Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institute for Dynamic Systems and Control
Institut für Dynamische Systeme und Regelungstechnik

# Dynamic Programming and Optimal Control

**Fall 2009**

## Problem Set:
## Problems with Perfect State Information

Notes:

- Problem marked with BERTSEKAS are taken from the book *Dynamic Programming and Optimal Control by Dimitri P. Bertsekas, Vol. I, 3rd edition, 2005, 558 pages, hardcover.*
- The solutions were derived by the teaching assistants in the previous class. Please report any error that you may find to `strimpe@ethz.ch` or `aschoellig@ethz.ch`.

# Problem Set

## Design a Linear Quadratic Regulator (LQR) for the Sideways Motion of a Quadrocopter

A controller is to be designer for our quadrocopter (see Figure 1(a)), which is currently executing first maneuvers in the 'Flying Machine Arena' (ML hall). The goal of the controller design is to perform fast sideways motion.


(a) Picture of the real quadrocopter. (b) Schematic of the 2D model.

*Figure 1: The Quadrocopter.*

The controller design is based on a 2D model of the quadrocopter as illustrated in Figure 1(b):

$$\begin{aligned}
\ddot{y}(t) &= -a(t)\sin(\theta(t)) \\
\ddot{z}(t) &= a(t)\cos(\theta(t)) - g \\
\ddot{\theta}(t) &= q(t),
\end{aligned}$$

where $a(t)$ and $u(t)$ represent the control inputs to the system. The gravitational constant $g$ is approximated by $10\,\mathrm{m/s^2}$. The position variables $y(t)$ and $z(t)$ have units of [m], $\theta$ is given in [rad], and the inputs $a(t)$ and $u(t)$ are in [m/s$^2$] and [rad/s$^2$], respectively.

Only concentrating on the horizontal control, the input $a(t)$ is set to

$$a(t) = \frac{10}{\cos(\theta(t))},$$

resulting in $\ddot{z}(t) = 0$ and the simplified dynamics

$$\ddot{y}(t) = -10\tan(\theta(t)) \tag{1}$$
$$\ddot{\theta}(t) = q(t). \tag{2}$$

## Problem 1 (Linearization)

Linearize Equations (1)-(2) about $\theta = 0$.

## Problem 2 (Discretization)

We control the system with a digital computer. Let $\tau$ be the sampling period and define time-discrete states $x_i(k),\ i = 1, 2, 3, 4$ as follows

$$\begin{aligned} x_1(k) &= \ y(k\tau) \\ x_2(k) &= \ \dot{y}(k\tau) \\ x_3(k) &= \ \theta(k\tau) \\ x_4(k) &= \ \dot{\theta}(k\tau), \qquad k = 0,\ 1,\ 2,\ \ldots \end{aligned}$$

Find a linear, time-discrete expression of the form

$$x(k+1) = Ax(k) + Bu(k),$$

with $x(k) = [x_1(k),\ x_2(k),\ x_3(k),\ x_4(k)]^T$ and $q(t) = u(k)$ for $k\tau \leq t \leq (k+1)\tau$.

## Problem 3 (Infinite Horizon LQR)

Our objective is to design an *infinite horizon linear quadratic regulator (LQR)* that moves the system from the initial state,

$$y(0) = 1, \quad \dot{y}(0) = \theta(0) = \dot{\theta}(0) = 0,$$

*as fast as possible* to the final state,

$$y(T) = \dot{y}(T) = \theta(T) = \dot{\theta}(t) = 0.$$

In particular, we want to find a gain matrix $F$, such that, for $u(k) = Fx(k)$ and the initial condition $x(0) = [1,\ 0,\ 0,\ 0]^T$,

$$x(k) \ \rightarrow \ 0 \qquad \text{for} \quad k \ \rightarrow \ \infty.$$

In addition, we have constraints on the input $u(k)$,

$$|u(k)| \leq 100, \qquad \forall\, k,$$

since the vehicle is limited in how quickly it can rotate. Furthermore, the angle $x_3(k)$ is constrained by

$$|x_3(k)| = |\theta(k\tau)| \leq \frac{\pi}{6}, \qquad \forall\, k,$$

guaranteeing that the linearization is reasonably accurate and also that $a(t) = 10/\left(\cos\left(\theta(t)\right)\right)$ is feasible. Finally, our sampling period is $\tau = 1/50$.

By appropriately choosing the matrices $Q$ and $R$ and using the `dare` function in MATLAB, find a feedback control strategy $u(k) = Fx(k)$, which brings the system to within

$$|x_i(k)| \leq 0.01, \qquad i = 1,\ 2,\ 3,\ 4, \tag{3}$$

as quickly as possible while satisfying the constraints.[1]

This will be an iterative process and numerical in nature. In particular, there is no direct way to capture the constraints in the LQR design or to minimize the time, it takes to get within a tolerance of the destination. You will have to find the solution iteratively by modifying the matrices $Q$ and $R$ based on your simulation results.

*Find a good strategy to solve this problem. What is your best set of parameters $Q$, $R$? And what is the resulting $F$ and $T$? Show plots illustrating the performance of your quadrocopter.*

[1]Note that the time to be minimized is the time at which conditions (3) are fulfilled for the first time.

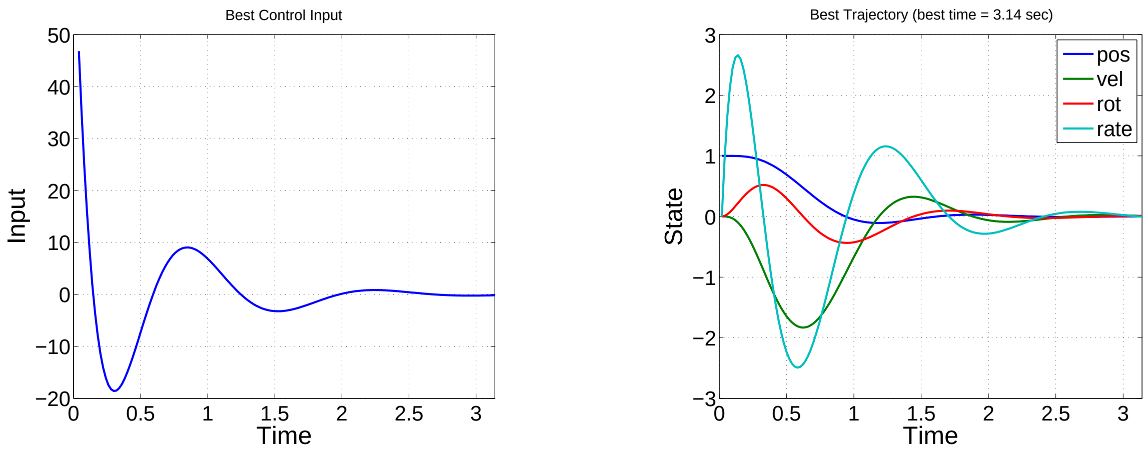
## Problem 4 (Finite Horizon LQR)

Using the results from Problem 3 as a starting point, how much you can improve your design by using a *finite horizon LQR*?

*Create plots showing the improvements and explain how you got your solution. What is your best choice for* $Q_k$, $R_k$ *and your minimum time* $T$*?*

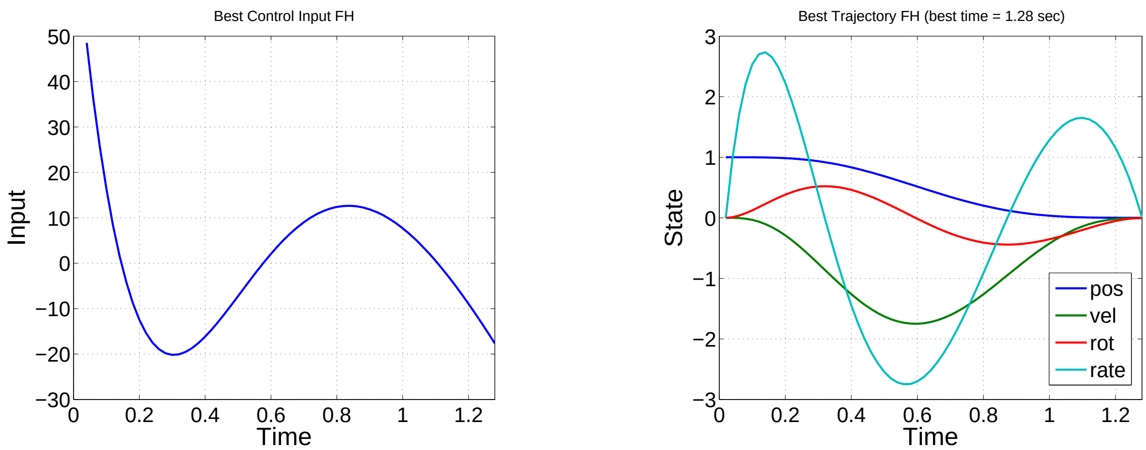
### "Who can do best?"

Prof. D'Andrea's results:




(a) The input. (b) The state trajectories.

*Figure 2: Results for the infinite horizon LQR.*




(a) The input. (b) The state trajectories.

*Figure 3: Results for the finite horizon LQR.*

## Problem 5 (BERTSEKAS, p. 211, exercise 4.22)

Consider a situation involving a blackmailer and his victim. In each period the blackmailer has a choice of: a) Accepting a lump sum payment of $R$ from the victim and promising not to blackmail again. b) Demanding a payment of $u$, where $u \in [0, 1]$. If blackmailed, the victim will either: 1) Comply with the demand and pay $u$ to the blackmailer. This happens with probability $1 - u$. 2) Refuse to pay and denounce the blackmailer to the police. This happens with probability $u$. Once known to the police, the blackmailer cannot ask for any more money. The blackmailer wants to maximize the expected amount of money he gets over $N$ periods by optimal choice of the payment demand $u_k$. (Note that there is no additional penalty for being denounced to the police). Write a DP algorithm and find the optimal policy.

## Problem 6 (BERTSEKAS, p. 212, exercise 4.23)

The Greek mythological hero Theseus is trapped in King Minos' Labyrinth maze. He can try each day one of $N$ passages. If he enters passage $i$ he will escape with probability $p_i$, he will be killed with probability $q_i$, and he will determine that the passage is a dead end with probability $(1-p_i-q_i)$, in which case he will return to the point from which he started. Use an interchange argument to show that trying passages in order of decreasing $p_i/q_i$ maximizes the probability of escape within $N$ days.

## Sample Solutions

### Problem 1 (Solution)

Consider only small deviations of the angle $\theta$ from 0. A Taylor series expansion about 0 gives

$$\tan\theta \approx \theta.$$

The linearized equations are:

$$\begin{aligned}
\ddot{y}(t) &= -10\,\theta(t) \\
\ddot{\theta}(t) &= q(t).
\end{aligned}$$

### Problem 2 (Solution)

With given definitions, the time-discrete quadrocopter dynamics are obtained by integration. For $k\tau \le t \le (k+1)\tau$,

$$\begin{aligned}
\int_{k\tau}^{t} \ddot{\theta}(\xi) d\xi = \dot{\theta}(t) - \dot{\theta}(k\tau) &= \dot{\theta}(t) - x_4(k) \\
&= \int_{k\tau}^{t} q(t) dt \\
&= u(k)\,(t - k\tau)
\end{aligned}$$

$$\Rightarrow \quad \dot{\theta}(t) = u(k)(t - k\tau) + x_4(k) \tag{4}$$

$$\begin{aligned}
\int_{k\tau}^{t} \big(u(k)(\xi - k\tau) + x_4(k)\big) d\xi &= \frac{1}{2}u(k)(t - k\tau)^2 + x_4(k)(t - k\tau) \\
&= \int_{k\tau}^{t} \dot{\theta}(\xi) d\xi
\end{aligned}$$

$$\Rightarrow \quad \theta(t) = x_3(k) + x_4(k)(t - k\tau) + \frac{1}{2}u(k)(t - k\tau)^2 \tag{5}$$

$$\begin{aligned}
\int_{k\tau}^{t} -10\,\theta(\xi) d\xi &= -10\left[x_3(k)(t - k\tau) + \frac{1}{2}x_4(k)(t - k\tau)^2 + \frac{1}{6}u(k)(t - k\tau)^3\right] \\
&= \int_{k\tau}^{t} \ddot{y}(\xi) d\xi = \dot{y}(t) - x_2(k)
\end{aligned}$$

$$\Rightarrow \quad \dot{y}(t) = x_2(k) - 10\left[x_3(k)(t - k\tau) + \frac{1}{2}x_4(k)(t - k\tau)^2 + \frac{1}{6}u(k)(t - k\tau)^3\right] \tag{6}$$

$$\begin{aligned}
\int_{k\tau}^{t} \dot{y}(\xi) d\xi &= x_2(k)(t - k\tau) - 10\left[\frac{1}{2}x_3(k)(t - k\tau)^2 + \frac{1}{6}x_4(k)(t - k\tau)^3 + \frac{1}{24}u(k)(t - k\tau)^4\right] \\
&= y(t) - x_1(k)
\end{aligned}$$

$$\Rightarrow \quad y(t) = x_1(k) + x_2(k)(t-k\tau) - 10\left[\frac{1}{2}x_3(k)(t-k\tau)^2 + \frac{1}{6}x_4(k)(t-k\tau)^3 + \frac{1}{24}u(k)(t-k\tau)^4\right] \tag{7}$$

We are interested in $x_i(k+1)$, $i = 1,2,3,4$.
From Eq. (4)-(7), we get:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \\ x_4(k+1) \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & \tau & \frac{-10\tau^2}{2} & \frac{-10\tau^3}{6} \\ 0 & 1 & -10\tau & \frac{-10\tau^2}{2} \\ 0 & 0 & 1 & \tau \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{A} x(k) + \underbrace{\begin{bmatrix} \frac{-10\tau^4}{24} \\ \frac{-10\tau^3}{6} \\ \frac{\tau^2}{2} \\ \tau \end{bmatrix}}_{B} u(k)\,.$$

## Problem 3 (Solution)

Infinite horizon LQR problem:

- System

$$x_{k+1} = Ax_k + Bu_k \qquad k = 0,1,2,3,\ldots$$

- Cost

$$\sum_{k=0}^{\infty} x_k^T Q x_k + u_k^T R u_k \qquad Q \geq 0, R > 0$$

- Optimal control

$$K = A^T\big(K - KB(R + B^T KB)^{-1}B^T K\big)A + Q \qquad \text{(Riccati Equation)}$$
$$F = -(R + B^T KB)^{-1}B^T KA$$

  and

$$u_k = Fx_k$$

*Interpretation:* $Q$ penalizes large values $x$, $R$ penalizes large values $u$.

One possible strategy is choosing $R = 1$ and only penalizing the $y$ position; that is,

$$Q_{inf\,hor} = \begin{pmatrix} q_0 & 0 & \cdots & 0 \\ 0 & 0 & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & & 0 \end{pmatrix}. \tag{8}$$

Find the optimal $q_0$.

You get

$$q_0 = 2880, \quad T_{inf\,hor} = 3.14$$
$$F_{inf\,hor} = \begin{bmatrix} 45.6615 & 25.5039 & -71.2248 & -11.7451 \end{bmatrix},$$

where $T_{inf\,hor}$ is the time at which conditions (6) are satisfied for the first time.

→ A MATLAB code example can be found at the end of this problem set.

## Problem 4 (Solution)

Finite horizon LQR problem:

- System

$$x_{k+1} = Ax_k + Bu_k \qquad k = 0, 1, \ldots, N-1$$

- Cost

$$\sum_{k=0}^{N-1} (x_k^T Q x_k + u_k^T R u_k) + x_N^T Q_N x_N$$

- Optimal control

$$\begin{aligned} K_N &= Q_N \\ K_k &= A^T \big(K_{k+1} - K_{k+1} B (B^T K_{k+1} B + R_k)^{-1} B K_{k+1}\big) A + Q_k \\ F_k &= -(B^T K_{k+1} B + R_k)^{-1} B^T K_{k+1} A \end{aligned}$$

  and

$$u_k = F_k x_k$$

*Interpretation:* $K_N = Q_N$ (starting value) represents the weight on the final state $x_N$.

One possible strategy is

- choosing $Q_k = \alpha\, Q_{inf\, hor}$ , see Eq. (8), $\quad 0 \le \alpha \le 1, \quad k = 0, 1, \ldots, N-1$
- iterating on the time horizon $0 \le T_{fin\, hor} \le T_{inf\, hor}$
- keeping $R = 1$ as before
- starting with $K_N = \beta K_{inf\, hor}$
- iterating on $T_{fin\, hor}$ and $\alpha$, $\beta$

This results in

$$\alpha = 0.95, \quad \beta = 4096, \quad T_{fin\, hor} = 1.28$$

→ A MATLAB code example can be found at the end of this problem set.

## Problem 5 (Optimal Stopping Problem)

Transform the problem to an optimal stopping problem:

- Time horizon

  $N$ periods

- State

$$x_k = \begin{pmatrix} B \\ T \end{pmatrix}$$

with

| | |
|---|---|
| B: | blackmailing (blackmailer has not accepted lump sum payment and has not been denounced to the police), |
| T: | termination (result of accepting the lump sum payment or of denouncement to the police |

- Input

$$u_k \in [0,1] \cup \{-1\}$$

corresponds to the decision of the blackmailer:

| | |
|---|---|
| $u_k = -1$ | accept lump sum payment |
| $u_k \in [0,1]$ | demand a payment of $u_k$ |

- Dynamics

$$\begin{aligned} x_{k+1} &= B && \text{if } x_k = B \text{ and } u_k \in [0,1] \text{ and } w_k \neq 0 \\ x_{k+1} &= T && \text{if } x_k = T \text{ or} \\ & && \text{if } x_k = B \text{ and } u_k = -1 \text{ or} \\ & && \text{if } x_k = B \text{ and } u_k \in [0,1] \text{ and } w_k = 0\,, \end{aligned}$$

where the random variable $w_k$ is defined by $w_k \in \{0, u_k\} = [0,1]$,

$$\begin{aligned} &P(w_k = 0) = u_k \\ &P(w_k = u_k) = 1 - u_k \end{aligned}$$

assuming $u_k \in [0,1]$ is demanded.

$\Rightarrow$ initial condition: $x_0 = B$

- Cost

$$\begin{aligned} g_N(x_N) &= 0 && \text{for both } x_N = B \text{ and } x_N = T \\ g_k(x_k, u_k, w_k) &= \begin{cases} R & \text{if } u_k = -1 \text{ and } x_k = B \\ w_k & \text{if } u_k \in [0,1] \text{ and } x_k = B \\ 0 & \text{if } x_k = T \end{cases} \end{aligned}$$

**Apply the DP Algorithm**

Nth stage:

$$J_N(x_N) = 0 \quad \to \text{last decision made at stage } N-1$$

Cost-to-go if $x_k = T$:

$$J_k(T) = 0 \quad \forall\, k = 1, 2, \ldots, N$$

Cost-to-go if $x_k = B$

I) (N-1)th stage:

$$\begin{aligned} J_{N-1}(B) &= \max_{\substack{u_{N-1} \in [0,1] \\ u_{N-1} = -1}} \mathrm{E}\big(g_{N-1}(x_{N-1}, u_{N-1}, w_{N-1}) + J_N(x_N)\big) \\ &= \max\big\{R, \max_{u_{N-1} \in [0,1]} (\underbrace{u_{N-1}(1 - u_{N-1})}_{L(u_{N-1})} + 0 \cdot u_{N-1})\big\} \end{aligned}$$

Find maximizing $u_{N-1}$:

$$\frac{\partial L}{\partial u_{N-1}} = 1 - 2u_{N-1} = 0 \quad \Leftrightarrow \quad u_{N-1} = \frac{1}{2}$$
$$\frac{\partial^2 L}{\partial u_{N-1}^2} = -2 < 0 \quad \Rightarrow \quad \text{maximum}$$

$$\Rightarrow \quad J_{N-1}(B) = \max\left\{R, \frac{1}{4}\right\}$$
$$= \begin{cases} R & \text{if } R > \frac{1}{4} \text{ with } \mu^*_{N-1}(B) = -1 \\ \frac{1}{4} & \text{if } R \leq \frac{1}{4} \text{ with } \mu^*_{N-1}(B) = \frac{1}{2} \end{cases} \tag{9}$$

II) (N-2)th stage:

a)

$$J_{N-2}(B) = \max_{\substack{u_{N-2}\in[0,1] \\ u_{N-2}=-1}} \mathrm{E}\big(g_{N-2}(x_{N-2}, u_{N-2}, w_{N-2}) + J_{N-1}(x_{N-1})\big)$$
$$= \max\Big\{R + J_{N-1}(T),$$
$$\max_{u_{N-2}\in[0,1]} \big[(1 - u_{N-2})\,(u_{N-2} + J_{N-1}\,(B)) + u_{N-2}\,(0 + J_{N-1}\,(T))\big]\Big\}$$

Note that after chosen $R$, blackmailing is terminated (first option). Otherwise, there is a probability of $u_{N-2}$ for denouncement to the police.

Find maximizing $u_{N-2}$:

$$\frac{\partial L}{\partial u_{N-2}} = 1 - 2u_{N-2} - J_{N-1}(B) = 0 \quad \Leftrightarrow \quad u_{N-2} = \frac{1 - J_{N-1}(B)}{2}$$
$$\frac{\partial^2 L}{\partial u_{N-2}^2} = -2 < 0 \quad \Rightarrow \quad \text{maximum, concave function}$$

Considering the input constraints $u_k \in [0, 1]$, we get

$$u_{N-2} = \begin{cases} \frac{1-J_{N-1}(B)}{2} & \text{if } J_{N-1}(B) < 1 \\ 0 & \text{if } J_{N-1}(B) \geq 1 \end{cases} \tag{10}$$

Note that with (9)

$$R < 1 \quad \Rightarrow \quad J_{N-1}(B) < 1$$
$$R \geq 1 \quad \Rightarrow \quad J_{N-1}(B) = R \geq 1$$

b)

$$J_{N-2}(B) = \begin{cases} \max(R, J_{N-1}(B)) = R, & \text{for } R \geq 1 \\ \max\Big(R, \Big(\frac{1+J_{N-1}(B)}{2}\Big)^2\Big), & \text{for } R < 1 \end{cases} \tag{11}$$

The second equation can be simplified since $J_{N-1} \geq R$

$$\left(\frac{1 + J_{N-1}(B)}{2}\right)^2 \geq \left(\frac{1+R}{2}\right)^2 \geq R$$

$$J_{N-2}(B) = \begin{cases} R & \text{if } R \geq 1 \\ \left(\frac{1+J_{N-1}(B)}{2}\right)^2 & \text{if } R < 1 \end{cases}$$

$$\mu^*_{N-2}(B) = \begin{cases} u_{N-2} = 0 \quad \text{or} \quad u_{N-2} = -1 & \text{if } R \geq 1 \\ u_{N-2} = \frac{1-J_{N-1}(B)}{2} & \text{if } R < 1 \end{cases}$$

III) Assumption:

$$J_k(B) = \begin{cases} R & \text{if } R \geq 1 \\ \left(\frac{1+J_{k+1}(B)}{2}\right)^2 < 1 \quad (!) & \text{if } R < 1 \end{cases}$$

$$\mu^*_k(B) = \begin{cases} 0 \text{ or } -1 & \text{if } R \geq 1 \\ \frac{1-J_{k+1}(B)}{2} & \text{if } R < 1 \end{cases} \tag{12}$$

for $k = 0, 1, \ldots, N-1$

Additionally, assume

$$J_k(B) \geq R. \tag{13}$$

IV) *Proof.*

Proof by Induction:
1) The relationship (12) holds for $k = N-2$.
2) Assume (12) is true for $k$.
3) Prove that (12) also holds for $k-1$.

We know,

$$J_{k-1}(B) = \max\left\{R, \max_{u_{k-1} \in [0,1]} \big((1-u_{k-1})(u_{k-1} + J_k(B))\big)\right\}. \tag{14}$$

With before's arguments, maximizing $u_{k-1}$ is given by

$$u_{k-1} = \frac{1 - J_k(B)}{2}.$$

Distinguish as in Eq. (10). With (12)

$$\begin{aligned} R < 1 \quad &\Rightarrow \quad J_k(B) < 1 \\ R \geq 1 \quad &\Rightarrow \quad J_k(B) \geq 1 \end{aligned}$$

Using (13) and proceeding as shown above, we get similar equations as (11) and finally

$$J_{k-1}(B) = \begin{cases} R & \text{if } R \geq 1 \\ \left(\frac{1+J_k(B)}{2}\right)^2 & \text{if } R < 1, \end{cases}$$

$$\mu^*_{k-1}(B) = \begin{cases} 0 \text{ or } -1 & \text{if } R \geq 1 \\ \frac{1-J_k(B)}{2} & \text{if } R < 1. \end{cases}$$

From the maximation (14), we know that $J_{k-1}(B) \geq R$ and, with $J_k(B) < 1$ if $R < 1$, see Eq. (12), we conclude

$$\left(\frac{1+J_k(B)}{2}\right)^2 < \left(\frac{1+1}{2}\right)^2 = 1.$$

$\square$

In brief, if $R \geq 1$, the blackmailer should accept $R$ right a the beginning, otherwise, he is better off demanding

$$\mu^*_k(B) = \frac{1-J_{k+1}(B)}{2}, \quad k = 0, 1, 2, \ldots, N-2$$

where $J_{k+1}(B)$ results from the recursion

$$J_k(B) = \left(\frac{1+J_{k+1}(B)}{2}\right)^2$$

with initial condition

$$J_{N-1}(B) = \max\left\{R, \frac{1}{4}\right\}$$

the last demand is

$$\mu^*_{N-1}(B) = -1 \qquad \text{if } R > \frac{1}{4}$$
$$\mu^*_{N-1}(B) = \frac{1}{2} \qquad \text{if } R \leq \frac{1}{4}.$$

## Problem 6 (Interchange Argument)

- $N$ different passages, Theseue can try each path only once
- define a sequence of attempted passages:

$$L = \{i_1, i_2, \ldots, i_N\}$$

- introduce rewards:
  - dead end on passage $i_k$: $R_{ik} = 0$
  - killed on passage $i_k$: $R_{ik} = 0$
  - *first* escape on passage $i_k$: $R_{ik} = 1$
  - after having been killed or having been escaped on passage $i_k$, all rewards: $R_{im} = 0, \quad m > k$

For a sequence $L$,

$$\begin{aligned}\text{E(reward of } L) &= p_1 + (1-p_1-q_1)p_2 + (1-p_1-q_1)(1-p_2-q_2)p_3 + \cdots + \prod_{i=1}^{N-1}(1-p_i-q_i)p_N \\ &\triangleq \text{Probability of escape within } N \text{ days}\end{aligned}$$

with $p_i$: P(escape on i-th passage), $(1-p_i-q_i)$: P(dead end on i-th passage), $\prod_{i=1}^{k-1}(1-p_i-q_i)p_k$: P(escape on $k$th passage)

Use interchange argument:

- Let $L = \{i_1, i_2, \ldots, i_{k-1}, i, j, i_{k+2}, \ldots, i_N\}$ be an optimal ordering.
- Let $\bar{L} = \{i_1, i_2, \ldots, i_{k-1}, j, i, i_{k+2}, \ldots, i_N\}$ be the swapped ordering.

$$\begin{aligned}\text{E(reward of } L) = \text{E}\big(&\text{reward of } (i_1, i_2, \ldots, i_{k-1})\big) \\ &+ (1-p_1-q_1)(1-p_2-q_2)\cdots(1-p_{k-1}-q_{k-1})p_i \\ &+ (1-p_1-q_1)(1-p_2-q_2)\cdots(1-p_{k-1}-q_{k-1})(1-p_i-q_i)p_j \\ &+ (1-p_1-q_1)(1-p_2-q_2)\cdots(1-p_{k-1}-q_{k-1})(1-p_i-q_i)(1-p_j-q_j) \\ &\quad \cdot \text{E}\big(\text{reward of } (i_{k+2}, \ldots, i_N)\big)\end{aligned}$$

$$\begin{aligned}\text{E(reward of } \bar{L}) = \text{E}\big(&\text{reward of } (i_1, i_2, \ldots, i_k - 1)\big) \\ &+ (1-p_1-q_1)(1-p_2-q_2)\cdots(1-p_{k-1}-q_{k-1})p_j \\ &+ (1-p_1-q_1)(1-p_2-q_2)\cdots(1-p_{k-1}-q_{k-1})(1-p_j-q_j)p_i \\ &+ (1-p_1-q_1)(1-p_2-q_2)\cdots(1-p_{k-1}-q_{k-1})(1-p_j-q_j)(1-p_i-q_i) \\ &\quad \cdot \text{E}\big(\text{reward of } (i_{k+2}, \ldots, i_N)\big)\end{aligned}$$

$$\begin{aligned}\text{E(reward of } L) &\geq \text{E(reward of } \bar{L}) \\ p_i + (1-p_i-q_i)p_j &\geq p_j + (1-p_j-q_j)p_i \\ -q_i p_j &\geq -q_j p_i \\ \frac{p_j}{q_j} &\leq \frac{p_i}{q_i}\end{aligned}$$

Conclusion:
Try passage with highest $\frac{p_i}{q_i}$ first and then, choose passages in the order of decreasing $\frac{p_i}{q_i}$.

```
clear

%%
%% 2D Quad Copter problem, for DP class.  Only concentrate on horizontal dynamics.
%%

TS = 0.02; % Sampling period, s
G = 10.0;  % Acceleration due to gravity, m/s/s

% The thresholds for determining if a maneuver is finished.
% (pos (m), posDot (m/s), rot (rad), rotDot (rad/s)
THRESH_VEC = [0.01; 0.01; 0.01; 0.01];

ANGLE_ACC_LIM = 100;  % maximum angular acceleration rad/s/s
ANGLE_LIM = pi/6;     % maximum angle deviation, rad/s

%%%%%%%%%%%%%%
%% Linearized equations of motion
%%%%%%%%%%%%%

% The state x = (y,yDot, r, rDot), where y is the horizontal position, r is the angle
% of the vehicle to horizontal.
% y'' = -10r
% r'' = u
A = [ 1 TS -10*(TS^2)/2 -10*(TS^3)/6; ...
     0 1 -10*TS -10*(TS^2)/2; ...
     0 0 1 TS; ...
     0 0 0 1];
B = [(TS^4)/24; (TS^3)/6; (TS^2)/2; TS];

% Initial condition
x0 = [1;0;0;0];

%%%%%%%%%%%%%%%
%% Set up LQR problem, infinite horizon
%%%%%%%%%%%%%%%

% without loss of generality, penalize control effort by 1
R = 1;

% Only penalize the position of the vehicle. The angle is indirectly penalized
% by the control efforts. This only leaves one parameter to optimize over.
% In fact, by manually playing around with the
% cost matrix Q, it seems that penalizing the velocity helps quite a bit, since this↙
decreases
% oscillations at the end (damping), causing the system to reach the required↙
tolerances faster. But
% we won't bother with that here.
Q = zeros(4,4);

%%%%%%%%%%%%%%%
%% Solve infinite horizon LQR problem, iterate until we hit limits
%%%%%%%%%%%%%%%
qMin = 0;
qMax = inf;
```

```
q = 1;
bestTime = inf;

% Outer loop, for optimizing q.  The objective here is to find a q that results in↙
the constraints
% being satisfied, and that achieves the fastest trajectory.
while (1)
    Q(1,1) = q;

    % Solve the DARE
    [K,L,Fneg] = dare(A,B,Q,R);

    k = 1;
    x = x0;
    u = [];
    validTrajectory = 1;

    % Build the trajectory, and check that it satisfies the constraints.
    while (1)

        % Control effort
        u(k) = -Fneg*x(:,k);

        % Have we violated our angular acceleration constraint?
        if abs(u(k)) > ANGLE_ACC_LIM
            validTrajectory = 0;
            break;
        end

        % Update the state, the trajectory is valid so far.
        x(:,k+1) = A*x(:,k) + B*u(k);
        k = k+1;

        % Have we violated our angle constraint?
        if (abs(x(3,k)) > ANGLE_LIM)
            validTrajectory = 0;
            break;
        end

        % Check to see if we are done
        if ( abs(x(:,k)) < THRESH_VEC)
            break;
        end
    end

    % If we have a valid trajectory, check if it is better than the best one to date;
    % if it is, we want to increase q and try again.  If it isn't, we are done.  Note↙
that
    % we include in the check if we haven't hit our upper bound yet; if we haven't,↙
we should
    % continue, irrespective if our time was better or not (it could mean that it is↙
decreasing
    % very slowly, because we are no-where close to being aggressive enough).
    if (validTrajectory)
        if (k < bestTime) || (qMax == inf)
```

```
            if (k < bestTime)
                bestTime = k;
                bestStateTrajectory = x;
                bestControlInput = u;
                bestQ = Q;
            end

            qMin = q;
            if (qMax == inf)
                q = 2*q;
            else
                q = (qMax + qMin)/2;
            end
        else
            break;
        end
    else
        % We don't have a valid trajectory.  We need to penalize our position less,↙
so that we
        % are less agressive with our maneuver;
        qMax = q;
        q = (qMax + qMin)/2;
    end
end

% Plot the results
figure(1)
plot((1:bestTime)*TS, bestStateTrajectory');
xlabel('Time')
title(['Best Trajectory (best time = ',num2str(bestTime*TS),' sec)']);
legend('pos','vel','rot','rate');
grid

figure(2)
plot((2:bestTime)*TS, bestControlInput);
xlabel('Time')
title('Best Control Input');
grid

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Finite Horizon Problem
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% We need a final cost for the state, which is the starting point for the iteration.↙
Picking the
% steady state solution will clearly give the same result as the infinite horizon↙
problem.  Picking
% 0 as the start of the iteration will result in a less agressive maneuver near the↙
end, which we
% clearly don't want.  So intuitively, we want to initialize with something that is↙
greater than the
% steady state solution.

% Give ourselves some wiggle room with Q.  Make it slightly smaller, which will↙
result in a less
```

```
% aggressive maneuver at the beginning of the trajectory (once we reach the steady↙
state gain); this
% is necessary since the aggressive portions at the end of the maneuver will trickle↙
down to what
% happens at the beginning, for short enough time horizons.
Qfh = bestQ * 0.95;

% Solve the DARE for the steady-state solution
[K,L,Fneg] = dare(A,B,Qfh,R);

% The interval where we will search for the fastest trajectory.
tMin = 0;
tMax = bestTime;
tm = tMax;

% Keep on looping to find the best time.  We use a bisection algorithm.
while(1)

    minK = 0;
    maxK = inf;
    muxK = 1;

    % Keep on looping until we find a valid trajectory.  In particular, if the↙
trajectory is not
    % valid, need to decrease our final cost, otherwise increase.  Use a bisection↙
algorithm.
    while (1)

        Kfh{1} = muxK*K;

        % Construct time varying feedback gains
        for l = 1:tm
            Kfh{l+1} = A'*(Kfh{l} - Kfh{l}*B*inv(R + B'*Kfh{l}*B)*B'*Kfh{l})*A + Qfh;
            Ffh{l+1} = -inv(R + B'*Kfh{l}*B)*B'*Kfh{l}*A;
        end

        k = 1;
        x = x0;
        u = [];
        validTrajectory = 1;
        finishedTrajectory = 0;

        for k = 1:tm

            % Control effort
            u(k) = Ffh{tm - k + 2} *x(:,k);

            % Have we violated our angular acceleration constraint?
            if abs(u(k)) > ANGLE_ACC_LIM
                validTrajectory = 0;
                break;
            end

            % Update the state, the trajectory is valid so far.
            x(:,k+1) = A*x(:,k) + B*u(k);
```

```
            % Have we violated our angle constraint?
            if (abs(x(3,k+1)) > ANGLE_LIM)
                validTrajectory = 0;
                break;
            end

            % Check to see if we are done
            if ( abs(x(:,k+1)) < THRESH_VEC)
                finishedTrajectory = 1;
                break;
            end
        end

        % If we managed to finish the trajectory, we are done.
        if (finishedTrajectory)
            break;
        end

        % If we did not finish the trajectory, but it was valid, it means that we can↙
be more
        % aggressive by increasing the final cost
        if (finishedTrajectory == 0) && (validTrajectory == 1)
            minK = muxK;
            if (maxK == inf)
                muxK = 2*muxK;
            else
                muxK = (maxK + minK)/2;
            end
        end

        % If we did not finish the trajectory, and it was not valid, we need to be↙
less aggressive
        if (finishedTrajectory == 0) && (validTrajectory == 0)
            maxK = muxK;
            muxK = (maxK + minK)/2;
        end

         % Quit if we are too close
         if (maxK - minK)/maxK < 0.01
            break;
         end

    end

    % If we managed to finish the trajectory, can decrease the time
    if (finishedTrajectory)

        bestTimeFh = tm+1;
        bestStateTrajectoryFh = x;
        bestControlInputFh = u;

        tMax = tm;
        tmNew = round((tMax + tMin)/2);
    else
```

```
        % If we didn't finish the trajectory, increase the time
        tMin = tm;
        tmNew = round((tMax + tMin)/2);

    end

    % If the time did not change, we are done
    if (tmNew == tm)
        break;
    else
        tm = tmNew;
    end
end

% Plot the results
figure(3)
plot((1:bestTimeFh)*TS, bestStateTrajectoryFh');
xlabel('Time')
title(['Best Trajectory FH (best time = ',num2str(bestTimeFh*TS),' sec)']);
legend('pos','vel','rot','rate');
grid

figure(4)
plot((2:bestTimeFh)*TS, bestControlInputFh);
xlabel('Time')
title('Best Control Input FH');
grid
```