IoT-Based Electric Vehicle

Presented by:
Alireza Ansari

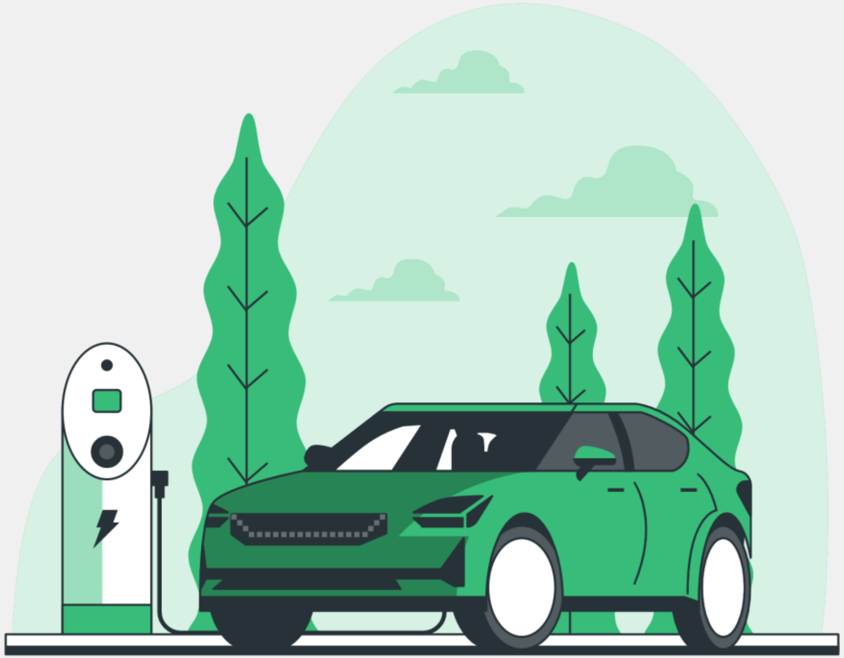


Table of Contents

01 System Model

02 LQR-based approaches

03 DLQR problem using RL

04 New Extension For Stochastic LQR

—Misson Statement

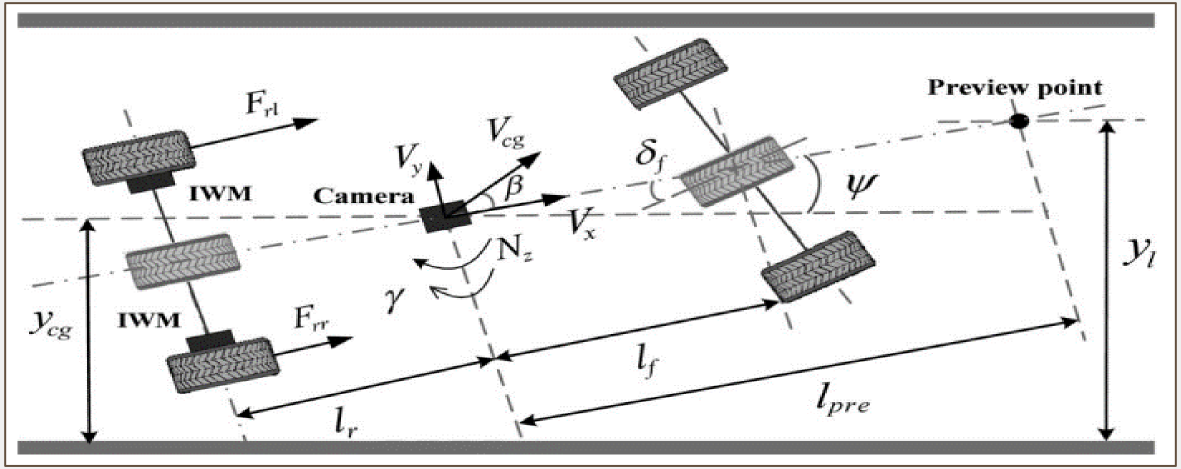
In this presentation, we are trying to study Implementations of different optimization approaches on IoT-based electric vehicle.

01

System Model

State-Space model of the system

Model Schematic



Dynamic Equations

$$\dot{\beta} = 2\,\frac{C_f}{mV_x}\left(\delta_f - \gamma\frac{l_f}{V_x} - \beta\right) - \frac{\gamma}{mV_x} + \frac{2C_r}{mV_x}\left(\gamma\frac{l_r}{V_x} - \beta\right)$$
$$\dot{\gamma} = 2\,\frac{l_f C_f}{I}\left(\delta_f - \gamma\frac{l_f}{V_x} - \beta\right) + \frac{N_z}{I} + \frac{2l_r C_r}{I}\left(\gamma\frac{l_r}{V_x} - \beta\right)$$

$$T_l = F_{rl}r = \frac{mra_x}{2} + \frac{rN_z}{d_r}, \qquad T_r = F_{rr}r = \frac{mra_x}{2} - \frac{rN_z}{d_r}$$

$$\dot{\psi} = \gamma, \qquad \dot{y}_l = V_x(\beta + \psi) + \gamma l_{pev}$$

State Space

$$A_c = \begin{bmatrix} -2\dfrac{C_r + C_f}{mV_x} & 2\dfrac{C_f l_f - C_f l_f}{mV_x^2} - 1 & 0 & 0 \\ 2\dfrac{C_r l_r - C_f l_f}{I} & 2\dfrac{-C_r l_r^2 - C_f l_f^2}{IV_x} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ V_x & l_{pre} & V_x & 0 \end{bmatrix}, \qquad B_c = \begin{bmatrix} 2\dfrac{C_f}{mV_x} & 2\dfrac{C_f l_f}{I} & 0 & 0 \\ 0 & \dfrac{1}{I} & 0 & 0 \end{bmatrix}'$$

Symbols	Values	Symbols	Values
m	380 kg	l_f	0.8 m
l_r	0.6 m	d_r	0.82 m
r	0.22 m	C_f	6000 N/rad
C_r	6000 N/rad	$\mathbf{Q}$	0.0005*$\mathbf{I}$
T	0.001 sec	$\mathbf{R}$	0.05*$\mathbf{I}$

Eig(A)= [0 0 -8.8723 -0.2446]

02

LQR-based approaches

Discrete LQR using Dynamic Programming and
Continuous LQR using HJB algorithm.

Discrete LQR vs Continuous LQR

System:

DLQR: $x(N+1) = Ax(N) + Bu(N)$

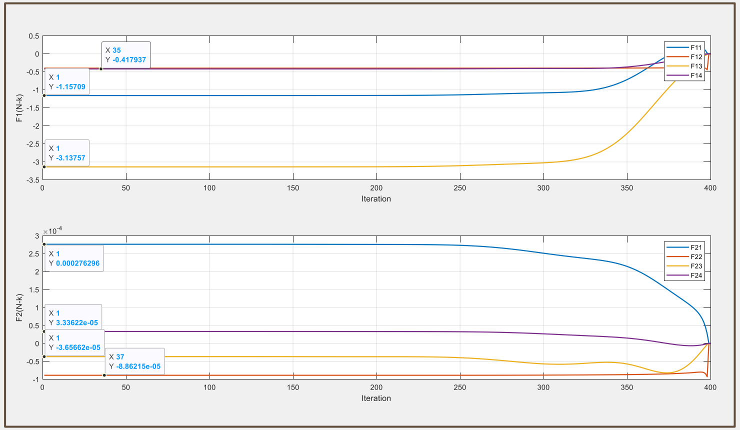
CLQR: $\dot{x}(t) = Ax(t) + Bu(t)$

Cost functions:

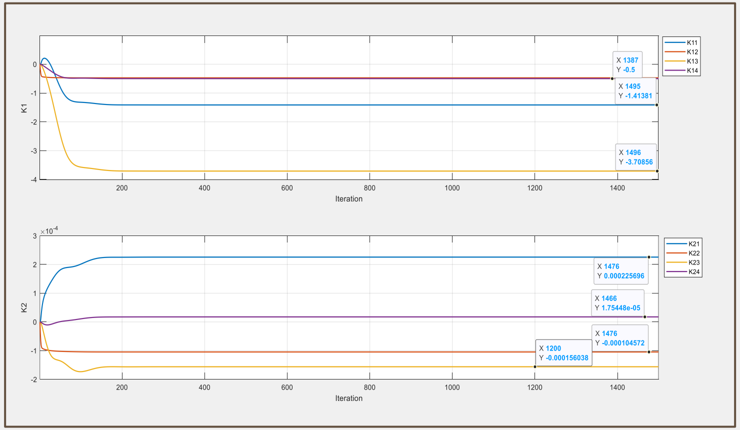
DLQR: $J = \frac{1}{2}x^T(N)Hx(N) + \frac{1}{2}\sum_{k=0}^{N-1}(x^T(k)Qx(k) + u^T(k)Ru0(k))$

CLQR: $J = \frac{1}{2}x^T(t_f)H(t_f) + \int_{t_0}^{t_f}\frac{1}{2}[x^T(t)Qx(t) + u^T(t)Ru(t)]dt$

Convergence gains

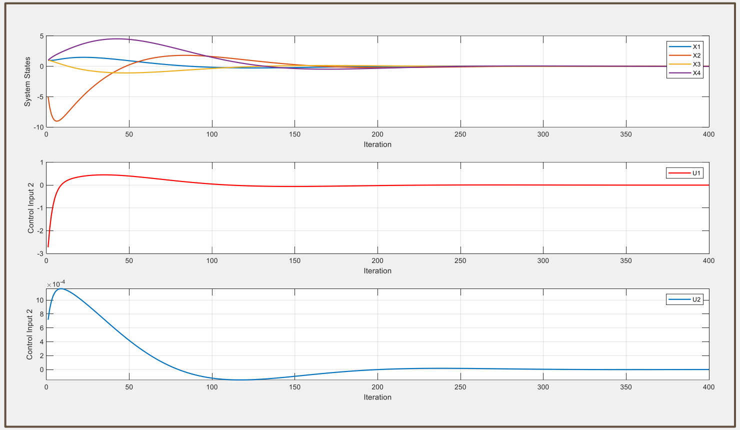


F(K) function convergence

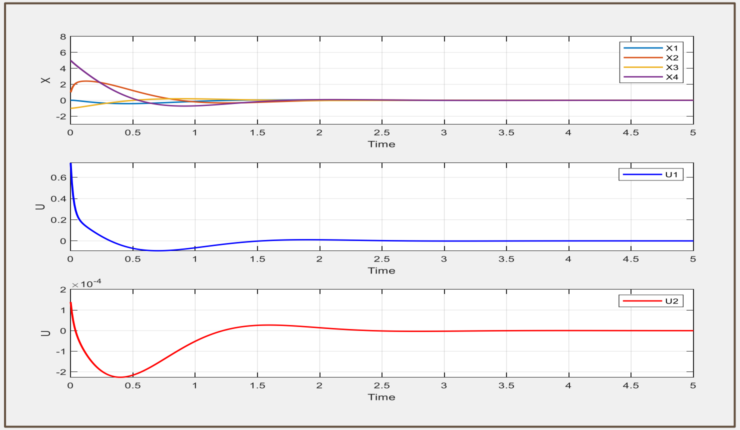


State feedback gains

System states



DLQR – system states
J = 80.4619



CLQR – system states
J = 0

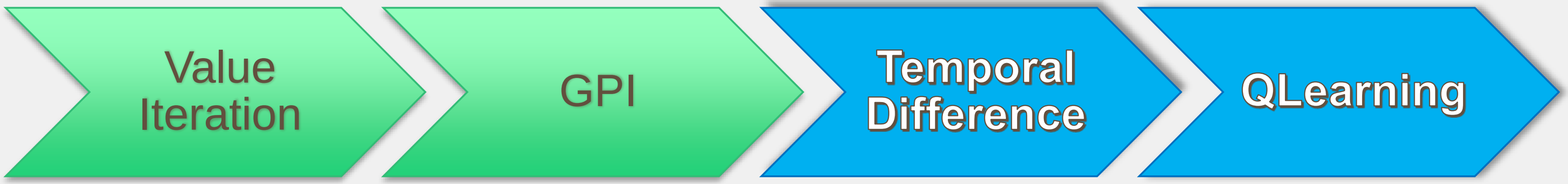
03

Solving DLQR problem using RL

Introduction

- The problem with the methods reviewed in previous chapter is being online and model-free.

- For this purpose, the methods that will be reviewed are:



Thesaurus

- Agent, Environment, Action.

- For each action at each state, a reward is received, and the goal is to **minimize sum of the rewards.**

- Function that chooses an action at each state, is called policy. Policies can be divided into two categories: Random and Deterministic.

$$\boldsymbol{\pi(x, u) = Pr(u|x)}$$

$$@\ \boldsymbol{Deterministic\ Policy: u = h(x) for\ discrete\ systems}$$

Reinforcement Learning

- Considering the state equation of a Linear Time Invariant Discrete system:

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

- According to reinforcement learning terminology, this system satisfies Markov's conditions.

- Therefore, the definite form of Bellman equations for the system can be written as follows:

$$V_h\big(x(k)\big) = r\Big(x(k), h\big(x(k)\big)\Big) + \gamma\, V_h\big(x(k+1)\big)$$

$$r\big(x(k), u(k)\big) = x(k)^T Q x(k) + u(k)^T R u(k)$$

Reinforcement Learning

- Since one of the most common form for Control signal (u) in the discrete LQR problem is the state feedback, the deterministic policy related to each state can be expressed as follows:

$$u(k) = h\big(x(k)\big) = -K.x(k)$$

- The Strategic Cost is defined as follows:

$$V_h\big(x(k)\big) = \sum_{i=k}^{k+T} \gamma^{i-k} r_i$$

$$0 < \gamma \leq 1$$

Reinforcement Learning

- In order to reach the analytical answer, the Strategic Cost is assumed as an infinite horizon:

$$V_h\big(x(k)\big) = \sum_{i=k}^{\infty} \gamma^{i-k} r_i = \sum_{i=k}^{\infty} x_i^T Q x_i + u_i^T R u_i$$

$$V_K\big(x(k)\big) = \sum_{i=k}^{\infty} x_i^T (Q + K^T R K) x_i$$

Reinforcement Learning

- As mentioned in the previous section:

$$V^*\big(x(k)\big) = x_k^T P x_k$$

$$V_h\big(x(k)\big) = x_k^T Q x_k + u_k^T R u_k + \; V_h(x(k+1))$$

$$x_k^T P x_k = x_k^T Q x_k + u_k^T R u_k + \; x_{k+1}^T P x_{k+1}$$

- Therefore, the **Policy Evaluation equation** is obtained as follows:

$$Q + K^T R K - P + (A - BK)^T P (A - BK) = 0$$

Reinforcement Learning

- Therefore:

$$V^*(x(k)) = x_k^T Q x_k + u_k^T R u_k + \ (Ax(k) + Bu(k))^T P (Ax(k) + Bu(k))$$

- Assuming that the control signal is unconstrained:

$$u^* = \min_u V^*\big(x(k)\big) \rightarrow Ru(k) + \ B^T P\big(Ax(k) + Bu(k)\big) = 0$$

$$u_k = -(R + B^T PB)^{-1} B^T PA\, x_k$$

$$K = (R + B^T PB)^{-1} B^T PA$$

Reinforcement Learning

- Therefore, the Policy Improvement equation is obtained as follows:

$$A^T PA - P + Q - A^T PB(R + B^T PB)^{-1} B^T PA = 0$$

- Getting the desired values using the idare command:

$$E = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; S = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}; H = \begin{bmatrix} 12 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}; Q = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}; R = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$

```
K_LQR =

    1.9124    0.8144    4.9522    0.6214
   -0.0018    0.0002   -0.0020   -0.0004
```

```
P_LQR =

   1.0e+03 *

    1.1298   -0.0201    1.5494    0.2667
   -0.0201    0.0068   -0.0168   -0.0040
    1.5494   -0.0168    3.0778    0.3716
    0.2667   -0.0040    0.3716    0.0960
```

Brief explanation about Fixed Point algorithm

- The goal is to determine the roots of the equation f(x) = 0. The answer can be rewritten in the form x = g(x) using different methods.

$$x \text{ is chosen is such a way that :}$$
$$|\acute{g}(x)| < 1 \quad and \quad if\ x \in [a, b] \rightarrow g(x) \in [a, b]$$

- If the above conditions are met, the following differential equation can be used to find the root of f(x) = 0:

$$x(j+1) = g(x(j))$$

Value Iteration algorithm

- Policy Evaluation:

$$P_{j+1} = \left(A - BK_j\right)^T P_j \left(A - BK_j\right) + Q + K_j^T R K_j$$

- Policy Improvement:

$$K_{j+1} = \left(R + B^T P_{j+1} B\right)^{-1} B^T P_{j+1} A$$

- Repeat the previous two steps until convergence:

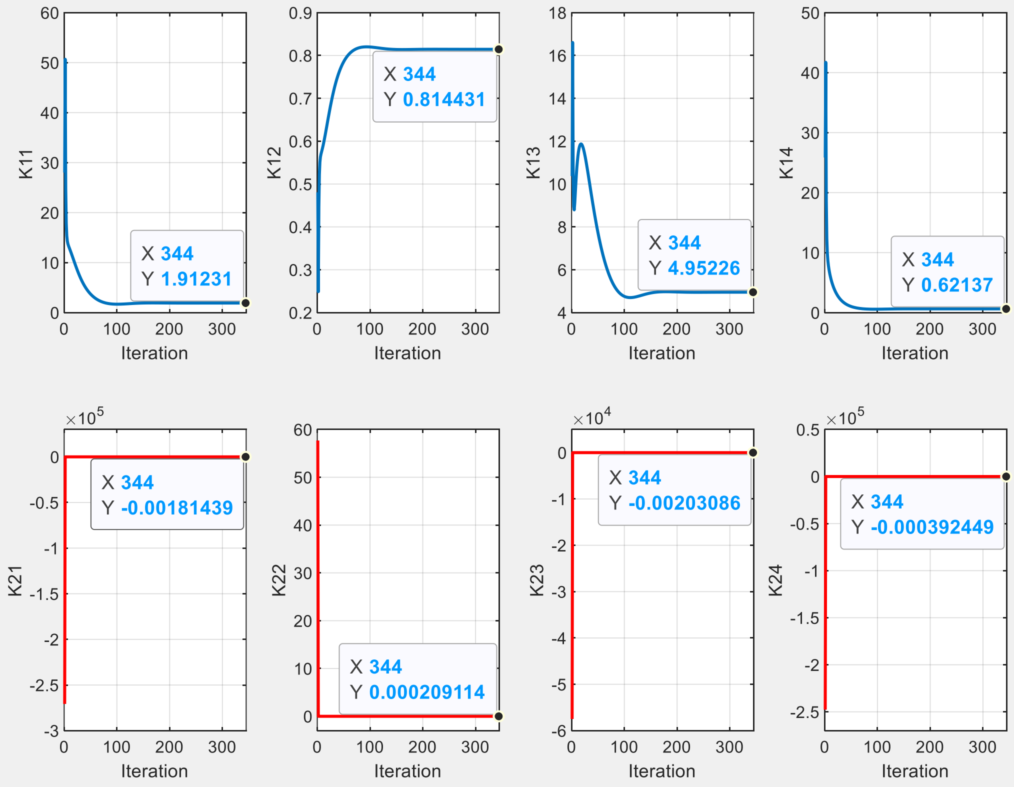
$$\left\|K_{j+1} - K_j\right\| < \varepsilon$$

Value Iteration algorithm

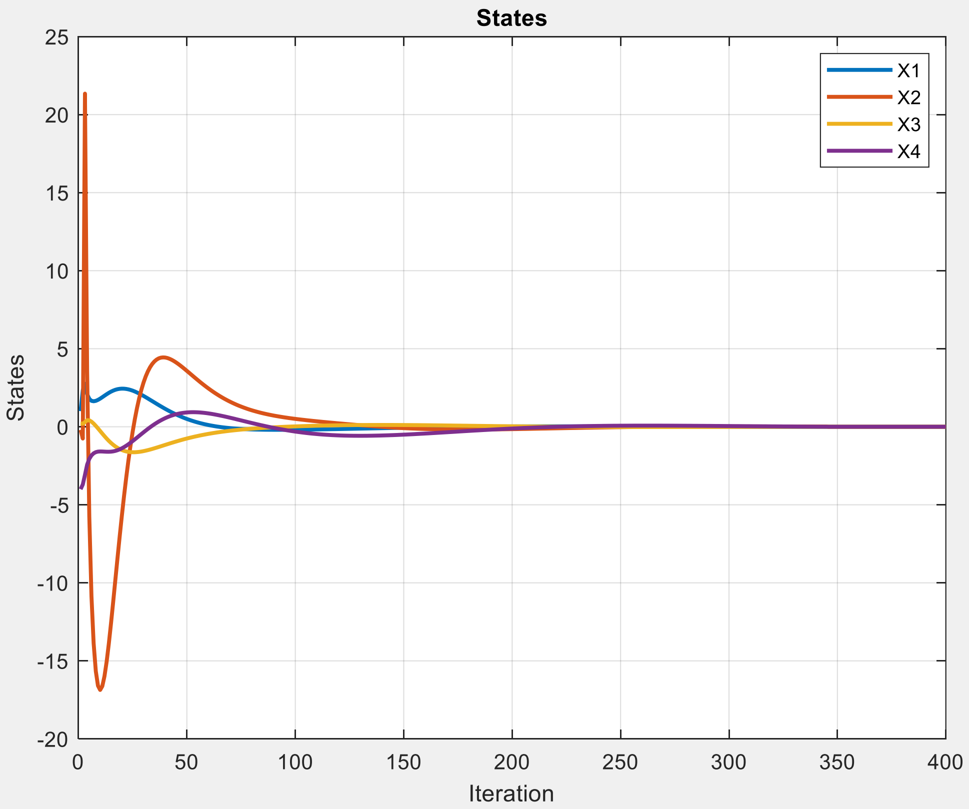
```
K_LQR =

    1.9124    0.8144    4.9522    0.6214
   -0.0018    0.0002   -0.0020   -0.0004

K_VI =
    1.9123    0.8144    4.9522    0.6214
   -0.0018    0.0002   -0.0020   -0.0004
```

```
P_LQR =

   1.0e+03 *

    1.1298   -0.0201    1.5494    0.2667
   -0.0201    0.0068   -0.0168   -0.0040
    1.5494   -0.0168    3.0778    0.3716
    0.2667   -0.0040    0.3716    0.0960

P_VI =
   1.0e+03 *

    1.1297   -0.0201    1.5494    0.2666
   -0.0201    0.0068   -0.0168   -0.0040
    1.5494   -0.0168    3.0778    0.3716
    0.2666   -0.0040    0.3716    0.0960
```

Value Iteration algorithm



Value Iteration algorithm



Value Iteration algorithm

```
Iteration(342)
Iteration(343)
Iteration(344)
Elapsed Time = 0.049294
```

```
The strategic Cost of Value Iteration method is:
  342.1940
```

GPI algorithm

- Basically, GPI Algorithm Value Iteration Algorithm :

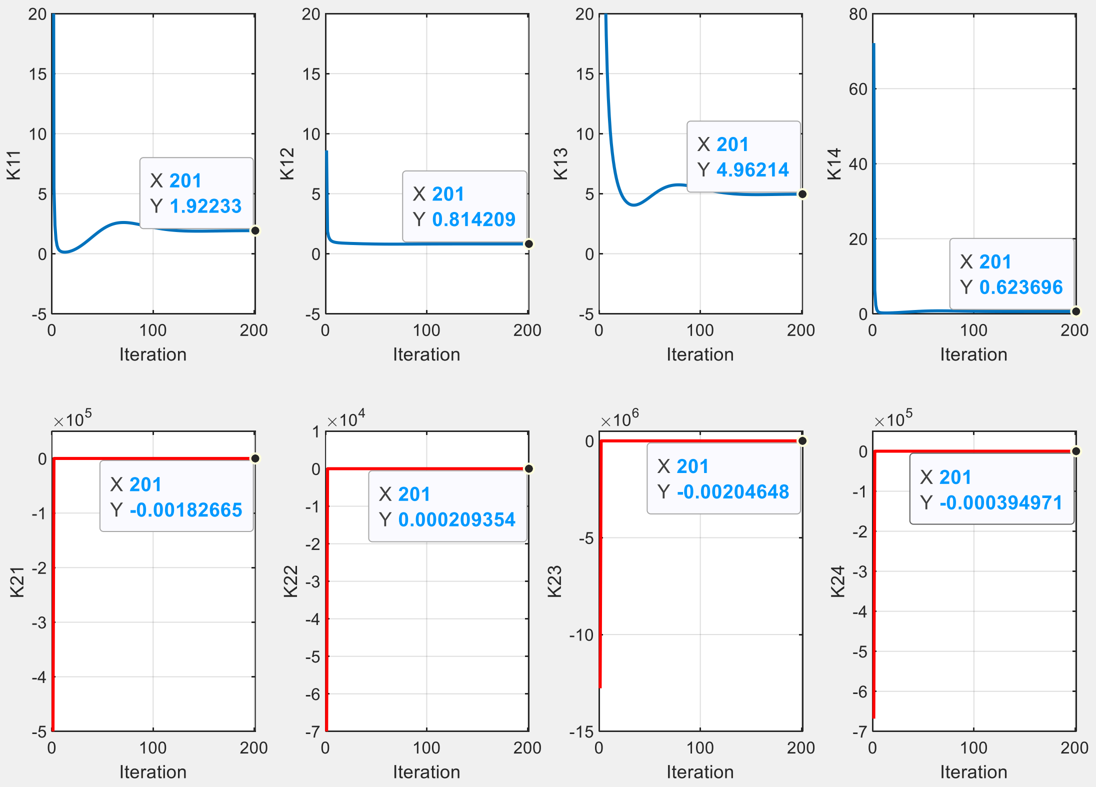
```
for j = 1:nP

    PP = P{j} ;

    for i = 1:nGPI
        PP = (A-B*K(:,:,j))'*PP*(A-B*K(:,:,j)) + Q + K(:,:,j)'*R*K(:,:,j);
    end
    P{j+1} = PP ;

    K(: , : , j+1) = inv((R + B' * P{j+1} * B)) * (B' * P{j+1} * A);
    x(:,j+1) = A*x(:,j) - B * (K(:,:,j) * x(:,j));
    disp(['Iteration(' num2str(j) ')']);

    if norm(K(: , : , j+1) - K(: , : , j)) < 1e-6
       break;
    end
end
```

GPI algorithm

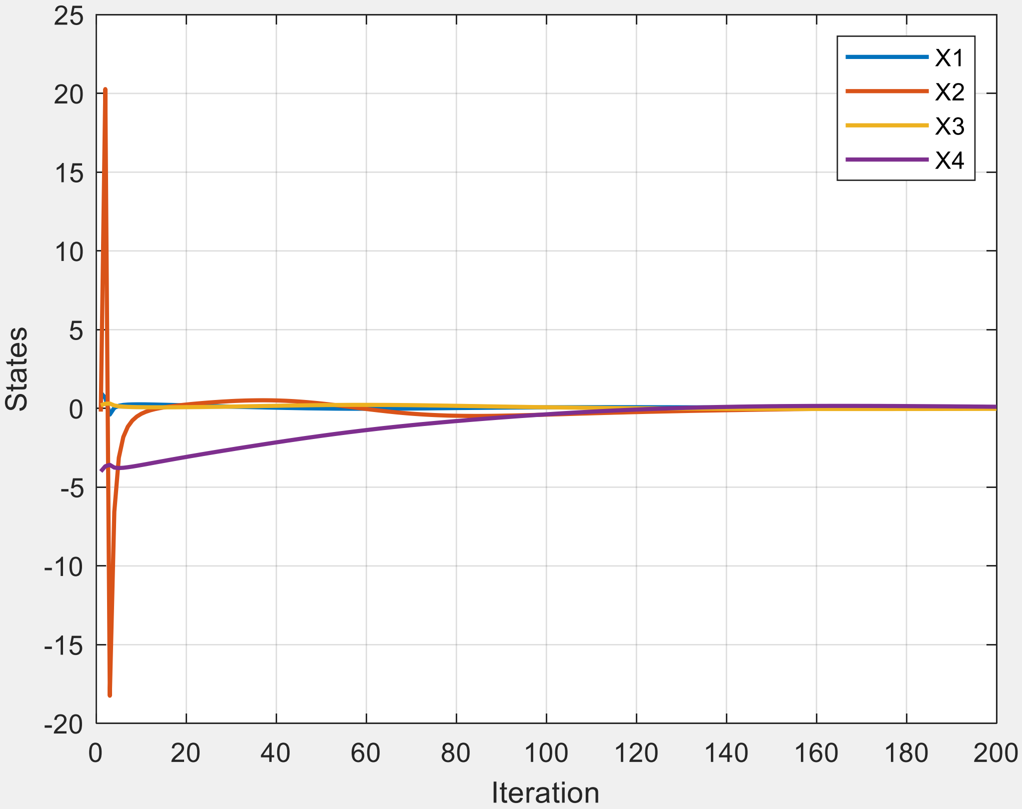
```
K_LQR =

    1.9124    0.8144    4.9522    0.6214
   -0.0018    0.0002   -0.0020   -0.0004


K_GPI =
    1.9223    0.8142    4.9621    0.6237
   -0.0018    0.0002   -0.0020   -0.0004
```

```
P_LQR =

   1.0e+03 *

    1.1298   -0.0201    1.5494    0.2667
   -0.0201    0.0068   -0.0168   -0.0040
    1.5494   -0.0168    3.0778    0.3716
    0.2667   -0.0040    0.3716    0.0960


P_GPI =
   1.0e+03 *

    1.1372   -0.0203    1.5584    0.2682
   -0.0203    0.0068   -0.0170   -0.0041
    1.5584   -0.0170    3.0873    0.3737
    0.2682   -0.0041    0.3737    0.0963
```

GPI algorithm



GPI algorithm



GPI algorithm

```
Iteration(198)
Iteration(199)
Iteration(200)
Elapsed Time = 0.030193
```

```
The strategic Cost of GPI method is:
  342.5654
```

Temporal Difference algorithm

- Bellman error equation:

$$e_k = r\left(x(k), h\left(x(k)\right)\right) + \gamma V_h\left(x(k+1)\right) - V_h\left(x(k)\right)$$

$$e_k = x_k^T Q x_k + u_k^T R u_k + x_{k+1}^T P x_{k+1} - x_k^T P x_k$$

$$V_K(x_k) = x_k^T P x_k = \left(vec(P)\right)^T (x_k \otimes x_k) \equiv \bar{P}^T \bar{x}_k$$

Kronecker product

$$x_k^T P x_k = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} a & b \\ b & c \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = a x_1^2 + 2b x_1 x_2 + c x_2^2$$

$$= \begin{bmatrix} a & 2b & c \end{bmatrix} \begin{bmatrix} x_1^2 \\ x_1 x_2 \\ x_2^2 \end{bmatrix}$$

$$\bar{\boldsymbol{P}} = \left(\boldsymbol{vec}(\boldsymbol{P})\right)^{\boldsymbol{T}}$$

$$(\boldsymbol{x_k} \otimes \boldsymbol{x_k})$$

$$if\ \ \bar{P} = \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} \rightarrow P = \begin{bmatrix} a & b/2 & c/2 \\ b/2 & d & e/2 \\ c/2 & e/2 & f \end{bmatrix}$$

Temporal Difference algorithm

$$e_k = x_k^T Q x_k + u_k^T R u_k + \bar{P}^T \bar{x}_{k+1} - \bar{P}^T \bar{x}_k$$
$$= r(x_k, u_k) + \bar{P}^T \bar{x}_{k+1} - \bar{P}^T \bar{x}_k$$

$$(\bar{x}_k{}^T - \bar{x}_{k+1}{}^T)\bar{P} = r(x_k, u_k)$$

$$\begin{cases} k = 0: (\bar{x}_0{}^T - \bar{x}_1{}^T)\bar{P} = r(x_0, u_0) \\ k = 1: (\bar{x}_1{}^T - \bar{x}_2{}^T)\bar{P} = r(x_1, u_1) \\ \vdots \\ k = M: (\bar{x}_M{}^T - \bar{x}_{M+1}{}^T)\bar{P} = r(x_M, u_M) \end{cases}$$

$$\varphi \bar{P} = \psi$$

$$\bar{P} = (\varphi^T \varphi)^{-1} \varphi^T \psi$$

Temporal Difference algorithm

- To get a unique answer:

$$u = -Kx + \color{blue}{White\ Noise}$$

- Policy Evaluation:

$$\bar{P}_{j+1}{}^{T}(\bar{x}_k - \bar{x}_{k+1}) = r\big(x_k, h_j(k)\big) = x_k^T(Q + K_j^T R K_j)x_k$$

- Policy Improvement:

$$K_{j+1} = \left(R + B^T P_{j+1} B\right)^{-1} B^T P_{j+1} A$$

Temporal Difference algorithm

K_LQR =

2.0571	0.6574	5.1232	0.7058
-0.0016	0.0002	-0.0013	-0.0003

K_TD =

2.0572	0.6574	5.1235	0.7056
-0.0016	0.0002	-0.0013	-0.0003

P_LQR =

1.0e+03 *

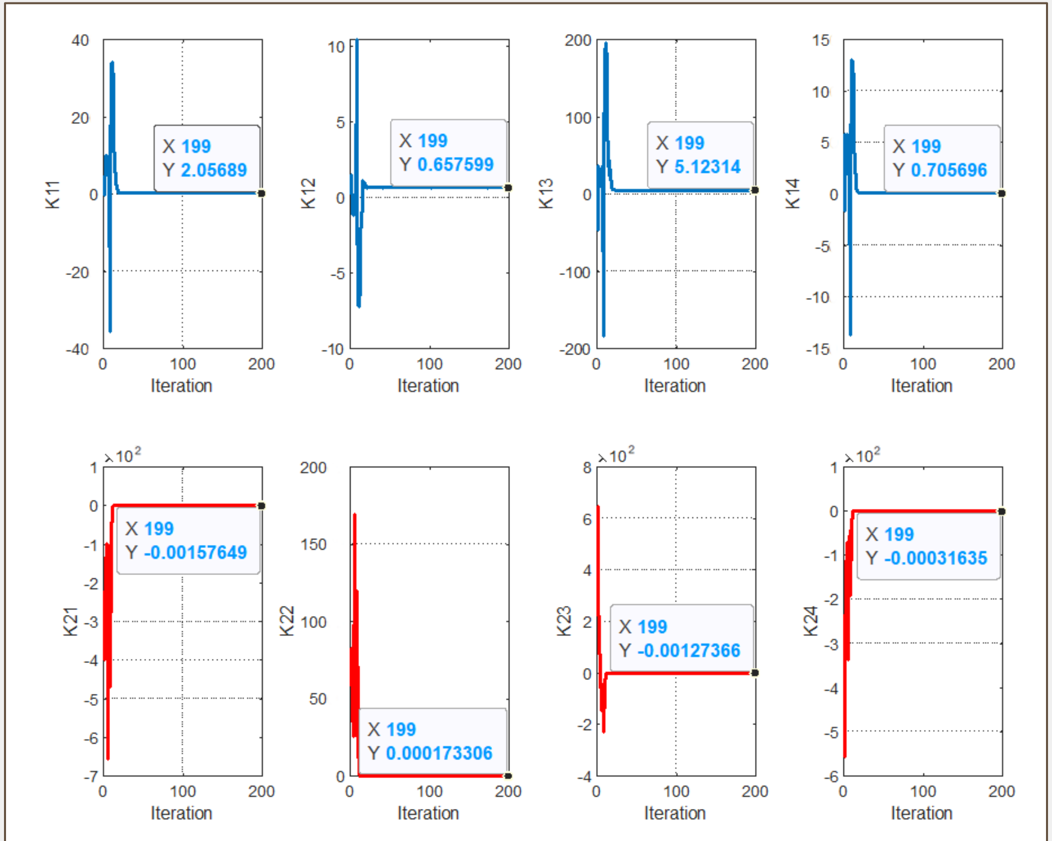
1.0238	-0.0185	1.1832	0.2368
-0.0185	0.0043	-0.0103	-0.0034
1.1832	-0.0103	2.1272	0.2800
0.2368	-0.0034	0.2800	0.0875

P_TD =

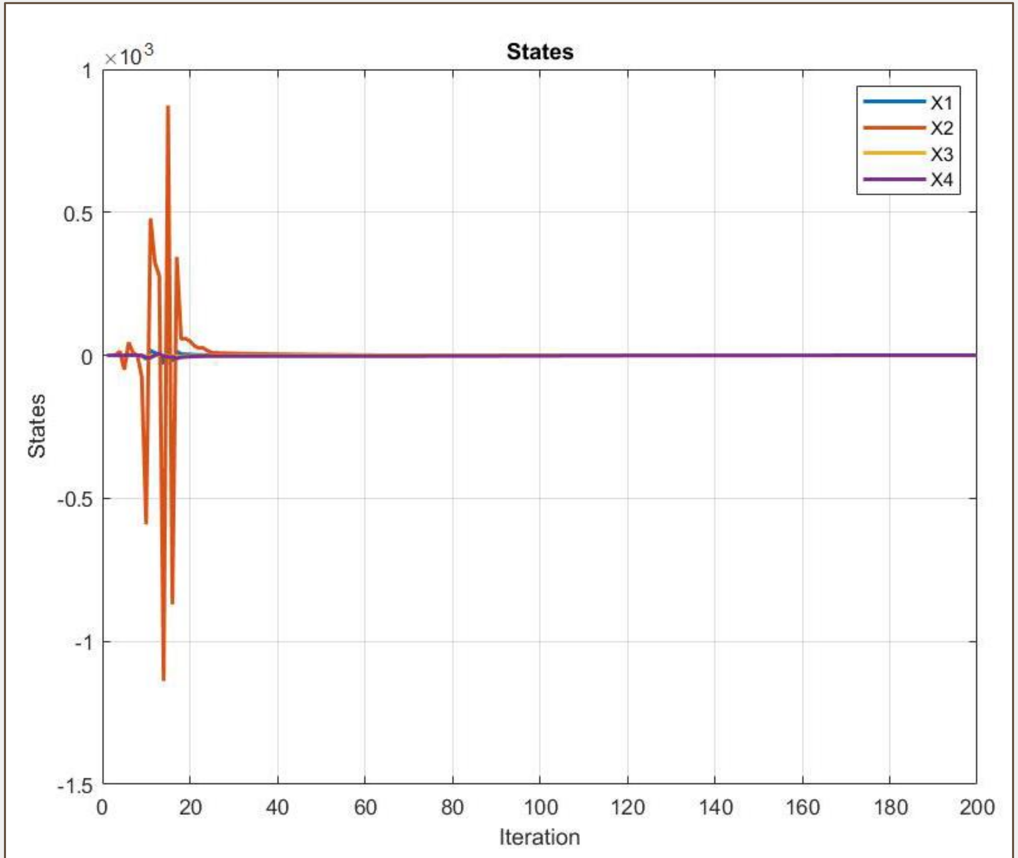
1.0e+03 *

1.0237	-0.0185	1.1832	0.2368
-0.0185	0.0043	-0.0103	-0.0034
1.1832	-0.0103	2.1272	0.2801
0.2368	-0.0034	0.2801	0.0874

Temporal Difference algorithm



Temporal Difference algorithm



Temporal Difference algorithm

```
Iteration(198)
Iteration(199)
Iteration(200)
Elapsed Time = 0.082139
```

```
The strategic Cost of Temporal Difference method is:
  388.4960
```

Optimal Adaptive controller based on Q-Learning

- First, a Q-Function or Quality Function is defined as follows:

$$Q_h(x_k, u_k) = r(x_k, u_k) + \gamma V_h(x_{k+1})$$

$$Q^*(x_k, u_k) = r(x_k, u_k) + \gamma V^*(x_{k+1})$$

$$V^*(x_k) = \min_u (Q^*(x_k, u))$$

$$\boldsymbol{h^*(x_k) = arg\,\min_u (Q^*(x_k, u))}$$

- Assuming that the control signal is unconstrained:

$$\frac{\partial}{\partial u}\left(Q^*(x_k, u)\right) = 0$$

Optimal Adaptive controller based on Q-Learning

- Bellman's optimality equation:

$$Q^*(x_k, u) = r(x_k, u_k) + \gamma Q^*(x_{k+1}, h^*(x_{k+1}))$$

$$Q_k(x_k, u) = x_k^T Q x_k + \ u_k^T R u_{k+1} + \ x_{k+1}^T P x_{k+1}$$

$$Q_k(x_k, u) = \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} Q + A^T P A & B^T P A \\ A^T P B & R + B^T P B \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} = z_k^T H z_k$$

$$Q^*(x_k, u_k) = r(x_k, u_k) + \ \gamma Q^*(x_{k+1}, h^*(x_{k+1}))$$

Evaluation $\longrightarrow$

$$\rightarrow \bar{H}^T \bar{z}_k = x_k^T Q x_k + u_k^T R u_k + \bar{H}^T \bar{z}_{k+1}$$

$$Q_k(x_k, u_k) = z_k^T H z_k = \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} H_{xx} & H_{xu} \\ H_{ux} & H_{uu} \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix}$$

Improvement $\longrightarrow$

$$H_{ux} x_k + H_{uu} u_k = 0$$

$$u_k = \ \boldsymbol{-(H_{uu})^{-1} H_{ux}} x_k$$

Optimal Adaptive controller based on Q-Learning

$$Q_k(x_k, u_k) = z_k^T H z_k = \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} \overset{nxn}{H_{xx}} & \overset{nxm}{H_{xu}} \\ \underset{mxn}{H_{ux}} & \underset{mxm}{H_{uu}} \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix}$$

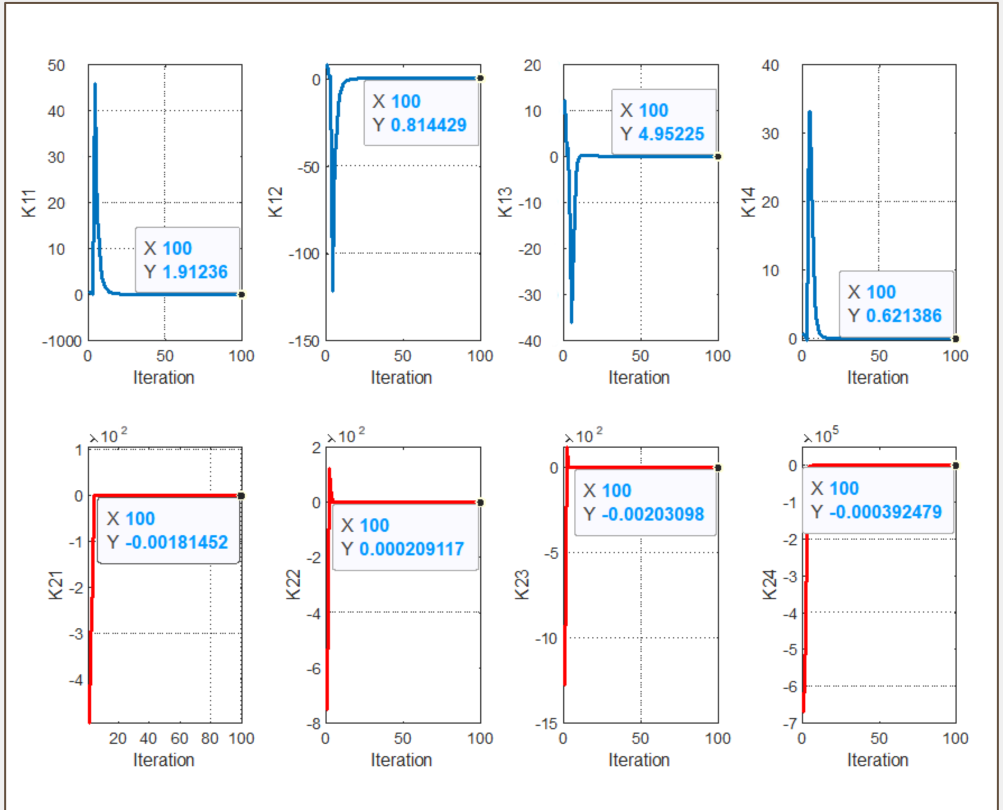
Optimal Adaptive controller based on Q-Learning

- Because the matrix p is symmetric, The number of unknown parameters is equal to:

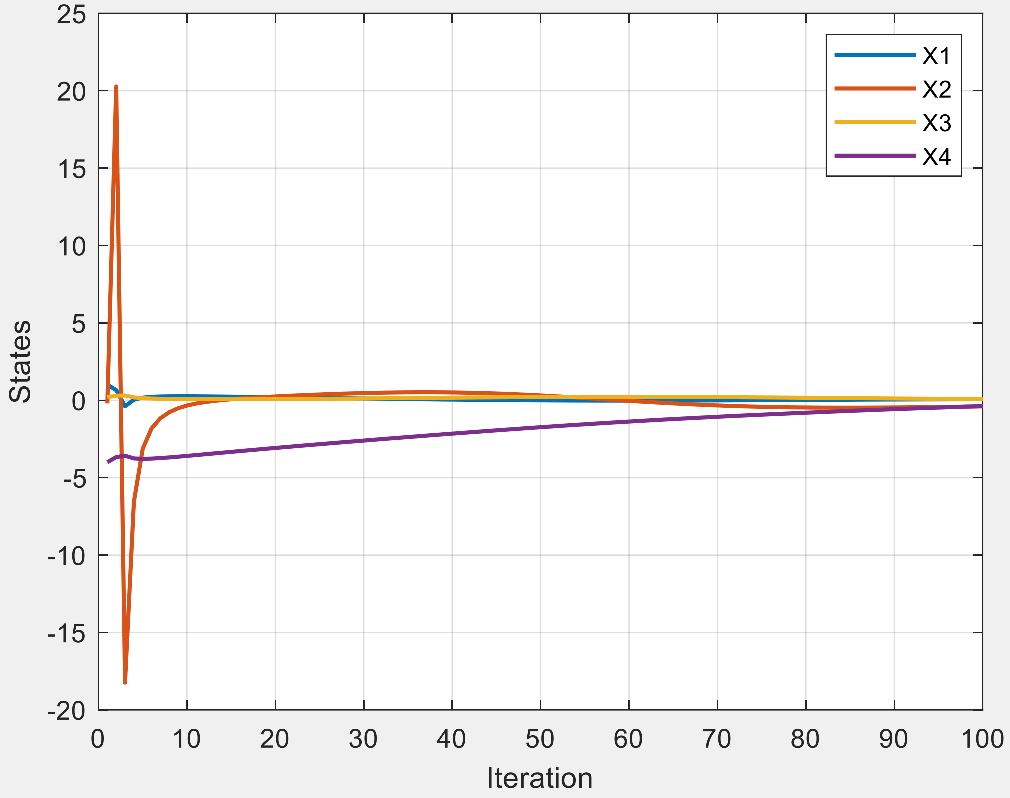
$$\frac{n(n+1)}{2}$$

```
K_LQR =

    1.9124    0.8144    4.9522    0.6214
   -0.0018    0.0002   -0.0020   -0.0004

K_Q Learning =

ans =

    1.9134    0.8147    4.9521    0.6215
   -0.0019    0.0020   -0.0020   -0.0050
```

Optimal Adaptive controller based on Q-Learning



Optimal Adaptive controller based on Q-Learning



Optimal Adaptive controller based on Q-Learning

```
The strategic Cost of Q Learning method is:
  490.4960
```

Comparison and Conclusion

Algorithm	Convergence duration(ms)	Number of Iterations	Strategic Cost
Value Iteration	50	344 / 400	342.19
GPI	30	200 / 200	342.5654
Temporal Difference	82	214 / 400	388.4960
Q Learning	7140	100 / 100	488.98

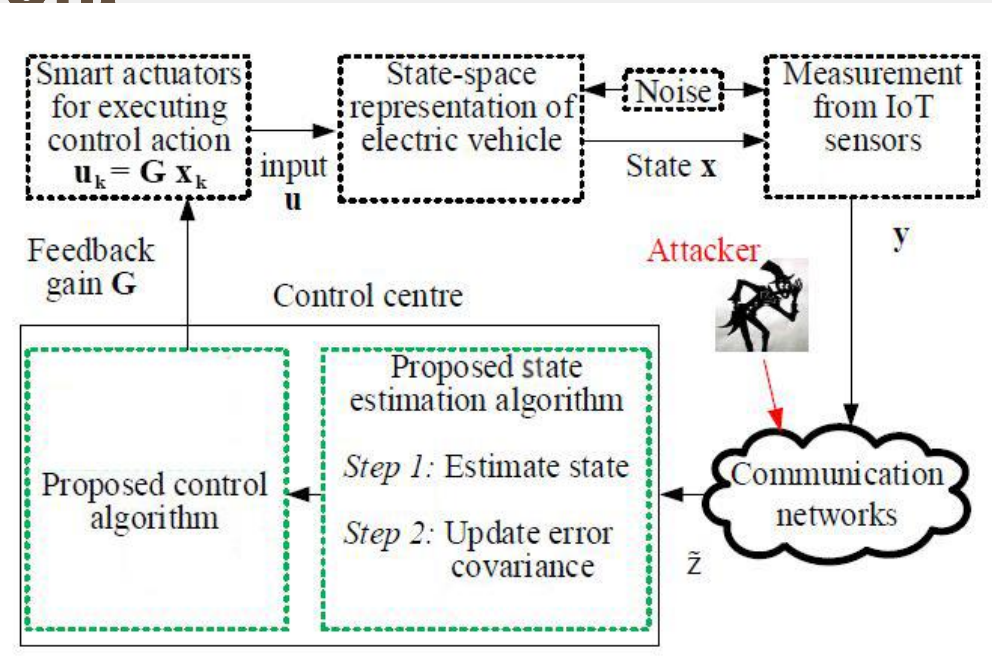
04

Stochastic Discrete LQR

Model-free optimal control of discrete-time systems with additive and multiplicative noises

Jing Lai, *student,* Junlin Xiong, *Member IEEE,* Zhan Shu, *Senior Member IEEE,*

State Estimation:



State Estimation:

- Information that is sent through the communication channel:

$$z_k = y_k - C\hat{x}_k^-$$

- Manipulated information received at the center:

$$\tilde{\mathbf{z}}_k = \boldsymbol{T}_k \mathbf{z}_k + \boldsymbol{a}_k$$

Attacker Matrix Channel noise

State Estimation:

$$\tilde{x}_k = \underbrace{\hat{x}_k^-}_{\text{Prediction}} + \underbrace{K\,\tilde{z}_k}_{\text{Correction}}$$

$$K_k = \tilde{P}C^T(C\tilde{P}C^T + R)^{-1}$$

$$\tilde{P}_k = \tilde{P}_k^- + \bar{P}C^T(\breve{P} - T_k^T\breve{P} - \breve{P}T_k)C\bar{P}$$

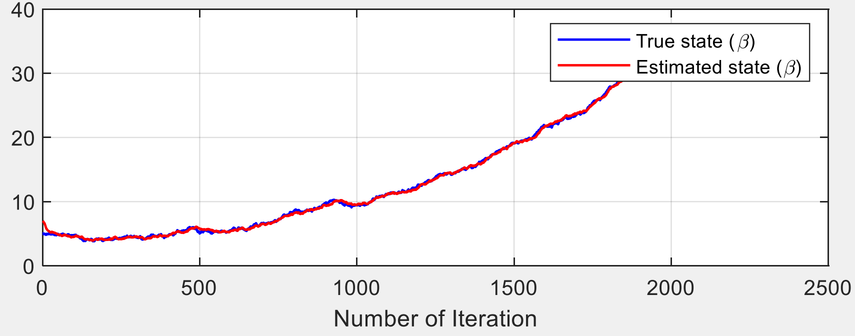
$$\tilde{P}_k^- = A_d\tilde{P}_{k-1}A_d^T + Q$$

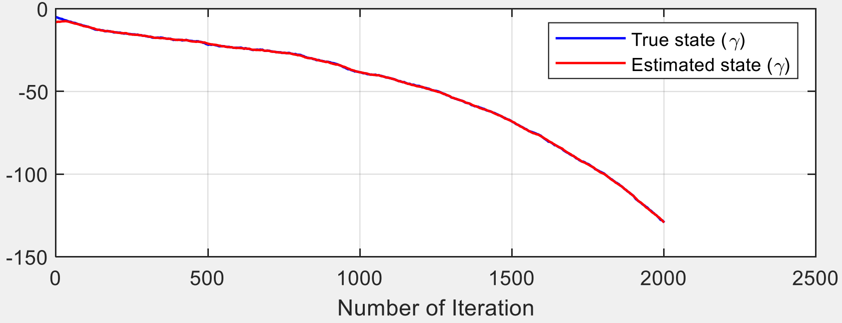
$$\breve{P} = (C\bar{P}C^T + R)^{-1}$$

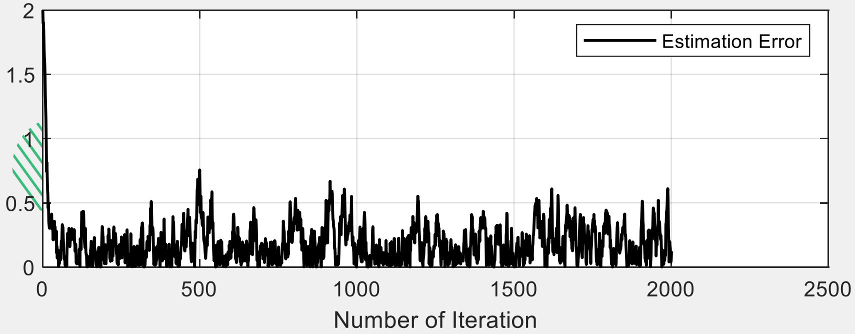
$$\bar{P} = \tilde{P}_0$$

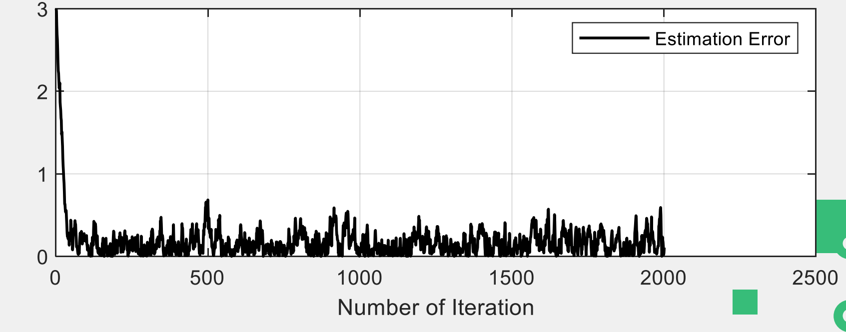
Simulation:

$$\bar{P} = \begin{bmatrix} 0.02 & 0 & 0 & 0 \\ 0 & 0.02 & 0 & 0 \\ 0 & 0 & 0.02 & 0 \\ 0 & 0 & 0 & 0.02 \end{bmatrix} \quad u_k = \begin{bmatrix} \exp(-10k) \\ \sin\left(\frac{k}{2}\right) \end{bmatrix}$$

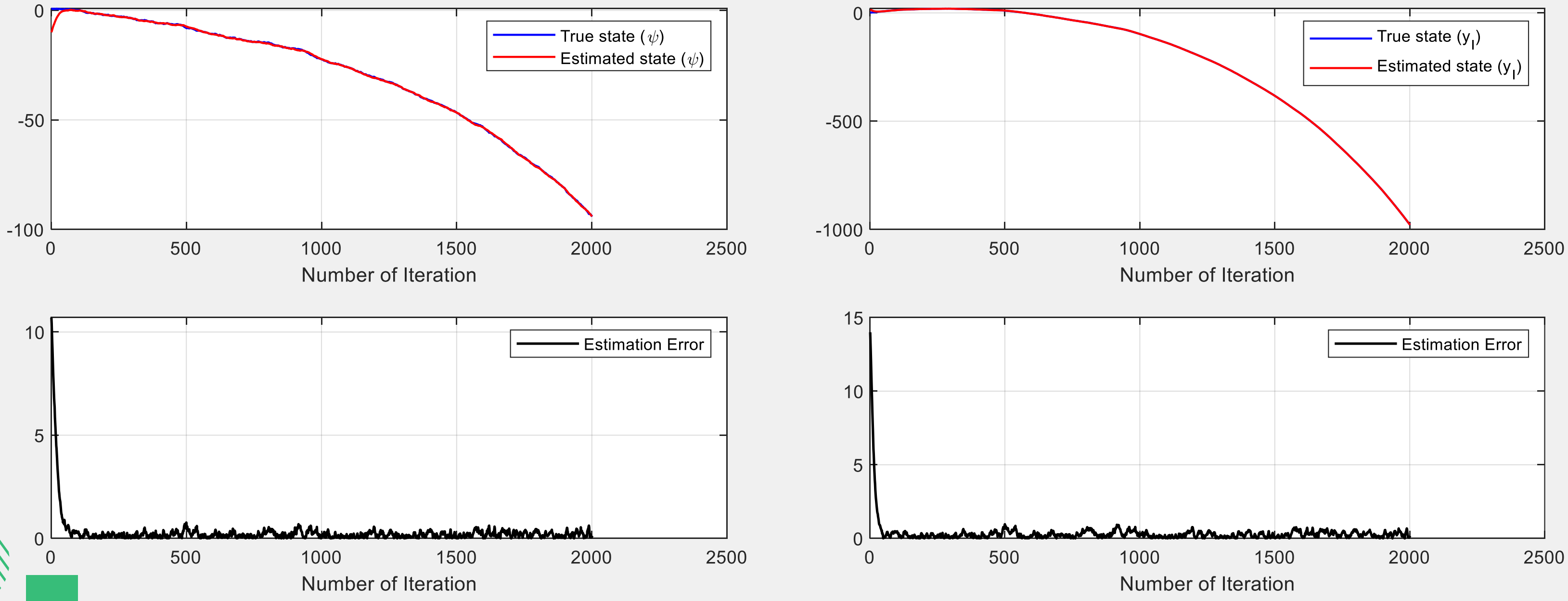








Simulation:


True state (ψ)
Estimated state (ψ)
Number of Iteration
True state (y_I)
Estimated state (y_I)
Number of Iteration
Estimation Error
Number of Iteration
Estimation Error
Number of Iteration

Control Signal:

- the optimal control algorithm is designed based on the semidefinite programming approach. According to the separation principle, the feedback control law is defined:

$$\boldsymbol{u}_k = \boldsymbol{G}\boldsymbol{x}_k$$

- Inspired by Bounded Real Lemma (without noise version):

$$\begin{aligned} &\text{minimise} \quad \boxed{\xi} \\ &\text{subject to} \quad \mathbf{A}'_{cl}\mathbf{P}\mathbf{A}_{cl} - \mathbf{P} + \xi < \mathbf{0}, \mathbf{P} > \mathbf{0}. \end{aligned}$$

Control Signal:

$$\boldsymbol{X} = \boldsymbol{P}^{-1}$$

$$(\mathbf{A}_d + \mathbf{B}_d\mathbf{G})'\mathbf{X}^{-1}(\mathbf{A}_d + \mathbf{B}_d\mathbf{G}) - \mathbf{X}^{-1} + \xi < \mathbf{0}.$$

$$\mathbf{X}(\mathbf{A}_d + \mathbf{B}_d\mathbf{G})'\mathbf{X}^{-1}(\mathbf{A}_d + \mathbf{B}_d\mathbf{G})\mathbf{X} - \mathbf{X} + \xi\mathbf{X}\mathbf{X} < \mathbf{0}.$$

- Applying the Schur's Complement:

$$\begin{bmatrix} -\mathbf{X} & \mathbf{X}(\mathbf{A}_d' + \mathbf{B}_d'\mathbf{G}') & \mathbf{X} \\ \mathbf{X}(\mathbf{A}_d' + \mathbf{B}_d'\mathbf{G}')' & -\mathbf{X} & \mathbf{0} \\ \mathbf{X} & \mathbf{0} & -\xi\mathbf{I} \end{bmatrix} < \mathbf{0}$$

Control Signal:

$$S = GX$$

YALMIP

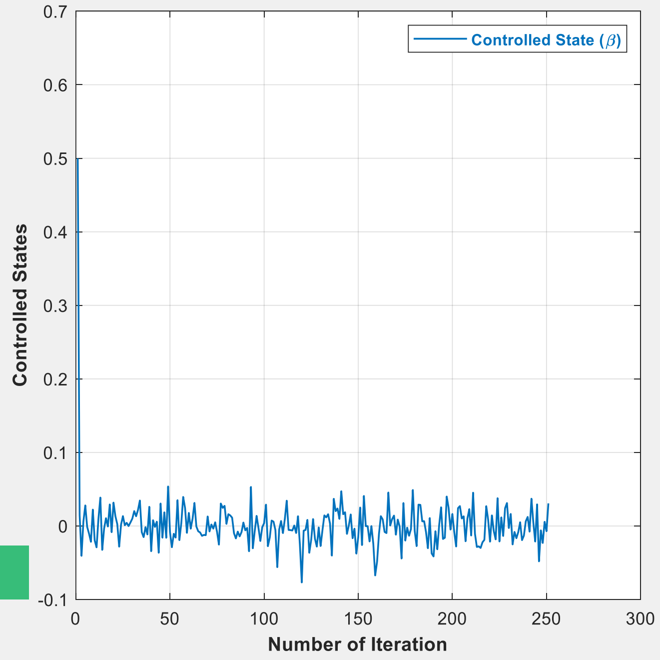
$$\begin{bmatrix} -\mathbf{X} & \mathbf{X}\mathbf{A}_d' + \mathbf{S}'\mathbf{B}_d' & \mathbf{X} \\ (\mathbf{X}\mathbf{A}_d' + \mathbf{S}'\mathbf{B}_d')' & -\mathbf{X} & \mathbf{0} \\ \mathbf{X} & \mathbf{0} & -\xi\mathbf{I} \end{bmatrix} < \mathbf{0}$$

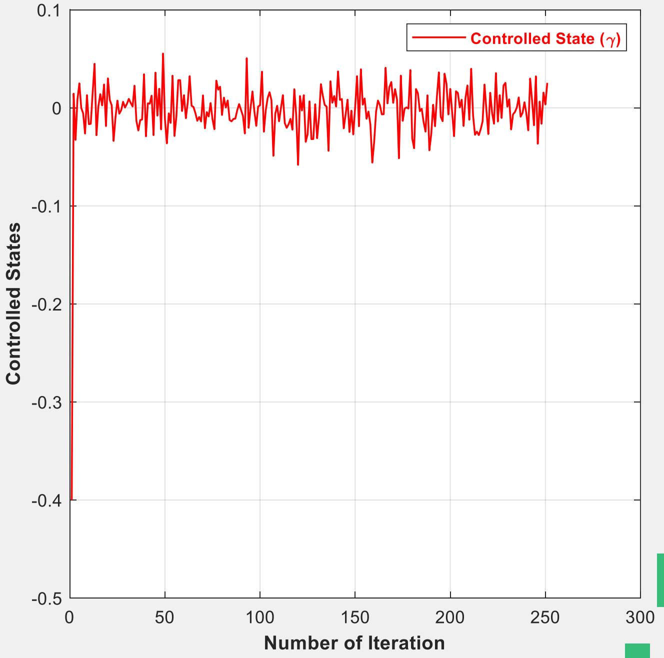
$$G = SX^{-1}$$

Control Signal:

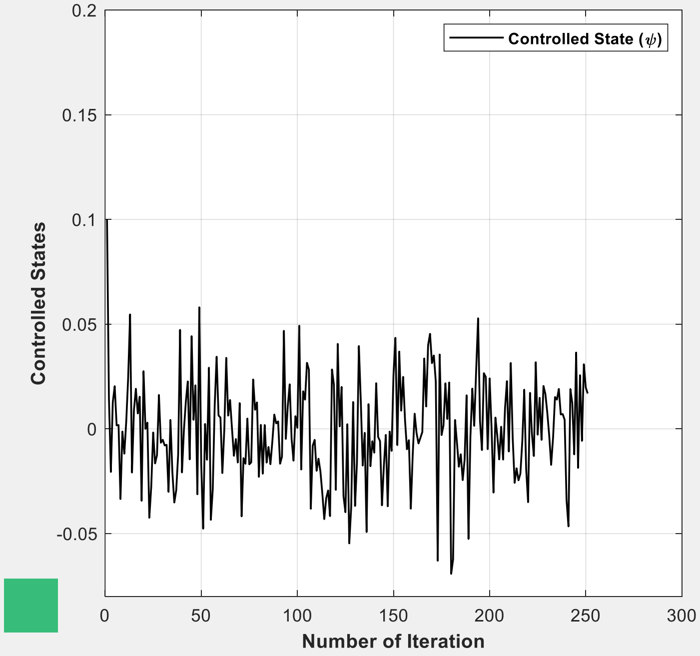
```
The optimal State feedback gain is:
   1.0e+04 *

   -0.0001    0.0000   -0.0000   -0.0000
    1.2405   -0.0229    0.0008    0.0148

The Closed-Loop system eigenvalues:
  -0.6326 + 0.0000i
   0.0000 + 0.0000i
   0.9998 + 0.0001i
   0.9998 - 0.0001i

The Optimal Value of Zeta is:
    0.0790

The Ellapsed Time to Calculate the Optimal StateFeedback gain is:2.5248
```

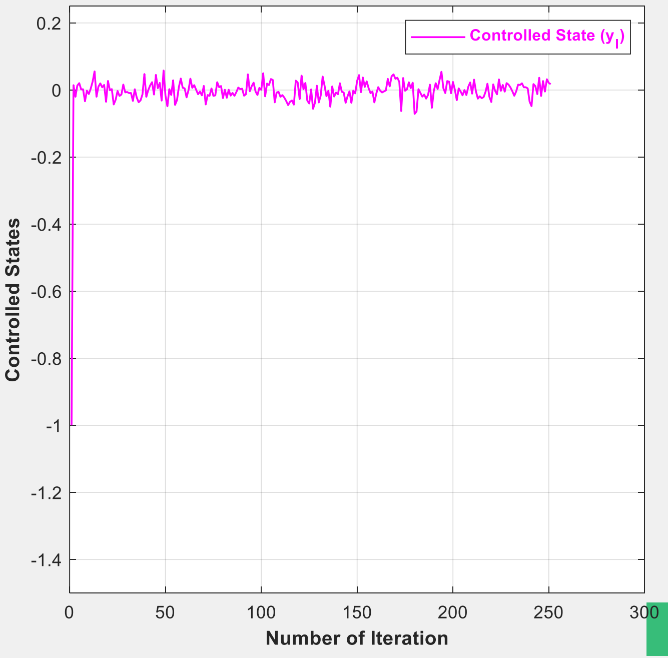
Simulation Result:





Simulation Result:





SDLQR:

$$V_h(x(k)) = r\left(x(k), h(x(k))\right) + \boxed{\gamma} V_h(x(k+1))$$

$$r(x(k), u(k)) = x(k)^T Q x(k) + u(k)^T R u(k)$$

$$u(k) = h(x(k)) = -K.x(k)$$

$$V_h(x(k)) = E(\sum_{i=k}^{k+T} \gamma^{i-k} r_i)$$

SDLQR:

- Discrete System with Additive and Multiplicative Noise:

$$x_{k+1} = Ax_k + Bu_k + (Cx_k + Du_k)d_k + w_k$$

$$E(x_0\, d_i) = E(x_0\, w_i) = E(w_i\, d_j) = 0 \; for \; all \; i, j$$

SDLQR:

- The ASS System (Unforced):

$$\rho(A \otimes A + C \otimes C) < 1$$

- Admissible Control Policy (u = Lx):

$$P = (A+BL)^\top P(A+BL)+(C+DL)^\top P(C+DL)+F$$

SDLQR:

- The ASS System (Unforced):

$$\rho(A \otimes A + C \otimes C) < 1$$

- Admissible Control Policy (u = Lx):

$$P = (A+BL)^\top P(A+BL)+(C+DL)^\top P(C+DL)+F.$$

SDLQR:

- Cost Function and Optimal Control Policy:

$$V_h(x(k)) = E(\sum_{i=k}^{k+T} \gamma^{i-k} c_i(x_i, u_i))$$

- Defining the SDLQR Problem as:

$$V^*(x_0) = \min_{u \in U_{ad}} V(x_0)$$

SDLQR:

- Cost Function and Optimal Control Policy:

$$V_h(x(k)) = E(\sum_{i=k}^{k+T} \gamma^{i-k} c_i(x_i, u_i))$$

- Defining the SDLQR Problem as:

$$V^*(x_0) = \min_{u \in U_{ad}} V(x_0)$$

SDLQR:

- Well-Posed SDLQR:

$$-\infty < V^*(x_k) < +\infty$$

- It’s proven that For an Admissible u = Lx:

$$V(x_k) = \mathrm{E}(x_k^\top P x_k) + \boxed{\frac{\gamma}{1-\gamma}\mathrm{tr}(PW)}$$

SL

E

$$P = \gamma(A + BL)^\top P(A + BL) + \gamma(C + DL)^\top P \times (C + DL) + L^\top RL + Q.$$

SDLQR:

- Back to the Cost Function:

$$V(x_k) = \mathrm{E}\big(c(x_k, u_k)\big) + \gamma \mathrm{E}\Big(\sum_{i=k+1}^{\infty} \gamma^{i-k-1} c(x_i, u_i)\Big)$$

Bellman's Equation

Stochastic form

$$V(x_k) = \mathrm{E}\big(c(x_k, u_k)\big) + \gamma V(x_{k+1})$$

$$\mathrm{E}(x_k^\top P x_k) = \mathrm{E}(x_k^\top Q x_k + u_k^\top R u_k) + \gamma \mathrm{E}(x_{k+1}^\top P x_{k+1}) - \gamma \mathrm{tr}(PW)$$

SDLQR:

- Hamiltonian for u = Lx:

$$H(x_k, L) = \mathrm{E}\big(x_k^\top (Q + L^\top R L) x_k\big) + \gamma \mathrm{E}(x_{k+1}^\top P x_{k+1}) \\ - \mathrm{E}(x_k^\top P x_k) - \gamma \mathrm{tr}(PW).$$

$$\frac{\partial H(x_k, L)}{\partial L}$$

$$L^* = -(R + \gamma B^\top P^* B + \gamma D^\top P^* D)^{-1} (\gamma B^\top P^* A + \gamma D^\top P^* C)$$

SAR

E

$$P^* = Q + \gamma A^\top P^* A + \gamma C^\top P^* C - (\gamma A^\top P^* B + \gamma C^\top P^* D) \\ \times (R + \gamma B^\top P^* B + \gamma D^\top P^* D)^{-1} (\gamma B^\top P^* A + \gamma D^\top P^* C)$$

SDLQR:

- Fixed Point Algorithm:

$$f(x) = 0$$

$$x = g(x)$$

- Conditions:

$$|\acute{g}(x)| < 1$$

$$if\ x \in [a, b] \rightarrow g(x) \in [a, b]$$

$$\longrightarrow \quad x(j + 1) = g(x(j))$$

SDLQR:

Input: Admissible control gain $L^{(0)}$, discount factor γ, maximum number of iterations i_{max}, convergence tolerance ε

Output: The estimated optimal control gain $\hat{L}$

for $i = 0 : i_{max}$ **do**

Policy Evaluation:

$$P^{(i)} = \gamma(A + BL^{(i)})^\top P^{(i)}(A + BL^{(i)}) + \gamma(C + DL^{(i)})^\top \times P^{(i)}(C + DL^{(i)}) + (L^{(i)})^\top RL^{(i)} + Q$$

Policy Improvement:

$$L^{(i+1)} = -(R + \gamma B^\top P^{(i)} B + \gamma D^\top P^{(i)} D)^{-1} \times (\gamma B^\top P^{(i)} A + \gamma D^\top P^{(i)} C)$$

if $\|L^{(i+1)} - L^{(i)}\| < \varepsilon$ **then**

Break

endif

endfor

$\hat{L} = L^{(i+1)}$

SDLQR:

- :(

$$x_{k+1} = \begin{bmatrix} 0.8 & 1 \\ 1.1 & 2 \end{bmatrix} x_k + \begin{bmatrix} 0.2 \\ 1.4 \end{bmatrix} u_k + \left(\begin{bmatrix} 0.7 & 0 \\ -1 & -0.5 \end{bmatrix} x_k + \begin{bmatrix} -1 \\ 0.8 \end{bmatrix} u_k \right) d_k + w_k$$

$$P^* = \begin{bmatrix} 8.2254 & 8.0704 \\ 8.0704 & 10.3873 \end{bmatrix}$$

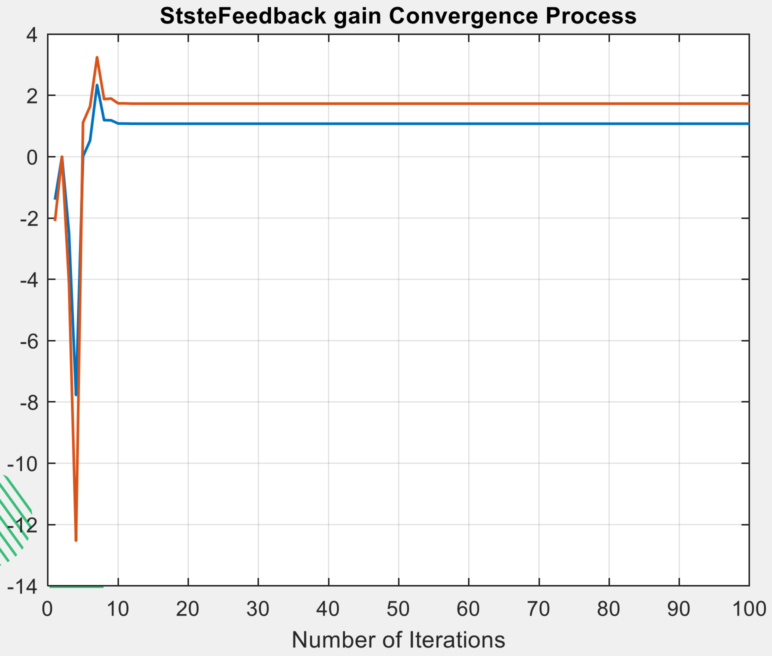
$$L^* = [-0.9319 \;\; -1.5784]$$

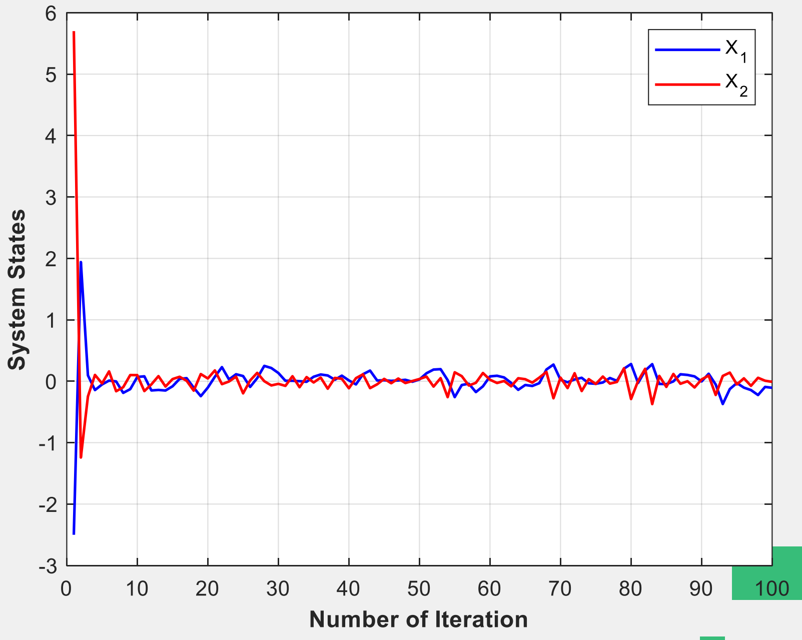
$$V^*(x_0) \;=\; 62.0422$$

```
Iteration(17)
Iteration(18)
Iteration(19)
Iteration(20)
Elapsed Time = 2.6345 Seconds
The Policy Iteration P Matrix is:
    8.6754    8.5527
    8.5527   10.9058

The Policy Iteration Method Gain is:
   -0.9424   -1.5907
```

$$\hat{V}(x_0) = 62.1118$$

SDLQR:


StsteFeedback gain Convergence Process
4
2
0
-2
-4
-6
-8
-10
-12
-14
0
10
20
30
40
50
60
70
80
90
100
Number of Iterations


6
5
4
3
2
1
0
-1
-2
-3
System States
0
10
20
30
40
50
60
70
80
90
100
Number of Iteration
X_1
X_2

SDLQR:

$$Q_h(x_k, u_k) = r(x_k, u_k) + \gamma V_h(x_{k+1})$$

- Q-Learning:

$$\begin{aligned}
&Q(x_k, u_k) \\
&= \gamma\Big(\mathrm{E}(x_{k+1}^\top P x_{k+1}) + \frac{\gamma}{1-\gamma}\mathrm{tr}(PW)\Big) + \mathrm{E}\big(c(x_k, u_k)\big) \\
&= \gamma\mathrm{E}\Big(\big(Ax_k + Bu_k + (Cx_k + Du_k)d_k + w_k\big)^\top P \\
&\quad \times \big(Ax_k + Bu_k + (Cx_k + Du_k)d_k + w_k\big) \\
&\quad + \frac{\gamma}{1-\gamma}\mathrm{tr}(PW)\Big) + \mathrm{E}(x_k^\top Q x_k + u_k^\top R u_k) \\
&= \mathrm{E}\big(x_k^\top(Q + \gamma A^\top PA + \gamma C^\top PC)x_k + 2\gamma x_k^\top(A^\top PB \\
&\quad + C^\top PD)u_k + u_k^\top(R + \gamma B^\top PB + \gamma D^\top PD)u_k\big) \\
&\quad + \frac{\gamma}{1-\gamma}\mathrm{tr}(PW) \\
&= \mathrm{E}\Big(\begin{bmatrix} x_k \\ u_k \end{bmatrix}^\top H \begin{bmatrix} x_k \\ u_k \end{bmatrix}\Big) + \frac{\gamma}{1-\gamma}\mathrm{tr}(PW), \qquad (28)
\end{aligned}$$

SDLQR:

- Q-Learning:

$$H = \begin{bmatrix} H_{xx} & H_{xu} \\ H_{ux} & H_{uu} \end{bmatrix} \in \mathcal{S}_{+}^{n+m},$$

$$\begin{aligned} H_{xx} &= Q + \gamma A^\top P A + \gamma C^\top P C \\ H_{xu} &= \gamma A^\top P B + \gamma C^\top P D = H_{ux}^\top \\ H_{uu} &= R + \gamma B^\top P B + \gamma D^\top P D. \end{aligned}$$

SDLQR:

- Optimal Control Policy:

$$Q^*(x_k, u_k) = r(x_k, u_k) + \gamma V^*(x_{k+1})$$

$$\frac{\partial Q^*(x_k, u_k)}{\partial u_k} = 0$$

$$L^* = -(H^*_{uu})^{-1} H^*_{ux}$$

$$Q(x_k, u_k) = \mathrm{E}\big(c(x_k, u_k)\big) + \gamma Q(x_{k+1}, u_{k+1})$$

SDLQR:

Input: Admissible control gain $L^{(0)}$, initial state covariance matrix X_0, additive noise covariance matrix W, discount factor γ, maximum number of iterations i_{max}, convergence tolerance ε

Output: The estimated optimal control gain $\hat{L}$

1: **for** $i = 0 : i_{max}$ **do**

2: **Policy Evaluation:**

$$\begin{aligned} &\mathrm{E}\left(\begin{bmatrix} x_k \\ u_k^{(i)} \end{bmatrix}^\top H^{(i)} \begin{bmatrix} x_k \\ u_k^{(i)} \end{bmatrix}\right) \\ &= \mathrm{E}\left(c(x_k, u_k^{(i)})\right) + \gamma \mathrm{E}\left(\begin{bmatrix} x_{k+1} \\ u_{k+1}^{(i)} \end{bmatrix}^\top H^{(i)} \begin{bmatrix} x_{k+1} \\ u_{k+1}^{(i)} \end{bmatrix}\right) \\ &\quad - \gamma \mathrm{tr}\left(H^{(i)} \begin{bmatrix} I \\ L^{(i)} \end{bmatrix} W \begin{bmatrix} I \\ L^{(i)} \end{bmatrix}^\top\right) \end{aligned} \tag{34}$$

3: **Policy Improvement:**

$$L^{(i+1)} = -(H_{uu}^{(i)})^{-1} H_{ux}^{(i)} \tag{35}$$

4: **if** $\|L^{(i+1)} - L^{(i)}\| < \varepsilon$ **then**

5: Break

6: **endif**

7: **endfor**

8: $\hat{L} = L^{(i+1)}$

SDLQR:

$$P^* = \begin{bmatrix} 8.2254 & 8.0704 \\ 8.0704 & 10.3873 \end{bmatrix}$$

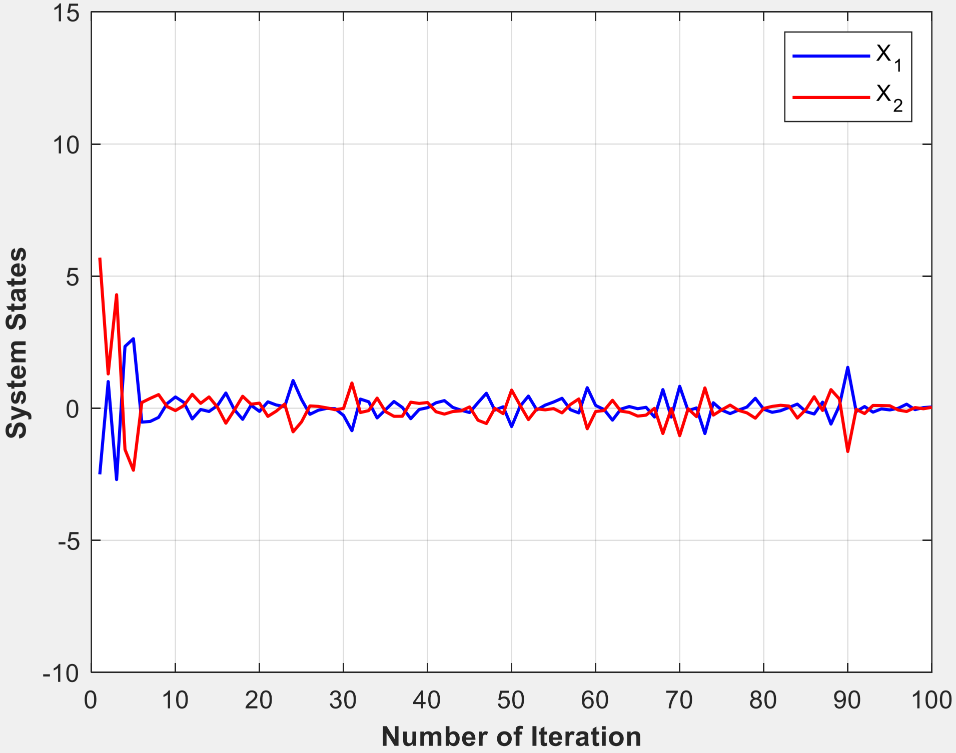
$$L^* = [-0.9319 \quad -1.5784]$$

$$V^*(x_0) = 62.0422$$

$$\hat{V}(x_0) = 62.2118$$

```
Iteration(1)
Iteration(2)
Iteration(3)
Iteration(4)
Iteration(5)
Iteration(6)
Iteration(7)
Elapsed Time = 0.033295 Seconds
Optimal Control Policy obtained by Qlearning =
   -0.9211   -1.5259
```

SDLQR:


15
10
5
0
-5
-10
System States
0
10
20
30
40
50
60
70
80
90
100
Number of Iteration
X_1
X_2

[1] M. M. Rana, "IoT-Based Electric Vehicle State Estimation and Control Algorithms Under Cyber Attacks," IEEE Internet of Things Journal, vol. 7, pp. 874--881, 2019.
[2] Y. a. N. B. M. a. F. H. a. H. Y. Wang, "Multirate estimation and control of body slip angle for electric vehicles based on onboard vision system," IEEE Transactions on Industrial Electronics, vol. 61, pp. 1133--1143, 2013.
[3] H. a. Z. G. a. W. J. Zhang, "Sideslip Angle Estimation of an Electric Ground Vehicle via Finite-Frequency $$\backslash$mathcal $\{$H$\}$ \_ $\{$$\backslash$infty$\}$ $ Approach," IEEE Transactions on Transportation Electrification, vol. 2, pp. 200-209, 2015.
[4] D. E. Kirk, Optimal control theory: an introduction, Courier Corporation, 2004.
[5] T. d. B. D. T. J. K. I. P. Lucian Bus¸oniua, "Reinforcement Learning for Control: Performance, Stability, and Deep Approximators," ELsevier, 2018.
[6] M. &. B. M. Farjadnasab, "Model-free LQR design by Q-function learning.," the International Federation of Automatic Control, 2022.
[7] L. M. É. A. M. M. A. S. R. G. &. G. L. C. Nepomuceno, "An LQR-LMI longitudinal stability augmentation system for a subscale fighter aircraft with variable center of gravity position," American Institute of Aeronautics and Astronautics., 2022.
[8] W. &. d. C. d. S. esús López Yánez, "On the effect of probing noise in optimal control LQR via Q-learning using adaptive filtering algorithms," European Journal of Control, 2022.
[9] R. &. B. B. Singh, "Reinforcement learning-based model-free controller for feedback stabilization of robotic systems," IEEE Transactions on Neural Networks and Learning Systems, 2022.
[10] M. Q. J. Z. W. X. W. Y. &. K. Y. Li, "Model-free design of stochastic LQR controller from a primal–dual optimization perspective," The Journal of IFAC, the International Federation of Automatic Control, 2022.

Thanks for your Attention!

Do you have any questions?