ME C231 Assignment: Finite-Time Optimal Control

1. **Unconstrained Linear Finite Time Optimal Control**
Consider the discrete-time dynamic system with the following state space representation:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 0.77 & -0.35 \\ 0.49 & 0.91 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0.04 \\ 0.15 \end{bmatrix} u(k) \tag{1}$$

We want to design a linear quadratic optimal control for this system with a finite horizon $N = 50$. We set the following cost matrices:

$$Q = \begin{bmatrix} 500 & 0 \\ 0 & 100 \end{bmatrix}, \quad R = 1, \quad P = \begin{bmatrix} 1500 & 0 \\ 0 & 100 \end{bmatrix},$$

and assume that the initial state is $x(0) = [1, -1]^T$;

(a) Determine the optimal set of inputs

$$U_0 = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

through the Batch Approach, i.e. by writing the dynamic equations as follows:

$$\begin{aligned} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ \vdots \\ x_N \end{bmatrix} &= \begin{bmatrix} I \\ A \\ \vdots \\ \vdots \\ A^N \end{bmatrix} x(0) + \begin{bmatrix} 0 & \cdots & \cdots & 0 \\ B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & \ddots & \ddots & 0 \\ A^{N-1}B & \cdots & AB & B \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ \vdots \\ u_{N-1} \end{bmatrix} \\ &= \mathcal{S}^x x(0) + \mathcal{S}^u U_0, \end{aligned}$$

and using the formula:

$$U_0^*(x(0)) = -(\mathcal{S}^{uT}\overline{Q}\mathcal{S}^u + \overline{R})^{-1}\mathcal{S}^{uT}\overline{Q}\mathcal{S}^x x(0)\,,$$

and calculate the optimal cost $J_0^*(x(0))$:

$$J_0^*(x(0)) = x(0)^T(\mathcal{S}^{xT}\overline{Q}\mathcal{S}^x - \mathcal{S}^{xT}\overline{Q}\mathcal{S}^u(\mathcal{S}^{uT}\overline{Q}\mathcal{S}^u + \overline{R})^{-1}\mathcal{S}^{uT}\overline{Q}\mathcal{S}^x)x(0)\,.$$

Paste your code in the hard copy and print out $U_0^*(x(0))^T$ and $J_0^*(x(0))$
Hint: To efficiently concatenate the matrices, use the MATLAB commands `kron`, `repmat`. We have already posted a code for doing this, it is ok to understand the code, check if is correct and use it.

(b) Verify the results of the previous point by solving a numerical optimization problem. In fact, the cost can be written as a function of U_0 as follows:

$$\begin{aligned} J_0(x(0), U_0) &= (\mathcal{S}^x x(0) + \mathcal{S}^u U_0)^T \overline{Q} (\mathcal{S}^x x(0) + \mathcal{S}^u U_0) + U_0^T \overline{R} U_0 \\ &= U_0^T H U_0 + 2x(0)^T F U_0 + x(0)^T \mathcal{S}^{xT} \overline{Q} \mathcal{S}^x x(0), \end{aligned}$$

where $H := \mathcal{S}^{uT} \overline{Q} \mathcal{S}^u + \overline{R}$ and $F := \mathcal{S}^{xT} \overline{Q} \mathcal{S}^u$, and then minimized by solving an quadratic minimization problem. Check that the optimizer U_0^* and the optimum $J_0(x(0), U_0^*)$ correspond to the ones determined analytically in the previous point. Paste your code in the hard copy.

Hint: Quadratic optimization problems can be solved in MATLAB using `quadprog`.

Note: Linear constraints on the inputs could have been specified when calling `quadprog`. Make sure you compute the unconstrained solution here and do not have any linear constraints.

(c) Design the optimal controller through the recursive approach and determine the optimal state-feedback matrices F_k. Start from the Riccati Difference Equations, assuming that $P_N = P$, and compute recursively the P_k:

$$P_k = A^T P_{k+1} A + Q - A^T P_{k+1} B (B^T P_{k+1} B + R)^{-1} B^T P_{k+1} A, \tag{2}$$

and then calculate F_k as a function of P_{k+1}:

$$F_k = -(B^T P_{k+1} B + R)^{-1} B^T P_{k+1} A.$$

Compare the optimal cost $J_0^*(x(0)) = x(0)^T P_0 x(0)$ with question 1.a and check that they are equal. Hand in your code in the hard copy.!

Hint: We have already posted a code for doing this, it is ok to understand the code, check if is correct and use it.

(d) We want to understand the effect of model uncertainty when using the two approaches in simulations (batch and recursive). Hence, let us add an additive process disturbance $Dw(k)$ to the right-hand side of Equation (1). Assume the matrix D to be:

$$D = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix},$$

while the process $w(k)$ a Gaussian white noise, with mean $m = 0$ and variance $\sigma^2 = 10$.

Consider the simulation length to be equal to N time steps and that the input u_k to the system are defined as follows:

$$u_k = \begin{cases} U_0[k+1] & \text{for Batch Approach} \\ F_k x_k & \text{for Recursive Approach} \end{cases}$$

where $U_0[k]$ is the k-th component of the vector U_0.

Plot a graph of the state evolution over time. What is the difference in the dynamic evolution? What happens if you modify the variance of the disturbance?

Hint: Create a MATLAB function that simulates your dynamic system, i.e. has output $x(k+1)$ and inputs $x(k)$ and $u(k)$, and calculate $u(k)$ based on the two different approaches. For the generation of the white noise, we suggest the MATLAB commands `randn` or `awgn`. Initialize the random number generator with the command `rng(0)`.

2. **Constrained Finite Time Optimal Control - Sparse vs Dense QP Formulations**
Consider CFTOC of a discrete-time double-integrator system:

$$\min_{\substack{u_0,\dots,u_{N-1}\\ x_1,\dots,x_N}} \quad x_N^T P x_N + \sum_{k=0}^{N-1} x_k^T Q x_k + u_k^T R u_k \tag{3a}$$

$$\text{subject to} \quad x_{k+1} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x_k + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_k \tag{3b}$$

$$-1 \le u_k \le 1,\ k \in \{0,\dots,N-1\} \tag{3c}$$

$$\begin{bmatrix} -15 \\ -15 \end{bmatrix} \le x(k) \le \begin{bmatrix} 15 \\ 15 \end{bmatrix},\ k \in \{1,\dots,N\} \tag{3d}$$

where $N = 3$, $P = Q = \mathcal{I}_{2\times 2}$, $R = 0.1$.

(a) Let $x_0 = [-1,-1]^T$. Determine the QP **dense** formulation when you substitute the dynamics in the CFTOC. That is, determine H, f, c, A, b for the following problem:

$$\begin{aligned} \min_{U_0} \quad & \frac{1}{2} U_0^T H U_0 + f^T U_0 + c \\ \text{subject to} \quad & A U_0 \le b. \end{aligned}$$

where $U_0 = \left[u_0^T, u_1^T, \dots, u_{N-1}^T\right]^T$. Write a script to synthesize these matrices and then compute the solution using quadprog. Note that you can drop the c term when using quadprog. You should NOT have any Aeq or beq (see `help quadprog`). Make sure to turn in your code and outputs (synthesized matrices, quadprog solution and optimal value) in the hard copy. How many 0's are in A? In H?

(b) Let $x_0 = [-1,-1]^T$. Determine the QP **sparse** formulation when you do NOT substitute the dynamics in the CFTOC. That is, what are H, f, A, b, Aeq, beq for the following problem:

$$\begin{aligned} \min_{z} \quad & \frac{1}{2} z^T H z + f^T z \\ \text{subject to} \quad & A z \le b \\ & A_{eq} z = b_{eq}. \end{aligned}$$

where $z = [x_1^T, \dots, x_N^T, u_0^T, \dots, u_{N-1}^T]^T$. This requires using Aeq and beq described in the `quadprog.m` help file. Again, write a script to synthesize these matrices and compute the solution using quadprog. Make sure to turn in your code and outputs (synthesized matrices, quadprog solution and optimal value) in the hard copy. How many 0's are in A? In H? In Aeq?

(c) Compare the quadprog solver times compare using `tic toc` commands. Check your solutions to make sure the two methods get the same answer.

3. **Linear CFTOC via Dynamic Programming**

Consider the CFTOC problem (3). Compute the state feedback solution using the DP algorithm.

(a) Upload your code on bCourse named cftocdp.m.

(b) For the same initial point $x_0 = [-1, -1]^T$ at the previous question, compare the input sequence obtained form the DP solution with the one obtained with the batch solution.

4. **Nonlinear CFTOC via Dynamic Programming- Parking a Vehicle (optional)**
The majority of the problem is taken from the past homework. Consider the same simplified kinematic bicycle model:

$$\begin{aligned}
\dot{x} &= v\cos(\psi + \beta) \\
\dot{y} &= v\sin(\psi + \beta) \\
\dot{v} &= a \\
\dot{\psi} &= \frac{v}{l_r}\sin(\beta) \\
\beta &= \tan^{-1}\left(\frac{l_r}{l_f + l_r}\tan(\delta_f)\right)
\end{aligned}$$

where

$x =$ global x CoG coordinate
$y =$ global y CoG coordinate
$v =$ speed of the vehicle
$\psi =$ global heading angle
$\beta =$ angle of the current velocity with respect to the longitudinal axis of the car
$a =$ acceleration of the center of mass into this direction
$l_r =$ distance from the center of mass of the vehicle to the rear axle
$l_f =$ distance from the center of mass of the vehicle to the front axle
$\delta_f =$ steering angle of the front wheels with respect to the longitudinal axis of the car

Collect the states in one vector $z = [x, y, v, \psi]^T$, and the inputs as $u = [a, \beta]^T$. Here, we approximate the continuous system by using a Forward Euler Discretization model with sampling time $\Delta t = 0.2$sec and use $l_f = l_r = 1.738$ in the simulation.
Note: To make the problem easier, we simplified the finite-time optimal control problem by removing the constraint, $|\beta_{k+1} - \beta_k| \leq \beta_d \ \forall k = \{0, \ldots, N-2\}$ in homework 3 and all the state constraints.

You are asked to solve the following finite-time optimal control problem via the dynamic programming approach.

$$\min_{z_0,\dots,z_N,u_0,\dots,u_{N_1}} \sum_{k=N-2}^{k=N} \|z_k - \bar{z}_N\|_2^2$$

$$\begin{aligned}
& z_{k+1} = z_k + f(z_k, u_k)\Delta t && \forall k = \{0,\dots,N-1\} \\
& z_{min} \le z_k \le z_{max} && \forall k = \{0,\dots,N\} \\
& u_{min} \le u_k \le u_{max} && \forall k = \{0,\dots,N\} \\
& z_0 = \bar{z}_0 \\
& z_N = \bar{z}_N
\end{aligned}$$

Our goal is to park the vehicle in the terminal state $\bar{z}_N = [0, 0, 0, -\pi/2]$ Consider the following constraints:

- The accelerations are bounded by $|a(k)| \le 1.5\Delta t$ m/s^2.
- The steering control inputs are limited to $|\beta(k)| \le 0.6$ rad.

(a) Write the DP algorithm to solve the problem for a $N = 70$. Since this can take hours or even days, restrict the computation of the policy to a small portion of the state space: $[-0.07, -0.15, -0.65, -1.7]^T \le z(k) \le [1.5, 3.2, 1, 0.08]^T$. Once solved, pick $\bar{z}_0 = [0,\ 3,\ 0,\ 0]^T$ as initial state and plot the vehicle states and inputs when it is controlled by the optimal feedback controller. Hand in with the plots and the code in the hard copy. Upload your code on bCourse named ParkingDP.m.

(b) For the same initial point at the previous step, compare the input sequence obtained form the DP solution with the one obtained with the batch solution.