ESE 619 Model Predictive Control **Homework 7: Explicit, Robust, & Hybrid MPC**
University of Pennsylvania, Spring 2023 updated April 21, 2023
Student Name: Wei-Cheng (Rory) Huang *Penn ID: 55353592*

The solutions of this homework are entirely my own. I have discussed these problems with several classmates, they are: Shiming Liang.

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:

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

and assume the constraints are:

$$-1 \le u_k \le 1 \quad k = 0, 1, \ldots, 3 \tag{3a}$$

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

In the previous homework, you calculated the state feedback solution $u_0^*(x_0), u_1^*(x_1), \ldots, u_3^*(x_3)$ using the batch approach and mpLPs. Based on the discussion in the class, now calculate the same state feedback solution $u_0^*(x_0), u_1^*(x_1), \ldots, u_3^*(x_3)$ using dynamic programming and mpLPs.
Solution:
As is describe in the MPC book Section 11.4.3, Theorem 11.6. The state feedback piecewise affine solution of the infinity norm CFTOC for can be obtained by solving multiple mp-LPs using dynamic programming, more specifically:
Consider the end step $j = N-1$ of dynamic programming, we have:

$$\begin{aligned} J_{N-1}^*(x_{N-1}) = \min_{u_{N-1}} \; & \|Qx_{N-1}\|_\infty + \|Ru_{N-1}\|_\infty + J_N^*(Ax_{N-1} + Bu_{N-1}) \\ \text{subj. to.} \quad & x_{N-1} \in \mathcal{X}, \quad u_{N-1} \in \mathcal{U} \\ & Ax_{N-1} + Bu_{N-1} \in \mathcal{X}_f \end{aligned}$$

$J_{N-1}^*(x_{N-1})$ and $u_{N-1}^*(x_{N-1})$ and $\mathcal{X}_{N-1}$ are computable via a single mp-LP. By theorem, $J_{N-1}^*(x_{N-1})$ is a convex and piecewise affine function of x_{N-1}, the corresponding optimizer $u_{N-1}^*(x_{N-1})$ is piecewise affine and continous, and feasible set $\mathcal{X}_{N-1}$ is a polyhedron. Without loss of generality, we actually can assume $J_{N-1}^*(x_{N-1})$ to be described as:

$$J_{N-1}^*(x_{N-1}) = \max_{i=1,\ldots,n_{N-1}} \left\{ c_i^{\mathrm{T}} x_{N-1} + d_i \right\}$$

where n_{N-1} is the number of affine conponents comprising $J_{N-1}^*(x_{N-1})$.
Then, at the step $j = N-2$, we have:

$$\begin{aligned} J_{N-2}^*(x_{N-2}) = \min_{u_{N-2}} \; & \|Qx_{N-2}\|_\infty + \|Ru_{N-2}\|_\infty + J_{N-1}^*(Ax_{N-2} + Bu_{N-2}) \\ \text{subj. to.} \quad & x_{N-2} \in \mathcal{X}, u_{N-2} \in \mathcal{U} \\ & Ax_{N-2} + Bu_{N-2} \in \mathcal{X}_{N-1} \end{aligned}$$

As is mentioned above $J^*_{N-1}(x_{N-1})$ is a convex and piecewise affine, $\mathcal{X}_{N-1}$ is a polyhedron. Therefore, the above problem again can be solved by a single mp-LP and would give us $J^*_{N-2}(x_{N-2})$ as a convex and piecewise affine function of x_{N-2}, piecewise affine and continous optimizer $u^*_{N-2}(x_{N-2})$ and polyhedron feasible set $\mathcal{X}_{N-2}$.
It is not hard to observe that the convexity and piecewisely linearity of J^*_j and polyhedroa representation of $\mathcal{X}_j$ would still hold of $j = N-3, \ldots, 0$ and hence the procudure can be iterated backwars in time. In summary, solving the whole dynamic programming requires solving N mp-LP, where N is the horizon.
See the MATLAB live script for the coding details, the required figure showing the state feedback solution $u^*_j(x_j)$ is shown below, it is exactly the same as what we derived using forward batch method in the previous homework.

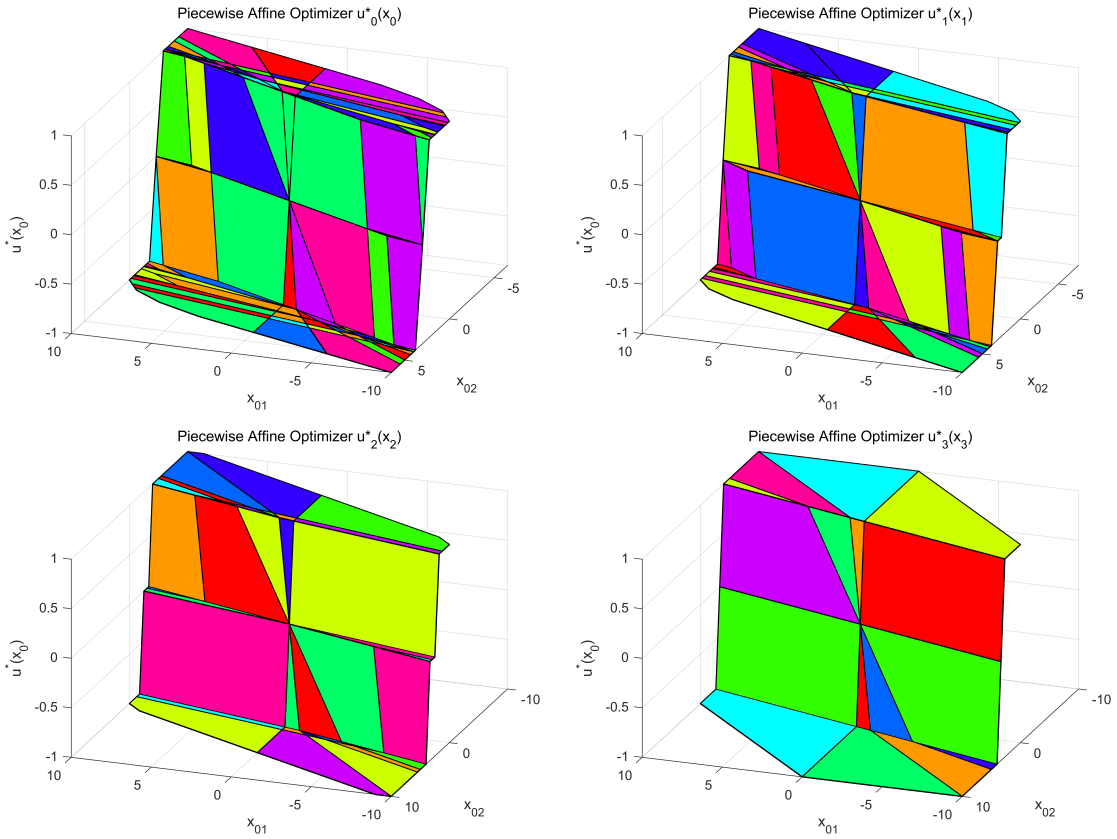

Figure 1: state feedback solution $u^*_0(x_0), u^*_1(x_1), \ldots, u^*_3(x_3)$

2. **Robust MPC**

Consider the following system:

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

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 $Q = I_2$, $R = 1$ 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 $\mathcal{O}_\infty$ defined by the closed loop solution $x(k+1) = (A + BF_\infty)x(k)$ To compute and plot $\mathcal{O}_\infty$ 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 $\mathcal{O}_\infty$ is then the set $\{x | \texttt{(Oinf.A)x} \leq \texttt{Oinf.b}\}$ where `Oinf.A` and `Oinf.b` are extracted from the computed $\mathcal{O}_\infty$.
Solution: See the live scripts for coding details. The required figure is shown below:

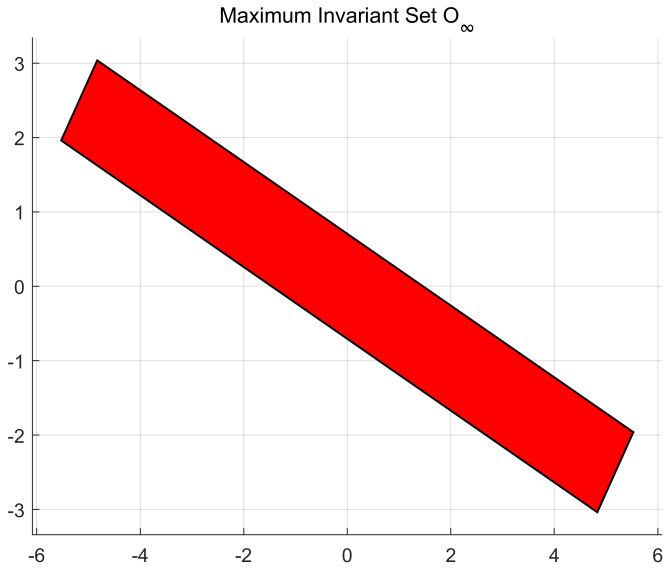

Figure 2: The maximum invariant set $\mathcal{O}_\infty$

2. **[5 pts]** Use the terminal cost and constraint computed in Part 1 to compute $u_0^*(x_0)$ using an mp-QP. Plot $u_0^*(x_0)$
Solution: See the live scripts for coding details. The required figure is shown below:

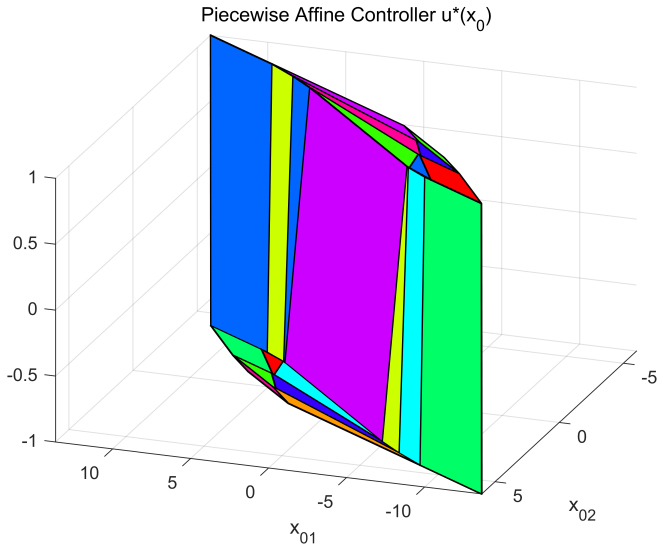

Figure 3: The piecewise affine controller $u_0^*(x_0)$ corresponds to LQR cost and terminal set

3. **[5 pts]** Let $x(0) = \begin{bmatrix} 2 & -1 \end{bmatrix}^{\mathrm{T}}$ 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.

   Solution: See the live scripts for coding details. The required figure is shown below:

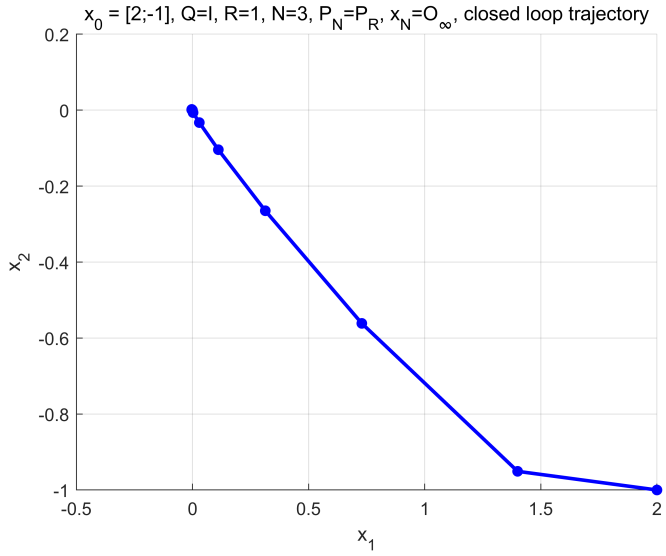

Figure 4: 25 step closed loop trajectory of the system

Consider now the same system with additive disturbance:

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

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 $Q = I_2$, $R = 1$ and $N = 3$.

4. **[5 pts]** Consider an open-loop input policy. Robustify the state constraints so that the MPC is persistently feasible.
   Solution:
   The open loop robustify need us to calculate the robust invariant set. it also reqiures us to calculate the worst case for all the possible noise and shrink our state constraint set, which would turn to doing a series of linear programming (or equivalently, Pontryagin difference for polytopes). For illustration, we here demonstratet the robust invariant set and the terminal set after tighteneing the robust invariant set.
   For more coding and implementing details, see the live script.

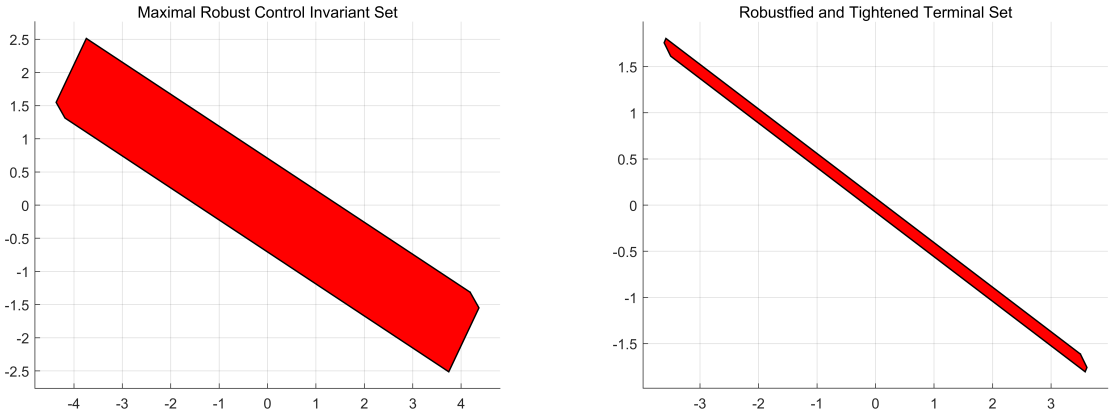

Figure 5: The Robust invariant Set and tightened terminal set of the open-loop robust MPC

5. **[5 pts]** Now, compute $u_0^*(x_0)$ using an mp-QP. Plot $u_0^*(x_0)$
   Solution: See the live scripts for coding details. The required figure is shown below:

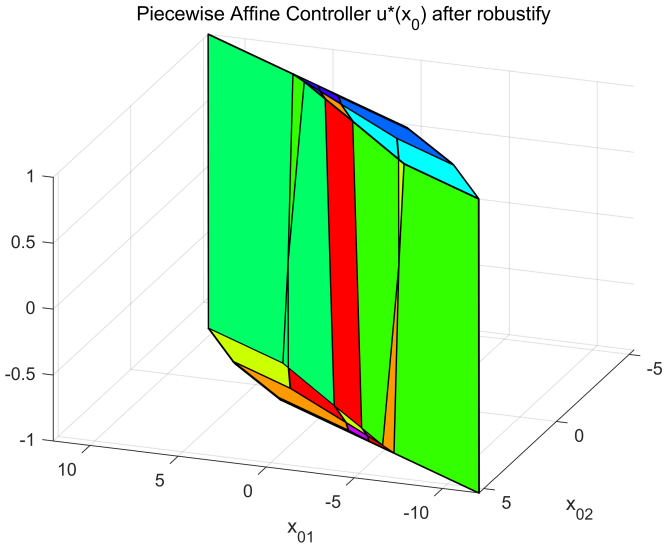

Figure 6: The piecewise affine controller $u_0^*(x_0)$ after robustifying the state constraints

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 $x(0) = \begin{bmatrix} 2 & -1 \end{bmatrix}^{\mathrm{T}}$ and plot the closed-loop state trajectory for 25 time steps. Note: Use `rng(0)` before using the command rand.
   Solution: See the live scripts for coding details. The required figure is shown below:

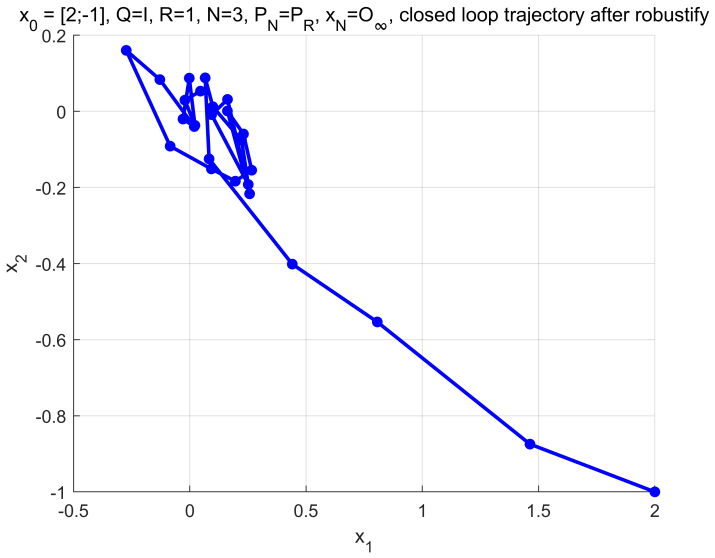

Figure 7: 25 step closed loop trajectory of the disturbed system

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 $v_o(t)$.

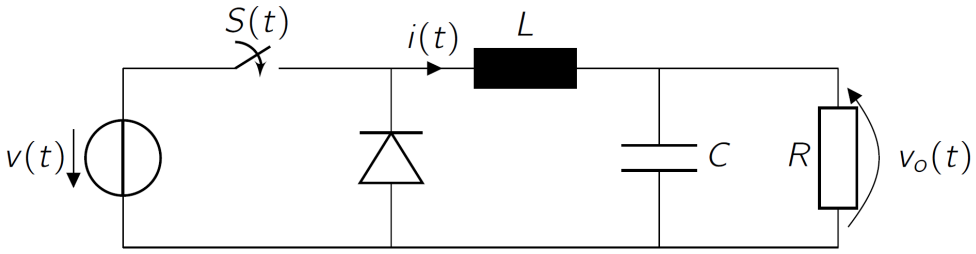

Figure 8: Idealized Buck converter circuit

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 continuous 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:

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

where $x_k = \begin{bmatrix} x_{1,k} & x_{2,k} \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} i_k & v_{o,k} \end{bmatrix}^{\mathrm{T}}$

We assume that the voltage $v(t) = v_{DC} \in [19\text{ V}, 21\text{ V}]$ is a constant parameter.

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

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

by appropriately defining the input u_k , the model parameters A_i, B_i and f_i , and the sets $\mathcal{X}_i$.
Solution:
The input u_k is a binary input $u_k \in \{0, 1\}$ stands for the switch. The hybrid system in (8) is then equivalently written as:

$$x_{k+1} = \begin{cases} A_1 x_k + B_1 u_k + f_1 & (x_k, u_k) \in \mathbb{R} \times \mathbb{R} \times \{1\} \\ A_2 x_k + B_2 u_k + f_2 & (x_k, u_k) \in \mathbb{R}^{+} \times \mathbb{R} \times \{0\} \\ A_3 x_k + B_3 u_k + f_3 & (x_k, u_k) \in \mathbb{R}^{\leq 0} \times \mathbb{R} \times \{0\} \end{cases}$$

where $A_1 = A_2 = A_c, A_3 = A_d, B_1 = B_2 = B_3 = E v_k, f_1 = f_2 = f_3 = 0, x_k = \begin{bmatrix} x_{1k} & x_{2k} \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} i_k & v_{o,k} \end{bmatrix}^{\mathrm{T}}, u_k \in \{0, 1\}$

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:

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

with the additional requirements that:

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

We can assume that $|x_{k,1}| \leq 1$ 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.

Solution:

We want to translate $x_{k,1} \leq 0 \Leftrightarrow \delta_k = 0$, note that the affine expression $x_k, 1$ is bounded, so the M_i, m_i should be selected as the minimum and maximum of the affine expression, as the following shows:

$$x_{k,1} \in \mathcal{X}_{k,1} = [-1, 1] \Rightarrow \begin{cases} m_i = \min_{\mathcal{X}_{k,1}} \; x_{k,1} = -1 \\ M_i = \max_{\mathcal{X}_{k,1}} \; x_{k,1} = 1 \end{cases}$$

And the corresponding big-M formulation is:

$$\begin{cases} x_{k,1} \leq M_i \delta_k \\ x_{k,1} > m_i \,(1 - \delta_k) \end{cases} \Rightarrow \begin{cases} x_{k,1} \leq \delta_k \\ x_{k,1} > \delta_k - 1 \end{cases}$$

Also note that considering numerical issues, we might need to introduce small positive number ϵ in the big-M formulation in real practice. e.g. $x_{k,1} \geq (m_i - \epsilon)\,(1 - \delta_k) + \epsilon, \epsilon > 0$ But the theoretical translation is listed above.

(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 $M_{ii}, m_{ii} \in \mathbb{R}^2$?

Solution:

We want to translate $z_k = x_k \delta_k \Leftrightarrow \delta_k = 1$, which is the "if-else" structure with the special case that $z_k = 0$ when $\delta_k = 0$, so we only need to consider a single set of M_{ii}, m_{ii}, and this set of parameter still need to be selected as the Infimum and supremum of the affine expression x_k, that is:

$$x_k \in \mathcal{X}_k = [-1, 1] \times [v_{0\,\min}, v_{0\,\max}] \Rightarrow \begin{cases} m_{ii} = \inf\limits_{\mathcal{X}_k} \; x_k = \begin{bmatrix} -1 & v_{0\,\min} \end{bmatrix}^{\mathrm{T}} \\ M_{ii} = \sup\limits_{\mathcal{X}_k} \; x_k = \begin{bmatrix} 1 & v_{0\,\max} \end{bmatrix}^{\mathrm{T}} \end{cases}$$

Also, the corresponding big-M formulation of this relationship is:

$$\begin{cases} -M_{ii}\delta_k + z_k \leq 0 \\ m_{ii}\delta_k - z_k \leq 0 \\ m_{ii}\,(1 - \delta_k) + z_k \leq x_k \\ -M_{ii}\,(1 - \delta_k) - z_k \leq -x_k \end{cases}$$

Also note that considering numerical computing issues, we might need to introduce small positive number ϵ in the big-M formulation in real practice. But the theoretical translation is listed above.

(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?
Solution:
It can be observed that when $u_k = 1$ and $\delta_k = 0$, we have the mode where the dynamics is $x_{k+1} = A_d x_k + E v_{DC} u_k$. It is a undefined and undesired mode in our problem setting and we have this issue because introducing two binary variables can at most represent 4 modes, but we only need 3 of them.
We can tackle this issue by introducing additional constraints that when $u_k = 1$, δ_k must also be 1. and this can be done by the following constraint:

$$\delta_k \geq u_k$$

This would eliminate the mode where $u_k = 1$ and $\delta_k = 0$

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 $v_{o,ref,k}$ 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 $[-1\text{ A}, 1\text{ A}]$. 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` or each piece of you piecewise affine dynamics in the following way

```
sys1 = LTISystem('A', A1, 'B', B1, 'C', C1, 'D', D1, ...'f', 'f1', 'Ts', Ts);
```

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', [xmin; umin], 'ub' [xmax; umax]));
```

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 `LTISystem` 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;
```

This system can now be used to create an MPC controller and a closed-loop simulation just as in the previous exercise.
Solution: See the live scripts for coding details.

(ii) **[3 pts]** We want to use the model to track a reference of 5 V for the output voltage v_0. 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?
Solution: See the live scripts for coding details. The required figure is shown below:
Discussions and Observations:

(a) In Fig.9, the output voltage settles at around 5 V with small ripples. The switch changes mode quiet frequently to make it track the voltage of 5 V DC with some minimal error.

(b) In Fig.10, the reference to be tracked is 1 V. It can be observed that the the switch is not changing mode frequently, or in other words, the duty cycle is small. A sampling time of $T_s = 0.02$s is too large for this application and the tracking performace is poor, it hardly settles at 1 V.

(c) In Fig.11, the reference voltage is time varying (a product of exponential and sinusoid, mostly sinusoid) and the controller is able to track the waveform with some error. However, the tracked voltage is still distorted due to large sampling time.

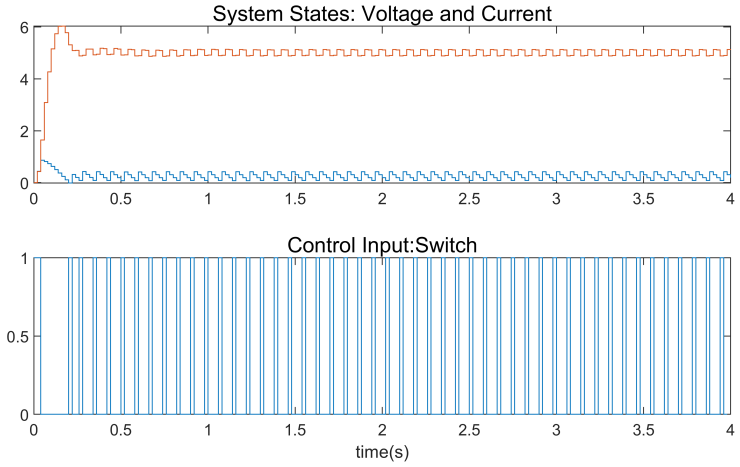

Figure 9: Constant 5 V tracking with $T_s = 0.02$ s and $N = 4$

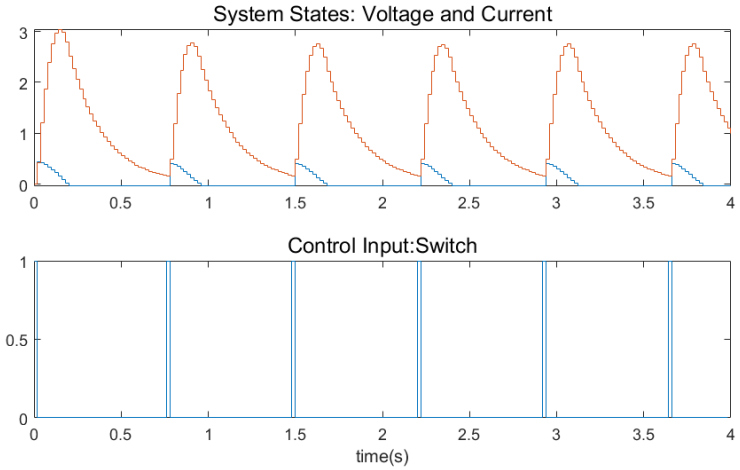

Figure 10: Constant 1 V tracking with $T_s = 0.02$ s and $N = 4$

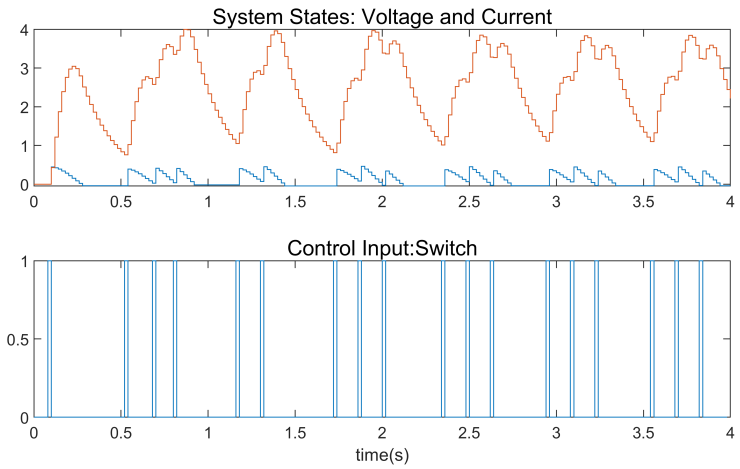

Figure 11: Time varying tracking with $T_s = 0.02$ s and $N = 4$

(iii) **[3 pts]** Play with the parameters N, the horizon length, x0, 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 $Ts = 0.005$ s given below:

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

Solution

There are five cases we need to compare, to make our comparison fair, we set the baseline as described below: System with sampling time $T_s = 0.02$ s, horizon $N = 4$, initial condition $x_0 = \begin{bmatrix} 0 & 0 \end{bmatrix}^{\mathrm{T}}$, Tracking reference 5 V, the cost function is the 2-norm quadratic function. The corresponding tracking result is shown below:

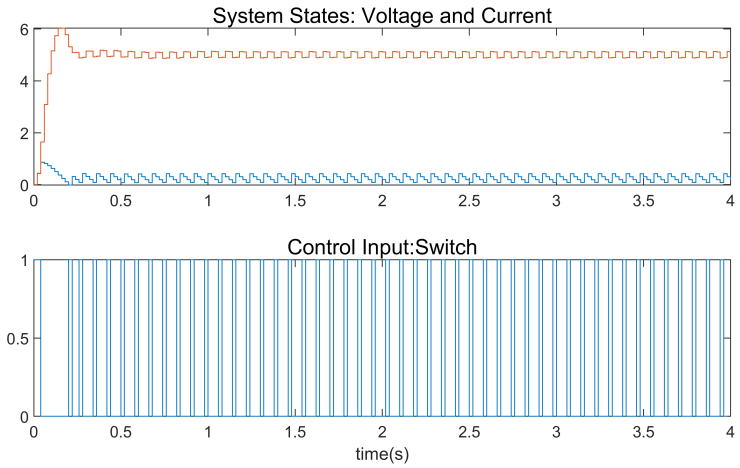

Figure 12: Baseline: 5 V reference tracking with $T_s = 0.02$ s $N = 4$, $x_0 = \begin{bmatrix} 0 & 0 \end{bmatrix}^{\mathrm{T}}$, 2-norm cost

(a) **Horizon**

In this comparison, we set the horizon from $N = 4$ to $N = 2$, the results are shown below. We observe that there are more oscillations, showing that by decreasing the horizon, the controller is not able to track the time varying reference correctly. A shorter horizon increases the error of tracking.

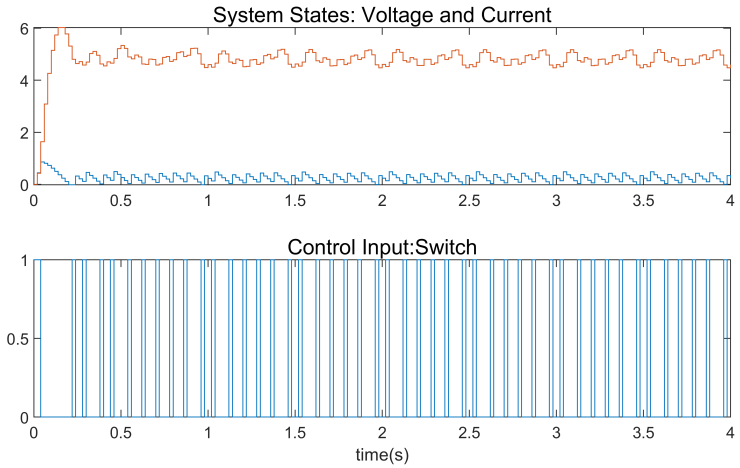

Figure 13: Horizon Comparison: 5 V reference tracking with $T_s = 0.02$ s $N = 2$, $x_0 = \begin{bmatrix} 0 & 0 \end{bmatrix}^{\mathrm{T}}$, 2-norm cost

(b) **Initial Condition**

In this comparison, we set the the initial conditon from $x_0 = \begin{bmatrix} 0 & 0 \end{bmatrix}^{\mathrm{T}}$ to $x_0 = \begin{bmatrix} 0.8 & 2.5 \end{bmatrix}^{\mathrm{T}}$, the results are shown below. We observe that it takes longer time for the tracking signal to converge (larger settling time) and there are silghtly more oscillations. However, the overall tracking effect is not largely influenced by the change of initial condition.

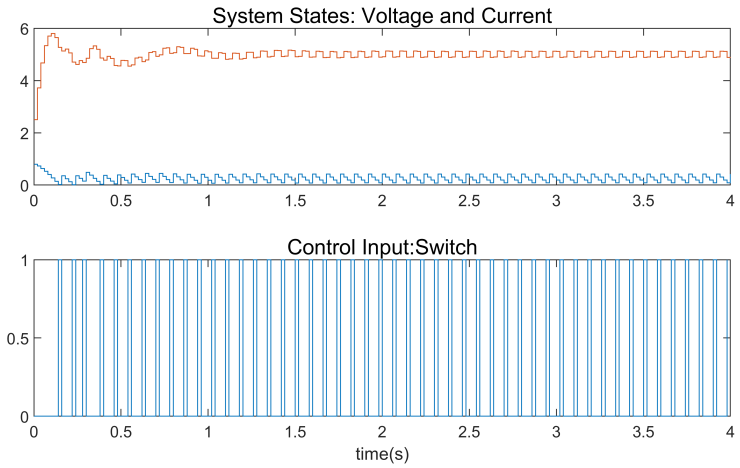

Figure 14: Initial Condition Comparison: 5 V reference tracking with $T_s = 0.02$ s $N = 2$, $x_0 = \begin{bmatrix} 0.8 & 2.5 \end{bmatrix}^{\mathrm{T}}$, 2-norm cost

(c) **Reference Signal**

In this comparison, we set the reference signal from the constant 5 V signal to the sinusoid reference signal similar to the one given above, but with different magnitude. The results are shown below. As is discussed before, for time varying signals, we can track it but due to sampling time limit, the signal is somewhat distorted and clearly the time invariant signal can be tracked better.

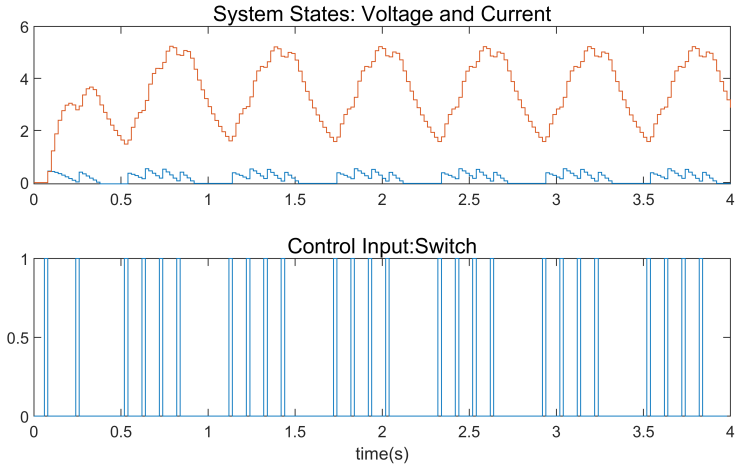

Figure 15: Reference siganl comparison: Time Varying (sinusoid) Tracking with $T_s = 0.02$ s $N = 4$, $x_0 = \begin{bmatrix} 0 & 0 \end{bmatrix}^{\mathrm{T}}$, 2-norm cost

(d) **Sampling Time**

In this comparison, we set the sampling time to be finer ($T_s = 0.005$ s) as suggested in the problem description. The results are shown below. It is clear that with finer sampling time, the final tracking result is much better. However, it can also be observed that the

overshoot increases with finer sampling time.

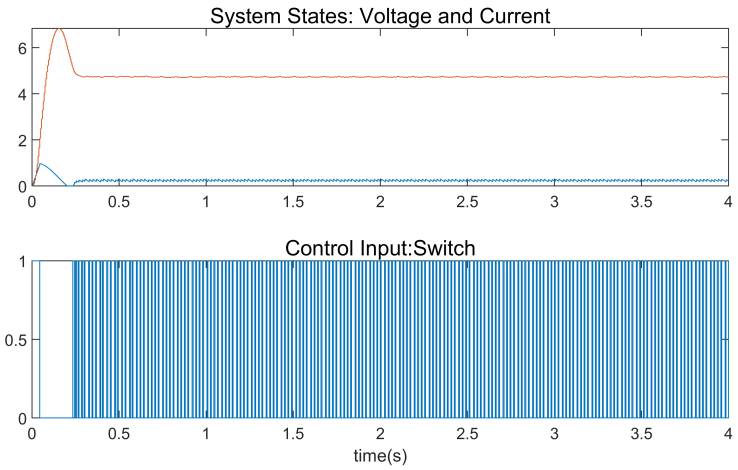

Figure 16: Samping time comparison: 5 V tracking with $T_s = 0.005$ s $N = 4$, $x_0 = \begin{bmatrix} 0 & 0 \end{bmatrix}^{\mathrm{T}}$, 2-norm cost

(e) **Objective Function**

In this comparison, we set the form of objective function from 2-norm to 1-norm. The results are shown below. In our comparison setting, the tracking results are nearly the same. But the computation time of using 1-norm cost is obviously longer. We also try to tune other parameters, the two types of costs would have different performaces in some cases and generally, 2-norm objective would have less overshoot. However, the difference is not significant.

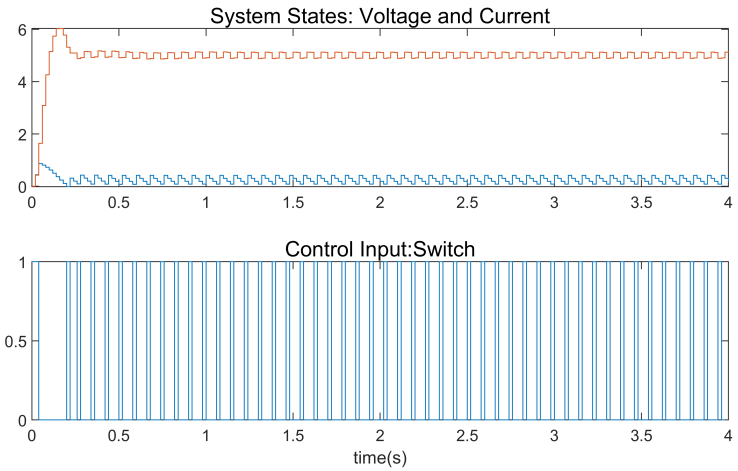

Figure 17: Objective comparison: 5 V tracking with $T_s = 0.005$ s $N = 4$, $x_0 = \begin{bmatrix} 0 & 0 \end{bmatrix}^{\mathrm{T}}$, 1-norm cost

For all the coding details, please refer to the live scripts.