Solution for Exercise 6: Linear MPC using qpOASES with condensing

TEMPO Summer School on Numerical Optimal Control and Embedded Optimization
University of Freiburg, July 27 - August 7, 2015
Rien Quirynen, Dimitris Kouzoupis, Joachim Ferreau and Moritz Diehl

Contents

The inverted pendulum

```
clear all;
close all;
clc;

l = 0.8;
nx = 4; nu = 1;
x0 = [0 0.45 0 0].';
Ts = 0.05;
input.Ts = Ts;
input.nSteps = 3;
options = qpOASES_options;
Fmax = 10;
pmax = 2;

load lqr.mat A B Q R K P

% Setup qpOASES data:
N = 40;
H = [];
g = zeros(N*(nx+nu)+nx,1);
lb = -1e12*ones(N*(nx+nu)+nx,1);
ub = 1e12*ones(N*(nx+nu)+nx,1);
Aeq = zeros(N*nx,N*(nx+nu)+nx);
beq = zeros(N*nx,1);
for i = 1:N
    H = blkdiag(H,Q,R);

    Aeq((i-1)*nx+1:i*nx,(i-1)*(nx+nu)+1:i*(nx+nu)+nx) = [A B -eye(nx)];
    % bounds on position:
    lb((i-1)*(nx+nu)+1) = -pmax;
    ub((i-1)*(nx+nu)+1) = pmax;
    % bounds on force input:
    lb(i*(nx+nu)) = -Fmax;
    ub(i*(nx+nu)) = Fmax;
end
H = blkdiag(H,Q);
P = Q;

lb_sparse = lb;
ub_sparse = ub;
```

```
time = 0;
Tf = 4;
state_LQR = x0;
us_LQR = [];
cost_LQR = 0;
state_MPC = x0;
us_MPC = [];
cost_MPC = 0;
iter = 0;
t_sparse = 0; t_cond = 0;
while time(end) < Tf
```

```
    % optimal feedback law from LQR
    u_LQR = min(max(-K*state_LQR(:,end),-Fmax),Fmax); us_LQR = [us_LQR u_LQR];
    cost_LQR = [cost_LQR cost_LQR(end)+u_LQR.'*R*u_LQR+state_LQR(:,end).'*Q*state_LQR(:
,end)];

    % apply control to nonlinear system
    input.x = state_LQR(:,end);
    input.u = u_LQR;
    output = RK4_integrator( @ode, input );
    state_LQR(:,end+1) = output.value;
```

The sparse QP for linear MPC

optimal feedback law from linear MPC

```
    lb_sparse(1:nx) = state_MPC(:,end);
    ub_sparse(1:nx) = state_MPC(:,end);

    % solve sparse QP
    tic;
    if iter == 0
        [QP_sparse,z_sparse] = qpOASES_sequence('i',H,g,Aeq,lb_sparse,ub_sparse,beq,beq
,options);
    else
        [z_sparse] = qpOASES_sequence('h',QP_sparse,g,lb_sparse,ub_sparse,beq,beq,optio
ns);
    end
    t_sparse = t_sparse + toc;
    u_MPC = z_sparse(nx+nu);
```

The condensed QP for linear MPC

solve condensed QP

```
    tic;

    % solve dense QP
    x0 = state_MPC(:,end);
    if iter == 0
        [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);
    else
        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);
    end
    u_MPC = u_dense(1);
%     z_sparse(nx+nu) - u_dense(1)

    % expand solution
    x_dense = AA*x0 + BB*u_dense + CC;
    t_cond = t_cond + toc;

    % add term to the cost
    us_MPC = [us_MPC u_MPC];
    cost_MPC = [cost_MPC cost_MPC(end)+u_MPC.'*R*u_MPC+state_MPC(:,end).'*Q*state_MPC(:
,end)];

    % apply control to nonlinear system
    input.x = state_MPC(:,end);
    input.u = u_MPC;
    output = RK4_integrator( @ode, input );
    state_MPC(:,end+1) = output.value;

    % next time step and visualize result
    iter = iter+1;
    time(end+1) = iter*Ts;
```

```
end
```

Comparison results: with and without condensing

```
disp(['timing without condensing : ' num2str(t_sparse) ' s'])
disp(['timing with condensing    : ' num2str(t_cond) ' s'])
disp(['        speedup factor    : ' num2str(t_sparse/t_cond)]);
```

```
timing without condensing : 7.4371 s
timing with condensing    : 0.18924 s
        speedup factor    : 39.2998
```

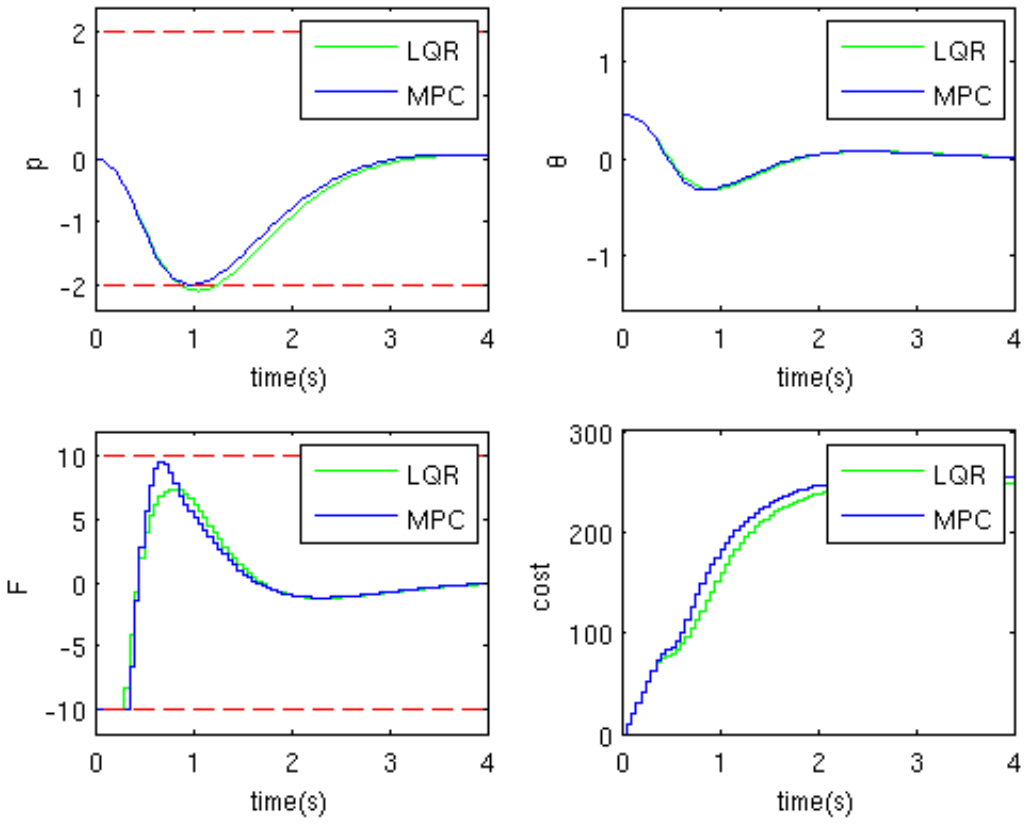
Comparison results: LQR versus linear MPC

```
figure(10);
subplot(2,2,1);
plot(time, state_LQR(1,:), '-g'); hold on;
plot(time, state_MPC(1,:), '-b');
xlabel('time(s)'); ylabel('p')
ylim([-pmax*1.2 pmax*1.2]);
xlim([0 Tf]);
legend('LQR', 'MPC')
plot([0 Tf], [pmax pmax], '--r'); hold on;
plot([0 Tf], [-pmax -pmax], '--r');
```

```
subplot(2,2,2);
plot(time, state_LQR(2,:), '-g'); hold on;
plot(time, state_MPC(2,:), '-b');
xlabel('time(s)'); ylabel('\theta')
ylim([-pi/2 pi/2]);
xlim([0 Tf]);
legend('LQR', 'MPC')

subplot(2,2,3);
stairs(time(1:end-1), us_LQR, '-g'); hold on;
stairs(time(1:end-1), us_MPC, '-b');
xlabel('time(s)'); ylabel('F')
ylim([-Fmax*1.2 Fmax*1.2]);
xlim([0 Tf]);
legend('LQR', 'MPC')
plot([0 Tf], [Fmax Fmax], '--r'); hold on;
plot([0 Tf], [-Fmax -Fmax], '--r');

subplot(2,2,4);
stairs(time, cost_LQR, '-g'); hold on;
stairs(time, cost_MPC, '-b');
xlabel('time(s)'); ylabel('cost')
xlim([0 Tf]);
legend('LQR', 'MPC')
```



Published with MATLAB® R2014a