F.L. Lewis & Draguna Vrabie
Moncrief-O'Donnell Endowed Chair
Head, Controls & Sensors Group

Supported by :
NSF - PAUL WERBOS

Automation & Robotics Research Institute (ARRI)
The University of Texas at Arlington

Adaptive Dynamic Programming (ADP) For Discrete-Time Systems

The University of Texas ARLINGTON

Talk available online at
http://ARRI.uta.edu/acs

ARRI

Bill Wolovich

"Linear Multivariable Systems" New York: Springer-Verlag, 1974.
"Robotics: Basic Analysis and Design" , 1987.
“Automatic Control Systems: Basic Analysis and Design,” Wolovich, 1994.

Interactor Matrix & Structure

Falb and Wolovich, “Decoupling in the design and synthesis of multivariable control systems, IEEE Trans. Automatic Control,” 1967.
Wolovich and Falb, “On the structure of multivariable systems,” SIAM J. Control, 1969.
Wolovich, “The use of state feedback for exact model matching,” SIAM J. Control, 1972.
Falb and Wolovich, “The role of the interactor in decoupling, JACC, 1977.
Invariants and canonical forms under dynamic compensation, W. Wolovich and P. Falb, SIAM, J. on Control, 14, 1976.

The solution of the input-output cover problems
WOLOVICH [1972], ***MORSE*** [1976], HAMMER and HEYMANN [1981], WONHAM [1974

Pole Placement via Static Output Feedback is NP-Hard
Morse, A.S., Wolovich, W.A., Anderson, B.D.O.. "GENERIC POLE ASSIGNMENT - PRELIMINARY-RESULTS." IEEE Transactions on Automatic Control 28 503 - 506, 1983.

Discrete-Time Optimal Control

system $x_{k+1} = f(x_k) + g(x_k)u_k$

cost $V_h(x_k) = \sum_{i=k}^{\infty} \gamma^{i-k} r(x_i, u_i)$

Example $r(x_k, u_k) = x_k^T Q x_k + u_k^T R u_k$

$$V_h(x_k) = r(x_k, u_k) + \gamma \sum_{i=k+1}^{\infty} \gamma^{i-(k+1)} r(x_i, u_i)$$

Value function recursion $V_h(x_k) = r(x_k, h(x_k)) + \gamma V_h(x_{k+1})$, $V_h(0) = 0$

Control policy $u_k = h(x_k)$ = the prescribed control input function

Example $u_k = -Kx_k$ Linear state variable feedback

Discrete-Time Optimal Control

cost $V_h(x_k) = \sum_{i=k}^{\infty} \gamma^{i-k} r(x_i, u_i)$

Value function recursion $V_h(x_k) = r(x_k, h(x_k)) + \gamma V_h(x_{k+1})$

$u_k = h(x_k)$ = the prescribed control policy

Hamiltonian $H(x_k, \nabla V(x_k), h) = r(x_k, h(x_k)) + \gamma V_h(x_{k+1}) - V_h(x_k)$

Optimal cost $V^*(x_k) = \min_h (r(x_k, h(x_k)) + \gamma V_h(x_{k+1}))$

Bellman's Principle $V^*(x_k) = \min_{u_k} (r(x_k, u_k) + \gamma V^*(x_{k+1}))$

Backwards in time solution

Optimal Control $h^*(x_k) = \arg\min_{u_k} (r(x_k, u_k) + \gamma V^*(x_{k+1}))$

System dynamics does not appear

The Solution: Hamilton-Jacobi-Bellman Equation

System $$x_{k+1} = f(x_k) + g(x_k)u_k$$

$$V(x_k) = \sum_{i=k}^{\infty} x_i^T Q x_i + u_i^T R u_i$$

DT HJB equation

Difficult to solve
Contains the dynamics

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

$$= \min_{u_k}\left[x_k^T Q x_k + u_k^T R u_k + V^*\left(f(x_k) + g(x_k)u_k \right) \right]$$

Minimize wrt u_k

$$2Ru_k + g(x_k)^T \frac{dV^*(x_{k+1})}{dx_{k+1}} = 0$$

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

DT Optimal Control – Linear Systems Quadratic cost (LQR)

system

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

cost $$V(x_k) = \sum_{i=k}^{\infty} x_i^T Q x_i + u_i^T R u_i$$

Fact. The cost is quadratic $V(x_k) = x_k^T P x_k$ for some symmetric matrix P

HJB = DT Riccati equation

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

Optimal Control $u_k = -Lx_k$

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

Optimal Cost

$$V^*(x_k) = x_k^T P x_k$$

Off-line solution
Dynamics must be known

Discrete-Time Optimal Adaptive Control

cost $V_h(x_k) = \sum_{i=k}^{\infty} \gamma^{i-k} r(x_i, u_i)$

Value function recursion $V_h(x_k) = r(x_k, h(x_k)) + \gamma V_h(x_{k+1})$

$u_k = h(x_k)$ = the prescribed control policy

Hamiltonian $H(x_k, \nabla V(x_k), h) = r(x_k, h(x_k)) + \gamma V_h(x_{k+1}) - V_h(x_k)$

Optimal cost $V^*(x_k) = \min_h (r(x_k, h(x_k)) + \gamma V_h(x_{k+1}))$

Bellman's Principle $V^*(x_k) = \min_{u_k} (r(x_k, u_k) + \gamma V^*(x_{k+1}))$

Optimal Control $h^*(x_k) = \arg\min_{u_k} (r(x_k, u_k) + \gamma V^*(x_{k+1}))$

Focus on these two eqs

Discrete-Time Optimal Control

Solutions by Comp. Intelligence Community

Value function recursion

$$V_h(x_k) = r(x_k, h(x_k)) + \gamma V_h(x_{k+1}), \qquad V_h(0) = 0$$

$u_k = h(x_k)$ = the prescribed control policy

The Lyapunov Equation

Theorem: Let $V_h(x_k)$ solve the Lyapunov equation. Then

$$V_h(x_k) = \sum_{i=k}^{\infty} \gamma^{i-k} r(x_i, h(x_i))$$

Gives value for any prescribed control policy

Policy Evaluation for any given current policy

Policy must be stabilizing

Optimal Control $h^*(x_k) = \arg\min_{u_k}(r(x_k, u_k) + \gamma V^*(x_{k+1}))$

Bellman's result

What about? -

$h'(x_k) = \arg\min_{u_k}(r(x_k, u_k) + \gamma V_h(x_{k+1}))$ for a given policy *h(.)* ?

Theorem. Bertsekas.

Let $V_h(x_k)$ be the value of any given policy $h(x_k)$.

Then

$$V_{h'}(x_k) \leq V_h(x_k)$$

Policy Improvement

One step improvement property of Rollout Algorithms

DT Policy Iteration

e.g. Control policy = SVFB

$$h(x_k) = -Lx_k$$

Cost for any given control policy $h(x_k)$ satisfies the recursion

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

Lyapunov eq.

Recursive form
Consistency equation

Recursive solution

Pick stabilizing initial control

Policy Evaluation

$$V_{j+1}(x_k) = r(x_k, h_j(x_k)) + \gamma V_{j+1}(x_{k+1})$$

f(.) and g(.) do not appear

Policy Improvement

$$h_{j+1}(x_k) = \arg\min_{u_k}(r(x_k, u_k) + \gamma V_{j+1}(x_{k+1}))$$

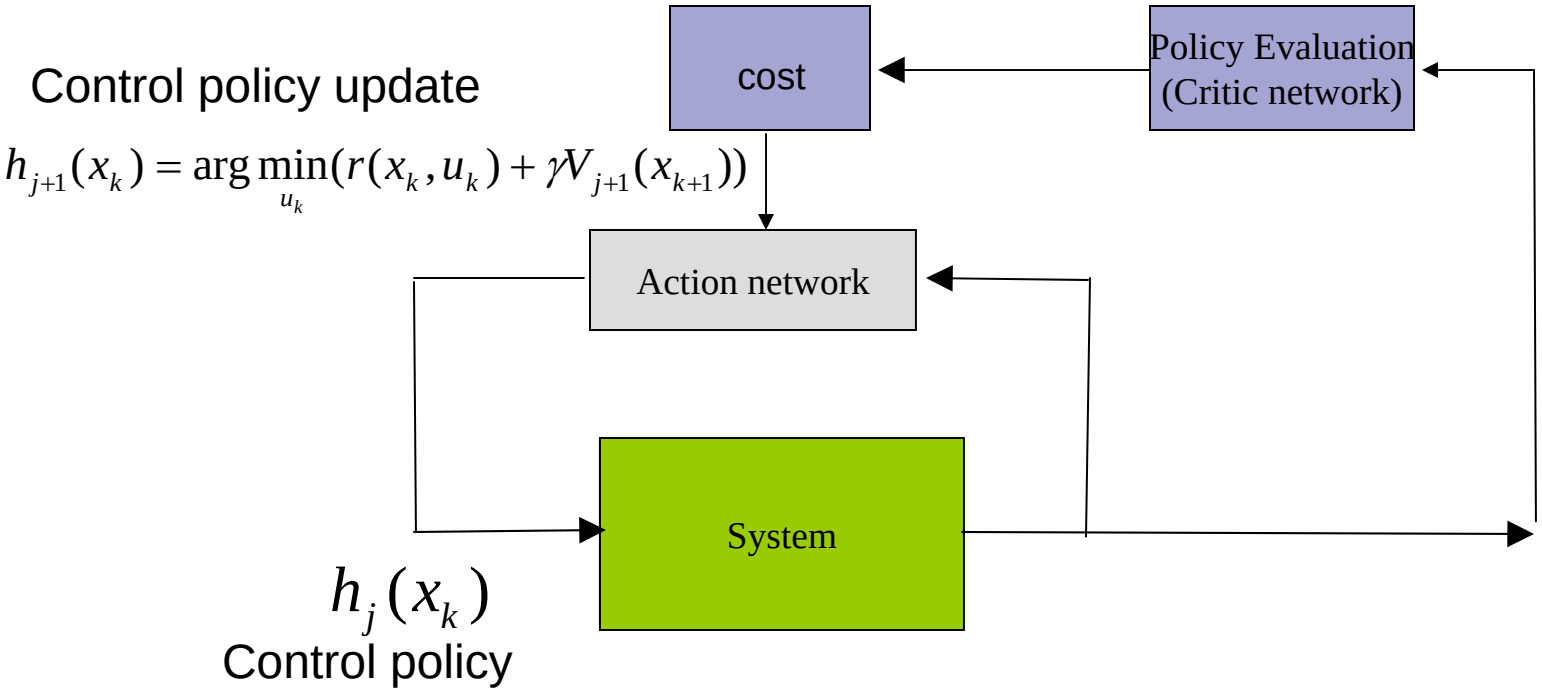
Howard (1960) proved convergence for MDP

Adaptive Critics

The Adaptive Critic Architecture

Value update

$$V_{j+1}(x_k) = r(x_k, h_j(x_k)) + \gamma V_{j+1}(x_{k+1})$$

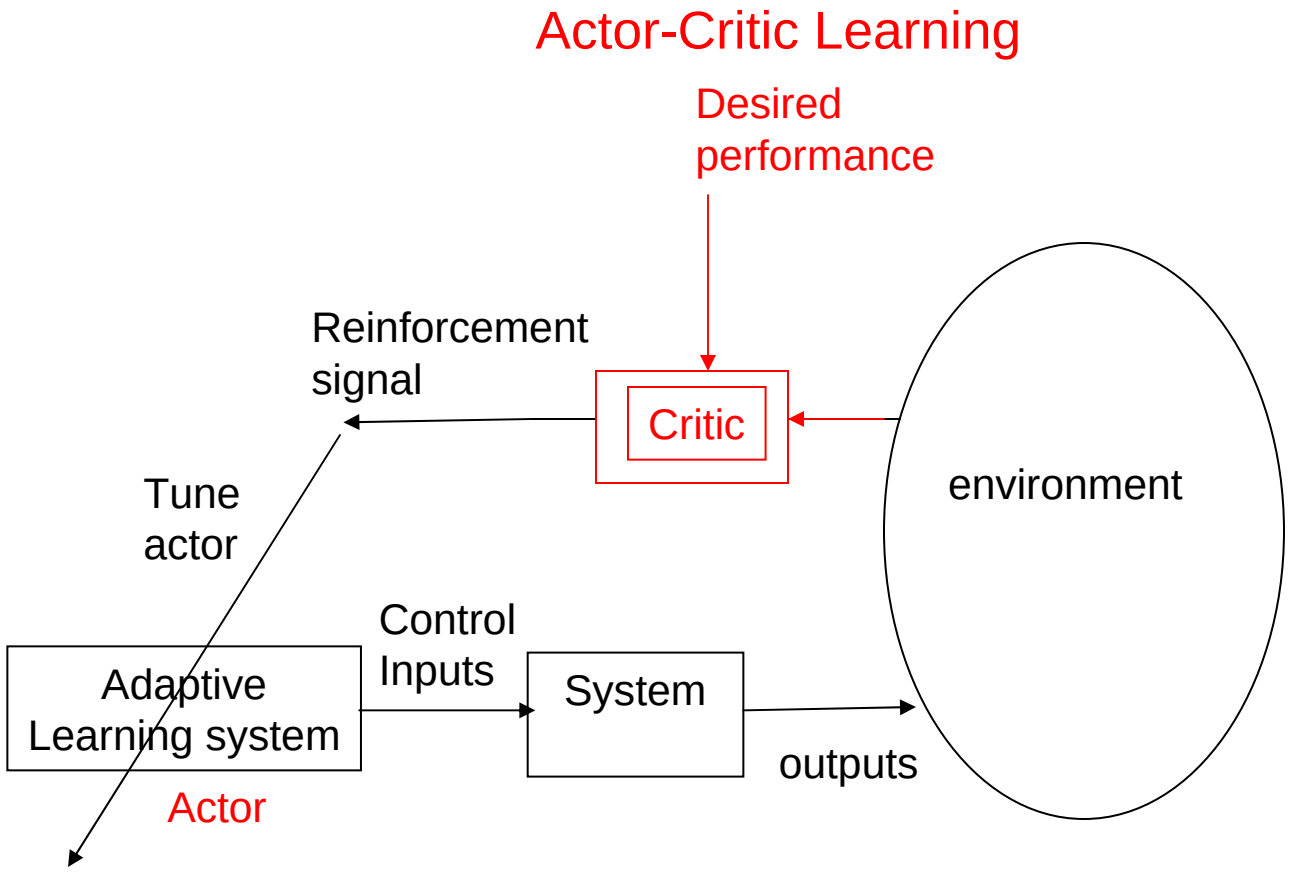


Leads to ONLINE FORWARD-IN-TIME implementation of optimal control

Different methods of learning

Reinforcement learning
Ivan Pavlov 1890s

We want OPTIMAL performance
- ADP- Approximate Dynamic Programming



Adaptive (Approximate) Dynamic Programming

Four ADP Methods proposed by Paul Werbos

Critic NN to approximate:

Heuristic dynamic programming

Value $V(x_k)$

AD Heuristic dynamic programming
(Watkins Q Learning)

Q function $Q(x_k, u_k)$

Dual heuristic programming

Gradient $\frac{\partial V}{\partial x}$

AD Dual heuristic programming

Gradients $\frac{\partial Q}{\partial x}, \frac{\partial Q}{\partial u}$

Action NN to approximate the Control

Bertsekas- Neurodynamic Programming

Barto & Bradtke- Q-learning proof (Imposed a settling time)

DT Policy Iteration – Linear Systems Quadratic Cost- LQR

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

For any stabilizing policy, the cost is

$$V(x_k) = \sum_{i=k}^{\infty} x_i^T Q x_i + u^T(x_i) R u(x_i)$$

LQR value is quadratic $V(x) = x^T P x$

Solves Lyapunov eq. without knowing A and B

DT Policy iterations

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

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

Equivalent to an Underlying Problem- DT LQR:

$$(A - BL_j)^T P_{j+1} (A - BL_j) - P_{j+1} = -Q - L_j^T R L_j$$

DT Lyapunov eq.

$$L_{j+1} = (R + B^T P_{j+1} B)^{-1} B^T P_{j+1} A$$

Hewer proved convergence in 1971

ADP Solves Riccati equation WITHOUT knowing System Dynamics

DT Policy Iteration – How to implement online?
Linear Systems Quadratic Cost- LQR

$$x_{k+1} = Ax_k + Bu_k \qquad V(x_k) = \sum_{i=k}^{\infty} x_i^T Q x_i + u(x_i) R u(x_i)$$

LQR cost is quadratic $V(x) = x^T P x$ for some matrix P

Solves Lyapunov eq. without knowing A and B

DT Policy iterations

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

$$x_k^T P_{j+1} x_k - x_{k+1}^T P_{j+1} x_{k+1} = x_k^T Q x_k + u_j^T R u_j$$

$$\begin{bmatrix} x_k^1 & x_k^2 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} \begin{bmatrix} x_k^1 \\ x_k^2 \end{bmatrix} - \begin{bmatrix} x_{k+1}^1 & x_{k+1}^2 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} \begin{bmatrix} x_{k+1}^1 \\ x_{k+1}^2 \end{bmatrix}$$

$$= \begin{bmatrix} p_{11} & p_{12} & p_{22} \end{bmatrix} \begin{bmatrix} (x_k^1)^2 \\ 2x_k^1 x_k^2 \\ (x_k^2)^2 \end{bmatrix} - \begin{bmatrix} p_{11} & p_{12} & p_{22} \end{bmatrix} \begin{bmatrix} (x_{k+1}^1)^2 \\ 2x_{k+1}^1 x_{k+1}^2 \\ (x_{k+1}^2)^2 \end{bmatrix}$$

$$= W_{j+1}^T \left[\varphi(x_k) - \varphi(x_{k+1}) \right]$$

Quadratic basis set

Implementation- DT Policy Iteration

Value Function Approximation (VFA)

$$V(x) = W^T \varphi(x)$$

weights

basis functions

LQR case- V(x) is quadratic

$$V(x) = x^T P x = W^T \varphi(x)$$

$$\varphi(x) = \left[x_1^2, \ldots, x_1 x_n, x_2^2, \ldots, x_2 x_n, \ldots, x_n^2\right]'.$$

Quadratic basis functions

$$W^T = [p_{11} \quad p_{12} \quad \cdots]$$

Nonlinear system case- use Neural Network

Implementation- DT Policy Iteration

Value function update for given control

$$V_{j+1}(x_k) = r(x_k, h_j(x_k)) + \gamma V_{j+1}(x_{k+1})$$

Assume measurements of x_k and x_{k+1} are available to compute u_{k+1}

VFA $\quad V_j(x_k) = W_j^T \varphi(x_k)$

Then

regression matrix

$$W_{j+1}^T [\varphi(x_k) - \gamma \varphi(x_{k+1})] = r(x_k, h_j(x_k))$$

Since x_{k+1} is measured,
do not need knowledge of f(x)
or g(x) for value fn. update

Indirect Adaptive control with identification of the optimal value

Solve for weights using RLS
or, many trajectories with different initial conditions over a compact set

Then update control using

$$h_j(x_k) = L_j x_k = (R + B^T P_j B)^{-1} B^T P_j A x_k$$

Need to know $f(x_k)$ AND $g(x_k)$
for control update

Model-Based Policy Iteration

Robustness??

1. Select control policy

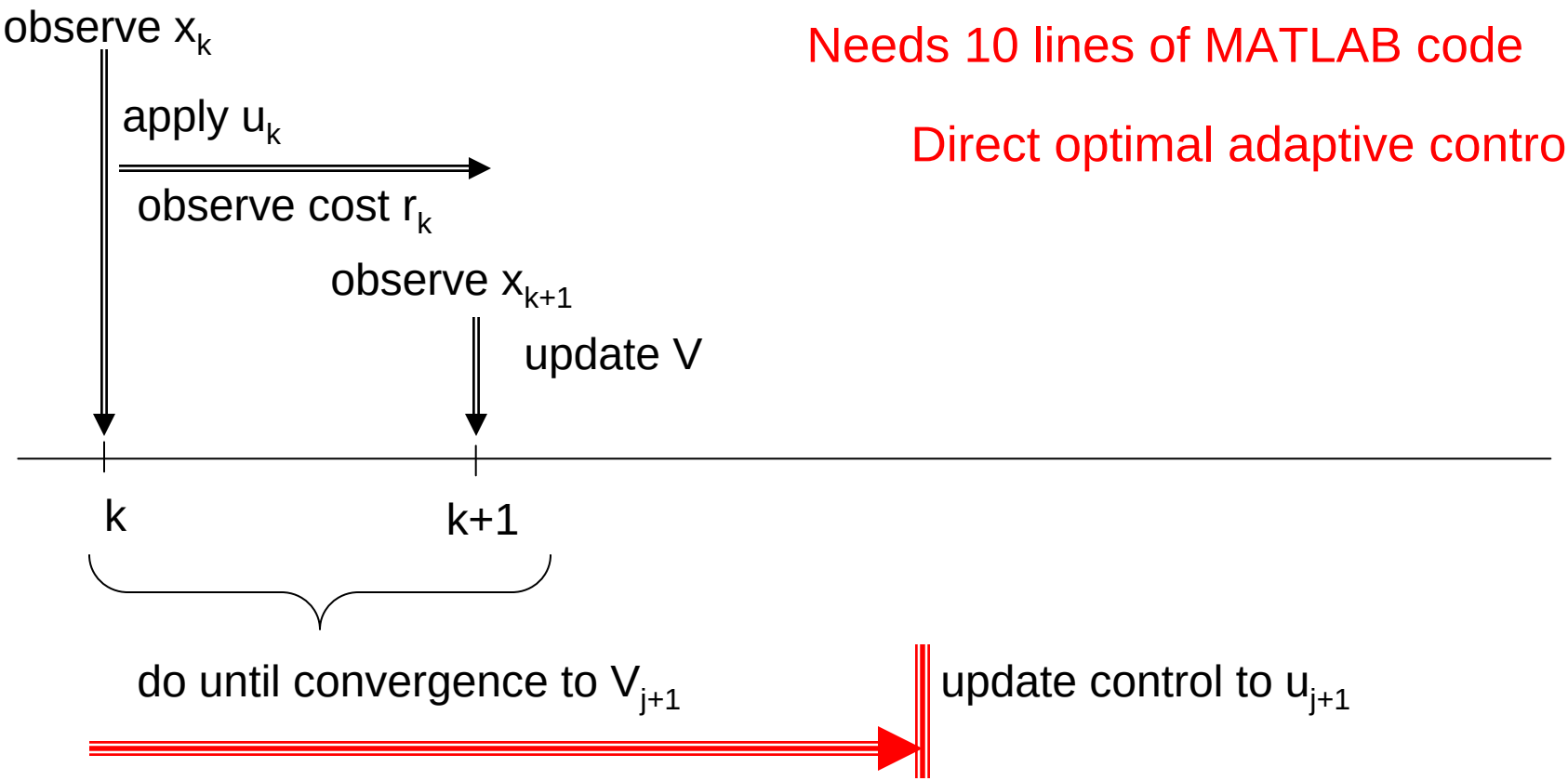
Solves Lyapunov eq. without knowing dynamics

2. Find associated cost $V_{j+1}(x_k) = r(x_k, h_j(x_k)) + \gamma V_{j+1}(x_{k+1})$

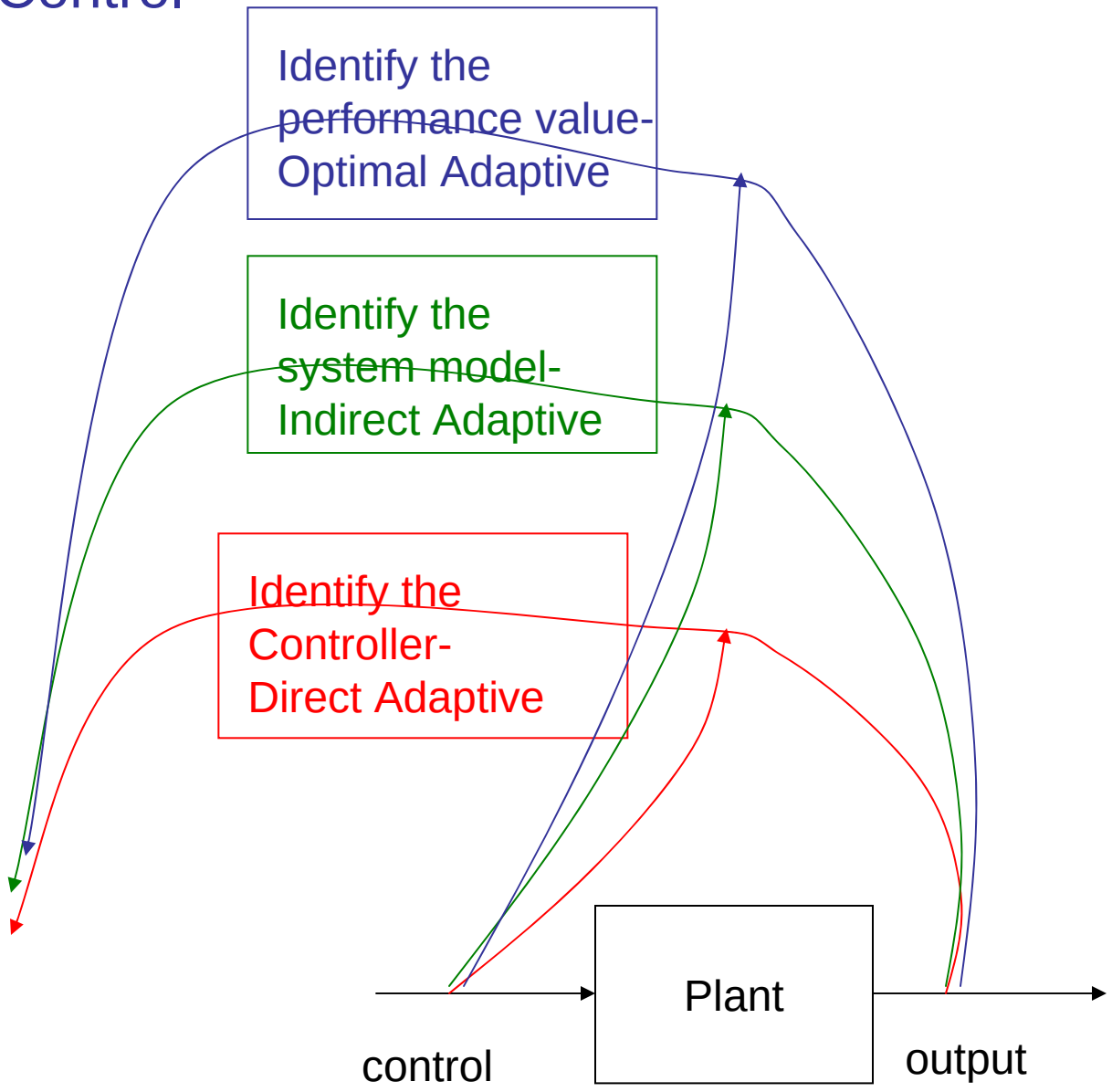
3. Improve control $u_{j+1}(x_k) = -\frac{1}{2} R^{-1} g(x_k)^T \frac{dV_j(x_{k+1})}{dx_{k+1}}$

Needs 10 lines of MATLAB code

Direct optimal adaptive control



Adaptive Control



Greedy Value Fn. Update- Approximate Dynamic Programming
ADP Method 1 - Heuristic Dynamic Programming (HDP)

Paul Werbos

Policy Iteration

$$V_{j+1}(x_k) = r(x_k, h_j(x_k)) + \gamma V_{j+1}(x_{k+1})$$

$$h_{j+1}(x_k) = \arg\min_{u_k}(r(x_k, u_k) + \gamma V_{j+1}(x_{k+1}))$$

Lyapunov eq.

For LQR Underlying RE

$$(A - BL_j)^T P_{j+1}(A - BL_j) - P_{j+1} = -Q - L_j^T R L_j$$

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

Hewer 1971

Initial stabilizing control is needed

ADP Greedy Cost Update

Two occurrences of cost allows def. of greedy update

$$V_{j+1}(x_k) = r(x_k, h_j(x_k)) + \gamma V_j(x_{k+1})$$

$$h_{j+1}(x_k) = \arg\min_{u_k}(r(x_k, u_k) + \gamma V_{j+1}(x_{k+1}))$$

Simple recursion

For LQR Underlying RE

$$P_{j+1} = (A - BL_j)^T P_j (A - BL_j) + Q + L_j^T R L_j$$

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

Lancaster & Rodman proved convergence

Initial stabilizing control is NOT needed

Implementation- DT HDP

Value function update for given control

$$V_{j+1}(x_k) = r(x_k, h_j(x_k)) + \gamma V_j(x_{k+1})$$

Since x_{k+1} is measured, do not need knowledge of f(x) or g(x) for value fn. update

Assume measurements of x_k and x_{k+1} are available to compute u_{k+1}

VFA $\quad V_j(x_k) = W_j^T \varphi(x_k)$

Then

regression matrix

Old weights

$$W_{j+1}^T \left[\varphi(x_k)\right] = r(x_k, h_j(x_k)) + \gamma W_j^T \left[\varphi(x_{k+1})\right]$$

Solve for weights using RLS

or, many trajectories with different initial conditions over a compact set

Then update control using

$$h_j(x_k) = L_j x_k = -(R + B^T P_j B)^{-1} B^T P_j A x_k$$

Need to know $f(x_k)$ AND $g(x_k)$ for control update

DT HDP vs. Receding Horizon Optimal Control

Forward-in-time HDP

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

$$P_0 = 0$$

Backward-in-time optimization – RHC

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

$P_N =$ Control Lyapunov Function overbounding P_∞

Adaptive Terminal Cost RHC

Hongwei Zhang
Dr. Jie Huang

Standard RHC

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

$$V(x_k) = \sum_{i=k}^{k+N-1} \left(x_i^T Q x_i + u_i^T R u_i \right) \quad + \quad x_{k+N}^T P_0 x_{k+N}$$

P_0 is the same for each stage

$$P_{i+1} = A^T P_i A + Q - A^T P_i B (R + B^T P_i B)^{-1} B^T P_i A, \quad P_0$$

$$u_{k+1}^{RH} = -(R + B^T P_{N-1} B)^{-1} B^T P_{N-1} A \, x_{k+1} = -L_N^{RH} x_{k+1}$$

Requires P_0 to be a CLF that overbounds the optimal inf. horizon cost, or large N

Our ATC RHC

Final cost from previous stage

$$V(x_k) = \sum_{i=k}^{k+N-1} \left(x_i^T Q x_i + u_i^T R u_i \right) \quad + \quad x_{k+N}^T P_{kN} x_{k+N}$$

$$P_{i+1} = A^T P_i A + Q - A^T P_i B (R + B^T P_i B)^{-1} B^T P_i A, \quad P_{kN}$$

HWZ Theorem- Let $N \geq 1$

under the usual suspect observability and controllability assumptions
ATC RHC guarantees ultimate uniform exponential stability
for ANY $P_0 > 0$.
Moreover, our solution converges to the optimal inf. horizon cost.

Q Learning - Action Dependent ADP

Value function recursion for given policy $h(x_k)$

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

Define Q function

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

u_k arbitrary

policy $h(.)$ used after time k

Note $Q_h(x_k, h(x_k)) = V_h(x_k)$

Recursion for Q $Q_h(x_k, u_k) = r(x_k, u_k) + \gamma Q_h(x_{k+1}, h(x_{k+1}))$

Simple expression of Bellman's principle

$$V^*(x_k) = \min_{u_k}(Q^*(x_k, u_k)) \qquad h^*(x_k) = \arg\min_{u_k}(Q^*(x_k, u_k))$$

Optimal Adaptive Control (for unknown DT systems)

Continuous-Time Optimal Control

Draguna Vrabie

System $\dot{x} = f(x,u)$

Cost $V(x(t)) = \int_t^\infty r(x,u)\,dt = \int_t^\infty (Q(x) + u^T R u)\,dt$

Off-line solution
Dynamics must be known

Hamiltonian

$$H(x, \frac{\partial V}{\partial x}, u) = \dot{V} + r(x,u) = \left(\frac{\partial V}{\partial x}\right)^T \dot{x} + r(x,u) = \left(\frac{\partial V}{\partial x}\right)^T f(x,u) + r(x,u)$$

c.f. DT Hamiltonian $H(x_k, \nabla V(x_k), h) = r(x_k, h(x_k)) + \gamma V_h(x_{k+1}) - V_h(x_k)$

Optimal cost

Bellman $$0 = \min_{u(t)} \left(r(x,u) + \left(\frac{\partial V^*}{\partial x}\right)^T \dot{x} \right) = \min_{u(t)} \left(r(x,u) + \left(\frac{\partial V^*}{\partial x}\right)^T f(x,u) \right)$$

Optimal control $$h^*(x(t)) = -\tfrac{1}{2} R^{-1} g^T(x) \frac{\partial V^*}{\partial x}$$

HJB equation $$0 = \left(\frac{dV^*}{dx}\right)^T f + Q(x) - \tfrac{1}{4}\left(\frac{dV^*}{dx}\right)^T g R^{-1} g^T \frac{dV^*}{dx} \quad , \quad V(0) = 0$$

Bill Wolovich

Interactor Matrix & Structure Theorem

The solution of the input-output cover problems

Pole Placement via Static Output Feedback

Thank you for your inspiration and motivation in 1970

Q Function Definition

Specify a control policy $u_j = h(x_j); \quad j = k, k+1, \ldots.$

Define Q function

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

u_k arbitrary

policy *h(.)* used after time k

Note $Q_h(x_k, h(x_k)) = V_h(x_k)$

Recursion for Q $Q_h(x_k, u_k) = r(x_k, u_k) + \gamma Q_h(x_{k+1}, h(x_{k+1}))$

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

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

Optimal control solution

$$V^*(x_k) = Q^*(x_k, h^*(x_k)) = \min_h(Q_h(x_k, h(x_k))) \qquad h*(x_k) = \arg\min_h(Q_h(x_k, h(x_k))$$

Simple expression of Bellman's principle

$$V^*(x_k) = \min_{u_k}(Q^*(x_k, u_k)) \qquad h*(x_k) = \arg\min_{u_k}(Q^*(x_k, u_k))$$

Q Function ADP – Action Dependent ADP

Q function for any given control policy $h(x_k)$ satisfies the recursion

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

Recursive solution

Pick stabilizing initial control policy

Find Q function

$$Q_{j+1}(x_k, u_k) = r(x_k, u_k) + \gamma Q_j(x_{k+1}, h_j(x_{k+1}))$$

Update control

$$h_{j+1}(x_k) = \arg\min_{u_k}(Q_{j+1}(x_k, u_k))$$

Now $f(x_k, u_k)$ not needed

Bradtke & Barto (1994) proved convergence for LQR

Q Learning does not need to know $f(x_k)$ or $g(x_k)$

For LQR $$V(x) = W^T \varphi(x) = x^T P x$$

V is quadratic in x

$$Q_h(x_k, u_k) = r(x_k, u_k) + V_h(x_{k+1})$$
$$= x_k^T Q x_k + u_k^T R u_k + (Ax_k + Bu_k)^T P(Ax_k + Bu_k)$$

$$= \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} Q + A^T PA & A^T PB \\ B^T PA & R + B^T PB \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} \equiv \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T H \begin{bmatrix} x_k \\ u_k \end{bmatrix} = \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} H_{xx} & H_{xu} \\ H_{ux} & H_{uu} \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix}$$

Q is quadratic in x and u

Control update is found by $$0 = \frac{\partial Q}{\partial u_k} = 2[B^T PAx_k + (R + B^T PB)u_k] = 2[H_{ux}x_k + H_{uu}u_k]$$

so $$u_k = -(R + B^T PB)^{-1} B^T PAx_k = -H_{uu}^{-1} H_{ux} x_k = L_{j+1} x_k$$

Control found only from Q function
A and B not needed

Implementation- DT Q Function Policy Iteration

For LQR

Q function update for control $u_k = L_j x_k$ is given by

$$Q_{j+1}(x_k, u_k) = r(x_k, u_k) + \gamma Q_{j+1}(x_{k+1}, L_j x_{k+1})$$

Assume measurements of u_k, x_k and x_{k+1} are available to compute u_{k+1}

QFA – Q Fn. Approximation

$$Q(x,u) = W^T \varphi(x,u)$$ Now u is an input to the NN- Werbos- Action dependent NN

Then

regression matrix

$$W_{j+1}^T \left[\varphi(x_k, u_k) - \gamma \varphi(x_{k+1}, L_j x_{k+1}) \right] = r(x_k, L_j x_k)$$

Since x_{k+1} is measured, do not need knowledge of f(x) or g(x) for value fn. update

Solve for weights using RLS or backprop.

For LQR case

$$\varphi(x) = \left[x_1^2, \ldots, x_1 x_n, x_2^2, \ldots, x_2 x_n, \ldots, x_n^2 \right]'.$$

Q Learning does not need to know $f(x_k)$ or $g(x_k)$

For LQR $$V(x) = W^T \varphi(x) = x^T P x$$

V is quadratic in x

$$Q_h(x_k, u_k) = r(x_k, u_k) + V_h(x_{k+1})$$
$$= x_k^T Q x_k + u_k^T R u_k + (Ax_k + Bu_k)^T P(Ax_k + Bu_k)$$

$$= \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} Q + A^T PA & A^T PB \\ B^T PA & R + B^T PB \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} \equiv \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T H \begin{bmatrix} x_k \\ u_k \end{bmatrix} = \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} H_{xx} & H_{xu} \\ H_{ux} & H_{uu} \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix}$$

Q is quadratic in x and u

Control update is found by $$0 = \frac{\partial Q}{\partial u_k} = 2[B^T PAx_k + (R + B^T PB)u_k] = 2[H_{ux}x_k + H_{uu}u_k]$$

so $$u_k = -(R + B^T PB)^{-1} B^T PAx_k = -H_{uu}^{-1} H_{ux} x_k = L_{j+1} x_k$$

Control found only from Q function
A and B not needed

Model-free policy iteration

Q Policy Iteration

$$Q_{j+1}(x_k,u_k) = r(x_k,u_k) + \gamma Q_{j+1}(x_{k+1}, L_j x_{k+1})$$

Bradtke, Ydstie, Barto

$$W_{j+1}^T\left[\varphi(x_k,u_k) - \gamma\varphi(x_{k+1}, L_j x_{k+1})\right] = r(x_k, L_j x_k)$$

Control policy update

Stable initial control needed

$$h_{j+1}(x_k) = \arg\min_{u_k}(Q_{j+1}(x_k,u_k))$$

$$u_k = -H_{uu}^{-1}H_{ux}x_k = L_{j+1}x_k$$

Greedy Q Fn. Update - Approximate Dynamic Programming

ADP Method 3. Q Learning

Action-Dependent Heuristic Dynamic Programming (ADHDP)

Paul Werbos

Greedy Q Update

Model-free ADP

$$Q_{j+1}(x_k,u_k) = r(x_k,u_k) + \gamma Q_j(x_{k+1}, h_j(x_{k+1}))$$

$$W_{j+1}^T\varphi(x_k,u_k) = r(x_k, L_j x_k) + W_j^T\gamma\varphi(x_{k+1}, L_j x_{k+1}) \equiv \text{target}_{j+1}$$

Update weights by RLS or backprop.

Q learning actually solves the Riccati Equation
WITHOUT knowing the plant dynamics

Model-free ADP

Direct OPTIMAL ADAPTIVE CONTROL

Works for Nonlinear Systems

Proofs?
Robustness?
Comparison with adaptive control methods?

Discrete-Time Zero-Sum Games

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

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

with quadratic cost

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

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

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

- The goal is to find the optimal strategies (State-feedback)

$$u^*(x) = Lx \qquad w^*(x) = Kx$$

Asma Al-Tamimi

DT Game
Heuristic Dynamic Programming: Forward-in-time Formulation

- An Approximate Dynamic Programming Scheme (ADP) where one has the following incremental optimization

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

which is equivalently written as

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

Game Algebraic Riccati Equation

- Using Bellman optimality principle “Dynamic Programming”

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

$$x_k^T P x_k = \min_{u_k} \max_{w_k} (r(x_k, u_k, w_k) + x_{k+1}^T P x_{k+1}).$$

- The Game Algebraic Riccati equation GARE

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

- The condition for saddle point are

$$I + B^T P B > 0$$

$$I - \gamma^{-2} E^T P E > 0$$

Game Algebraic Riccati Equation

The optimal policies for control and disturbance are

$$L = (I + B^T PB - B^T PE(E^T PE - \gamma^2 I)^{-1} E^T PB)^{-1} \times (B^T PE(E^T PE - \gamma^2 I)^{-1} E^T PA - B^T PA).$$

$$K = (E^T PE - \gamma^2 I - E^T PB(I + B^T PB)^{-1} B^T PE)^{-1} \times (E^T PB(I + B^T PB)^{-1} BPA - E^T PA).$$

Linear Quadratic case- V and Q are quadratic

Asma Al-Tamimi

$$V^*(x_k) = x_k^T P x_k$$

Q learning for H-infinity Control

$$Q^*(x_k, u_k, w_k) = r(x_k, u_k, w_k) + V^*(x_{k+1})$$
$$= \begin{bmatrix} x_k^T & u_k^T & w_k^T \end{bmatrix} H \begin{bmatrix} x_k^T & u_k^T & w_k^T \end{bmatrix}^T$$

Q function update

$$Q_{i+1}(x_k, \hat{u}_i(x_k), \hat{w}_i(x_k)) = x_k^T R x_k + \hat{u}_i(x_k)^T \hat{u}_i(x_k) - \gamma^2 \hat{w}_i(x_k)^T \hat{w}_i(x_k) +$$
$$Q_i(x_{k+1}, \hat{u}_i(x_{k+1}), \hat{w}_i(x_{k+1}))$$

$$[x_k^T \; u_k^T \; w_k^T] H_{i+1} [x_k^T \; u_k^T \; w_k^T]^T = x_k^T R x_k + u_k^T u_k - \gamma^2 w_k^T w_k + [x_{k+1}^T \; u_{k+1}^T \; w_{k+1}^T] H_i [x_{k+1}^T \; u_{k+1}^T \; w_{k+1}^T]^T$$

Control Action and Disturbance updates

$$u_i(x_k) = L_i x_k, \qquad w_i(x_k) = K_i x_k$$

$$\begin{bmatrix} H_{xx} & H_{xu} & H_{xw} \\ H_{ux} & H_{uu} & H_{uw} \\ H_{wx} & H_{wu} & H_{ww} \end{bmatrix}$$

$$L_i = (H_{uu}^i - H_{uw}^i H_{ww}^{i\,-1} H_{wu}^i)^{-1} (H_{uw}^i H_{ww}^{i\,-1} H_{wx}^i - H_{ux}^i),$$
$$K_i = (H_{ww}^i - H_{wu}^i H_{uu}^{i\,-1} H_{uw}^i)^{-1} (H_{wu}^i H_{uu}^{i\,-1} H_{ux}^i - H_{wx}^i).$$

A, B, E NOT needed ☺

Compare to Q function for H_2 Optimal Control Case

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

$$= x_k^T Q x_k + u_k^T R u_k + (Ax_k + Bu_k)^T P (Ax_k + Bu_k)$$

$$= \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} Q + A^T PA & A^T PB \\ B^T PA & R + B^T PB \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} \equiv \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T H \begin{bmatrix} x_k \\ u_k \end{bmatrix} = \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} H_{xx} & H_{xu} \\ H_{ux} & H_{uu} \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix}$$

H-infinity Game Q function

$$H_{i+1} = \begin{bmatrix} A^T P_i A + R & A^T P_i B & A^T P_i E \\ B^T P_i A & B^T P_i B + I & B^T P_i E \\ E^T P_i A & E^T P_i B & E^T P_i E - \gamma^2 I \end{bmatrix}.$$

Asma Al-Tamimi

Quadratic Basis set is used to allow on-line solution

$\hat{Q}(\bar{z}, h_i) = z^T H_i z = h_i^T \bar{z}$ where $z = \begin{bmatrix} x^T & u^T & w^T \end{bmatrix}^T$ and $\bar{z} = (z_1^2, \ldots, z_1 z_q, z_2^2, z_2 z_3, \ldots, z_{q-1} z_q, z_q^2)$

Quadratic Kronecker basis

Q function update

$$Q_{i+1}(x_k, \hat{u}_i(x_k), \hat{w}_i(x_k)) = x_k^T R x_k + \hat{u}_i(x_k)^T \hat{u}_i(x_k) - \gamma^2 \hat{w}_i(x_k)^T \hat{w}_i(x_k) + Q_i(x_{k+1}, \hat{u}_i(x_{k+1}), \hat{w}_i(x_{k+1}))$$

Solve for 'NN weights' - the elements of kernel matrix H

$$h_{i+1}^T \bar{z}(x_k) = x_k^T R x_k + \hat{u}_i(x_k)^T \hat{u}_i(x_k) - \gamma^2 \hat{w}_i(x_k)^T \hat{w}_i(x_k) + h_i^T \bar{z}(x_{k+1})$$

Use batch LS or online RLS

Control and Disturbance Updates

$$\hat{u}_i(x) = L_i x \qquad \hat{w}_i(x) = K_i x$$

Probing Noise injected to get Persistence of Excitation

$$\hat{u}_{ei}(x_k) = L_i x_k + n_{1k} \qquad \hat{w}_{ei}(x_k) = K_i x_k + n_{2k}$$

Proof- Still converges to exact result

Lemma 1 Iterating on equations (20), and (34) is equivalent to

$$H_{i+1} = G + \begin{bmatrix} A & B & E \\ L_i A & L_i B & L_i E \\ K_i A & K_i B & K_i E \end{bmatrix}^T H_i \begin{bmatrix} A & B & E \\ L_i A & L_i B & L_i E \\ K_i A & K_i B & K_i E \end{bmatrix} \quad (35)$$

Lemma 2 The matrices H_{i+1}, L_{i+1} and K_{i+1} can be written

$$H_{i+1} = \begin{bmatrix} A^T P_i A + R & A^T P_i B & A^T P_i E \\ B^T P_i A & B^T P_i B + I & B^T P_i E \\ E^T P_i A & E^T P_i B & E^T P_i E - \gamma^2 I \end{bmatrix}. \quad (36)$$

$$L_{i+1} = (I + B^T P_i B - B^T P_i E(E^T P_i E - \gamma^2 I)^{-1} E^T P_i B)^{-1} \times (B^T P_i E(E^T P_i E - \gamma^2 I)^{-1} E^T P_i A - B^T P_i A), \quad (37)$$

$$K_{i+1} = (E^T P_i E - \gamma^2 I - E^T P_i B(I + B^T P_i B)^{-1} B^T P_i E)^{-1} \times (E^T P_i B(I + B^T P_i B)^{-1} B^T P_i A - E^T P_i A). \quad (38)$$

where P_i is given as

$$P_i = \begin{bmatrix} I & L_i^T & K_i^T \end{bmatrix} H_i \begin{bmatrix} I & L_i^T & K_i^T \end{bmatrix}^T. \quad (39)$$

Lemma 3: Iterating on H_i is similar to iterating on P_i as

$$P_{i+1} = A^T P_i A + R - \begin{bmatrix} A^T P_i B & A^T P_i E \end{bmatrix} \begin{bmatrix} I + B^T P_i B & B^T P_i E \\ E^T P_i A & E^T P_i E - \gamma^2 I \end{bmatrix}^{-1} \begin{bmatrix} B^T P_i A \\ E^T P_i A \end{bmatrix} \quad (40)$$

with P_i defined as in (39).

Theorem 1: Assume that the linear quadratic zero-sum game is solvable and has a value under the state feedback information structure. Then, iterating on equation(35) in Lemma 1, with $H_0 = 0$, $L_0 = 0$ and $K_0 = 0$ converges with $H_i \to H$. where H is corresponds to $Q^*(x_k, u_k, w_k)$ as in (10) and (12) with corresponding P solving the GARE (5).

ADHDP Application for Power system

- System Description

$$x(t) = [\Delta f(t) \quad \Delta P_g(t) \quad \Delta X_g(t) \quad \Delta F(t)]^T$$

$$A = \begin{bmatrix} -1/T_p & K_p/T_p & 0 & 0 \\ 0 & -1/T_T & 1/T_T & 0 \\ -1/RT_G & 0 & -1/T_G & -1/T_G \\ K_E & 0 & 0 & 0 \end{bmatrix}$$

$$B^T = \begin{bmatrix} 0 & 0 & 1/T_G & 0 \end{bmatrix}$$

$$E^T = \begin{bmatrix} 1 - K_p/T_p & 0 & 0 & 0 \end{bmatrix}$$

$$1/T_p \in [0.033, 0.1]$$

$$K_p/T_p \in [4, 12]$$

$$1/T_T \in [2.564, 4.762]$$

$$1/T_G \in [9.615, 17.857]$$

$$1/RT_G \in [3.081, 10.639]$$

- The Discrete-time Model is obtained by applying ZOH to the CT

ADHDP Application for Power system

- The system state

  Δf _incremental frequency deviation (Hz)

  ΔP_g _incremental change in generator output (p.u. MW)

  ΔX_g _incremental change in governor position (p.u. MW)

  ΔF _incremental change in integral control.

  ΔP_d _is the load disturbance (p.u. MW); and

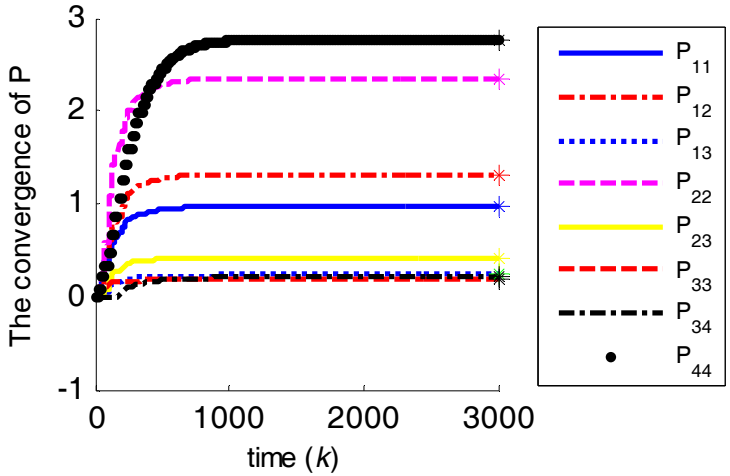
- The system parameters are:

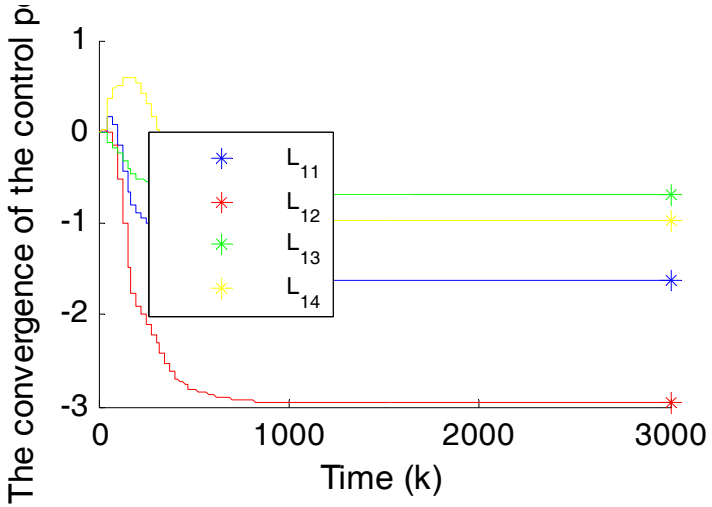
  T_G _the governor time constant

- T_T _turbine time constant
- T_P _plant model time constant
- K_p _planet model gain
- R _speed regulation due to governor action
- K_E_ integral control gain.

ADHDP Application for Power system

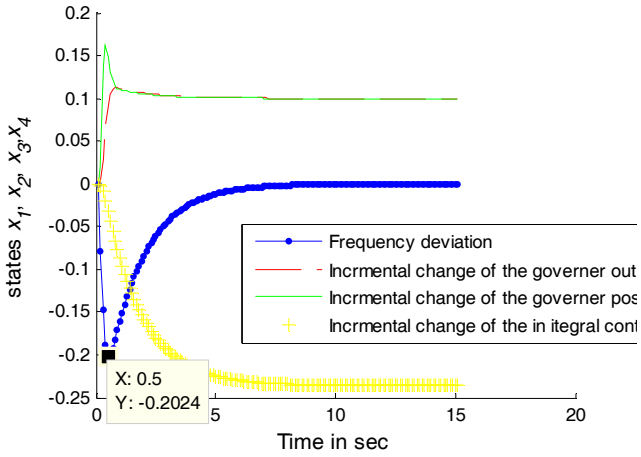
- ADHDP policy tuning

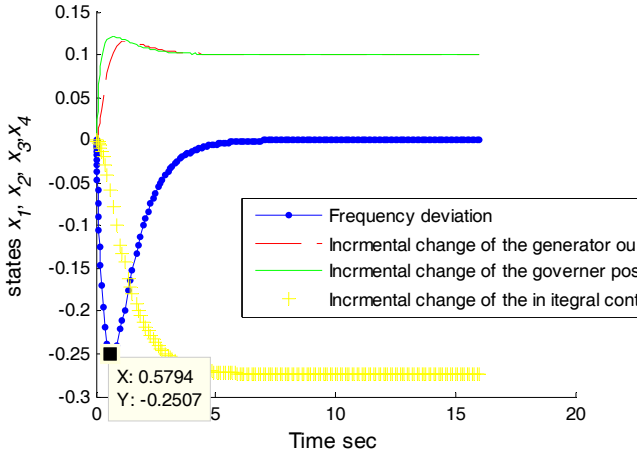




ADHDP Application for Power system

- *Comparison*





The ADHDP controller design

The design from [1]

- The maximum frequency deviation when using the ADHDP controller is improved by 19.3% from the controller designed in [1]

- [1] Wang, Y., R. Zhou, C. Wen, “Robust load-frequency controller design for power systems”, IEE Proc.-C, Vol. 140, No. I , 1993

Discrete-time nonlinear HJB solution using Approximate dynamic programming : Convergence Proof

- Problem Formulation

$$x_{k+1} = f(x_k) + g(x_k)u_k \qquad V^*(x_k) = \min_{u_k} \sum_{i=k}^{\infty} x_i Q x_i + u_i R u_i$$

- requires solving the DT HJB

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

$$= \min_{u_k} \left[x_k^T Q x_k + u_k^T R u_k + V^* \left(f(x_k) + g(x_k)u_k \right) \right]$$

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

Asma Al-Tamimi

Discrete-time Nonlinear Adaptive Dynamic Programming:

System dynamics

$$x_{k+1} = f(x_k) + g(x_k)u(x_k)$$

$$V(x_k) = \sum_{i=k}^{\infty} x_i^T Q x_i + u_i^T R u_i$$

Value function recursion

$$\begin{aligned} V(x_k) &= x_k^T Q x_k + u_k^T R u_k + \sum_{i=k+1}^{\infty} x_i^T Q x_i + u_i^T R u_i \\ &= x_k^T Q x_k + u_k^T R u_k + V(x_{k+1}) \end{aligned}$$

HDP

$$u_i(x_k) = \arg\min_u (x_k^T Q x_k + u^T R u + V_i(x_{k+1}))$$

$$\begin{aligned} V_{i+1} &= \min_u (x_k^T Q x_k + u^T R u + V_i(x_{k+1})) \\ &= x_k^T Q x_k + u_i^T(x_k) R u_i(x_k) + V_i(f(x_k) + g(x_k)u_i(x_k)) \end{aligned}$$

Asma Al-Tamimi

Proof of convergence of DT nonlinear HDP

Lemma 1 Let μ_i be any arbitrary sequence of control policies, and u_i is the policies as in (10). Let V_i be as in (11) and Λ_i as

$$\Lambda_{i+1}(x_k) = x_k Q x_k + \mu_i^T R \mu_i + \Lambda_i(x_{k+1}). \quad (12)$$

If $V_0 = \Lambda_0 = 0$, then $V_i \le \Lambda_i \quad \forall i$.

Lemma 2 Let the sequence $\{V_i\}$ be defined as in (11). If the system is controllable, then there is an upper bound Y such that $0 \le V_i \le Y \quad \forall i$.

Theorem 1 Define the sequence $\{V_i\}$ as in (11), with $V_0 = 0$. Then $\{V_i\}$ is a nondecreasing sequence in which $V_{i+1}(x_k) \ge V_i(x_k) \ \forall i$, and converge to the value function of the DT HJB, *i.e.* $V_i \Rightarrow V^*$ as $i \Rightarrow \infty$.

Flavor of proofs

Proof: Let $V_0 = \Phi_0 = 0$ where V_i is updated as in (11) and, and Φ_i is updated as

$$\Phi_{i+1}(x_k) = (x_k Q x_k + u_{i+1}^T R u_{i+1} + \Phi_i(x_{k+1})) \quad (11)$$

with the policies u_i as in (10). We will first prove by induction that $\Phi_i(x_k) \le V_{i+1}(x_k)$. Note that

$$V_1(x_k) - \Phi_0(x_k) = x_k^T Q x_k \ge 0$$

$$V_1(x_k) \ge \Phi_0(x_k)$$

Assume that $V_i(x_k) \ge \Phi_{i-1}(x_k) \ \forall x_k$. Since

$$\Phi_i(x_k) = x_k Q x_k + u_i^T R u_i + \Phi_{i-1}(x_{k+1})$$

$$V_{i+1}(x_k) = x_k Q x_k + u_i^T R u_i + V_i(x_{k+1}),$$

then

$$V_{i+1}(x_k) - \Phi_i(x_k) = V_i(x_{k+1}) - \Phi_{i-1}(x_{k+1}) \ge 0,$$

and therefore

$$\Phi_i(x_k) \le V_{i+1}(x_k). \quad (12)$$

From Lemma 1 $V_i(x_k) \le \Phi_i(x_k)$ and therfore

$$V_i(x_k) \le \Phi_i(x_k) \le V_{i+1}(x_k)$$

$$V_i(x_k) \le V_{i+1}(x_k)$$

hence proving that $\{V_i\}$ is a nondecreasing sequence bounded from above as shown in Lemma 2. Hence $V_i \to V^*$ as $i \to \infty$. ■

Standard Neural Network VFA for On-Line Implementation

NN for Value - Critic

$$\hat{V}_i(x_k, W_{Vi}) = W_{Vi}^T \phi(x_k)$$

NN for control action

$$\hat{u}_i(x_k, W_{ui}) = W_{ui}^T \sigma(x_k)$$

(can use 2-layer NN)

HDP

$$V_{i+1} = \min_u (x_k^T Q x_k + u^T R u + V_i(x_{k+1}))$$
$$= x_k^T Q x_k + u_i^T(x_k) R u_i(x_k) + V_i(f(x_k) + g(x_k) u_i(x_k))$$

$$u_i(x_k) = \arg\min_u (x_k^T Q x_k + u^T R u + V_i(x_{k+1}))$$

Define target cost function

$$d(\phi(x_k), W_{Vi}^T) = x_k^T Q x_k + \hat{u}_i^T(x_k) R \hat{u}_i(x_k) + \hat{V}_i(x_{k+1})$$
$$= x_k^T Q x_k + \hat{u}_i^T(x_k) R \hat{u}_i(x_k) + W_{Vi}^T \phi(x_{k+1})$$

Explicit equation for cost – use LS for Critic NN update

$$W_{Vi+1} = \arg\min_{W_{Vi+1}} \{\int_\Omega | W_{Vi+1}^T \phi(x_k) - d(\phi(x_k), W_{Vi}^T)|^2 \, dx_k\} \Longrightarrow W_{Vi+1} = \left(\int_\Omega \phi(x_k)\phi(x_k)^T dx\right)^{-1} \int_\Omega \phi(x_k) d^T(\phi(x_k), W_{Vi}^T, W_{ui}^T) dx$$

Implicit equation for DT control- use gradient descent for action update

$$W_{ui} = \arg\min_\alpha \left(\begin{matrix} x_k^T Q x_k + \hat{u}^T(x_k, \alpha) R \hat{u}(x_k, \alpha) + \\ \hat{V}_i(f(x_k) + g(x_k)\hat{u}(x_k, \alpha)) \end{matrix} \right)\Big|_\Omega \Longrightarrow$$

$$W_{ui(j+1)} = W_{ui(j)} - \alpha \frac{\partial (x_k^T Q x_k + \hat{u}_{i(j)}^T R \hat{u}_{i(j)} + \hat{V}_i(x_{k+1}))}{\partial W_{ui(j)}}$$

$$W_{ui}^{j+1} = W_{ui}^j - \alpha \sigma(x_k)(2R\hat{u}_{i(j)} + g(x_k)^T \frac{\partial \phi(x_{k+1})}{\partial x_{k+1}} W_{Vi})^T$$

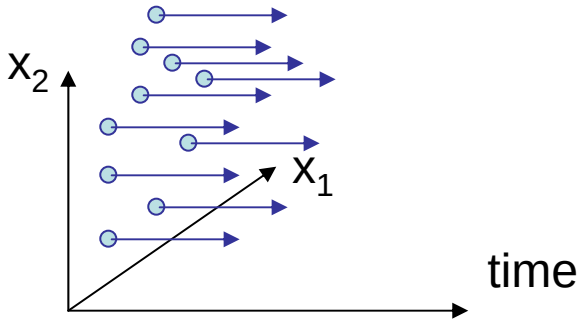
Backpropagation- P. Werbos

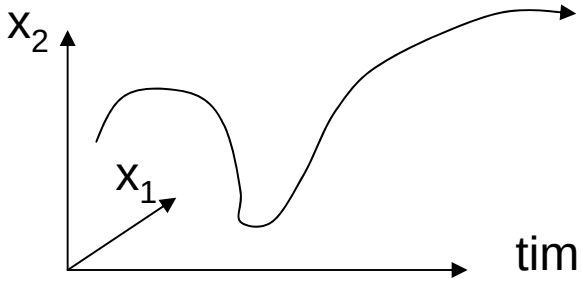
Issues with Nonlinear ADP

LS solution for Critic NN update

Selection of NN Training Set

$$W_{Vi+1} = \left(\int_{\Omega} \phi(x_k)\phi(x_k)^T dx \right)^{-1} \int_{\Omega} \phi(x_k) d^T(\phi(x_k), W_{Vi}^T, W_{ui}^T) dx$$





Integral over a region of state-space
Approximate using a set of points

Batch LS

Take sample points along a single trajectory

Recursive Least-Squares RLS

Set of points over a region vs. points along a trajectory

For Linear systems- these are the same

Conjecture- For Nonlinear systems
They are the same under a persistence of excitation condition
- Exploration

Interesting Fact for HDP for Nonlinear systems

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

must know system A and B matrices

NN for control action

$$\hat{u}_i(x_k, W_{ui}) = W_{ui}^T \sigma(x_k)$$

Implicit equation for DT control- use gradient descent for action update

$$W_{ui} = \arg\min_{\alpha} \left(\begin{array}{l} x_k^T Q x_k + \hat{u}^T(x_k, \alpha) R \hat{u}(x_k, \alpha) + \\ \hat{V}_i(f(x_k) + g(x_k)\hat{u}(x_k, \alpha)) \end{array} \right)\Bigg|_{\Omega} \Longrightarrow$$

$$W_{ui(j+1)} = W_{ui(j)} - \alpha \frac{\partial (x_k^T Q x_k + \hat{u}_{i(j)}^T R \hat{u}_{i(j)} + \hat{V}_i(x_{k+1}))}{\partial W_{ui(j)}}$$

$$W_{ui}^{j+1} = W_{ui}^{j} - \alpha \sigma(x_k)(2R\hat{u}_{i(j)} + g(x_k)^T \frac{\partial \phi(x_{k+1})}{\partial x_{k+1}} W_{Vi})^T$$

Note that state internal dynamics *f(x$_k$)* is NOT needed in nonlinear case since:

1. NN Approximation for action is used
2. *x$_{k+1}$* is measured

Discrete-time nonlinear HJB solution using Approximate dynamic programming : Convergence Proof

- Simulation Example 1
- The linear system – Aircraft longitudinal dynamics

$$A=\begin{bmatrix} 1.0722 & 0.0954 & 0 & -0.0541 & -0.0153 \\ 4.1534 & 1.1175 & 0 & -0.8000 & -0.1010 \\ 0.1359 & 0.0071 & 1.0 & 0.0039 & 0.0097 \\ 0 & 0 & 0 & 0.1353 & 0 \\ 0 & 0 & 0 & 0 & 0.1353 \end{bmatrix}$$

$$B=\begin{bmatrix} -0.0453 & -0.0175 \\ -1.0042 & -0.1131 \\ 0.0075 & 0.0134 \\ 0.8647 & 0 \\ 0 & 0.8647 \end{bmatrix}$$

Unstable, Two-input system

- The HJB, i.e. ARE, Solution

$$P=\begin{bmatrix} 55.8348 & 7.6670 & 16.0470 & -4.6754 & -0.7265 \\ 7.6670 & 2.3168 & 1.4987 & -0.8309 & -0.1215 \\ 16.0470 & 1.4987 & 25.3586 & -0.6709 & 0.0464 \\ -4.6754 & -0.8309 & -0.6709 & 1.5394 & 0.0782 \\ -0.7265 & -0.1215 & 0.0464 & 0.0782 & 1.0240 \end{bmatrix}$$

$$L=\begin{bmatrix} -4.1136 & -0.7170 & -0.3847 & 0.5277 & 0.0707 \\ -0.6315 & -0.1003 & 0.1236 & 0.0653 & 0.0798 \end{bmatrix}$$

Discrete-time nonlinear HJB solution using Approximate dynamic programming : Convergence Proof

- Simulation
- The Cost function approximation

$$\hat{V}_{i+1}(x_k, W_{Vi+1}) = W_{Vi+1}^T \phi(x_k)$$

$$\phi^T(x) = \begin{bmatrix} x_1^2 & x_1x_2 & x_1x_3 & x_1x_4 & x_1x_5 & x_2^2 & x_2x_3 & x_4x_2 & x_2x_5 & x_3^2 & x_3x_4 & x_3x_5 & x_4^2 & x_4x_5 & x_5^2 \end{bmatrix}$$

$$W_V^T = \begin{bmatrix} w_{V1} & w_{V2} & w_{V3} & w_{V4} & w_{V5} & w_{V6} & w_{V7} & w_{V8} & w_{V9} & w_{V10} & w_{V11} & w_{V12} & w_{V13} & w_{V14} & w_{V15} \end{bmatrix}$$

- The Policy approximation

$$\hat{u}_i = W_{ui}^T \sigma(x_k)$$

$$\sigma^T(x) = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{bmatrix}$$

$$W_u^T = \begin{bmatrix} w_{u11} & w_{u12} & w_{u13} & w_{u14} & w_{u15} \\ w_{u21} & w_{u22} & w_{u23} & w_{u24} & w_{u25} \end{bmatrix}$$

Discrete-time nonlinear HJB solution using Approximate dynamic programming : Convergence Proof

- Simulation

The convergence of the cost

$$W_V^T = [55.5411 \quad 15.2789 \quad 31.3032 \quad -9.3255 \quad -1.4536 \quad 2.3142 \quad 2.9234 \quad -1.6594 \quad -0.2430$$

$$24.8262 \quad -1.3076 \quad 0.0920 \quad 1.5388 \quad 0.1564 \quad 1.0240]$$

$$\begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} & P_{15} \\ P_{21} & P_{22} & P_{23} & P_{24} & P_{25} \\ P_{31} & P_{32} & P_{33} & P_{34} & P_{35} \\ P_{41} & P_{42} & P_{43} & P_{44} & P_{45} \\ P_{51} & P_{52} & P_{53} & P_{54} & P_{55} \end{bmatrix} = \begin{bmatrix} w_{V1} & 0.5w_{V2} & 0.5w_{V3} & 0.5w_{V4} & 0.5w_{V5} \\ 0.5w_{V2} & w_{V6} & 0.5w_{V7} & 0.5w_{V8} & 0.5w_{V9} \\ 0.5w_{V3} & 0.5w_{V7} & w_{V10} & 0.5w_{V11} & 0.5w_{V12} \\ 0.5w_{V4} & 0.5w_{V8} & 0.5w_{V11} & w_{V13} & 0.5w_{V14} \\ 0.5w_{V5} & 0.5w_{V9} & 0.5w_{V12} & 0.5w_{V14} & w_{V15} \end{bmatrix}$$

$$P = \begin{bmatrix} 55.8348 & 7.6670 & 16.0470 & -4.6754 & -0.7265 \\ 7.6670 & 2.3168 & 1.4987 & -0.8309 & -0.1215 \\ 16.0470 & 1.4987 & 25.3586 & -0.6709 & 0.0464 \\ -4.6754 & -0.8309 & -0.6709 & 1.5394 & 0.0782 \\ -0.7265 & -0.1215 & 0.0464 & 0.0782 & 1.0240 \end{bmatrix}$$

Discrete-time nonlinear HJB solution using Approximate dynamic programming : Convergence Proof

- Simulation

  The convergence of the control policy

$$W_u = \begin{bmatrix} 4.1068 & 0.7164 & 0.3756 & -0.5274 & -0.0707 \\ 0.6330 & 0.1005 & -0.1216 & -0.0653 & -0.0798 \end{bmatrix} \qquad L = \begin{bmatrix} -4.1136 & -0.7170 & -0.3847 & 0.5277 & 0.0707 \\ -0.6315 & -0.1003 & 0.1236 & 0.0653 & 0.0798 \end{bmatrix}$$

$$\begin{bmatrix} L_{11} & L_{12} & L_{13} & L_{14} & L_{15} \\ L_{21} & L_{22} & L_{23} & L_{24} & L_{25} \end{bmatrix} = -\begin{bmatrix} w_{u11} & w_{u12} & w_{u13} & w_{u14} & w_{u15} \\ w_{u21} & w_{u22} & w_{u23} & w_{u24} & w_{u25} \end{bmatrix}$$

Note- In this example, internal dynamics matrix A is NOT Needed.