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
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

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

Mechanical Motion Systems (Vehicles, Robots)

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

- inertia
- Coriolis/centripetal force
- gravity
- friction
- disturbances
- Control Input
- Actuator problems
Darwinian Selection & Population Dynamics

Volterra’s fishes
\[ \dot{x}_1 = ax_1 - bx_1 x_2 \]
\[ \dot{x}_2 = -cx_2 + dx_1 x_2 \]

\( x_1 = \text{prey} \)
\( x_2 = \text{predator} \)

Effects of Overcrowding
Limited food and resources
\[ \dot{x}_1 = ax_1 - bx_1 x_2 - ex_1^2 \]
\[ \dot{x}_2 = -cx_2 + dx_1 x_2 \]

Favorable to Prey!
Dynamical System Models

Continuous-Time Systems

Nonlinear system
\[
\dot{x} = f(x) + g(x)u \\
y = h(x)
\]

Linear system
\[
\dot{x} = Ax + Bu \\
y = Cx
\]

Discrete-Time Systems

Nonlinear system
\[
x_{k+1} = f(x_k) + g(x_k)u_k \\
y_k = h(x_k)
\]

Linear system
\[
x_{k+1} = Ax_k + B_k \\
y_k = Cx_k
\]

Control Inputs  \quad \text{ Internal States } \quad \text{ Measured Outputs}
Issues in Feedback Control

- Desired trajectories
- Feedforward controller
- Control inputs
- System
- Measured outputs
- Feedback controller
- Sensor noise
- Disturbances

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

Direct Scheme

Feedback/Feedforward Scheme
Cell Homeostasis

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

Permeability control of the cell membrane

Optimality in Control Systems Design

R. Kalman 1960

Rocket Orbit Injection

Dynamics

\[ \dot{r} = \omega \]

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

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

\[ \dot{m} = -Fm \]

Objectives

- Get to orbit in minimum time
- Use minimum fuel

http://microsat.sm.bmstu.ru/e-library/Launch/Dnepr_GEO.pdf
**Performance Index, Cost, or Value function**

CT

\[
J = \int_{0}^{T} [Q(x) + R(u)] \, dt = \int_{0}^{T} r(x, u) \, dt
\]

DT

\[
J = \sum_{k=0}^{N} r(x_k, u_k)
\]

**Strategic utility**

**utility**

- Minimum energy
  \[
r(x, u) = x^T Qx + u^T Ru
  \]
- 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)

Fuzzy Logic Rule Base

Input Membership Fns.  Output Membership Fns.

Input x  Output u

Neural Network (NN)

(Includes Adaptive Control)

Input x  Output u

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

Desired trajectory

\[ r(t) = \dot{e}(t) + \Lambda e(t) \]

Actual trajectory

Unity-Gain Tracking Loop

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
\[ e(t) = q_d(t) - q(t) \quad r = \dot{e} + \Lambda e \]

Error dynamics
\[ M\dot{r} = -V_m r + f(x) + \tau_d - \tau \]
The equations give the FB controller structure

Tracking error

\[ e(t) = q_d(t) - q(t) \]

Sliding variable

\[ r = \dot{e} + \Lambda e \]

Robot dynamics

\[ M(q)\ddot{q} + V_m(q,\dot{q})\dot{q} + G(q) + F(\dot{q}) + \tau_d = \tau \]
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\[ e(t) = q_d(t) - q(t) \quad r = \dot{e} + \Lambda e \]

Error dynamics\[ M\ddot{r} = -V_m r + f(x) + \tau_d - \tau \]

Universal Approximation Property

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

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

Closed-loop dynamics\[ M\ddot{r} = -V_m r - K_v r + W^T \sigma(V^T x) + \varepsilon - \hat{W}^T \sigma(\hat{V}^T x) + \tau_d + \nu(t) \]
\[ M\ddot{r} = -V_m r - K_v r + \tilde{f} + \tau_d + \nu(t) \]
Neural Network Robot Controller

**Universal Approximation Property**

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
Stability Proof based on Lyapunov Extension

Define a Lyapunov Energy Function

\[ L = \frac{1}{2} r^T M r + \frac{1}{2} \text{tr}(\tilde{W}^T \tilde{W}) + \frac{1}{2} \text{tr}(\tilde{V}^T \tilde{V}) \]

Differentiate

\[ \dot{L} = -r^T K_v r + \frac{1}{2} r^T (\dot{M} - 2V_m) r \\
+ \text{tr} \tilde{W}^T (\tilde{W} + \hat{\sigma} r^T - \hat{\sigma}' \tilde{V} x r^T) \\
+ \text{tr} \tilde{V}^T (\tilde{V} + x r^T \tilde{W}^T \hat{\sigma}') + r^T (w + \nu) \]

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

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
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 \( Z \).

Take the control input as

\[
\tau = \hat{W}^T \sigma (\hat{V}^T x) + K_v r - v
\]

with

\[
v(t) = -K_Z (\|Z\|_F + Z_M) r.
\]

Let weight tuning be provided by

\[
\hat{W} = F \hat{\sigma} r^T - F \hat{\sigma} \hat{V}^T x r^T - \kappa F \|r\| \hat{W},
\]

\[
\dot{V} = G x (\hat{\sigma}^T \hat{W} r)^T - \kappa G \|r\| \hat{V}
\]

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

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 \( K_v \).

Can also use simplified tuning- Hebbian

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

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

Velocity
Dynamic NN and Passivity

Feedforward Loop

Nonlinear Inner Loop

Tracking Loop

Robust Control Term

\[ q_d \]

\[ e \]

\[ r \]

\[ v(t) \]

\[ \dot{q}_d \]

\[ \hat{f}(x) \]

\[ \tau \]

\[ q \]

Static NN \Rightarrow 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

\[ \dot{\hat{W}} = F\hat{\sigma}^T r - F\hat{\sigma}'\hat{V}^T xr^T \]
\[ \dot{\hat{V}} = Gx(\hat{\sigma}'^T \hat{Wr})^T \]

make the closed-loop system passive

The enhanced tuning algorithms

\[ \dot{\hat{W}} = F\hat{\sigma}^T r - F\hat{\sigma}'\hat{V}^T xr^T - \kappa F\|r\|\hat{W} \]
\[ \dot{\hat{V}} = Gx(\hat{\sigma}'^T \hat{Wr})^T - \kappa G\|r\|\hat{V} \]

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

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

Flexible dynamics

Problem: only one control input!

Flexible link pointing system

acceleration  velocity  Flex. modes
Neural network controller for Flexible-Link robot arm
### Coupled Systems

**Robotic mechanical dynamics**

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

**Motor electrical dynamics**

\[
Li + R(i, \dot{q}) + \tau_e = u_e
\]

- **Problem:** only one control input!

---

**Diagram:**

- **Sprung mass (car body)** $m_s$
- **Unsprung mass (tire)** $m_u$
- Terrain

**Active Suspension control**

**Vehicle**

- Load
- Drive
- Rotary damping (modeled as viscous)
Neural network backstepping controller for Flexible-Joint robot arm

Advantages over traditional Backstepping- no regression functions needed
Actuator Nonlinearities

Applied Control inputs → Actuator nonlinearity → Actual Control inputs → System → Outputs

Feedback controller

\[ \tau = D(u) \]

Deadzone

Backlash
**NN in Feedforward Loop - Deadzone Compensation**

\[ \dot{q}_d \]

\[ [\Lambda^T \ I] \]

**Estimate of Nonlinear Function**

\[ \hat{f}(x) \]

**NN Deadzone Precompensator**

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

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

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

PD control-backlash chops off tops & bottoms

NN control fixes the problem
NN Observers

Needed when all states are not measured

\[
\begin{align*}
\dot{\hat{x}}_1 &= \hat{x}_2 + k_D \hat{x}_1 \\
\dot{\hat{x}}_2 &= \hat{W}_o^T \sigma_o (\hat{x}_1, \hat{x}_2) + M^{-1}(x_1) \tau(t) + K \hat{x}_1 \\
\hat{r}(t) &= \hat{e}(t) + \Lambda \hat{e}(t) \\
\tau(t) &= \hat{W}_c^T \sigma_c (\hat{x}_1, \hat{x}_2) + K_v \hat{r}(t) - v_c(t)
\end{align*}
\]

\[
\begin{align*}
\hat{W}_o &= -k_D F_o \sigma_o (\hat{x}) \hat{x}_1^T \\
&\quad - \kappa_0 F_o \| \hat{x}_1 \| \hat{W}_o - \kappa_0 F_o \hat{W}_o \\
\hat{W}_c &= F_c \sigma_c (\hat{x}_1, \hat{x}_2) \hat{r}^T \\
&\quad - \kappa_c F_c \| \hat{r} \| \hat{W}_c
\end{align*}
\]
NN Control for Discrete Time Systems

\[ x(k + 1) = f(x(k)) + g(x(k))u(k) \]

NN Tuning
\[
\hat{W}_i(k + 1) = \hat{W}_i(k) - \alpha_i \phi_i(k) \hat{y}_i^T(k) - \Gamma \left\| I - \alpha_i \phi_i(k) \phi_i^T(k) \right\| \hat{W}_i(k)
\]

Error-based tuning
\[
\hat{y}_i(k) = \hat{W}_i^T(k) \phi_i(k) + K_v r(k), \quad \text{for } i = 1, \ldots, N - 1 \quad \text{and} \quad \hat{y}_N(k) = r(k + 1), \quad \text{for last layer}
\]

U.S. Patent- Jagannathan, Lewis
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
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

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

Tuning laws

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

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

$$\hat{W} = K_w (\Phi - A\hat{a} - B\hat{b})r^T - k_w K_w \hat{W} \| r \|$$
Dynamic Focusing of Awareness

Initial MFs

Final MFs
Elastic Fuzzy Logic- c.f. P. Werbos

\[ \phi(z,a,b,c) = \phi_B(z,a,b)^{c^2} \quad \text{Weights importance of factors in the rules} \]

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

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
\[ \dot{W} = K_w (\dot{\Phi} - A\dot{a} - B\dot{b} - C\dot{c})r^T - k_w K_w \dot{W}\| r \| \]

Tune Membership Functions
\[ \dot{\hat{a}} = K_a A^T \dot{\hat{W}} r - k_a K_a \hat{a}\| r \| \]
\[ \dot{\hat{b}} = K_b B^T \dot{\hat{W}} r - k_b K_b \hat{b}\| r \| \]
\[ \dot{\hat{c}} = K_c C^T \dot{\hat{W}} r - k_c K_c \hat{c}\| r \| \]
Better Performance
Fuzzy Logic Critic NN controller

Action Generating NN

\[ f(x) \]

Desired Trajectory

Performance Evaluator

\[ r(t) \]

Instantaneous Utility

\[ R(t) \]

FL Critic

\[ \hat{f}(x) \]

x(t)

d(t)

Unknown Plant

\[ u(t) \]

tuning

\[ R(t) \]
Learning FL Critic Controller

Tune Action generating NN (controller)

\[
\hat{W}_2 = \Gamma \sigma(\chi_2)^R T - \Gamma \sigma(\chi_2)^R T \hat{W}_1 \mu^*(\hat{V}_r^T \hat{V}_r^T) - \Gamma \hat{W}_2
\]

Tune Fuzzy Logic Critic

\[
\dot{\hat{W}}_1 = -\mu(\hat{V}_1^T r) R^T - \Gamma \hat{W}_1,
\]
\[
\dot{\hat{V}}_1 = -r H^T \hat{W}^T \mu^*(\hat{V}_1^T r) - \Phi_1 \hat{V}_1,
\]

Critic requires MEMORY

FL Critic

Action generating NN

Critic requires MEMORY

ACTION GENERATING NN

---

Reference input membership functions

Fuzzy rule base

Output membership functions

Learning FL Critic Controller

未知

植物性能评估器

\(d(t)\)
Reinforcement Learning NN Controller

Reference Signal
Performance Measurement Mechanism

Reinforcement Signal $R(t)$

Critic Element

Robust Term $K_v$

User input: Reference Signal $q_d(t)$

Performance Measurement Mechanism

Utility $r(t)$

Action Generating Neural Net

Input Pre-processing $z_1, z_2, \ldots, z_N$

Input Layer $x_1, x_2, \ldots, x_n$

Hidden Layer

Output Layer $y_1, y_2, \ldots, y_m$

Action $u(t) = \sum \hat{g}(x)$

Control Action

Plant $q(t)$

$fr(t), d(t)$

Input $q_d(t)$
High-Level NN Controllers Need Exotic Lyapunov Fns.

Reinforcement NN control

Simplified critic signal

\[ R(t) = \text{sgn}(r(t)) = \pm 1 \]

Lyapunov Fn

\[ L(t) = \sum_{i=1}^{n} |r_i| + \frac{1}{2} \text{tr}(\tilde{W}^T F^{-1}\tilde{W}) \]

\[ \dot{L} = \text{sgn}(r)^T \dot{r} + \text{tr}(\tilde{W}^T F^{-1}\dot{\tilde{W}}) \]

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

Tuning Law only contains \( R(t) \)

\[ \dot{\hat{W}} = F \sigma(x) R^T - \kappa F \tilde{W} \]

Adaptive Reinforcement Learning

Critic is output of NN #1

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

\[ L(t) = \ln(1 + e^{-\alpha r(t)}) + \ln(1 + e^{\alpha r(t)}) + \frac{1}{2} \text{tr}(\tilde{W}^T F^{-1}\tilde{W}) \]

\[ \dot{L} = \left( \frac{\alpha^+}{1 + e^{-\alpha^+ r(t)}} + \frac{-\alpha^-}{1 + e^{\alpha^- r(t)}} \right) \dot{r}(t) + \text{tr}(\tilde{W}^T F^{-1}\dot{\tilde{W}}) \]

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

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

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

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

Performance output

\[ \dot{x} = f(x) + g(x)u + k(x)d \]
\[ y = x \]
\[ z = \psi(x, u) \]

\[ u = l(y) \]

where

\[ \|z\|^2 = h^T h + \|u\|^2 \]

L₂ Gain Problem

Find control \( u(t) \) so that

\[ \int_0^\infty \|z(t)\|^2 \, dt = \int_0^\infty (h^T h + \|u\|^2) \, dt \leq \gamma^2 \]

For all L₂ disturbances

And a prescribed gain \( \gamma^2 \)

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 $\|u\|^2 = u^T Ru$ and $\|d\|^2 = d^T d$

Hamilton-Jacobi Isaacs (HJI) equation

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

Stationary Point

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

Optimal control

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

Worst-case disturbance

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

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

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

$$\|u\|^2_q = 2 \int_0^u \phi^{-T}(v) d\nu$$

(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

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

Stationarity conditions

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

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

Optimal inputs

\[ u^* = -\frac{1}{2} \phi(g^T(x)V_x) \]

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

Leibniz’s Formula

Solve for \( u(t) \)

Note \( u(t) \) is bounded!
Cannot solve HJI !!

Successive Solution- Algorithm 1:
Let $\gamma$ be prescribed and fixed.

$u_0$, a stabilizing control with region of asymptotic stability $\Omega_0$

1. Outer loop- update control
   Initial disturbance $d^0 = 0$

2. Inner loop- update disturbance
   Solve Value Equation

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

   Inner loop update

   \[
   d^{i+1} = \frac{1}{2\gamma^2} k^T(x) \frac{\partial V^i_{j}}{\partial x}
   \]

   go to 2.
   Iterate $i$ until convergence to $d^\infty, V^\infty_j$ with RAS $\Omega^\infty_j$

Outer loop update

\[
 u_{j+1} = -\frac{1}{2} \phi \left( g^T(x) \frac{\partial V^\infty_j}{\partial x} \right)
\]

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 \( \Omega_0 \)

Sometimes the algorithm converges to the optimal HJI solution \( V^*, \Omega^*, u^*, d^* \)

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

For every iteration on the disturbance \( d^i \) one has
\[
V^i \leq V^{i+1} \quad \text{the value function increases}
\]
\[
\Omega^i \supseteq \Omega^{i+1} \quad \text{the RAS decreases}
\]

For every iteration on the control \( u_j \) one has
\[
V^\infty \geq V^{\infty, j+1} \quad \text{the value function decreases}
\]
\[
\Omega^\infty \subseteq \Omega^{\infty, j+1} \quad \text{the RAS does not decrease}
\]
Problem- Cannot solve the Value Equation!

Neural Network Approximation for Computational Technique

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

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

Value function gradient approximation is

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

Substitute into Value Equation to get

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

Therefore, one may solve for NN weights at iteration $(i,j)$
Neural Network Feedback Controller

Optimal Solution

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

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

A NN feedback controller with nearly optimal weights
Example: Linear system

\[
\begin{bmatrix}
\dot{x}_1 \\
\dot{x}_2
\end{bmatrix} = \begin{bmatrix}
0 & -0.5 \\
1 & 1.5
\end{bmatrix} \begin{bmatrix}
x_1 \\
x_2
\end{bmatrix} + \begin{bmatrix}
0 \\
-1
\end{bmatrix} u, \quad |u| \leq 1
\]

\[
V_{15}(x_1, x_2) = w_1 x_1^2 + w_2 x_2^2 + w_3 x_1 x_2 + w_4 x_1^4 + w_5 x_2^4 + w_6 x_1^3 x_2 + w_7 x_1^2 x_2^2 + w_8 x_1 x_2^3 + w_9 x_1^3 + w_{10} x_2^6 + w_{11} x_1^5 x_2 + w_{12} x_1^4 x_2^2 + w_{13} x_1^3 x_2^3 + w_{14} x_1^2 x_2^4 + w_{15} x_1 x_2^5
\]

Activation functions = even polynomial basis up to order 6
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

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

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

\[ \varepsilon = 0.2 \]
Minimum-Time Control

\[ V = \int_{0}^{\infty} \left[ \tanh(x^T Q x) + 2 \int_{0}^{u} (\phi^{-1}(\mu))^T R d\mu \right] dt \]

State Evolution for both controllers

Encode into Valua Function
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

\[ x_{k+1} = f(x_k, u_k) \quad \text{ \quad } V(x_k) = \sum_{i=k}^{N} \gamma^{i-k} r(x_k, u_k) \]

Value in difference form -

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

Howard Policy Iteration- Iterate the following until convergence

1. Find the value for the prescribed policy

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

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 \( V(x_k) \)

**Dual heuristic programming**

Gradient \( \frac{\partial V}{\partial x} \)

AD Heuristic dynamic programming
(Watkins Q Learning)

Q function \( Q(x_k, u_k) \)

AD Dual heuristic programming

Gradients \( \frac{\partial Q}{\partial x}, \frac{\partial Q}{\partial u} \)

Action NN to approximate the Control

Bertsekas- Neurodynamic Programming

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

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

\[ u^*(x(t)) = -\frac{1}{2} g^T(x) \frac{\partial V^*}{\partial x} \]

\[ d^*(x(t)) = \frac{1}{2\gamma^2} k^T(x) \frac{\partial V^*}{\partial x} \]

**HJB equation**

\[ 0 = \left( \frac{dV^*}{dx} \right)^T f + h^T h - \frac{1}{4} \left( \frac{dV^*}{dx} \right)^T g g^T \frac{dV^*}{dx} + \frac{1}{4\gamma^2} \left( \frac{dV^*}{dx} \right)^T k k^T \frac{dV^*}{dx} \]
**Continuous Time Policy Iteration**

Select a stabilizing initial control

1. **Outer loop**- update control
   
   Initial disturbance set to zero

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**

\[
u_{j+1} = -\frac{1}{2} \left( g^T (x) \frac{\partial V^i_j}{\partial x} \right)
\]

Go to 1.

Until convergence

*Abu-Khalaf and Lewis- H inf*

*Saridis – H\(_2\)*

c.f. Howard work in DT Systems
Neural Network Approximation of Value Function

\[ \hat{V}(x, w^i_j) = w_j^i \sigma(x) \]

Lyapunov equation becomes

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

Control action

\[ u^*(x) = -\frac{1}{2} R^{-1} g^T(x) \nabla \sigma^T(x) v^* \]

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

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

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

\[ E = \frac{1}{2} |\delta|^2 \]

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

Gradient

\[ \dot{w} = -\alpha \nabla \sigma(x) f(x, u) \delta \]

Update weights using, e.g., gradient descent

Or RLS
\textbf{Action NN}

\[ Y_2 = -\frac{1}{2} R^{-1} g^T (x) \nabla \sigma^T (x) w = \phi^T w \]

\[ \phi^T (x) = -\frac{1}{2} R^{-1} g^T (x) \nabla \sigma^T (x) \]

Activation fns depend on system dynamics

\[ \hat{Y}_2 = \phi^T (x) \nu \quad \text{Action NN} \]

\[ e_2 (x) = \hat{Y}_2 - Y_2 = -\frac{1}{2} R^{-1} g^T (x) \nabla \sigma^T (x) [\nu - w] = \phi^T (x)[\nu - w] \]

\[ \dot{\nu} = -\beta \phi (x) e_2 (x) \quad \text{update weights by gradient descent} \]

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

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

\textit{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
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 $\gamma$ 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 $\gamma$, $Q$, $R$

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

\[
P_n A + A^T P_n + Q + \frac{1}{\gamma^2} P_n DD^T P_n - P_n BR^{-1} B^T P_n + L_n^T R^{-1} L_n = 0
\]

Evaluate gain and update $L$

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

\[
L_{n+1} = RK_{n+1} C - B^T P_n
\]

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

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

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

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

OPFB is a special case
Chaos in Dynamic Neural Networks
c.f. Ron Chen

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