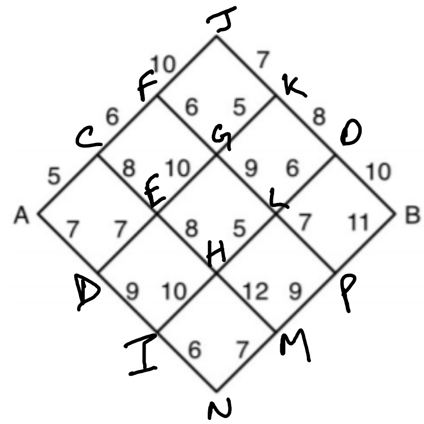
PS 2 Problem 1

Sunday, April 14, 2019 2:51 AM

a)



$J_B = 0$

$J_O = 10$ $\quad J_K = 8 + J_O = 18$ $\quad J_J = 7 + J_K = 25$

$J_P = 11$ $\quad J_M = 9 + J_P = 20$ $\quad J_N = 7 + J_M = 27$

$J_L = \min(6 + J_O, 7 + J_P) = \min(16, 18) = 16$

$u^*: L \to O$

$J_H = \min(5 + J_L, 12 + J_M) = \min(21, 32) = 21$

$u^*: H \to L$

$J_G = \min(5 + J_K, 9 + J_L) = \min(23, 25) = 23$

$u^*: G \to K$

$J_F = \min(10 + J_J, 6 + J_G) = \min(35, 29) = 29$

$u^*: F \to G$

$J_E = \min(10 + J_G, 8 + J_H) = \min(33, 29) = 29$

$u^*: E \to H$

$J_I = \min(10 + J_H, 6 + J_N) = \min(31, 33) = 31$

$u^*: I \to H$

$J_C = \min(6 + J_F, 8 + J_E) = \min(35, 37) = 35$

$u^*: C \to F$

$J_D = \min(7 + J_E, 9 + J_I) = \min(36, 40) = 36$

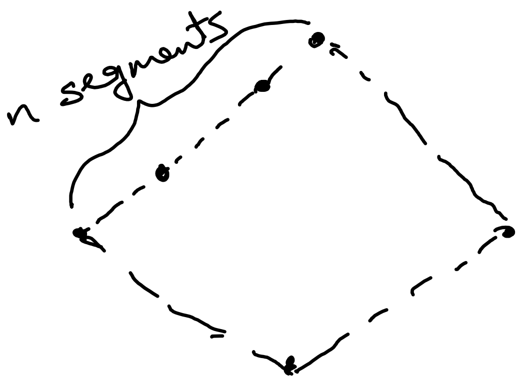
$u^*: D \to E$

$J_A = \min(5 + J_C, 7 + J_D) = \min(40, 43) = 40$

$u^*: A \to C$

$$\boxed{\begin{array}{l}\text{Shortest path} \\ A \xrightarrow[5]{} C \xrightarrow[6]{} F \xrightarrow[6]{} G \xrightarrow[5]{} K \xrightarrow[8]{} O \xrightarrow[10]{} B \\ \text{Cost} = 40\end{array}}$$

b) For DP, we need 1 computation per node (except terminal node $J = 0$).



$\Rightarrow (n+1)$ nodes on each line & $(n+1)$ lines

Total # of nodes $= (n+1)^2$

DP evals $= (n+1)^2 - 1$ ← no eval for terminal node

$= n(n+2)$

$3 \times 5 = 15$

For exhaustive search, # computations is equal to # of possible routes.

Can think of this as a sequence of n up moves & n down moves

$$\left.\begin{array}{l} UUUDDD \\ UDUDUD \\ \vdots \end{array}\right\} {}^{6}C_{3} \text{ or } {}^{2n}C_{n} \text{ routes}$$

exhaustive search evals

$$\boxed{\begin{array}{l} \therefore \ \#DP \text{ evals} = n(n+2) \\ \#\text{exhaustive search evals} = {}^{2n}C_{n} = \dfrac{(2n)!}{n!\,n!} \end{array}}$$

PS 2 Problem 2

Wednesday, April 17, 2019 12:31 AM

$\dot{x}_1(t) = u_1(t)$

$\dot{x}_2(t) = u_2(t)$

$\|u(t)\| = 1 \quad \Rightarrow u_1^2(t) + u_2^2(t) = 1$

Starts at $x(0)$

Ends at $x(T)$

$\min \int_0^T r(x(t))\,dt$

$r(\cdot) \geq 0$ and continuous

From $\bar{x} = (\bar{x}_1, \bar{x}_2)$

we can go to $(\bar{x}_1 + \Delta, \bar{x}_2)$

$(\bar{x}_1 - \Delta, \bar{x}_2)$

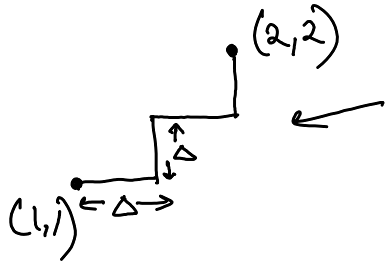
$(\bar{x}_1, \bar{x}_2 + \Delta)$

$(\bar{x}_1, \bar{x}_2 - \Delta)$

cost $r(\bar{x})\Delta$

Consider going from $(1,1)$ to $(2,2)$

With this discretization, let $\Delta = 0.5$ at first



shortest path

Here, we need $2\,\Delta$ steps in (+)ve x_1 dir^n

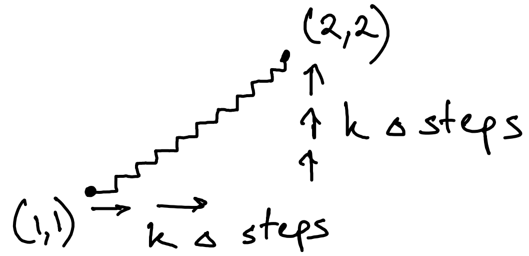
and $2\,\Delta$ steps in (+)ve x_2 dir^n

Assume $r(\cdot)$ uniform everywhere $= r^*$

Cost for this path $= 2\Delta r^* + 2\Delta r^*$

$= 4(0.5)r^* = 2r^*$

Let's make Δ smaller $= 1/k$ where k large



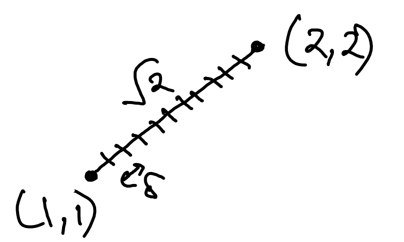
Again, we need k Δ steps in (+)ve x_1 &
k Δ steps in (+)ve x_2

$$\text{Cost} = 2k(\Delta)\, r^{\alpha} = 2r^{\alpha}$$

Even in the limit $\Delta \to 0$ $k \to \infty$

$$\text{cost} = 2r^{*}$$

However, optimal cost of original problem is a straight line of length $\sqrt{2}$.



The optimal path involves small steps δ in the $\left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)$ direction such that $\|u(t)\| = 1$.

$$\text{Cost of this path} = \delta\left(\frac{\sqrt{2}}{\delta}\right) r^{\alpha} = \sqrt{2}\, r^{\alpha}$$

step length

\# of steps

True optimal $= \sqrt{2}\, r^{*} <$ Cost of discretized $= 2r^{\alpha}$

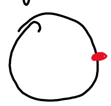
$\therefore$ This is a bad discretization of the original problem.

PS 2 Problem 3

Tuesday, April 16, 2019 4:32 PM

N side polygon w/ maximal perimeter

Placement of 1^{st} pt doesn't matter, say at 0°



2^{nd} pt placed at θ

"perimeter" $= c$

$c^2 = r^2 + r^2 - 2r^2\cos\theta$ by law of cosines

$c^2 = 2r^2(1-\cos\theta) = 4r^2\sin^2\theta/2$

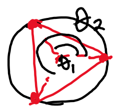
gradient $\Rightarrow \sin\theta = 0$ or $\sin\theta/2 = 1$

$\hookrightarrow$ c max for $\cos\theta = -1 \Rightarrow \theta = \pi$

$\Rightarrow c = 2r$

 $\leftarrow$ optimal 2^{nd} pt

Given soln for 2 pts, let's find soln for 3 pts



$$\text{perimeter} = 2r\sin\theta_1/2 + 2r\sin\left(\frac{\theta_2-\theta_1}{2}\right) + 2r\sin\left(\frac{2\pi-\theta_2}{2}\right)$$

$\therefore$ For N pts,

$$\text{perimeter} = 2r\sin\theta_1/2 + 2r\sin\frac{\theta_2-\theta_1}{2} + 2r\sin\frac{\theta_3-\theta_2}{2} + \dots$$

$$\dots + 2r\sin\frac{\theta_N-\theta_{N-1}}{2} + 2r\sin\underbrace{\left(\frac{2\pi-\theta_N}{2}\right)}_{=\sin\frac{\theta_N}{2}}$$

Let $\theta_0 = 0$ by def^n

$$\Rightarrow \text{perimeter} = 2r\sin\left(\frac{\theta_N}{2}\right) + \sum_{i=1}^{N} 2r\sin\frac{\theta_i-\theta_{i-1}}{2}$$

$$\boxed{\begin{array}{l}\therefore \text{ objective is to place } \theta_1, \dots \theta_N \text{ such that we maximize} \\ \quad 2r\sin\left(\frac{\theta_N}{2}\right) + \sum_{i=1}^{N} 2r\sin\frac{\theta_i-\theta_{i-1}}{2}\end{array}}$$

b) Additive cost so apply Dynamic Programming

Cost at N^{th} step

$$J_N(\theta_N) = 2r\sin\left(\frac{\theta_N}{2}\right) \overset{\text{optimal}}{=} J^*_N(\theta_N)$$

Cost at $N-1^{th}$ step

$$J_{N-1}(\theta_{N-1}) = 2r\sin\left(\frac{\theta_N}{2}\right) + 2r\sin\left(\frac{\theta_N-\theta_{N-1}}{2}\right)$$

$$J^*_{N-1}(\theta_{N-1}) = \max_{\theta_N}\left(2r\sin\frac{\theta_N}{2} + 2r\sin\left(\frac{\theta_N-\theta_{N-1}}{2}\right)\right)$$

differentiate wrt θ_N & set to 0

$$\Rightarrow \frac{2r}{2}\cos\left(\frac{\theta_N}{2}\right) + \frac{2r}{2}\cos\left(\frac{\theta_N-\theta_{N-1}}{2}\right) = 0$$

$$\Rightarrow \cos\left(\frac{\theta_N}{2}\right) = -\cos\left(\frac{\theta_N-\theta_{N-1}}{2}\right)$$

satisfied if $\frac{\theta_N}{2} = \pi - \left(\frac{\theta_N-\theta_{N-1}}{2}\right)$

$$\Rightarrow \theta_N = \pi + \frac{\theta_{N-1}}{2}$$

$$\Rightarrow J^*_{N-1}(\theta_{N-1}) = 2r\sin\left(\frac{\pi}{2}+\frac{\theta_{N-1}}{4}\right) + 2r\sin\left(\frac{\pi}{2}-\frac{\theta_{N-1}}{4}\right)$$

$$\frac{\pi}{2}+\frac{\theta_{N-1}}{4} = \pi - \left(\frac{\pi}{2}-\frac{\theta_{N-1}}{4}\right)$$

$$\Rightarrow J^*_{N-1}(\theta_{N-1}) = 4r\sin\left(\frac{\pi}{2}-\frac{\theta_{N-1}}{4}\right)$$

$$J^*_{N-2}(\theta_{N-2}) = \max_{\theta_{N-1}}\left[2r\sin\left(\frac{\theta_{N-1}-\theta_{N-2}}{2}\right) + J^*_{N-1}(\theta_{N-1})\right]$$

$$\Rightarrow 2r\frac{1}{2}\cos\left(\frac{\theta_{N-1}-\theta_{N-2}}{2}\right) + 4r\left(-\frac{1}{4}\right)\cos\left(\frac{\pi}{2}-\frac{\theta_{N-1}}{4}\right) = 0$$

$$\Rightarrow \cos\left(\frac{\theta_{N-1}-\theta_{N-2}}{2}\right) = \cos\left(\frac{\pi}{2}-\frac{\theta_{N-1}}{4}\right)$$

satisfied if $\frac{\theta_{N-1}}{2} - \frac{\theta_{N-2}}{2} = \frac{\pi}{2} - \frac{\theta_{N-1}}{4}$

$$\Rightarrow 3\frac{\theta_{N-1}}{4} = \frac{\pi}{2} + \frac{\theta_{N-2}}{2}$$

$$\Rightarrow \theta_{N-1} = \frac{2\pi}{3} + \frac{2\theta_{N-2}}{3}$$

$$\Rightarrow J^*_{N-2}(\theta_{N-2}) = 2r\sin\left(\frac{\pi}{3}-\frac{\theta_{N-2}}{6}\right) + 4r\sin\left(\frac{\pi}{2}-\frac{\pi}{6}-\frac{\theta_{N-2}}{6}\right)$$

$$= 2r\sin\left(\frac{\pi}{3}-\frac{\theta_{N-2}}{6}\right) + 4r\sin\left(\frac{\pi}{3}-\frac{\theta_{N-2}}{6}\right)$$

$$= 6r\sin\left(\frac{\pi}{3}-\frac{\theta_{N-2}}{6}\right)$$

$$J^*_{N-3}(\theta_{N-3}) = \max_{\theta_{N-2}}\left[2r\sin\left(\frac{\theta_{N-2}-\theta_{N-3}}{2}\right) + J^*_{N-2}(\theta_{N-2})\right]$$

Diff wrt $\theta_{N-2} \rightarrow r\cos\left(\frac{\theta_{N-2}-\theta_{N-3}}{2}\right) + 6r\left(-\frac{1}{6}\right)\cos\left(\frac{\pi}{3}-\frac{\theta_{N-2}}{6}\right) = 0$

$$\Rightarrow \frac{\theta_{N-2}}{2} - \frac{\theta_{N-3}}{2} = \frac{\pi}{3} - \frac{\theta_{N-2}}{6}$$

$$\Rightarrow \theta_{N-2}\frac{(2)}{3} = \frac{\pi}{3} + \frac{\theta_{N-3}}{2}$$

$$\Rightarrow \theta_{N-2} = \frac{\pi}{2} + \frac{3\theta_{N-3}}{4}$$

So far, we have

$$\theta_N = \pi + \frac{\theta_{N-1}}{2} \qquad J^*_{N-1}(\theta_{N-1}) = 4r\sin\left(\frac{\pi}{2} - \frac{\theta_{N-1}}{4}\right)$$

$$\theta_{N-1} = \frac{2\pi}{3} + \frac{2\theta_{N-2}}{3} \qquad J^*_{N-2}(\theta_{N-2}) = 6r\sin\left(\frac{\pi}{3} - \frac{\theta_{N-2}}{6}\right)$$

$$\theta_{N-2} = \frac{2\pi}{4} + \frac{3\theta_{N-3}}{4}$$

Pattern emerging.

Let's guess $\theta_{N-l} = \frac{2\pi}{l+2} + \frac{(l+1)\theta_{N-l-1}}{(l+2)}$

$$J^*_{N-k}(\theta_{N-k}) = 2(k+1)r\sin\left(\frac{\pi}{k+1} - \frac{\theta_{N-k}}{2(k+1)}\right)$$

We know this holds for $l=0,1,2$ & $k=1,2$

Let's show that if it holds for k, it also holds for $k+1$

$$J^*_{N-k-1}(\theta_{N-k-1}) = \max_{\theta_{N-k}}\left[2r\sin\left(\frac{\theta_{N-k}-\theta_{N-k-1}}{2}\right) + J^*_{N-k}(\theta_{N-k})\right]$$

diff wrt $\theta_{N-k} \Rightarrow r\cos\left(\frac{\theta_{N-k}-\theta_{N-k-1}}{2}\right) + \frac{2(k+1)r(-1)}{2(k+1)}\cos\left(\frac{\pi}{k+1} - \frac{\theta_{N-k}}{2(k+1)}\right) = 0$

$$\Rightarrow \frac{\theta_{N-k}}{2} - \frac{\theta_{N-k-1}}{2} = \frac{\pi}{k+1} - \frac{\theta_{N-k}}{2(k+1)}$$

$$\Rightarrow \frac{\theta_{N-k}(k+2)}{2(k+1)} = \frac{\pi}{k+1} + \frac{\theta_{N-k-1}}{2}$$

$$\Rightarrow \theta_{N-k} = \frac{2\pi}{k+2} + \frac{(k+1)\theta_{N-k-1}}{k+2}$$

$$\Rightarrow J^*_{N-k-1}(\theta_{N-k-1}) = 2r\sin\left(\frac{\pi}{k+2} - \frac{\theta_{N-k-1}}{2(k+2)}\right) + 2(k+1)r\sin\left(\frac{\pi}{k+1} - \frac{2\pi}{2(k+1)(k+2)} - \frac{(k+1)\theta_{N-k-1}}{2(k+1)(k+2)}\right)$$

$$= (2+2k+2)r\sin\left(\frac{\pi}{k+2} - \frac{\theta_{N-k-1}}{2(k+2)}\right)$$

$$J^*_{N-k-1}(\theta_{N-k-1}) = 2(k+2)\sin\left(\frac{\pi}{k+2} - \frac{\theta_{N-k-1}}{2(k+2)}\right)$$

$\therefore$ if it holds for k, also holds for $k+1$.

By induction, we've proven

$$\theta_{N-k} = \frac{2\pi}{k+2} + \frac{(k+1)\theta_{N-k-1}}{k+2}$$

Assume $\theta_1 = 0$ rad

$$\Rightarrow \theta_2 = \frac{2\pi}{(N-2)+2} + \frac{(N-2+1)\overset{\to 0}{\theta_1}}{(N-2+2)} = \frac{2\pi}{N}$$

$$\theta_3 = \frac{2\pi}{N-3+2} + \frac{N-3+1 \;\; \theta_2}{N-3+2}$$

$$= \frac{2\pi}{N-1} + \frac{N-2}{N-1}\,\frac{2\pi}{N}$$

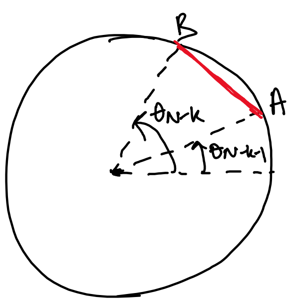
$$= \frac{2\pi}{N(N-1)}(N+N-2) = 2\cdot\frac{2\pi}{N}$$

Similarly, we can again show by induction

$$\theta_K = (k-1)\,\frac{2\pi}{N} \quad \text{for } k=2,3,\ldots N$$

$$\therefore \theta_{diff} = \theta_k - \theta_{k-1} = \frac{2\pi}{N} \quad \text{for } k=2,3,\ldots N$$

Since angles between any 2 adjacent points of polygon is const



Given two angles for points $N-k-1$ & $N-k$, side length

$$AB = l = 2r\sin\frac{\theta_{diff}}{2}$$

$$= 2r\sin\left(\frac{\pi}{N}\right)$$

$\therefore$ This is a regular polygon in order to maximize perimeter

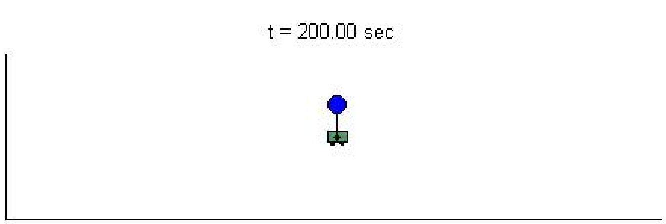
Question 4

Code

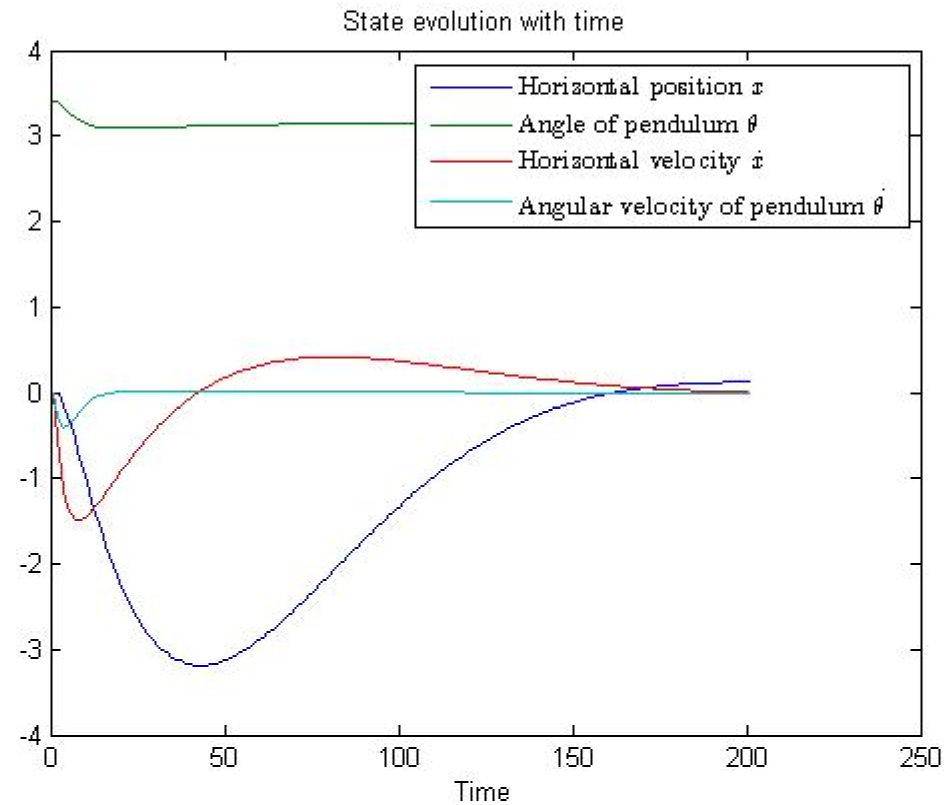
```
function [L, P] = lqr_infinite_horizon_solution(Q, R)

% find the infinite horizon L and P through running LQR back-ups
%   until norm(L_new - L_current, 2) <= 1e-4
dt = 0.1;
mc = 10; mp = 2.; l = 1.; g= 9.81;

% TODO write A,B matrices
a1 = mp*g/mc;
a2 = (mc+mp)*g/(l*mc);
dfds = [0 0 1 0;
    0 0 0 1;
    0 a1 0 0;
    0 a2 0 0];
dfdu = [0; 0; 1/mc; 1/(l*mc)];
A = eye(4) + dt*dfds;
B = dt*dfdu;

% TODO implement Riccati recursion
k = 1;
while k==1 || norm(L_new - L_current, 2) > 1e-4
    if k == 1
        L_current = 0;
        P_current = Q;
    else
        L_current = L_new;
        P_current = P_new;
    end
    L_new = -inv(R + B'*P_current*B)*(B'*P_current*A);
    P_new = Q + L_new'*R*L_new + (A + B*L_new)'*P_current*(A + B*L_new);
    diff = norm(L_new - L_current, 2);
    k = k+1;
end
L = L_new;
P = P_new;
end
```

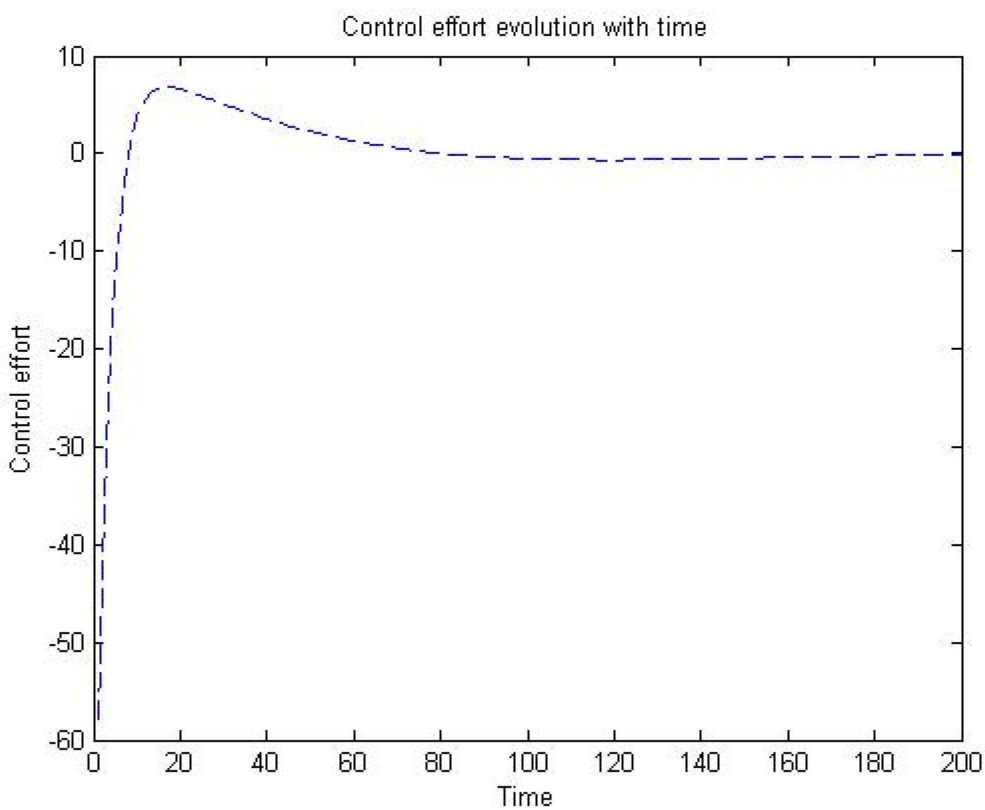
Plots

All the simulations end at this point in the animation

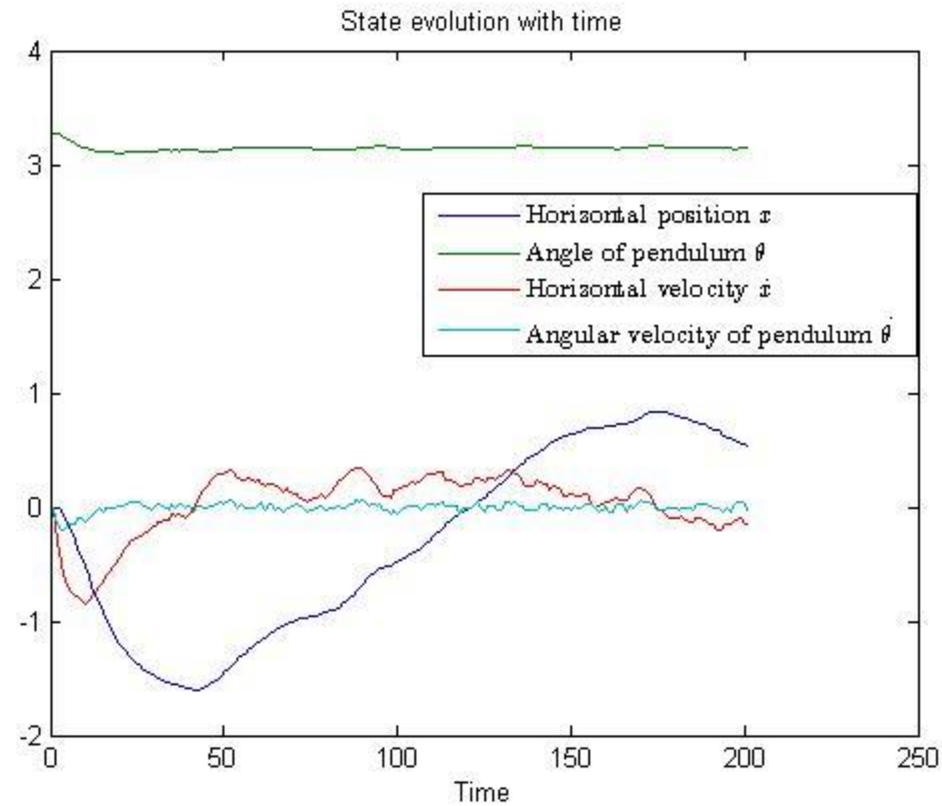


Without Noise

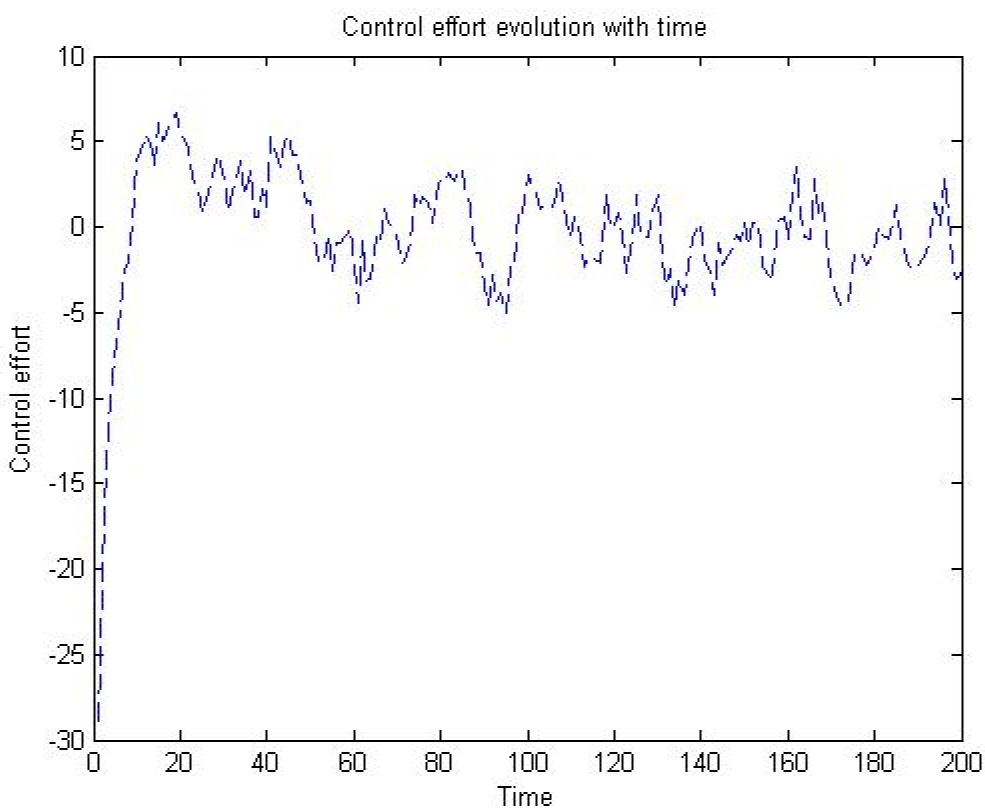




With Noise



State evolution with time
Horizontal position x
Angle of pendulum θ
Horizontal velocity $\dot{x}$
Angular velocity of pendulum $\dot{\theta}$
Time



Control effort evolution with time
Control effort
Time

PS 2 Problem 5

Wednesday, April 17, 2019 1:04 PM

$$\delta s_{k+1} = A_k \delta s_k + B_k \delta u_k$$

$$\text{Cost} = \frac{1}{2}(s_N - s^*)^T Q_N (s_N - s^*) + \sum_{k=0}^{N-1} \left(\frac{1}{2}(s_k - s^*)^T Q (s_k - s^*) + \frac{1}{2} u_k^T R u_k \right)$$

$$s_N - \bar{s} = \delta s_N$$
$$s_N = \bar{s} + \delta s_N$$
$$s_N - s^* = (\bar{s} - s^*) + \delta s_N$$

$$\frac{1}{2}(s_N - s^*)^T Q_N (s_N - s^*) = \frac{1}{2}\Big[(\bar{s} - s^*)^T Q_N (\bar{s} - s^*)^T + 2(\bar{s} - s^*)^T Q_N \delta s_N + (\delta s_N)^T Q_N (\delta s_N)\Big]$$

$$= \frac{1}{2}(\delta s_N)^T Q_N (\delta s_N) + (\bar{s} - s^*)^T Q_N \delta s_N + \underbrace{\frac{1}{2}(\bar{s} - s^*)^T Q_N (\bar{s} - s^*)}_{\text{this term doesn't depend on } \delta s_N}$$

Matching to code

$$= \frac{1}{2}(dx)^T Q_N (dx) + (qf)^T dx + \text{const}$$

$$\Rightarrow \boxed{qf = Q_N^T (\bar{s} - s^*)}$$

Similarly,

$\therefore s_k = \bar{s} + \delta s_k$

$$\frac{1}{2}(s_k - s^*)^T Q (s_k - s^*) = \frac{1}{2}(\delta s_k)^T Q (\delta s_k) + (\bar{s} - s^*)^T Q \, \delta s_k + \underbrace{\frac{1}{2}(\bar{s} - s^*)^T Q (\bar{s} - s^*)}_{\text{this term doesn't depend on } \delta s_k}$$

$$\text{Cost} = \overset{\text{quad}}{\cdots} + q' \delta s_k + \overset{\text{const}}{\cdots}$$

$\therefore$ linear term $= \boxed{q = Q^T (\bar{s} - s^*)}$

$$u_k - \bar{u} = \delta u_k$$

$$u_k = \bar{u} + \delta u_k$$

$$\frac{1}{2} u_k^T R u_k = \frac{1}{2}(\bar{u} + \delta u_k)^T R (\bar{u} + \delta u_k)$$

$$= \underbrace{\frac{1}{2} \bar{u}^T R \bar{u}}_{\text{const wrt } \delta u_k} + \underbrace{\bar{u}^T R \delta u_k}_{\text{linear}} + \underbrace{\frac{1}{2} \delta u_k^T R \delta u_k}_{\text{quadratic}}$$

cost = quad + linear + const

$$(R^T \bar{u})^T \delta u_k$$

$$\therefore \boxed{r = R^T \bar{u}}$$

Question 5

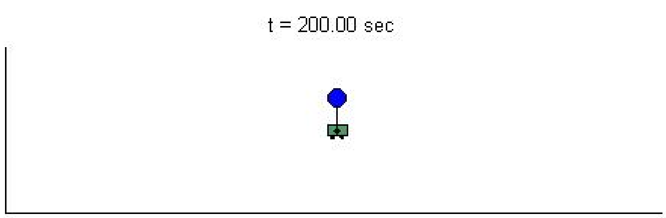
Code

```
function[x_bar,u_bar,l,L] = ilqr_solution(f,linearize_dyn, Q, R, Qf, goal_state, x0, u_bar, num_steps, dt)

% init l,L
n = size(Q,1);
m = size(R,1);

l = zeros(m,num_steps);
L = zeros(m,n,num_steps);

% init x_bar, u_bar_prev
x_bar = zeros(n,num_steps+1);
x_bar(:,1) = x0;
u_bar_prev = 100*ones(m,num_steps); %arbitrary value that will not result in termination

% termination threshold for iLQR
epsilon = 0.001;

% initial forward pass
for t=1:num_steps
    x_bar(:,t+1) = f(x_bar(:,t),u_bar(:,t),dt);
end
x_bar_prev = x_bar;

while norm(u_bar - u_bar_prev) > epsilon
    % we use a termination condition based on updates to the nominal
    % actions being small, but many termination conditions are possible.

    % ----- backward pass

    % We quadratize the terminal cost C_T around the current nominal trajectory
    % C_T(dx,du) = 1/2 dx' * QT * dx + qf' * dx + const

    % the quadratic term QT=Qf, but you will need to compute qf

    % the constant terms in the cost function are only used to compute the
    % value of the function, we can ignore them if we only care about
    % getting our control

    % TODO: compute linear terms in cost function
    qf = Qf' * (x_bar(:,end) - goal_state);

    % initialize value terms at terminal cost
    P = Qf;
    p = qf;

    for t=num_steps:-1:1
        % linearize dynamics
        [A,B,c] = linearize_dyn(x_bar(:,t),u_bar(:,t),dt);

        % TODO: again, only need to compute linear terms in cost function
        q = Q' * (x_bar(:,t) - goal_state);
        r = R' * u_bar(:,t);

        [lt,Lt,P,p] = backward_riccati_recursion(P,p,A,B,Q,q,R,r);
        l(:,t) = lt;
        L(:,:,t) = Lt;
    end

    % ----- forward pass
    u_bar_prev = u_bar; % used to check termination condition

    for t=1:num_steps
        % TODO: implement control update
        dx = x_bar(:,t) - x_bar_prev(:,t);
        du = l(:,t) + L(:,:,t) * dx;
        u_bar(:,t) = u_bar_prev(:,t) + du;
        x_bar(:,t+1) = f(x_bar(:,t),u_bar(:,t),dt);
    end

    x_bar_prev = x_bar; % used to compute dx

end

end
```

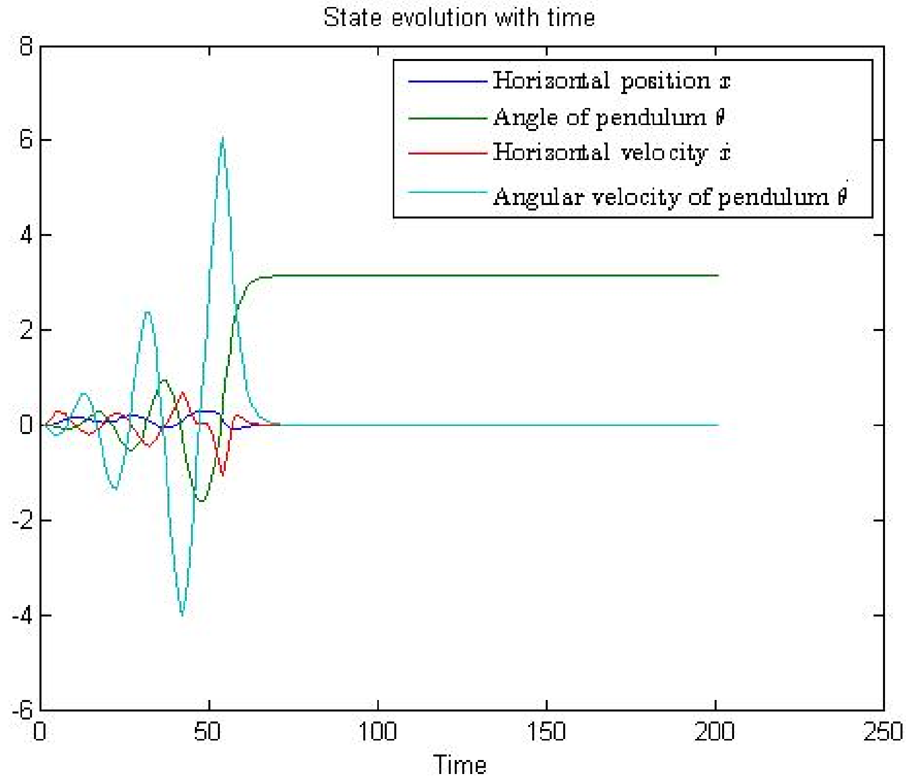
```
function [l,L,P,p] = backward_riccati_recursion(P,p,A,B,Q,q,R,r)
% TODO: write backward riccati recursion step,
% return controller terms l,L and value terms p,P
% refer to lecture 4 slides
n = size(Q,1);
m = size(R,1);
H = zeros(m,n); % no cross term for us
Q_uuk = R + B' * P * B;
Q_xxk = Q + A' * P * A;
Q_uxk = H + B' * P * A;
Q_uk = r + B' * p;
Q_xk = q + A' * p;
L = -Q_uuk\Q_uxk;
l = -Q_uuk\Q_uk;
pnew = Q_xk - L'*Q_uuk*l;
Pnew = Q_xxk - L'*Q_uuk*L;
P = Pnew;
p = pnew;
end
```

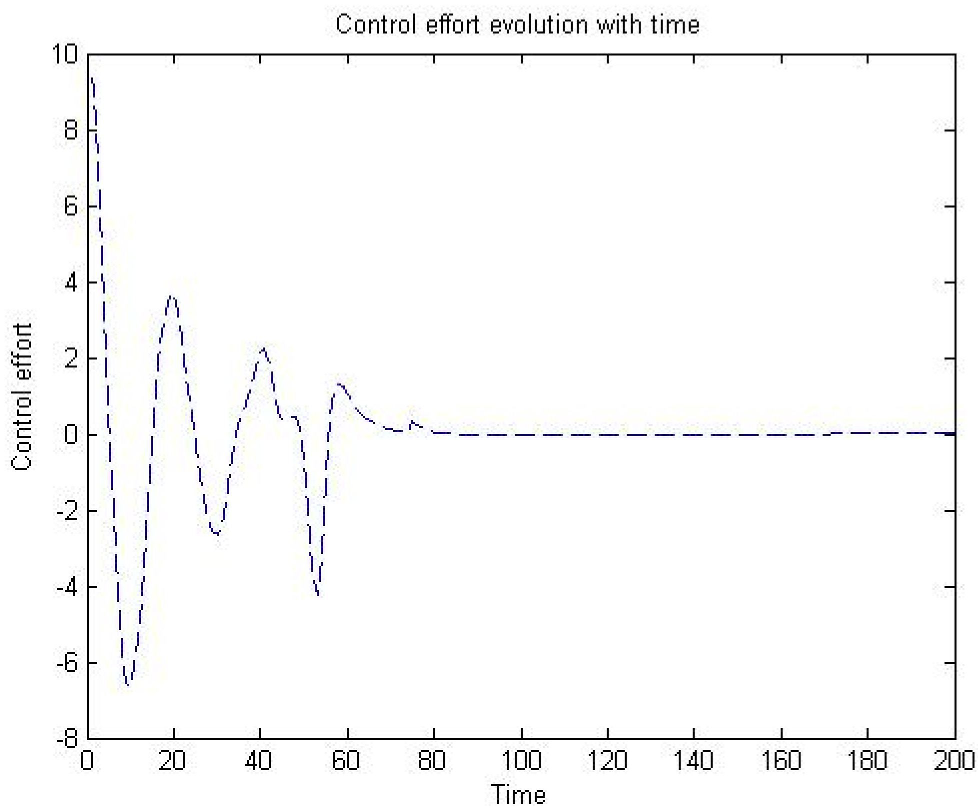
Plots

All the simulations end at this point in the animation

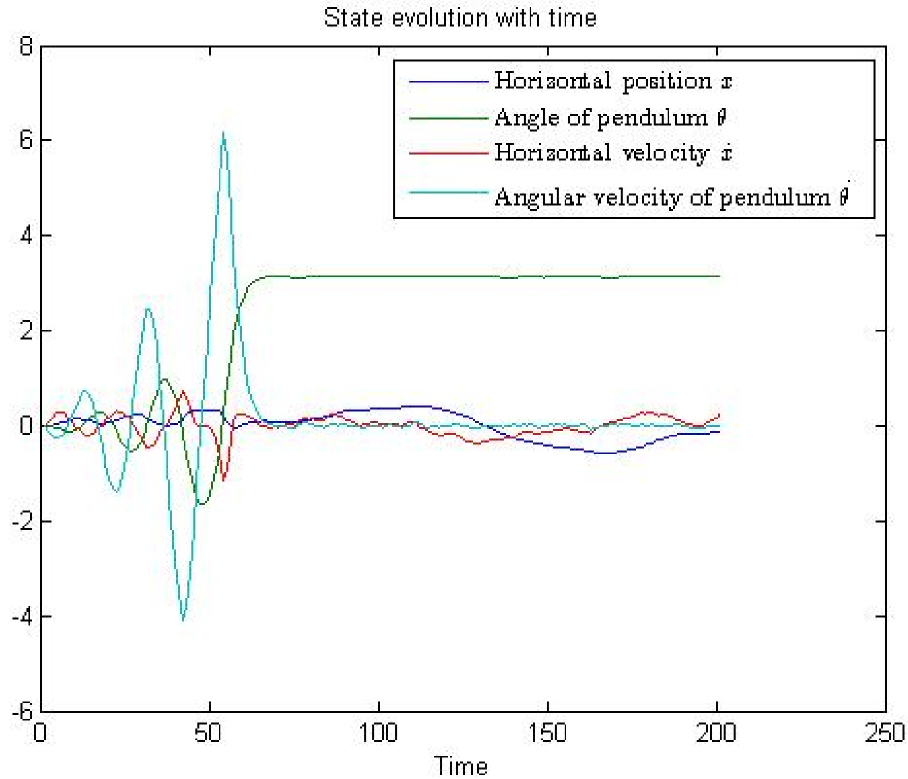


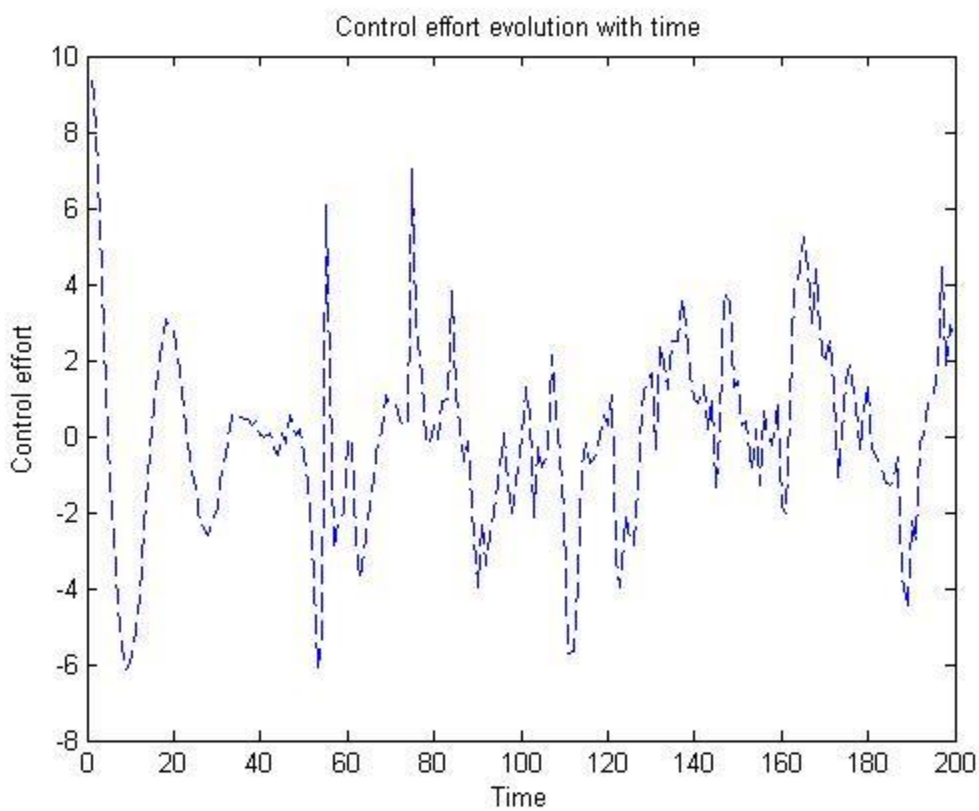
Without Noise





With Noise


State evolution with time
Horisontal position x
Angle of pendulum θ
Horisontal velocity $\dot{x}$
Angular velocity of pendulum $\dot{\theta}$
Time
8
6
4
2
0
-2
-4
-6
0
50
100
150
200
250


Control effort evolution with time
Control effort
Time
10
8
6
4
2
0
-2
-4
-6
-8
0
20
40
60
80
100
120
140
160
180
200