Exercises for Lecture Course on Numerical Optimal Control (NOC)
Albert-Ludwigs-Universität Freiburg – Summer Term 2017

Exercise 10 (Bonus): Model Predictive Control

Prof. Dr. Moritz Diehl, Andrea Zanelli, Dimitris Kouzoupis

Consider the following optimal control problem:

$$\begin{aligned} \min_{x(t),\, u(t)} \quad & \frac{1}{2}\int_0^T x(t)^T Q_c x(t) + u(t)^T R_c u(t)\,\mathrm{d}t + \frac{1}{2} x(T)^T P x(T) \\ \text{s.t.} \quad & x(0) = \hat{x}_0 \\ & \dot{x}(t) = f(x(t), u(t)), \quad t \in [0, T] \\ & -\bar{u} \le u(t) \le \bar{u}, \end{aligned} \tag{1}$$

where

$$f(x(t), u(t)) := \begin{bmatrix} x_2(t) \\ -x_1(t) + (0.3x_2(t) + 1)u(t) \end{bmatrix} \tag{2}$$

and $\hat{x}_0 = [1,\, 3]^T$ is the initial state of the system. In this exercise, we will design two receding horizon controllers using either an exact solution to the discretized version of (1) or an approximate scheme called *Real-Time Iteration*.

1. (a) Set up a script that, using CasADi, discretizes (1) using multiple shooting and solves the resulting problem using IPOPT. The discretized problem should have the following form:

$$\begin{aligned} \min_{x,\, u} \quad & \frac{1}{2}\sum_{i=0}^{N-1} \left(x_i^T Q x_i + u_i^T R u_i \right) + \frac{1}{2} x_N^T P x_N \\ \text{s.t.} \quad & x_0 = \hat{x}_0 \\ & x_{i+1} = f(x_i, u_i), \ i = 0, ..., N-1, \\ & -\bar{u} \le u_i \le \bar{u}, \quad i = 0, ..., N-1. \end{aligned} \tag{3}$$

Discretize the dynamics with an explicit RK4 integrator, use a horizon of $T = 3$, $N = 20$ intervals and cost matrices $Q = P = \text{diag}(10,\, 0.1)$ and $R = 0.1$. Fix the control bound to $\bar{u} = 5$. Choose the step-size of the integrator to be $h = 0.15$ (no intermediate integration steps). Plot the obtain state and control trajectories over time in two separate plots (group state trajectories in the first one). Use `stairs` for the control trajectory. We will refer to the obtained solution as *open-loop* solution.

(2 bonus points)

(b) Using the code from point (a), set up a script where the system is controlled in *closed-loop* by applying the first optimal control input u_0^* and shifting the prediction horizon forward in a receding horizon fashion. Plot the obtained trajectories on top of the open-loop ones. When controlling the system, a new optimization problem will be formulated and solved based on the current measured or estimated state of the system $\hat{x}_0$. In this way, the trajectories can be adjusted in order to compensate for disturbances.

Consider the nominal case where no disturbances occur and the behavior of the system can be predicted exactly using the model in (2), are the closed-loop trajectories going to be different than the open-loop ones computed at point (a)? Motivate your answer. *Hint: in order to solve several instances of problem* (3) *for different* $\hat{x}_0$*, recall that you can define a parametric optimization problem in CasADi as follows:*

```
    % Create an NLP solver
    prob = struct('f', J, 'x', w, 'g', g, 'p', x_0);
```

where the parameter p can be fed to calls to the solves without having to define a new problem.

(2 bonus points)

(c) Since solving nonlinear optimization problem online can be rather computationally demanding, schemes are present in the literature that exploit approximate solutions. In the following, you will implement the so-called Real-Time Iteration (RTI) scheme, which relies on the solution of a single convex quadratic problem (QP) that locally approximates the optimal control problem (3). In order to do so, consider the following formulation:

$$\begin{aligned}
\min_{x,\,u} \quad & \frac{1}{2}\sum_{i=0}^{N-1}\left(x_i^T Q x_i + u_i^T R u_i\right) + \frac{1}{2}x_N^T P x_N \\
\text{s.t.} \quad & x_0 = \hat{x}_0 \\
& x_{i+1} = A_i(x_i - \tilde{x}_i) + B_i(u_i - \tilde{u}_i) + f_i,\ \ i = 0, ..., N-1, \\
& -\bar{u} \le u_i \le \bar{u}, \quad i = 0, ..., N-1,
\end{aligned} \tag{4}$$

where $(\tilde{x},\,\tilde{u})$ represent the linearization point at which the local approximation is computed. Notice that the nonlinear discretized dynamics have been replaced by a local time-varying model obtained by computing a Taylor series expansion of first order of the RK4 equations. Once a solution to the QP is obtained, the first control input is applied to the system and the linearization point is updated using the obtained state-input vector. Implement the RTI scheme in CasADi and solve the resulting QPs with qpOASES. Plot the closed-loop trajectories in the same figure used for point (a) and (b). *Hint: in order to compute the linearized model in a simple way, you can store the expressions defining the equality constraints in a vector `g` and compute directly its Jacobian with the CasADi function `jacobian`. Moreover, in order to avoid redefining a new problem at every iteration, you can define a parametric problem that has not only $\hat{x}_0$ as a parameter, but also the linearization point $(\tilde{x},\,\tilde{u})$:*

```
    G = Function('G', {w, x0_hat},{g});
    JG = Function('JG', {w, x0_hat},{jacobian(g,w)});

    wk = MX.sym('wk',length(w),1);  % Define linearization point
    g_l = G(wk, x0_hat) + JG(wk, x0_hat)*(w - wk);
    H = kron(eye(N), diag([diag(Q);diag(R)]));
    H = diag([diag(H);diag(Q)]);
    J = 1/2*w.'*H*w;

    p_in = [wk;x0_hat];

    qp = struct('x',w, 'f',J,'g',[g_l],'p', p_in);  % Allocate QP solver
    solver = qpsol('solver', 'qpoases', qp);
```

(4 bonus points)

(d) Shorten the prediction horizon used in point (b) and (c) from $N = 20$ to $N = 5$, keeping $h = 0.15$. How does the performance of the two controllers change?

(2 bonus points)

This sheet gives in total 10 bonus points