Model Predictive Control
Prof. Manfred Morari

ESE 6190, Spring 2023
Due: April 19

Exercise sheet 7
Explicit, Robust, & Hybrid MPC

Instructions:

You could use MATLAB publisher to print out your solutions. This allows the comments and the plots to appear inline as they are in the MATLAB script.

Exercise 1 **Multiparametric Programming and Dynamic Programming**

[8 pt] Consider the discrete-time system model

$$\begin{cases} x_{k+1} &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x_k + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_k \\ y_k &= \begin{bmatrix} 1 & 0 \end{bmatrix} x_k. \end{cases} \tag{1}$$

Define the following cost function

$$\left\| \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x_4 \right\|_\infty + \sum_{k=0}^{3} \left(\left\| \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x_k \right\|_\infty + |0.8u_k| \right) \tag{2}$$

and assume the constraints are

$$-1 \leq u_k \leq 1 \quad k = 0, 1, \cdots, 3, \tag{3a}$$

$$\begin{bmatrix} -10 \\ -10 \end{bmatrix} \leq x_k \leq \begin{bmatrix} 10 \\ 10 \end{bmatrix} \quad k = 0, 1, \cdots, 4. \tag{3b}$$

In the previous homework, you calculated the state feedback solution $u_0^\star(x_0)$, $u_1^\star(x_1)$, $\cdots$, $u_3^\star(x_3)$ using the batch approach and mpLPs. Based on the discussion in the class, now calculate the same state feedback solution $u_0^\star(x_0)$, $u_1^\star(x_1)$, $\cdots$, $u_3^\star(x_3)$ using dynamic programming and mpLPs.

Hint: To extract data from the cost-to-go at each step, read the second last entry of the MPT FAQ: `https://www.mpt3.org/Main/FAQ`

Exercise 2 **Robust MPC**

Consider the following system

$$x(k+1) \quad = \quad \begin{bmatrix} 1.2 & 1 \\ 0 & 1 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k) \tag{4}$$

The state and input constraints are

$$\mathcal{U} : -1 \leq u(k) \leq 1; \tag{5a}$$

$$\mathcal{X} : \begin{bmatrix} -15 \\ -15 \end{bmatrix} \leq x(k) \leq \begin{bmatrix} 15 \\ 15 \end{bmatrix}; \tag{5b}$$

We will design a 2-norm MPC with $Q = eye(2)$, $R = 1$ and $N = 3$.

1. **[2 pts]** Choose P to be the solution to the discrete-time algebraic Riccati equation (DARE), which is the infinite LQR solution. Then, set $\mathcal{X}_f$ to be the maximal invariant set O_∞ defined by the closed loop solution $x(k+1) = (A + BF_\infty)x(k)$. To compute and plot O_∞ using MPT3 tools, use the following code:

```
system = LTISystem('A',Acl);
Xtilde = Polyhedron('A',[eye(2);-eye(2);Finf;-Finf],'b',[15;15;15;15;1;1]);
Oinf = system.invariantSet('X',Xtilde)
figure
plot(Oinf)
```

   where `Acl` is the closed loop matrix $A + BF_\infty$ and `Finf` is the LQR control law gain. Note that the set `Xtilde` is just the intersection of $\mathcal{X}$ and the set of x such that $F_\infty x \in \mathcal{U}$. The set O_∞ is then the set $\{x|(\texttt{Oinf.A})x \leq \texttt{Oinf.b}\}$ where `Oinf.A` and `Oinf.B` are extracted from the computed O_∞.

2. **[5 pts]** Use the terminal cost and constraint computed in Part 1 to compute $u_0^*(x(0))$ using an mp-QP. Plot $u_0^*(x(0))$.

3. **[5 pts]** Let $x(0) = [2, -1]^T$ and plot the closed-loop state trajectory (in x-space) for 25 time steps. You should use $u_0^*(x(0))$ computed in the previous step.

   Hint: Visit the page `https://www.mpt3.org/ParOpt/ParOpt` if you need help remembering how to do parametric programming in MPT3.

Consider now the same system with additive disturbance

$$x(k+1) \quad = \quad \begin{bmatrix} 1.2 & 1 \\ 0 & 1 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k) + d(k) \tag{6}$$

The state, input, and disturbance constraints are

$$\mathcal{U} : -1 \leq u(k) \leq 1; \tag{7a}$$

$$\mathcal{X} : \begin{bmatrix} -15 \\ -15 \end{bmatrix} \leq x(k) \leq \begin{bmatrix} 15 \\ 15 \end{bmatrix}; \tag{7b}$$

$$\mathcal{D} : \begin{bmatrix} -0.1 \\ -0.1 \end{bmatrix} \leq d(k) \leq \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix}; \tag{7c}$$

We will now design a robust 2-norm MPC with $Q = eye(2)$, $R = 1$ and $N = 3$.

4. **[5 pts]** Consider an open-loop input policy. Robustify the state constraints so that the MPC is persistently feasible.

5. **[5 pts]** Now, compute $u_0^*(x(0))$ using an mp-QP. Plot $u_0^*(x(0))$.

6. **[5 pts]** Create a disturbance matrix `d` of size 2×25 by sampling uniformly in $\mathcal{D}$ such that `d(:,t)`$= d(t-1)$. Let $x(0) = [2, -1]^T$ and plot the closed-loop state trajectory for 25 time steps. Note: Use `rng(0)` before using the command `rand`.

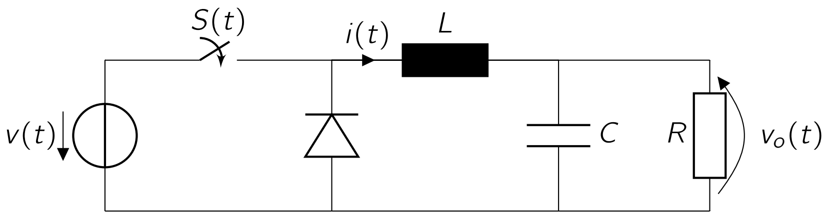

Figure 1: Idealized Buck converter circuit

Exercise 3 **Hybrid MPC**

We consider an idealized Buck converter as depicted in Figure 1. With the supply voltage $v(t)$, the controlled switch $S(t)$ and the regulated output voltage $v_o(t)$. This means that given a supply voltage $v(t)$ (that we cannot control) we want to choose the switches $S(t)$ such that we can produce a desired output voltage signal $v_o(t)$.

The Buck converter operates in three modes.

1. Conducting, i.e. S is **closed**
2. Discharging in continuous mode, i.e. S is **open** and $i(t) > 0$
3. Discharging in discontinuous mode, i.e. S is **open** and $i(t) = 0$

We model the discrete-time dynamics of the hybrid system given in Figure 1 as follows:

$$x_{k+1} = \begin{cases} A_c x_k + E v_k & \text{if } S(t_k) \text{ is } \textbf{closed} \\ A_c x_k & \text{if } S(t_k) \text{ is } \textbf{open} \text{ and } x_{1,k} > 0 \\ A_d x_k & \text{if } S(t_k) \text{ is } \textbf{open} \text{ and } x_{1,k} \le 0 \end{cases} \tag{8}$$

where $x_k = \begin{pmatrix} x_{1,k} & x_{2,k} \end{pmatrix}^T = \begin{pmatrix} i_k & v_{o,k} \end{pmatrix}^T$.
We assume that the voltage $v(t) = v_{\mathrm{DC}} \in [19\,\mathrm{V}, 21\,\mathrm{V}]$ is a constant parameter.

1. **[6 pts]** First re-write (8) as an equivalent piecewise affine model in the following form:

$$x_{k+1} = A_i x_k + B_i u_k + f_i\,, \quad \text{if } (x_k, u_k) \in \mathcal{X}_i\,,$$

by appropriately defining the input u_k, the model parameters A_i, B_i and f_i, as well as the sets $\mathcal{X}_i$.

2. Now we will transform the piecewise affine model into a mixed logical dynamical (MLD) model. To distinguish between three modes, at least two binary variables are needed. The input u_k is binary already, i.e. $u_k \in \{0, 1\}$ and we can introduce an additional binary variable $\delta_k \in \{0, 1\}$ to distinguish between mode two and three, these are the two required binary variables. Furthermore we introduce a continuous auxiliary variable z_k, the dynamics can then be written as an linear equality constraint involving x_k, u_k and z_k:

$$x_{k+1} = (A_c - A_d) z_k + E v_{\mathrm{DC}} u_k + A_d x_k\,,$$

with the additional requirements that

$$\{\delta_k = 0\} \Leftrightarrow \{x_{k,1} \le 0\}\,, \quad \text{and}$$
$$z_k = x_k \delta_k\,.$$

We can assume that $|x_{k,1}| \leq 1\,\mathrm{A}$. We will now use the Big-M reformulation of the two requirements and bring them into a form where they can be represented by affine equalities and inequalities.

i) **[4 pts]** Firstly, give the Big-M formulation for the logical relationship between δ_k and $x_{k,1}$:

$$\{\delta_k = 0\} \Leftrightarrow \{x_{k,1} \leq 0\},$$

and compute M_i, m_i explicitly.

ii) **[3 pts]** Secondly, we would like to replace $z_k = x_k \delta_k$ by reformulating this relationship using the Big-M reformulation. How can we choose $M_{ii}, m_{ii} \in \mathbb{R}^2$?

iii) **[3 pts]** Consider the case where $u_k = 1$ and $\delta_k = 0$. Any observations? How can we fix this problem by adding an additional constraint?

3. Given the piecewise affine model derived in Part 1 we want to implement a hybrid model predictive controller using Mpt in Matlab. The model parameters are given as follows:

$$A_c := \begin{bmatrix} 0.9786 & -0.021 \\ 1.8892 & 0.8841 \end{bmatrix}, \quad E := \begin{bmatrix} 0.0221 \\ 0.0214 \end{bmatrix}, \quad A_d := \begin{bmatrix} 1 & 0 \\ 0 & 0.9048 \end{bmatrix}.$$

We assume that the full state x_k can be measured. The supply voltage v_{DC} is 20 V and we would like to track an output voltage reference signal $v_{o,\mathrm{ref},k}$, while keeping the magnitude of the output voltage $v_{o,k}$ bounded by 10 V. Furthermore we would like the current i_k to stay within $[-1\,\mathrm{A}, 1\,\mathrm{A}]$. The sampling time T_s is 0.02 s.

i) **[3 pts]** Create the piecewise affine model using Mpt's `PWASystem` command. Do this by defining an `LTISystem` for each piece of you piecewise affine dynamics in the following way

» `sys1 = LTISystem('A',` A_1`, 'B',` B_1`, 'C',` C_1`, 'D',` D_1`, ...`

» `'f', '`f_1`', 'Ts',` T_s`);`

with appropriate A_1, B_1, C_1, D_1, f_1 and T_s for each piece. State and input constraints can be added via

» `sys.setDomain('xu', ...`

» `Polyhedron('lb', [`x_{min}`;` u_{min}`], 'ub' [`x_{max}`;` u_{max}`]));`

Note: The upper and lower bounds will be different for the different pieces. If you have an LTI model (not hybrid) then you can just use `LTISystem` to generate a system and treat it the same way we have treated `PWASystem` in this exercise.

Then combine the `LTISystems` using `PWASystem` like so

» `model = PWASystem([sys1, sys2, sys3]);`

and add the constraints on states and inputs also to this system

» `model.x.min = [-1,-10]; model.x.max = [1,10];`

» `model.u.min = 0; model.u.max = 1;`

Note: Check `https://www.mpt3.org/UI/Systems` for more information on `PWASystem`.

This system can now be used to create an MPC controller and a closed-loop simulation just as in the previous exercise.

ii) **[3 pts]** We want to use the model to track a reference of 5 V for the output voltage v_o. Set

```
» model.x.with('reference');
» model.x.reference = 'free';
```
to indicate that we want to have a time-varying reference on the state. And set a 2-norm penalty on x using an appropriate quadratic cost matrix Q to track the second state.
```
» model.x.penalty = QuadFunction(Q);
```

Then
```
» ctrl = MPCController(model, 4);
```
generates a hybrid MPC controller with horizon $N = 4$. The resulting optimization problem is a mixed-integer quadratic program, which usually needs commerical solvers such as CPLEX, Gurobi or MOSEK to solve, however for N not too large we can use Mpt to generate an explicit solution, just as we did in the previous exercise using Yalmip. In Mpt when we have a controller we can just call `toExplicit()` to generate an explicit solution. We do this by
```
» ehmpc = ctrl.toExplicit();
```
We could now plot the partition of the explicit solution, as we did in the previous exercise, using `ehmpc.partition.plot()`, however in this case this is not in 2D or 3D and can therefore not be visualized.

Then you can run a closed-loop simulation by first defining a plant. In this case we use the model, however to run a simulation with model mismatch you may sometimes want to use a plant that is different from your model
```
» plant = model;
```
then we setup a closed-loop object that consists of our controller and the plant.
```
» loop = ClosedLoop(ehmpc, plant);
```
We define initial condition, number of time steps that we want to simulate, the reference and then run a simulation using `loop.simulate()`.
```
» x0 = [0; 0];
» Nsim = 200;
» xref = [0;5];
» data = loop.simulate(x0, Nsim, 'x.reference', xref);
» figure(); subplot(2,1,1); stairs(data.X');
» subplot(2,1,2); stairs(data.U');
```
What do you observe? What if
```
» xref = [0; 1];
```
or
```
» xref = [zeros(1,Nsim); ...
» 2.5*(1-exp(-(0:Nsim-1)/10)).*(1+0.5*sin((0:Nsim-1)/30*2*pi))];;
```
How do you explain this behavior?

iii) **[3 pts]** Play with the parameters N, the horizon length, x_0, the initial condition and the reference signal. What do you observe? You can also change the objective function replacing `QuadFunction(Q)` with `OneNormFunction(Q)` or `InfNormFunction(Q)`. You can try the system sampled with $T_s = 0.005\,\text{s}$ given below:

$$A_c := \begin{bmatrix} 0.9986 & -0.0055 \\ 0.4936 & 0.9739 \end{bmatrix}, \quad E := \begin{bmatrix} 0.0056 \\ 0.0014 \end{bmatrix}, \quad A_d := \begin{bmatrix} 1 & 0 \\ 0 & 0.9753 \end{bmatrix}.$$