Model Predictive Control ME-425
Prof. Colin Jones

Exercise sheet 4
Implement Linear MPC

Exercise 1 : Implement MPC

Consider the discrete-time linear time-invariant system defined by

$$x^+ = \begin{bmatrix} 0.9752 & 1.4544 \\ -0.0327 & 0.9315 \end{bmatrix} x + \begin{bmatrix} 0.0248 \\ 0.0327 \end{bmatrix} u$$

with constraints

$$\mathbb{X} = \{x \mid |x_1| \leq 5,\ |x_2| \leq 0.2\} \qquad \mathbb{U} = \{u \mid |u| \leq 1.75\}$$

This is a second-order system with a natural frequency of $0.15r/s$, a damping ratio of $\zeta = 0.1$ which has been discretized at $1.5r/s$. The first state is the position, and the second is velocity.

Your goal is to implement an MPC controller for this system with a horizon of $N = 10$ and a stage cost given by $l(x, u) := 10x^T x + u^T u$.

Tasks:

1. Compute a terminal controller, weight and set that will ensure recursive feasibility and stability of the closed-loop system.
   Compute the sets and weights using your code from last week, and then repeat the procedure to validate your results using MPT3 as follows:
   - Define the system `sys = LTISystem('A',A,'B',B)`
   - Define the constraints on the signals by setting the values `sys.x.max = ...`, `sys.x.min = ...`, etc
   - Define the stage costs by setting the penalty terms for x and u, e.g., `sys.x.penalty = QuadFunction(Q)`
   - Extract desired sets and weights with `sys.LQRGain`, `sys.LQRPenalty.weight` and `sys.LQRSet`

2. Compute matrices so that the MPC problem can be solved using the Matlab optimization function `[zopt, fval, flag] = quadprog(H, h, G, g, T, t)`, which solves the optimization problem

$$\begin{aligned} fval = \min\ & \frac{1}{2} z^T H z + h^T z \\ \text{s.t. } & Gz \leq g \\ & Tz = t \end{aligned}$$

   You must check the `flag` every time you call an optimization routine to confirm that an optimal solution was found (only if `flag == 1` for quadprog). If the solver did not find a solution, the variable zopt (and hence your control input) will be nonsense.

3. Simulate the closed-loop system starting from the state $x = \begin{bmatrix} 3 & 0 \end{bmatrix}^T$
   Confirm that your constraints are met.
   Change the tuning parameters Q and R. Does the system respond as expected?

Deliverables (submit to Moodle):

1. Plot of the position, velocity and input of the system. Confirm that your constraints were met.

2. Matlab code implementing your simulation

Exercise 2 : Implement MPC using YALMIP

Repeat the first exercise, but now make use of the Matlab optimization toolbox YALMIP. You will use this toolbox for the rest of the exercises, and the project.

A simple example of implementing MPC in YALMIP is given below:

```
% Define optimization variables
x = sdpvar(2,N,'full');
u = sdpvar(1,N,'full');

% Define constraints and objective
con = [];
obj = 0;
for i = 1:N-1
  con = con + set(x(:,i+1) == A*x(:,i) + B*u(:,i)); % System dynamics
  con = con + set(F*x(:,i) <= f);                   % State constraints
  con = con + set(M*u(:,i) <= m);                   % Input constraints
  obj = obj + x(:,i)'*Q*x(:,i) + u(:,i)'*R*u(:,i);  % Cost function
end
con = con + set(Ff*x(:,N) <= ff); % Terminal constraint
obj = obj + x(:,N)'*Qf*x(:,N);    % Terminal weight

% Compile the matrices
ctrl = optimizer(con, obj, [], x(:,1), u(:,1));

% Can now compute the optimal control input using
[uopt,isfeasible] = ctrl{x0}
% isfeasible == 1 if the problem was solved successfully
```

Tasks:

1. Read the webpage `http://users.isy.liu.se/johanl/yalmip/pmwiki.php?n=Examples.StandardMPC`
2. Implement your controller from the first exercise again, now using YALMIP. Confirm that the solution is the same.

Deliverables (submit to Moodle):

1. Plot of the position, velocity and input of the system. Confirm that your solution is the same as for exercise 1.
2. Matlab code implementing your simulation