SSY281 Model Predictive Control

Assignment 5

Yongzhao Chen

March 7, 2023

1 Linear MPC design

The system is:

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

The input and state constraints are

$$\begin{aligned} \mathcal{U} : -1 \leq u(k) \leq 1; \\ \mathcal{X} : \begin{bmatrix} -15 \\ -15 \end{bmatrix} \leq x(k) \leq \begin{bmatrix} 15 \\ 15 \end{bmatrix} \end{aligned} \tag{2}$$

1.1 a

To find the terminal weight P_f for the constrained receding horizon controller, several steps are taken. First, the system is defined using the provided matrices for A and B. Next, the input and state constraints are defined as polyhedrons $\mathcal{U}$ and $\mathcal{X}$, respectively.
The cost function is then set as $Q = I_2$ and $R = 100$. A terminal constraint $X_f = 0$ is defined using an equality constraint $Ax = b$, where $A = I_2$ and $b = 0$. The reachable set X_N is computed with a horizon length of $N = 4$ using the pre_operation function.
To obtain the LQR penalty, which includes the optimal P_f, the LQRPenalty function is used. P_f is then assigned to the terminal weight of the system through the terminalPenalty property of the LTISystem object.
Overall, the process involves defining the system, constraints, and cost function, computing the reachable set, and using the LQRPenalty function to obtain the optimal P_f value.
The result of P_f is:

$$P_f = \begin{bmatrix} 16.0929 & 32.9899 \\ 32.9899 & 93.9396 \end{bmatrix} \tag{3}$$

And the plot for X_N is:

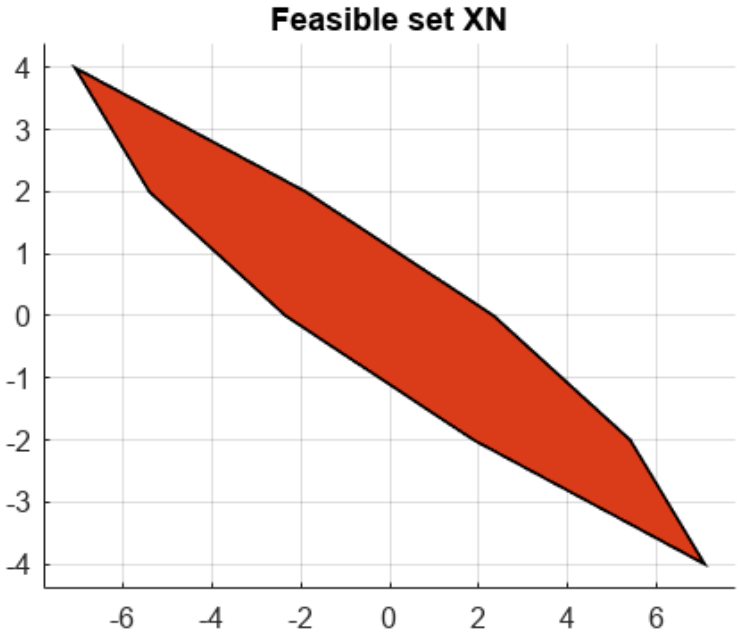

Figure 1: Feasible Set XN

1.2 b

The task is to design an MPC controller for the system described in Part (a) using prediction horizons of N = 10, 15, and 20, with an initial state of $x0 = [7; -4]$. The same system model and constraints as in Part (a) are used, and the controller's performance is evaluated by simulating the closed-loop system for Nsim = 40 time steps, which has been determined to be sufficient after some attempts.
The closed-loop system is simulated with the updated MPC controller for Nsim time steps using the simulate function and the initial state x0. The evaluate function is used to assess the open-loop performance of the controller, and the predicted state trajectories are stored in a cell array. Finally, figures are created for each value of N, and the predicted and actual state trajectories over time for each state variable are plotted.

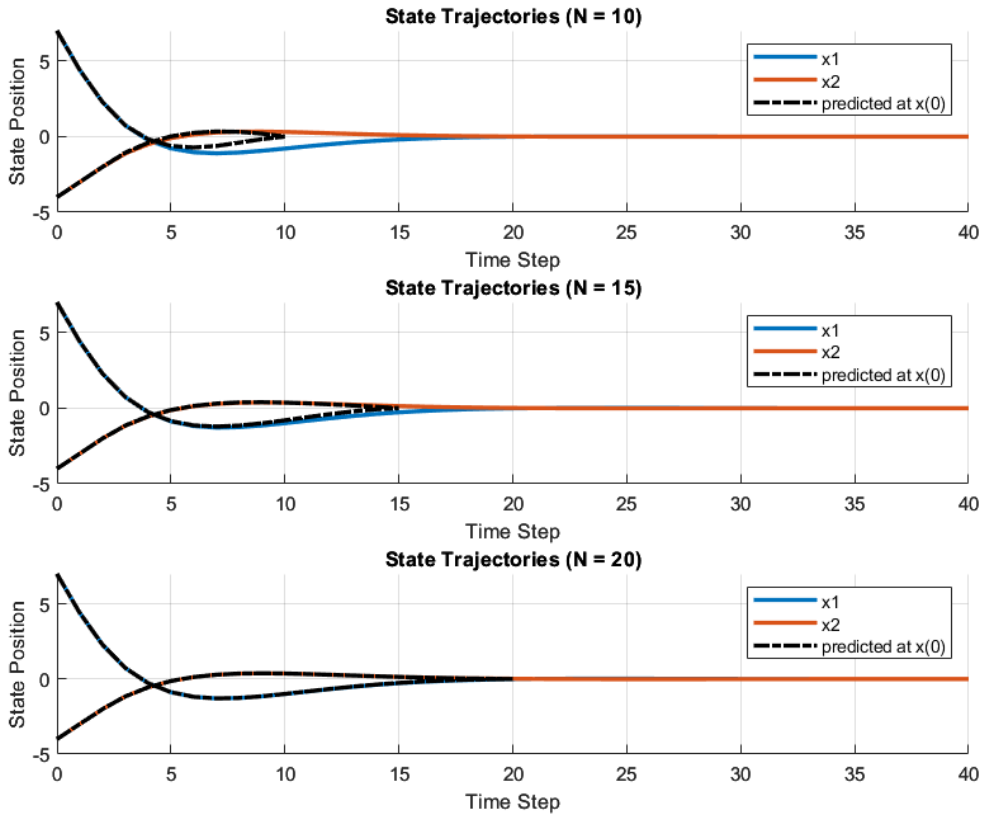

Figure 2: Prediction for different N

From the figure2, it shows that the controllers with small horizons length do not predict the trajectories well.This is because the MPC controller makes predictions based on a finite time horizon, which is limited by N. Therefore, when N is small, the controller can only consider a limited number of future time steps, and its ability to anticipate and react to future disturbances and system dynamics is limited

To conclusion, for choosing N, it is better to choose it larger, and the minimum value of N can set around the step when system reaches stable, which is around 20 in this quesiton.

1.3 c

Part (c) of the problem requires finding the explicit-MPC solution for the given system when N=20, the terminal constraint is $X_f = \{x : ||x||_\infty \leq 0.01\}$, and the terminal weight is Pf. To solve this problem, the constraints, cost function,

and terminal constraint and penalty are first defined as in parts (a) and (b) of the problem. Next, the MPC controller is defined with N=20, and the explicit MPC solution is computed using the toExplicit() method of the MPCController class.And the figure for state-space partitions is below as figure 3

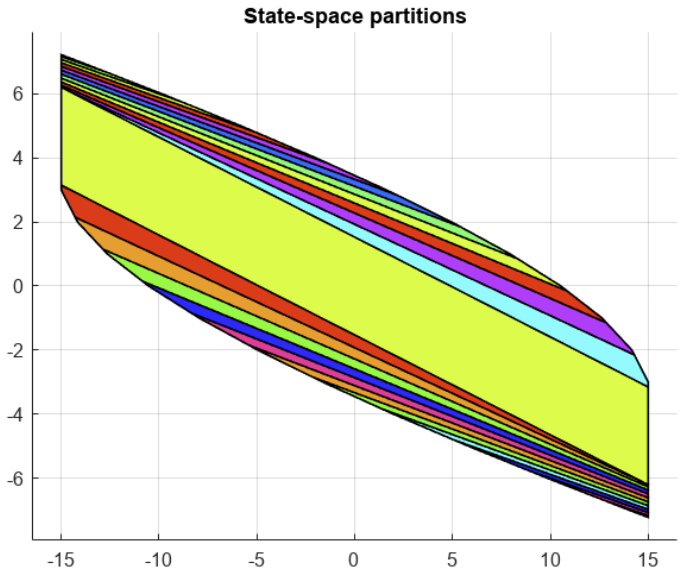

Figure 3: state-space partitions.

1.4 d

To guarantee persistent feasibility for all x_0 belonging to C_∞ when $N = 1$, a new X_f can be chosen. In the my code, the reachableSet function is used with N=1 and direction='forward' to compute a new X_f, denoted as Xf3. To ensure that the new terminal constraint satisfies the persistent feasibility condition, Xf3 is intersected with the invariant set C_∞. The resulting sets are visualized using the plot function, where X denotes the state constraints, C_∞ denotes the invariant set, and Xf3 denotes the reachable set of the invariant set. The resulting plot shows that the new terminal constraint intersects with the invariant set, indicating that all initial states belonging to C_∞ satisfy the persistent feasibility condition.

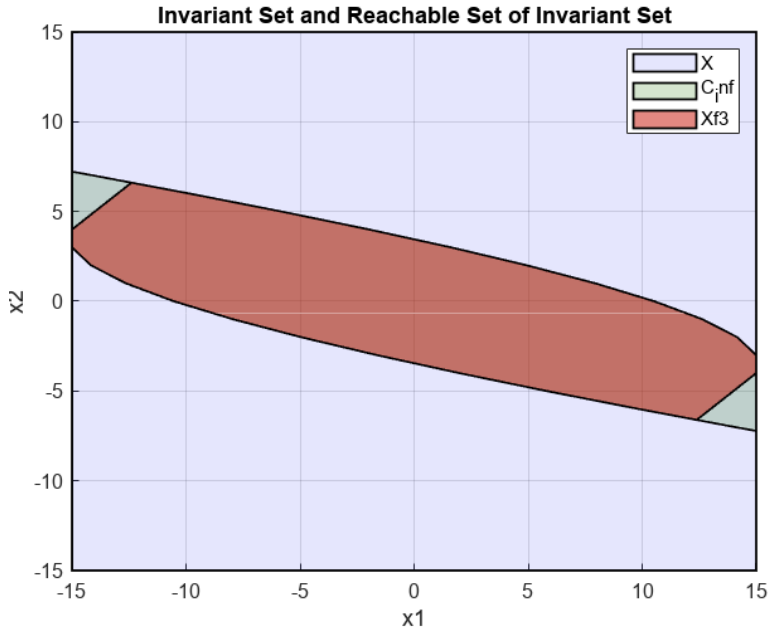

Figure 4: Invariant Set and Reachable Set of Invariant Set

2 Finite time control of a DC motor

The model is

$$\begin{aligned} x(k+1) &= Ax|(k) + BV(k) \\ T(k) &= Cx(k) \end{aligned} \tag{4}$$

x is define as $x = [\theta_L \ \dot{\theta}_L \ \theta_M \ \dot{\theta}_M]^\top$. System output is the torsional torque T and input is the DC voltage V . The DC voltage is constrained within the following range: $|V| \leq 200\,\mathrm{V}$.

2.1 a

Use *ss* function and *c2d* function can get the discreate model matrices easily:

$$
\begin{aligned}
Ad &= \begin{bmatrix} 0.7637 & 0.0873 & 0.0118 & 0.0003 \\ -4.4281 & 0.6764 & 0.2214 & 0.0086 \\ 0.4444 & 0.0159 & 0.9778 & 0.0621 \\ 7.1286 & 0.4285 & -0.3564 & 0.3449 \end{bmatrix} \\
Bd &= \begin{bmatrix} 0.0000 \\ 0.0003 \\ 0.0036 \\ 0.0621 \end{bmatrix} \\
Cd &= \begin{bmatrix} 1.2802 & 0 & -0.0640 & 0 \end{bmatrix}
\end{aligned} \tag{5}
$$

2.2 b

The discreate time matrices are already got in (a). Input constraints were defined based on the voltage limits of -200V to 200V, this condition is fullfilled by set umax and umin.
Terminal state constraints were set to bring the system to a standstill, i.e., $\dot{\theta}_L = \dot{\theta}_M = 0$. this is turned into Aeq and beq condition.

```
umin = -200;
umax = 200;
U = Polyhedron('lb', umin, 'ub', umax);
model.u.min = umin;
model.u.max = umax;
Xf = Polyhedron('Ae', [0 1 0 0; 0 0 0 1], 'be', [0;0]);
model.x.with('terminalSet');
model.x.terminalSet = Xf;
```

In my code, an MPC controller was created for each time step, and the input value for the current state was computed using the evaluate() function of the MPC controller. The next state was computed using the current state and input, and the states were stored in a matrix. The output trajectories were computed using the output matrix, and the plot function was used to plot the predicted system states and output.
The result for minimum time is 0.3 seconds.

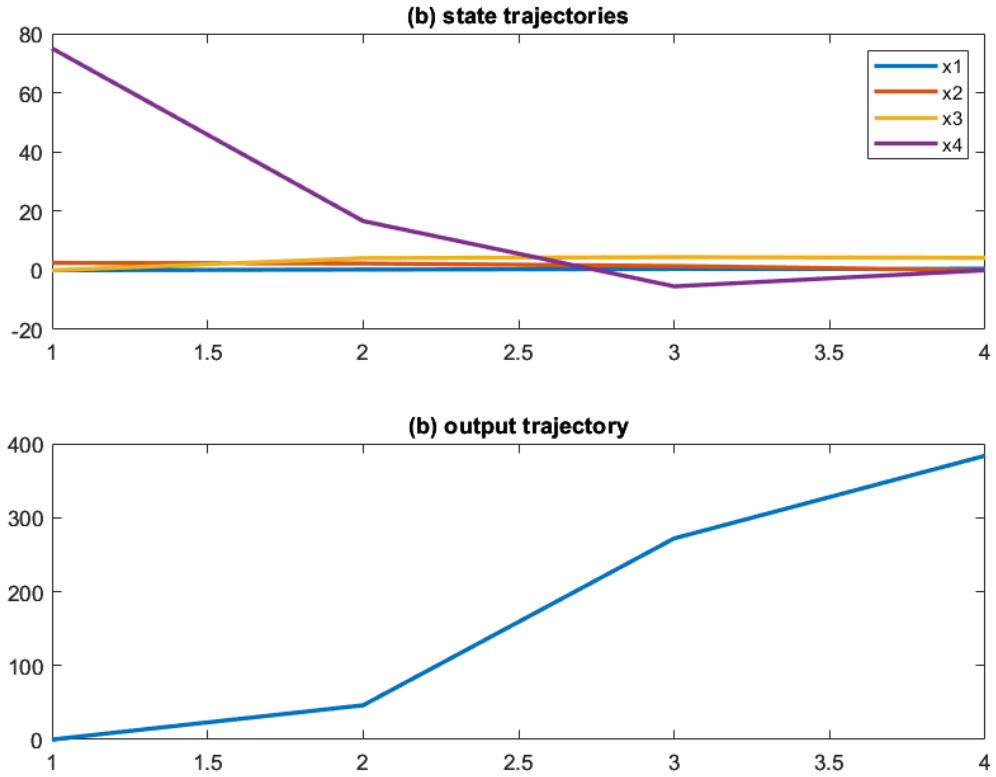

Figure 5: Predicted system states and output for b

2.3 c

The only diffrence of c with b is to add a constraint for Torque, which is the output.Then I use the code below to describe this constraint:

```
X = Polyhedron('A', [Cd; -Cd], 'b', [150; 150]);
```

And the result is that, the minimum time is 0.5s.

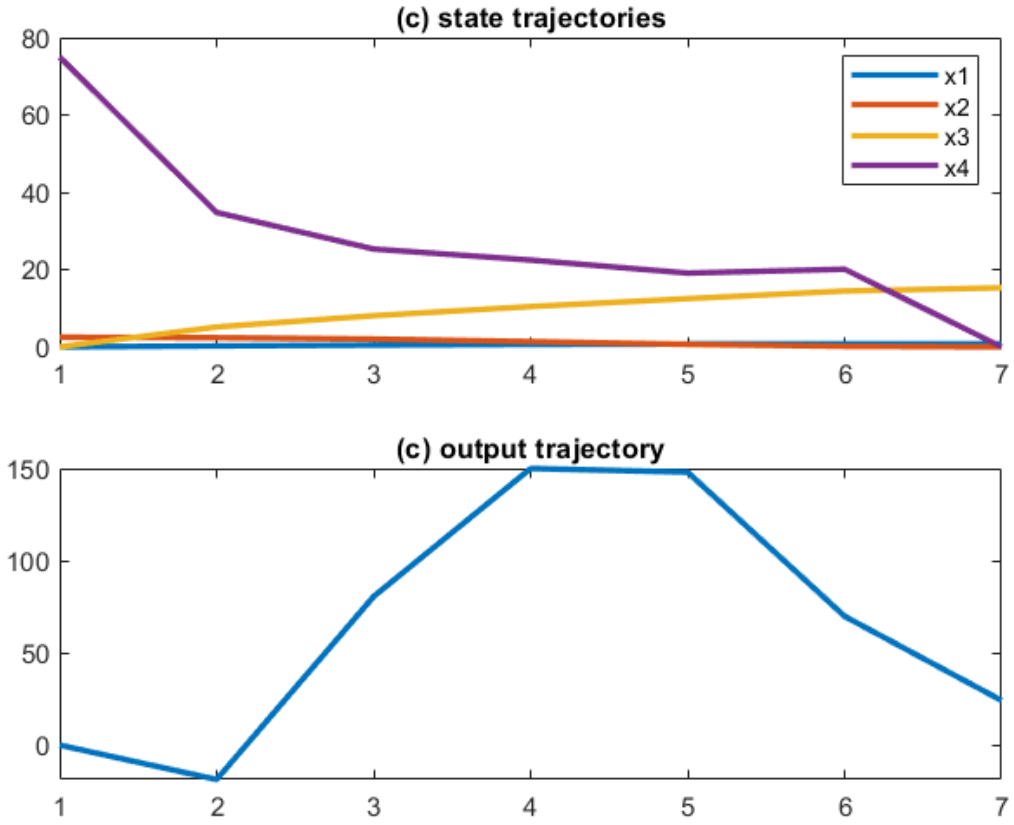

Figure 6: Predicted system states and output for C

2.4 d

I added constraint for Xtarget as:

```
set=[10;0.01;10;0.01];
Xtarget = Polyhedron('A',[eye(4); -eye(4)],'b',[lset;lset]);
```

And kept other code the same for c, then I can not reach the result even using 1000 steps to loop, I do not understand how to solve this part.