EE364b Homework 7

1. *MPC for output tracking.* We consider the linear dynamical system

$$x(t+1) = Ax(t) + Bu(t), \quad y(t) = Cx(t), \quad t = 0, \ldots, T-1,$$

with state $x(t) \in \mathbf{R}^n$, input $u(t) \in \mathbf{R}^m$, and output $y(t) \in \mathbf{R}^p$. The matrices A and B are known, and $x(0) = 0$. The goal is to choose the input sequence $u(0), \ldots, u(T-1)$ to minimize the output tracking cost

$$J = \sum_{t=1}^{T} \|y(t) - y_{\rm des}(t)\|_2^2,$$

subject to $\|u(t)\|_\infty \leq U^{\max}$, $t = 0, \ldots, T-1$.

In the remainder of this problem, we will work with the specific problem instance with data

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0.5 \\ 1 \end{bmatrix}, \quad C = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix},$$

$T = 100$, and $U^{\max} = 0.1$. The desired output trajectory is given by

$$y_{\rm des}(t) = \begin{cases} 0 & t < 30, \\ 10 & 30 \leq t < 70, \\ 0 & t \geq 70. \end{cases}$$

(a) Find the optimal input $u^\star$, and the associated optimal cost $J^\star$.

(b) *Rolling look-ahead.* Now consider the input obtained using an MPC-like method: At time t, we find the values $u(t), \ldots, u(t+N-1)$ that minimize

$$\sum_{\tau=t+1}^{t+N} \|y(\tau) - y_{\rm des}(\tau)\|_2^2,$$

subject to $\|u(\tau)\|_\infty \leq U^{\max}$, $\tau = t, \ldots, t+N-1$, and the state dynamics, with $x(t)$ fixed at its current value. We then use $u(t)$ as the input to the system. (This is an informal description, but you can figure out what we mean.)

In a tracking context, we call N the amount of *look-ahead*, since it tells us how much of the future of the desired output signal we are allowed to access when we decide on the current input.

Find the input signal for look-ahead values $N = 8$, $N = 10$, and $N = 12$. Compare the cost J obtained in these three cases to the optimal cost $J^\star$ found in part (a). Plot the output $y(t)$ for $N = 8$, $N = 10$, and $N = 12$.

2. *Branch and bound for partitioning.* We consider the two-way partitioning problem (see pages 219, 226, and 285 in the book),

$$\begin{array}{ll} \text{minimize} & x^T W x \\ \text{subject to} & x_i^2 = 1, \quad i = 1, \ldots, n. \end{array}$$

We can, without any loss of generality, assume that $x_1 = 1$.

You will perform several iterations of branch and bound for a random instance of this problem, with, say, $n = 100$.

To run branch and bound, you'll need to find lower and upper bounds on the optimal value of the partitioning problem, with some of the variables fixed to given values, *i.e.*,

$$\begin{array}{ll} \text{minimize} & x^T W x \\ \text{subject to} & x_i^2 = 1, \quad i = 1, \ldots, n \\ & x_i = x_i^{\text{fixed}}, \quad i \in \mathcal{F}, \end{array}$$

where $\mathcal{F} \subseteq \{1, \ldots, n\}$ is the set of indices of the fixed entries of x, and $x_i^{\text{fixed}} \in \{-1, 1\}$ are the associated values.

Lower bound. To get a lower bound, you will solve the SDP

$$\begin{array}{ll} \text{minimize} & \mathbf{Tr}(WX) \\ \text{subject to} & X_{ii} = 1, \quad i = 1, \ldots, n \\ & \begin{bmatrix} X & x \\ x^T & 1 \end{bmatrix} \succeq 0 \\ & x_i = x_i^{\text{fixed}}, \quad i \in \mathcal{F}, \end{array}$$

with variables $X \in \mathbf{S}^n$, $x \in \mathbf{R}^n$. (If we add the constraint that the $(n+1) \times (n+1)$ matrix above is rank one, then $X = xx^T$, and this problem gives the exact solution.)

Upper bound. To get an upper bound we'll find a feasible point $\hat{x}$, and evaluate its objective. To do this, we solve the SDP relaxation above, and find an eigenvector v associated with the largest eigenvalue of the $(n+1) \times (n+1)$ matrix appearing in the inequality. Choose $\hat{x}_i = \mathbf{sign}(v_{n+1})\mathbf{sign}(v_i)$ for $i = 1, \ldots, n$.

Branch and bound. Develop a partial binary tree for this problem, as on page 21 of the lecture slides. Use $n = 100$. At each node, record upper and lower bounds. Along each branch, indicate on which variable you split. At each step, develop the node with the smallest lower bound. Develop four nodes.

You can do this 'by hand'; you do not need to write a complete branch and bound code in Matlab (which would not be a pretty sight). You can use `cvx` to solve the SDPs, and manually constrain some variables to fixed values.