S26 AerE 5740: Optimal Control

Practice Set 06: Discrete-time Finite Horizon LQ Tracking

Instructor: Abhishek Halder

Problem 1. Platooning.

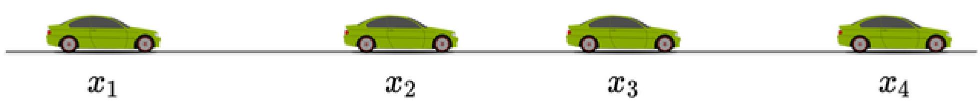

Figure 1: Platooning along a single lane straight road.

In a single lane straight road, N vehicles are moving to the right with respective 1D position coordinates $x_1, x_2, ..., x_N$. See Fig. 3 showing an example scenario for $N = 4$.

Suppose that all vehicles have identical discrete time controlled dynamics

$$x_i(k+1) = x_i(k) + h(u_i(k) - v), \quad i = 1, 2, ..., N,$$

for time index $k = 0, ..., T-1$. The parameter $h > 0$ is known constant sampling time. The control u_i can be thought of as the speed of the ith vehicle, and v is a known speed limit.

Here is the high level question of interest: what should be the optimal controls such that all consecutive vehicles maintain a separation close to some (known) desired distance d at all times? A separation smaller than d may be unsafe, and thus undesirable. A separation more than d reduces traffic throughput, and therefore also undesirable. We also want all vehicles to move at a speed close to the known speed limit v.

(a) OCP formulation.

Motivated by the aforesaid objective, consider minimizing

$$\frac{1}{2}\sum_{i=1}^{N-1}(x_{i+1}(T) - x_i(T) - d)^2 + \frac{1}{2}\sum_{k=0}^{T-1}\left\{\sum_{i=1}^{N-1}(x_{i+1}(k) - x_i(k) - d)^2 + \sum_{i=1}^{N}(u_i(k) - v)^2\right\}$$

subject to $x_i(k+1) = x_i(k) + h(u_i(k) - v)$, $i = 1, 2, ..., N$. Consider the final time T fixed.

Recast this problem as discrete time finite horizon LQ tracking by clearly defining the state vector $\boldsymbol{x}$ and its dimension, the control vector $\boldsymbol{u}$ and its dimension, the output vector $\boldsymbol{y}$ and its dimension, the desired output trajectory $\boldsymbol{y}_d$ to track, the system matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$, and the weight matrices $\boldsymbol{M}, \boldsymbol{Q}, \boldsymbol{R}$ in the cost function.

Solution for part (a):

Let $\mathbf{1}_n$ denote an $n \times 1$ column vector of ones, and let $\boldsymbol{I}_n$ denote the $n \times n$ identity matrix. We define the state vector $\boldsymbol{x}$, the control vector $\boldsymbol{u}$, the output vector $\boldsymbol{y}$, and the desired output vector $\boldsymbol{y}_d$ as

$$\boldsymbol{x} := \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix} \in \mathbb{R}^N, \; \boldsymbol{u} := \begin{pmatrix} u_1 - v \\ u_2 - v \\ \vdots \\ u_N - v \end{pmatrix} \in \mathbb{R}^N, \; \boldsymbol{y} := \begin{pmatrix} x_2 - x_1 \\ x_3 - x_2 \\ \vdots \\ x_N - x_{N-1} \end{pmatrix} \in \mathbb{R}^{N-1}, \; \boldsymbol{y}_d := d\mathbf{1}_{N-1} \in \mathbb{R}^{N-1}.$$

Thus, the system matrices are

$$\boldsymbol{A} := \boldsymbol{I}_N, \quad \boldsymbol{B} := hI_N, \quad \boldsymbol{C} := \begin{pmatrix} -1 & 1 & 0 & 0 & \dots & 0 \\ 0 & -1 & 1 & 0 & \dots & 0 \\ 0 & 0 & -1 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & -1 & 1 \end{pmatrix} \in \mathbb{R}^{(N-1)\times N}.$$

The cost weight matrices are

$$\boldsymbol{M} = \boldsymbol{Q} = \boldsymbol{I}_{N-1} \succ \mathbf{0}, \quad \boldsymbol{R} = \boldsymbol{I}_N \succ \mathbf{0}.$$

With the above definitions in place, we can transcribe the given problem into standard discrete time LQ tracking problem:

$$\underset{\{\boldsymbol{u}_k\}_{k=0}^{T-1}}{\text{minimize}} \quad \frac{1}{2}\left\{(\boldsymbol{y}(T)-\boldsymbol{y}_d(T))'\boldsymbol{M}(\boldsymbol{y}(T)-\boldsymbol{y}_d(T))+\sum_{k=0}^{T-1}(\boldsymbol{y}(k)-\boldsymbol{y}_d(k))'\boldsymbol{Q}(\boldsymbol{y}(k)-\boldsymbol{y}_d(k))+(\boldsymbol{u}(k))'\boldsymbol{R}\boldsymbol{u}(k)\right\}$$

subject to $\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k)$, $\boldsymbol{y}(k) = \boldsymbol{C}\boldsymbol{x}(k)$, $k = 0, 1, \dots, T-1$.

(b) Discrete time LQ tracking solution.

Solve the discrete-time LQ tracking problem in standard form:

$$\underset{\{\boldsymbol{u}_k\}_{k=0}^{T-1}}{\text{minimize}} \quad \frac{1}{2}\left\{(\boldsymbol{y}(T)-\boldsymbol{y}_d(T))'\boldsymbol{M}(\boldsymbol{y}(T)-\boldsymbol{y}_d(T))+\sum_{k=0}^{T-1}(\boldsymbol{y}(k)-\boldsymbol{y}_d(k))'\boldsymbol{Q}(\boldsymbol{y}(k)-\boldsymbol{y}_d(k))+(\boldsymbol{u}(k))'\boldsymbol{R}\boldsymbol{u}(k)\right\}$$

subject to $\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k)$, $\boldsymbol{y}(k) = \boldsymbol{C}\boldsymbol{x}(k)$, $k = 0, 1, \dots, T-1$.

Just like the continuous-time LQ tracking solution in Lec. 13 p. 10-13, here too you should get optimal control as a sum of a linear state feedback term, and a feedforward term. You need to derive a backward vector recursion for the feedforward control. Also derive how the Riccati backward recursion needs to be modified in this case, compared to the same for LQR given in Lec. 13 p. 21-22 and Lec. 14 p. 1-2.

Solution for part (b):

As usual, we start with the Hamiltonian

$$H(k) = \frac{1}{2}(\boldsymbol{C}\boldsymbol{x}(k) - d\mathbf{1})'\boldsymbol{Q}(\boldsymbol{C}\boldsymbol{x}(k) - d\mathbf{1}) + \frac{1}{2}(\boldsymbol{u}(k))'\boldsymbol{R}\boldsymbol{u}(k) + (\boldsymbol{\lambda}(k+1))'(\boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k)),$$

which gives the following necessary conditions for optimality:

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k), \boldsymbol{\lambda}(k) = \frac{\partial H(k)}{\partial \boldsymbol{x}(k)} = \boldsymbol{C}'\boldsymbol{Q}\boldsymbol{C}\boldsymbol{x}(k) - \boldsymbol{C}'\boldsymbol{Q}d\mathbf{1} + \boldsymbol{A}'\boldsymbol{\lambda}(k+1),$$

$$\mathbf{0} = \frac{\partial H(k)}{\partial \boldsymbol{u}(k)} = \boldsymbol{R}\boldsymbol{u}(k) + \boldsymbol{B}'\boldsymbol{\lambda}(k+1),$$

with boundary conditions

$$\boldsymbol{\lambda}(T) = \frac{\partial \phi(\boldsymbol{x}(T), T)}{\partial \boldsymbol{x}(T)} = \boldsymbol{C}'\boldsymbol{M}(\boldsymbol{C}\boldsymbol{x}(T) - d\mathbf{1}), \quad \boldsymbol{x}(0) \text{ given.}$$

This gives us the 2PBVP

$$\begin{pmatrix} \boldsymbol{x}(k+1) \\ \boldsymbol{\lambda}(k) \end{pmatrix} = \begin{pmatrix} \boldsymbol{A} & -\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}' \\ \boldsymbol{C}'\boldsymbol{Q}\boldsymbol{C} & \boldsymbol{A}' \end{pmatrix} \begin{pmatrix} \boldsymbol{x}(k) \\ \boldsymbol{\lambda}(k+1) \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ -\boldsymbol{C}'\boldsymbol{Q} \end{pmatrix} d\mathbf{1},$$

subject to the above boundary conditions.

To solve this 2PBVP, we consider the ansatz $\boldsymbol{\lambda}(k) = \boldsymbol{P}(k)\boldsymbol{x}(k) - \boldsymbol{v}(k)$, which upon comparing with our boundary condition, yields

$$\boldsymbol{P}(T) = \boldsymbol{C}'\boldsymbol{M}\boldsymbol{C}, \quad \boldsymbol{v}(T) = \boldsymbol{C}'\boldsymbol{M}d\mathbf{1}.$$

Now we proceed as in Lec. 13 mutatis mutandis, to obtain

$$\boldsymbol{\lambda}(k+1) = \boldsymbol{P}(k+1)\boldsymbol{x}(k+1) - \boldsymbol{v}(k+1) = \boldsymbol{P}(k+1)\left(\boldsymbol{A}\boldsymbol{x}(k) - \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}'\boldsymbol{\lambda}(k+1)\right) - \boldsymbol{v}(k+1)$$

$$\Rightarrow \boldsymbol{\lambda}(k+1) = \left(\boldsymbol{I} + \boldsymbol{P}(k+1)\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}'\right)^{-1}\boldsymbol{P}(k+1)\boldsymbol{A}\boldsymbol{x}(k) - \left(\boldsymbol{I} + \boldsymbol{P}(k+1)\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}'\right)^{-1}\boldsymbol{v}(k+1).$$

Notice that the costate equation gives $\boldsymbol{\lambda}(k) = \boldsymbol{C}'\boldsymbol{Q}\boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{A}'\boldsymbol{\lambda}(k+1) - \boldsymbol{C}'\boldsymbol{Q}d\mathbf{1}$, wherein the LHS equals $\boldsymbol{P}(k)\boldsymbol{x}(k) - \boldsymbol{v}(k)$ and in the RHS, we substitute the expression for $\boldsymbol{\lambda}(k+1)$ derived above, to arrive at

$$\begin{aligned} \boldsymbol{P}(k)\boldsymbol{x}(k) - \boldsymbol{v}(k) = &\left(\boldsymbol{C}'\boldsymbol{Q}\boldsymbol{C} + \boldsymbol{A}'\left(\boldsymbol{I} + \boldsymbol{P}(k+1)\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}'\right)^{-1}\boldsymbol{P}(k+1)\boldsymbol{A}\right)\boldsymbol{x}(k) \\ &- \boldsymbol{A}'\left(\boldsymbol{I} + \boldsymbol{P}(k+1)\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}'\right)^{-1}\boldsymbol{v}(k+1) - \boldsymbol{C}'\boldsymbol{Q}d\mathbf{1}. \end{aligned}$$

Since the above must hold for all initial conditions $\boldsymbol{x}_0$, and thus for arbitrary $\boldsymbol{x}(k)$, we must have:

$$\boldsymbol{P}(k) = \boldsymbol{C}'\boldsymbol{Q}\boldsymbol{C} + \boldsymbol{A}'\left(\boldsymbol{I} + \boldsymbol{P}(k+1)\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}'\right)^{-1}\boldsymbol{P}(k+1)\boldsymbol{A}, \quad \boldsymbol{P}(T) = \boldsymbol{C}'\boldsymbol{M}\boldsymbol{C},$$
$$\boldsymbol{v}(k) = \boldsymbol{A}'\left(\boldsymbol{I} + \boldsymbol{P}(k+1)\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}'\right)^{-1}\boldsymbol{v}(k+1) + \boldsymbol{C}'\boldsymbol{Q}d\mathbf{1}, \quad \boldsymbol{v}(T) = \boldsymbol{C}'\boldsymbol{M}d\mathbf{1}.$$

Once we solve the above matrix-vector backward recursions for $k = T-1, \ldots, 0$, we can compute the optimal control $\boldsymbol{u}^{\text{opt}}$ as

$$\begin{aligned}
&\boldsymbol{u}^{\text{opt}}(k) \overset{\text{(PMP)}}{=} -\boldsymbol{R}^{-1}\boldsymbol{B}'\boldsymbol{\lambda}(k+1) = -\boldsymbol{R}^{-1}\boldsymbol{B}^{\top}\boldsymbol{P}(k+1)\left(\boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}^{\text{opt}}(k)\right) + \boldsymbol{R}^{-1}\boldsymbol{B}'\boldsymbol{v}(k+1)\\
&\Rightarrow \left(\boldsymbol{I} + \boldsymbol{R}^{-1}\boldsymbol{B}'\boldsymbol{P}(k+1)\boldsymbol{B}\right)\boldsymbol{u}^{\text{opt}}(k) = -\boldsymbol{R}^{-1}\boldsymbol{B}'\boldsymbol{P}(k+1)\boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{R}^{-1}\boldsymbol{B}'\boldsymbol{v}(k+1)\\
&\Rightarrow \boldsymbol{u}^{\text{opt}}(k)\\
&= -\left(\boldsymbol{I} + \boldsymbol{R}^{-1}\boldsymbol{B}'\boldsymbol{P}(k+1)\boldsymbol{B}\right)^{-1}\boldsymbol{R}^{-1}\boldsymbol{B}'\boldsymbol{P}(k+1)\boldsymbol{A}\boldsymbol{x}(k) + \left(\boldsymbol{I} + \boldsymbol{R}^{-1}\boldsymbol{B}'\boldsymbol{P}(k+1)\boldsymbol{B}\right)^{-1}\boldsymbol{R}^{-1}\boldsymbol{B}'\boldsymbol{v}(k+1)\\
&= -\left(\boldsymbol{R}^{-1}\boldsymbol{R} + \boldsymbol{B}'\boldsymbol{P}(k+1)\boldsymbol{B}\right)^{-1}\boldsymbol{R}^{-1}\boldsymbol{B}'\boldsymbol{P}(k+1)\boldsymbol{A}\boldsymbol{x}(k) + \left(\boldsymbol{R}^{-1}\boldsymbol{R} + \boldsymbol{B}'\boldsymbol{P}(k+1)\boldsymbol{B}\right)^{-1}\boldsymbol{R}^{-1}\boldsymbol{B}'\boldsymbol{v}(k+1)\\
&= -\left(\boldsymbol{R} + \boldsymbol{B}'\boldsymbol{P}(k+1)\boldsymbol{B}\right)^{-1}\boldsymbol{R}\boldsymbol{R}^{-1}\boldsymbol{B}'\boldsymbol{P}(k+1)\boldsymbol{A}\boldsymbol{x}(k) + \left(\boldsymbol{R} + \boldsymbol{B}'\boldsymbol{P}(k+1)\boldsymbol{B}\right)^{-1}\boldsymbol{R}\boldsymbol{R}^{-1}\boldsymbol{B}'\boldsymbol{v}(k+1)\\
&= -\left(\boldsymbol{R} + \boldsymbol{B}'\boldsymbol{P}(k+1)\boldsymbol{B}\right)^{-1}\boldsymbol{B}'\boldsymbol{P}(k+1)\boldsymbol{A}\boldsymbol{x}(k) + \left(\boldsymbol{R} + \boldsymbol{B}'\boldsymbol{P}(k+1)\boldsymbol{B}\right)^{-1}\boldsymbol{B}'\boldsymbol{v}(k+1)\\
&= \boldsymbol{u}_{\text{feedback}} + \boldsymbol{u}_{\text{feedforward}},
\end{aligned}$$

where $\boldsymbol{u}_{\text{feedback}} := -\boldsymbol{K}(k)\boldsymbol{x}(k)$, $\boldsymbol{K}(k) := \boldsymbol{K}_v\boldsymbol{P}(k+1)\boldsymbol{A}$, $\boldsymbol{K}_v := \left(\boldsymbol{R} + \boldsymbol{B}'\boldsymbol{P}(k+1)\boldsymbol{B}\right)^{-1}\boldsymbol{B}'$, $\boldsymbol{u}_{\text{feedforward}} := \boldsymbol{K}_v\boldsymbol{v}(k+1)$. This completes the solution.

(c) Optimal vs. desired output trajectories and optimal control for platooning. Fix $T = 200$, $h = 0.01$, $N = 4$, $d = 245$ ft, and the initial conditions $x_1(0) = 0$ ft, $x_2(0) = 250$ ft, $x_3(0) = 480$ ft, $x_4(0) = 780$ ft. Apply your answer in part (b) to the formulation in part (a), to compute and plot $\boldsymbol{y}^{\text{opt}}(k)$ superimposed with $\boldsymbol{y}_d(k)$. Also plot $\boldsymbol{u}^{\text{opt}}(k)$.

Solution for part (c):

See the posted MATLAB code in `PS06Code.zip` within CANVAS Files folder: Problem Sets. This code computes then plots the optimal output trajectory and optimal control shown in the following two figures.

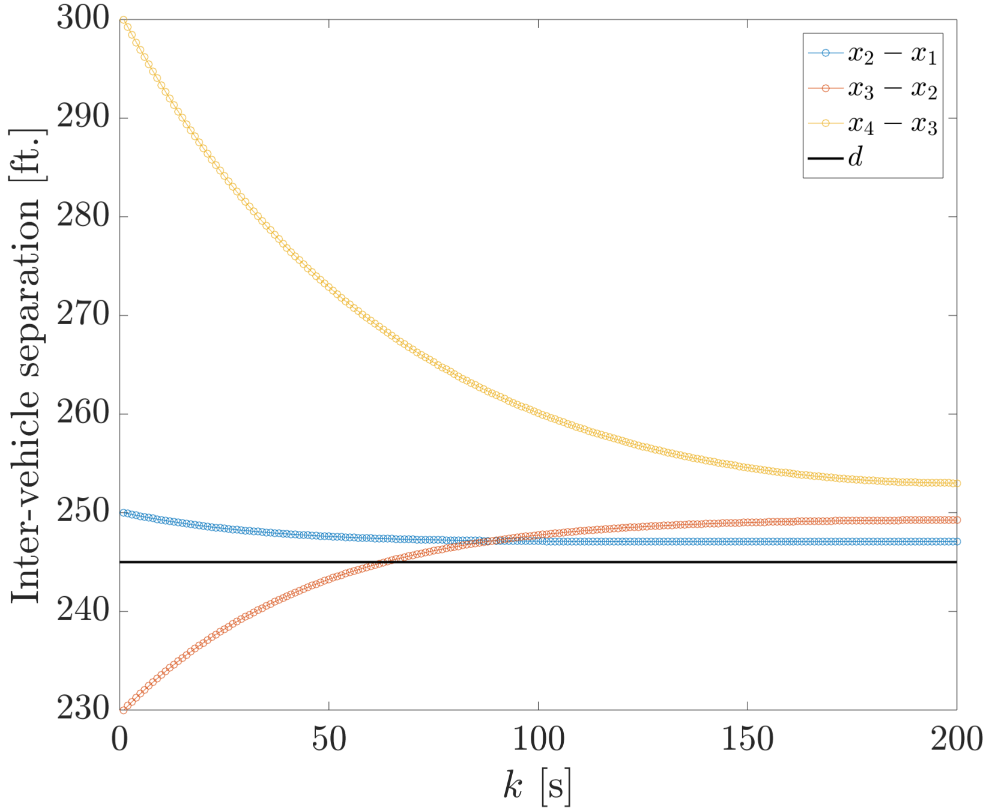

Figure 2: Optimal output.

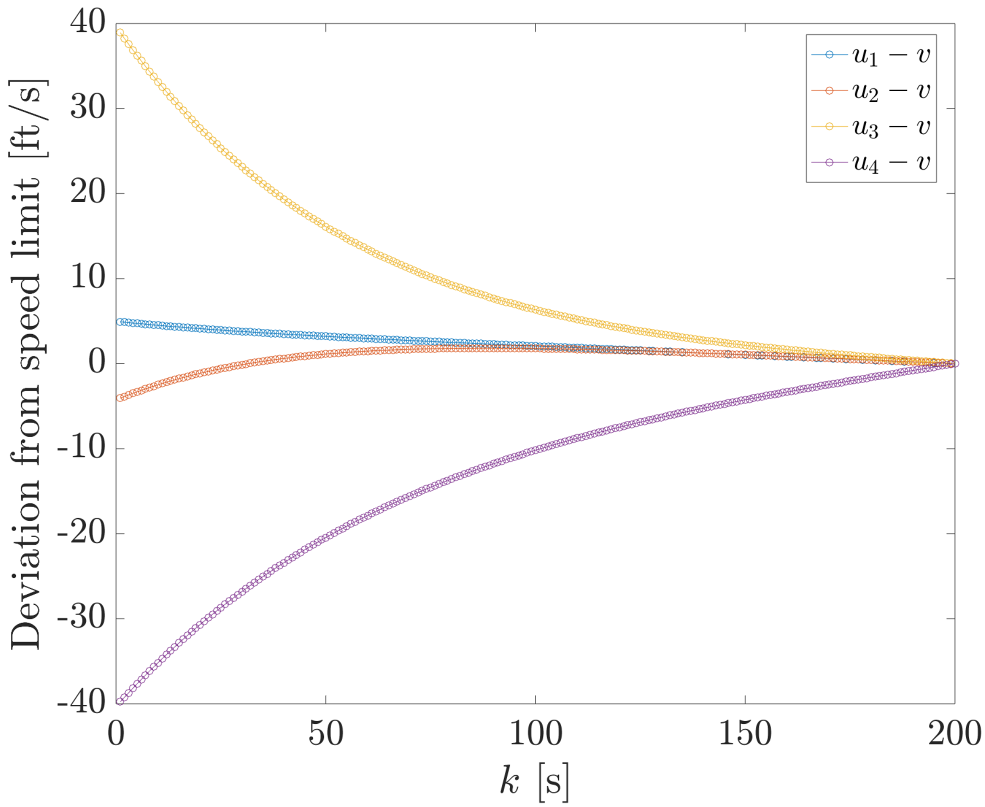

Figure 3: Optimal control.