TEMPO Summer School on Numerical Optimal Control and Embedded Optimization
University of Freiburg, July 27 - August 7, 2015

Exercise 6: Linear MPC using qpOASES with condensing

Rien Quirynen   Dimitris Kouzoupis   Joachim Ferreau   Moritz Diehl

`http://syscop.de/teaching/numerical-optimal-control/`

Linear MPC using qpOASES: inverted pendulum Let us look again at the example of an inverted pendulum, mounted on top of a cart, of which the system dynamics are described by the same nonlinear ODE system as in Exercise 1.

6.1 We will reuse the linearization of the discrete time RK4 system from Exercise 1 to make an approximate system of the form $x_{k+1} = A\,x_k + B\,u_k$ where the states are $x_k := (p_k, \theta_k, v_k, \omega_k)$.

6.2 The task is to stabilize the pendulum in its unstable upward position, i.e. $p^{\mathrm{ref}} = 0\,\mathrm{m}$ and $\theta^{\mathrm{ref}} = 0\,\mathrm{rad}$. Let us assume that the control input as well as the position of the cart are both bounded, respectively by $-10 \le F \le 10$ and $-1 \le p \le 1$. The discrete time OCP formulation is the following

$$\begin{aligned}
\underset{X,U}{\text{minimize}} \quad & \frac{1}{2}\sum_{k=0}^{N-1} \left\| \begin{bmatrix} x_k \\ u_k \end{bmatrix} - \begin{bmatrix} x_k^{\mathrm{ref}} \\ u_k^{\mathrm{ref}} \end{bmatrix} \right\|_W^2 + \|x_N - x_N^{\mathrm{ref}}\|_P^2 && \text{(1a)}\\
\text{subject to} \quad & x_0 = \bar{x}_0, && \text{(1b)}\\
& 0 = A\,x_k + B\,u_k - x_{k+1}, \quad \forall k = 0,\ldots,N-1, && \text{(1c)}\\
& -2 \le p_k \le 2, \quad \forall k = 0,\ldots,N, && \text{(1d)}\\
& -10 \le u_k \le 10, \quad \forall k = 0,\ldots,N-1, && \text{(1e)}
\end{aligned}$$

where the weighting matrix $W := \mathrm{blkdiag}(Q,R)$ and with state trajectory $X = [x_0^\top, \ldots, x_{N-1}^\top, x_N^\top]^\top$ and control trajectory $U = [u_0^\top, \ldots, u_{N-1}^\top]^\top$. The problem formulation employs $N = 40$ intervals over the control horizon $T = 2$ s.

6.3 Write (first **on paper**) the above Optimal Control Problem in the following Quadratic Programming (QP) formulation, expected by qpOASES:

$$\begin{aligned}
\min_{w} \quad & \frac{1}{2} w^\top H\,w + w^\top g && \text{(2a)}\\
\text{s.t.} \quad & \mathrm{lb} \le w \le \mathrm{ub} \\
& \mathrm{lb_A} \le A\,w \le \mathrm{ub_A} && \text{(2b)}
\end{aligned}$$

where the optimization variables are defined as $w = [x_0^\top, u_0^\top, \ldots, x_{N-1}^\top, u_{N-1}^\top, x_N^\top]^\top$.
NOTE: you can introduce equality constraints by making both bound values equal for the corresponding inequality constraints. Therefore, the initial value condition $x_0 = \bar{x}_0$ can for example be written as simple bounds on the first state $\bar{x}_0 \le x_0 \le \bar{x}_0$. For the case of the dynamics, the equality constraints $A_{\mathrm{eq}}\,w = b_{\mathrm{eq}}$ can be rewritten as inequality constraints $b_{\mathrm{eq}} \le A_{\mathrm{eq}}\,w \le b_{\mathrm{eq}}$.

6.4 Based on the file `closed_loop.m`, implement a closed-loop simulation of linear MPC using the formulation above with qpOASES directly solving the QP at each sampling time. Note that qpOASES has been especially developed to solve such a sequence of closely related parametric optimization problems. Therefore, the first QP can be solved by calling:

```
    [QP_sparse,z_sparse] = qpOASES_sequence('i',H,g,Aeq,...
            lb_sparse,ub_sparse,beq,beq,options);
```

All subsequent QPs can then be solved in a much more efficient manner by calling:

```
    [z_sparse] = qpOASES_sequence('h',QP_sparse,g,...
            lb_sparse,ub_sparse,beq,beq,options);
```

6.5 Compare the performance of the linear MPC scheme with the LQR controller from Exercise 1. For this, it is important to choose the same weighting matrices for both schemes. Also, note that the LQR controller needs to saturate the control inputs while MPC can directly take these constraints into account. Do you observe any other differences in the closed-loop trajectories of both schemes? Additionally, you should plot and compare the closed-loop cost for both controllers.

6.6 Condensing: It is well known that qpOASES is not an ideal solver to directly tackle the sparse optimal control problem in Eq. (1). However, by eliminating the state variables using a procedure called *condensing*, we can form an equivalent problem which is much smaller but dense:

$$\min_{U} \frac{1}{2} U^\top H_c U + U^\top g_c \tag{3a}$$

$$\begin{aligned} \text{s.t.} \quad \mathrm{lb_u} &\leq U \leq \mathrm{ub_u} \\ \mathrm{lb_A} &\leq A_c U \leq \mathrm{ub_A} \end{aligned} \tag{3b}$$

where again $U = [u_0^\top, \ldots, u_{N-1}^\top]^\top$. The latter problem is much more suitable for a dense linear algebra solver like qpOASES. Note that the condensed Hessian H_c and the matrix A_c need to be computed only once (offline) before we start the closed-loop simulation:

```
    [Hc,gc0,gc1,lbU,ubU,Ac,lbA0,ubA0,bA1,AA,BB,CC] = ...
        condensing(Q,R,P,g,A,B,lb,ub,beq);
    gc = gc0 + gc1*x0;
    lbA = lbA0 + bA1*x0;
    ubA = ubA0 + bA1*x0;
    [QP_dense,u_dense] = qpOASES_sequence('i',Hc,gc,Ac,lbU,ubU,lbA,ubA,options);
```

The function `condensing` is part of the provided template code for this exercise and it allows one to expand the control trajectory to obtain the state trajectory by evaluating `x_dense = AA*x0 + BB*u_dense + CC`. In the following (online) iterations, one can update the gradient g_c and the bounds $\mathrm{lb_A}$ and $\mathrm{ub_A}$ given the new current state estimate x_0:

```
    gc = gc0 + gc1*x0;
    lbA = lbA0 + bA1*x0;
    ubA = ubA0 + bA1*x0;
    [u_dense] = qpOASES_sequence('h',QP_dense,gc,lbU,ubU,lbA,ubA,options);
```

Check whether the solution of the condensed QP is indeed equal to that of the original, sparse problem. Finally, using the MATLAB routines `tic/toc`, you can measure the computation time needed by qpOASES both with and without condensing.

6.7 **Extra**: You can also try to solve the same sparse optimal control problem from Eq. (1) using YALMIP, similar to what you did in Exercise 5. For this, you can still use qpOASES as the underlying solver. Note that YALMIP cannot use the `qpOASES_sequence` feature for embedded optimization. So even though your problem becomes much easier to formulate in the YALMIP environment, the actual MPC simulation might slow down quite a bit.