Institute of Robotics, University of Innopolis

Computational Intelligence
Linear Programming

May 16, 2021

1 Task 01

$$\begin{aligned} \max_{\mathbf{x}} \quad & x_1 + x_2 \\ \text{s.t.} \quad & 9x_1 + 3x_2 \leq 56, \\ & -7x_1 + 9x_2 \leq 56, \\ & -1 \leq \mathbf{x} \leq 1 \end{aligned} \tag{1}$$

1. Formulate the problem using CVXPY and scipy.optimize.linprog `https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.linprog.html`

2 Task 02

You are given three non empty sets:

$$x_{(1)}, \ldots, x_{(n)}$$
$$y_{(1)}, \ldots, y_{(m)}$$
$$z_{(1)}, \ldots, z_{(p)}$$

in $\mathbb{R}^n$ where you have to find corresponding affine functions in the following form:

$$f_i(\mu) = a_i^T \mu - b_i, \; i = 1, 2, 3, \; \mu = x, y, z \tag{2}$$

subject to the following constraints:

$$\begin{aligned} & f_1(x_{(j)}) > max\{f_2(x_{(j)}), f_3(x_{(j)})\}, \; j = 1, \ldots, n \\ & f_2(y_{(j)}) > max\{f_1(y_{(j)}), f_3(y_{(j)})\}, \; j = 1, \ldots, m \\ & f_3(z_{(j)}) > max\{f_1(z_{(j)}), f_2(z_{(j)})\}, \; j = 1, \ldots, p \\ & a_1 + a_2 + a_3 = 0, \\ & b_1 + b_2 + b_3 = 0, \end{aligned}$$

1. Use the following script for generating three sets in $\mathbf{R}^2$ and solve (2) using CVXPY

```python
import numpy as np
import cvxpy as cp
```

```python
import matplotlib.pyplot as plt
import random
from random import random
import math


def get_circle(U):
    cx = cp.Variable()
    cy = cp.Variable()
    obj = cp.Minimize(cp.norm(cp.vstack((U[0,:] - cx, U[1,:]
        - cy))))
    prob = cp.Problem(obj, [])
    prob.solve()

    cx, cy = map(lambda x: x.value, [cx, cy])
    xc = np.array([cx, cy])
    r_hat = (U.T - xc)
    mean_r = np.sum(r_hat * r_hat, axis=1).mean()
    r = np.sqrt(mean_r)
    return xc, r

def draw_circles(sets):
    ax = plt.gca()
    for set_i in sets:
        xc, r = get_circle(set_i)
        circle = plt.Circle(xc, r, fill=False)
        ax.add_patch(circle)

def clusters(n, points, centers, r):
    sets = []
    def cluster(points, center, radius):
        npoints = 50
        r = radius
        t = np.linspace(0, 2*np.pi, npoints, endpoint=False)
        x = center[0] + r * np.cos(t)
                    + np.random.uniform(-0.2,0.2,t.shape[0])
        y = center[1] + r * np.sin(t)
                    + np.random.uniform(-0.2,0.2,t.shape[0])
        return np.vstack((x,y))

    for i in range(n):
```

```
        set_i = cluster(points, centers[i], r)
        sets.append(set_i)
    return sets

sets = np.array(clusters(3, 100, [(2,2), (4,6), (3, 8)], 1.0))

X = sets[0]
Y = sets[1]
Z = sets[2]
```

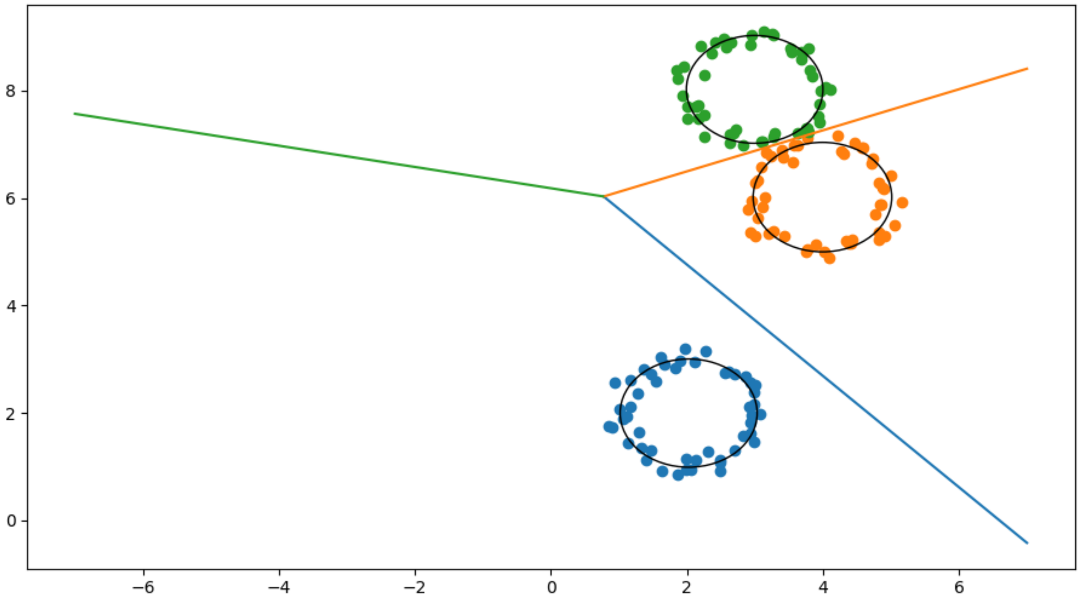

Figure 1: The expected output

3 Task 03

Now we are going to consider trajectory's state prediction x_k each time instance k, in terms of control input sequence $\mathbf{u_k}$ for a given initial condition, i.e., $x_{0|k}$. To estimate the optimal state prediction, an optimal control sequence (or control policy) has to be calculated. Such a control policy can be estimated minimizing the following quadratic cost:

$$J(x_k, \mathbf{u_k}) = \sum_{i=0}^{N-1} \|x_{k+i}\|_Q^2 + \|u_{k+i}\|_R^2 + \|x_{k+N}\|_P^2 \tag{3}$$

How do you determine the weight matrices: Q, R, and P? For a linear system, state prediction sequence can be written in a compact sequence as follows:

$$\mathbf{x_k} = M x_k + C \mathbf{u_k}, \quad M = \begin{bmatrix} I \\ A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}, \; C = \begin{bmatrix} 0 & 0 & \dots & 0 \\ B & & & \\ AB & B & & \\ \vdots & \vdots & \ddots & \\ A^{N-1}B & A^{N-2}B & \dots & B \end{bmatrix} \tag{4}$$

The defined quadratic cost (3) can be written in terms of $\mathbf{x_k}$ and $\mathbf{u_k}$ as

$$J = \mathbf{x}_k^T \tilde{Q} \mathbf{x_k} + \mathbf{u_k}^T \tilde{R} \mathbf{u_k} = \mathbf{u_k}^T H \mathbf{u_k} + 2 x_k^T F^T \mathbf{u_k} + x_k^T G x_k \tag{5}$$

Can you define the $\tilde{Q}$ and $\tilde{R}$? as well as prove that H, F, and G are given by $C^T \tilde{Q} C + \tilde{R}$, $C^T \tilde{Q} M$, and $M^T \tilde{Q} M$, respectively. Let's say there is no additional constraints are given, 5 has a closed-form solution which can be derived by minimizing the J with respect to $\mathbf{u}$. Show that $\mathbf{u}^* = -H^{-1} F x_k$. What can you say about when H is singular (i.e., positive semi-definite rather than positive definite); this implies there are multiple optimal solutions can be exits. Since H and F are constant matrices, $\mathbf{u_k} = L x_k$, where $L = -H^{-1} F$.

Now let's try to find out the feedback control law, namely L, considering following second order system with

$$A = \begin{bmatrix} 1.1 & 2 \\ 0 & 0.95 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0.0787 \end{bmatrix} \tag{6}$$

for horizon N = 4, you may assume $Q = C^T C$, $R = 0.01$, and $P = Q$.

4 Task 04

Here we are interested on terminal constraints set which helps to guarantee the recursive feasibility. In details description about recursive feasibility [1, 2]. Let u_{min} and u_{max} be the minimum and maximum values for u, respectively. Let's consider N horizon state prediction as we did before. To ensure u stays within its boundary constraints, i.e., u_{min} and u_{max}, u always within the Ω for all $i = 0, \dots, N$

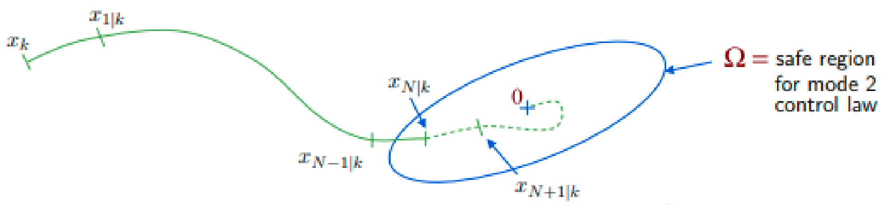

Figure 2: The terminal constraint set Ω

$$\Omega = \{x : u_{min} \leq K(A + BK)^i x \leq u_{max}, \; i = 0, \dots, N\} \tag{7}$$

where K is the LQ optimal gain. If you do not get whats going on that's all right. Let's get to the the point. Consider the system we examined in (14) and assume $K = [-1.19 \;\; -7.88]$. The terminal constraint set can be calculated as follows:

$$
\begin{aligned}
\Omega_0 &= \{x : -1 \leq [-1.19 \;\; -7.88]x \leq 1\} \\
\Omega_1 = \Omega_0 &\cap \{x : -1 \leq [-0.5702 \;\; -4.9572]x \leq 1\} \\
\Omega_2 = \Omega_1 &\cap \{x : -1 \leq [-0.1621 \;\; -2.7826]x \leq 1\} \\
&\vdots
\end{aligned}
\tag{8}
$$

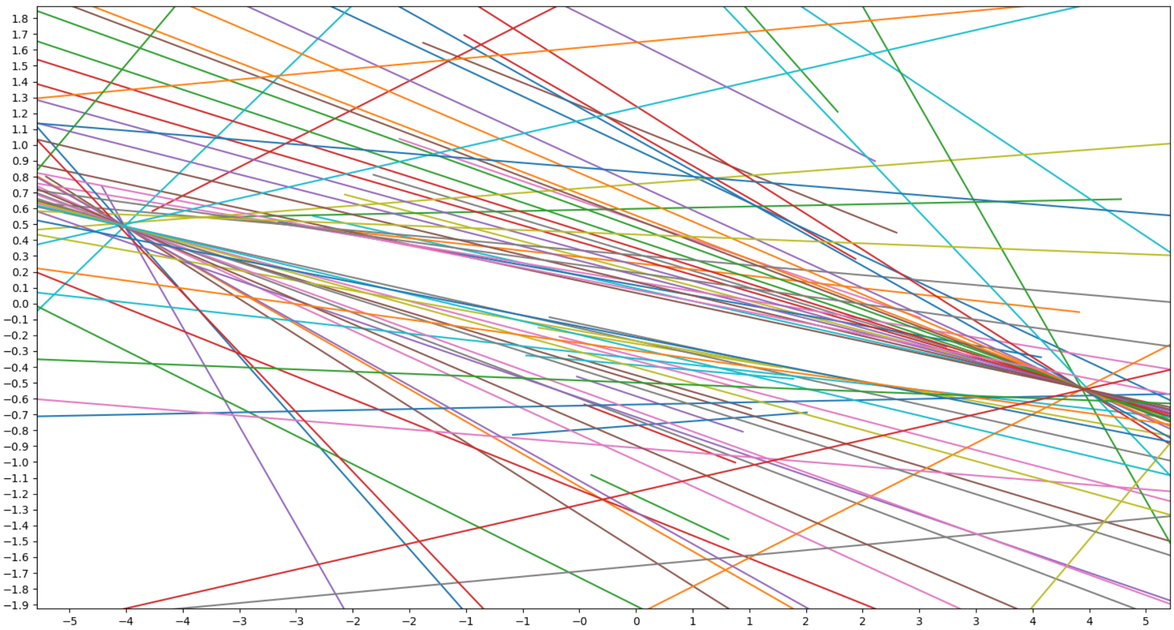

Figure 3: $\Omega_N = \Omega_{N+1}, \ldots, = \Omega_\infty$

If the input u has n_u dimension, how can we estimate $u_{max,j}$ and $u_{min,j}$ for $j = 0, .., n_u$?

$$
\begin{aligned}
u_{max,j} &= \max_x \; K_j(A+BK)^{N+1}x \quad s.t.\; u_{min} \leq K(A+BK)^i x \leq u_{max}, \;\; i = 0, \ldots, N \\
u_{min,j} &= \min_x \; K_j(A+BK)^{N+1}x \quad s.t.\; u_{min} \leq K(A+BK)^i x \leq u_{max}, \;\; i = 0, \ldots, N
\end{aligned}
\tag{9}
$$

Hence, terminal constraints set finding can be formulated in the following way:

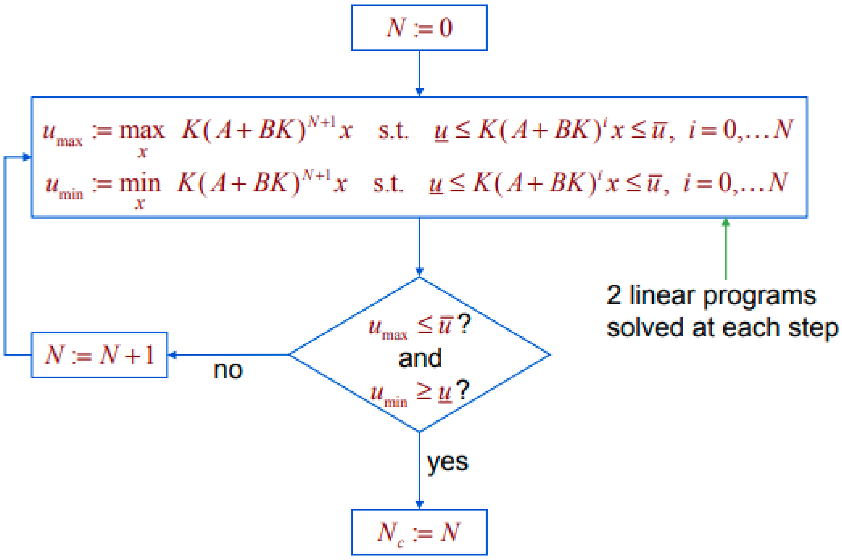

Figure 4: A algorithm for constraint checking [1]

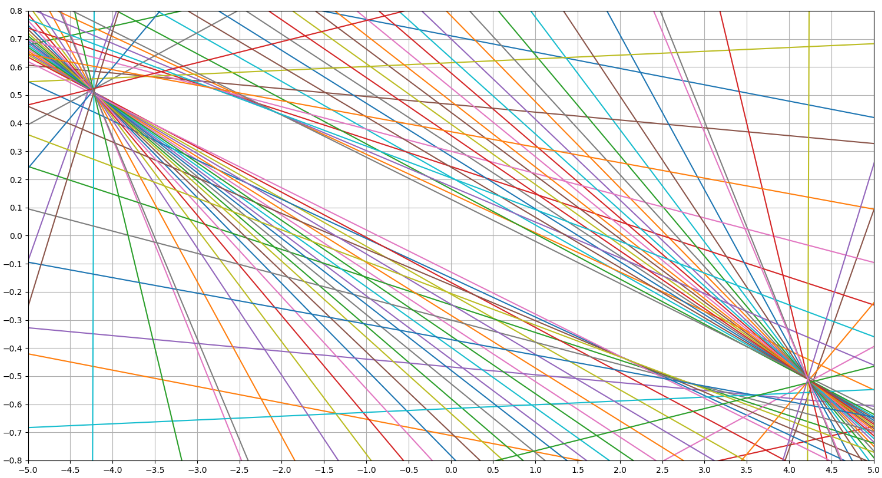

Figure 5: The terminal set after applying algorithm 4

1. Use linprog for formulate Algorithm.4, you may use following function for plotting the linear inequalities

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import linprog

def plot_line(slope, intercept):
    axes = plt.gca()
    x_vals = np.array(axes.get_xlim())
    y_vals = intercept + slope * x_vals
```

```python
    start, end = axes.get_xlim()
    axes.xaxis.set_ticks(np.arange(start, end, 0.5))
    start, end = axes.get_ylim()
    axes.yaxis.set_ticks(np.arange(start, end, 0.1))
    plt.plot(x_vals, y_vals, '-')

plt.ylim((-0.8, 0.8))
plt.xlim((-5,5))
```

5 Task 05

Assume a system dynamics is described in terms of LTI (linear time invariant) state-space model

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k, \end{aligned} \tag{10}$$

where $x_k \in \mathbb{R}^{n_x}$, $u_k \in \mathbb{R}^{n_u}$, and $y_k \in \mathbb{R}^{n_y}$. Such control system is assumed be enforced by a set of linear constraints, i.e., may involve both state and inputs. In general, those can be expressed as

$$Fx + Gu \leq 1, \tag{11}$$

where $F \in \mathbb{R}^{n_c \times n_x}$ and $G \in \mathbb{R}^{n_c \times n_u}$.

Theorem 1 *The MPI (Maximum Positive Invariant) set for the system with dynamics (10) and the system constraints set (11) can be defined as:*

$$X^{MPI} \doteq \{x : (F + GK)\Phi^i x \leq 1,\ i = 0, \ldots, v\}, \tag{12}$$

where v is the smallest positive integer such that $(F + GK)\Phi^{v+1}x \leq 1$, $\forall x$ satisfying $(F + GK)\Phi^i x \leq 1,\ i = 0, \ldots, v$. Φ is determined as $A + B \cdot K$. The value of v can be computed by solving following LPs, namely

$$\begin{aligned} \max_x \quad & (F + GK)_j \Phi^{n+1} x \\ \text{s.t.} \quad & (F + GK)\Phi^i x \leq 1,\ i = 0, \ldots, n \end{aligned} \tag{13}$$

for $j = 1, \ldots, n_c$, n=1,...,v, where $(F + GK)_j$ denotes the jth row of F+GK.

Now considering following second order system with

$$A = \begin{bmatrix} 1.1 & 2 \\ 0 & 0.95 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0.0787 \end{bmatrix} \tag{14}$$

whose constraints are given by $-1 \leq x/8 \leq 1$ and $-1 \leq u \leq 1$. With listed constraints we can define the F and G as follows:

$$F = \begin{bmatrix} 0 & 1/8 \\ 1/8 & 0 \\ 0 & -1/8 \\ -1/8 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \ G = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ -1 \end{bmatrix} \tag{15}$$

Calculate the MPI set for this system where assume value of v some value in between 5 to 20. The following helper functions may help you.

```python
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import linprog

def plot_line(slope, intercept, range_xval=(-8, 8)):
    axes = plt.gca()
    x_vals = np.array(axes.get_xlim())
    y_vals = intercept + slope * x_vals
    start, end = axes.get_xlim()
    axes.xaxis.set_ticks(np.arange(start, end, 0.5))
    start, end = axes.get_ylim()
    axes.yaxis.set_ticks(np.arange(start, end, 0.1))
    plt.plot(x_vals, y_vals, '-')
```

```python
bnd = [(-8, 8), (-8, 8)]
opt = linprog(c=obj, A_ub=lhs_ineq, b_ub=rhs_ineq, bounds=bnd)
```

The expected output something similar to this:

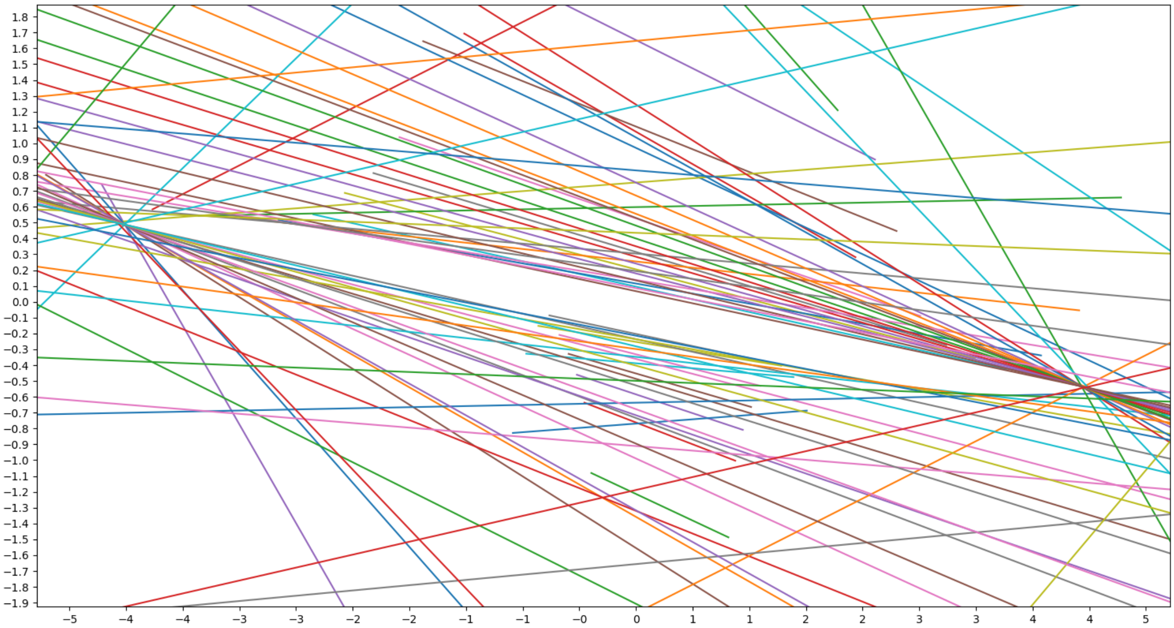

Figure 6

References

[1] *https://markcannon.github.io/assets/downloads/teaching/C21_Model_Predictive_Control/mpc_notes.pdf*. 2020.

[2] Saša V Raković. “Model predictive control: classical, robust, and stochastic [bookshelf]”. In: *IEEE Control Systems Magazine* 36.6 (2016), pp. 102–105.