

# Model-Free Adaptive Dynamic Programming for Uncertain Linear Continuous and Discrete-Time Systems †

---

**Frank Lewis**

*with*

**Murad Abu-Khalaf**

**Asma Al-Tamimi, Draguna Vrabie**

*Automation & Robotics Research Institute*

*The University of Texas at Arlington*

Presented at

**2006 NSF ADP Workshop**

† This work is supported by NSF ECS-0140490 grant, and by the Army Research Office DAAD 19-02-1-0366 grant.



# Highlights: ADP & Control Theory

---

- Unlike decision theory (MDP), in control theory:
  1. State space is continuous
  2. Action space is continuous
  3. The dynamics evolve in continuous-time or in discrete-time



# Highlights: ADP & Control Theory

---

## **I. Optimal control theory and policy iterations: HJB equations**

- Continuous-time
- Discrete-time

## **II. Dynamic games and policy iterations: HJI equations**

- Continuous-time
- Discrete-time

Main characteristics of Parts I and II:

- Optimization is carried **offline**
- The **model** is required
- Initial stabilizing **policy** is required



# Highlights: ADP & Control Theory

---

## III. Optimal control theory and heuristic dynamic programming

- Discrete-time **Convergence is established**
- Continuous-time **??**

## IV. Dynamic games and heuristic dynamic programming

- Discrete-time **Convergence is established**
- Continuous-time **??**

Main characteristics of Parts III and IV:

- Optimization is carried **online**
- The **model may not** be required
- Initial stabilizing **policy is NOT** required

# The Big Picture

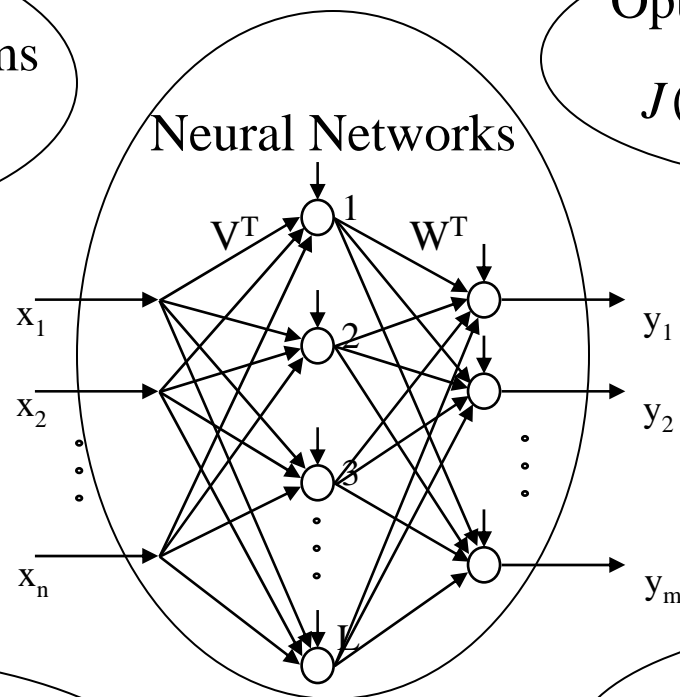
Nonlinear Control Systems

$$\dot{x} = f(x) + g(x)u$$

$$|u_i| \leq M.$$

Optimal Control Theory

$$J(x_0) = \min_u \int_0^{\infty} w(x, u) dt$$



Game Theory

$$J(x_0) = \min_u \max_d \int_0^{\infty} w(x, u, d) dt$$

April 5, 2006

ADP,  
Machine Learning



# Part I

---

## Optimal control theory and policy iterations

## Continuous-time (CT)

---

Consider the following linear system with a constrained input,

$$\dot{x}_1(t) = x_2(t)$$

$$\dot{x}_2(t) = -x_2(t) + u(t), \quad -1 \leq u(t) \leq +1 \quad \forall t \in [t_0, t_f].$$

It is desired to solve the following optimal control problem

$$V(x_0) = \min_u \int_{t_0}^{t_f} \frac{1}{2} [x_1^2(t) + u^2(t)] dt$$

Using Pontryagin's Minimum Principle and calculus of variations, the optimal controller  $u^*(t)$  solves the following split boundary conditions differential equations

# Feedback Strategies and Dynamic Programming

---

It is well known that the dynamic programming principle (DPP) finds the optimal controller in feedback form

$$\begin{aligned} V(x_0, 0) &= \min_{u(t) \in U} \int_0^{\infty} [x^T Q x + W(u)] dt, \quad Q(x) > 0, W(u) > 0, \\ &= \min_{\substack{u(t) \in U \\ 0 \leq t \leq T}} \left\{ \int_0^T [x^T Q x + W(u)] dt + \int_T^{\infty} [x^T Q x + W(u)] dt \right\} \\ &= \min_{\substack{u(t) \in U \\ 0 \leq t \leq T}} \left\{ \int_0^T [x^T Q x + W(u)] dt + V(x_0(T), T) \right\} \end{aligned}$$

# Feedback Strategies and the Hamilton-Jacobi-Bellman (HJB) Equation

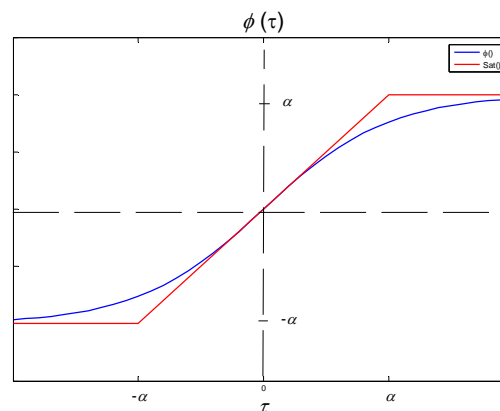
Therefore, one obtains the Hamilton-Jacobi-Bellman equation

$$0 = \min_{u(t) \in U} \left\{ x^T Q x + W(u) + V_x^T(x) [f(x) + g(x)u] \right\}.$$

For constrained inputs,  $W(u)$  maybe

$$W(u) = \int_0^u \phi^{-1}(v) R dv,$$

where  $\phi(u)$  is a smooth function approximating the saturation as shown.



# The Hamilton-Jacobi-Bellman (HJB) Equation

---

The HJB equation becomes

$$0 = \min_{u(t)} \left\{ x^T Q x + \int_0^u \phi^{-1}(v) dv + V_x^T(x) [f(x) + g(x)u] \right\}.$$

and the optimal controller is given by

$$u^*(x) = -\phi\left(\frac{1}{2}R^{-1}g^T(x)V_x(x)\right).$$

The optimal controller depends on the value function of the optimal control problem  $V(x)$ . Once  $V(x)$  is determined,  $u^*(x)$  is. Note that  $V(x)$  solves a complicated nonlinear differential equation given by

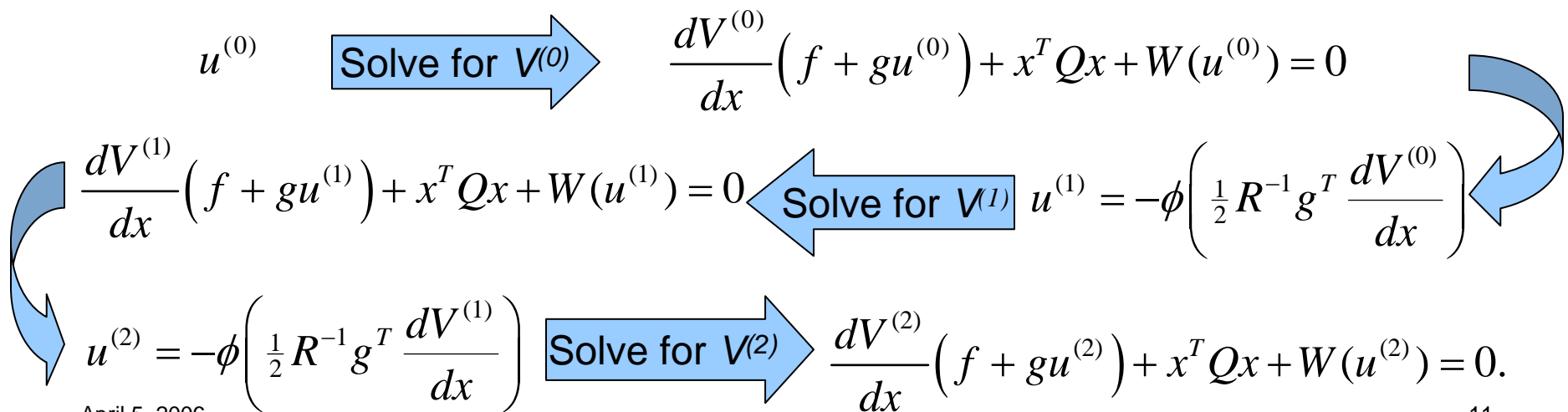
$$V_x^T \left( f - g \cdot \phi\left(\frac{1}{2}R^{-1}g^T V_x\right) \right) + x^T Q x + 2 \int_0^{-\phi\left(\frac{1}{2}R^{-1}g^T V_x\right)} \phi^{-T}(v) dv = 0, \quad V(0) = 0.$$

## Policy Iterations and Optimal Control

- A stable policy always have a value or a cost:

$$V_j(x_0) = \int_0^{\infty} [Q(x) + W(u_j)] dt \quad u_{j+1}(x) = -\phi \left( \frac{1}{2} R^{-1} g' \frac{dV_j}{dx} \right)$$

- Using policy iterations to break the HJB of constrained input systems into a sequence of Lyapunov equations linear in  $V$ .



## Policy Iterations and CT LQR

- The HJB for linear systems is:

$$x^T (A^T P + PA - PBB^T P + Q)x = 0$$

which requires solving an algebraic Riccati equation.

- Policy iterations in this case is Newton's method to solve the algebraic Riccati equation

$$(A - BB^T P_j)^T P_{j+1} + P_{j+1} (A - BB^T P_j) + P_j BB^T P_j + Q = 0$$



$$P_{j+1} = P_j - \left( Ric'_{P_j} \right)^{-1} Ric(P_j), \quad i = 0, 1, \dots$$

# Neural Networks Least Squares Solution for $V_i(x)$ in Policy Iterations - I

- First, a neural network is used to arbitrarily approximate the cost function in the Lyapunov equations

$$V_L^{(i)}(x) = \sum_{j=1}^L w_j^{(i)} \sigma_j(x) = \mathbf{w}_L^{T(i)} \boldsymbol{\sigma}_L(x).$$

- The performance function  $W(u)$  is selected as

$$W(u) = 2 \int_0^u \tanh^{-1}(v)^T R dv.$$

- When  $V_L$  is substituted in the Lyapunov equation, a residual error appears

$$\mathbf{w}_L^{T(i)} \nabla \boldsymbol{\sigma}_L^T (f + g u^{(i)}) + Q + 2u^{(i)} R \left[ A \tanh^{-1} \left( \frac{u^{(i)}}{A} \right) \right] + A^2 R \ln \left( 1 - \frac{[u^{(i)}]^2}{A^2} \right) = \varepsilon(x).$$

# Discrete-time (DT)

- Optimal control

$$x_{k+1} = f(x_k) + g(x_k)u_k \quad V(x_0) = \sum_{k=0}^{\infty} x_k^T Q x_k + u_k^T R u_k$$

requires solving the DT HJB

$$\begin{aligned} V^*(x_k) &= \min_{u_k} \left[ x_k^T Q x_k + u_k^T R u_k + V^*(x_{k+1}) \right] \\ &= \min_{u_k} \left[ x_k^T Q x_k + u_k^T R u_k + V^*(f(x_k) + g(x_k)u_k) \right] \end{aligned}$$

$$u^*(x_k) = -\frac{1}{2} R^{-1} g(x_k)^T \frac{dV^*(x_{k+1})}{dx_{k+1}}$$

## DT policy iterations

---

- For any stabilizing policy

$$V_j(x_0) = \sum_{k=0}^{\infty} x_k^T Q x_k + u_j^T(x_k) R u_j(x_k)$$

- DT Policy iterations

$$V_j(x_k) = x_k^T Q x_k + u_j^T(x_k) R u_j(x_k) + V_j(x_{k+1})$$

$$u_{j+1}(x_k) = -\frac{1}{2} R^{-1} g(x_k)^T \frac{dV_j(x_{k+1})}{dx_{k+1}}$$

- DT LQR:

$$(A + BL_j)^T P_j (A + BL_j) - P_j = -Q - L_j^T R L_j$$

$$L_j = -(I + B^T P_j B)^{-1} B^T P_j A$$

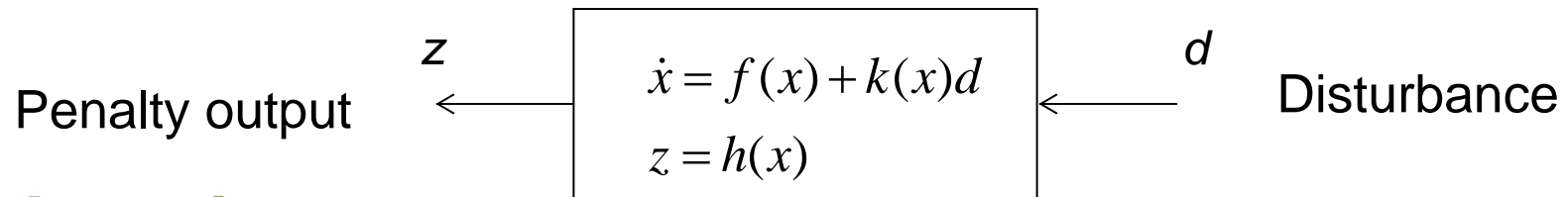


## Part II

---

# Dynamic games and policy iterations

# Dissipative Dynamical Systems.



## $L_2$ -gain

Find smallest  $\gamma^2$  so that

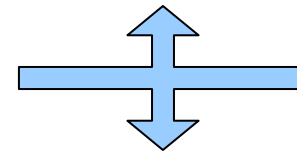
$$\frac{\int_0^{\infty} \|z(t)\|^2 dt}{\int_0^{\infty} \|d(t)\|^2 dt} = \frac{\int_0^{\infty} (h^T h) dt}{\int_0^{\infty} \|d(t)\|^2 dt} \leq \gamma^2$$

for all  $L_2$  disturbances and a prescribed gain  $\gamma^2$  when the system is at rest,  $x_0=0$ .

$$\int_0^{\infty} (h^T h - \gamma^2 \|d\|^2) dt \leq 0$$



$$\max_d \int_0^{\infty} (h^T h - \gamma^2 \|d\|^2) dt \leq 0$$



$$0 \leq V_a(x_0) = \max_d \int_0^{\infty} (h^T h - \gamma^2 \|d\|^2) dt < \infty$$

# Dissipative Systems

The Hamiltonian of the above optimization problem is

$$H(p, d) = p^T (f(x) + k(x)d) + h^T(x)h(x) - \gamma^2 d^T d$$

Stationary point  $\frac{\partial H}{\partial d} = 0 \Rightarrow d(x)^* = \frac{1}{2\gamma^2} k(x)^T p$

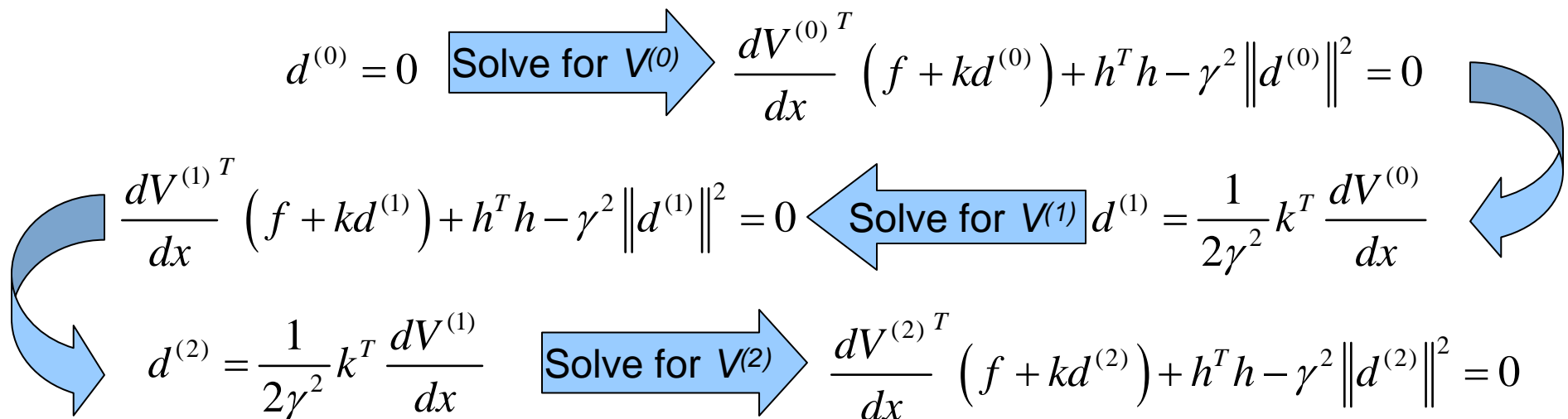
Costate  $p = \frac{dV_a}{dx}$

Worst-case disturbance

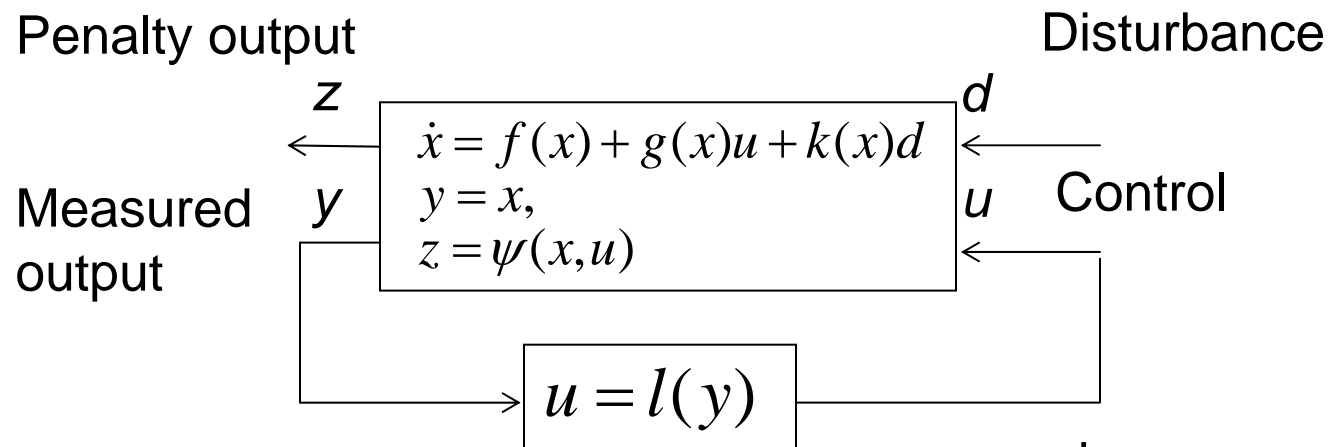
$$0 = \max_d \left\{ \frac{dV_a^T}{dx} (f(x) + k(x)d) + \|z(t)\|^2 - \gamma^2 \|d(t)\|^2 \right\} \Rightarrow d(x)^* = \frac{1}{2\gamma^2} k(x)^T \frac{dV_a}{dx}$$
$$\frac{dV_a^T}{dx} f + \frac{1}{4\gamma^2} \frac{dV_a^T}{dx} k^T k \frac{dV_a}{dx} + h^T h = 0, \quad V_a(0) = 0$$

## Solving for $V_a$ Iteratively

- The iterative solution goes as follows



# H<sub>∞</sub> Optimal Control for Constrained Input Systems



$$\|z\|^2 = h^T h + \|u\|^2$$

Find control  $u(t)$  so that

$$\frac{\int_0^\infty \|z(t)\|^2 dt}{\int_0^\infty \|d(t)\|^2 dt} \leq \gamma^2$$

for all  $L_2$  disturbances and a prescribed gain  $\gamma^2$  when the system is at rest,  $x_0=0$ .

# The Hamilton-Jacobi-Isaacs (HJI) Equation

---

$$V(u, d) = \int_0^{\infty} \left( h^T h + \|u\|^2 - \gamma^2 \|d\|^2 \right) dt \leq 0$$
$$V(x_0) = \min_u \max_d \int_0^{\infty} \left( h^T h + \|u\|^2 - \gamma^2 \|d\|^2 \right) dt < \infty$$

Hamilton-Jacobi-Isaacs (HJI) equation

$$0 = V_x^T f + h^T h - \frac{1}{4} V_x^T g R^{-1} g^T V_x + \frac{1}{4\gamma^2} V_x^T k k^T V_x$$

Stationary Point

$$u^* = -\frac{1}{2} R^{-1} g^T(x) V_x$$

Optimal control

$$d^* = \frac{1}{2\gamma^2} k^T(x) V_x$$

Worst-case disturbance

If HJI has a positive definite solution  $V$  and the associated closed-loop system is AS, then  $L_2$  gain is bounded by  $\gamma^2$

# Policy Iterations and Game Theory

**Start:**  $u_0$  a stabilizing control with region of asymptotic stability  $\Omega_0$

→ **Outer loop- update control**

Initial disturbance  $d^0 = 0$

→ **Inner loop- update disturbance**

$$\frac{\partial(V_j^i)^T}{\partial x} (f + gu_j + kd) + h^T h + u_j^T u_j - \gamma^2 (d^i)^T d^i = 0$$

$$\text{Inner loop update } d^{i+1} = \frac{1}{2\gamma^2} k^T(x) \frac{\partial V_j^i}{\partial x}$$

Iterate  $i$  until convergence to  $d^\infty, V_j^\infty$  with RAS  $\Omega_j^\infty$

Outer loop update

$$u_{j+1} = -\frac{1}{2} g^T(x) \frac{\partial V_\infty^j}{\partial x}$$

Iterate  $j$  until convergence to  $u_\infty, V_\infty^\infty$ , with RAS  $\Omega_\infty^\infty$

April 5, 2006

**End**

## CT LQ Game

- For linear systems:

$$\dot{x} = Ax + Bu + Ed$$

$$V = \min_u \max_d \int_0^{\infty} (x^T Qx + \|u\|^2 - \gamma^2 \|d\|^2) dt$$

- Inner loop policy iterations solve the bounded real lemma

$$(A_j + EE^T P_{j+1}^i)^T P_{j+1}^{i+1} + P_{j+1}^{i+1} (A_j + EE^T P_{j+1}^i) + Q_j - \frac{1}{\gamma^2} P_{j+1}^i BB^T P_{j+1}^i = 0$$

$$A_j = A - BB^T P_j^\infty, \quad Q_j = P_j^\infty BB^T P_j^\infty + Q$$

$$u_{j+1} = -BB^T P_j^\infty x$$

$$d_j^{i+1} = \frac{1}{\gamma^2} EE^T P_j^i x$$

## DT LQ Game

---

- For linear systems:

$$x_{k+1} = Ax_k + Bu_k + Ed_k$$

$$V(x_0) = \min_u \max_d \sum_{k=0}^{\infty} x_k^T Q x_k + u_k^T u_k - \gamma^2 d_k^T d_k$$

- Inner loop policy iterations solve the bounded real lemma

$$P_j^{i+1} - A_j^T P_j^{i+1} A_j = Q + L_j^T R L_j - \gamma^2 K_j^{iT} K_j^i$$

$$A_j = A + B L_j$$

$$K_j^i = -\gamma^{-2} (E^T P_j^i E - \gamma^2 I)^{-1} E^T P_j^i A_j$$

$$L_j = -(I + B^T P_i B)^{-1} B^T P_i (A + E K_j^\infty)$$



## Part III and IV

---

# Model-free online optimization: Approximate Dynamic Programming (ADP)



# From machine learning to System Theory

---

- In this part, results for both discrete-time and continuous-time are shown for the case of linear systems and Algebraic Riccati Equation.
- The results here are an extension to already known technique for Machine Learning. In particular, Markov Decision Problems (MDP).
- System theory is more involved than MDPs due to the fact that both the action space and the state space are continuous for both discrete-time and continuous-time.



# Solution of the Discrete-Time Zero-Sum Game using Approximate Dynamic Programming

---

- This research combines between the following different areas
  - H-infinity Optimal control
  - Game Theory (Zero-sum games)
  - Reinforcement Learning (Q-Learning & Heuristic Dynamic Programming)
  - Neural Networks (NN)

# Dynamic Programming: Backward-in-time Formulation

- Consider the following continuous-state and action spaces discrete-time dynamical system

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + Ew_k & x &\in R^n & u_k &\in R^{m_1} \\ y_k &= x_k, & y &\in R^p & w_k &\in R^{m_2} \end{aligned}$$

- The zero-sum game problem can be formulated as follows:

$$V(x_k) = \min_u \max_w \sum_{i=k}^{\infty} [x_i^T Q x_i + u_i^T u_i - \gamma^2 w_i^T w_i]$$

- The goal is to find the optimal strategies (State-feedback) for this multi-agent problem

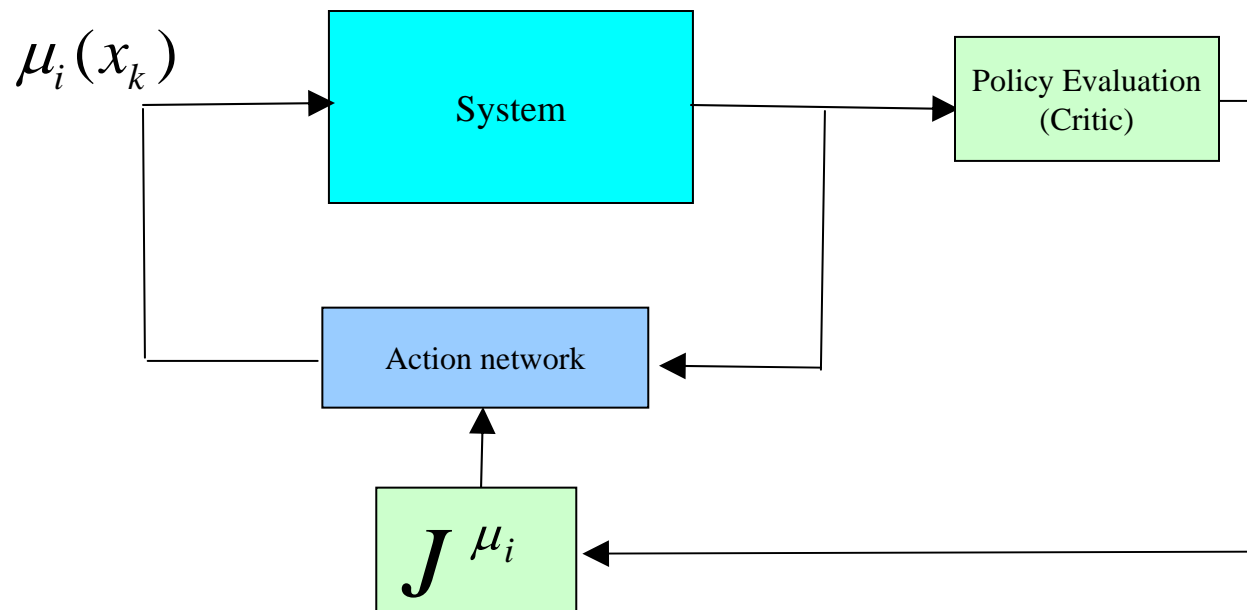
$$u^*(x) = L^* x \quad w^*(x) = K^* x$$

- Using Bellman optimality principle “Dynamic Programming”

$$\begin{aligned} V^*(x_k) &= \min_{u_k} \max_{w_k} (r(x_k, u_k) + V^*(x_{k+1})) \\ &= \min_{u_k} \max_{w_k} (x_k^T Q x_k + u_k^T u_k - \gamma^2 w_k^T w_k + x_{k+1}^T P x_{k+1}). \end{aligned}$$

# ADP Adaptive Critics

- The Adaptive Critics Architecture



# Heuristic Dynamic Programming: Forward-in-time Formulation

---

- This is an Approximate Dynamic Programming Scheme (ADP) where one has the following incremental optimization

$$V_{i+1}(x_k) = \min_{u_k} \max_{w_k} \{x_k^T Q x_k + u_k^T u_k - \gamma^2 w_k^T w_k + V_i(x_{k+1})\}$$

which is equivalently written as

$$V_{i+1}(x_k) = x_k^T Q x_k + u_i^T(x_k) u_i(x_k) - \gamma^2 w_i^T(x_k) w_i(x_k) + V_i(x_{k+1})$$

- Neural networks are used to have closed form representation of

$$\hat{V}(x, p_i) = p_i^T \bar{x} \quad \hat{u}(x, L_i) = L_i^T x \quad \hat{w}(x, K_i) = K_i^T x$$

$$d(x_k, p_i) = x_k^T Q x_k + (L_i x_k)^T (L_i x_k) - \gamma^2 (K_i x_k)^T (K_i x_k) + p_i^T \bar{x}_{k+1}$$

$$p_{i+1} = \arg \min_{p_{i+1}} \left\{ \int_{\Omega} |p_{i+1}^T \bar{x} - d(x, p_i)|^2 dx \right\}$$

- The HDP algorithm in fact is iteration on the Riccati equation

$$P_{i+1} = A^T P_i A + Q - [A^T P_i B \quad A^T P_i E] \begin{bmatrix} I + B^T P_i B & B^T P_i E \\ E^T P_i A & E^T P_i E - \gamma^2 I \end{bmatrix}^{-1} \begin{bmatrix} B^T P_i A \\ E^T P_i A \end{bmatrix}$$

## Two ways for Discrete-time Policy Iterations

---

$$P_j^{i+1} - A_j^T P_j^{i+1} A_j = Q + L_j^T R L_j - \gamma^2 K_j^{iT} K_j^i$$

$$A_j = A + B L_j$$

$$K_j^i = -\gamma^{-2} (E^T P_j^i E - \gamma^2 I)^{-1} E^T P_j^i A_j$$

$$L_j = -(I + B^T P_i B)^{-1} B^T P_i (A + E K_j^\infty)$$

Requires stable  
initial policy

$$P_{i+1} = A^T P_i A + Q - [A^T P_i B \quad A^T P_i E] \begin{bmatrix} I + B^T P_i B & B^T P_i E \\ E^T P_i A & E^T P_i E - \gamma^2 I \end{bmatrix}^{-1} \begin{bmatrix} B^T P_i A \\ E^T P_i A \end{bmatrix}$$

$$P_0 = 0$$

Does not  
require a stable  
initial policy

## DT HDP vs. Finite Horizon Optimal Control

---

$$P_{i+1} = A^T P_i A + Q - A^T P_i B (I + B^T P_i B)^{-1} B^T P_i A$$

$$P_0 = 0$$

Forward-in-time policy iteration

$$P_k = A^T P_{k+1} A + Q - A^T P_{k+1} B (I + B^T P_{k+1} B)^{-1} B^T P_{k+1} A$$

$$P_N = 0$$

Backward-in-time optimization

## DT DHP

□ DHP solution of the GARE

$$\begin{aligned}\lambda_{i+1}(x_k) &= \frac{\partial V_{i+1}(x_k)}{\partial x_k} \\ &= \frac{\partial r(x_k, u_i(x_k), w_i(x_k))}{\partial x_k} + \left( \frac{\partial u_i(x_k)}{\partial x_k} \right)^T \frac{\partial r(x_k, u_i(x_k), w_i(x_k))}{\partial u_i(x_k)} + \\ &\quad \left( \frac{\partial w_i(x_k)}{\partial x_k} \right)^T \frac{\partial r(x_k, u_i(x_k), w_i(x_k))}{\partial w_i(x_k)} + \\ &\quad \left( \frac{\partial x_{k+1}}{\partial x_k} \right)^T \frac{\partial V_i(x_{k+1})}{\partial x_{k+1}} + \left( \frac{\partial u_i(x_k)}{\partial x_k} \right)^T \left( \frac{\partial x_{k+1}}{\partial u_i(x_k)} \right)^T \frac{\partial V_i(x_{k+1})}{\partial x_{k+1}} + \\ &\quad \left( \frac{\partial w_i(x_k)}{\partial x_k} \right)^T \left( \frac{\partial x_{k+1}}{\partial w_i(x_k)} \right)^T \frac{\partial V_i(x_{k+1})}{\partial x_{k+1}}.\end{aligned}$$

# Q-Learning: Action Dependent Heuristic Dynamic Programming

- Dynamic Programming: Backward-in-time

$$Q^*(x_k, u_k, w_k) = (x_k^T R x_k + u_k^T u_k - \gamma^2 w_k^T w_k + V^*(x_{k+1}))$$

$$\Rightarrow (u_k^*, w_k^*) = \arg\{ \min_{u_k} \max_{w_k} Q^*(x_k, u_k, w_k) \}$$

- Heuristic Dynamic Programming: Forward-in-time

$$Q_{i+1}(x_k, u_k, w_k) = x_k^T R x_k + u_k^T u_k - \gamma^2 w_k^T w_k + \min_{u_{k+1}} \max_{w_{k+1}} Q_i(x_{k+1}, u_{k+1}, w_{k+1})$$

$$= x_k^T R x_k + u_k^T u_k - \gamma^2 w_k^T w_k + V_i(x_{k+1})$$

$$= x_k^T R x_k + u_k^T u_k - \gamma^2 w_k^T w_k + V_i(Ax_k + Bu_k + Ew_k)$$

- Neural networks are used to have closed form representation of

$$h_{i+1}^T \bar{z}(x_k) = d(\bar{z}(x_k), h_i) \quad \hat{u}_{ei}(x_k) = L_i x_k + n_{1k} \quad \hat{w}_{ei}(x_k) = K_i x_k + n_{2k}$$

$$d(z_k(x_k), H_i) = x_k^T R x_k + \hat{u}_i(x_k)^T \hat{u}_i(x_k) - \gamma^2 \hat{w}_i(x_k)^T \hat{w}_i(x_k) +$$

$$Q_i(x_{k+1}, \hat{u}_i(x_{k+1}), \hat{w}_i(x_{k+1}))$$

# Convergence Proofs

$$h_{i+1} = (ZZ^T)^{-1}ZY$$

$$Z = [\bar{z}(p1) \quad \bar{z}(p2) \quad \cdots \quad \bar{z}(pN)]$$

$$Y = [d(z(p1), h_i) \quad d(z(p2), h_i) \quad \cdots \quad d(z(pN), h_i)]^T.$$

## □ Convergence

$$H_{i+1} = \begin{bmatrix} Q & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & -\gamma^2 I \end{bmatrix} + \begin{bmatrix} A & B & E \\ L_i A & L_i B & L_i E \\ K_i A & K_i B & K_i E \end{bmatrix}^T H_i \begin{bmatrix} A & B & E \\ L_i A & L_i B & L_i E \\ K_i A & K_i B & K_i E \end{bmatrix}$$

- **The result is a model free Adaptive Controller that converge to an H-infinity optimal controller**
- **No requirement what so ever on the model plant matrices**

Start of the Zero-Sum  
AD HDP

Initialization

$$h_0 = v(h_0) = 0 : P_0 = 0$$

$$i = 0, L_0 = 0, K_0 = 0.$$

Solving the least-squares

$$Z = [\bar{z}|_{x_{k-N-1}} \quad \bar{z}|_{x_{k-N-2}} \quad \cdots \quad \bar{z}|_{x_{k-1}}]$$

$$Y = [d(\bar{z}_{k-N-1}, h_i) \quad d(\bar{z}_{k-N-2}, h_i) \quad \cdots \quad d(\bar{z}_{k-1}, h_i)]^T.$$

$$h_{i+1} = (ZZ^T)^{-1}ZY \quad ; H_{i+1} = f(h_{i+1})$$

Policy iteration

$$L_{i+1} = (H_{uu} - H_{uw}H_{ww}^{-1}H_{wu})(H_{uw}H_{ww}^{-1}H_{wx} - H_{ux}),$$

$$K_{i+1} = (H_{ww} - H_{wu}H_{uu}^{-1}H_{uw})(H_{wu}H_{uu}^{-1}H_{ux} - H_{wx})$$

$$\|h_{i+1} - h_i\|_F < \varepsilon$$

No

$$i \rightarrow i + 1$$

Yes

Finish

# Online ADP H-infinity Autopilot Controller Design for an F-16 Aircraft

- The F-16 short period dynamics has three states given as

$$x^T = [\alpha \quad q \quad \delta_e]$$

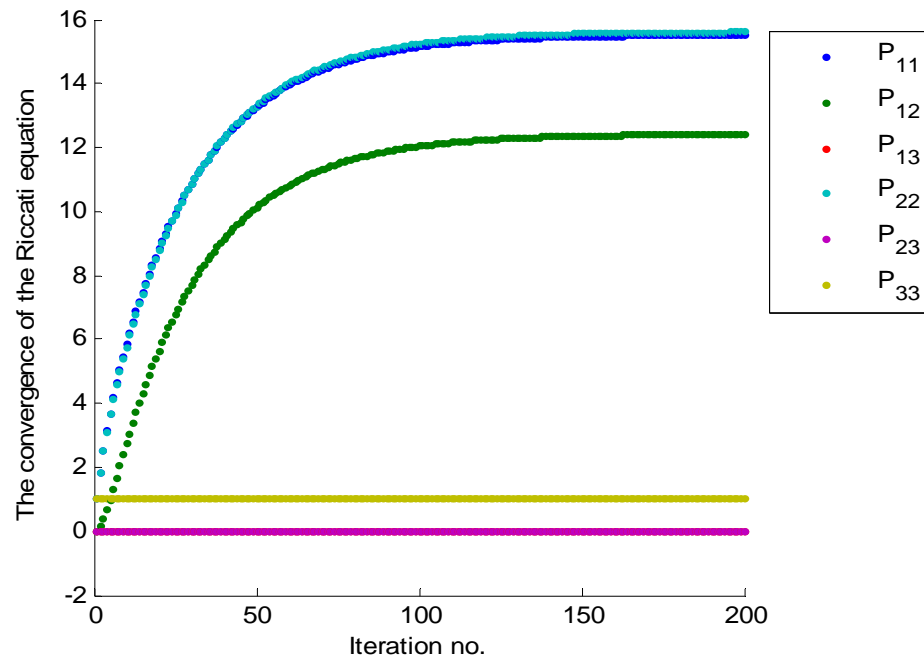
- Where  $\alpha$  is the angle of attack,  $q$  is the pitch rate and  $\delta_e$  is the elevator deflection angle.
- The zero-order hold discretization technique used to discretized the continuous time model. The discretized model is

$$A = \begin{bmatrix} 0.906488 & 0.0816012 & -0.0005 \\ 0.0741349 & 0.90121 & -0.000708383 \\ 0 & 0 & 0.132655 \end{bmatrix}$$

$$B = \begin{bmatrix} -0.00150808 \\ -0.0096 \\ 0.867345 \end{bmatrix} \quad E = \begin{bmatrix} 0.00951892 \\ 0.00038373 \\ 0 \end{bmatrix}$$

# The Offline Results

## □ Solution Based on the Riccati Equation



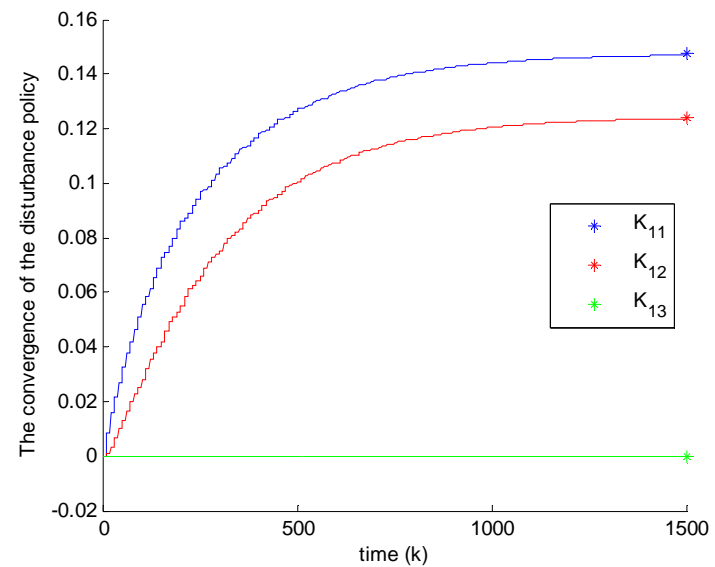
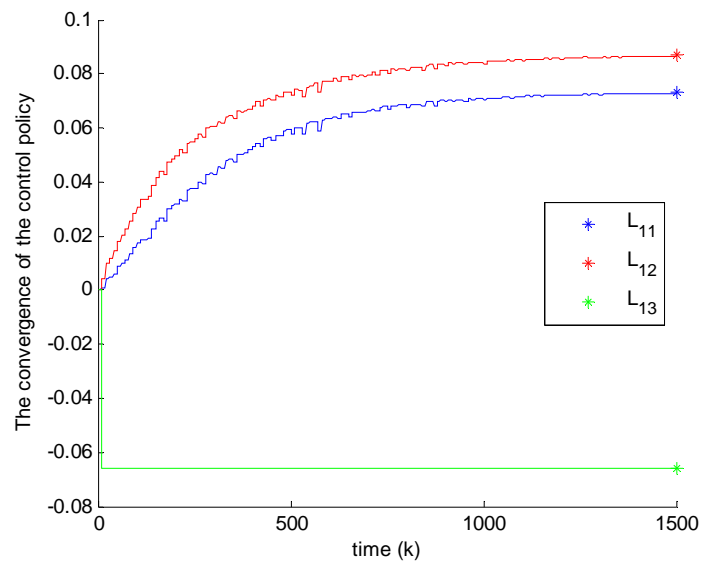
$$P = \begin{bmatrix} 15.5109 & 12.4074 & -0.0089 \\ 12.4074 & 15.5994 & -0.0078 \\ -0.0089 & -0.0078 & 1.0101 \end{bmatrix}$$

$$L = [0.0733 \quad 0.0872 \quad -0.0661]$$

$$K = [0.1476 \quad 0.1244 \quad 0]$$

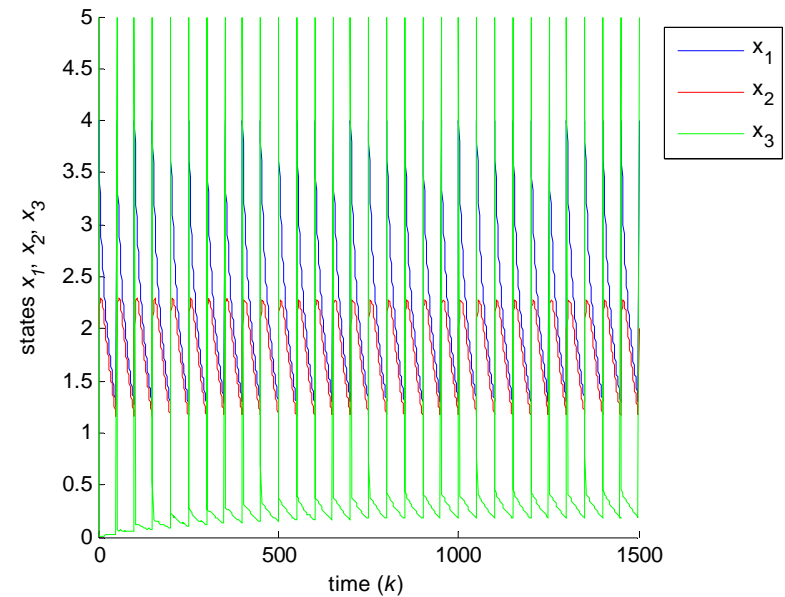
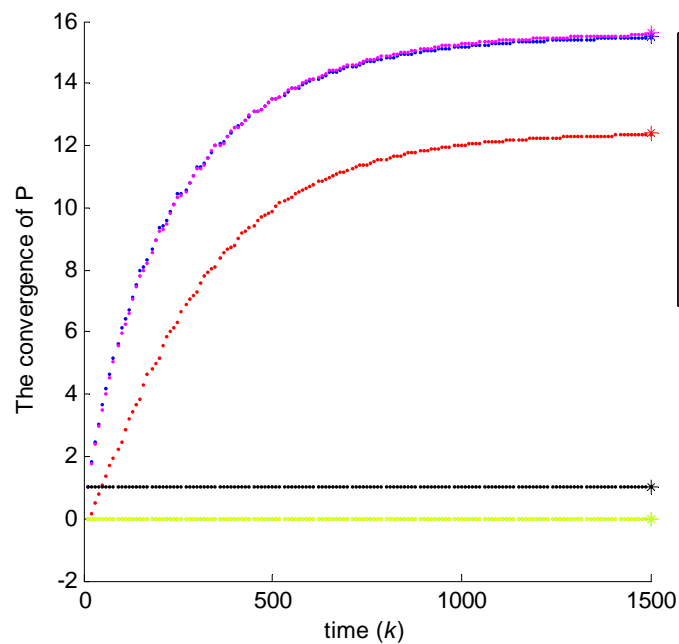
# Online-tuning: HDP based Autopilot Controller Design

## □ Convergence of the *Actions networks*



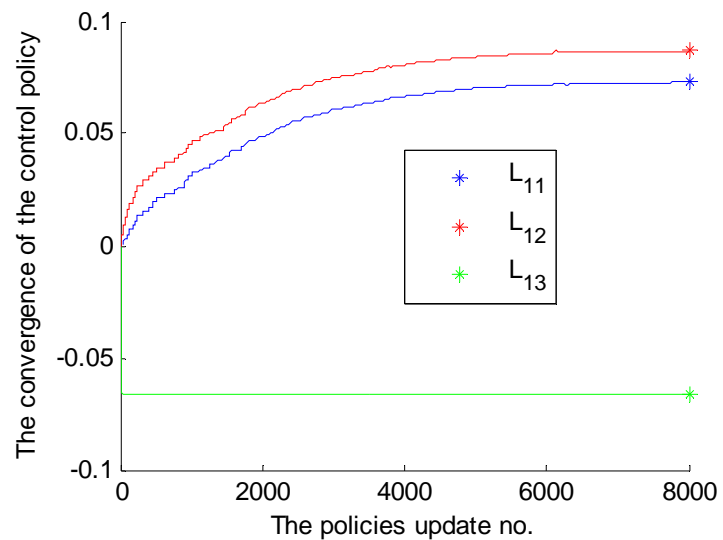
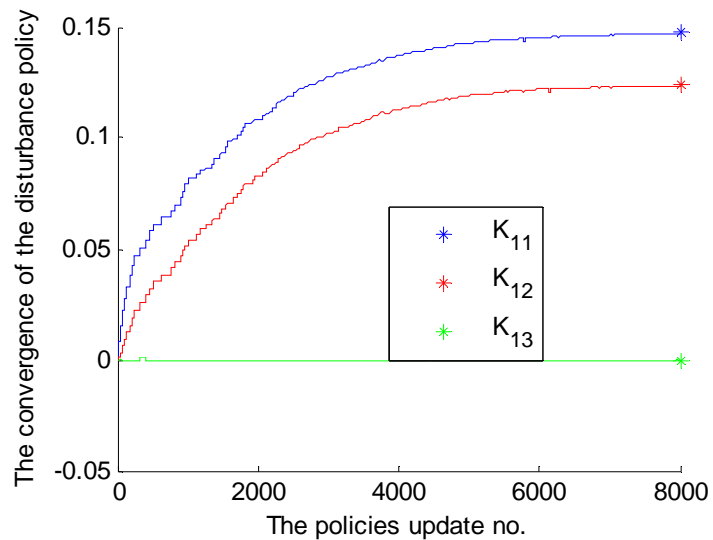
# Online-tuning: HDP based Autopilot Controller Design

## □ Convergence of the *Critic Network*



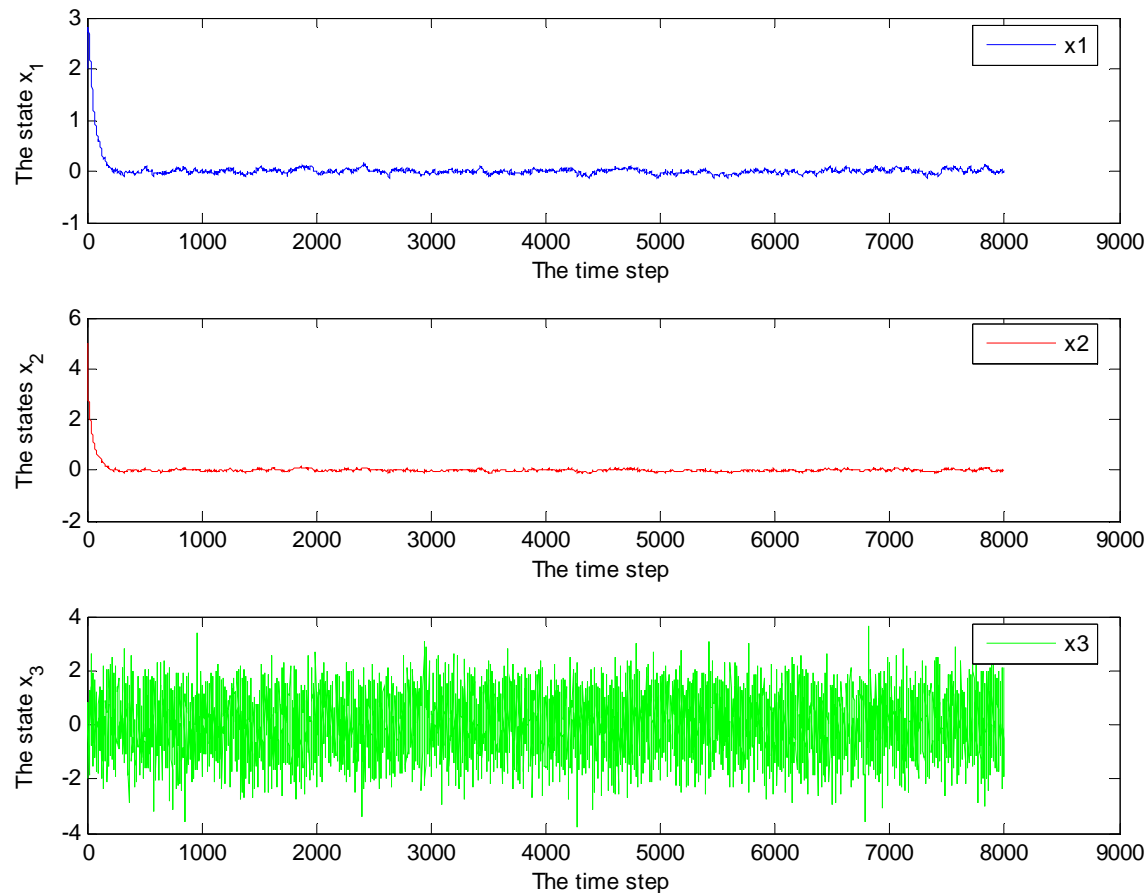
# Online-tuning: Q-learning based HDP Autopilot Controller Design

## □ Convergence of the *Action* Networks



# Online-tuning: Q-learning based HDP Autopilot Controller Design

## □ States trajectories



Thanks for your attention!