CS164 PRECLASS WORK EXERCISES - MIXED-INTEGER PROGRAMMING

In this exercise, you will implement a collision-free trajectory optimizer for a simple point-mass vehicle. We will define a vector $x(k)$ with four components, where the first two components $[x_1, x_2]^T$ represent the position of the vehicle (in two dimensions), and the next two components $[x_3, x_4]^T$ represent the velocity. We can write the state update equations for this vehicle, sampled every second, by the equation

$$x(k+1) = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x(k) + \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} u(k), \tag{1}$$

where $u(k)$ is a vector of the force applied at time k in the two coordinate directions. Given an initial state $x(0)$, we can then determine the position at any future time N by using (1) to find the trajectory $x(0), x(1), \ldots x(N)$. For spacecraft applications in particular the one norm of the input vector $\|u(k)\|_1$ gives a measure of the fuel use at time k. We will assume that the maximum force that can be applied at any step is 0.5: this is a constraint of the form $\|u(k)\|_\infty \leq 0.5$, for all k.

Our vehicle starts with zero velocity at position $(0, 0)$, and our goal is to reach a square target zone with centre at $(10, 10)$ and side length 1. There is one square obstacle centred at $(5, 5)$, with side length 3.

You are provided with a code template here. Find an optimal trajectory by filling out the following code segments:

(1) **Dynamics:** Fill out the matrices A and B for the dynamics of the spacecraft, and the matrix C to extract the position states from x.

(2) **Cost Function:** We want to find the $u(0), u(1), \ldots, u(N-1)$ that minimizes $\sum_{k=0}^{N-1} \|u(k)\|_1$. Add in this objective function into the code where indicated.

(3) **Input Constraints:** Add in the constraint that the input magnitude must be less than 0.5 in each coordinate direction where indicated.

(4) **Obstacle Avoidance:** Using an appropriate big-M formulation with the binary variables b, add in a constraint to prevent the point-wise trajectory from impinging upon the obstacle defined in half-space form by obsLHS * $Cx \leq$ obsRHS. We need the C matrix here, since the obstacle only affects the positions.

Once the missing code is filled out, run the template to solve the optimization problem in CVXPY and produce a plot of the optimal trajectory and inputs.

Extensions:

(1) How might we ensure that the segments in-between the points also do not impinge on the obstacle? Modify the constraints so that the connecting line segment between trajectory points does not impinge on the obstacle.

(2) It may be possible that a trajectory shorter than ten steps may have a better fuel economy. One way to optimize over the trajectory length is to have a constraint on every $x(1), x(2), \ldots, x(10)$ being within the terminal zone, and use a big-M formulation with binary variables to have each of these constraints toggled on and off with a binary variable. We can then enforce a constraint that sets the sum of these binary variables to 1 (meaning exactly one of these states has to be within the target zone).