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

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

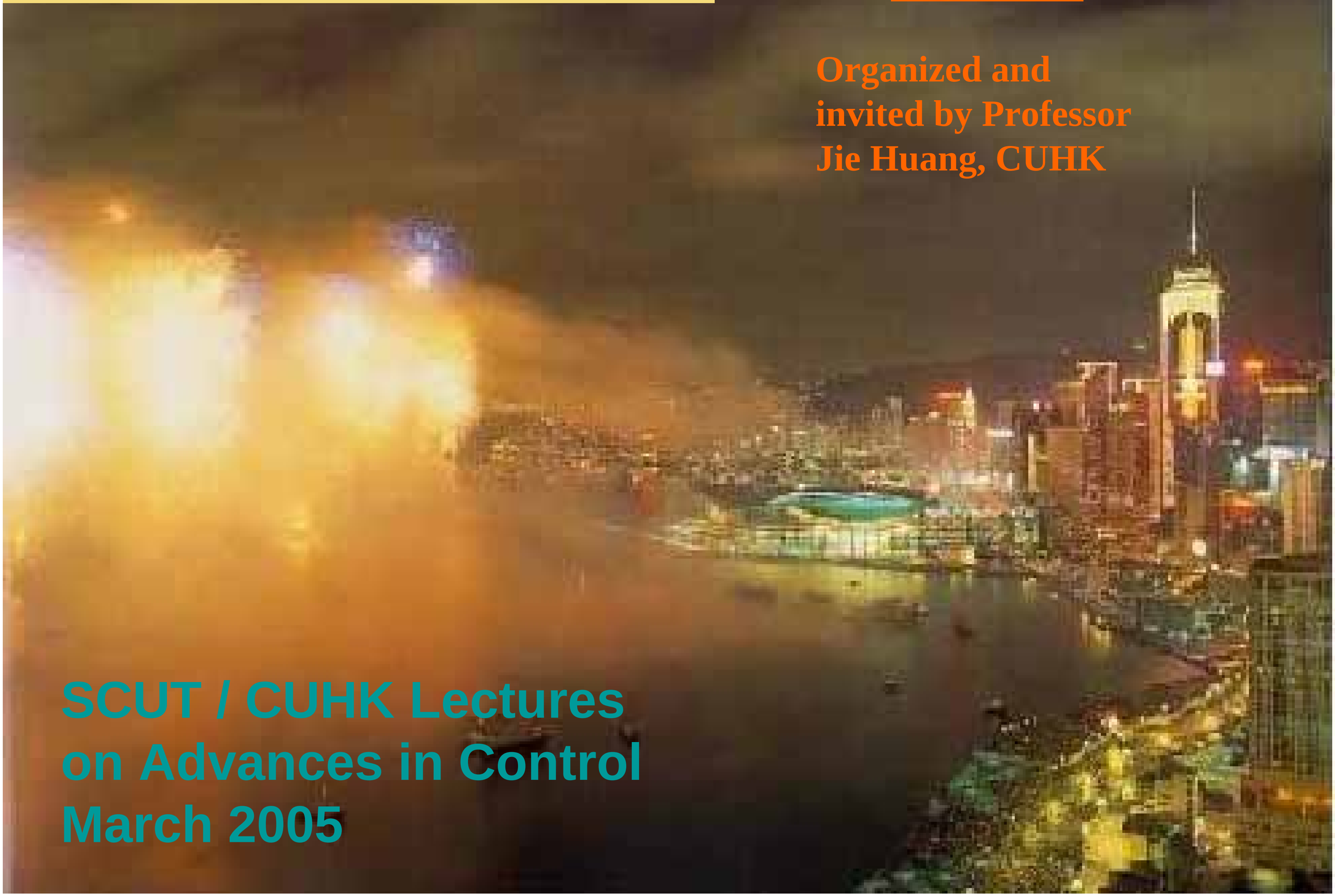
Nonlinear Network Structures for Feedback Control

http://ARRI.uta.edu/acs

香港中文大學
The Chinese University of Hong Kong

NUS National University of Singapore

UTA

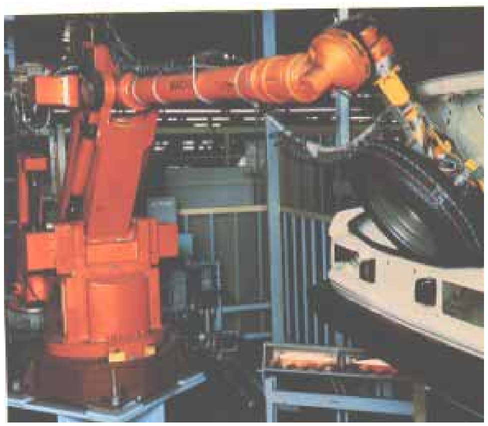


Organized and invited by Professor Jie Huang, CUHK

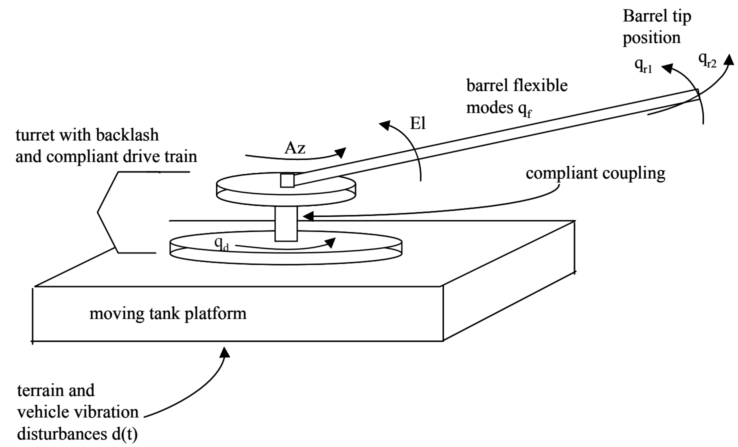
SCUT / CUHK Lectures on Advances in Control
March 2005

Relevance- Machine Feedback Control

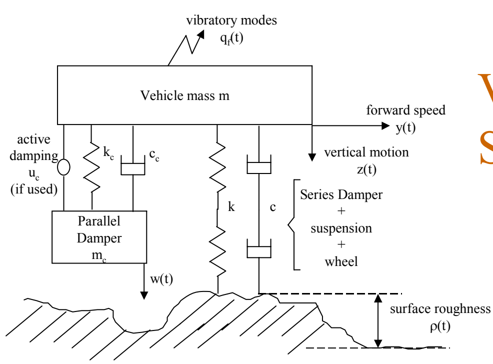
High-Speed Precision Motion Control with unmodeled dynamics, vibration suppression, disturbance rejection, friction compensation, deadzone/backlash control



Industrial Machines



Military Land Systems



Single-Wheel/Terrain System with Nonlinearities

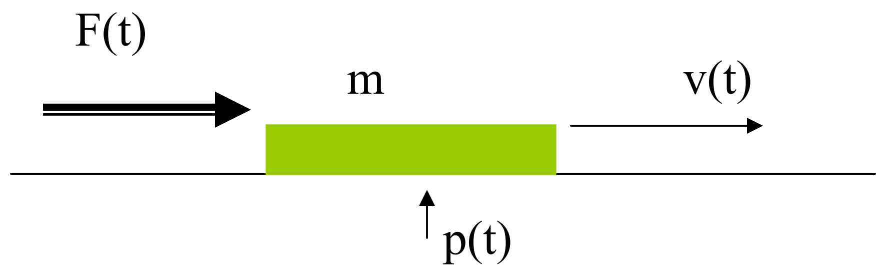
Vehicle Suspension

Aerospace

Newton’s Law

$$F = ma = m\ddot{x}$$

$$\ddot{x} = \frac{F(t)}{m} \equiv u(t)$$



LaGrange’s Eqs. Of Motion $\Longrightarrow$

Mechanical Motion Systems (Vehicles, Robots)

Control Input

$$M(\dot{q})\ddot{q} + V_m(q,\dot{q})\dot{q} + G(q) + F(\dot{q}) + \tau_d = B(q)\tau$$

Actuator problems

inertia

Coriolis/centripetal force

gravity

friction

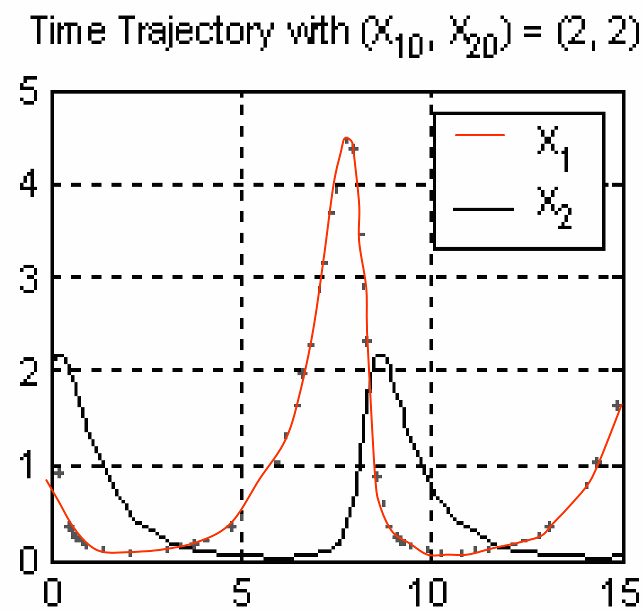
disturbances

Darwinian Selection & Population Dynamics

Volterra's fishes

$$\dot{x}_1 = ax_1 - bx_1x_2$$
$$\dot{x}_2 = -cx_2 + dx_1x_2$$

x_1= prey
x_2= predator



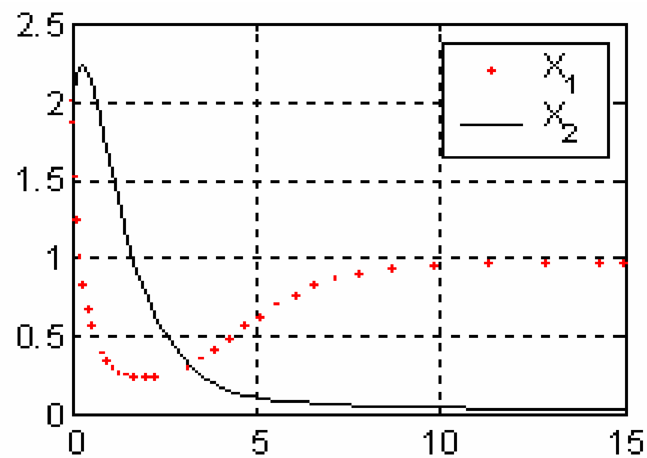
Stable
Limit Cycle

Effects of Overcrowding

Limited food and resources

$$\dot{x}_1 = ax_1 - bx_1x_2 - ex_1{}^2$$
$$\dot{x}_2 = -cx_2 + dx_1x_2$$

Favorable to Prey!



Stable
Equilibrium POINT

Dynamical System Models

Continuous-Time Systems	Discrete-Time Systems

Nonlinear system

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

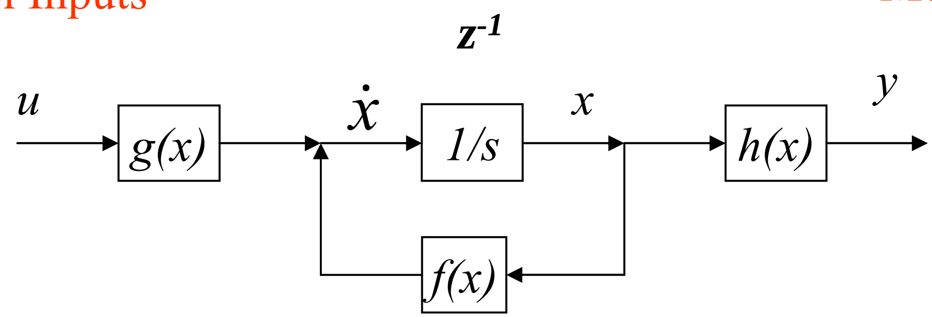
Linear system

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

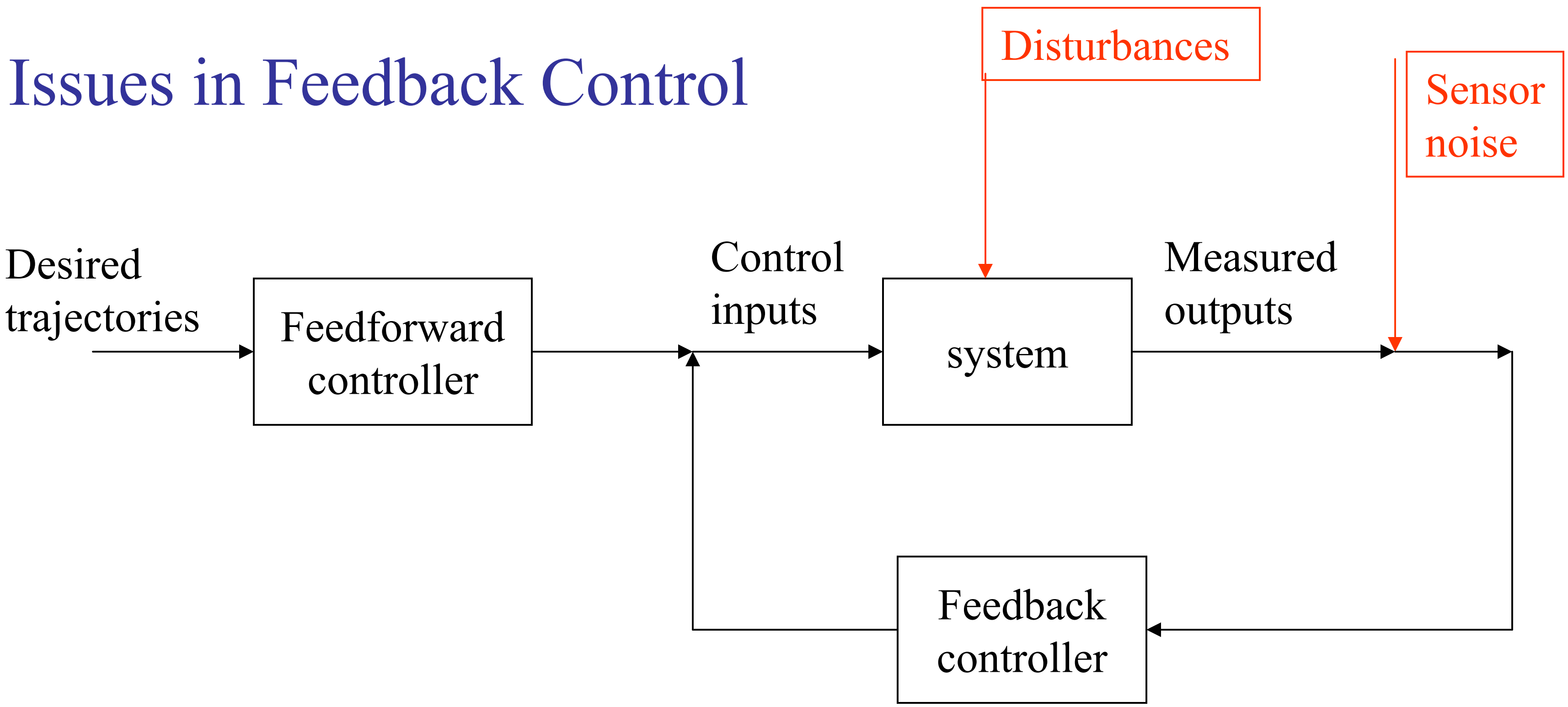
Control Inputs

Internal States

Measured Outputs



Issues in Feedback Control

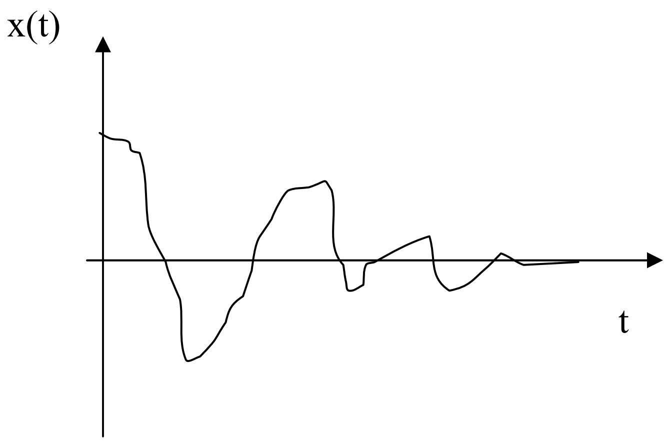


Stability
Tracking
Boundedness
Robustness
- to disturbances
- to unknown dynamics

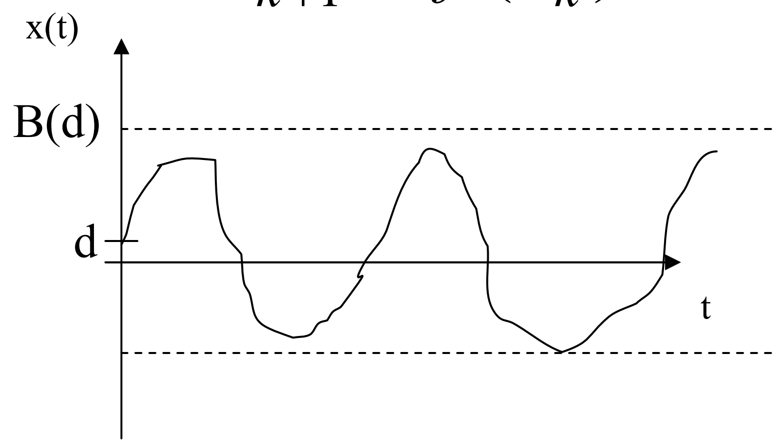
Definitions of System Stability

$$\dot{x} = f(x)$$

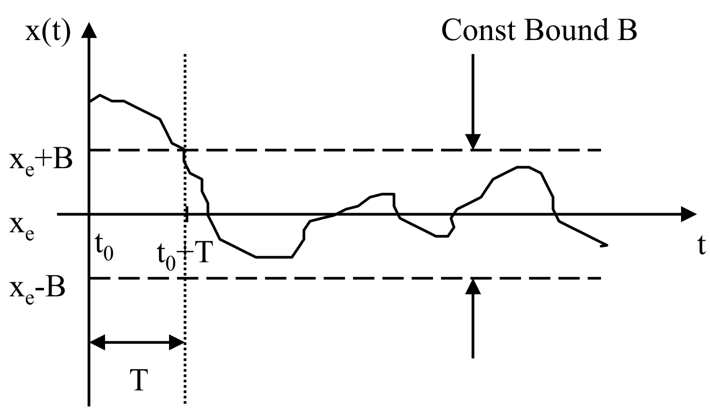
$$x_{k+1} = f(x_k)$$



Asymptotic Stability

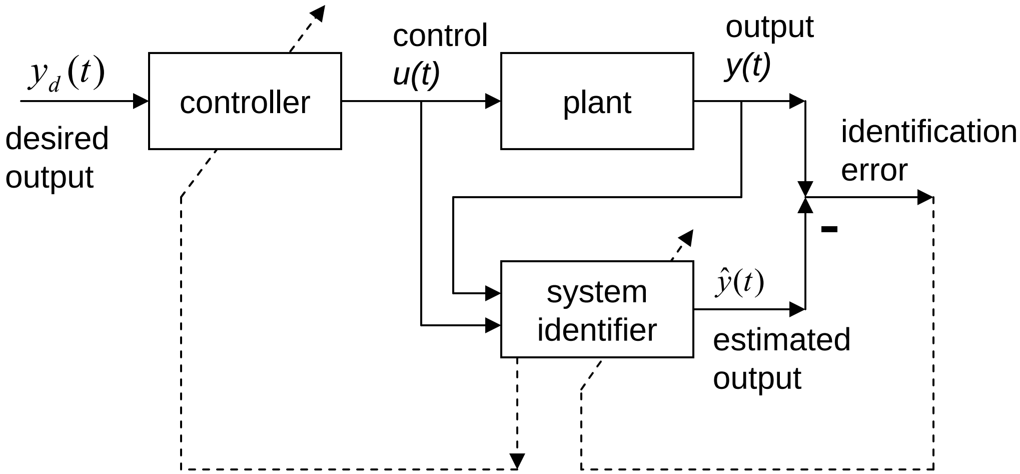


Marginal Stability

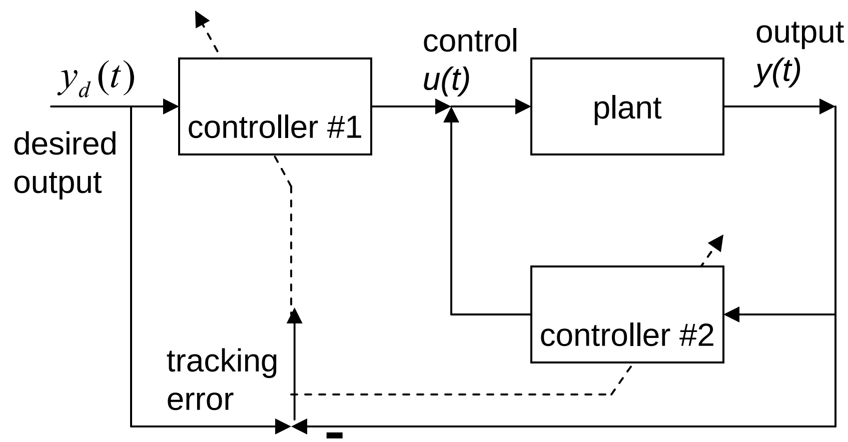


Uniform Ultimate Boundedness

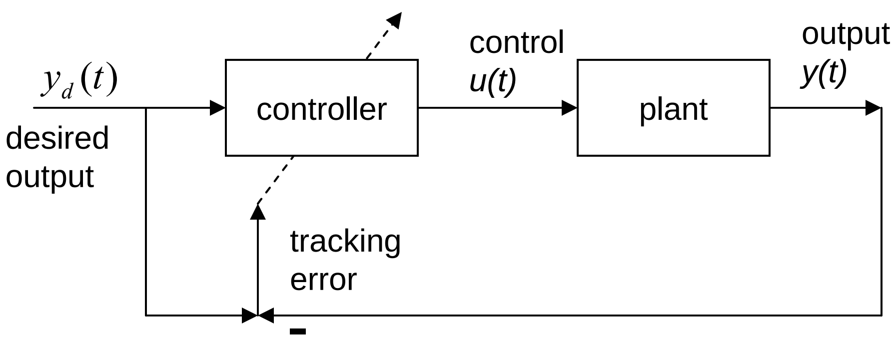
Controller Topologies



Indirect Scheme



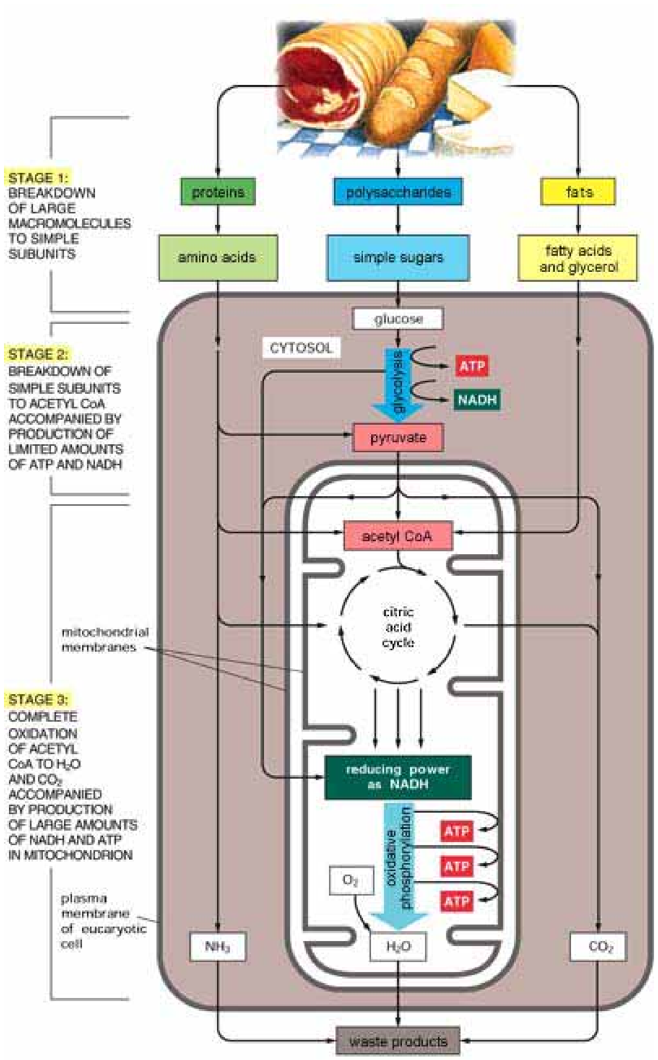
Feedback/Feedforward Scheme



Direct Scheme

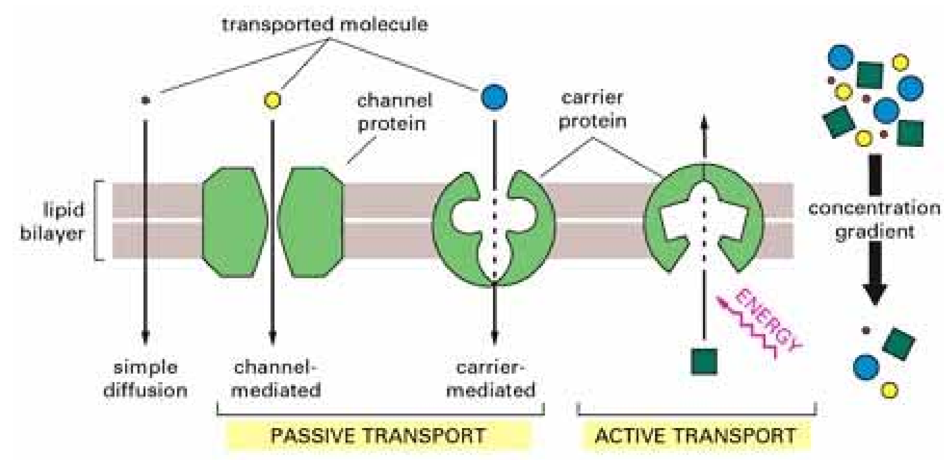
Optimality in Biological Systems

Cell Homeostasis



Cellular Metabolism

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.



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

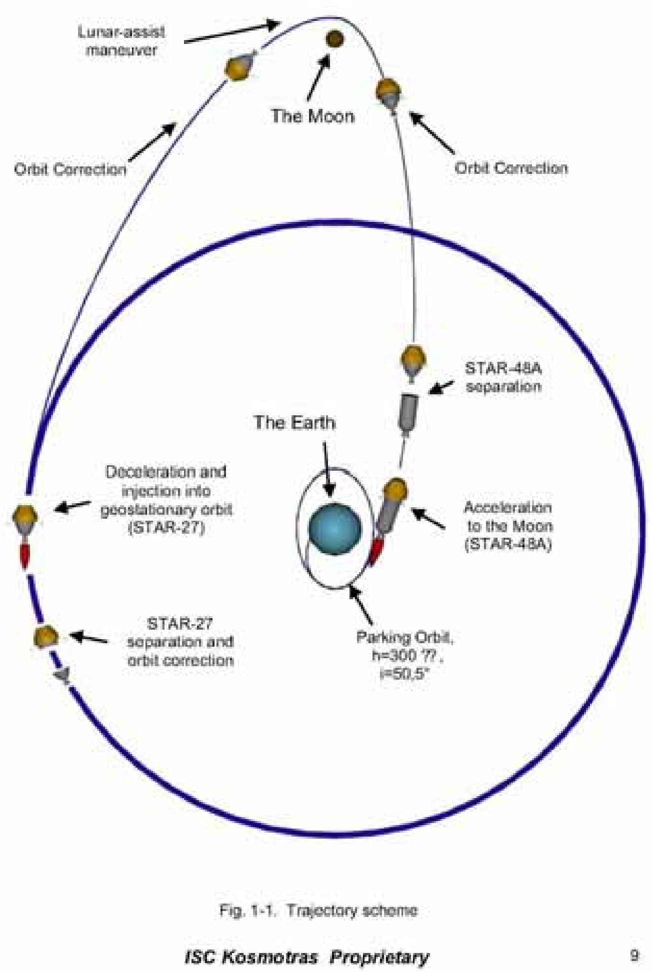


Fig. 1-1. Trajectory scheme

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

Performance Index, Cost, or Value function

CT $$J = \int_0^T [Q(x) + R(u)]\, dt = \int_0^T r(x,u)\, dt$$

DT $$J = \sum_{k=0}^{N} \underbrace{r(x_k, u_k)}_{\text{utility}}$$

Strategic utility

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

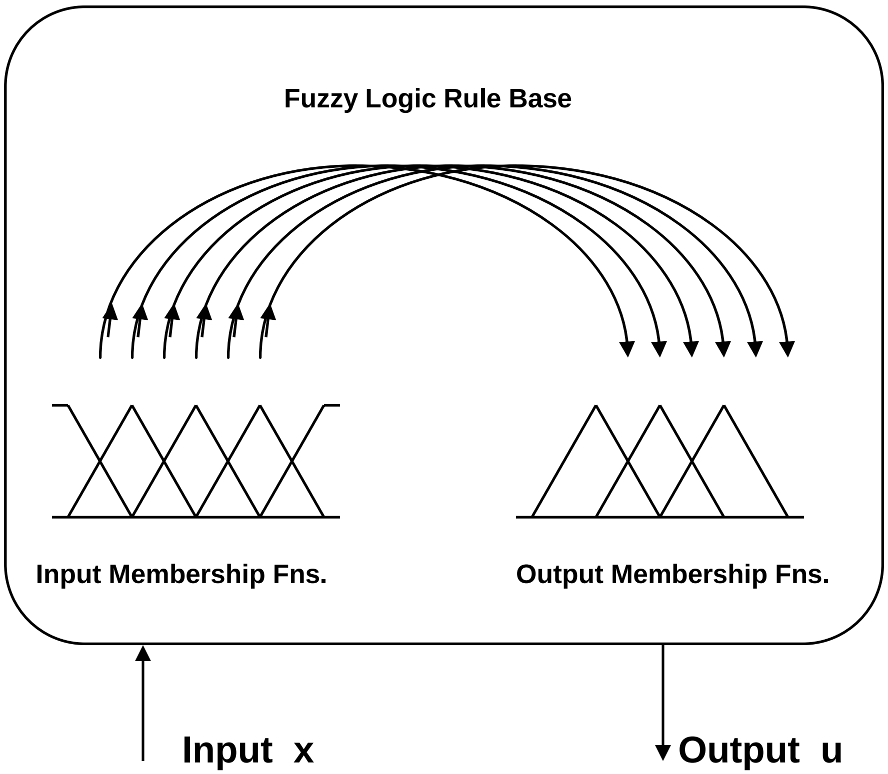
Minimum fuel $$r(x,u) = |u|$$

Minimum time $$r(x,u) = 1$$ Then $$J = \int_0^T r(x,u)\, dt = T$$

Discounting $$J = \sum_{k=0}^{N} \gamma^k r(x_k, u_k)$$ $$J = \int_0^T e^{-\gamma t} r(x,u)\, dt$$

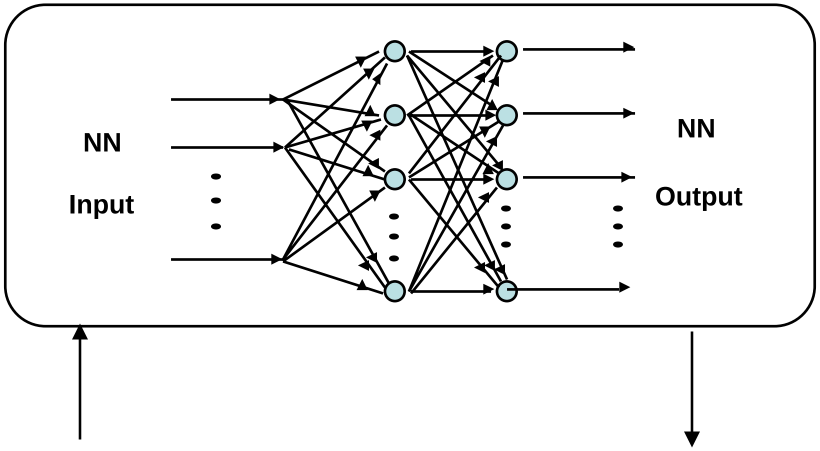
INTELLIGENT CONTROL TOOLS

Fuzzy Associative Memory (FAM)



Neural Network (NN)

(Includes Adaptive Control)



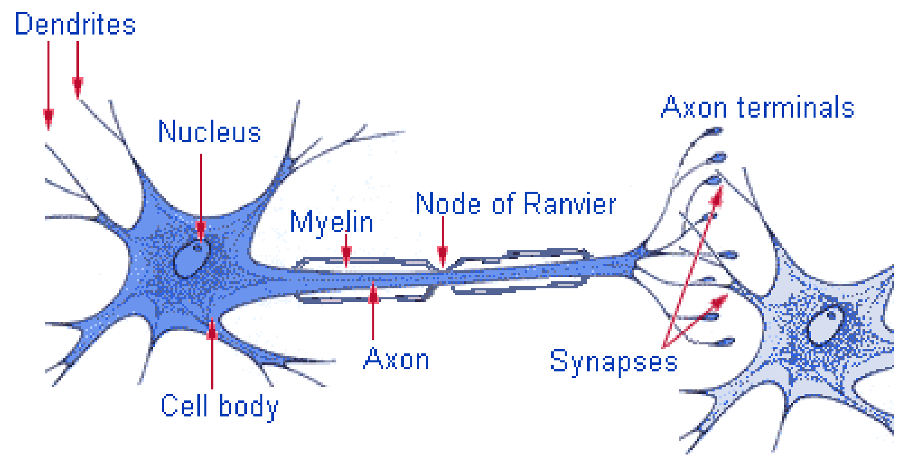
Both FAM and NN define a function u= f(x) from inputs to outputs

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

First groups working on NN Feedback Control in CS community

Werbos

Narendra
Sanner & Slotine
F.C. Chen & Khalil
Lewis
Polycarpou & Ioannou
Christodoulou & Rovithakis

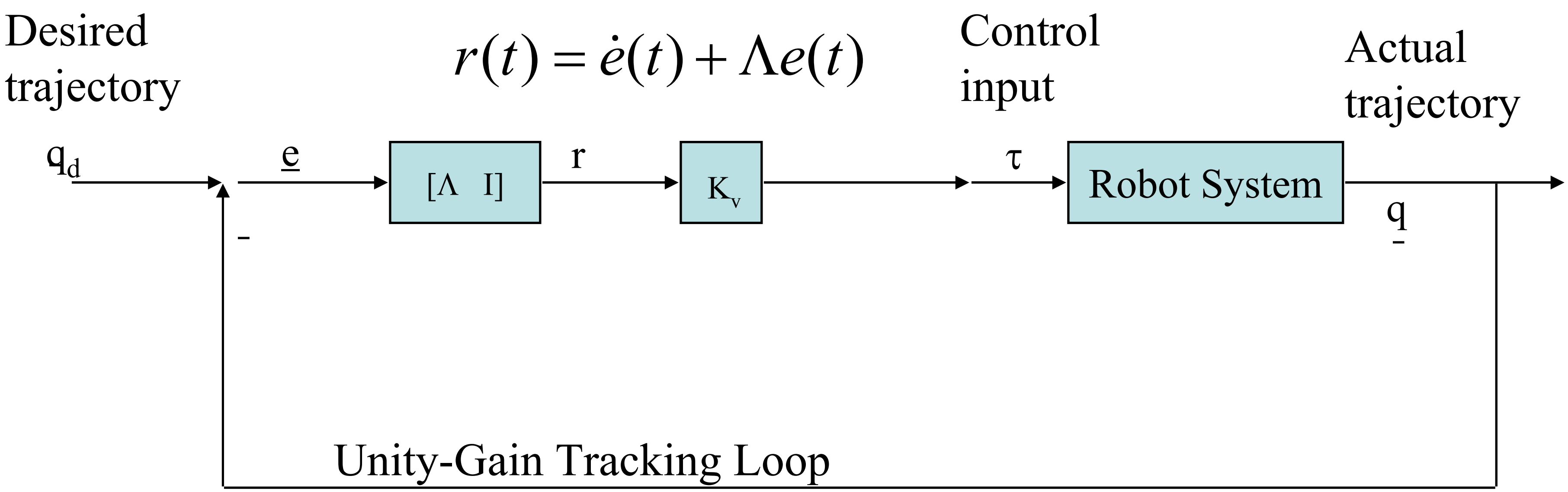
c. 1995

A.J. Calise, McFarland, Naira Hovakimyan
Edgar Sanchez & Poznyak
Sam Ge, Zhang, et al.

Jun Wang, Chinese Univ. Hong Kong

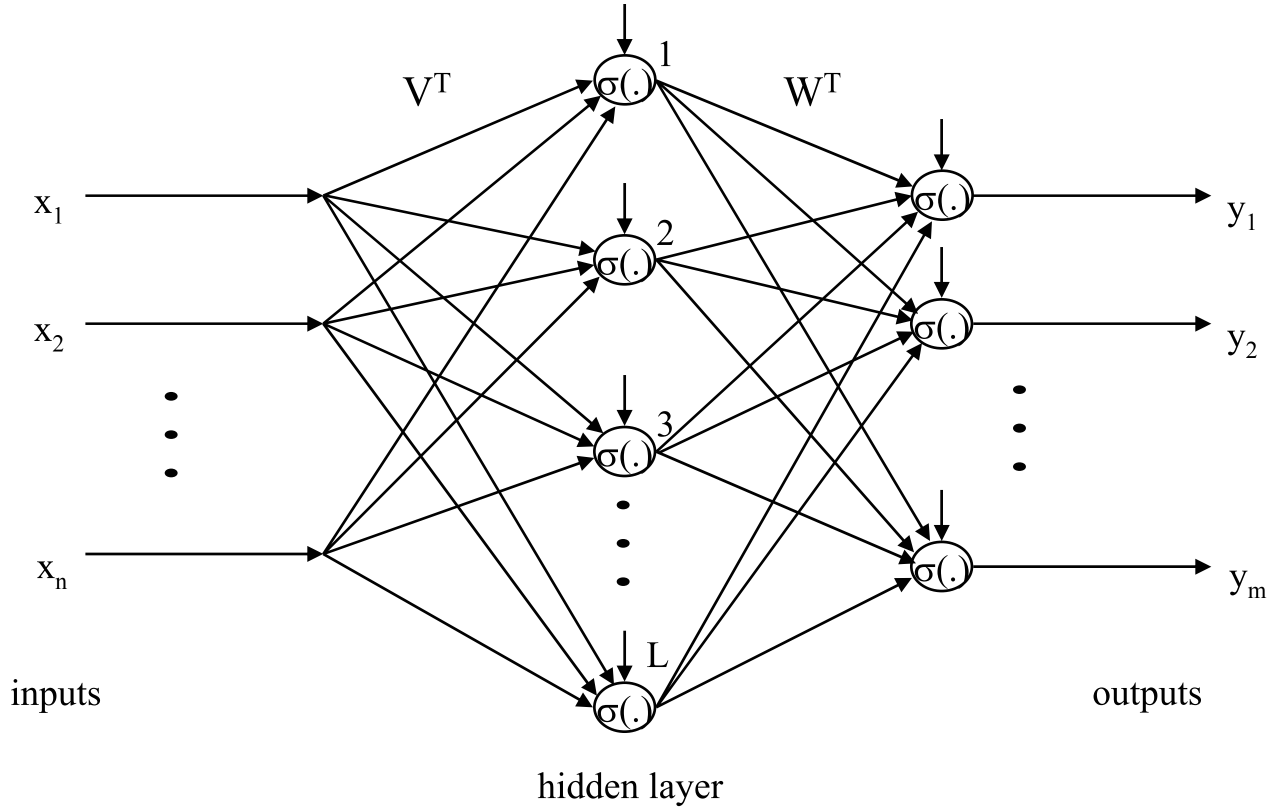
Industry Standard- PD Controller

Easy to implement with COTS controllers
Fast
Can be implemented with a few lines of code- e.g. MATLAB



But -- Cannot handle-
High-order unmodeled dynamics
Unknown disturbances
High performance specifications for nonlinear systems
Actuator problems such as friction, deadzones, backlash

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

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

Control System Design Approach

Robot dynamics

$$M(q)\ddot{q} + V_m(q,\dot{q})\dot{q} + G(q) + F(\dot{q}) + \tau_d = \tau$$

Tracking Error definition

$$e(t) = q_d(t) - q(t) \qquad r = \dot{e} + \Lambda e$$

Error dynamics

$$M\dot{r} = -V_m r + f(x) + \tau_d - \tau$$

Tracking error

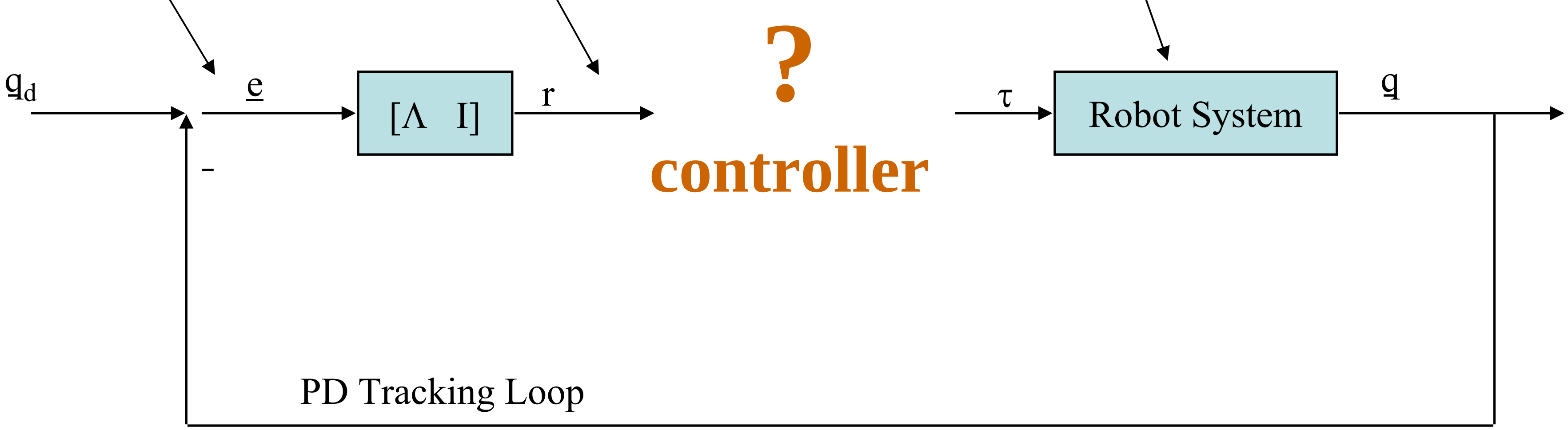
$$e(t) = q_d(t) - q(t)$$

Robot dynamics

$$M(q)\ddot{q} + V_m(q,\dot{q})\dot{q} + G(q) + F(\dot{q}) + \tau_d = \tau$$

Sliding variable

$$r = \dot{e} + \Lambda e$$



The equations give the FB controller structure

Control System Design Approach

Robot dynamics $$M(q)\ddot{q} + V_m(q,\dot{q})\dot{q} + G(q) + F(\dot{q}) + \tau_d = \tau$$

Tracking Error definition $$e(t) = q_d(t) - q(t) \qquad r = \dot{e} + \Lambda e$$

Error dynamics $$M\dot{r} = -V_m r + f(x) + \tau_d - \tau$$

UNKNOWN FN.

Universal Approximation Property

Approx. unknown function by NN $$f(x) = W^T \sigma(V^T x) + \varepsilon$$

Define control input $$\tau = \hat{W}^T \sigma(\hat{V}^T x) + K_v r - v$$

Closed-loop dynamics

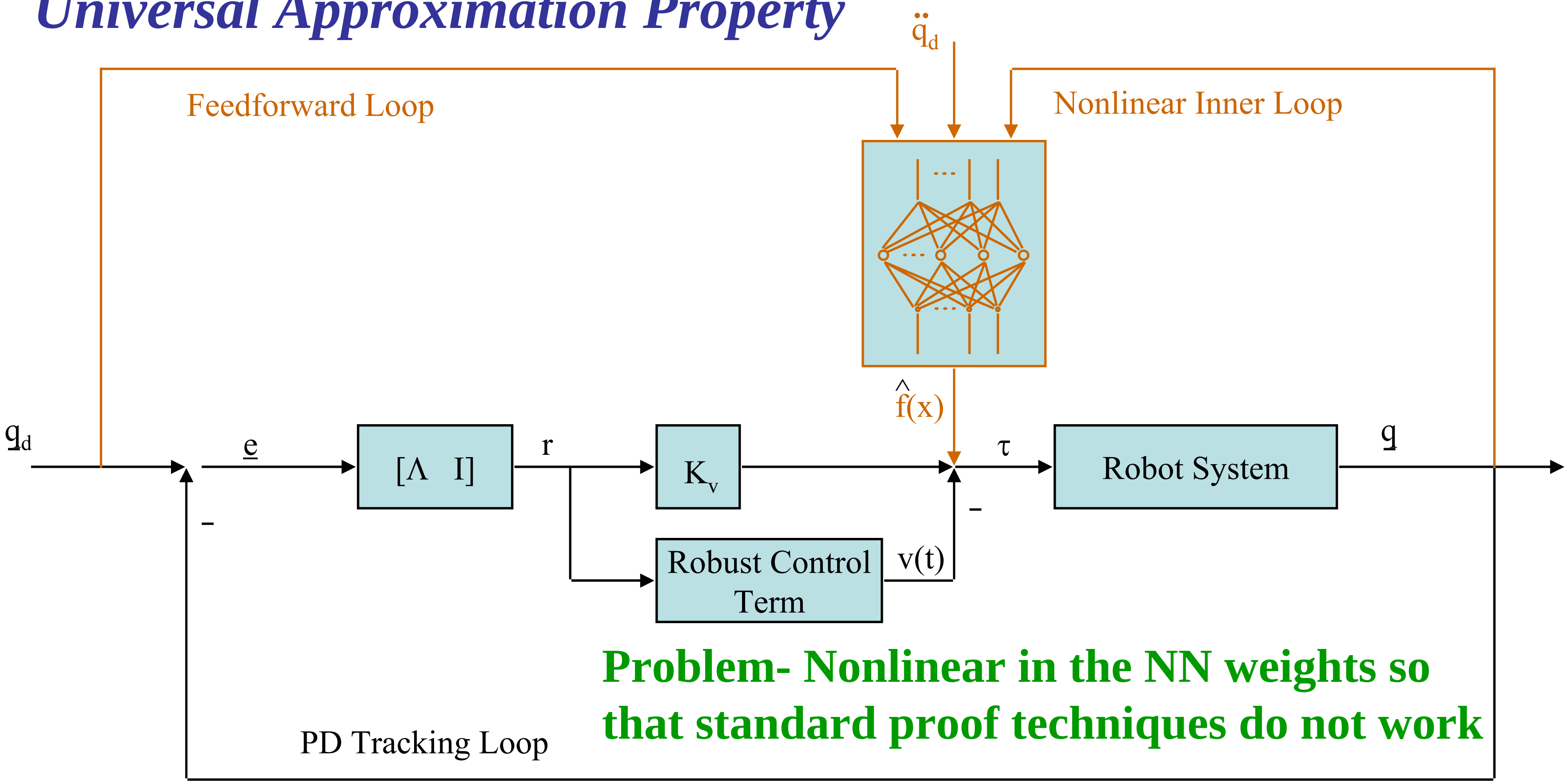
$$M\dot{r} = -V_m r - K_v r + W^T \sigma(V^T x) + \varepsilon - \hat{W}^T \sigma(\hat{V}^T x) + \tau_d + v(t)$$

$$M\dot{r} = -V_m r - K_v r + \tilde{f} + \tau_d + v(t)$$

Neural Network Robot Controller

Universal Approximation Property

Feedback linearization



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

Stability Proof based on Lyapunov Extension

Define a Lyapunov Energy Function

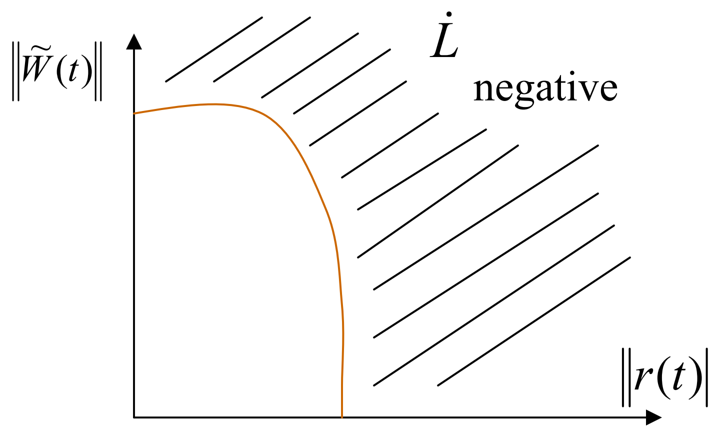
$$L = \tfrac{1}{2} r^T M r + \tfrac{1}{2} tr(\widetilde{W}^T \widetilde{W}) + \tfrac{1}{2} tr(\widetilde{V}^T \widetilde{V})$$

Differentiate

$$\dot{L} = -r^T K_v r + \tfrac{1}{2} r^T (\dot{M} - 2V_m) r$$
$$+ tr\, \widetilde{W}^T (\dot{\tilde{W}} + \hat{\sigma} r^T - \hat{\sigma}' \hat{V}^T x r^T)$$
$$+ tr\, \widetilde{V}^T (\dot{\tilde{V}} + x r^T \hat{W}^T \hat{\sigma}') + r^T (w + v)$$

Problems—
1. How to characterize the NN weight errors as 'small'?- use Frobenius Norm
2. Nonlinearity in the parameters requires extra care in the proof

Using certain special tuning rules, one can show that the energy derivative is negative outside a compact set.



This proves that all signals are bounded

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

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

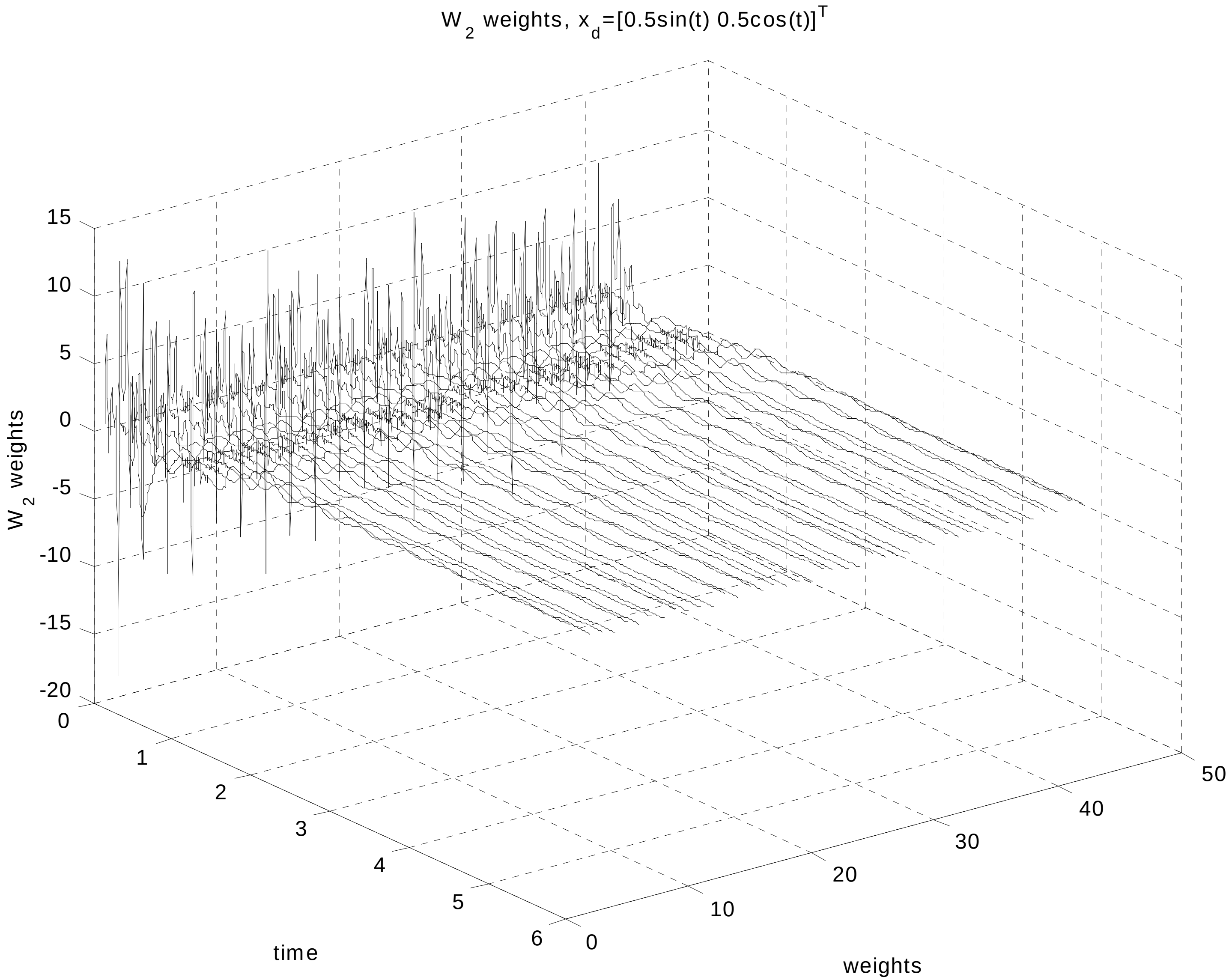
$$\dot{\hat{W}} = F\hat{\sigma}r^T - F\hat{\sigma}'\hat{V}^T x r^T - \kappa F\|r\|\hat{W}, \qquad \dot{\hat{V}} = Gx(\hat{\sigma}'^T \hat{W} r)^T - \kappa G\|r\|\hat{V}$$

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

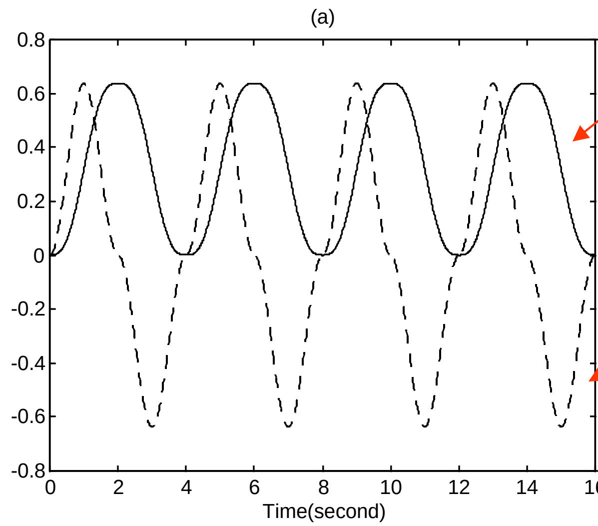


NN weights converge to the best learned values for the given system

NN Friction Compensator

Trajectory Tracking Controller

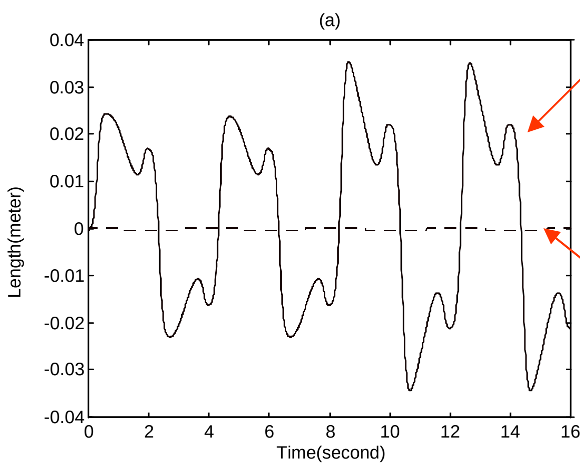
Desired trajectory



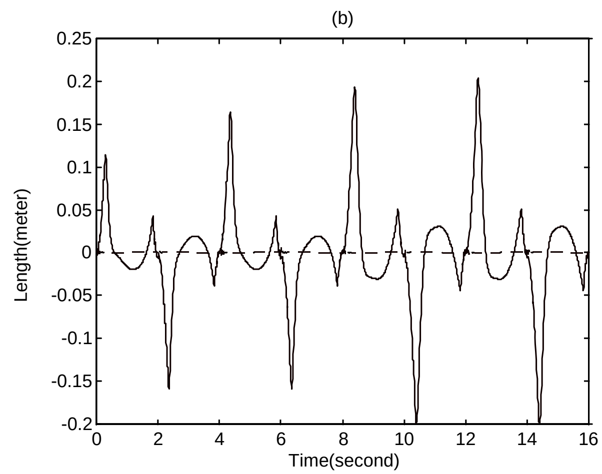
position

velocity

Fixed gain



NN

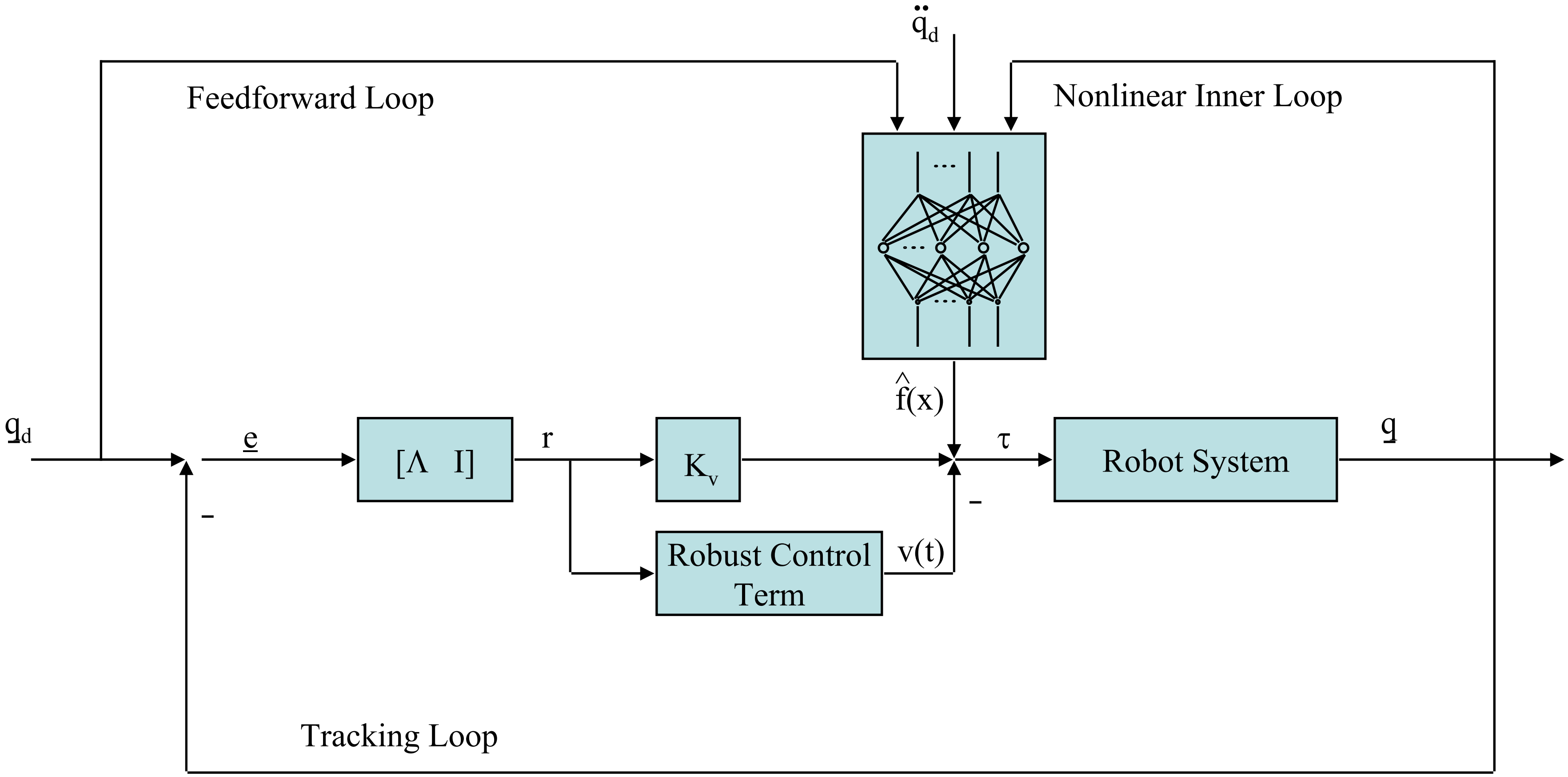


Position

Velocity

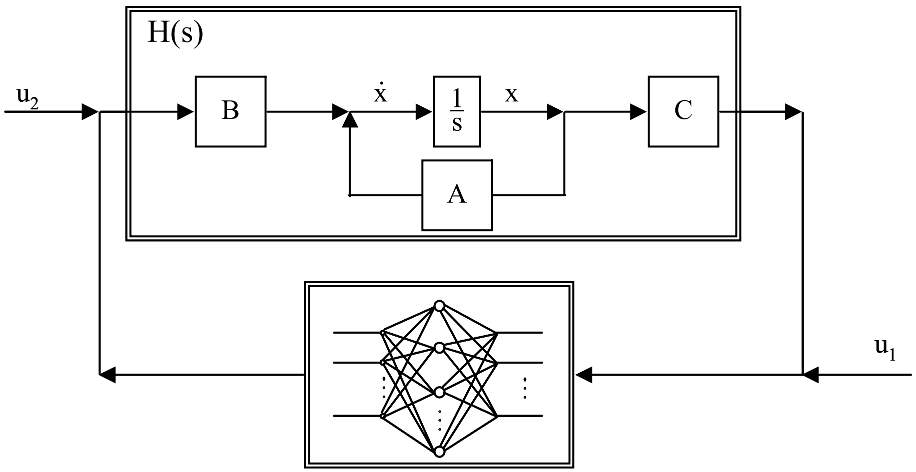
Tracking errors- solid = fixed gain controller, dashed= NN controller

Dynamic NN and Passivity



Static NN => Dynamic NN Feedback Controller

Closed-Loop System wrt Neural Network is a Dynamic (Recursive NN)



Discrete time case

$$x_{k+1} = Ax_k + W^T \sigma(V^T x_k) + u_k$$

The backprop tuning algorithms

$$\dot{\hat{W}} = F\hat{\sigma}r^T - F\hat{\sigma}'\hat{V}^T xr^T$$

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

make the closed-loop system passive

The enhanced tuning algorithms

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

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

make the closed-loop system **state-strict** passive

SSP gives extra robustness properties to disturbances and HF dynamics

Force Control

Flexible pointing systems

SBIR Contracts

Vehicle active suspension

What about practical Systems?

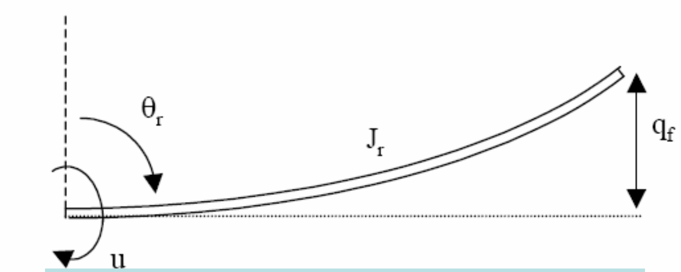
Flexible Systems with Vibratory Modes

Rigid dynamics

$$\begin{bmatrix} M_{rr} & M_{rf} \\ M_{fr} & M_{ff} \end{bmatrix}\begin{bmatrix} \ddot{q}_r \\ \ddot{q}_f \end{bmatrix} + \begin{bmatrix} V_{rr} & V_{rf} \\ V_{fr} & V_{ff} \end{bmatrix}\begin{bmatrix} \dot{q}_r \\ \dot{q}_f \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ o & K_{ff} \end{bmatrix}\begin{bmatrix} q_r \\ q_f \end{bmatrix} + \begin{bmatrix} F_r \\ 0 \end{bmatrix} + \begin{bmatrix} G_r \\ 0 \end{bmatrix} = \begin{bmatrix} B_r \\ B_f \end{bmatrix}\tau$$

Flexible dynamics

Problem- only one control input !



Flexible link pointing system

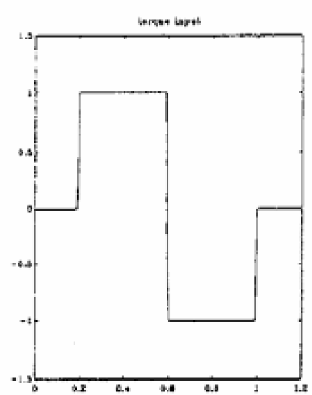


Fig. 2 Acceleration/deceleration torque profile.

acceleration

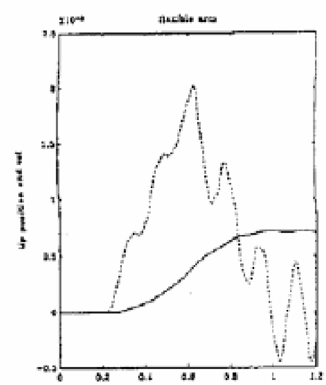


Fig. 3a Open-loop response of flexible arm. Tip position (solid) and vel. (dashed)

velocity
position

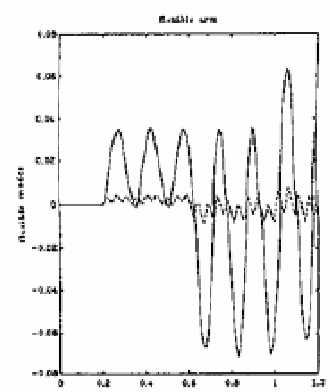
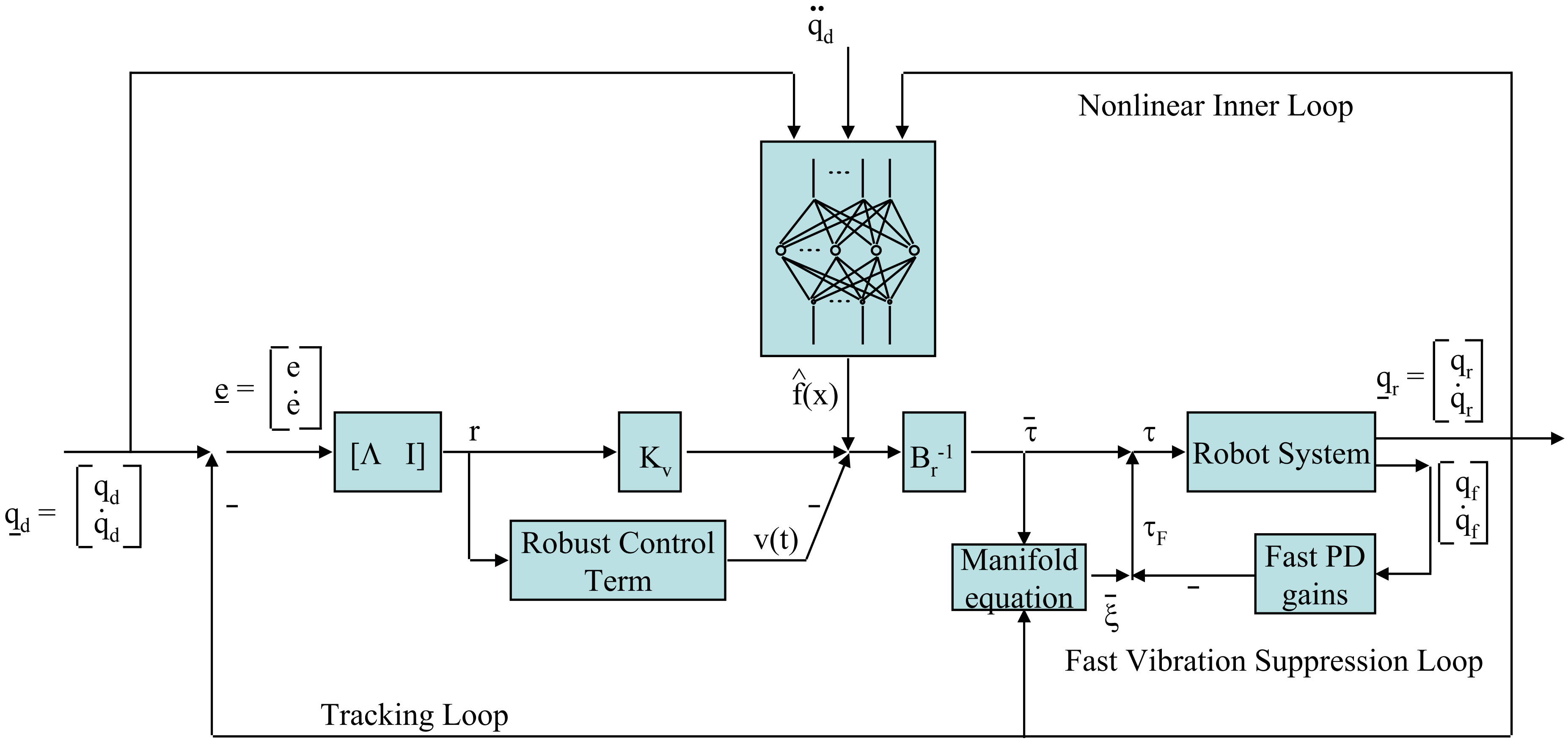


Fig. 3b Open-loop response of flexible arm. Flexible modes.

Flex. modes

Singular Perturbations

Add an extra feedback loop
Use passivity to show stability



Neural network controller for Flexible-Link robot arm

Coupled Systems

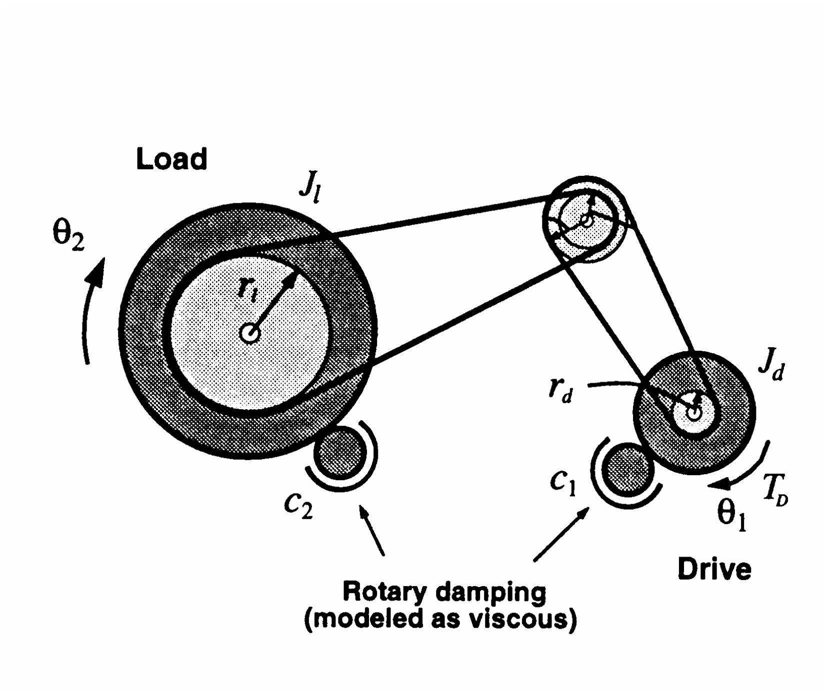
Problem- only one control input !

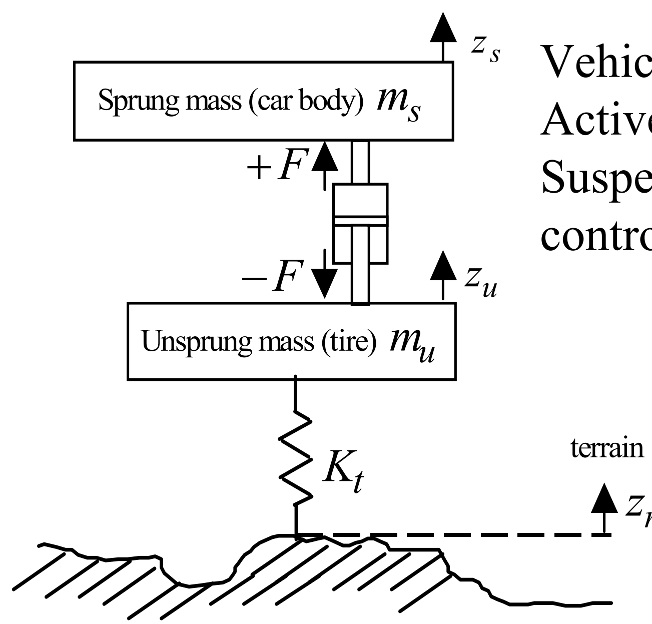
Robot mechanical dynamics

$$M(q)\ddot{q} + V_m(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + \tau_d = K_T i$$

$$L\dot{i} + R(i,\dot{q}) + \tau_e = u_e$$

Motor electrical dynamics

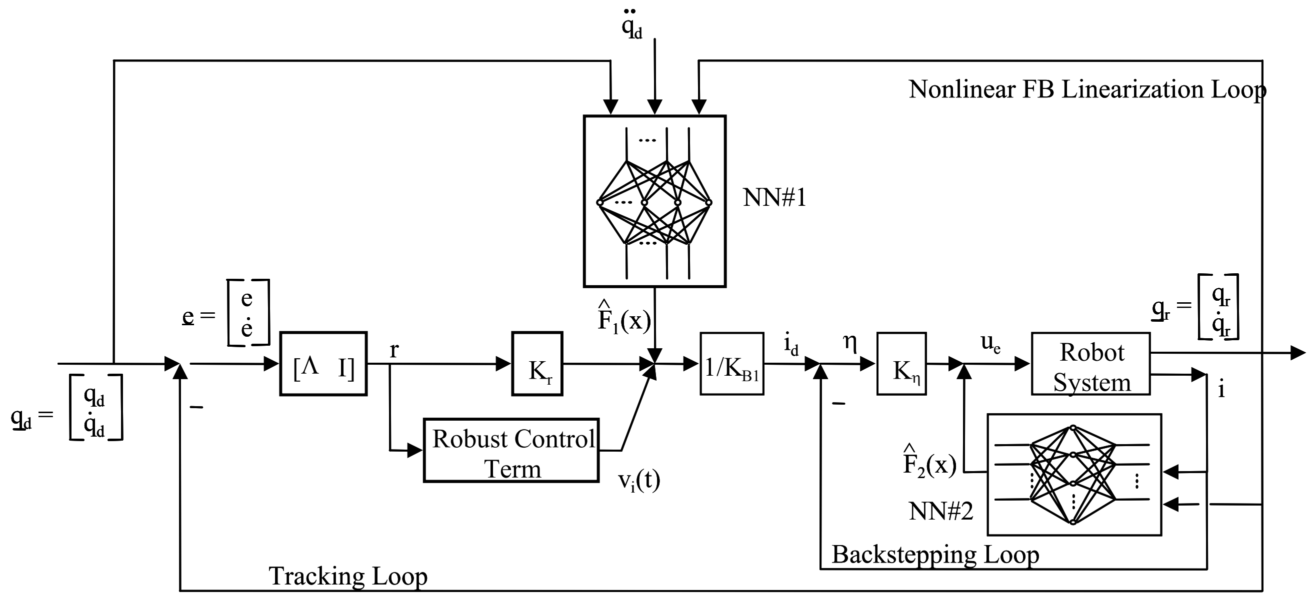




Vehicle Active Suspension control

Backstepping

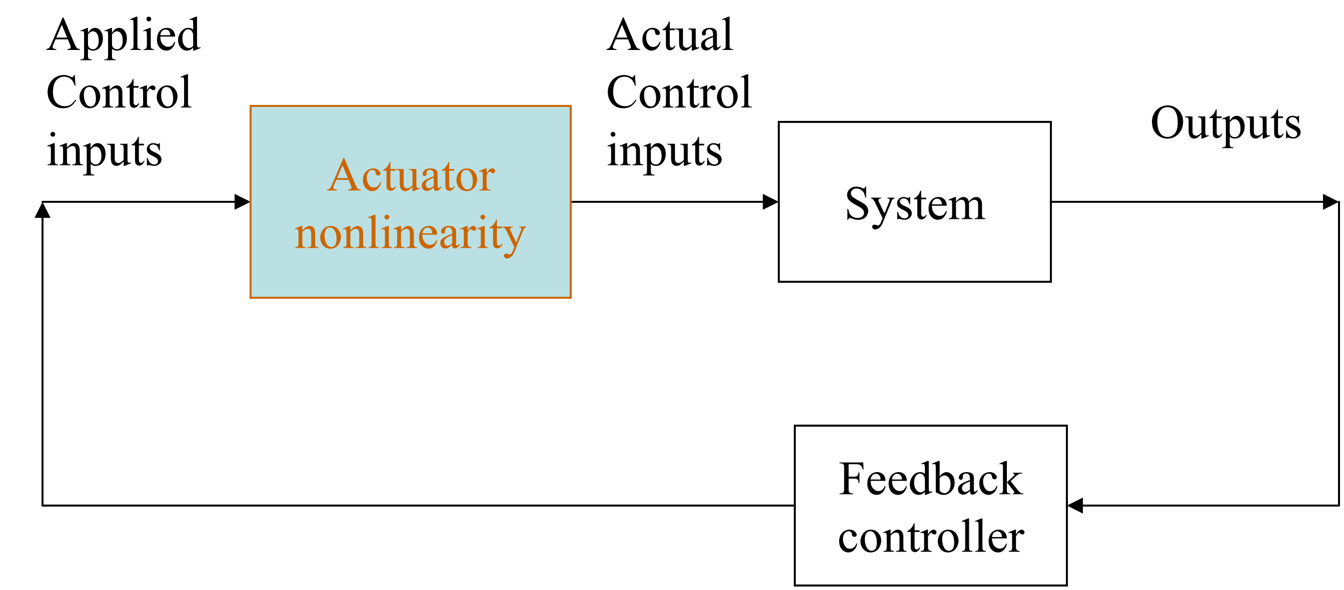
Add an extra feedback loop
Two NN needed
Use passivity to show stability

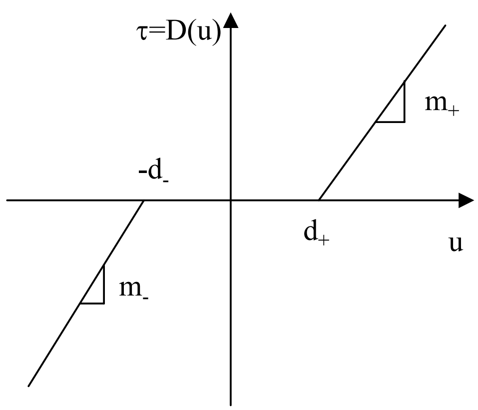


Neural network backstepping controller for Flexible-Joint robot arm

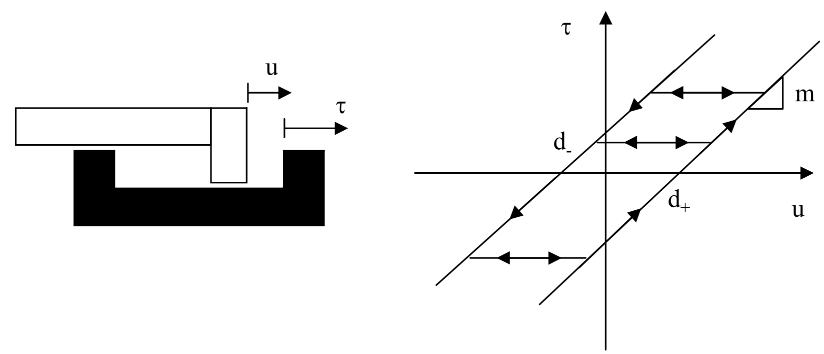
Advantages over traditional Backstepping- no regression functions needed

Actuator Nonlinearities



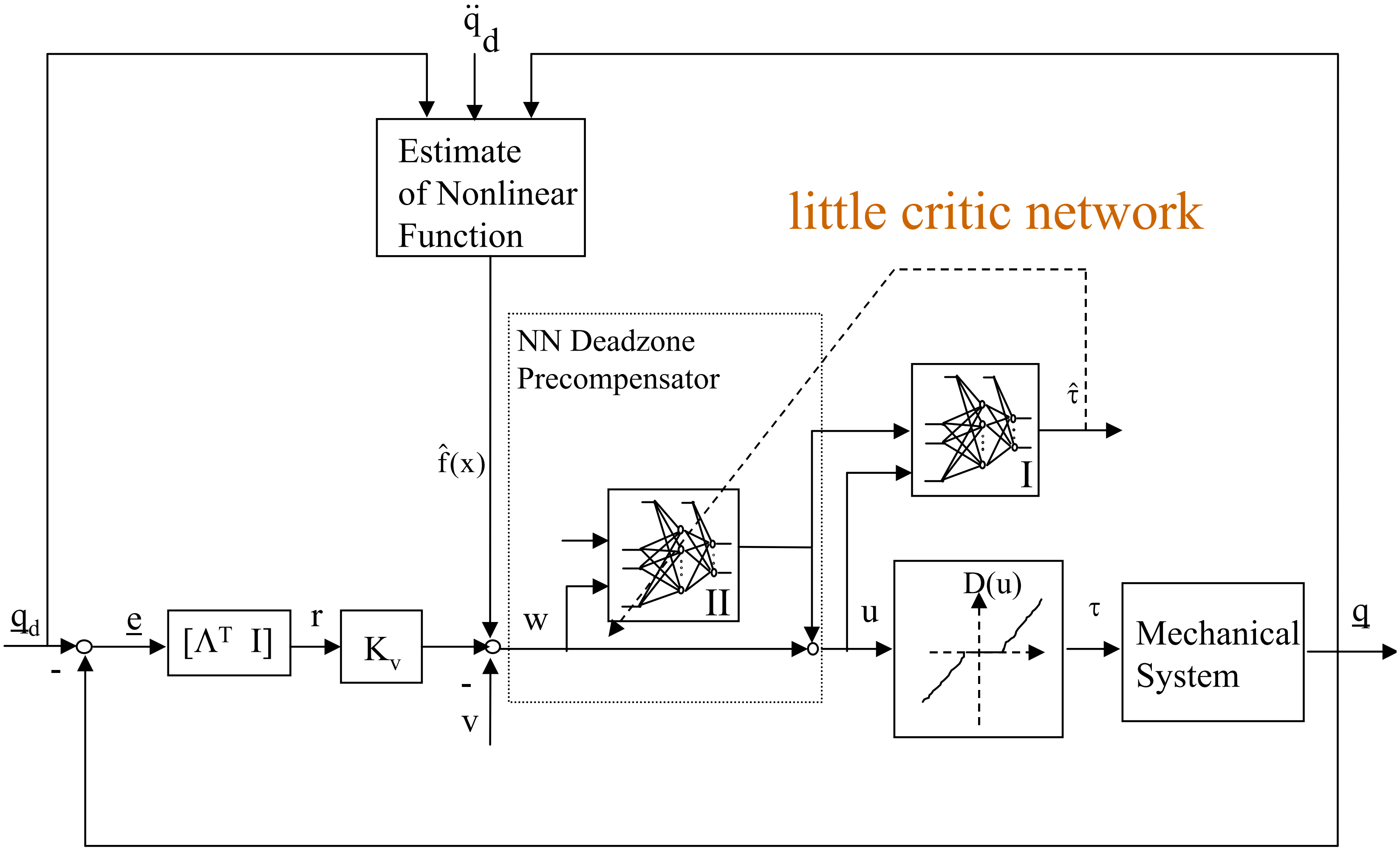


Deadzone



Backlash

NN in Feedforward Loop- Deadzone Compensation

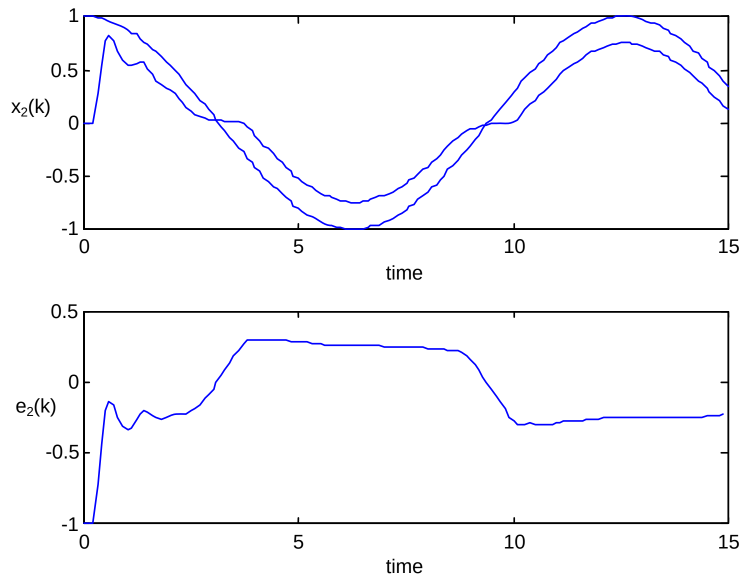


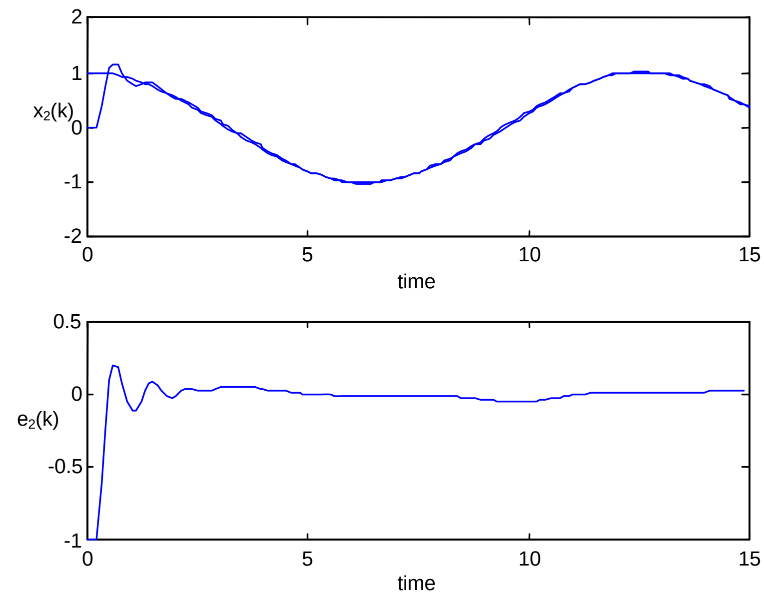
$$\dot{\hat{W}}_i = T\sigma_i(U_i^T w)r^T \hat{W}^T \sigma'(U^T u)U^T - k_1 T\|r\|\hat{W}_i - k_2 T\|r\|\|\hat{W}_i\|\hat{W}_i$$

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

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

Performance Results

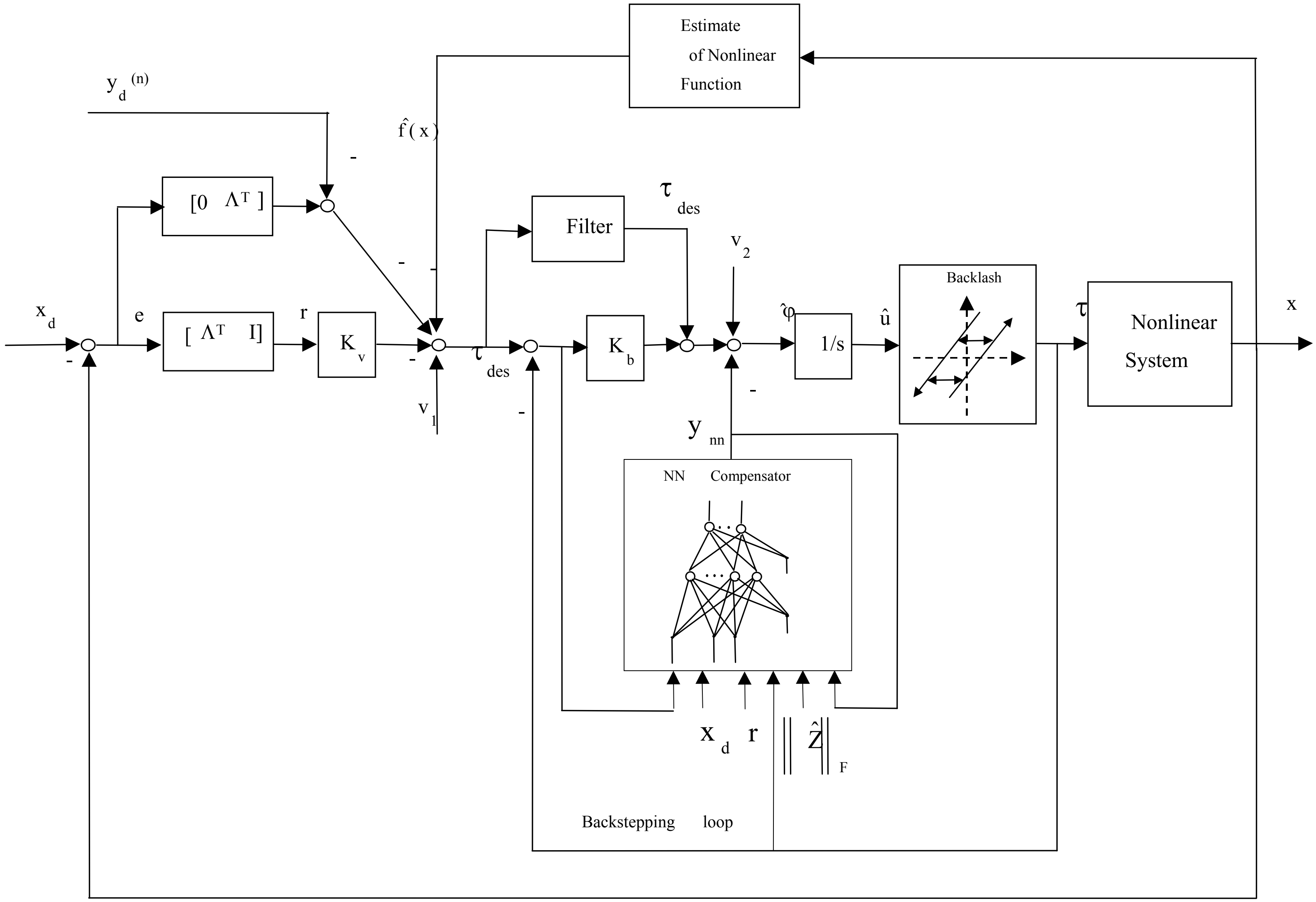




PD control-
deadzone chops out the middle

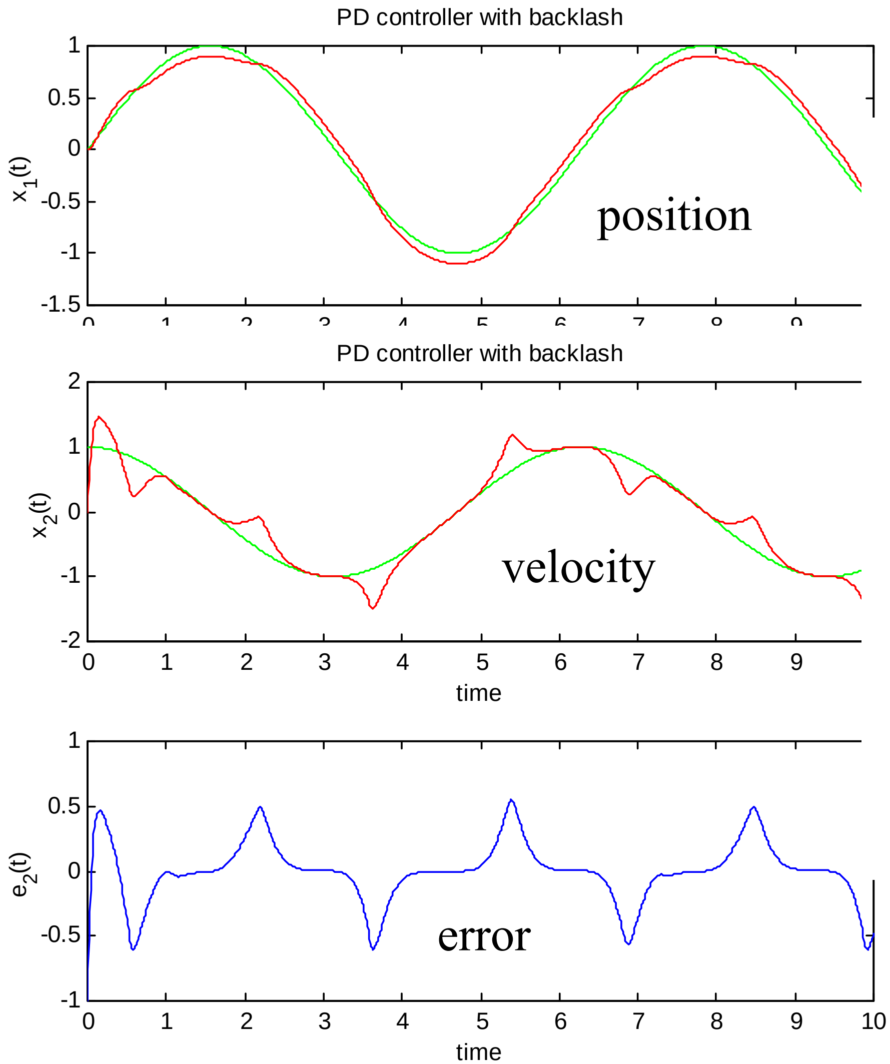
NN control fixes the problem

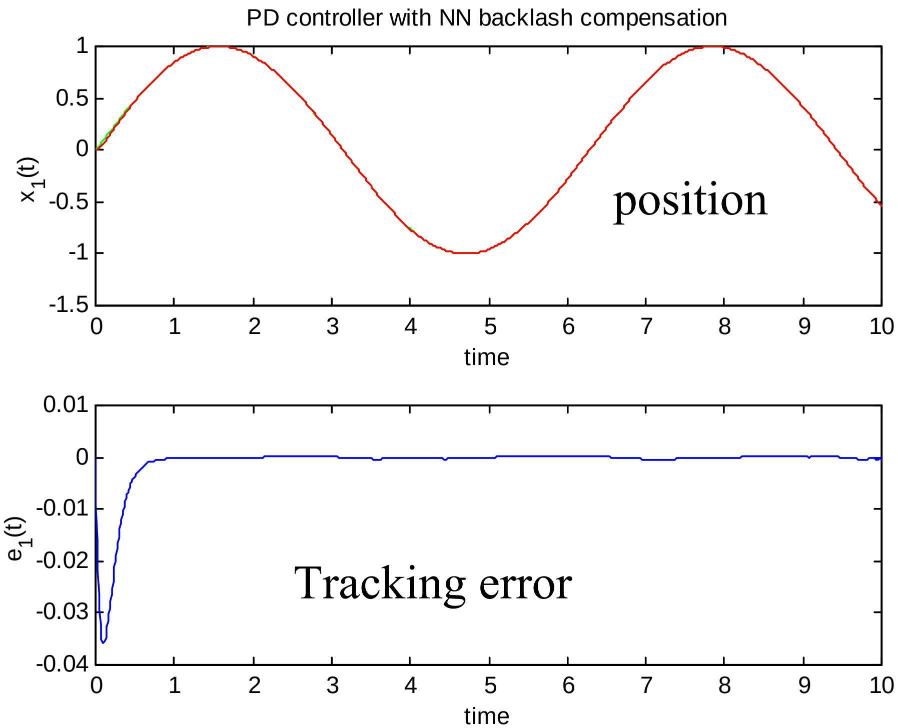
Dynamic inversion NN compensator for system with Backlash



U.S. patent- Selmic, Lewis, Calise, McFarland

Performance Results





NN control fixes the problem

PD control-
backlash chops off tops & bottoms

NN Observers

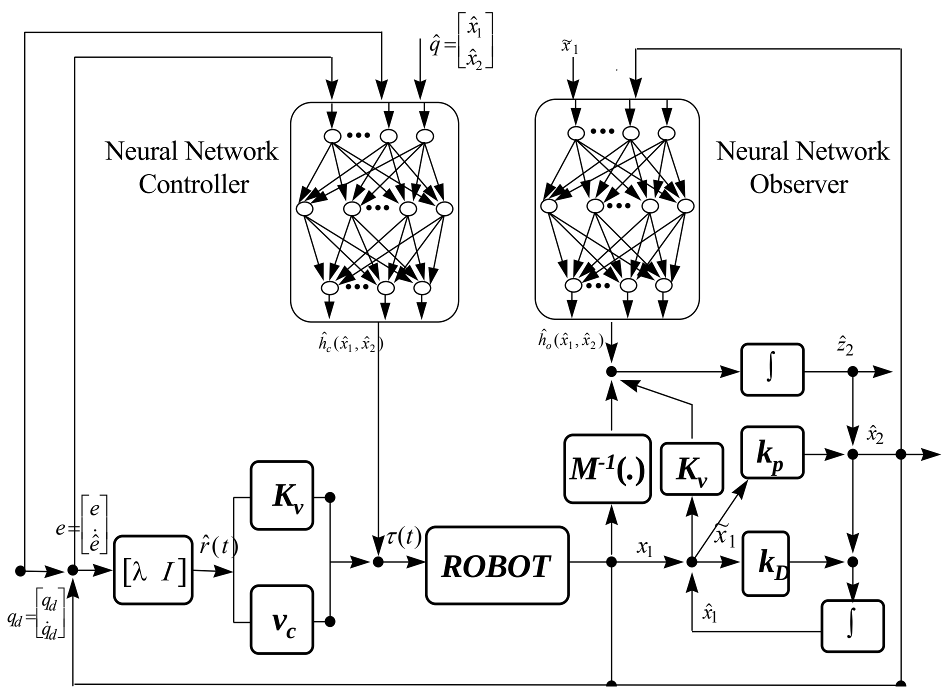
Needed when all states are not measured

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

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

$$\hat{\boldsymbol{r}}(t) = \dot{\hat{\boldsymbol{e}}}(t) + \Lambda \boldsymbol{e}(t)$$

$$\tau(t) = \hat{\boldsymbol{W}}_c^T \sigma_c(\hat{\boldsymbol{x}}_1, \hat{\boldsymbol{x}}_2) + \boldsymbol{K}_v \hat{\boldsymbol{r}}(t) - \boldsymbol{v}_c(t)$$



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

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

NN Control for Discrete Time Systems

dynamics

$$x(k+1) = f(x(k)) + g(x(k))u(k)$$

Gradient descent with momentum

NN Tuning

$$\hat{W}_i(k+1) = \hat{W}_i(k) - \alpha_i \hat{\phi}_i(k) \hat{y}_i^T(k) \quad - \quad \Gamma \left\| I - \alpha_i \hat{\varphi}_i(k) \hat{\varphi}_i^T(k) \right\| \hat{W}_i(k)$$

Extra robust term

Error-based tuning

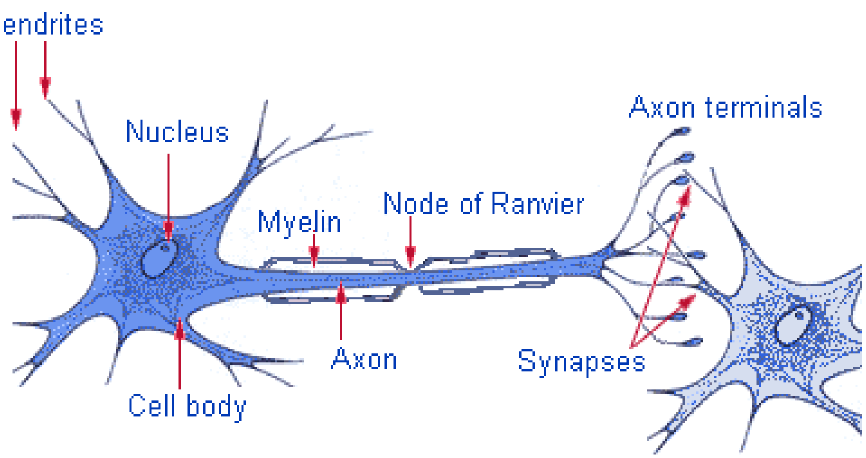
$$\boxed{\hat{y}_i(k) \equiv \hat{W}_i^T(k)\hat{\varphi}_i(k) + K_v r(k), \quad \mathit{for}\ i = 1, \cdots, N-1 \quad \mathit{and} \quad \hat{y}_N(k) \equiv r(k+1), \quad \mathit{for\ last\ layer}}$$

U.S. Patent- Jagannathan, Lewis

Neural Network Properties

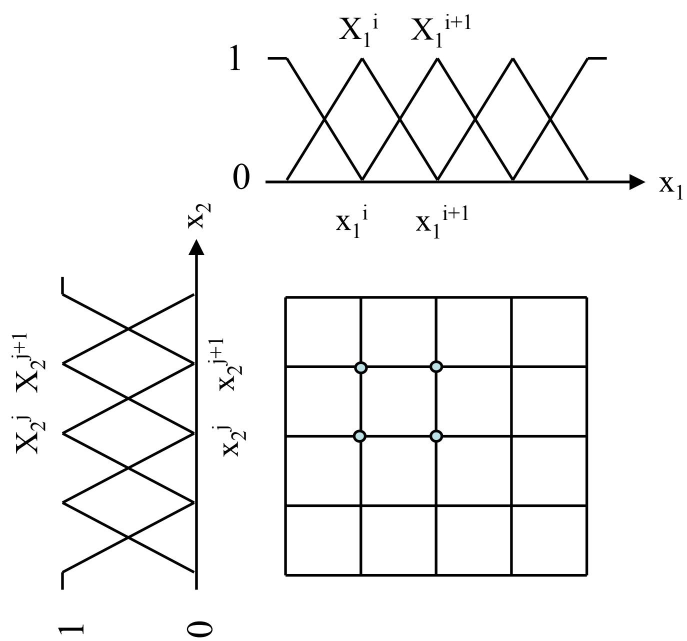
USED

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

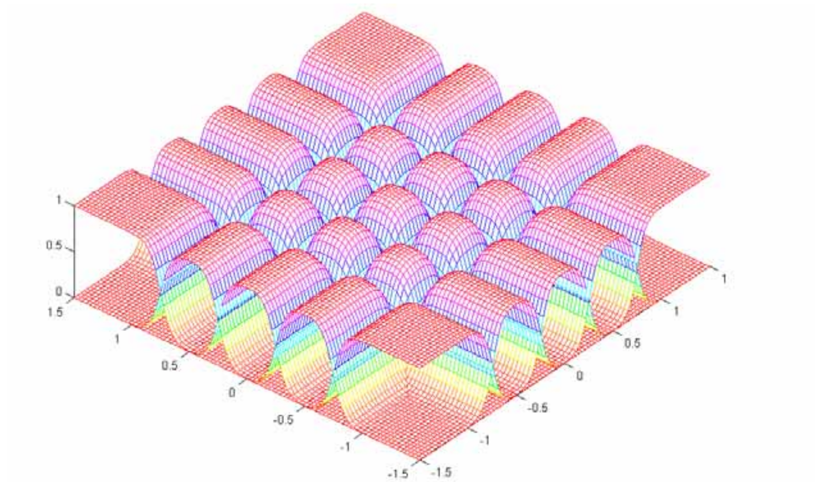


Nervous system cell.
http://www.sirinet.net/~jgjohnso/index.html

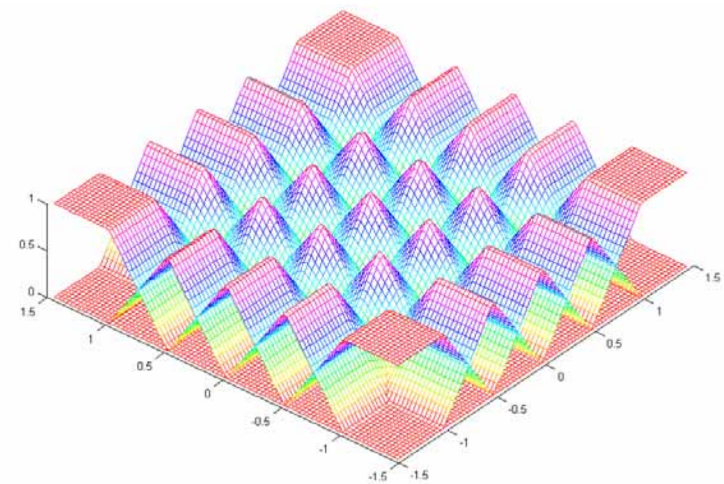
Relation Between Fuzzy Systems and Neural Networks



FL Membership Functions for 2-D Input Vector x

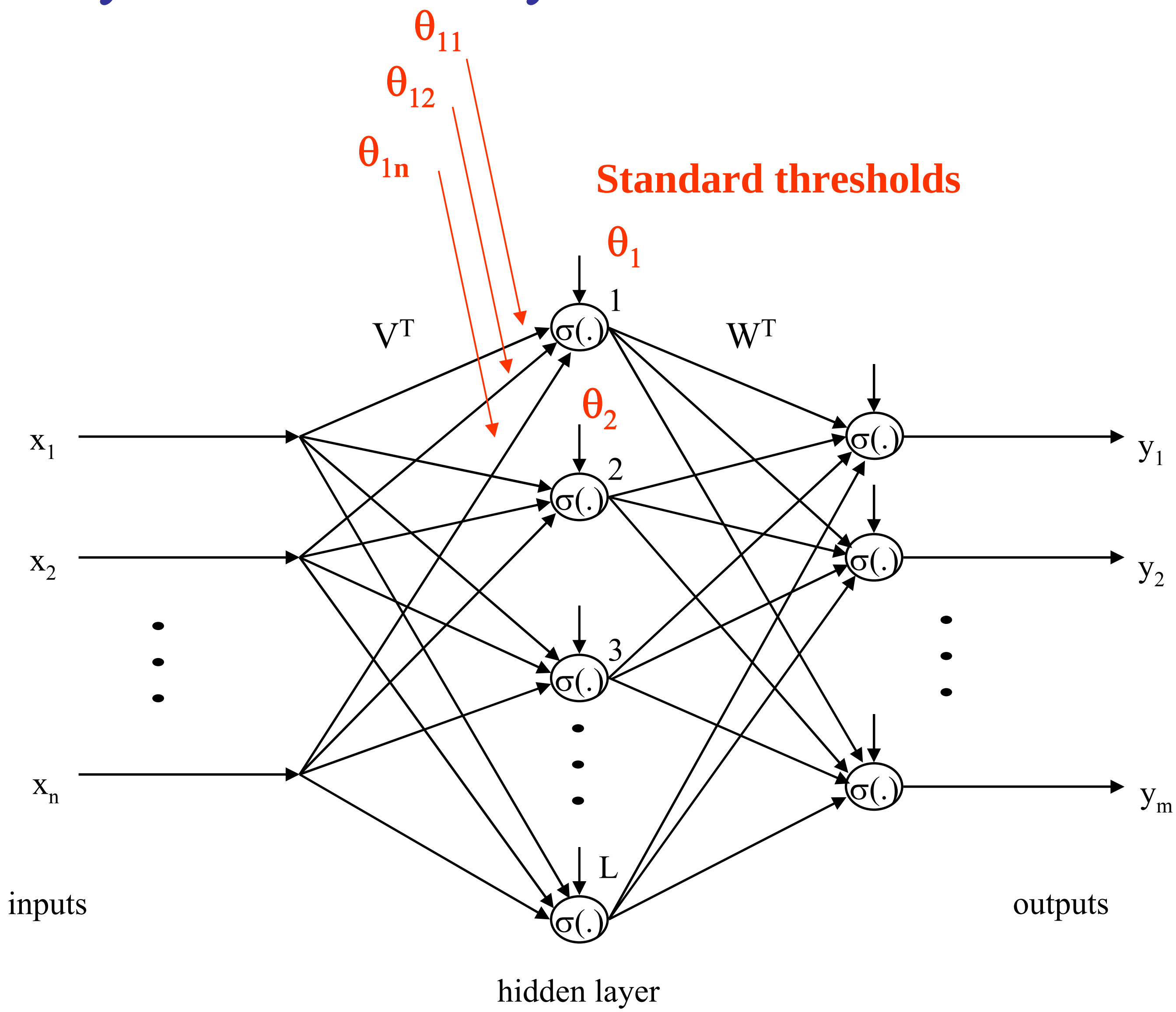


Separable Gaussian activation functions for RBF NN



Separable triangular activation functions for CMAC NN

Two-layer NN as FL System



FL system = NN with VECTOR thresholds

Fuzzy Logic Controllers

Gaussian membership function

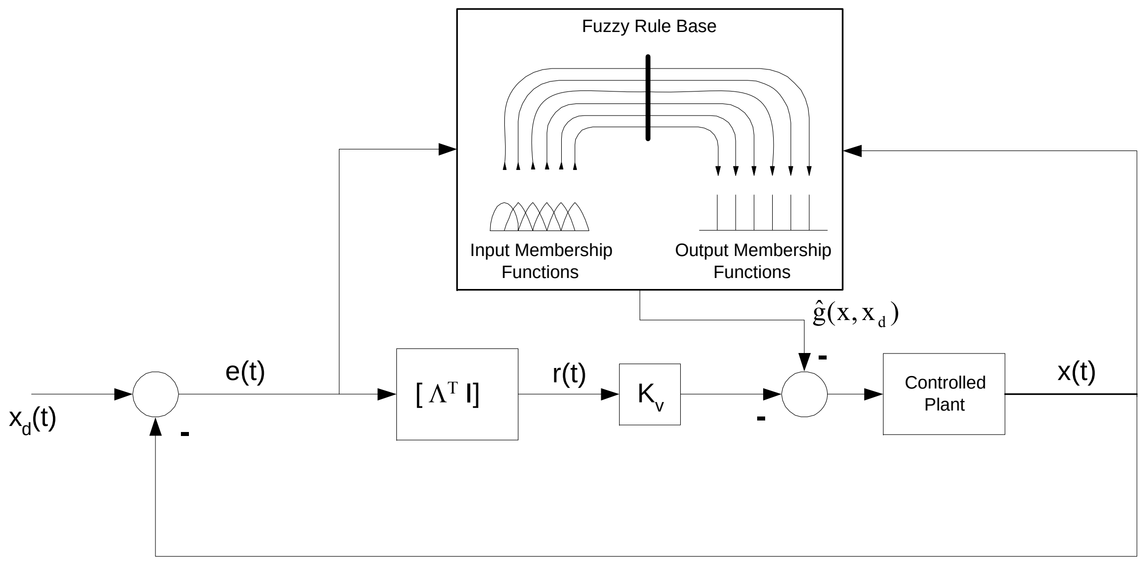
$$\phi_{A_i^l}(z_i, a_i^l, b_i^l) = e^{\left(-a_i^{l^2}(z_i - b_i^l)^2\right)}.$$

Tuning laws

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

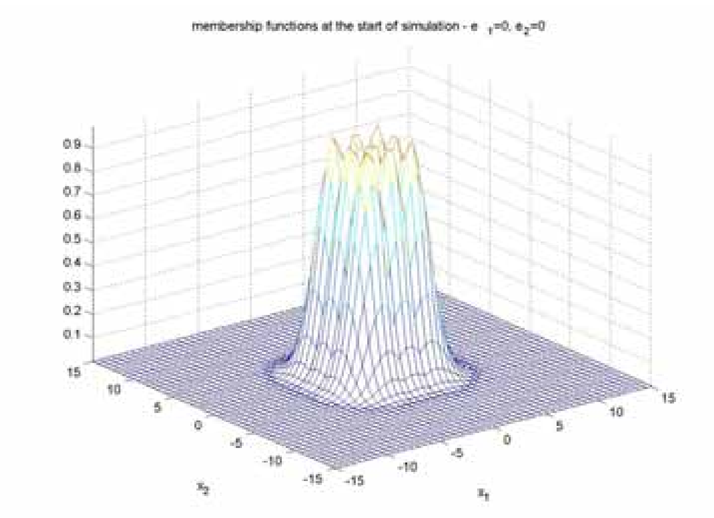
$$\dot{\hat{b}} = K_b B^T \hat{W} r - k_b K_b \hat{b}\|r\|$$

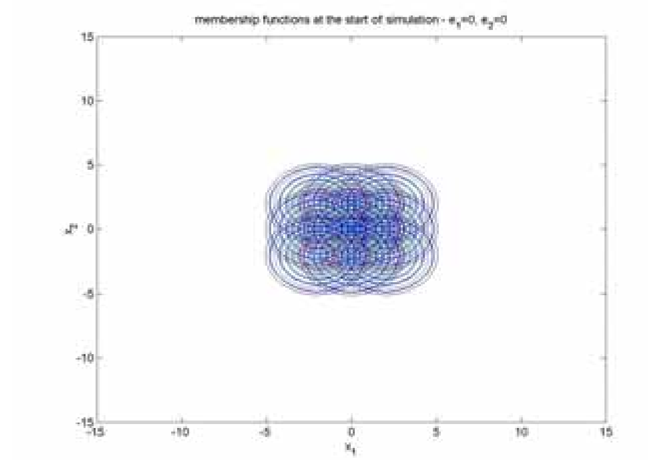
$$\dot{\hat{W}} = K_W(\hat{\Phi} - A\hat{a} - B\hat{b})r^T - k_W K_W \hat{W}\|r\|$$



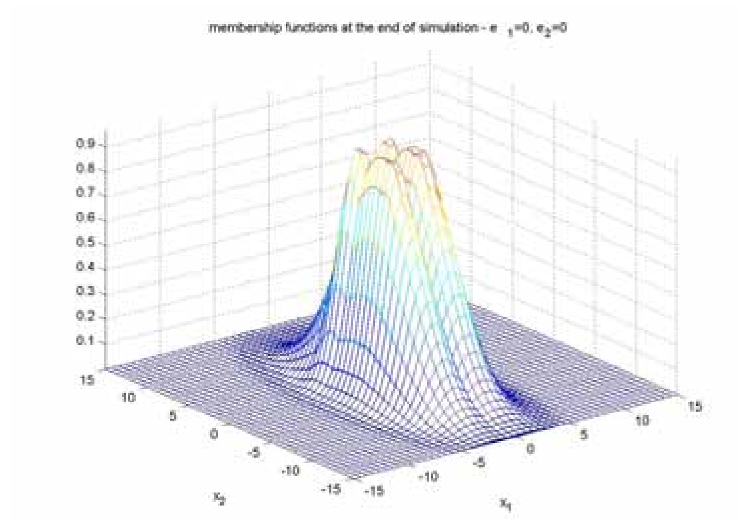
Dynamic Focusing of Awareness

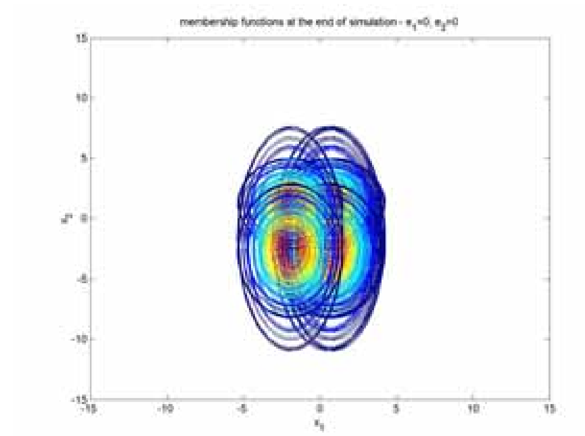
Initial MFs





Final MFs



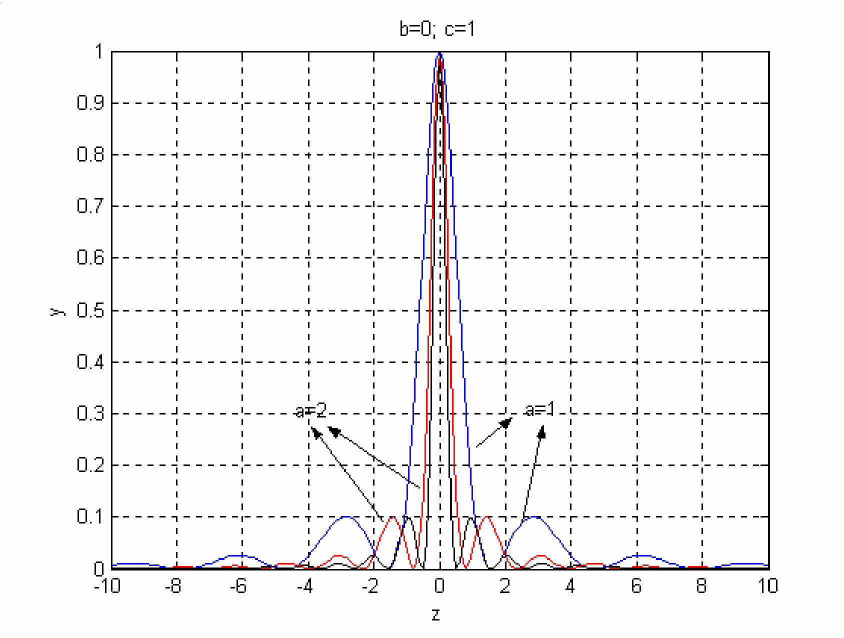


Elastic Fuzzy Logic- c.f. P. Werbos

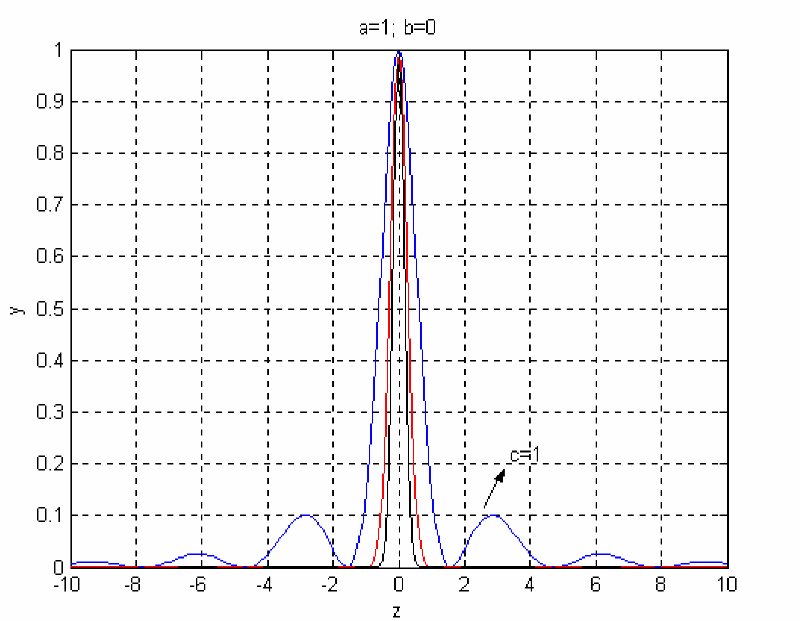
$$\phi(z,a,b,c) = \phi_B(z,a,b)^{c^2}$$

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

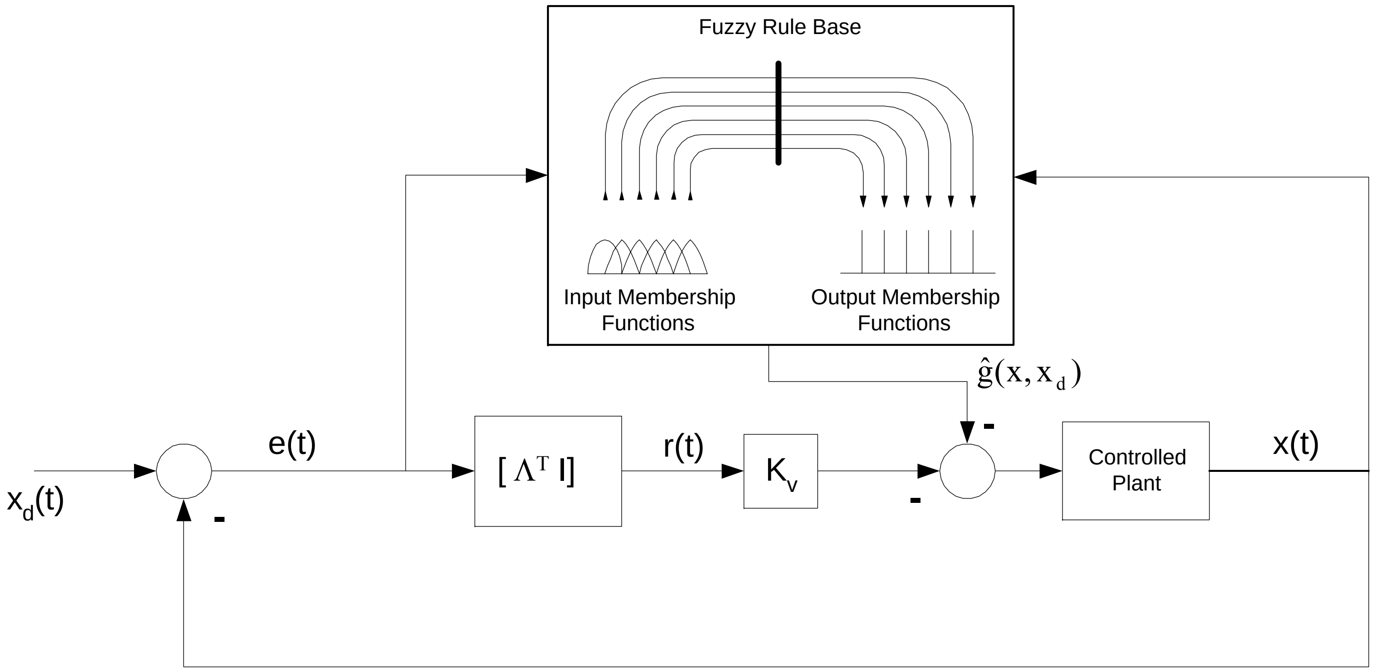
$$\dot{\hat{W}} = K_W(\hat{\Phi} - A\hat{a} - B\hat{b} - C\hat{c})r^T - k_W K_W \hat{W}\|r\|$$

Tune Membership Functions

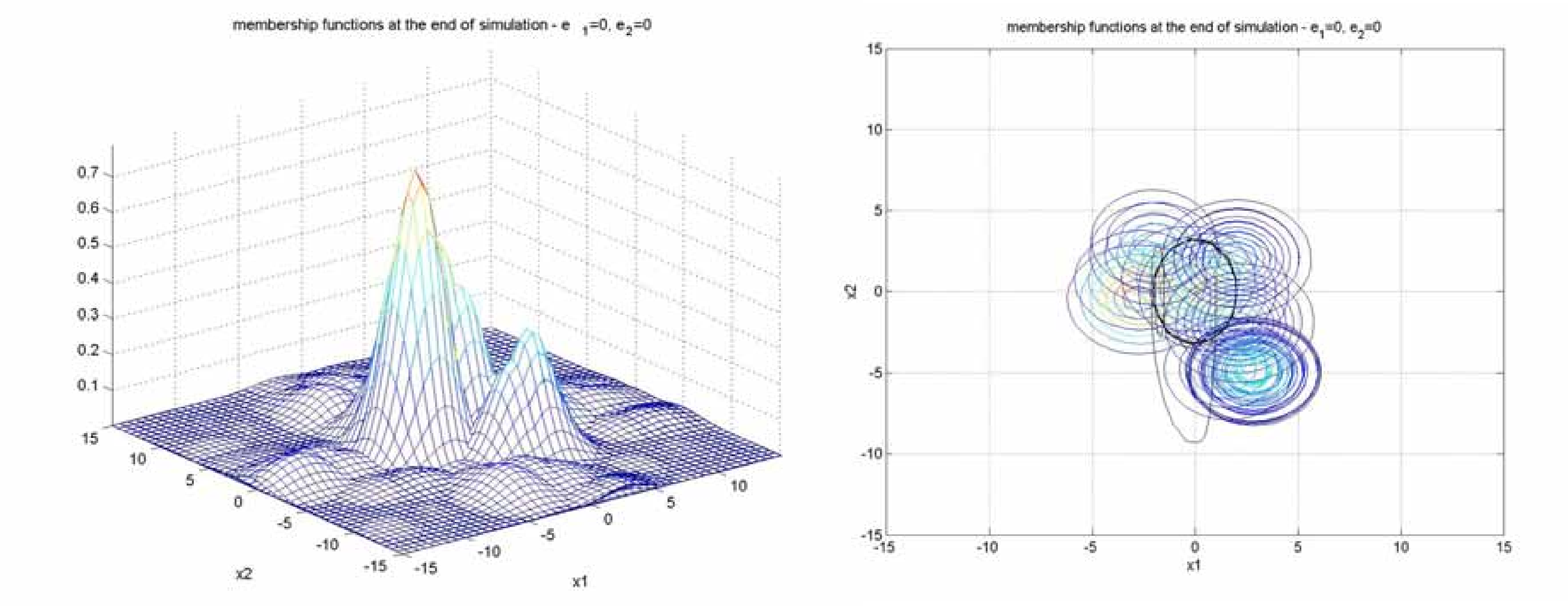
$$\dot{\hat{a}} = K_a A^T \hat{W} r - k_a K_a \hat{a}\|r\|$$

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

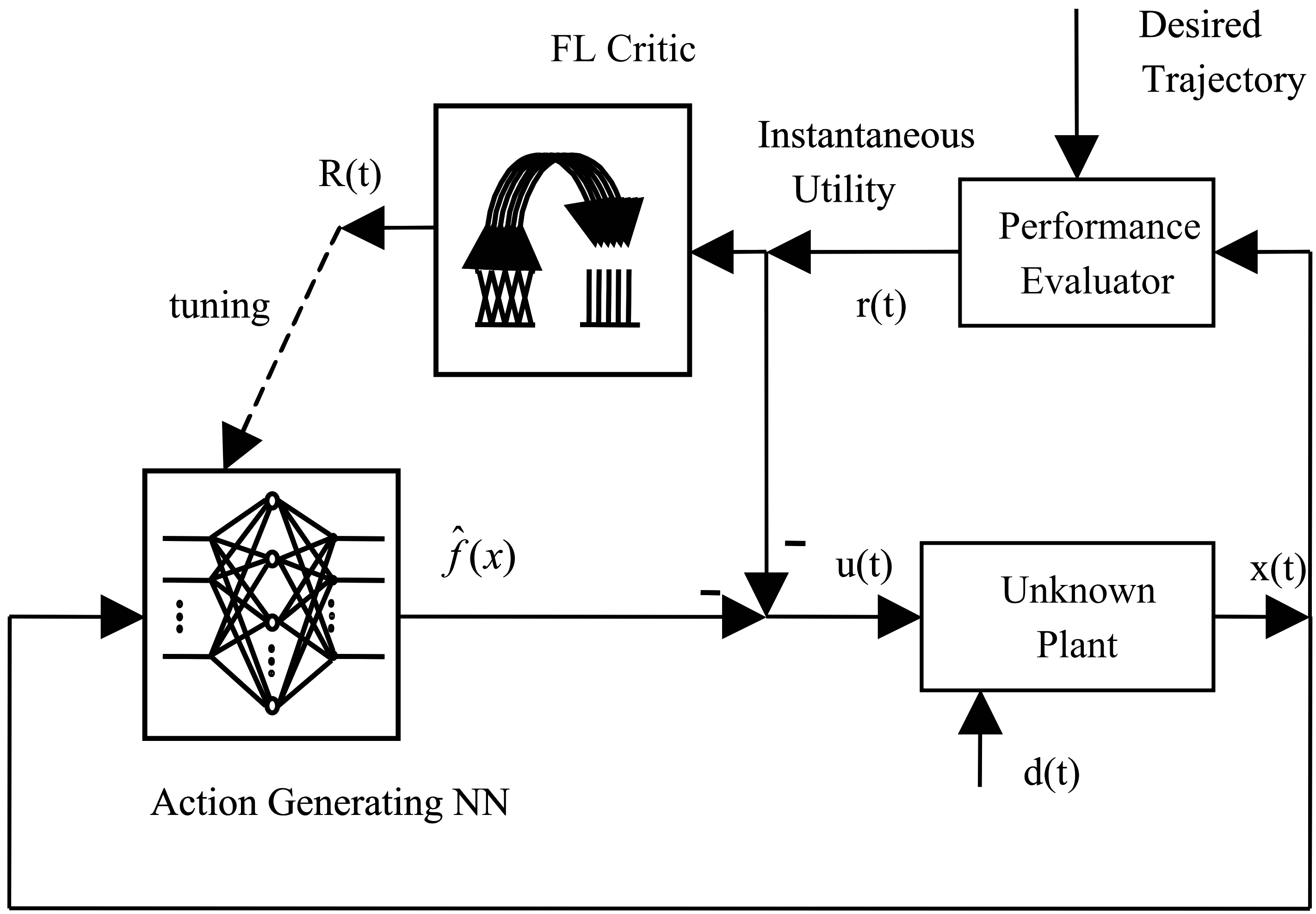
$$\dot{\hat{c}} = K_c C^T \hat{W} r - k_c K_c \hat{c}\|r\|$$



Better Performance



Fuzzy Logic Critic NN controller



Learning FL Critic Controller

Tune Action generating NN (controller)

$$\dot{\hat{W}}_2 = \Gamma_2 \sigma(\chi_2) r^T - \Gamma_2 \sigma(\chi_2) R^T \hat{W}_1^T \mu'(\hat{V}_1^T r)\hat{V}_1^T - \Gamma_2 \hat{W}_2$$

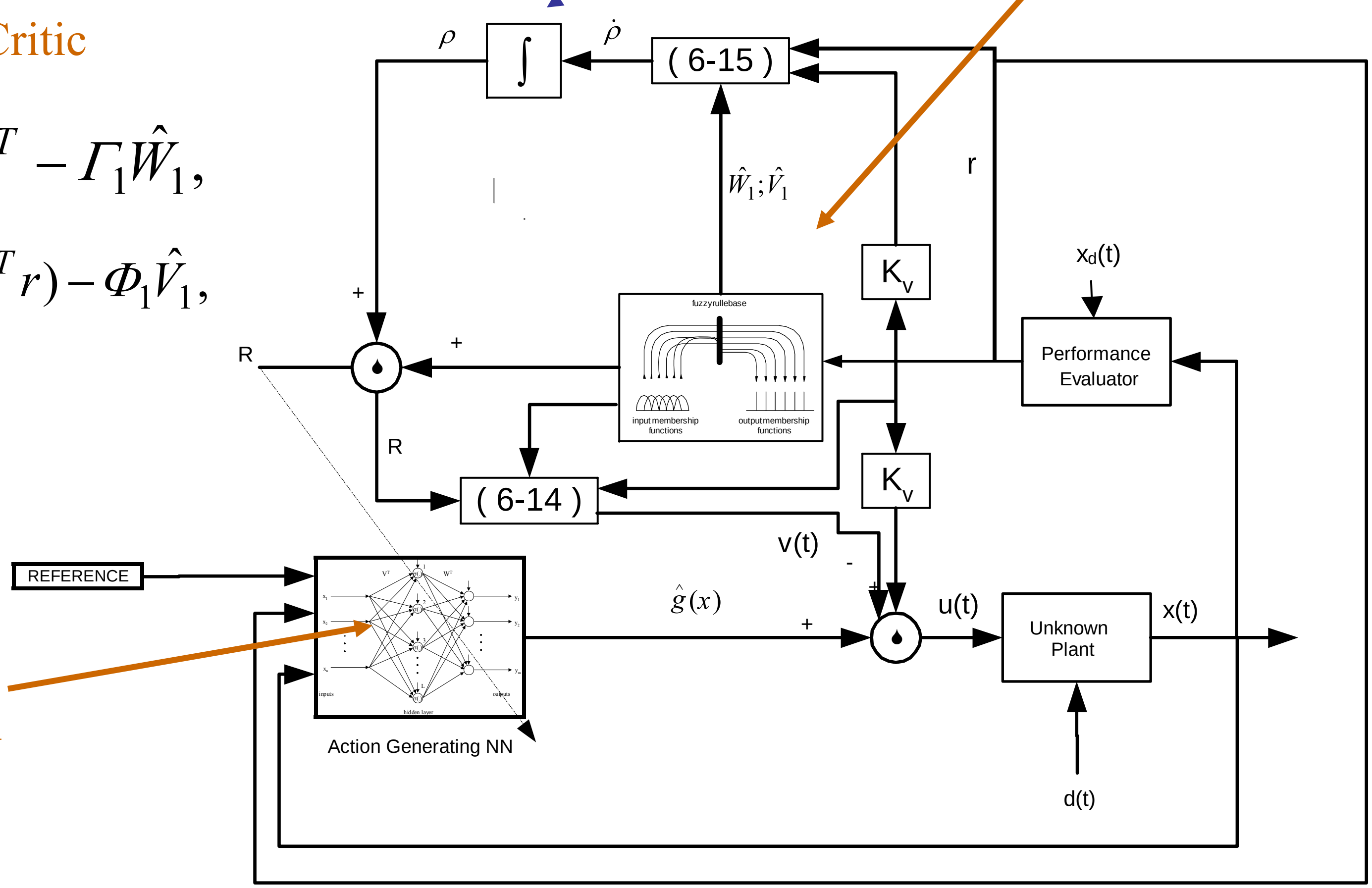
Tune Fuzzy Logic Critic

$$\dot{\hat{W}}_1 = -\mu(\hat{V}_1^T r)R^T - \Gamma_1 \hat{W}_1,$$

$$\dot{\hat{V}}_1 = -rH^T \hat{W}^T \mu'(\hat{V}_1^T r) - \Phi_1 \hat{V}_1,$$

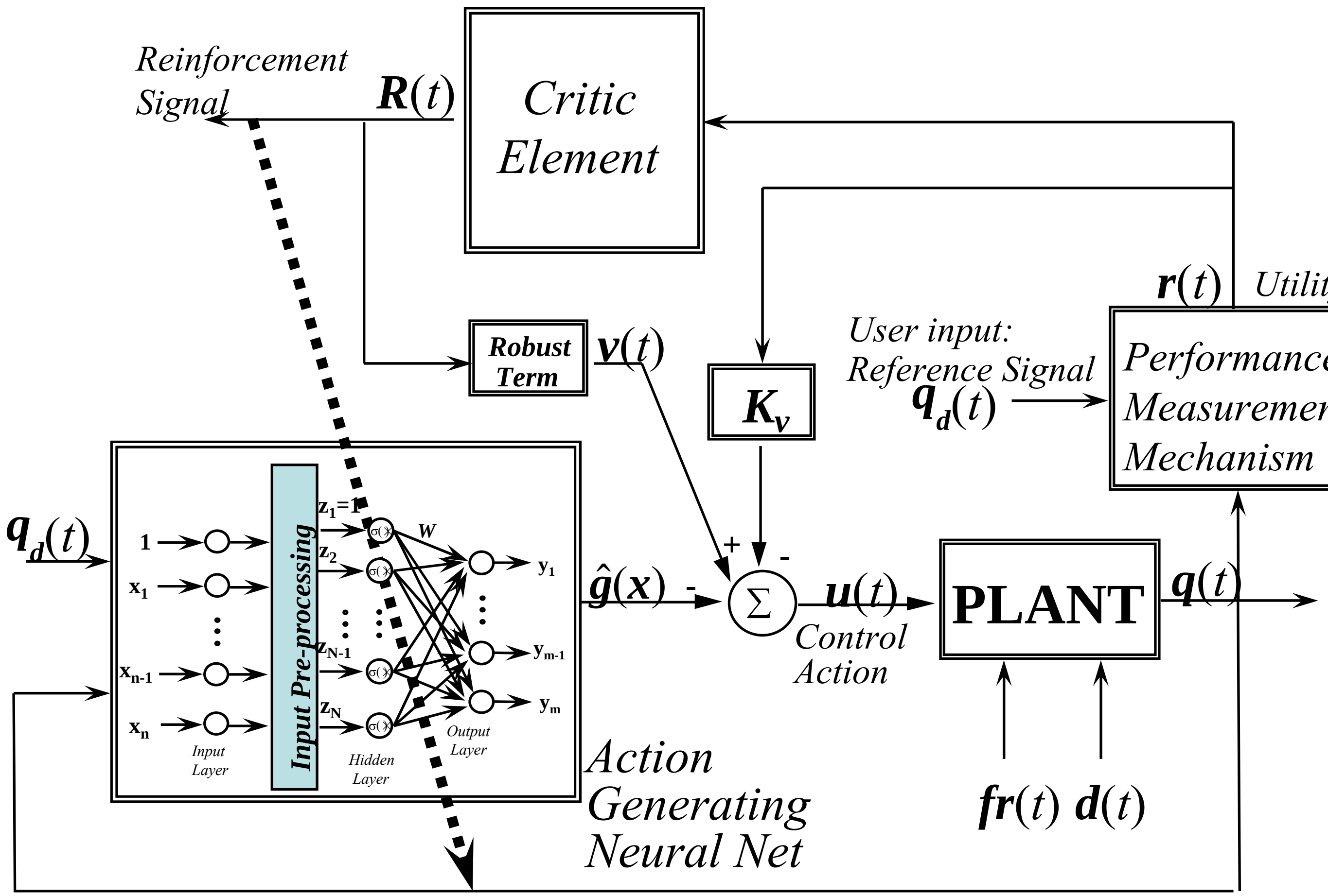
Critic requires MEMORY

FL Critic



Action generating NN

Reinforcement Learning NN Controller



High-Level NN Controllers Need Exotic Lyapunov Fns.

Reinforcement NN control

Simplified critic signal

$$R(t) = \mathrm{sgn}(r(t)) = \pm 1$$

Lyapunov Fn

$$L(t) = \sum_{i=1}^{n} |r_i| \quad + \quad \frac{1}{2} tr(\widetilde{W}^T F^{-1} \widetilde{W})$$

$$\dot{L} = \boldsymbol{sgn}(\mathbf{r})^T \dot{\mathbf{r}} + tr(\widetilde{\mathbf{W}}^T \mathbf{F}^{-1} \dot{\widetilde{\mathbf{W}}})$$

Lyap. Deriv. contains *R(t)* !!

Tuning Law only contains *R(t)*

$$\dot{\hat{W}} = F\sigma(x)R^T - \kappa F \widehat{W}$$

Adaptive Reinforcement Learning

Critic is output of NN #1

$$R = \hat{W}_1^T \cdot \sigma(\chi_1) + \rho,$$

$$L(t) = \ln(1 + e^{-\alpha r(t)}) + \ln(1 + e^{\alpha r(t)}) \quad + \quad \frac{1}{2} tr(\widetilde{W}^T F^{-1} \widetilde{W})$$

$$\dot{L} = \left(\frac{\alpha^+}{1 + e^{-\alpha^+ r(t)}} + \frac{-\alpha^-}{1 + e^{\alpha^- r(t)}} \right) \dot{r}(t) + tr(\widetilde{\mathbf{W}}^T \boldsymbol{F}^{-1} \dot{\widetilde{\mathbf{W}}})$$

Action is output of second NN

$$\hat{g}(x, x_d) = \hat{W}_2^T \sigma(\chi_2)$$

The tuning algorithm treats this as a SINGLE 2-layer NN

$$\dot{\hat{W}}_1 = -\sigma(\chi_1) R^T - \hat{W}_1,$$

$$\dot{\hat{W}}_2 = \Gamma \sigma(\chi_2) \cdot \left(r + V_1 \sigma'(\chi_1)^T \hat{W}_1 R \right)^T - \Gamma \hat{W}_2,$$

Encode Information into the Value Function

Principe- Entropy

Information-Theoretic Learning

$$H(x_0, u(x,t), p(u)) = -\int\int p(x_0, u) \ln p(x_0, u)\, du\, dx_0$$

Renyi's entropy
Corentropy

Brockett- Minimum-Attention Control
awareness & effort (partial derivatives in PM)

$$V(x_0, u) = \int r(x,u)\, dt \quad + \quad \int\int a\left(\frac{\partial u}{\partial t}\right)^2 + b\left(\frac{\partial u}{\partial x}\right)^2 dx\, dt$$

2. Neural Network Solution of Optimal Design Equations

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

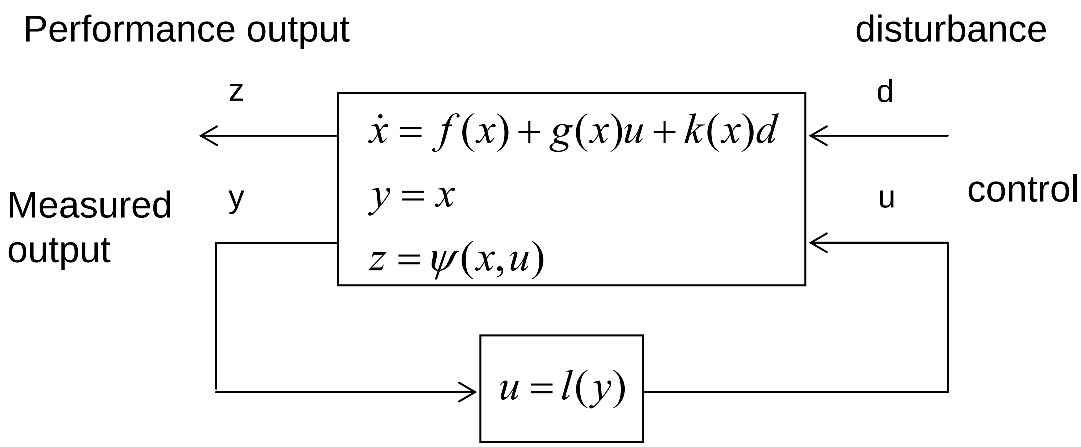
Before-

1. Neural Networks for Feedback Control

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

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 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

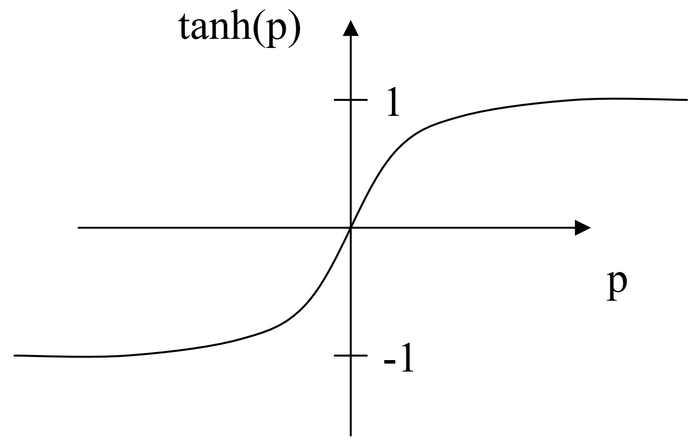
Problems to solve HJI

Beard proposed a successive solution method using Galerkin approx.

Viscosity Solution

Bounded L_2 Gain Problem for Constrained Input Systems

Control constrained by saturation function $\phi(.)$



Encode constraint into Value function

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

$$\|u\|_q^{\;2} = 2\int_0^u \phi^{-T}(v)\,dv$$

(Used by Lyshevsky for H_2 control)

This is a quasi-norm

Weaker than a norm –
homogeneity property is replaced by the weaker symmetry property $\|x\|_q = \|-x\|_q$

Hamiltonian

$$H(x, V_x, u, d) \equiv \frac{\partial V}{\partial x}^T \left(f + gu + kd\right) + h^T h + 2\int_0^u \phi^{-T}(\nu)d\nu - \gamma^2 d^T d$$

Stationarity conditions

$$0 = \frac{\partial H}{\partial u} = g^T V_x + 2\phi^{-1}(u)$$

Leibniz's Formula

$$0 = \frac{\partial H}{\partial d} = k^T V_x - 2\gamma^2 d$$

Solve for u(t)

Optimal inputs

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

Note u(t) is bounded!

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

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 →

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

Inner loop update

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

go to 2.

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

Outer loop update

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

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--- c.f. Howard

Results for this Algorithm

The algorithm converges to $V^*(\Omega_0), \Omega_0, u^*(\Omega_0), d^*(\Omega_0)$

the optimal solution on the RAS Ω_0

Sometimes the algorithm converges to the optimal HJI solution V^*, Ω^*, u^*, d^*

For this to occur it is required that $\Omega^* \subseteq \Omega_0$

For every iteration on the disturbance d^i one has

$V^i{}_j \le V^{i+1}{}_j$ the value function increases

$\Omega^i{}_j \supseteq \Omega^{i+1}{}_j$ the RAS decreases

For every iteration on the control u_j one has

$V^\infty{}_j \ge V^\infty{}_{j+1}$ the value function decreases

$\Omega^\infty{}_j \subseteq \Omega^\infty{}_{j+1}$ the RAS does not decrease

Problem- Cannot solve the Value Equation!

Neural Network Approximation for Computational Technique

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

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

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)***

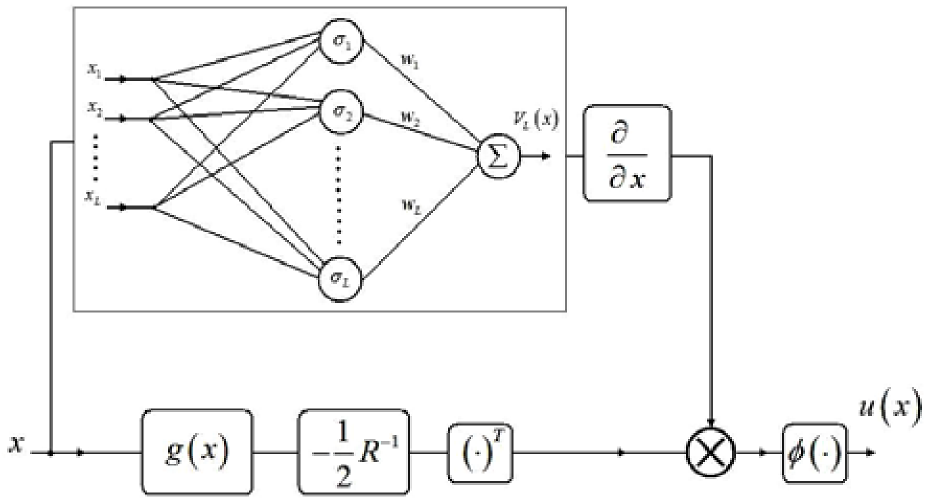
Neural Network Feedback Controller

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



Example: Linear system

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & -0.5 \\ 1 & 1.5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \end{bmatrix} u, \qquad |u| \le 1$$

$$V_{15}(x_1, x_2) = w_1 x_1^2 + w_2 x_2^2 + w_3 x_1 x_2 + w_4 x_1^4 +$$

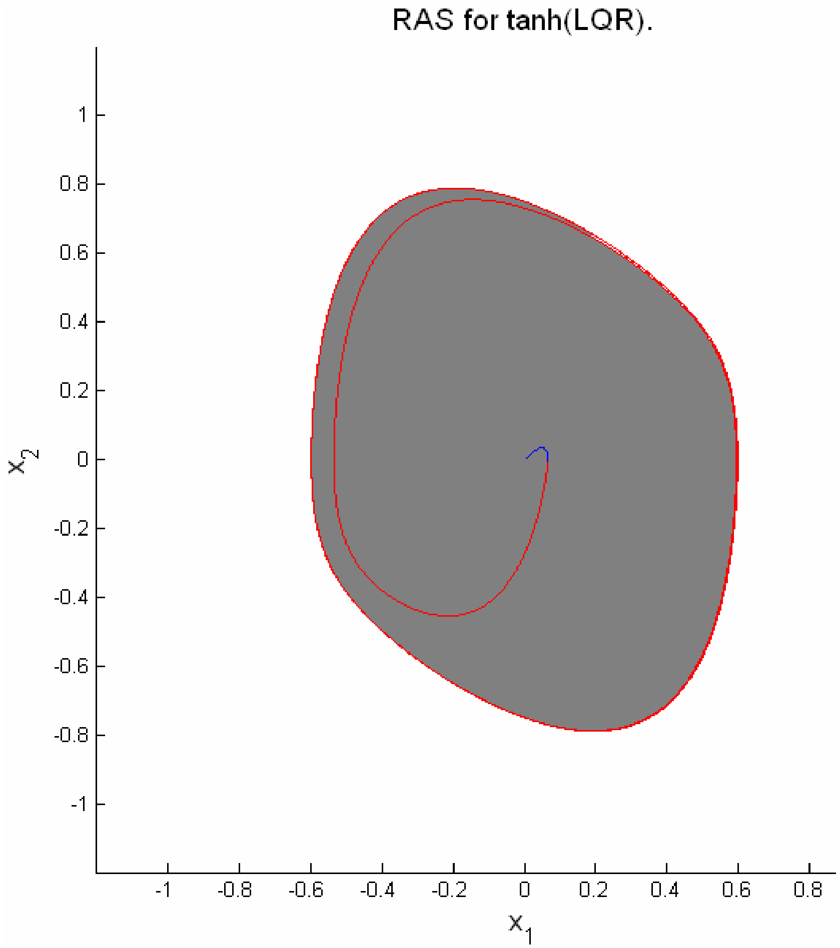
$$w_5 x_2^4 + w_6 x_1^3 x_2 + w_7 x_1^2 x_2^2 + w_8 x_1 x_2^3 + w_9 x_1^6 + w_{10} x_2^6$$

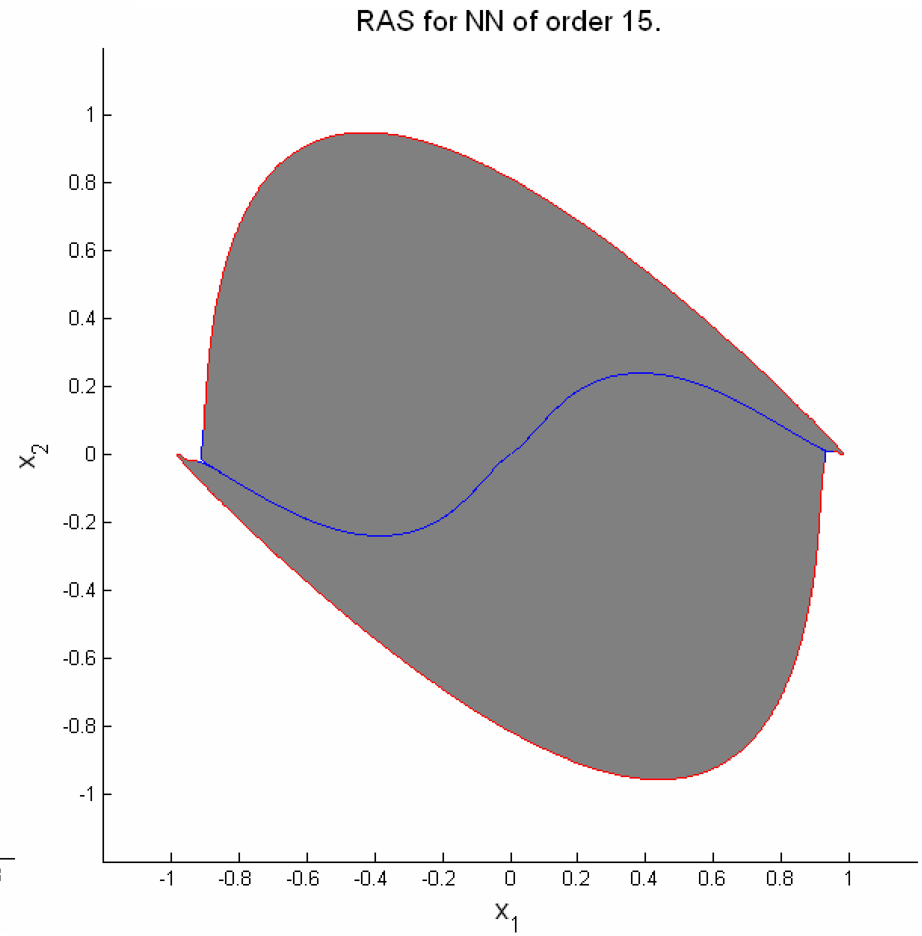
$$w_{11} x_1^5 x_2 + w_{12} x_1^4 x_2^2 + w_{13} x_1^3 x_2^3 + w_{14} x_1^2 x_2^4 + w_{15} x_1 x_2^5$$

Activation functions = even polynomial basis up to order 6

Initial Gain found by LQR

Optimal NN solution

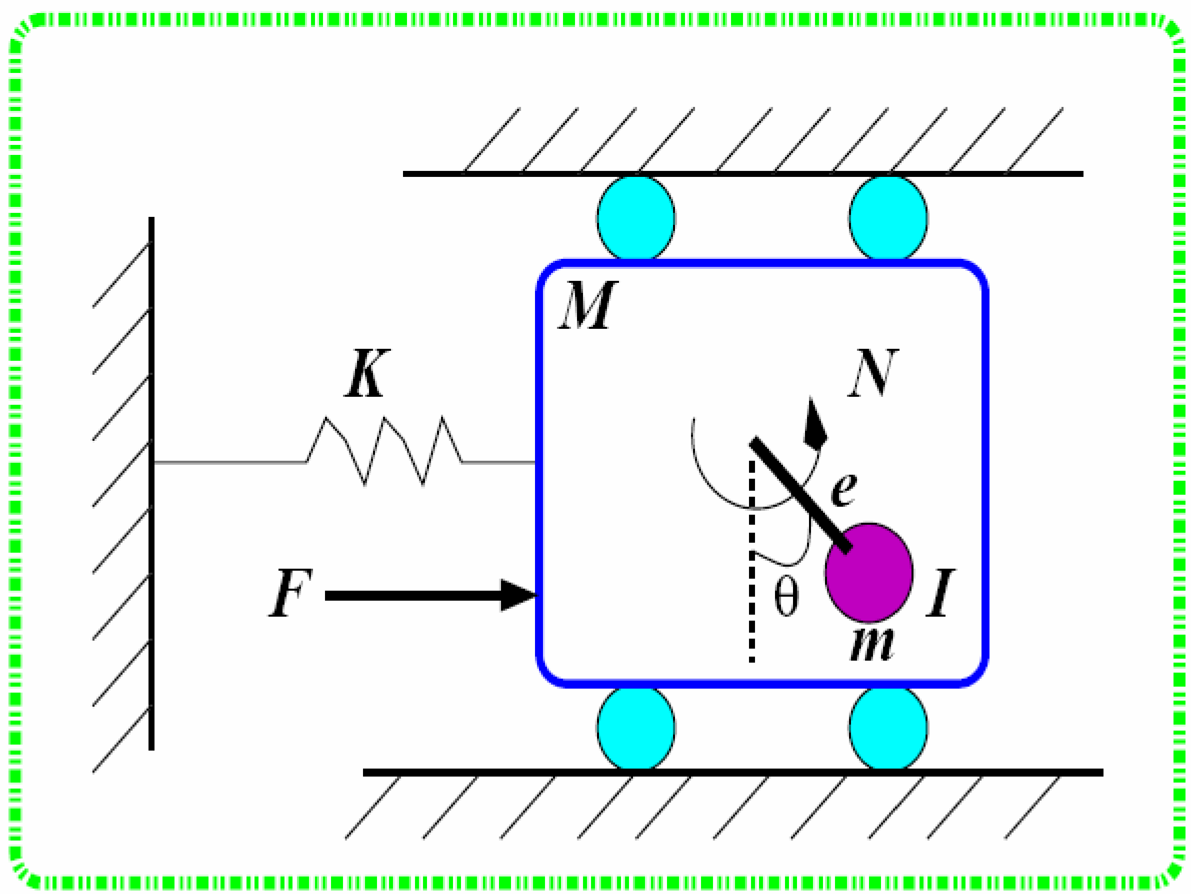




RAS found by integrating $\dot{x} = -f(x)$

That is, reverse time $dt = -d\tau$

Rotational-Translational Actuator Benchmark Problem



F is a disturbance

Control input is torque N

Rotational-Translational Actuator Benchmark Problem

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

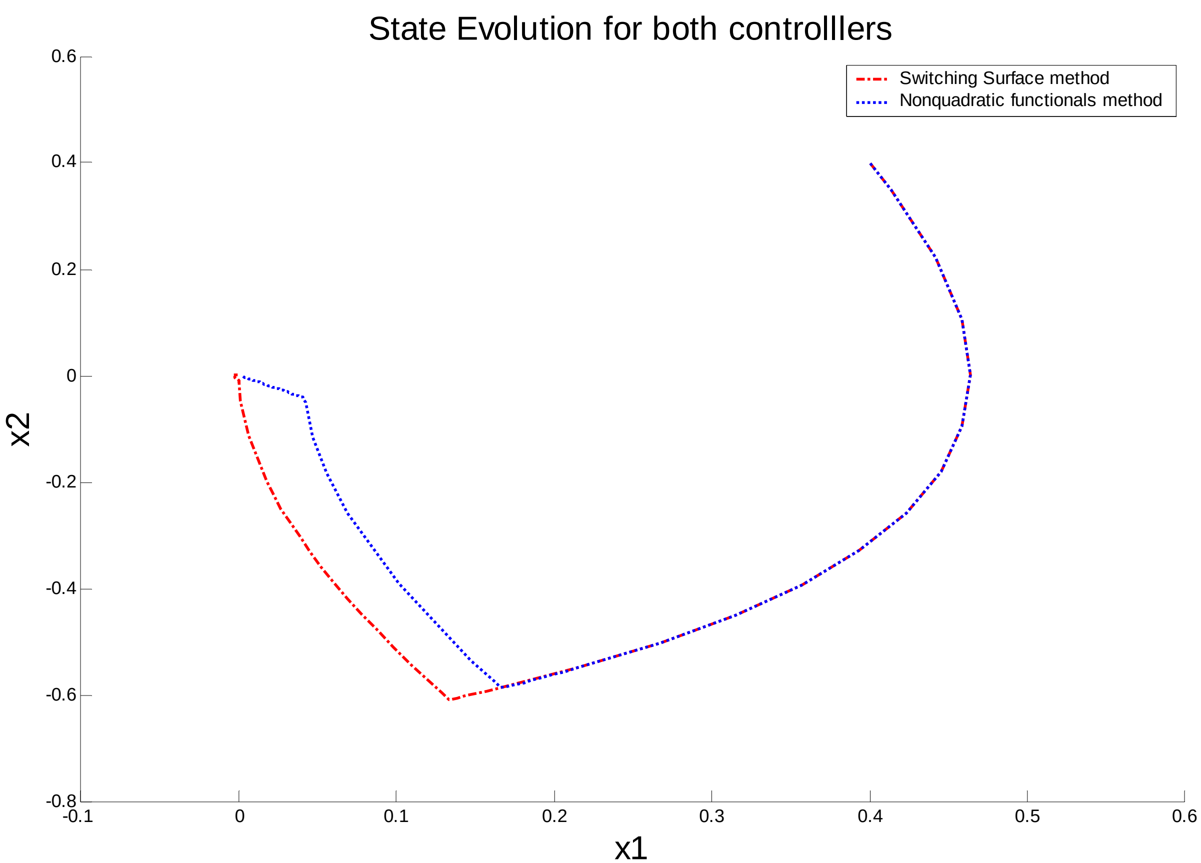
$$f(x) = \begin{bmatrix} x_2 \\ \dfrac{-x_1 + \varepsilon x_4^2 \sin x_3}{1 - \varepsilon^2 \cos^2 x_3} \\ x_4 \\ \dfrac{\varepsilon \cos x_3 (x_1 - \varepsilon x_4^2 \sin x_3)}{1 - \varepsilon^2 \cos^2 x_3} \end{bmatrix}, \quad g(x) = \begin{bmatrix} 0 \\ \dfrac{-\varepsilon \cos x_3}{1 - \varepsilon^2 \cos^2 x_3} \\ 0 \\ \dfrac{1}{1 - \varepsilon^2 \cos^2 x_3} \end{bmatrix}$$

$$\varepsilon = 0.2$$

Minimum-Time Control

Encode into Valua Function

$$V = \int_0^{\infty} \left[\tanh(x^T Q x) + 2 \int_0^{u} \left(\phi^{-1}(\mu) \right)^T R d\mu \right] dt$$



3. Approximate Dynamic Programming

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

Before-

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

IEEE Trans. Neural Networks

Special Issue on Neural Networks for Feedback Control

Lewis, Wunsch, Prokhorov, Jie Huang, Parisini

Due date 1 December

Bring together:
Feedback control system community
Approximate Dynamic Programming community
Neural Network community

Discrete-Time Systems

$$x_{k+1} = f(x_k, u_k) \qquad V(x_k) = \sum_{i=k}^{N} \gamma^{i-k} r(x_k, u_k)$$

Value in difference form -

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

Recursive form
Consistency equation

Howard Policy Iteration- Iterate the following until convergence

1. Find the value for the prescribed policy

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

solve completely

2. Policy improvement

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

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

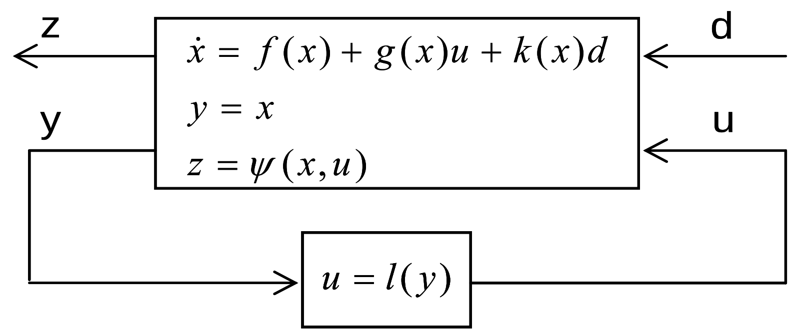
Gradients $\dfrac{\partial Q}{\partial x}, \quad \dfrac{\partial Q}{\partial u}$

Action NN to approximate the Control

Bertsekas- Neurodynamic Programming

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

Continuous-Time Systems



$$V(x(t)) = \int_t^T r(x,u,d)\, dt$$

Value in differential form -

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

Consistency equation

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

$$u^*(x(t)) = -\frac{1}{2} g^T(x) \frac{\partial V^*}{\partial x} \qquad d^*(x(t)) = \frac{1}{2\gamma^2} k^T(x) \frac{\partial V^*}{\partial x}$$

HJB equation

$$0 = \left(\frac{dV^*}{dx}\right)^T f + h^T h - \frac{1}{4}\left(\frac{dV^*}{dx}\right)^T gg^T \frac{dV^*}{dx} + \frac{1}{4\gamma^2}\left(\frac{dV^*}{dx}\right)^T kk^T \frac{dV^*}{dx}$$

Continuous Time Policy Iteration

Select a stabilizing initial control

1. Outer loop- update control

Initial disturbance set to zero

Abu-Khalaf and Lewis- H inf

Saridis – H_2

2. Inner loop- update disturbance

Solve Lyapunov equation

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

Inner loop disturbance update

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

go to 2.

Until convergence

Outer loop update

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

Go to 1.

Until convergence

c.f. Howard work in DT Systems

Neural Network Approximation of Value Function

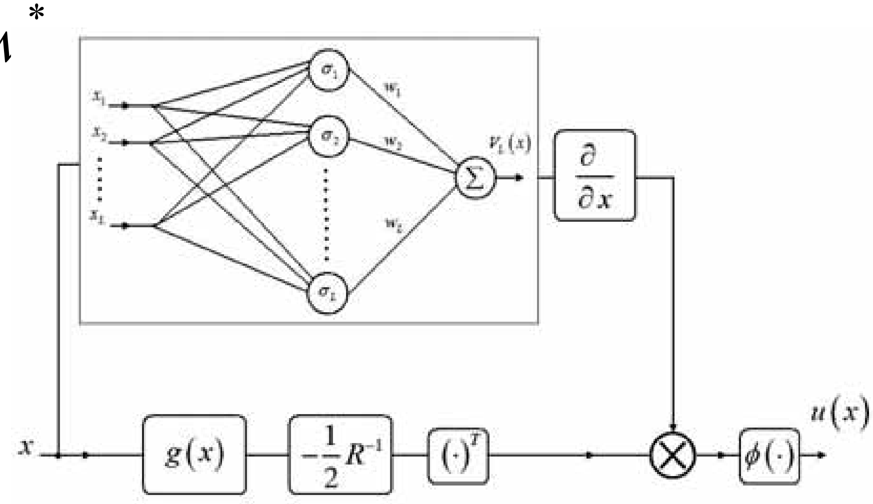
$$\hat{V}(x, w^i{}_j) = w_j^{i\,T} \sigma(x)$$

Lyapunov equation becomes

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

Control action

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



CT Approx Policy Iteration
Abu-Khalaf & Lewis

Nearly optimal FB control
Off-line tuning
Known dynamics

CT Nearly Optimal NN feedback

Continuous-time adaptive critic

Abu-Khalaf & Lewis
(c.f. Doya)

On-line tuning

Critic NN $$V(x) = w^T \sigma(x)$$

Hamiltonian (CT consistency check)

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

residual eq error

$$\delta = \frac{dw^T \sigma}{dt} + r(x,u) = w^T \nabla \sigma(x) \dot{x} + r(x,u) = w^T \nabla \sigma(x) f(x,u) + r(x,u)$$

Target value

$$E = \tfrac{1}{2} |\delta|^2$$

$$\frac{\partial E}{\partial w} = \delta(t) \frac{\partial \delta}{\partial w} = \delta(t)\, \nabla \sigma(x) f(x,u)$$ gradient

$$\dot{w} = -\alpha\, \nabla \sigma(x) f(x,u)\, \delta$$ Update weights using, e.g., gradient descent
Or RLS

Action NN

Critic weights

$$Y_2 = -\tfrac{1}{2} R^{-1} g^T(x) \nabla \sigma^T(x) w = \overline{\phi}^T w$$

Target action

$$\overline{\phi}^T(x) = -\tfrac{1}{2} R^{-1} g^T(x) \nabla \sigma^T(x)$$

Activation fns depend on system dynamics

$$\hat{Y}_2 = \overline{\phi}^T(x) v$$

Action NN

$$e_2(x) = \hat{Y}_2 - Y_2 = -\tfrac{1}{2} R^{-1} g^T(x) \nabla \sigma^T(x)\,[v - w] = \overline{\phi}^T(x)[v - w]$$

$$\dot{v} = -\beta\, \overline{\phi}(x) e_2(x)$$

update weights by gradient descent

Alternative, simply set $u(x) = Y_2 = -\tfrac{1}{2} R^{-1} g^T(x) \nabla \sigma^T(x) w = \overline{\phi}^T w$

Does not work- proof development so far indicates that
critic NN must be tuned faster than action NN
i.e. $\alpha > \beta$

c.f. Bradtke & Barto DT Q learning work

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

Sampled data systems

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

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

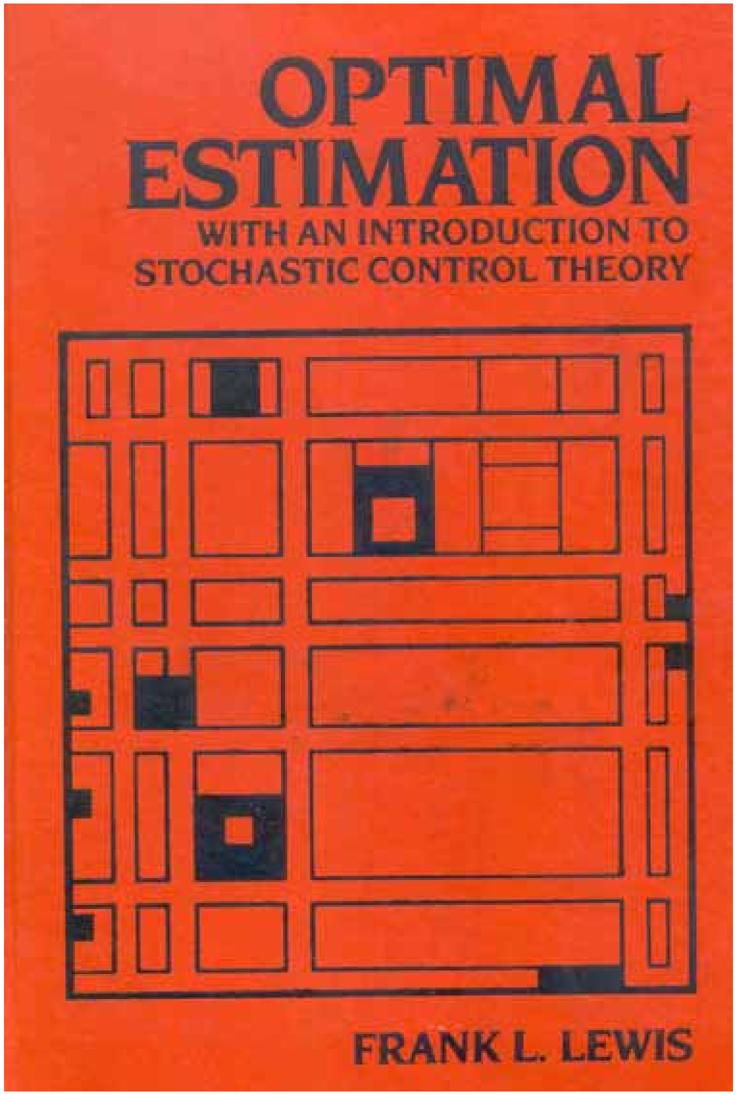
Baird’s Advantage function

This is not in standard DT form

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

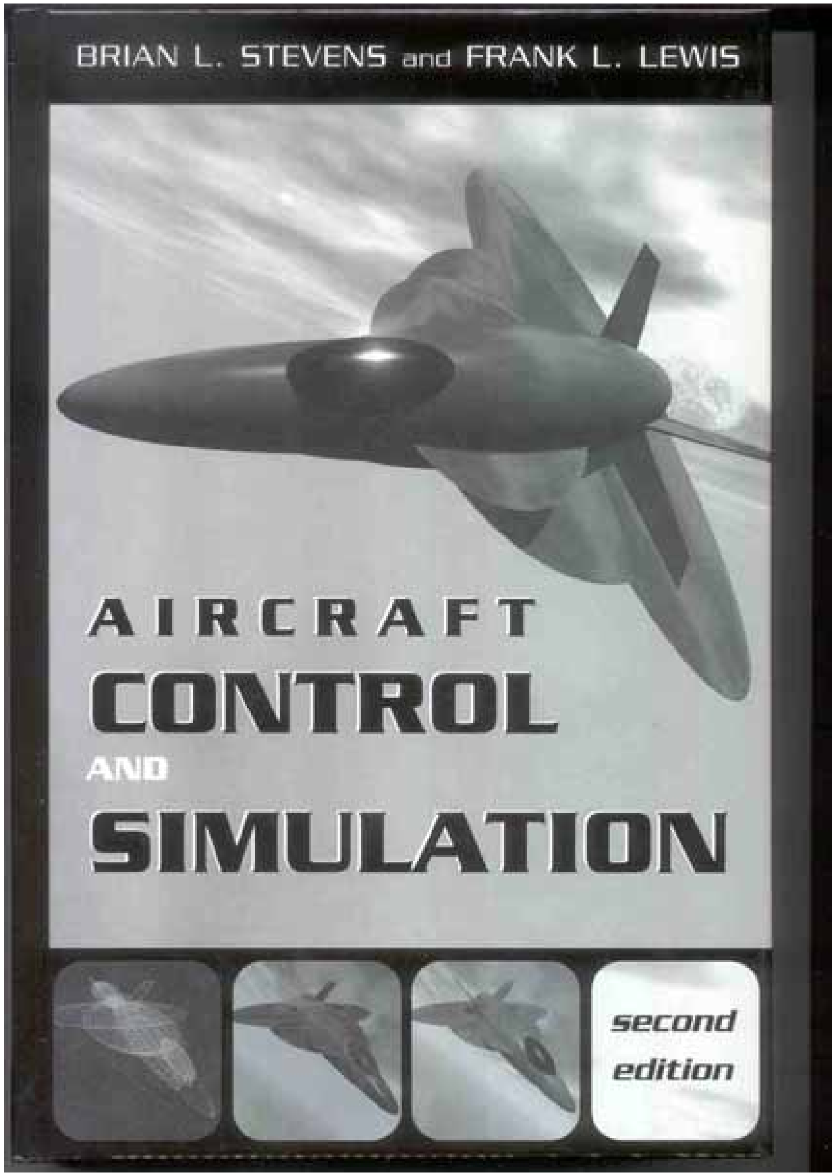
For More Information

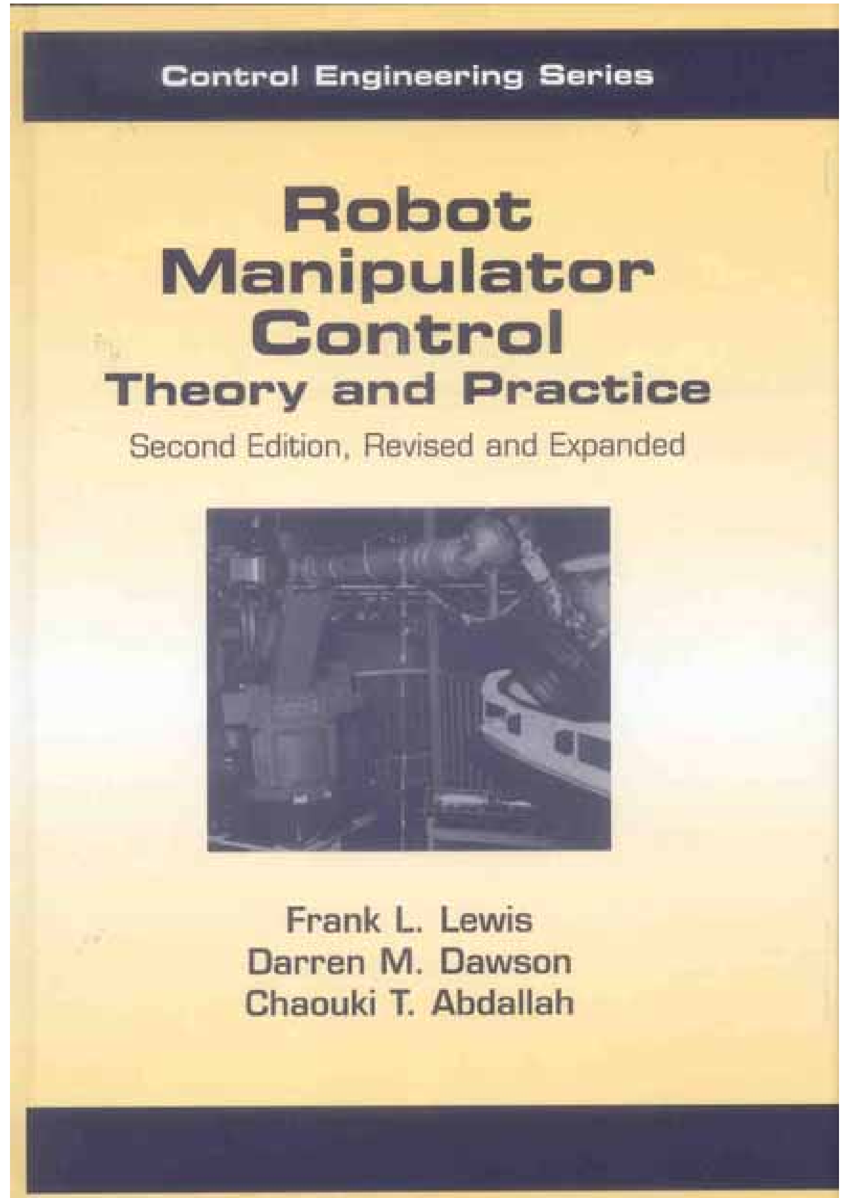
Journal papers on http://arri.uta.edu/acs

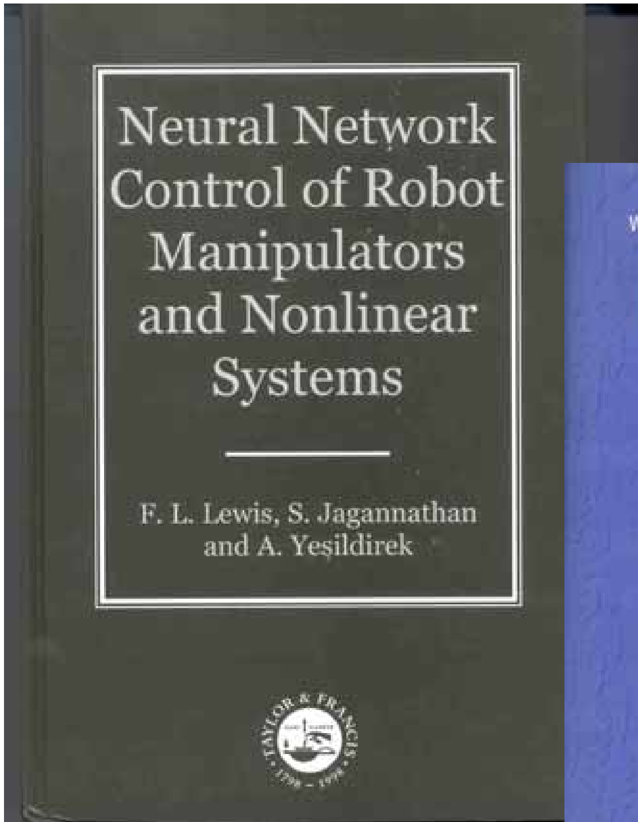


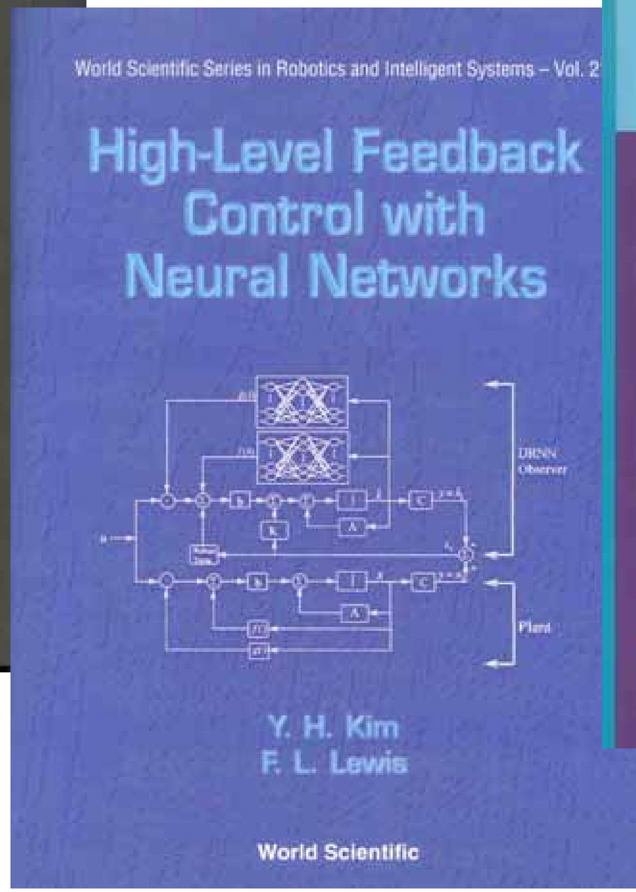
Optimal Control

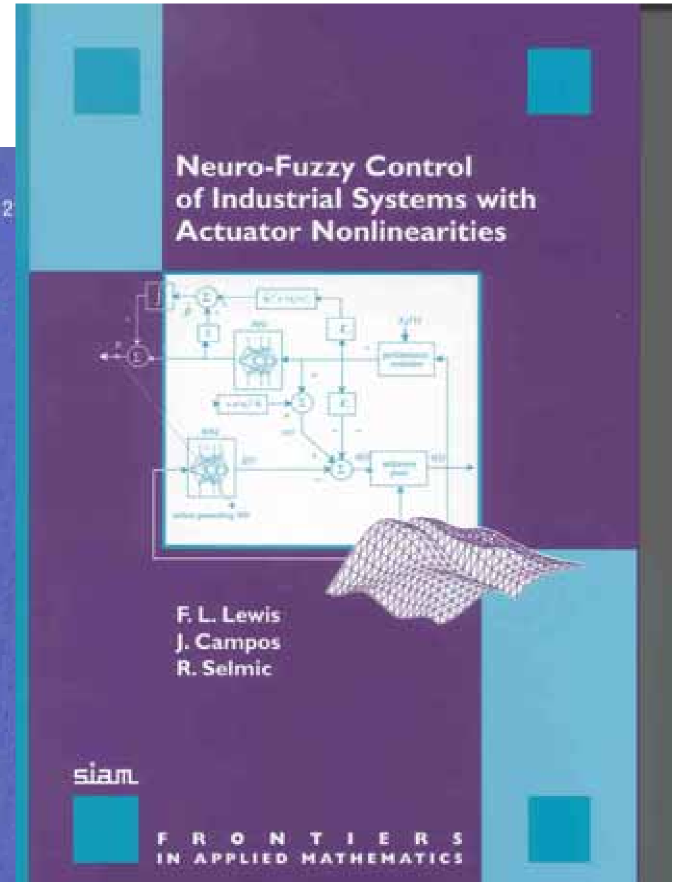
Lewis & Syrmos 1995


BRIAN L. STEVENS and FRANK L. LEWIS
AIRCRAFT
CONTROL
AND
SIMULATION
second
edition


Control Engineering Series
Robot
Manipulator
Control
Theory and Practice
Second Edition, Revised and Expanded
Frank L. Lewis
Darren M. Dawson
Chaouki T. Abdallah







In Progress: M. Abu-Khalaf, Jie Huang, F.L. Lewis
Nearly Optimal Control by HJ Equation Solution Using Neural Networks

Theorem 1.

Necessary and Sufficient Conditions for H-infinity Static OPFB Control

Assume that Q>0, then system (1) is output-feedback stabilizable with L_2 gain bounded by γ If and only if:

i. (A, C) is detectable

ii. There exist matrices *K** and *L* such that

$$K^* C = R^{-1}(B^T P + L)$$

where $P>0$, $P^T = P$, is a solution of

$$PA + A^T P + Q + \frac{1}{\gamma^2} PDD^T P - PBR^{-1}B^T P + L^T R^{-1} L = 0$$

ONLY TWO COUPLED EQUATIONS

c.f. results by Kucera and De Souza

Note there is an (A,B) stabilizability condition hidden in the existence of Solution to the Riccati eq.

Solution Algorithm 1- c.f. Geromel

1. Initialize:
   Set *n=0,* $L_0 = 0$, and select γ, Q, R

2. *n-th* iteration:
   solve for P_n in the ARE

$$P_n A + A^T P_n + Q + \frac{1}{\gamma^2} P_n D D^T P_n - P_n B R^{-1} B^T P_n + L_n{}^T R^{-1} L_n = 0$$

Evaluate gain and update *L*

$$K_{n+1} = R^{-1} (B^T P_n + L) C^T (C C^T)^{-1}$$

$$L_{n+1} = R K_{n+1} C - B^T P_n$$

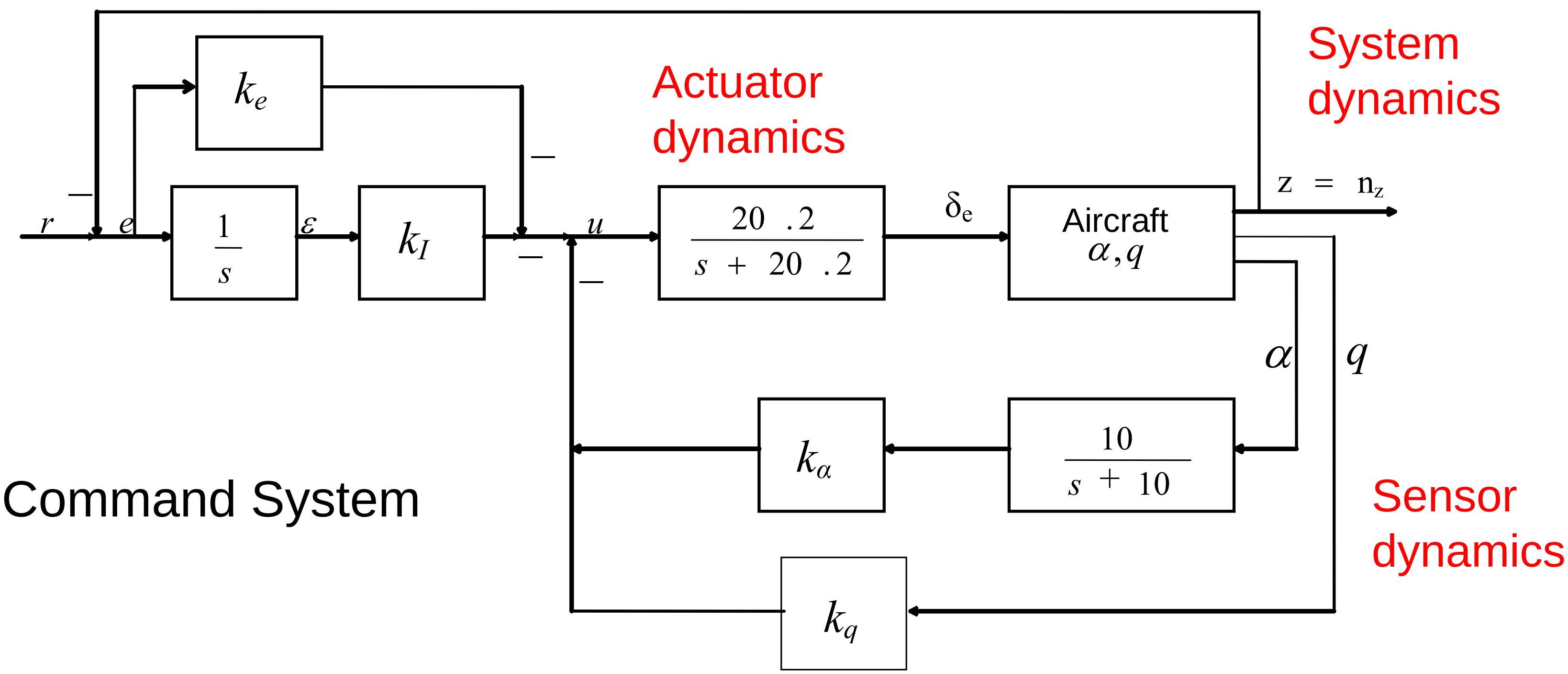
Until Convergence

Based on ARE, so no initial stabilizing gain needed !!

Tries to project gain onto nullspace perp. of C using degrees of freedom in L

Aircraft Autopilot Design

F-16 Normal Acceleration Regulator Design



G Command System

$$y = [\alpha_F \quad q \quad e \quad \varepsilon]^T$$

$$u = -Ky = -[k_\alpha \quad k_q \quad k_e \quad k_I]y$$

Theorem 2. - new work

Parametrization of all H-infinity Static SVFB Controls

Assume that Q>0, then K is a stabilizing SVFB with L_2 gain bounded by γ If and only if:

i.(A, B) is stabilizable

ii.There exist a matrix *L* such that

$$K = R^{-1}(B^T P + L)$$

where $P>0$, $P^T = P$, is a solution of

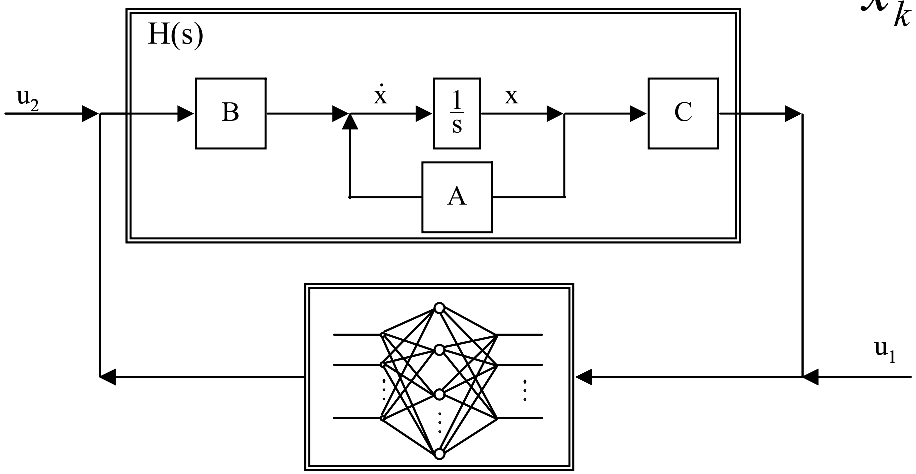
$$PA + A^T P + Q + \frac{1}{\gamma^2} PDD^T P - PBR^{-1}B^T P + L^T R^{-1} L = 0$$

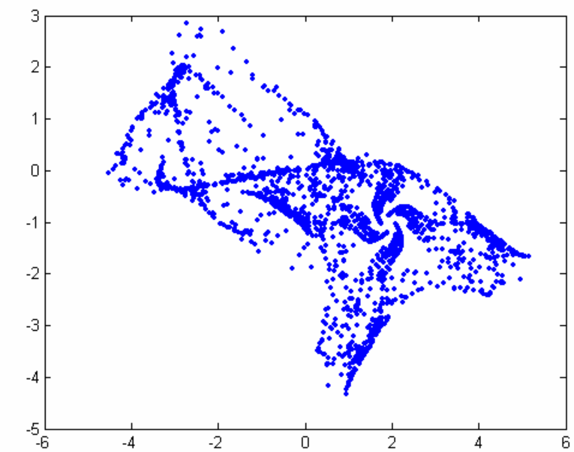
OPFB is a special case

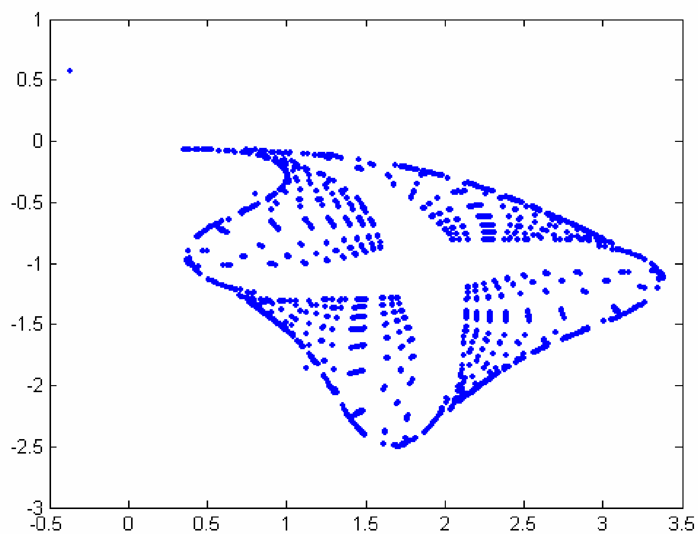
Chaos in Dynamic Neural Networks

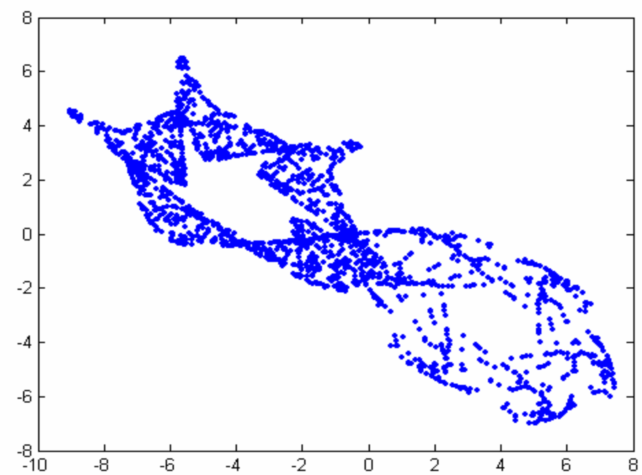
c.f. Ron Chen

$$x_{k+1} = Ax_k + W^T \sigma(V^T x_k) + u_k$$





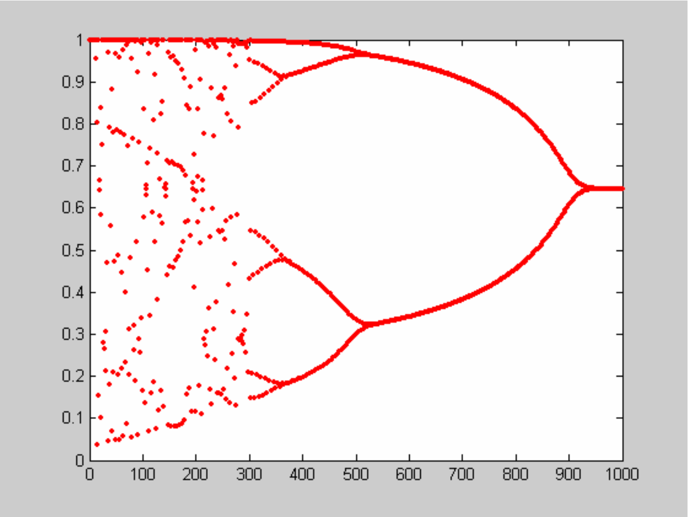


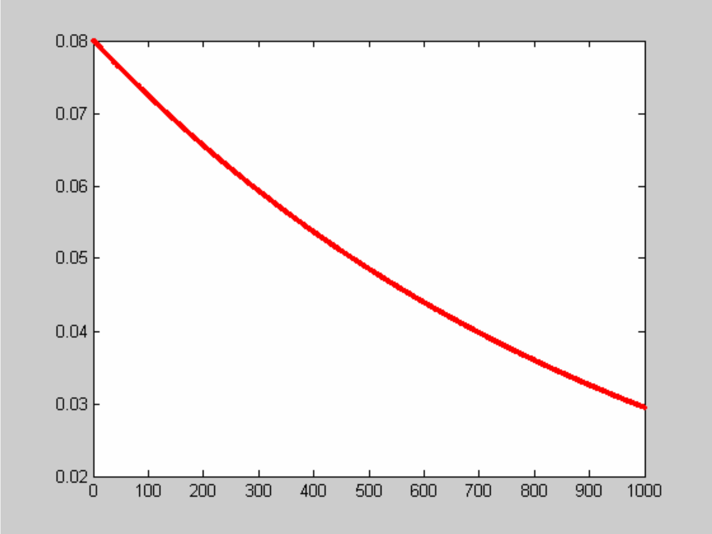


Jun Wang

$$z_{k+1} = \beta z_k$$

$$y_{k+1} = \alpha y_k + g - z_k \left(\frac{1}{1+e^{-y_k/\rho}} - I \right)$$





%MATLAB file for chaotic NN from **Jun Wang's** paper

```
function [ki,x,y,z]=tcnn(N);
y(1)= rand; ki(1)=1; z(1)= 0.08;
a=0.9; e= 1/250; Io=0.65;
g= 0.0001; b=0.001;

for k=1: N-1;
   ki(k+1)= k+1;
   x(k)= 1/(1+exp(-y(k)/e));
   y(k+1)= a*y(k) + g -
z(k)*(x(k) - Io);
   z(k+1)= (1-b)*z(k);
end
x(N)= 1/(1+exp(-y(N)/e));
```