) Perform a closed-loop MPC simulation with the simultaneous OCP with input constraints and `IPOPT` as an NLP solver over a time horizon of $T = 10$ s.

   In the closed-loop simulation, the MPC controller manages to bring the pendulum in a still upright position. Notice the deviation w.r.t. to the open-loop solution at $k = 0$ due to the finite horizon.



   (b) Now choose `CasADi`'s SQP method (`sqpmethod` as an NLP solver, with `qpOASES` as an underlying QP solver. Limit the number of solver iterations to 1 and initialize the problem at the next sampling instant with the obtained solution (using `Opti.set_initial()`). Compare the closed-loop responses. Compare the average computation time per sampling interval.

   By performing only one optimization iteration per sampling interval, the average computation time is reduced massively: from $t_{\text{MPC}} \approx 0.5$ s to $t_{\text{RTI}} \approx 0.06$ s. By warmstarting the optimization problem with the prediction result from the previous sampling interval, the RTI-MPC controller converges fast to the optimal solution. The difference in closed-loop performance is only very small.