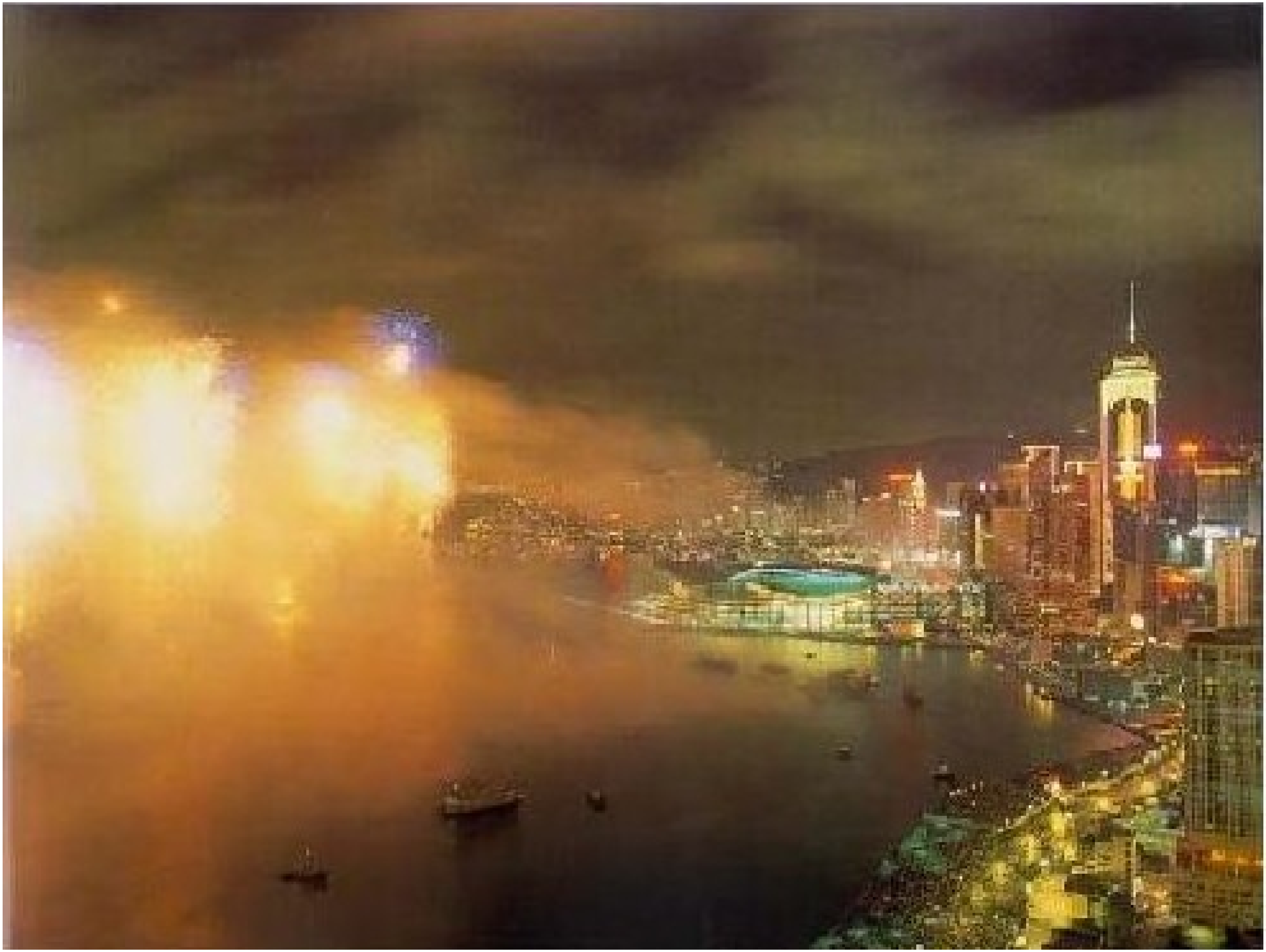


F.L. Lewis
Moncrief-O'Donnell Endowed Chair
Head, Controls & Sensors Group

Supported by :
NSF - PAUL WERBOS
ARO – JIM OVERHOLT

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

Approximate Dynamic Programming for Feedback Control

The University of Texas ARLINGTON

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

ARRI

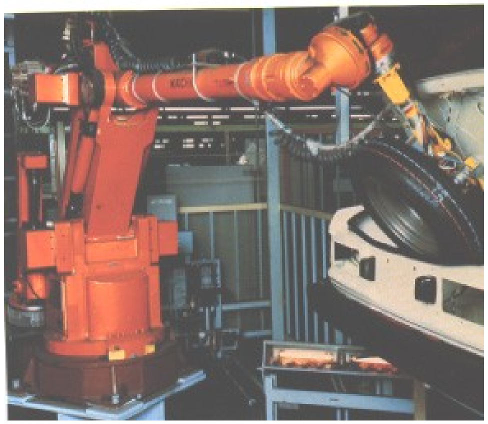
香港城市大學
City University of Hong Kong

The University of Texas
ARLINGTON

Automation & Robotics Research Institute (ARRI)

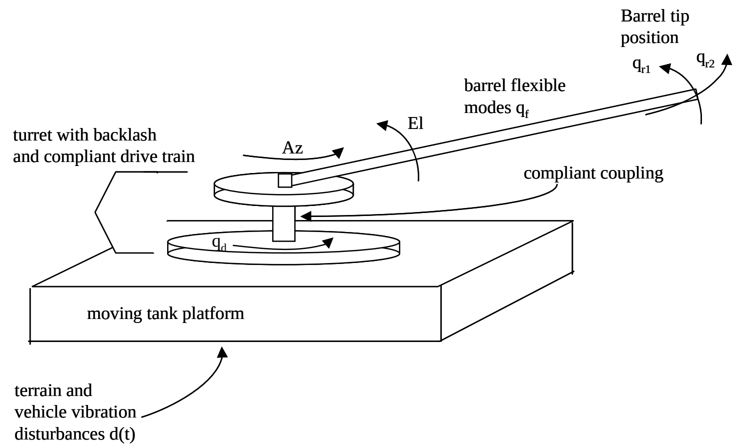
Relevance- Machine Feedback Control

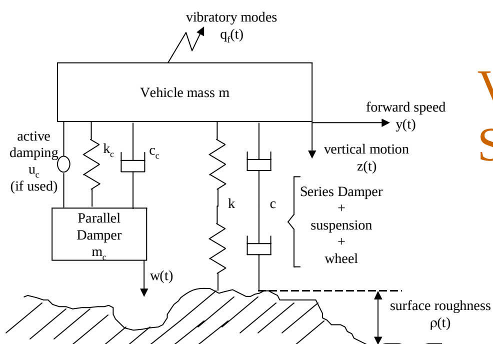
High-Speed Precision Motion Control with unmodeled dynamics, vibration suppression, disturbance rejection, friction compensation, deadzone/backlash control



Industrial Machines

Military Land Systems





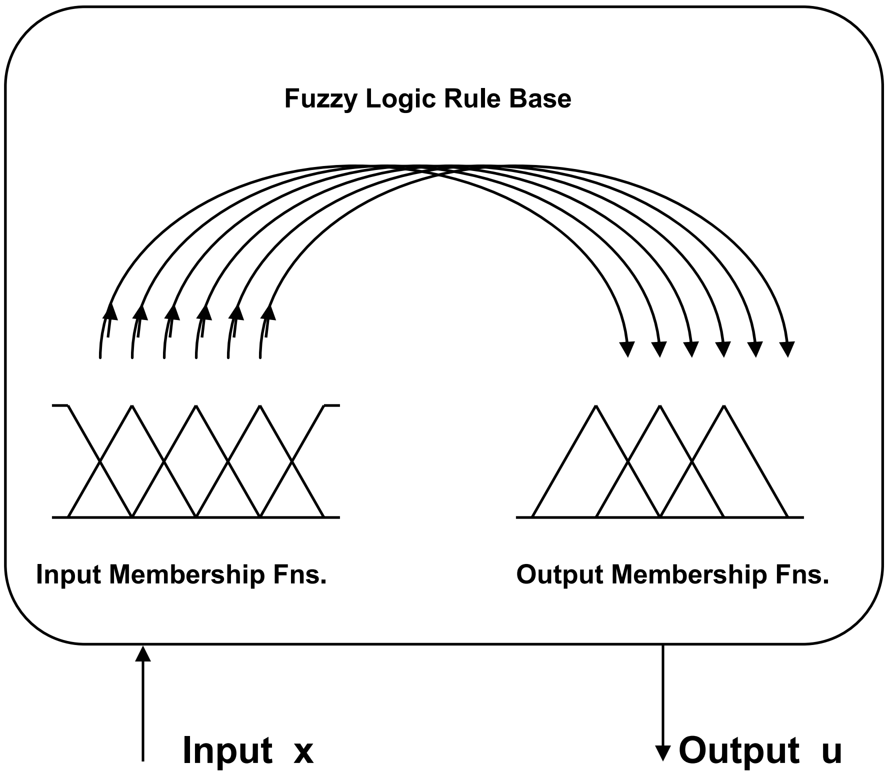
Single-Wheel/Terrain System with Nonlinearities

Vehicle Suspension

Aerospace

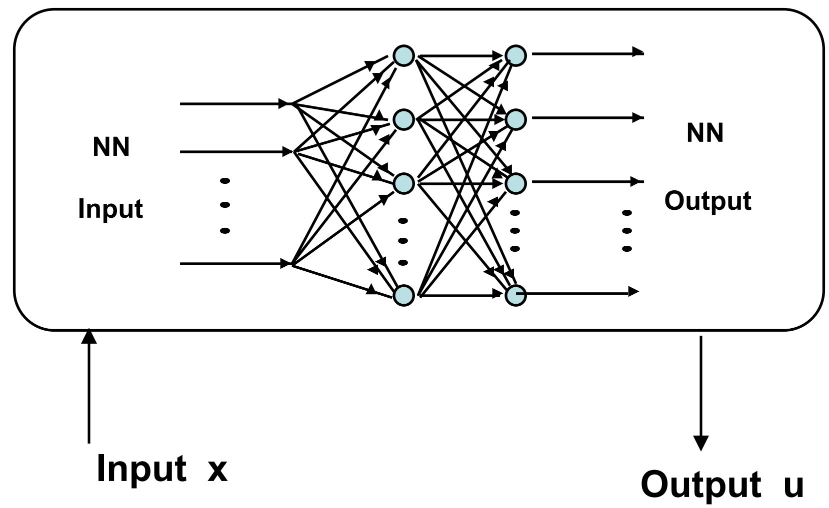
INTELLIGENT CONTROL TOOLS

Fuzzy Associative Memory (FAM)



Neural Network (NN)

(Includes Adaptive Control)



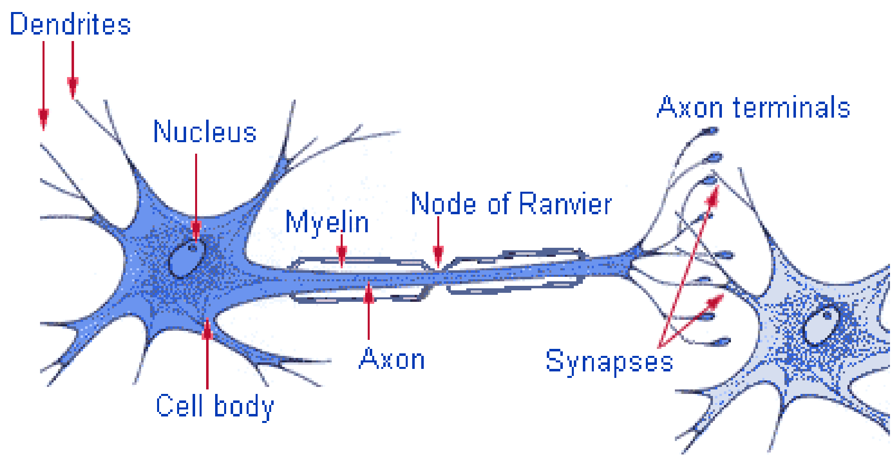
Both FAM and NN define a function u= f(x) from inputs to outputs

FAM and NN can both be used for: 1. Classification and Decision-Making
2. Control

NN Includes Adaptive Control (Adaptive control is a 1-layer NN)

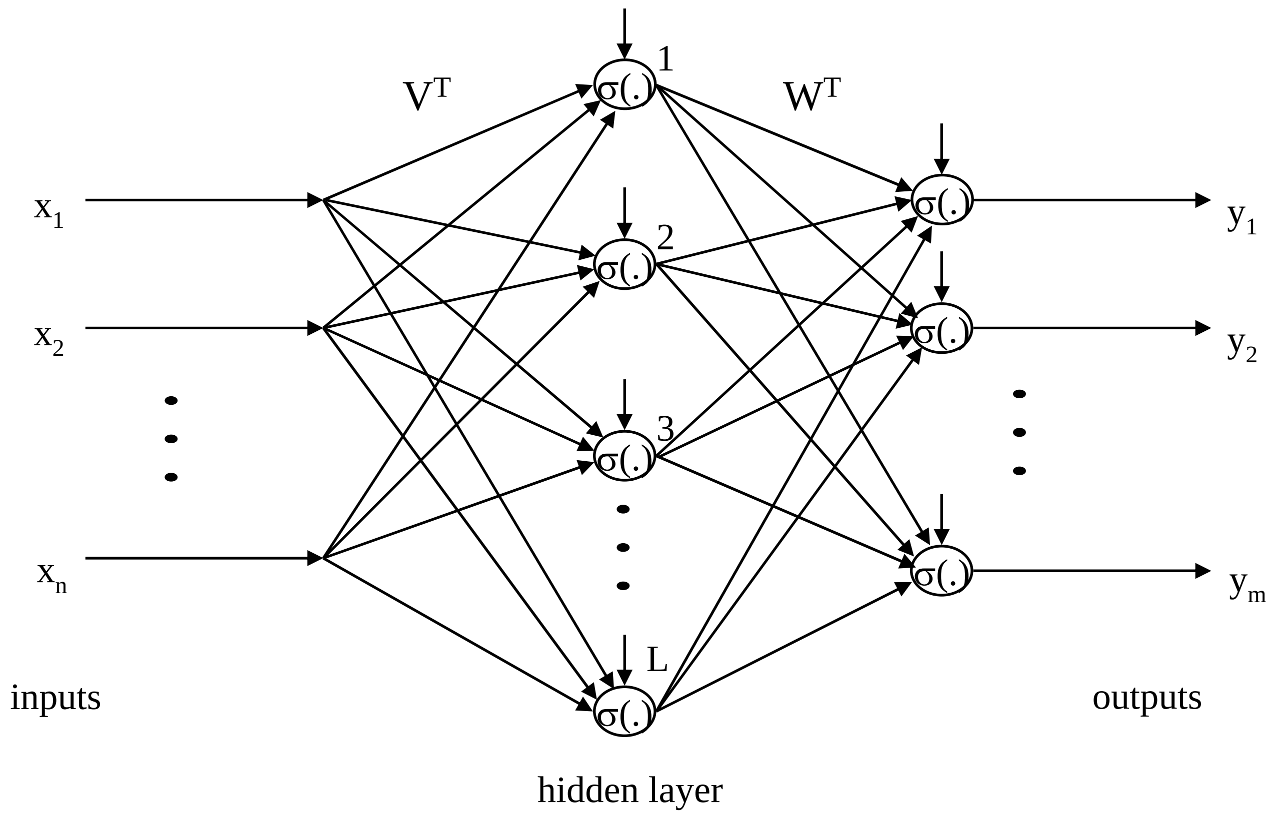
Neural Network Properties

- Learning
- Recall
- Function approximation
- Generalization
- Classification
- Association
- Pattern recognition
- Clustering
- Robustness to single node failure
- Repair and reconfiguration



Nervous system cell.
http://www.sirinet.net/~jgjohnso/index.html

Two-layer feedforward static neural network (NN)



Summation eqs

$$y_i = \sigma\left(\sum_{k=1}^{K} w_{ik}\sigma\left(\sum_{j=1}^{n} v_{kj}x_j + v_{k0}\right) + w_{i0}\right)$$

Matrix eqs

$$y = W^T\sigma(V^T x)$$

Have the universal approximation property

Overcome Barron's fundamental accuracy limitation of 1-layer NN

Dynamical System Models

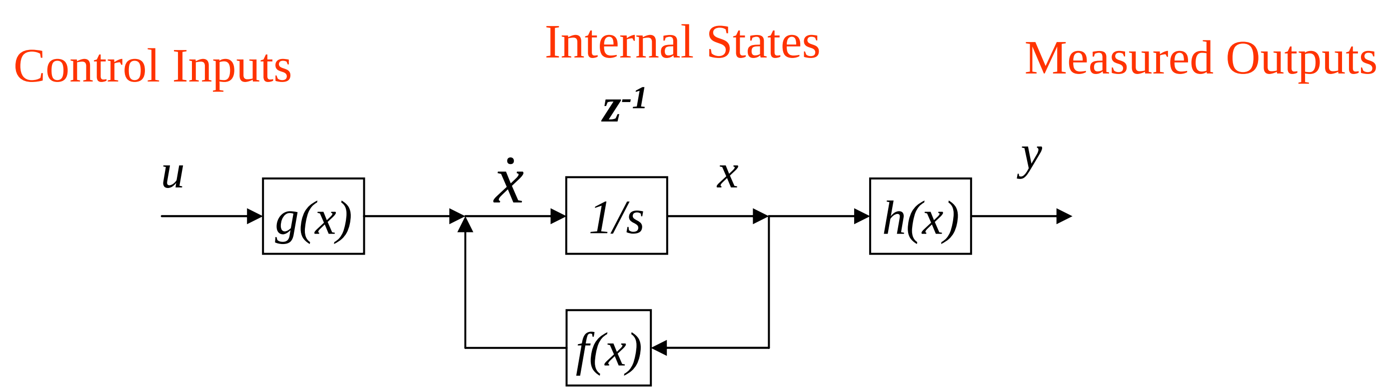
Continuous-Time Systems	Discrete-Time Systems

Nonlinear system

$$\dot{x} = f(x) + g(x)u \qquad x_{k+1} = f(x_k) + g(x_k)u_k$$
$$y = h(x) \qquad y_k = h(x_k)$$

Linear system

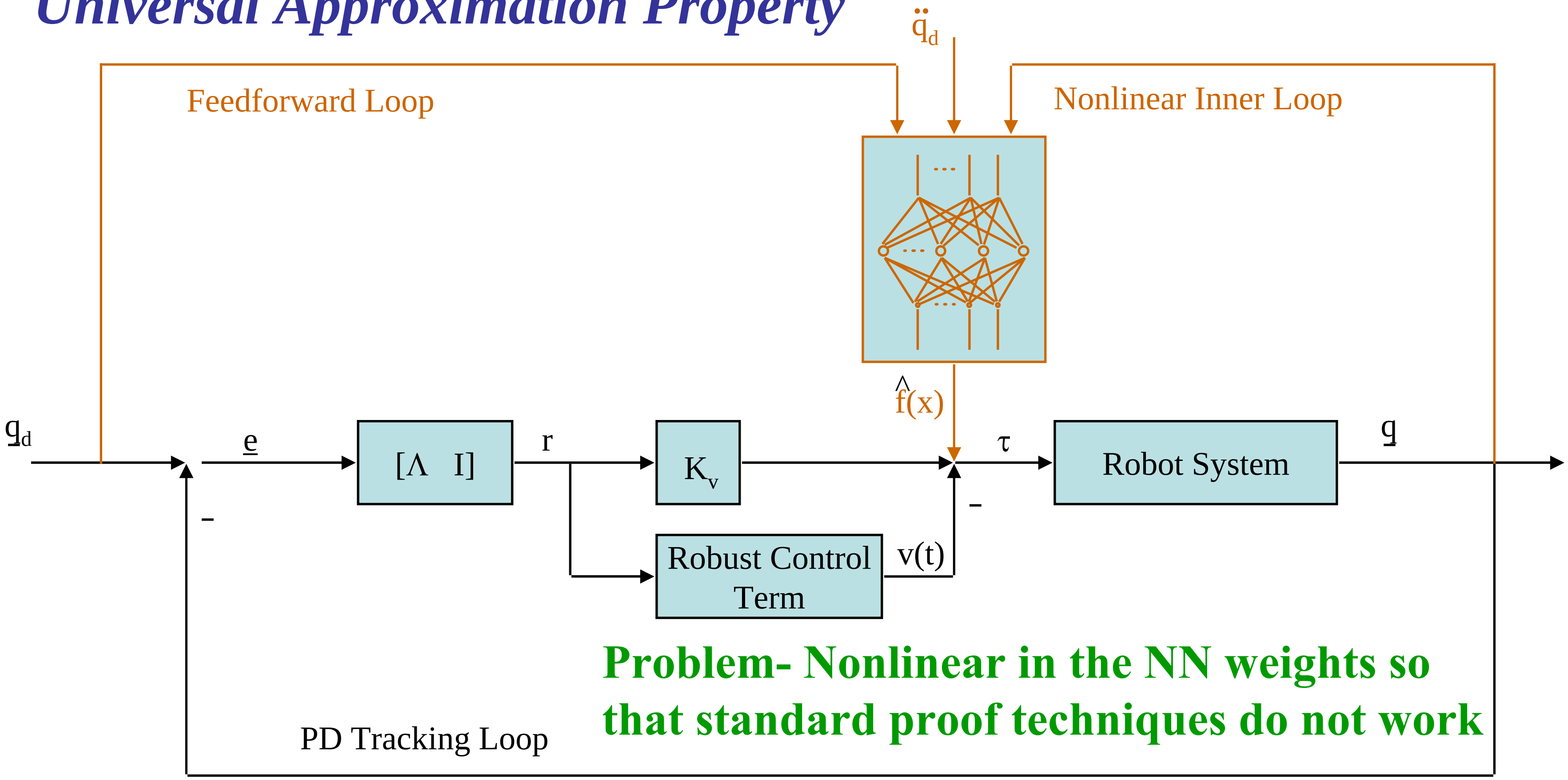
$$\dot{x} = Ax + Bu \qquad x_{k+1} = Ax_k + B_k$$
$$y = Cx \qquad y_k = Cx_k$$



Neural Network Robot Controller

Universal Approximation Property

Feedback linearization



Problem- Nonlinear in the NN weights so that standard proof techniques do not work

Easy to implement with a few more lines of code
Learning feature allows for on-line updates to NN memory as dynamics change
Handles unmodelled dynamics, disturbances, actuator problems such as friction
NN universal basis property means no regression matrix is needed
Nonlinear controller allows faster & more precise motion

Extension of Adaptive Control to nonlinear-in parameters systems

No regression matrix needed

Theorem 1 (NN Weight Tuning for Stability)

Let the desired trajectory $q_d(t)$ and its derivatives be bounded. Let the initial tracking error be within a certain allowable set U. Let Z_M be a known upper bound on the Frobenius norm of the unknown ideal weights Z.

Can also use simplified tuning- Hebbian
But tracking error is larger

Take the control input as

$$\tau = \hat{W}^T \sigma(\hat{V}^T x) + K_v r - v \qquad \text{with} \qquad v(t) = -K_Z(\|Z\|_F + Z_M)r .$$

Forward Prop term?

Let weight tuning be provided by

$$\dot{\hat{W}} = F\hat{\sigma}r^T - F\hat{\sigma}'\hat{V}^T x r^T - \kappa F\|r\|\hat{W} ,$$

$$\dot{\hat{V}} = Gx(\hat{\sigma}'^T \hat{W} r)^T - \kappa G\|r\|\hat{V}$$

with any constant matrices $F = F^T > 0, G = G^T > 0$, and scalar tuning parameter $\kappa > 0$. Initialize the weight estimates as $\hat{W} = 0, \hat{V} = random$.

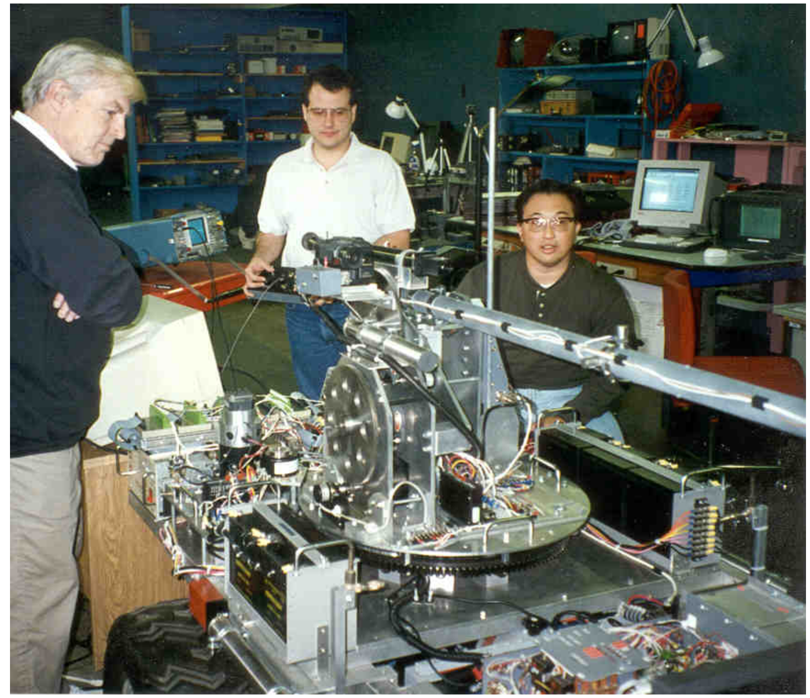
Then the filtered tracking error $r(t)$ and NN weight estimates $\hat{W}, \hat{V}$ are uniformly ultimately bounded. Moreover, arbitrarily small tracking error may be achieved by selecting large control gains K_v.

Backprop terms-
Werbos

Extra robustifying terms-
Narendra's e-mod extended to NLIP systems

More complex Systems?

Force Control



Flexible pointing systems

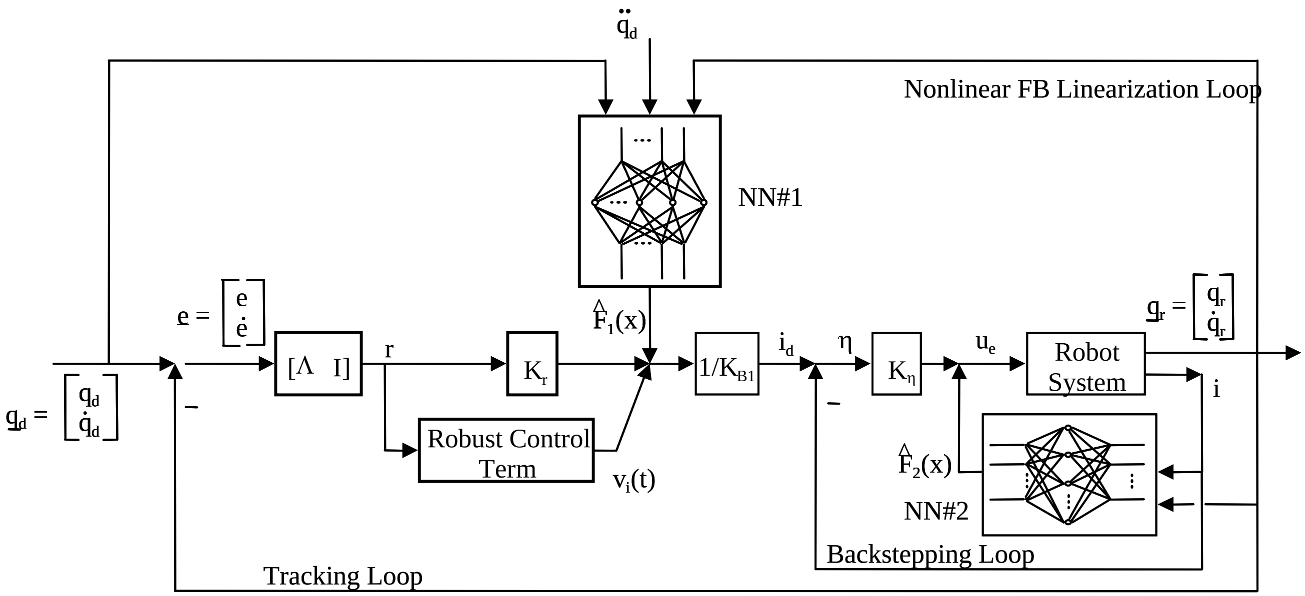
Vehicle active suspension

SBIR Contracts
Won 1996 SBA Tibbets Award
4 US Patents
NSF Tech Transfer to industry

Flexible & Vibratory Systems

Backstepping

Add an extra feedback loop
Two NN needed
Use passivity to show stability

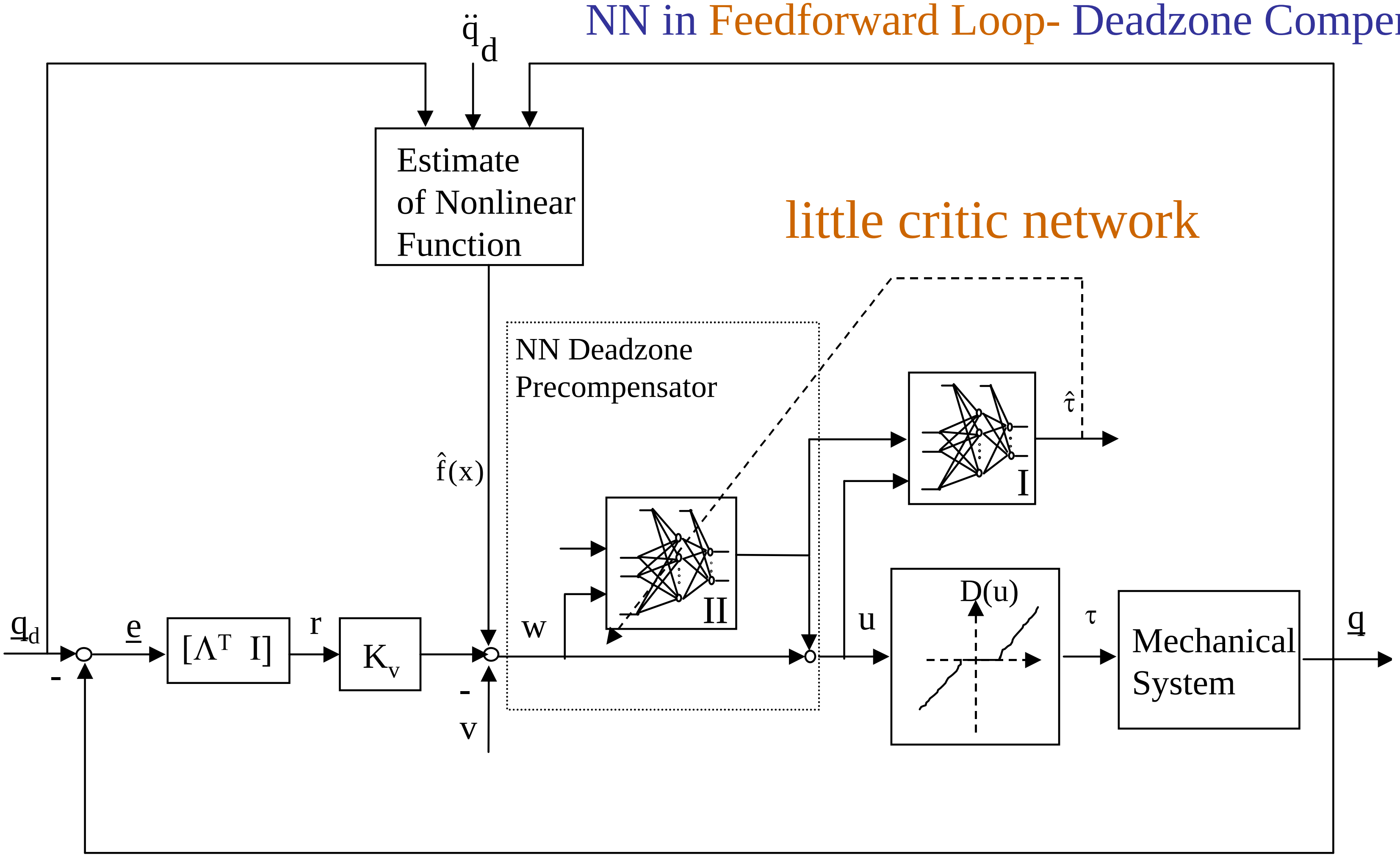


Neural network backstepping controller for Flexible-Joint robot arm

Advantages over traditional Backstepping- no regression functions needed

Actuator Nonlinearities - Deadzone, saturation, backlash

NN in Feedforward Loop- Deadzone Compensation



Critic: $\dot{\hat{W}}_i = T\sigma_i(U_i^T w) r^T \hat{W}^T \sigma'(U^T u) U^T - k_1 T \|r\| \hat{W}_i - k_2 T \|r\| \|\hat{W}_i\| \hat{W}_i$

Actor: $\dot{\hat{W}} = -S\sigma'(U^T u) U^T \hat{W}_i \sigma_i(U_i^T w) r^T - k_1 S \|r\| \hat{W}$

Acts like a 2-layer NN
With enhanced
backprop tuning !

NN Observers

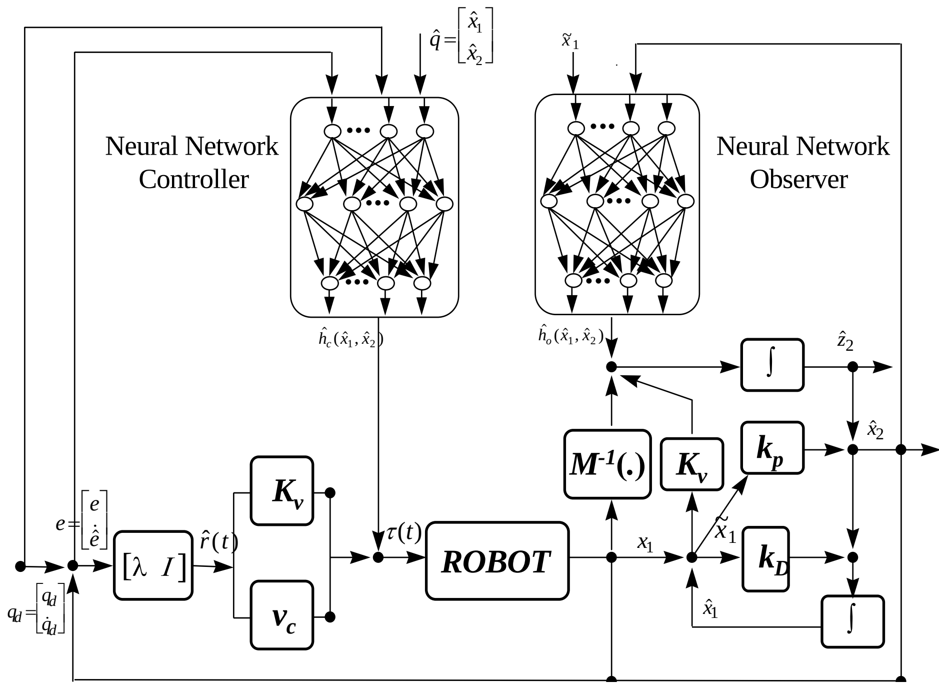
Needed when all states are not measured

i.e. Output feedback

Recurrent NN Observer

$$\dot{\hat{\mathbf{z}}}_1 = \hat{\mathbf{x}}_2 + k_D \tilde{\mathbf{x}}_1$$

$$\dot{\hat{\mathbf{z}}}_2 = \hat{\mathbf{W}}_o^T \sigma_o(\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2) + \mathbf{M}^{-1}(\mathbf{x}_1)\tau(t) + \mathbf{K}\tilde{\mathbf{x}}_1$$



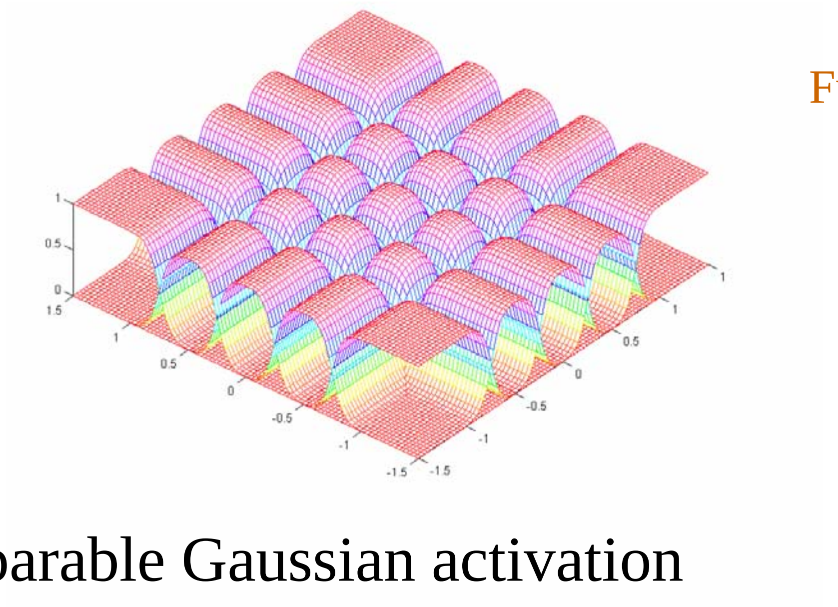
Tune NN observer -

$$\dot{\hat{W}}_o = -k_D F_o \sigma_o(\hat{x})\tilde{x}_1^T - \kappa_o F_o \|\tilde{x}_1\| \hat{W}_o - \kappa_o F_o \hat{W}_o$$

Tune Action NN -

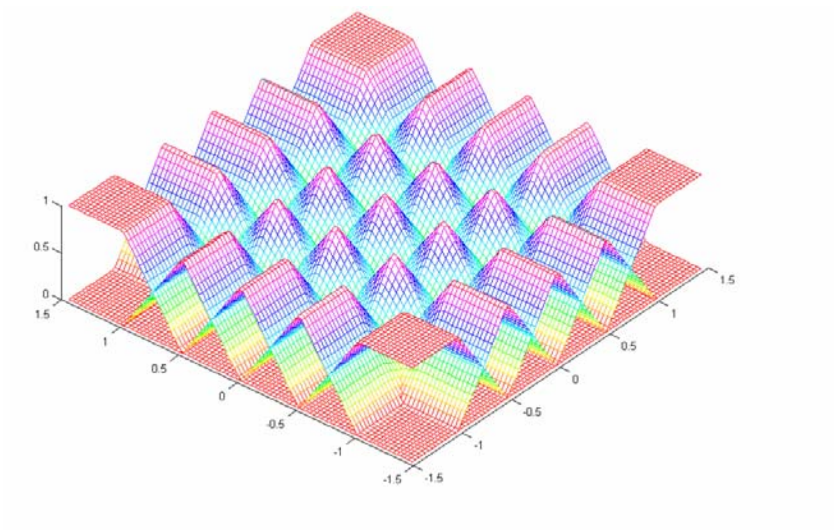
$$\dot{\hat{W}}_c = F_c \sigma_c(\hat{x}_1, \hat{x}_2)\hat{r}^T - \kappa_c F_c \|\hat{r}\| \hat{W}_c$$

Also Use CMAC NN, Fuzzy Logic systems



Separable Gaussian activation functions for RBF NN

Fuzzy Logic System
= NN with VECTOR thresholds



Tune first layer weights, e.g.
Centroids and spreads-
Activation fns move around

Dynamic Focusing of Awareness

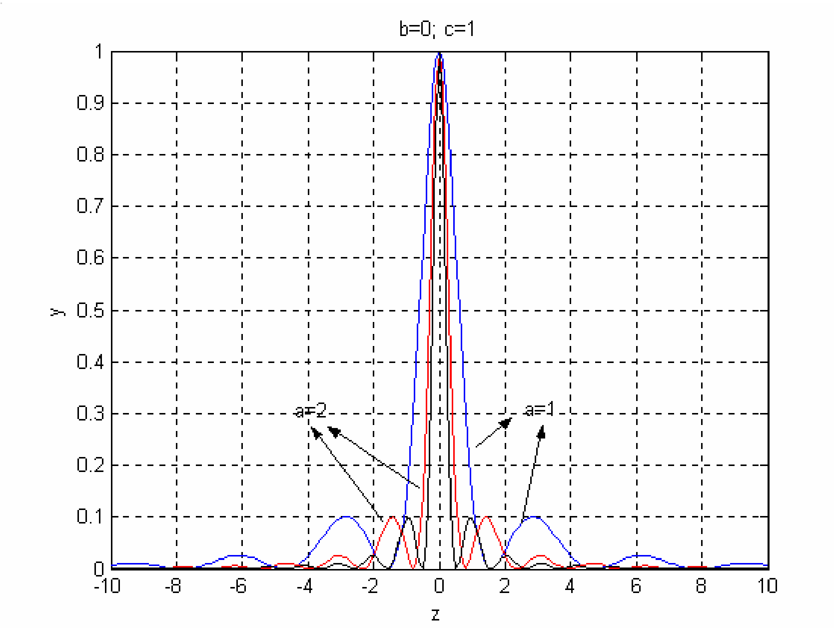
Separable triangular activation functions for CMAC NN

Elastic Fuzzy Logic- c.f. P. Werbos

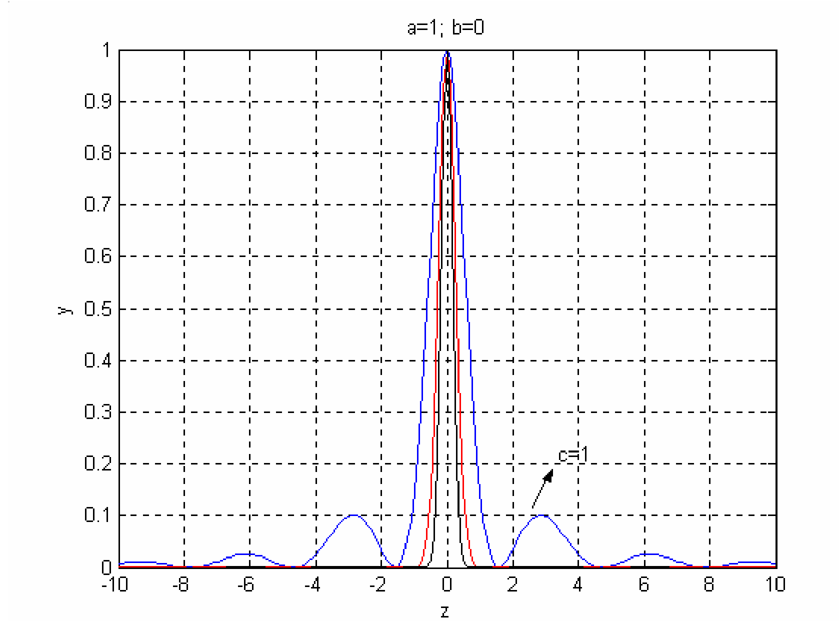
$$\phi(z,a,b,c) = \phi_B(z,a,b)^{c^2}$$

← Weights importance of factors in the rules

$$\phi(z,a,b,c) = \left[\frac{\cos^2(a(z-b))}{1+a^2(z-b)^2}\right]^{c^2}$$



Effect of change of membership function spread "a"



Effect of change of membership function elasticities "c"

Elastic Fuzzy Logic Control

Control

$$u(t) = -K_v r - \hat{g}(x, x_d)$$

Tune Control Rep. Values

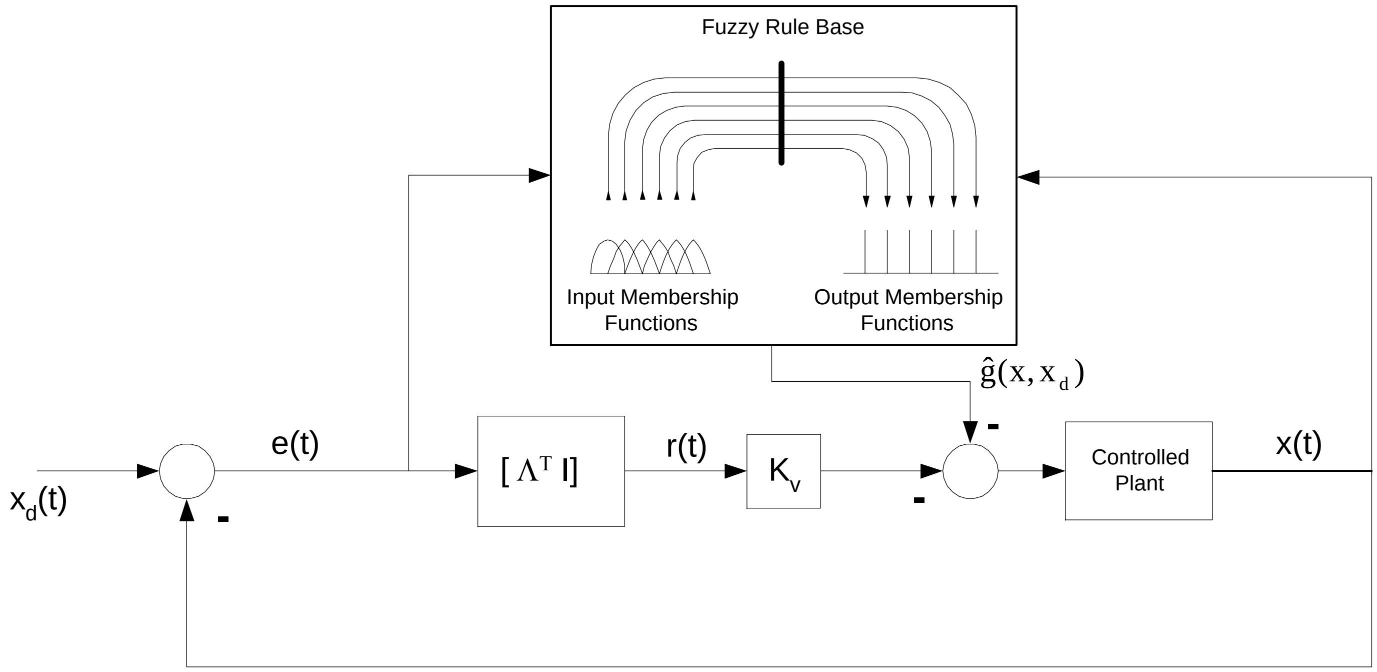
$$\dot{\hat{W}} = K_W(\hat{\Phi} - A\hat{a} - B\hat{b} - C\hat{c})r^T - k_W K_W \hat{W}\|r\|$$

Tune Membership Functions

$$\dot{\hat{a}} = K_a A^T \hat{W} r - k_a K_a \hat{a}\|r\|$$

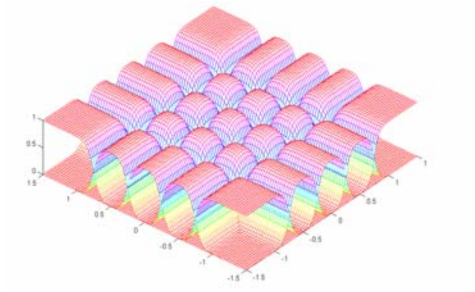
$$\dot{\hat{b}} = K_b B^T \hat{W} r - k_b K_b \hat{b}\|r\|$$

$$\dot{\hat{c}} = K_c C^T \hat{W} r - k_c K_c \hat{c}\|r\|$$

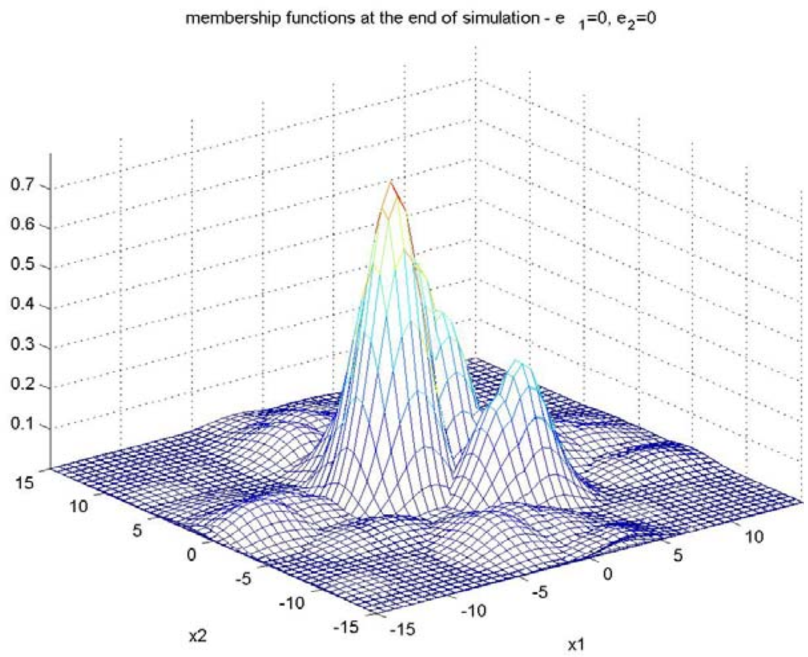


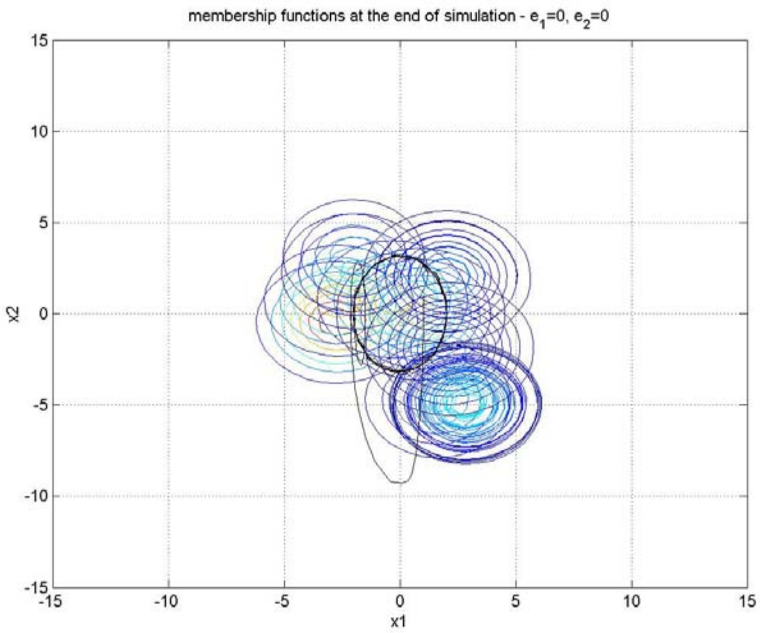
Better Performance

Start with 5x5 uniform grid of MFS



After tuning-



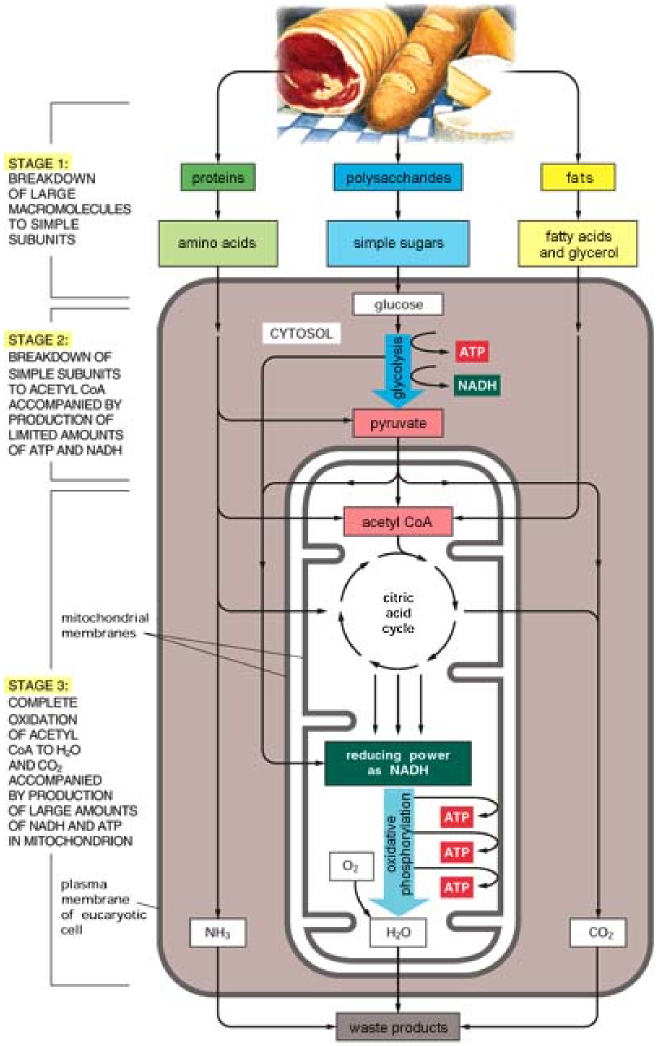


Builds its own basis set-
Dynamic Focusing of Awareness

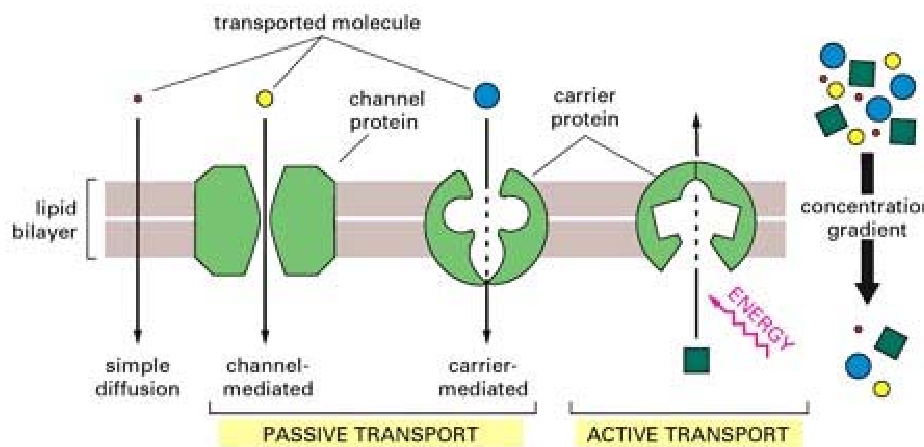
Optimality in Biological Systems

Cell Homeostasis

The individual cell is a complex feedback control system. It pumps ions across the cell membrane to maintain homeostatis, and has only limited energy to do so.



Cellular Metabolism



Permeability control of the cell membrane

http://www.accessexcellence.org/RC/VL/GG/index.html

R. Kalman 1960

Optimality in Control Systems Design

Rocket Orbit Injection

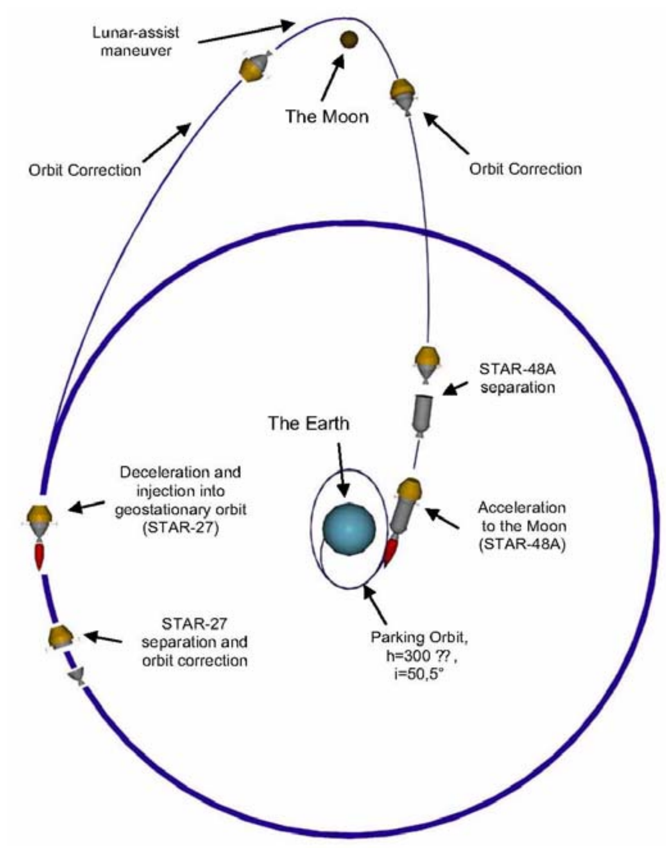


Fig. 1-1. Trajectory scheme

ISC Kosmotras Proprietary 9

http://microsat.sm.bmstu.ru/e-library/Launch/Dnepr_GEO.pdf

Dynamics

$$\dot{r} = w$$

$$\dot{w} = \frac{v^2}{r} - \frac{\mu}{r^2} + \frac{F}{m}\sin\phi$$

$$\dot{v} = \frac{-wv}{r} + \frac{F}{m}\cos\phi$$

$$\dot{m} = -Fm$$

Objectives

- Get to orbit in minimum time
- Use minimum fuel

2. Neural Network Solution of Optimal Design Equations

Nearly Optimal Control
Based on HJ Optimal Design Equations
Known system dynamics
Preliminary Off-line tuning

1. Neural Networks for Feedback Control

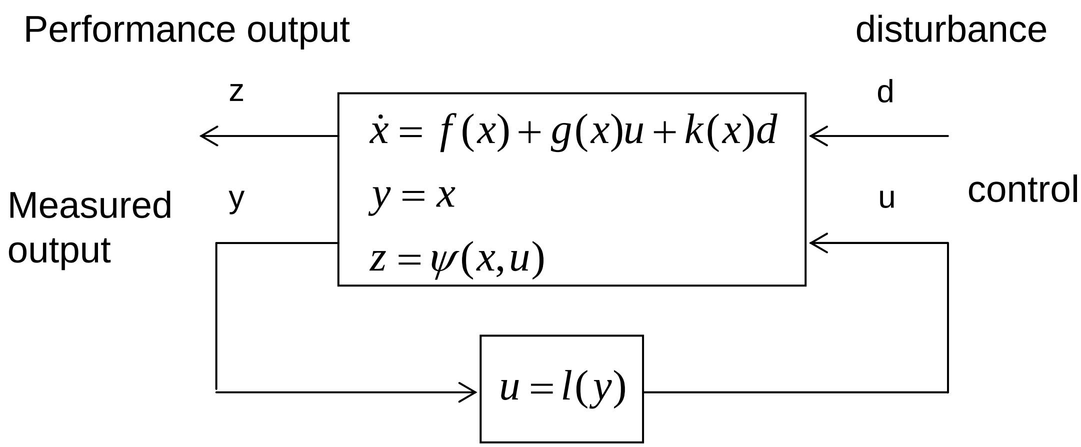
Based on FB Control Approach
Unknown system dynamics
On-line tuning

Extended adaptive control
to NLIP systems
No regression matrix

Murad Abu Khalaf

H-Infinity Control Using Neural Networks

System



where

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

L_2 Gain Problem

Find control *u(t)* so that

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

For all L_2 disturbances
And a prescribed gain γ^2

Zero-Sum differential Nash game

Standard Bounded L_2 Gain Problem

$$J(u,d) = \int_0^\infty \left(h^T h + \|u\|^2 - \gamma^2 \|d\|^2 \right) dt$$

Game theory value function

Take $\|u\|^2 = u^T R u$ and $\|d\|^2 = d^T d$

Hamilton-Jacobi Isaacs (HJI) equation

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

Stationary Point

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

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 γ^2

Problems to solve HJI

Beard proposed a successive solution method using Galerkin approx.

Viscosity Solution

Murad Abu Khalaf

Cannot solve HJI !!

Successive Solution- Algorithm 1:

Let γ *be prescribed and fixed.*

u_0 *a stabilizing control with region of asymptotic stability* Ω_0

1. *Outer loop- update control*

   Initial disturbance $d^0 = 0$

   2. *Inner loop- update disturbance*

      Solve Value Equation

Consistency equation For Value Function →

$$\frac{\partial (V^i{}_j)}{\partial x}^T \left(f + gu_j + kd\right) + h^T h + 2\int_0^{u_j} \phi^{-T}(v)dv - \gamma^2 (d^i)^T d^i = 0$$

Inner loop update disturbance

$$d^{i+1} = \frac{1}{2\gamma^2} k^T(x) \frac{\partial V^i{}_j}{\partial x}$$

go to 2.

Iterate i until convergence to $d^\infty, V^\infty{}_j$ *with RAS* $\Omega^\infty{}_j$

Outer loop update control action

$$u_{j+1} = -\tfrac{1}{2}\phi\left(g^T(x)\frac{\partial V^\infty{}_j}{\partial x}\right)$$

Go to 1.

Iterate j until convergence to $u_\infty, V^\infty{}_\infty$ *, with RAS* $\Omega^\infty{}_\infty$

CT Policy Iteration for H-Infinity Control

Murad Abu Khalaf

Problem- Cannot solve the Value Equation!

$$\frac{\partial (V^i{}_j)}{\partial x}^T \left(f + gu_j + kd\right) + h^T h + 2\int_0^{u_j} \phi^{-T}(\nu)d\nu - \gamma^2 (d^i)^T d^i = 0$$

Neural Network Approximation for Computational Technique

Neural Network to approximate $V^{(i)}(x)$

$$V_L^{(i)}(x) = \sum_{j=1}^{L} w_j^{(i)} \sigma_j(x) = W_L^{T(i)} \overline{\sigma}_L(x),$$

(Can use 2-layer NN!)

Value function gradient approximation is

$$\frac{\partial V_L{}^{(i)}}{\partial x} = \frac{\partial \overline{\sigma}_L(L)}{\partial x}^T W_L{}^{(i)} = \nabla \overline{\sigma}_L{}^T(x) W_L{}^{(i)}$$

Substitute into Value Equation to get

$$0 = w_j^{i\,T} \nabla\sigma(x)\dot{x} + r(x, u_j, d^i) = w_j^{i\,T} \nabla\sigma(x) f(x, u_j, d^i) + h^T h + \left\| u_j \right\|^2 - \gamma^2 \left\| d^i \right\|^2$$

Therefore, one may solve for NN weights at iteration *(i,j)*

VFA converts partial differential equation into algebraic equation in terms of NN weights

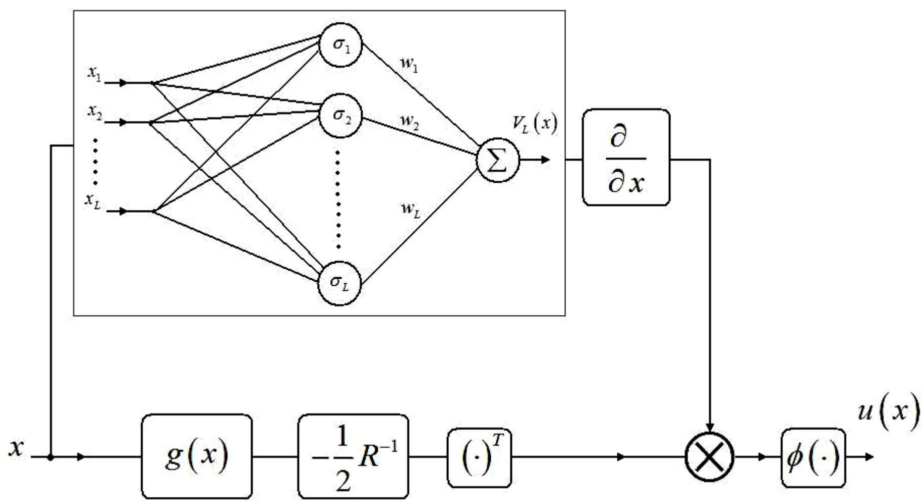
Neural Network Optimal Feedback Controller

Optimal Solution

$$d = \frac{1}{2} k^T(x) \nabla \bar{\sigma}_L{}^T W_L .$$

$$u = -\tfrac{1}{2} \phi\left(g^T(x) \nabla \bar{\sigma}_L{}^T W_L\right)$$

A NN feedback controller with nearly optimal weights



Cheng Tao

Fixed-Final-Time HJB Optimal Control

Optimal cost

$$-\frac{\partial V(x,t)^*}{\partial t} = \min_{u(t)}\left(L + \left(\frac{\partial V(x,t)^*}{\partial x}\right)^T \left(f(x) + g(x)u(x)\right)\right)$$

Optimal control

$$u^*(x) = -\frac{1}{2} R^{-1} g(x)^T \frac{\partial V(x,t)^*}{\partial x}$$

This yields the time-varying Hamilton-Jacobi-Bellman (HJB) equation

$$\frac{\partial V(x,t)^*}{\partial t} + \frac{\partial V(x,t)^*}{\partial x} f(x) + Q(x) - \frac{1}{4} \frac{\partial V(x,t)^{*T}}{\partial x} g(x) R^{-1} g(x)^T \frac{\partial V(x,t)^*}{\partial x} = 0$$

HJB Solution by NN Value Function Approximation

Cheng Tao

$$V_L(x,t)=\sum_{j=1}^{L} w_j(t)\sigma_j(x)=w_L^T(t)\sigma_L(x)$$

Time-varying weights

Irwin Sandberg

Note that

$$\frac{\partial V_L(x,t)}{\partial x}=\frac{\partial \boldsymbol{\sigma}_L^T(x)}{\partial x}\mathbf{w}_L(t)\equiv \nabla\boldsymbol{\sigma}_L^T(x)\mathbf{w}_L(t)$$

where $\nabla\boldsymbol{\sigma}_L(x)$ is the Jacobian $\partial\boldsymbol{\sigma}_L(x)/\partial x$

Policy iteration not needed!

$$\frac{\partial V_L(x,t)}{\partial t}=\dot{\mathbf{w}}_L^T(t)\boldsymbol{\sigma}_L(x)$$

Approximating $V(x,t)$ in the HJB equation gives an ODE in the NN weights

$$-\dot{\mathbf{w}}_L^T(t)\boldsymbol{\sigma}_L(x)-\mathbf{w}_L^T(t)\nabla\boldsymbol{\sigma}_L(x)f(x)$$
$$+\frac{1}{4}\mathbf{w}_L^T(t)\boldsymbol{\sigma}_L(x)g(x)R^{-1}g^T(x)\boldsymbol{\sigma}_L^T(x)\mathbf{w}_L(t)$$
$$-Q(x)=e_L(x)$$

Solve by least-squares – simply integrate backwards to find NN weights

Control is $u^*(x)=-\frac{1}{2}R^{-1}g(x)^T\nabla\sigma_L^T w_L(t)$

ARRI Research Roadmap in Neural Networks

3. Approximate Dynamic Programming – 2006-

Nearly Optimal Control
Based on recursive equation for the optimal value
Usually Known system dynamics (except Q learning)
The Goal – unknown dynamics
On-line tuning
Optimal Adaptive Control

> Extend adaptive control to yield OPTIMAL controllers.
> No canonical form needed.

2. Neural Network Solution of Optimal Design Equations – 2002-2006

Nearly Optimal Control
Based on HJ Optimal Design Equations
Known system dynamics
Preliminary Off-line tuning

> Nearly optimal solution of controls design equations.
> No canonical form needed.

1. Neural Networks for Feedback Control – 1995-2002

Based on FB Control Approach
Unknown system dynamics
On-line tuning
NN- FB lin., sing. pert., backstepping, force control, dynamic inversion, etc.

> Extended adaptive control to NLIP systems
> No regression matrix

Four ADP Methods proposed by 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 $\dfrac{\partial V}{\partial x}$

AD Dual heuristic programming

Gradients $\dfrac{\partial Q}{\partial x}, \quad \dfrac{\partial Q}{\partial u}$

Action NN to approximate the Control

Bertsekas- Neurodynamic Programming

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

Dynamical System Models

Continuous-Time Systems	Discrete-Time Systems

Nonlinear system

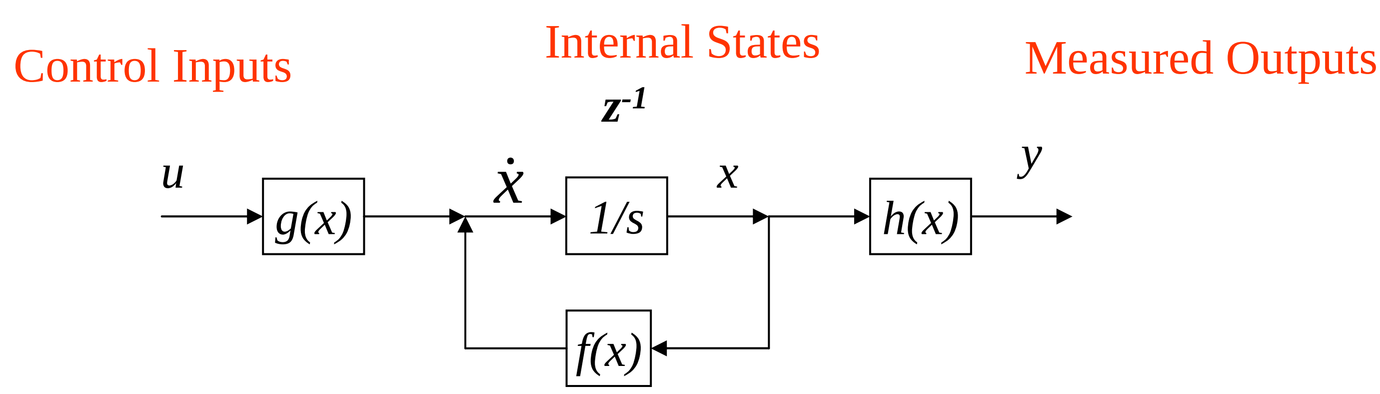
$$\dot{x} = f(x) + g(x)u$$
$$y = h(x)$$

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

Linear system

$$\dot{x} = Ax + Bu$$
$$y = Cx$$

$$x_{k+1} = Ax_k + B_k$$
$$y_k = Cx_k$$



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 input function

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

System dynamics does not appear

Solutions by Comp. Intelligence Community

Use System Dynamics

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

$$V(x_0) = \sum_{k=0}^{\infty} x_k Q x_k + u_k R u_k$$

DT HJB equation

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

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

Difficult to solve

Few practical solutions by Control Systems Community

DT Policy Iteration

Cost for any given control $h(x_k)$ 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

Find value

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

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

Update control

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

Howard (1960) proved convergence for MDP

DT Policy Iteration – Linear Systems

- For any stabilizing policy, the cost is

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

- DT Policy iterations

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

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

- Equivalent to an Underlying Problem- DT LQR:

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

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

DT Lyapunov eq.

Hewer proved convergence in 1971

Implementation- DT Policy Iteration

Value Function Approximation (VFA)

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

approximation error is neglected in the literature

weights

basis functions

LQR case- V(x) is quadratic

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

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

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

Quadratic basis functions

Use only the upper triangular basis set to get symmetric P
- Jie Huang 1995

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

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

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

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

Then update control using

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

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

Model-Based Policy Iteration

Robustness??

This gives $u_{k+1}(x_{k+1})$ – it is OK

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

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

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

Hewer 1971

Initial stabilizing control is needed

ADP Greedy Cost 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

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

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

Lancaster & Rodman
proved convergence

Initial stabilizing control is NOT needed

DT HDP vs. Receding Horizon Optimal Control

Forward-in-time HDP

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

$$P_0 = 0$$

Backward-in-time optimization – RHC

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

$P_N =$ Control Lyapunov Function

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

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

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

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

Q Function Definition

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

Define Q function

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

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

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

Bradtke & Barto (1994) proved convergence for LQR

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

$$Q_h(x_k, u_k) = r(x_k, u_k) + V_h(x_{k+1})$$
$$= x_k^T Q x_k + u_k^T R u_k + (Ax_k + Bu_k)^T P (Ax_k + Bu_k)$$
$$= \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} Q + A^T P A & A^T P B \\ B^T P A & R + B^T P B \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} \equiv \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T H \begin{bmatrix} x_k \\ u_k \end{bmatrix} = \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} H_{xx} & H_{xu} \\ H_{ux} & H_{uu} \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix}$$

Q is quadratic in x and u

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

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

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

$$h_{j+1}(x_k) = \arg\min_{u_k}(Q_{j+1}(x_k, u_k)) \qquad u_k = -H_{uu}^{-1} H_{ux} x_k = L_{j+1} x_k$$

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?

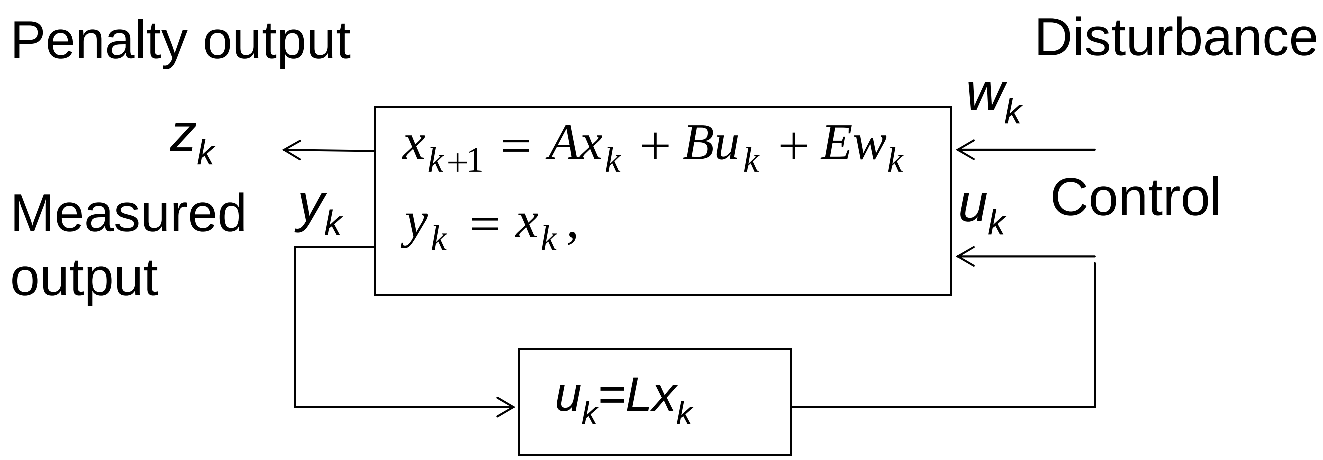
Asma Al-Tamimi

ADP for Discrete-Time H-infinity Control
Finding Nash Game Equilbrium

- HDP
- DHP
- AD HDP – Q learning
- AD DHP

Asma Al-Tamimi

ADP for DT H_∞ Optimal Control Systems



where $z_k^T z_k = x_k^T Q x_k + u_k^T u_k$

Find control u_k so that

$$\frac{\sum_{k=0}^{\infty} x_k^T Q x_k + u_k^T u_k}{\sum_{i=0}^{\infty} w_k^T w_k} \le \gamma^2$$

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

Asma Al-Tamimi

Two known ways for Discrete-time H-infinity iterative solution

Policy iteration for game solution

$$P_{i+1} - \mathbf{A}^T P_{i+1} \mathbf{A} = Q + L_i^T R L_i - \gamma^2 K_i^T K_i$$

$$\mathbf{A} = A + EK_j + BL_i$$

$$A_i = A + EK_j$$

$$A_j = A + BL_i$$

$$L_i = -(I + B^T P_i B)^{-1} B^T P_i A_i$$

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

Requires stable initial policy

ADP Greedy iteration

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

Both require full knowledge of system dynamics

Does not require a stable initial policy

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

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

Asma Al-Tamimi

HDP- Linear System Case

$$\hat{V}(x, p_i) = p_i^T \bar{x}$$

$$\bar{x} = (x_1^2, \ldots, x_1 x_n, x_2^2, x_2 x_3, \ldots, x_{n-1} x_n, x_n^2)$$

Value function update

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

Solve by batch LS or RLS

Control update $\hat{u}(x, L_i) = L_i^T x$ $\quad \hat{w}(x, K_i) = K_i^T x$

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

Control gain

A, B, E needed ☹

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

Disturbance gain

Showed that this is equivalent to iteration on the Underlying Game Riccati equation

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

Which is known to converge- Stoorvogel, Basar

Q-Learning for DT H-infinity Control: Action Dependent Heuristic Dynamic Programming

Asma Al-Tamimi

- Dynamic Programming: Backward-in-time

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

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

- Adaptive Dynamic Programming: Forward-in-time

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

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

$$= x_k^T R x_k + u_k^T u_k - \gamma^2 w_k^T w_k + V_i(A x_k + B u_k + E w_k)$$

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

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

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

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

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

A, B, E NOT needed ☺

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

Asma Al-Tamimi

H-inf Q learning Convergence Proofs

- Convergence – H-inf Q learning is equivalent to solving

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

without knowing the system matrices

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

Direct H-infinity Adaptive Control

Asma Al-Tamimi

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).

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

ADP for Nonlinear Systems: Convergence Proof

- HDP

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

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

HDP

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

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

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

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

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

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

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 $V_i(x_k) \ge \Phi_{i-1}(x_k) \quad \forall x_k$. 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

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

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

$$W_{ui} = \arg\min_\alpha \left. \begin{pmatrix} x_k^T Q x_k + \hat{u}^T(x_k, \alpha) R \hat{u}(x_k, \alpha) + \\ \hat{V}_i(f(x_k) + g(x_k) \hat{u}(x_k, \alpha)) \end{pmatrix} \right|_\Omega \Longrightarrow W_{ui(j+1)} = W_{ui(j)} - \alpha \frac{\partial (x_k^T Q x_k + \hat{u}_{i(j)}^T R \hat{u}_{i(j)} + \hat{V}_i(x_{k+1})}{\partial W_{ui(j)}}$$

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

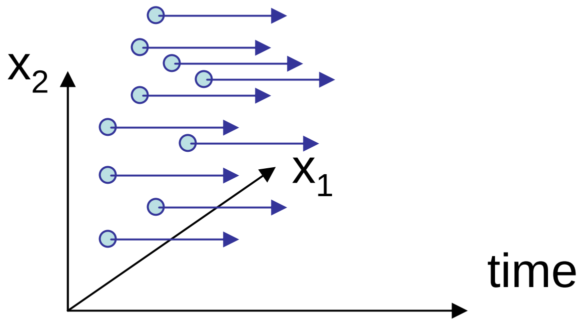
Backpropagation- P. Werbos

Issues with Nonlinear ADP

LS solution for Critic NN update

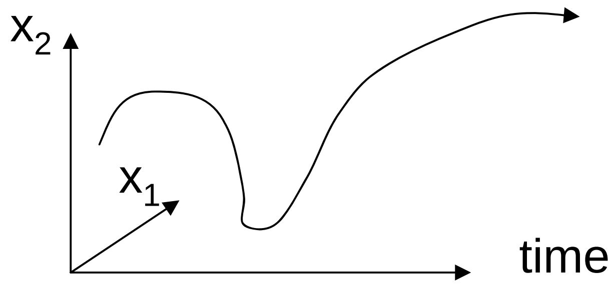
Selection of NN Training Set

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



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

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

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

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

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

Draguna Vrabie

ADP for Continuous-Time Systems

- Policy Iteration
- HDP

Continuous-Time Optimal Control

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

c.f. DT value recursion, where f(), g() do not appear

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

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

Hamiltonian

$$0 = \dot{V} + r(x,u) = \left(\frac{\partial V}{\partial x}\right)^T \dot{x} + r(x,u) = \left(\frac{\partial V}{\partial x}\right)^T f(x,u) + r(x,u) \equiv H(x, \frac{\partial V}{\partial x}, u) \qquad V(0) = 0$$

Optimal cost

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

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

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

Linear system, quadratic cost -

System: $\dot{x} = Ax + Bu$

Utility: $r(x,u) = x^T Qx + u^T Ru;\ R > 0, Q \geq 0$

The cost is quadratic $$V(x(t)) = \int_t^{\infty} r(x,u)d\tau = x^T(t)Px(t)$$

Optimal control (state feed-back):

$$u(t) = -R^{-1}B^T(x)Px(t) = -Lx(t)$$

HJB equation is the *algebraic Riccati equation* (ARE):

$$0 = PA + A^T P + Q - PBR^{-1}B^T P$$

CT Policy Iteration

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

Cost for any given u(t)

$$0 = \left(\frac{\partial V}{\partial x}\right)^T f(x,u) + r(x,u) \equiv H(x, \frac{\partial V}{\partial x}, u)$$

Lyapunov equation

Iterative solution

Pick stabilizing initial control

Find cost

$$0 = \left(\frac{\partial V_j}{\partial x}\right)^T f(x, h_j(x)) + r(x, h_j(x))$$

$$V_j(0) = 0$$

Update control

$$h_{j+1}(x) = -\tfrac{1}{2} R^{-1} g^T(x) \frac{\partial V_j}{\partial x}$$

- Convergence proved by Saridis 1979 if Lyapunov eq. solved exactly
- Beard & Saridis used complicated Galerkin Integrals to solve Lyapunov eq.
- Abu Khalaf & Lewis used NN to approx. V for nonlinear systems and proved convergence

Full system dynamics must be known

LQR Policy iteration = Kleinman algorithm

1. For a given control policy $u = -L_k x$ solve for the cost:

$$0 = A_k{}^T P_k + P_k A_k + C^T C + L_k{}^T R L_k$$ Lyapunov eq.

$$A_k = A - BL_k$$

2. Improve policy:

$$L_k = R^{-1} B^T P_{k-1}$$

- If started with a stabilizing control policy L_0 the matrix P_k monotonically converges to the unique positive definite solution of the Riccati equation.
- Every iteration step will return a stabilizing controller.
- The system has to be known.

Kleinman 1968

Policy Iteration Solution

Policy iteration

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

This is in fact a Newton's Method

$$Ric(P) \equiv A^T P + PA + P - PBB^T P$$

Then, Policy Iteration is

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

Frechet Derivative

$$Ric'_{P_i}(P) \equiv (A - BB^T P_i)^T P + P(A - BB^T P_i)$$

Synopsis on Policy Iteration and ADP

Discrete-time

Policy iteration

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

If x_{k+1} is measured,
do not need knowledge of
f(x) or g(x)

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

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

ADP Greedy cost update

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

Continuous-time

Policy iteration

$$0 = \left(\frac{\partial V_j}{\partial x}\right)^T \dot{x} + r(x, h_j(x)) = \left(\frac{\partial V_j}{\partial x}\right)^T [f(x) + g(x)h_j(x)] + r(x, h_j(x))$$

Either measure *dx/dt*
or must know *f(x), g(x)*

$$h_{j+1}(x) = -\tfrac{1}{2} R^{-1} g^T(x) \frac{\partial V_j}{\partial x}$$

Need to know ONLY
g(x) for control update

What is Greedy ADP for CT Systems ??

Draguna Vrabie

Policy Iterations without Lyapunov Equations

- An alternative to using policy iterations with Lyapunov equations is the following form of policy iterations:

$$V_j(x_0) = \int_0^\infty [Q(x) + W(u_j)]dt$$

Measure the cost

$$u_{j+1}(x) = -\phi\left(\tfrac{1}{2} R^{-1} g' \frac{dV_j}{dx} \right)$$

- Note that in this case, to solve for the Lyapunov function, you do not need to know the information about ***f*(*x*).**

Murray, Saeks, and Lendaris

Methods to obtain the solution

- Dynamic programming
  - built on Bellman's optimality principle – alternative form for CT Systems [Lewis & Syrmos 1995]

$$V^*(x(t)) = \min_{\substack{u(\tau) \\ t \le \tau \le t+\Delta t}} \left\{ \int_t^{t+\Delta t} r(x(\tau), u(\tau))d\tau \quad + \quad V^*(x(t+\Delta t)) \right\}$$

$$r(x(\tau), u(\tau)) = x^T(\tau)Qx(\tau) + u^T(\tau)Ru(\tau)$$

Draguna Vrabie

Solving for the cost – Our approach

For a given control $u = -Lx$

The cost satisfies
$$V(x(t)) = \int_t^{t+T} (x^T Qx + u^T Ru)dt \quad + \quad V(x(t+T))$$

c.f. DT case

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

f(x) and *g(x)* do not appear

LQR case
$$x(t)^T Px(t) = \int_t^{t+T} (x^T Qx + u^T Ru)dt + x(t+T)^T Px(t+T)$$

Optimal gain is $L = R^{-1}B^T P$

Policy Evaluation – *Critic update*

Let *K* be *any* state feedback gain for the system (1). One can measure the associated cost over the infinite time horizon

$$V(t,x(t)) = \int_t^{t+T} x(\tau)^T (Q + K^T R K) x(\tau) d\tau + \underbrace{W(t+T, x(t+T))}$$

where $W(t+T, x(t+T))$ is an initial infinite horizon cost to go.

What to do about the tail – issues in Receding Horizon Control

Draguna Vrabie

Now Greedy ADP can be defined for CT Systems

Solving for the cost – Our approach

CT ADP Greedy iteration

Control policy $u^k(t) = -L_k x(t)$

Cost update $$V_{k+1}(x(t_0)) = \int_{t_0}^{t_0+T} (x^T Q x + u^{k^T} R u^k) dt + V_k(x(t_0+T))$$

LQR $$x_0^T P_{k+1} x_0 = \int_{t_0}^{t_0+T} (x^T Q x + u^{k^T} R u^k) dt + x_1^T P_k x_1$$

A and B do not appear

Control gain update

$$L_{k+1} = R^{-1} B^T P_{k+1}$$

B needed for control update

Implement using quadratic basis set

$$\overline{p}_{i+1}^T \overline{x}(t) = \int_t^{t+T} x(\tau)^T (Q + P_i B R^{-1} B^T P_i) x(\tau) d\tau + \overline{p}_i^T \overline{x}(t+T)$$

u(t+T) in terms of x(t+T) - OK

- No initial stabilizing control needed

Direct Optimal Adaptive Control for Partially Unknown CT Systems

Algorithm Implementation

Measure cost increment by adding V as a state. Then $\dot{V} = x^T Q x + u^{k^T} R u^k$

The Critic update

$$x^T(t)P_{i+1}x(t) = \int_t^{t+T} x^T(\tau)(Q + K_i^T R K_i)x(\tau)d\tau + x^T(t+T)P_i x(t+T)$$

can be setup as

Quadratic basis set

$$\bar{p}_{i+1}^{\,T}\bar{x}(t) = \int_t^{t+T} x(\tau)^T(Q + K_i^T R K_i)x(\tau)d\tau + \bar{p}_i^{\,T}\bar{x}(t+T) \equiv d(\bar{x}(t), K_i)$$

Evaluating $d(\bar{x}(t), K_i)$ for $n(n+1)/2$ trajectory points, one can setup a least squares problem to solve

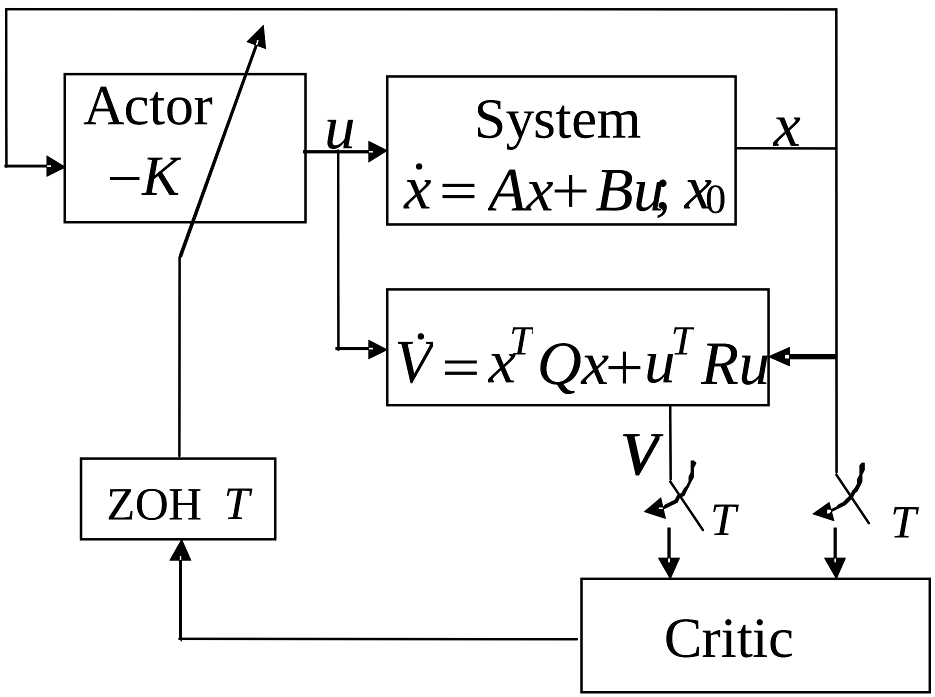
$$\bar{p}_{i+1} = (XX^T)^{-1}XY$$

$$X = [\bar{x}^1(t) \quad \bar{x}^2(t) \quad \ldots \quad \bar{x}^N(t)]$$

$$Y = [d(\bar{x}^1, K_i) \quad d(\bar{x}^2, K_i) \quad \ldots \quad d(\bar{x}^N, K_i)]^T$$

Or use recursive Least-Squares along the trajectory

Direct Optimal Adaptive Controller



A hybrid continuous/discrete dynamic controller
whose internal state is the observed value over the interval

Draguna Vrabie

Analysis of the algorithm

For a given control policy $u^k = -L_k x$ with $L_k = R^{-1} B^T P_k$

$$\dot{x} = Ax + Bu; \quad x(0)$$

$$x = e^{A_k t} x(0) \qquad\qquad A_k = A - BR^{-1} B^T P_k$$

Greedy update $V_{i+1}(x(t)) = \int_t^{t+T} \left\{ x^T Q x + u_i^T R u_i \right\} d\tau + V_i(x(t+T)), \quad V_0 = 0$ is equivalent to

$$P_{k+1} = \int_{t_0}^{t_0+T} e^{A_k^T t} (Q + L_k{}^T R L_k) e^{A_k t} dt + e^{A_k^T (T+t_0)} P_k e^{A_k (T+t_0)}$$

a strange pseudo-discretized RE

c.f. DT RE

$$P_{k+1} = \overline{A}^T P_k \overline{A} + Q - \overline{A}^T P_k B \left(P_k + B^T P_K B \right)^{-1} B^T P_k \overline{A}$$

$$P_{k+1} = \overline{A}_k^T P_k \overline{A}_k + Q + L_k^T \left(P_k + B^T P_K B \right) L_k$$

Analysis of the algorithm

Lemma 1. The ADP iteration between (13) and (14) is equivalent to the Quasi-Newton method

$$P_{i+1} = P_i - (Ric'_{P_i})^{-1}\left(Ric(P_i) - e^{A_i T^T} Ric(P_i) e^{A_i T} \right). \qquad (19)$$

This extra term means the initial
Control action need not be stabilizing

Lemma 2. CT HDP is equivalent to

$$P_{k+1} - P_k = \int_0^T e^{A_k^T t}(P_k A + AP_k + Q - L_k^T R L_k) e^{A_k t} dt \qquad A_k = A - BR^{-1}B^T P_k$$

When ADP converges, the resulting P satisfies the Continuous-Time ARE !!

Lemma 3. Let the ADP algorithm converge so that $P_i \to P^*$. Then P^* satisfies $Ric(P^*) = 0$, i.e. P^* is the solution the continuous-time ARE.

ADP solves the CT ARE without knowledge of the system dynamics f(x)

Solve the Riccati Equation
WITHOUT knowing the plant dynamics

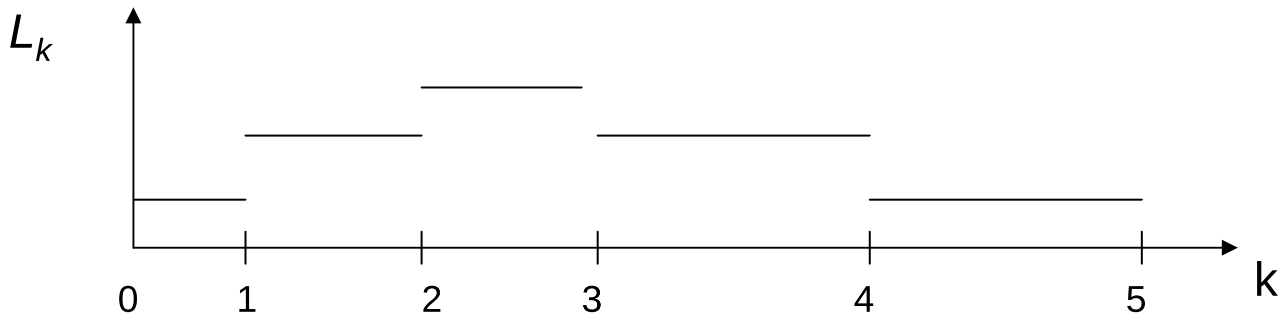
Model-free ADP

Direct OPTIMAL ADAPTIVE CONTROL

Works for Nonlinear Systems

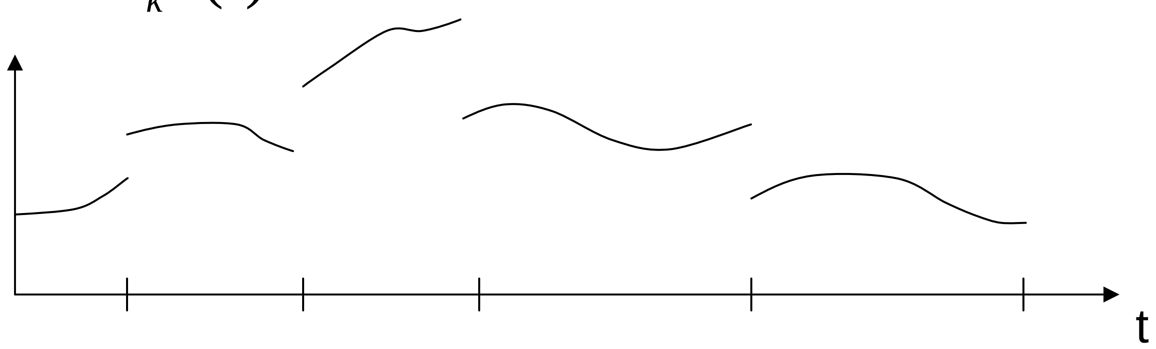
Proofs?
Robustness?
Comparison with adaptive control methods?

Gain update (Policy)



Control

$u^k(t) = -L_k x(t)$



Sample periods need not be the same

Continuous-time control with discrete gain updates

Neurobiology

Higher Central Control of Afferent Input

Descending tracts from the brain influence not only motor neurons but also the gamma-neurons which regulate sensitivity of the muscle spindle.

Central control of end-organ sensitivity has been demonstrated.

Many brain structures exert control of the first synapse in ascending systems.

Role of cerebello rubrospinal cortex and Purkinje Cells?

T.C. Rugh and H.D. Patton, *Physiology and Biophysics*, p. 213, 497,Saunders, London, 1966.

Cerebral Cortex operates at:

alpha waves 8-12 Hz – thalamus

activity of the visual cortex in an idle state

theta waves 4-8 Hz

Integration of sensory information with motor output

Muscular system operates at 200 Hz

Small Time-Step Approximate Tuning for Continuous-Time Adaptive Critics

$$H(x,\frac{\partial V}{\partial x},u)=\dot{V}(x)+r(x,u)\approx\frac{V_{t+1}-V_t}{\Delta t}+r(x,u)\approx\frac{V_{t+1}-V_t}{\Delta t}+\frac{r^D(x_t,u_t)}{\Delta t}$$

$$A_1^*(x_t,u_t)=\frac{r^D(x_t,u_t)+V(x_{t+1})-V^*(x_t)}{\Delta t}$$

Baird's Advantage function

Advantage learning is a sort of first-order approximation to our method

Results comparing the performances of DT-ADHDP and CT-HDP

Submitted to IJCNN'07 Conference

Asma Al-Tamimi and Draguna Vrabie

System, cost function, optimal solution

System – power plant

$$\dot{x} = Ax + Bu \quad x \in R^n, u \in R^m$$

$$A = \begin{bmatrix} -0.665 & 8 & 0 & 0 \\ 0 & -3.663 & 3.663 & 0 \\ -6.86 & 0 & -13.736 & -13.736 \\ 6 & 0 & 0 & 0 \end{bmatrix}$$

$$B^T = \begin{bmatrix} 0 & 0 & 13.736 & 0 \end{bmatrix}$$

Wang, Y., R. Zhou, C. Wen - 1993

Cost

$$V^*(x_0) = \min_{u(t)} \int_0^\infty (x^T Q x + u^T R u) d\tau$$

$$Q = I_n; R = I_m$$

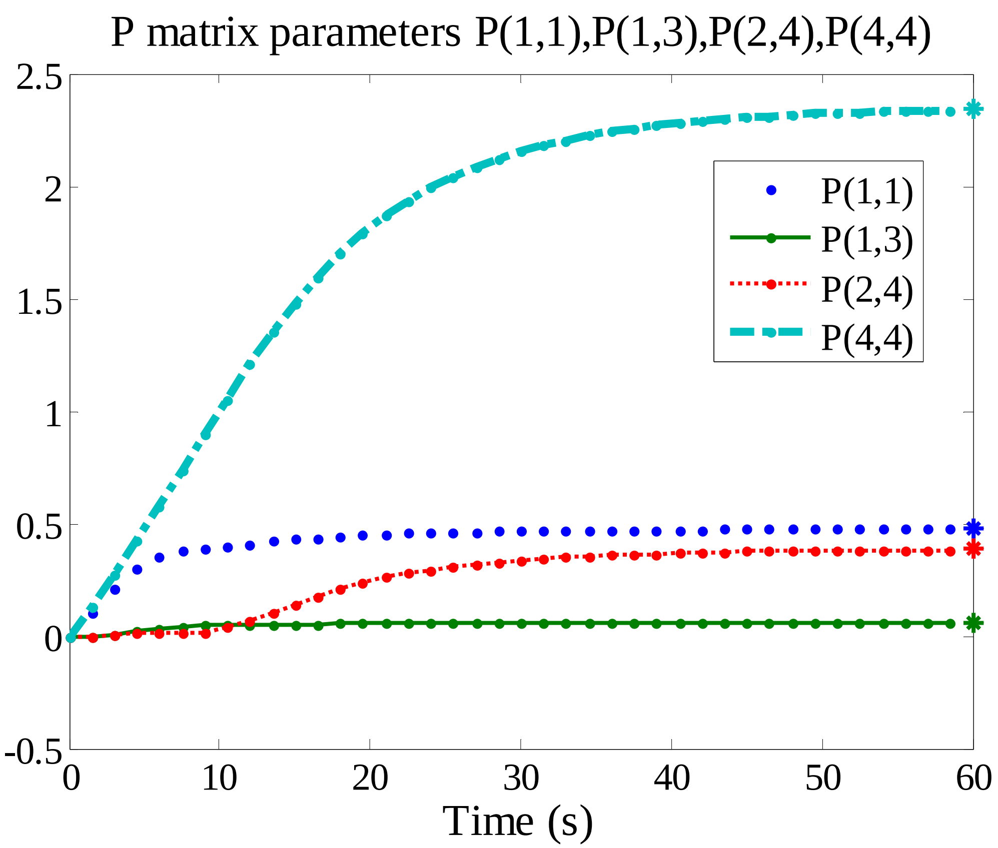
CARE:

$$A^T P + PA - PBR^{-1}B^T P + Q = 0$$

$$P_{CARE} = \begin{bmatrix} 0.4750 & 0.4766 & 0.0601 & 0.4751 \\ 0.4766 & 0.7831 & 0.1237 & 0.3829 \\ 0.0601 & 0.1237 & 0.0513 & 0.0298 \\ 0.4751 & 0.3829 & 0.0298 & 2.3370 \end{bmatrix}$$

CT HDP results

$$V^*(x_0) = \min_{u(t)} \int_0^\infty (x^T Q_{CT} x + u^T R_{CT} u) d\tau$$



Convergence of the P matrix parameters for CT HDP

The state measurements were taken at each 0.1s time period.

A cost function update was performed at each 1.5s.

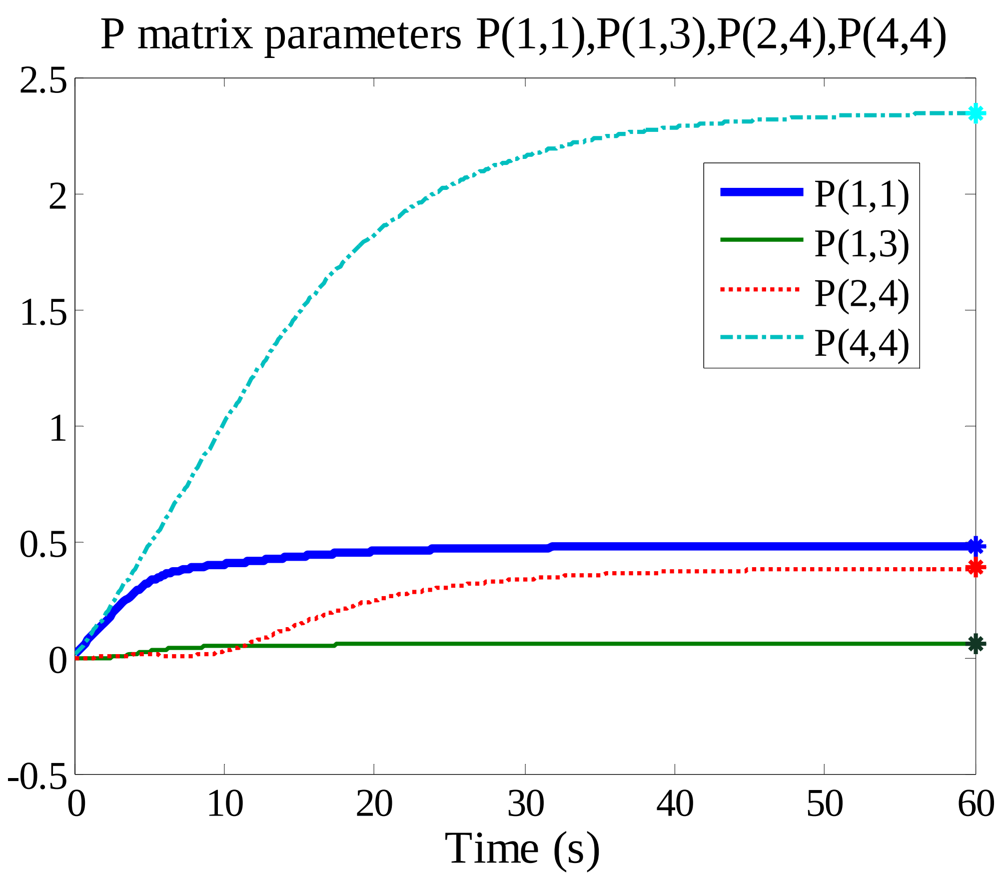
For the 60s duration of the simulation a number of 40 iterations (control policy updates) were performed.

$$P_{CT-HDP} = \begin{bmatrix} 0.4753 & 0.4771 & 0.0602 & 0.4770 \\ 0.4771 & 0.7838 & 0.1238 & 0.3852 \\ 0.0602 & 0.1238 & 0.0513 & 0.0302 \\ 0.4770 & 0.3852 & 0.0302 & 2.3462 \end{bmatrix}$$

The discrete version was obtained by discretizing the continuous time model using zero-order hold method with the sample time T=0.01s.

DT ADHDP results

$$V^*(x_k) = \min_{u_{t\in[k,\infty]}} \sum_{t=k}^{\infty} \left[x_t^T (Q_{CT} T) x_t + u_t^T (R_{CT} T) u_t \right]$$



Convergence of the P matrix parameters for DT ADHDP

The state measurements were taken at each 0.01s time period.

A cost function update was performed at each .15s.

For the 60s duration of the simulation a number of 400 iterations (control policy updates) were performed.

$$P_{DT-ADHDP} = \begin{bmatrix} 0.4802 & 0.4768 & 0.0603 & 0.4754 \\ 0.4768 & 0.7887 & 0.1239 & 0.3834 \\ 0.0603 & 0.1239 & 0.0567 & 0.0300 \\ 0.4754 & 0.3843 & 0.0300 & 2.3433 \end{bmatrix}$$

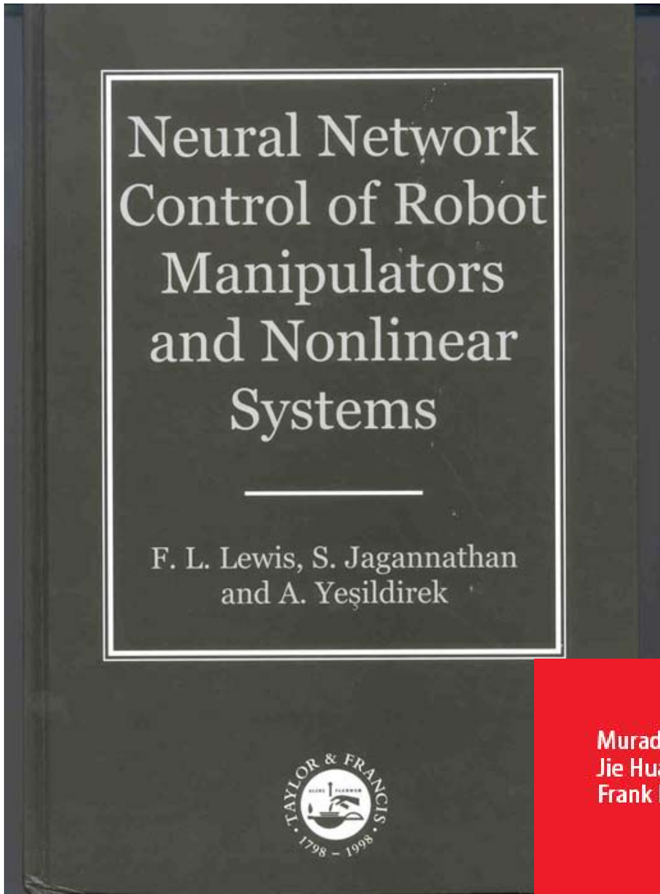
Continuous-time used only 40 iterations!

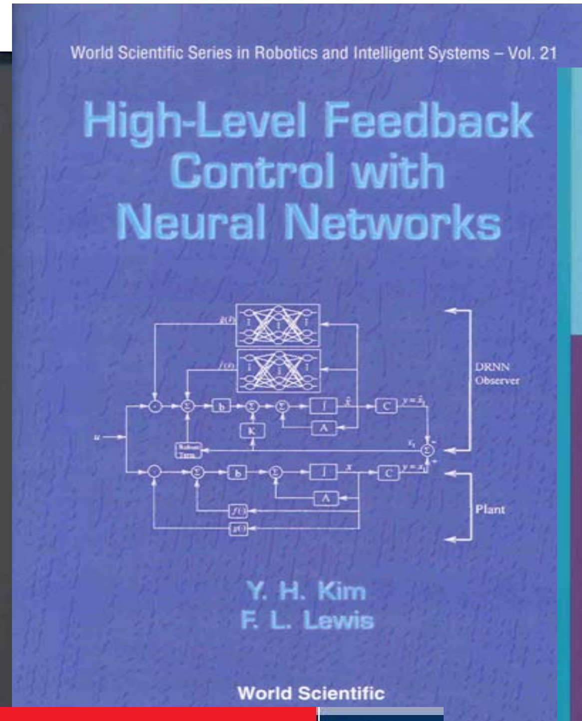
Comparison of CT and DT ADP

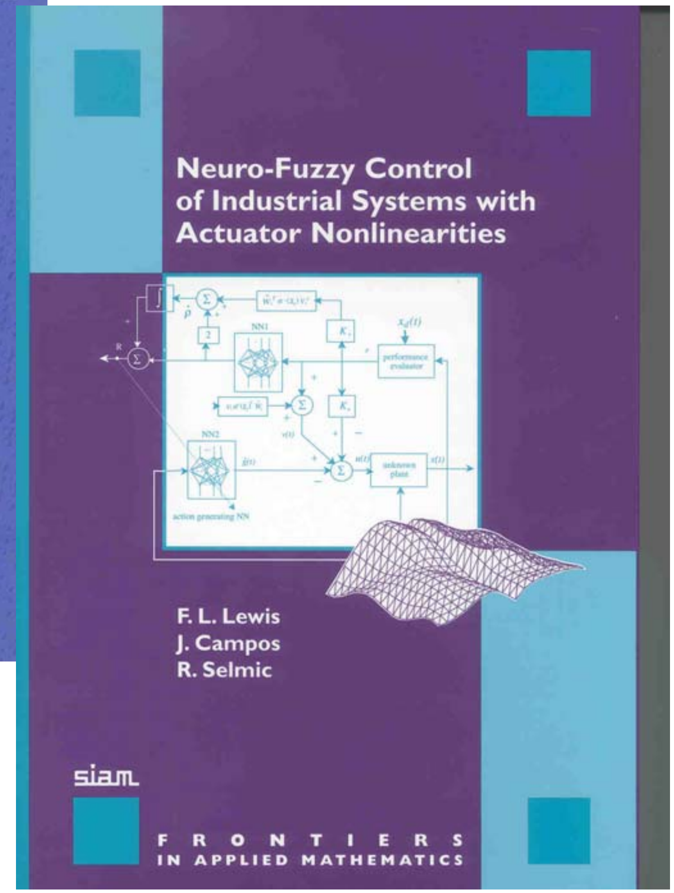
- CT HDP
  - Partially model free (the system A matrix is not required to be known)

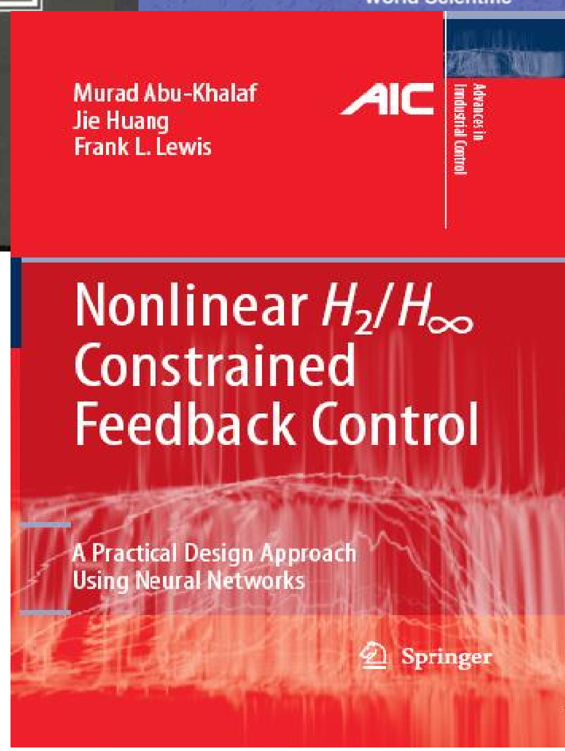
- DT ADHDP – Q learning
  - Completely model free

The DT ADHP algorithm is computationally more intensive than the CT HDP since it is using a smaller sampling period









4 US Patents

Sponsored by Paul Werbos
NSF

Call for Papers

IEEE Transactions on Systems, Man, & Cybernetics- Part B

Special Issue on

Adaptive Dynamic Programming and Reinforcement Learning in Feedback Control

George Lendaris
Derong Liu
F.L Lewis

Papers due 1 August 2007


bebelebe