ESE 619 Model Predictive Control **Homework 4: MPC Theory I**
University of Pennsylvania, Spring 2023 updated April 7, 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. **Programming Exercise**
Consider the second-order system with sampling time $T = 0.1$s:

$$\begin{aligned} x(k+1) &= \begin{bmatrix} 0.7115 & -0.4345 \\ 0.4345 & 0.8853 \end{bmatrix} x(k) + \begin{bmatrix} 0.2173 \\ 0.0573 \end{bmatrix} u(k) \\ y(k) &= \begin{bmatrix} 0 & 1 \end{bmatrix} x(k) \end{aligned} \tag{1}$$

Your task is to develop an MPC controller that regulates the system to the origin while fulfilling the input constraint:

$$-5 \leq u(k) \leq 5, \quad \forall k$$

and in a second step you will also need to make sure that your controller fulfils the additional state constraint:

$$0 \leq x_2(k), \quad \forall k$$

Let us first refresh the definition of a quadratic programming based MPC controller for linear systems. The optimization problem used in an MPC controller is given by

$$\min_{u} \sum_{k=0}^{N-1} \left(x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k \right) + x_N^{\mathrm{T}} P x_N \tag{2a}$$

$$\text{s.t.} \quad u_{\min} \leq u_k \leq u_{\max} \tag{2b}$$

$$x_{k+1} = A x_k + B u_k \tag{2c}$$

$$x_{\min} \leq x_k \leq x_{\max} \tag{2d}$$

where $x_0 = x(k)$ and the matrices A and B are as defined in (1). In this exercise, the matrices Q, R and P are treated as design variables. To simplify the implementation, it helps to introduce a vectorized notation.

$$\begin{aligned} \mathbf{x} &= \begin{bmatrix} x_0^{\mathrm{T}} & x_1^{\mathrm{T}} & x_2^{\mathrm{T}} & \dots & x_n^{\mathrm{T}} \end{bmatrix} \\ \mathbf{u} &= \begin{bmatrix} u_0^{\mathrm{T}} & u_1^{\mathrm{T}} & \dots & u_n^{\mathrm{T}} \end{bmatrix} \end{aligned}$$

With these definitions, we can write the state sequence as:

$$\mathbf{x} = \mathcal{A} x_0 + \mathcal{B} \mathbf{u}$$

where

$$\mathcal{A} = \begin{bmatrix} I \\ A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}, \quad \mathcal{B} = \begin{bmatrix} 0 & \dots & \dots & 0 \\ B & 0 & \dots & 0 \\ AB & B & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ A^{N-1}B & \dots & AB & B \end{bmatrix}$$

By introducing block-diagonal matrices $\mathcal{Q}$ and $\mathcal{R}$, the quadratic cost can be written as $\mathbf{x}^{\mathrm{T}} \mathcal{Q} \mathbf{x} + \mathbf{u}^{\mathrm{T}} \mathcal{R} \mathbf{u}$

1. **[2 pt]** What do $\mathcal{Q}$ and $\mathcal{R}$ look like?
Solution:

$$\mathcal{Q} = \text{blockdiag}(\underbrace{Q, \dots, Q}_{N \text{ times}}, P), \quad \mathcal{R} = \text{blockdiag}(\underbrace{R, \dots, R}_{N \text{ times}})$$

2. **[4 pt]** The first goal in this exercise is to write the whole MPC problem in a format that can be sent to a quadratic programming solver (such as `quadprog`). To do this, expand the expression for the optimization cost $\mathbf{x}^{\mathrm{T}}\mathcal{Q}\mathbf{x} + \mathbf{u}^{\mathrm{T}}\mathcal{R}\mathbf{u}$ into the format $\frac{1}{2}\mathbf{u}^{\mathrm{T}}H\mathbf{u} + x_0^{\mathrm{T}}F\mathbf{u} + \frac{1}{2}x_0^{\mathrm{T}}Yx_0$. Derive the matrices H, F and Y interms of Q, $\mathcal{A}$, $\mathcal{B}$, $\mathcal{Q}$ and $\mathcal{R}$. Neglect the state constraints $x_{\min}$ and $x_{\max}$ for now.
Solution:
$$H = 2\left(\mathcal{B}^{\mathrm{T}}\mathcal{Q}\mathcal{B} + \mathcal{R}\right), \quad F = 2\mathcal{A}^{\mathrm{T}}\mathcal{Q}\mathcal{B}, \quad Y = 2\mathcal{A}^{\mathrm{T}}\mathcal{Q}\mathcal{A}$$

3. **[2 pt]** Is the term $\frac{1}{2}x_0^{\mathrm{T}}Yx_0$ needed to solve the optimization problem? Why or why not?
Solution:
The term $\frac{1}{2}x_0^{\mathrm{T}}Yx_0$ is not needed to solve the optimization problem since we the optimization variables are $\mathbf{u}$ and this term does not contain it. We can regard it as a constant term when solving optimization and hence its only influence is on the optimal value rather than the optimizer.

4. **[12 pt]** The next exercise requires some advanced MATLAB programming. Write a MATLAB function `u = mympc(A,B,Q,R,P,N,umin,umax,xmin,xmax,x)`, i.e. a function which takes an arbitrary system (you may assume single input-single output), cost matrices of suitable dimensions, a desired prediction horizon, control constraints, state constraints (boxconstraints), the current state, and returns the optimal control input. The MATLAB hints at the end should be enough to get you through the tricky parts. For your and the assistant's sanity, try to structure your code by using sub-functions or similar concepts. For the next couple of questions we will neglect the state constraints. (you can set them to $x_{\min} = \begin{bmatrix}-\infty & -\infty\end{bmatrix}^{\mathrm{T}}$ and $x_{\max} = \begin{bmatrix}\infty & \infty\end{bmatrix}^{\mathrm{T}}$)
Solution: See the MATLAB live script for the details

5. **[2 pt]** Calculate a terminal state weight-matrix P_L by solving the discrete Lyapunov equation (`help dlyap`) and an alternative weight-matrix P_R by solving the associated LQR problem (`help dlqr`) (please read the associated pages in the script now). What does the choice of P_L or P_R for the terminal weight P imply for $u_k, k \geq N$. Which of the two matrices has the larger 2-norm (i.e. largest maximum singular value)? Why?
Solution: For codes, See the MATLAB live script for the details
According to the calculation, we have:
$$P_L = \begin{bmatrix} 5.5319 & -0.9995 \\ -0.9995 & 5.8982 \end{bmatrix}, P_R = \begin{bmatrix} 3.8097 & 0.9640 \\ 0.9640 & 4.5173 \end{bmatrix}$$
Using P_L assumes that the system is inherently stable without any control input while using P_R assumes that no constraints will be active for $k \geq N$. The 2-norm of these two matrices is shown below:
$$\|P_L\|_2 = 6.7312, \|P_R\|_2 = 5.1904$$
We can see that P_L has the larger 2-norm (i.e. largest maximum singular value). This can be explained from the assumptions of setting these two matrices. Assuming that the system is stable (P_L) is stricter than assuming the constraint not to be active (P_R). Hence using would P_L would result in driving the state to original more aggresively, thus have larger 2-norm.

6. **[6 pt]** You are now supposed to actually use your MPC controller. Start with, e.g. $Q = I$, $R = 1$ and $N = 5$ and your favourite terminal state weight. Write a simple loop to simulate the (discrete-time) closed-loop behavior from the initial condition $\begin{bmatrix}0 & 10\end{bmatrix}^{\mathrm{T}}$. Plot both the states and the optimal input on the same figure for the first 5 s of the simulation. Are the results satisfactory? Is the prediction horizon reasonable? Try to tune the MPC controller to reduce the oscillations and the over-shoot.
Solution: The plot is shown here, see more coding details in live scripts

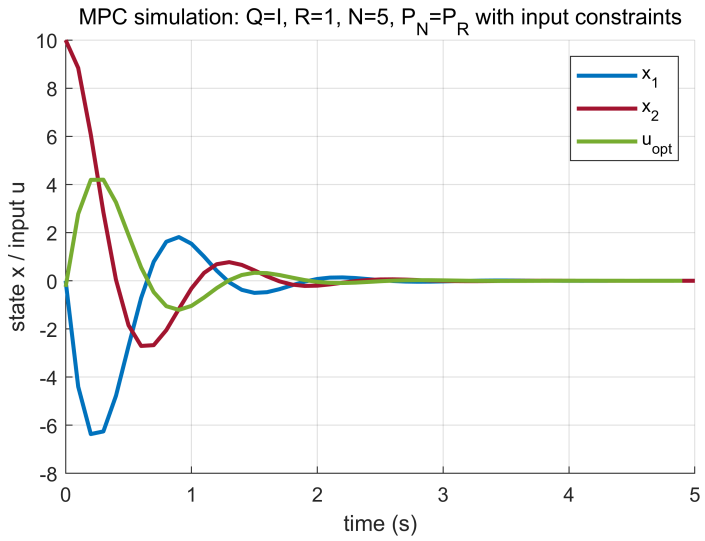

Figure 1: MPC closed loop simulation: $Q = I, R = 1, N = 5, P_N = P_R$

For me, the results as well as the prediction horizon are satisfactory, because the input constraints are not violated and the system is stable and goes to the original in the end.
In order to reduce the oscillations and the overshoot, we can simply tune the state cost gain matrix, for example, if we set $Q = 100I$, the results are shown below:

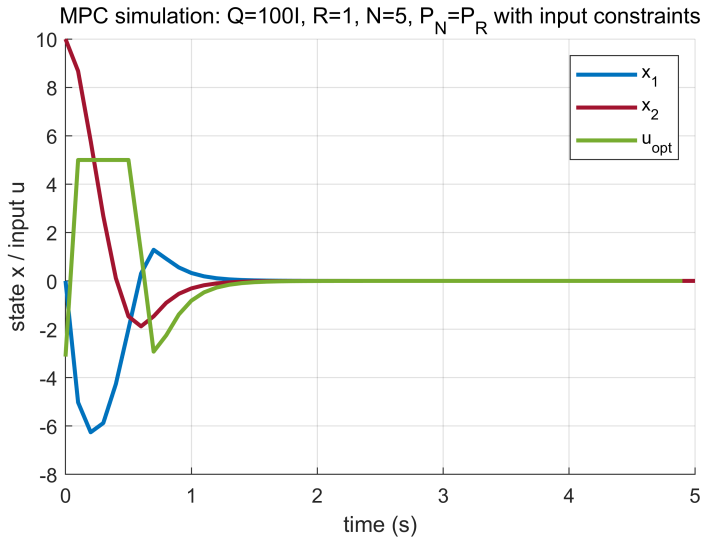

Figure 2: MPC closed loop simulation: $Q = 100I, R = 1, N = 5, P_N = P_R$

We can see that the oscillations in both state is effectively reduced and the overshoot is slightly reduced, the other design parameters can also be further tuned to get more optimizations.

7. **[4 pt]** Reduce your prediction horizon to $N = 1$ and check if there is any major degradation in performance. Now keep the short prediction horizon but change the terminal state weight to Q, i.e. $P = Q$. How does the MPC controller perform with a (too) short horizon and this choice of the terminal state weight?
Solution: The plot is shown here, see coding more details in live scripts

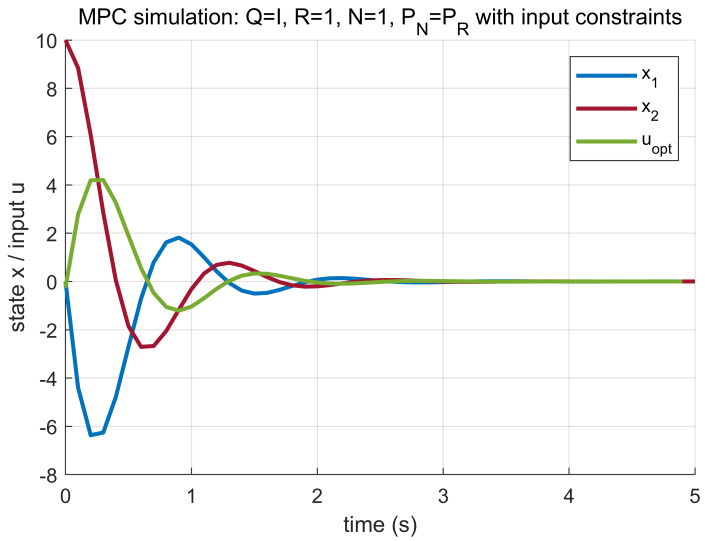

Figure 3: MPC closed loop simulation: $Q = I, R = 1, N = 1, P_N = P_R$

No major degradation in performance can be observed in this case. It is because from the simulation above (i.e. $Q = I, R = 1, N = 5, P_N = P_R$), we know that no constraints would be active during the whole simulation time. Therefore, that mpc controller is actually equivalent to the steady state LQR. So even if we shortened the horzion, as long as we use the LQR solution P_R as the terminal cost, since this cost perfectly represent the situations after the horzion, it would produce the same results and the controller would have the same performance.
If we set the terminal cost to be Q, the results are shown below:

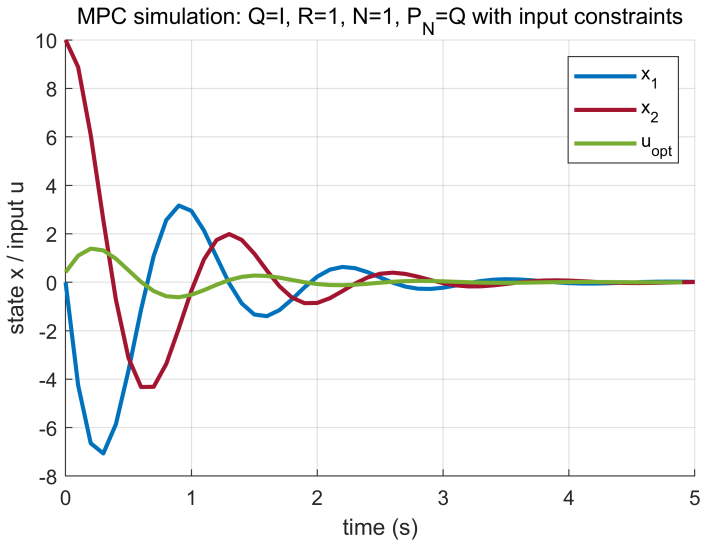

Figure 4: MPC closed loop simulation: $Q = I, R = 1, N = 1, P_N = Q$

We can see that the system overshoot more and has more oscillations now. This is because now we have a too short-sighted horzion and the terminal cost we chose can not accurately mimic the situations after the horzion, only if we set the horizon to be long enough or we set the terminal cost to be P_R can we have the same performance.

8. **[4 pt]** Compute the optimal input for the state $\begin{bmatrix} 0 & 10 \end{bmatrix}^{\mathrm{T}}$ using the LQR based terminal state weight, $Q = 100I, R = 1$ and $N = 2$ (make sure to use the same weights in the MPC controller and in the computation of P_R and the associated LQR controller). Compare the optimal control input to the control input that an LQR controller would give at the same state. Do the same thing when you use the terminal state weight based on the zero input assumption (i.e. the terminal

state weight computed using the Lyapunov equation). Now change the state to $\begin{bmatrix}0.1 & 0.1\end{bmatrix}^{\mathrm{T}}$ and repeat the procedure. Observations?

Solution:

The optimal control input when $Q = 100I, R = 1$ and $N = 2$ for the state $\begin{bmatrix}0 & 10\end{bmatrix}^{\mathrm{T}}$ is:

$$u_{\mathrm{mpc},P_R} = -5 \quad u_{\mathrm{mpc},P_L} = -3.0946 \quad u_{\mathrm{LQR}} = -13.2966$$

The control input comparison at each timestep is shown below:

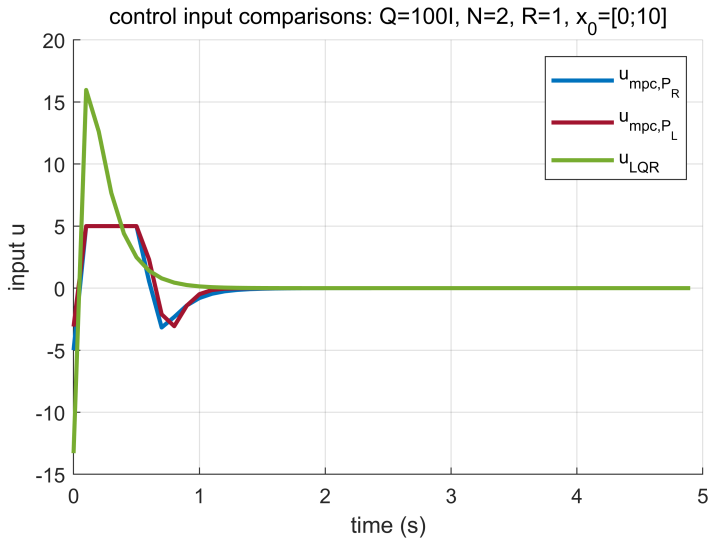

Figure 5: MPC control input comparisons: $Q = 100I, R = 1, N = 2, x_0 = \begin{bmatrix}0 & 10\end{bmatrix}^{\mathrm{T}}$

The optimal control input when $Q = 100I, R = 1$ and $N = 2$ for the state $\begin{bmatrix}0.1 & 0.1\end{bmatrix}^{\mathrm{T}}$ is:

$$u_{\mathrm{mpc},P_R} = -0.5024 \quad u_{\mathrm{mpc},P_L} = -0.7538 \quad u_{\mathrm{LQR}} = -0.5024$$

The control input comparison at each timestep is shown below:

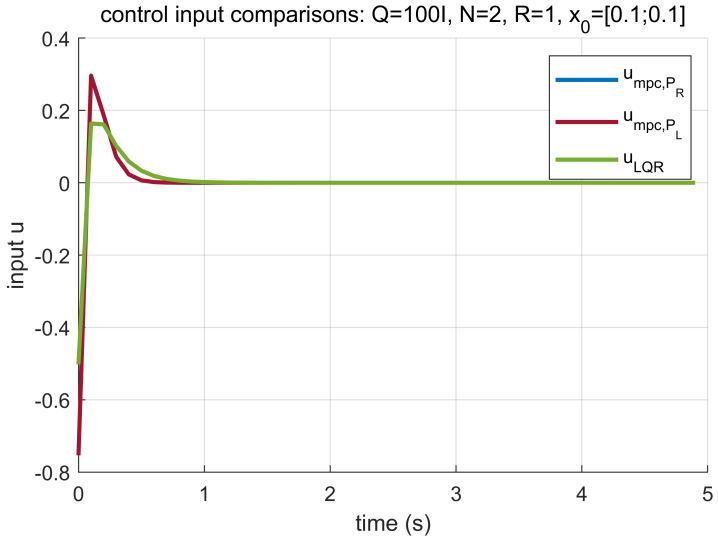

Figure 6: MPC control input comparisons: $Q = 100I, R = 1, N = 2, x_0 = \begin{bmatrix}0.1 & 0.1\end{bmatrix}^{\mathrm{T}}$

9. **[4 pt]** For the last task of the programming exercise you will need to implement state constraints. For `quadprog` you need to bring the state constraints into the form $A_{\mathrm{ineq}}\mathbf{u} \leq \mathbf{b}$. Hint: See slide

28 of MPC chapter 6.

Solution: The box constraints can be written in the form:

$$\mathcal{X} = \{x \mid A_x x \leq b_x\} \quad \mathcal{U} = \{u \mid A_u u \leq b_u\} \quad \mathcal{X}_f = \{x \mid A_f x \leq b_f\}$$

Where:

$$A_x = A_f = \begin{bmatrix} I_n \\ -I_n \end{bmatrix}, \quad b_x = b_f = \begin{bmatrix} x_{\max} \\ -x_{\min} \end{bmatrix} \quad A_u = \begin{bmatrix} I_m \\ -I_m \end{bmatrix}, \quad b_u = \begin{bmatrix} u_{\max} \\ -u_{\min} \end{bmatrix}$$

and m, n are the state and input dimensions. The constraints can further be rearranged as the form of $G_0 U_0 \leq w_0 + E_0 x\,(0)$, where:

$$G_0 = \begin{bmatrix} \underbrace{\begin{matrix} A_u & 0 & \dots & 0 \\ 0 & A_u & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & A_u \end{matrix}}_{\mathcal{U}} \\ \underbrace{\begin{matrix} 0 & 0 & \dots & 0 \\ A_x B & 0 & \dots & 0 \\ A_x A B & A_x B & \ddots & 0 \\ \vdots & \vdots & \dots & \vdots \\ A_x A^{N-2} B & A_x A^{N-3} B & \dots & 0 \end{matrix}}_{\mathcal{X}} \\ \underbrace{\begin{matrix} A_f A^{N-1} B & A_f A^{N-2} B & \dots & A_f B \end{matrix}}_{\mathcal{X}_f} \end{bmatrix}, \quad E_0 = \begin{bmatrix} \underbrace{\begin{matrix} 0 \\ 0 \\ \vdots \\ 0 \end{matrix}}_{\mathcal{U}} \\ \underbrace{\begin{matrix} -A_x \\ -A_x A \\ -A_x A^2 \\ \vdots \\ -A_x A^{N-1} \end{matrix}}_{\mathcal{X}} \\ \underbrace{\begin{matrix} -A_f A^N \end{matrix}}_{\mathcal{X}_f} \end{bmatrix}, \quad w_0 = \begin{bmatrix} \underbrace{\begin{matrix} b_u \\ b_u \\ \vdots \\ b_u \end{matrix}}_{\mathcal{U}} \\ \underbrace{\begin{matrix} b_x \\ b_x \\ b_x \\ \vdots \\ b_x \end{matrix}}_{\mathcal{X}} \\ \underbrace{\begin{matrix} b_f \end{matrix}}_{\mathcal{X}_f} \end{bmatrix}$$

see more coding details in live scripts

10. **[4 pt]** Take the parameters from task 6. Consider as an additional constraint, that there should be no undershoot in x_2 and start from an initial condition $\begin{bmatrix} 0 & 6 \end{bmatrix}^{\mathrm{T}}$. What happens if you start from an initial condition $\begin{bmatrix} 0 & 10 \end{bmatrix}^{\mathrm{T}}$

Solution: For initial condition of $\begin{bmatrix} 0 & 6 \end{bmatrix}^{\mathrm{T}}$ The simulation results are shown below:

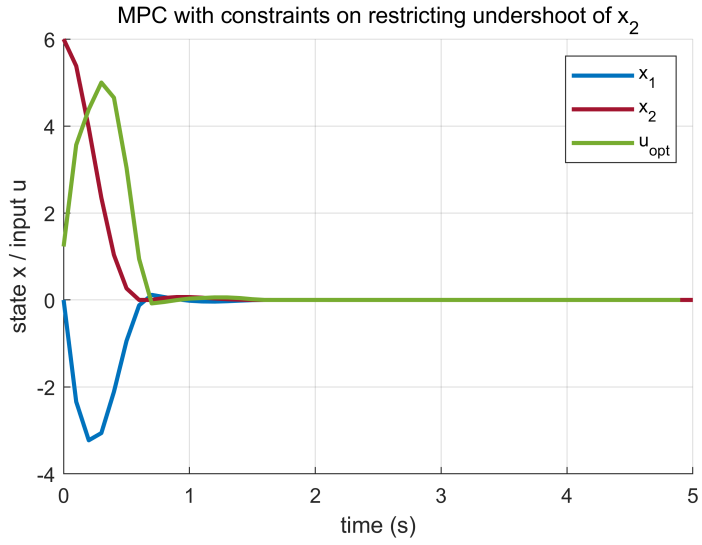

Figure 7: MPC closed loop simulation with constraints on restricting undershoot of x_2

For initial condition of $\begin{bmatrix} 0 & 10 \end{bmatrix}^{\mathrm{T}}$, the problem is infeasible and no control input satisfying the constraints can be found.

11. **[8 pt]** We want to prove asymptotic stability of the MPC control law generated by (2). For simplicity, we assume there exists a terminal set $\mathcal{X}_f$ and constraint $x_N \in \mathcal{X}_f$. Additionally, assume there exist a linear state feedback controller K, such that $v(x_N) = Kx_N$ (see slide 37 of MPC chapter 7).

(i) State the implicit conditions on $\mathcal{X}_f, P, Q, R$ and K, so that we can prove asymptotic stability in this general setting.
Solution: The implicit conditions on $\mathcal{X}_f, P, Q, R$ and K is:

A. The stage cost is positive definite, i.e. $Q \succeq 0, R \succ 0$

B. The terminal set $\mathcal{X}_f$ is invariant under the local control law $v(x_k)$:

$$x_{k+1} = Ax_k + Bv(x_k) \in \mathcal{X}_f \text{ for all } x_k \in \mathcal{X}_f$$

and all state and input constraints are satisfied in $\mathcal{X}_f$:

$$\mathcal{X}_f \subseteq \mathcal{X}, v(x_k) \in \mathcal{U} \text{ for all } x_k \in \mathcal{X}_f$$

C. The terminal cost is a continuous Lyapunov function in the terminal set $\mathcal{X}_f$ and satisfies:

$$p(x_{k+1}) - p(x_k) \leq -q(x_k, v(x_k)) \text{ for all } x_k \in \mathcal{X}_f$$

That is to say:

$$P \succ 0, x_{N+1}^{\mathrm{T}} P x_{N+1} - x_N^{\mathrm{T}} P x_N \leq -\left[x_N^{\mathrm{T}} Q x_N + v(x_N)^{\mathrm{T}} R v(x_N)\right]$$

(ii) Show that under these assumptions, $J_0^*(x_0)$ is a Lyapunov function decreasing along the closed loop trajectories.
Proof:
(Feasibility)
Assume feasibility of x_0 and let $\{u_0^*, u_0^*, \ldots, u_{N-1}^*\}$ be the optimal control sequence computed at x_0 and $\{x(0), x_1, \ldots, x_N\}$ be the corresponding state trajectory that lead the state to the terminal region.
At $x(1)$, the control sequence be $\{u_1^*, u_2^*, \ldots, u_{N-1}^*, v(x_N)\}$ would be feasible. (apply until u_{N-1}^* would bring the state to $\mathcal{X}_f$, and based on our assumption, local control law $v(x_k)$ is feasible and would keep the state in $\mathcal{X}_f$). Therefore, we prove the recursive feasibility.
(Stability)
Consider:

$$J_0^*(x_0) = p(x_N) + \sum_{i=0}^{N-1} q(x_i, u_i^*)$$

Apply the feasible, sub-optimal sequence $\{u_1^*, u_2^*, \ldots, u_{N-1}^*, v(x_N)\}$ and hence we have:

$$\begin{aligned} J_0^*(x_1) \leq \tilde{J}_0(x_1) &= \sum_{i=1}^{N} q(x_i, u_i^*) + p(Ax_N + Bv(x_N)) \\ &= \sum_{i=0}^{N-1} q(x_i, u_i^*) + p(x_N) - q(x_0, u_0^*) - p(x_N) + q(x_N, v(x_N)) + p(Ax_N + Bv(x_N)) \\ &= J_0^*(x_0) - \underbrace{q(x_0, u_0^*)}_{>0} + \underbrace{[p(Ax_N + Bv(x_N)) - p(x_N) + q(x_N, v(x_N))]}_{\leq 0 \quad \because \text{assumption } p(x_{k+1}) - p(x_k) \leq -q(x_k, v(x_k))} \end{aligned}$$

i.e.:

$$J_0^*(x_1) \leq J_0^*(x_0) - q(x_0, u_0^*) + p, q(x_0, u_0^*) > 0, p < 0 \Rightarrow J_0^*(x_1) < J_0^*(x_0)$$

Hence we prove that $J_0^*(x)$ is a Lyapunov function decreasing along the closed loop trajectories and The closed-loop system under the MPC control law is asymptotically stable.

12. **[8 pt]** For the last task, we consider the system:

$$x(k+1) = \begin{bmatrix} \cos(0.5) & -\sin(0.5) \\ \sin(0.5) & \cos(0.5) \end{bmatrix} x(k) + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} u(k) \tag{3}$$

with inputs constraints of the from

$$\mathcal{U} = \left\{ u = \left(u^{(1)}, u^{(2)} \right) \,\middle|\, -5 \leq u^{(i)} \leq 5, i = 1, 2 \right\}$$

The goal of this exercise is to design a state feedback controller $u = Kx$, such that the set

$$\mathcal{X}_f = \left\{ x = \left(x^{(1)}, x^{(2)} \right) \,\middle|\, -10 \leq x^{(i)} \leq 10, i = 1, 2 \right\}$$

is an invariant set of the closed loop system under the given actuator constraints. Please compute all values of K, which satisfy the given constraints.

Hint 1: You may restrict your attention to diagonal feedback matrix of the from,

$$K = \begin{bmatrix} k_1 & 0 \\ 0 & k_2 \end{bmatrix}$$

Hint 2: Argue why it is sufficient to only check the corner points of the set $\mathcal{X}_f$

Solution:

Remark:

First of all, we can see that we have the linear system and the control law is also linear, so the resulting closed-loop system is also linear. From the perpective of linear transformation, this transformation essentially is sequentially doing rotation, scaling, and relfection (consider singular value decomposition).

Therefore, given a box constraint, i.e. rectangle, the linear transformation would result in a parallelogram, with each vertices corresponding to original vertices in rectangle. And we know that the bounds of the parallelogram can also be decided by the vertices (corner points). i.e. we only need to check the corner points.

Problem Formulation:

According to the problem description, the state feedback controller $u(x) = Kx : \mathcal{X} \to \mathcal{U}$ should give an invaraint set $\mathcal{X}_f$ while satisfying the input constraints, i.e.

$$\begin{aligned} x^+ = f_{\text{cl}}(x, u(x)) \in \mathcal{X}_f, \quad & \forall x \in \mathcal{X}_f \\ u(x) \in \mathcal{U}, \quad & \forall x \in \mathcal{X}_f \end{aligned}$$

According to the remarks on box constriants and problem formulations above, we have:

$$(A+BK) \begin{bmatrix} 10 \\ 10 \end{bmatrix} \in \mathcal{X}_f \Rightarrow -1.3982 \leq k_1 \leq 0.6018, \quad -2.3570 \leq k_2 \leq -0.3570$$

$$(A+BK) \begin{bmatrix} 10 \\ -10 \end{bmatrix} \in \mathcal{X}_f \Rightarrow -2.3570 \leq k_1 \leq -0.3570, \quad -1.3982 \leq k_2 \leq 0.6018$$

$$(A+BK) \begin{bmatrix} -10 \\ 10 \end{bmatrix} \in \mathcal{X}_f \Rightarrow -2.3570 \leq k_1 \leq -0.3570, \quad -1.3982 \leq k_2 \leq 0.6018$$

$$(A+BK) \begin{bmatrix} -10 \\ -10 \end{bmatrix} \in \mathcal{X}_f \Rightarrow -1.3982 \leq k_1 \leq 0.6018, \quad -2.3570 \leq k_2 \leq -0.3570$$

$$K \begin{bmatrix} 10 \\ 10 \end{bmatrix} \in \mathcal{U} \Rightarrow -0.5 \leq k_1 \leq 0.5, -0.5 \leq k_2 \leq 0.5$$

$$K \begin{bmatrix} 10 \\ -10 \end{bmatrix} \in \mathcal{U} \Rightarrow -0.5 \leq k_1 \leq 0.5, -0.5 \leq k_2 \leq 0.5$$

$$K \begin{bmatrix} -10 \\ 10 \end{bmatrix} \in \mathcal{U} \Rightarrow -0.5 \leq k_1 \leq 0.5, -0.5 \leq k_2 \leq 0.5$$

$$K \begin{bmatrix} -10 \\ -10 \end{bmatrix} \in \mathcal{U} \Rightarrow -0.5 \leq k_1 \leq 0.5, -0.5 \leq k_2 \leq 0.5$$

Taking the intesection of all the above result we get:

$$(k_1, k_2) \in [-0.5, -0.3570] \times [-0.5, -0.3570]$$

In fact, with the help of MPT toolbox, we can calculate this problem is a more convenient manner, see the live scripts for more detials.

2. **Batch Approach**

Consider the double integrator:

$$y(t) = \frac{1}{s^2} u(t) \tag{4}$$

and its equivalent discrete-time state-space representation

$$\begin{cases} x(t+1) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \\ y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} x(t) \end{cases} \tag{5}$$

obtained by setting $\ddot{y}(t) = \frac{\dot{y}(t+T)-\dot{y}(t)}{T}, \dot{y}(t) = \frac{y(t+T)-y(t)}{T}, T = 1$ s Define the following cost function

$$J(U_0, x(0)) \triangleq x_N^{\mathrm{T}} P x_N + \sum_{k=0}^{N-1} x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k \tag{6}$$

$N = 5, Q = I_2, R = 0.1, P = I_2$.. The constraints are

$$-1 \leq u(k) \leq 1 \quad k = 0, 1, \cdots 5 \tag{7a}$$

$$\begin{bmatrix} -15 \\ -15 \end{bmatrix} \leq x(k) \leq \begin{bmatrix} 15 \\ 15 \end{bmatrix} \quad k = 0, 1, \cdots 5 \tag{7b}$$

1. **[14 pt]** Select 100 initial conditions x_0 (by gridding the space of initial conditions with 10×10 points) and compute the solution to the finite time optimization control problem for each x_0 ($\in$ 7b) using `quadprog` and the batch approach. Plot $u_0^*(x_0)$ at the grid points.
   The plot is shown below, if the problem is infeasible, the control input value is set to NAN and the corresponding grid is a blank. 10×10 grid is somewhat too sparse to shown the distribution and boundary of the solution, so I also plot the 50×50 grid solution:

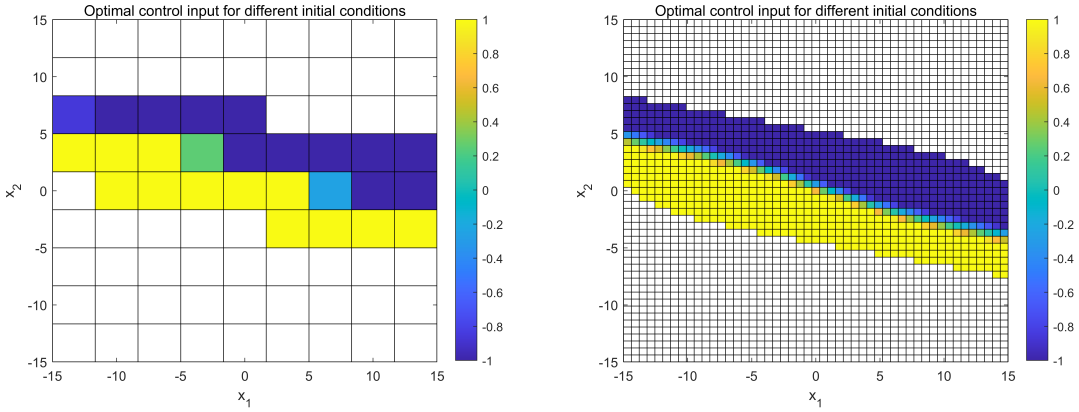

Figure 8: Finite time optimization control input for different initial conditions

See the live script for coding detials.

2. **[6 pt]** Report the solution $[u_0^*; u_1^*; u_2^*; u_3^*; u_4^*]$ for $x_0 = \begin{bmatrix} 1 & 1 \end{bmatrix}^{\mathrm{T}}$ and $x_0 = \begin{bmatrix} -1 & -1 \end{bmatrix}^{\mathrm{T}}$
   According to the calculation, the optimal control input for these two cases is:

$$x_0 = \begin{bmatrix} 1 & 1 \end{bmatrix}^{\mathrm{T}}: \qquad U_0 = \begin{bmatrix} -1 & -1 & 0.4032 & 0.3856 & 0.1921 \end{bmatrix}^{\mathrm{T}}$$
$$x_0 = \begin{bmatrix} -1 & -1 \end{bmatrix}^{\mathrm{T}}: \qquad U_0 = \begin{bmatrix} 1 & 1 & -0.4032 & -0.3856 & -0.1921 \end{bmatrix}^{\mathrm{T}}$$