

Robust Reinforcement Learning with Static and Dynamic Stability

Chuck Anderson

with Peter Young, Douglas Hittle,
Matt Kretchmar, Michael Anderson, Jilin Tu,
Chris Delnero, David Hodgson

Colorado State University, Fort Collins, Colorado

www.cs.colostate.edu/~anderson

www.engr.colostate.edu/nnhvac

Supported by NSF Grants CMS-9804757 and 9732986,
Siemens Building Technologies, Colorado State University

Overview

Reinforcement learning agent in parallel to engineered controller.

Potential for combining reinforcement learning and robust control theory.

Very brief reviews of

- small gain theorem

- integral quadratic constraints (IQC's)

- robust control

Integral Quadratic Constraints for neural network

- static nonlinearity (tanh)

- time-varying weights (learning algorithm)

Results on simulated control tasks

Other topics in approximate dynamic programming

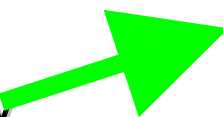
Motivation

From the Machine Learning point of view, how can we train neural networks with reinforcement learning while guaranteeing stability?

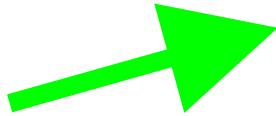
From the Electrical and Computer Engineering view point, how can neural networks be used with robust control systems to improve performance?

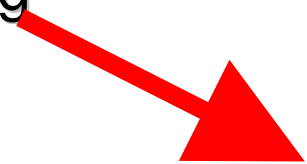
From the Mechanical Engineering perspective, how can neural networks be applied to highly non-linear, time varying HVAC systems?

Motivation

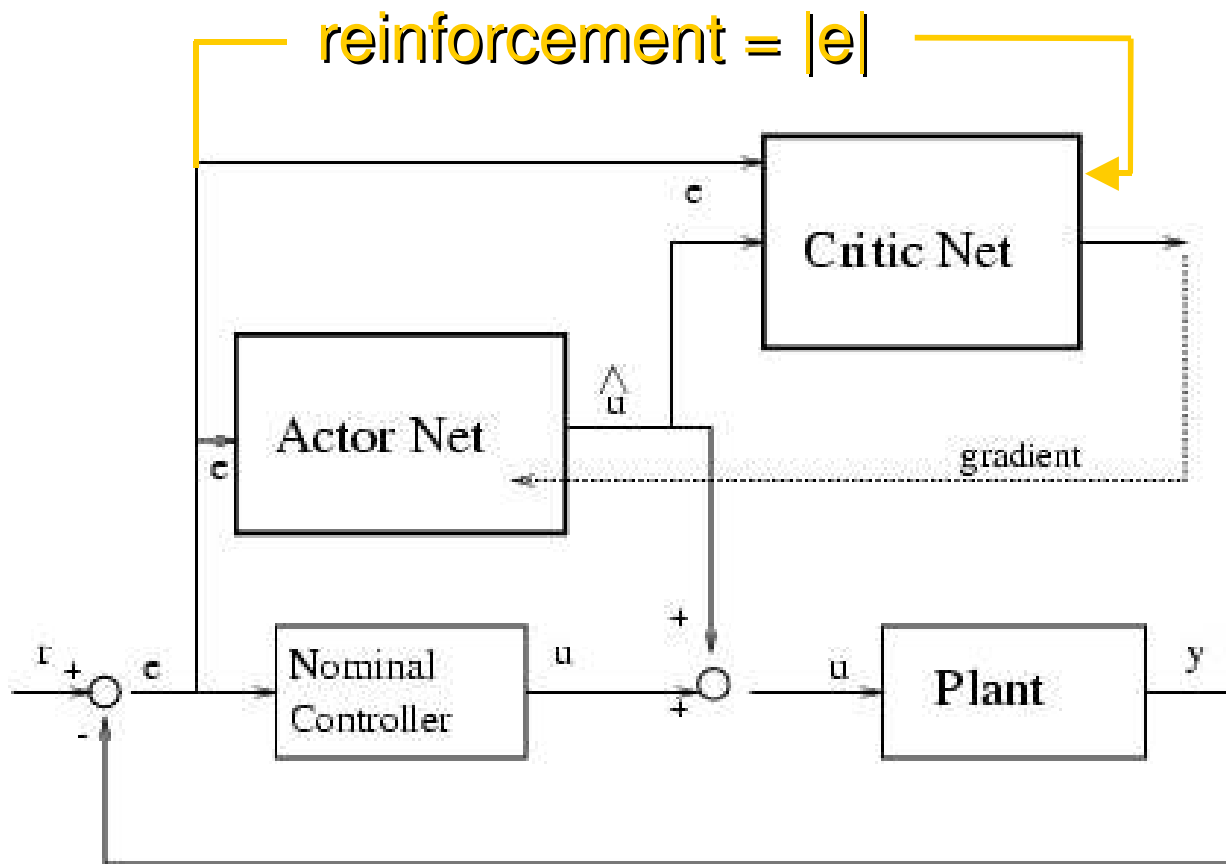
Robust control theory  Guarantees stability

 Results in less aggressive controllers

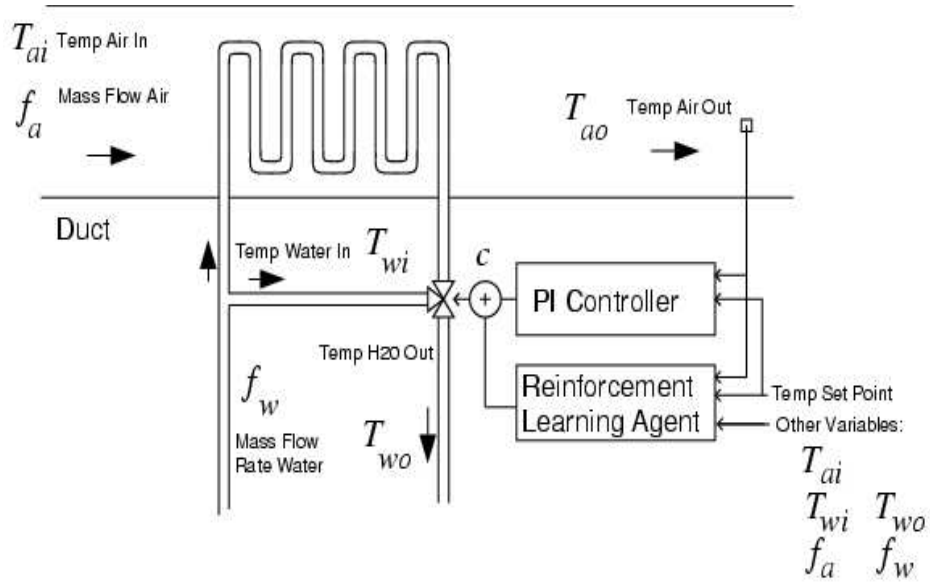
Reinforcement learning  Optimizes the performance of a controller

 No guarantee of stability while learning

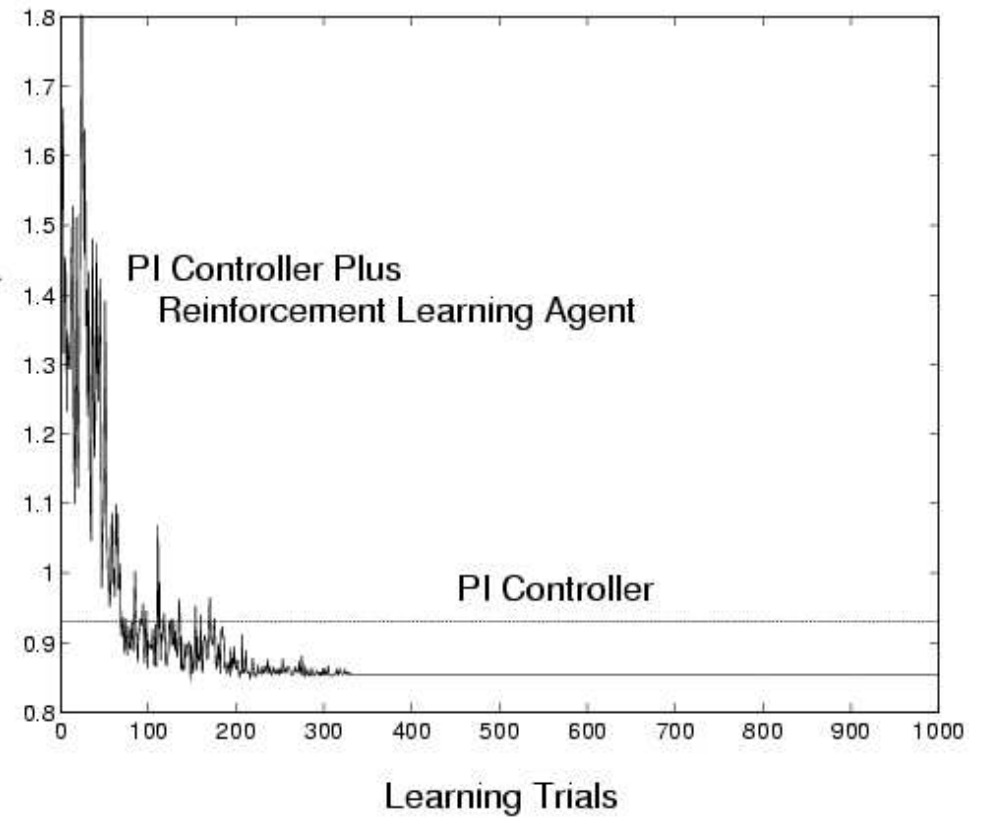
Reinforcement Learning Agent in Parallel with Controller



Prior Results



RMS Error
Between
Setpoint
and T_{ao}



Robust Reinforcement Learning?

Learns improved control, but no guarantee of stability.

Can we formulate combination of PI control and RL within robust control theory?

Robust control theory is based on linear, time-invariant transfer functions.

RL agents are nonlinear, because of the units' activation functions.

RL agents are time-varying, because they update their parameters to produce improved behavior.

First, a brief introduction to an LMI (Linear Matrix Inequality) formulation of robust control . . .

Small Gain Theorem

Norms

$$\|u\|_2 = \sqrt{\int_0^{\infty} |u(t)|^2 dt}$$

$$\|u\|_2 = \sqrt{\frac{1}{2\pi} \int_{-\infty}^{\infty} |u(j\omega)|^2 d\omega}$$

$$\|G\|_{\infty} = \sup_{\omega} |G(j\omega)| = \sup_u \frac{\|y\|_2}{\|u\|_2}$$

Laplace Transform

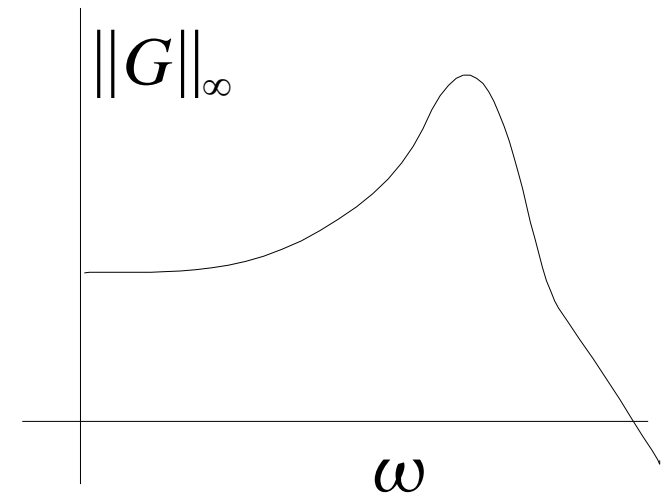
$$\dot{y} + y = u$$

$$sY(s) + Y(s) = U(s)$$

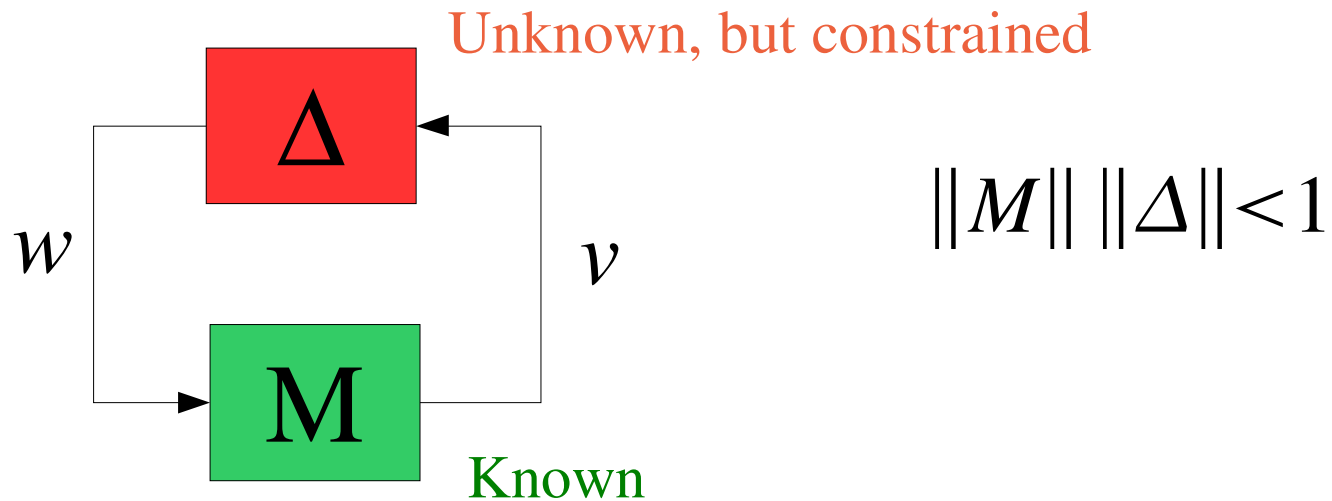
$$(s+1)Y(s) = U(s)$$

$$\frac{Y(s)}{U(s)} = \frac{1}{s+1} = G(s)$$

$$Y(s) = G(s)U(s)$$

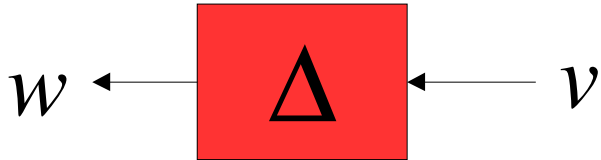


Small Gain Theorem



If $\|\Delta\| < 1$, then system is stable provided $\|M\| \leq 1$

Integral Quadratic Constraints



$$\|\Delta\|_{\infty} < 1$$

$$\|w\|_2^2 < \|v\|_2^2$$

$$\|w\|_2^2 - \|v\|_2^2 < 0$$

$$\int_{-\infty}^{\infty} |w(j\omega)|^2 d\omega - \int_{-\infty}^{\infty} |v(j\omega)|^2 d\omega < 0$$

$$\int_{-\infty}^{\infty} \left(w^*(j\omega)w(j\omega) - v^*(j\omega)v(j\omega) \right) d\omega < 0$$

$$\int_{-\infty}^{\infty} \left(\begin{bmatrix} v(j\omega) \\ w(j\omega) \end{bmatrix}^* \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} v(j\omega) \\ w(j\omega) \end{bmatrix} \right) d\omega \geq 0$$

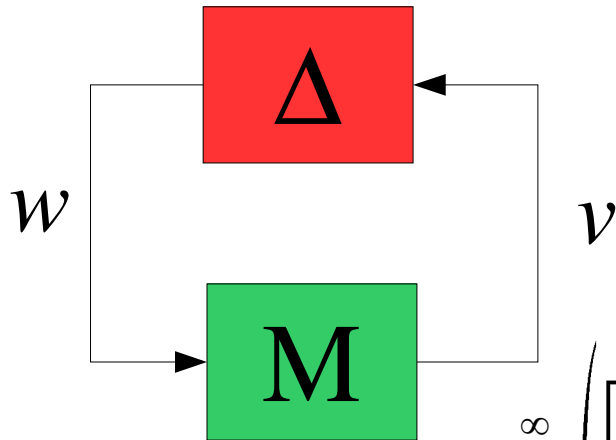
Complex Conjugate

$$|v|^2 = v^* v$$

$$v^* v = (x - iy)(x + iy)$$

$$v^* v = x^2 + y^2$$

Integral Quadratic Constraints



$$\int_{-\infty}^{\infty} \begin{pmatrix} \begin{bmatrix} v(j\omega) \\ w(j\omega) \end{bmatrix}^* \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} v(j\omega) \\ w(j\omega) \end{bmatrix} \end{pmatrix} d\omega \geq 0$$

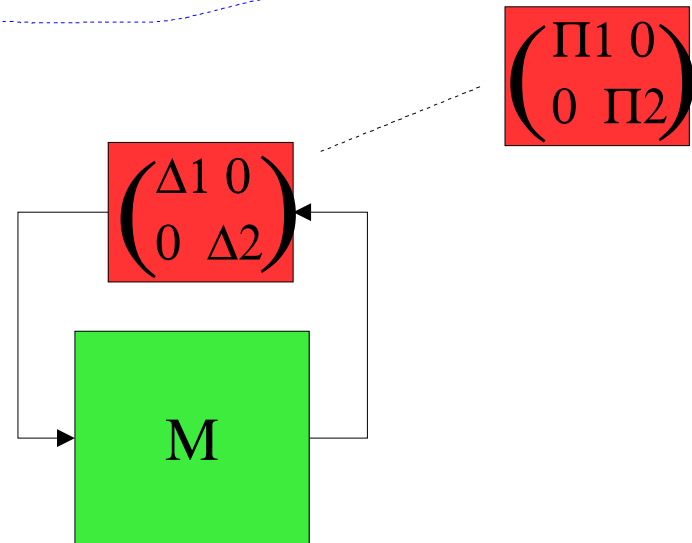
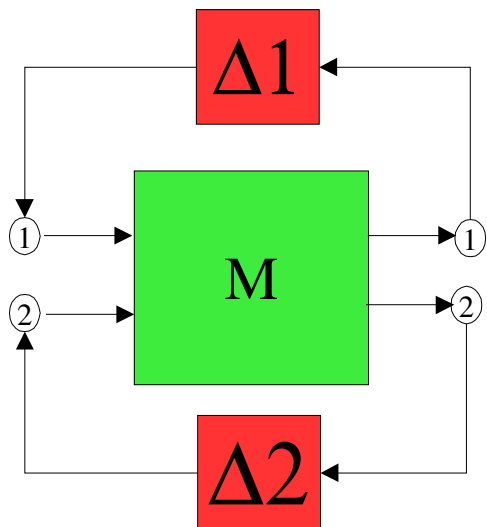
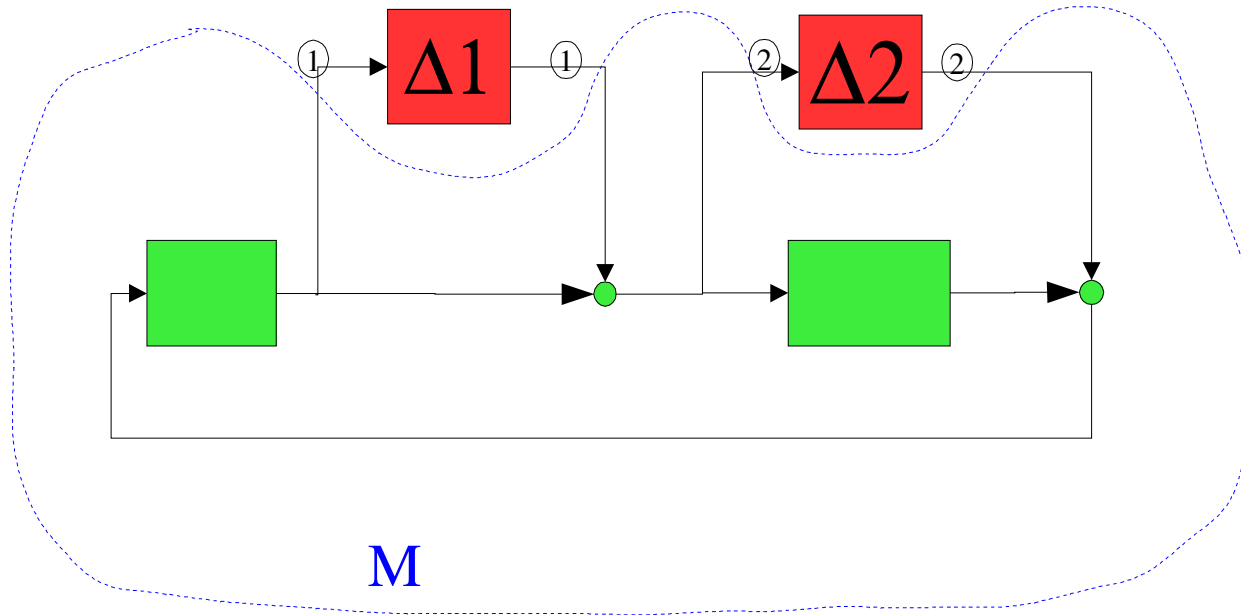
If $\exists \epsilon > 0$ such that $\begin{bmatrix} M(j\omega) \\ I \end{bmatrix}^* \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} M(j\omega) \\ I \end{bmatrix} \leq -\epsilon I, \quad \forall \omega$

then the feedback interconnection of M, Δ is stable.

$$|M(j\omega)|^2 - I < -\epsilon I$$

$$|M(j\omega)|^2 < (1 - \epsilon) I$$

Integral Quadratic Constraints



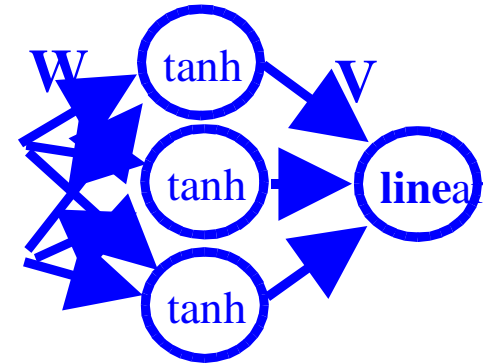
Integral Quadratic Constraints

$$\begin{bmatrix} M(j\omega) \\ I \end{bmatrix}^* \Pi_i(j\omega) \begin{bmatrix} M(j\omega) \\ I \end{bmatrix} = \begin{bmatrix} (j\omega I - A)^{-1} B \\ I \end{bmatrix}^* P_i \begin{bmatrix} (j\omega I - A)^{-1} B \\ I \end{bmatrix} \leq -\epsilon I$$

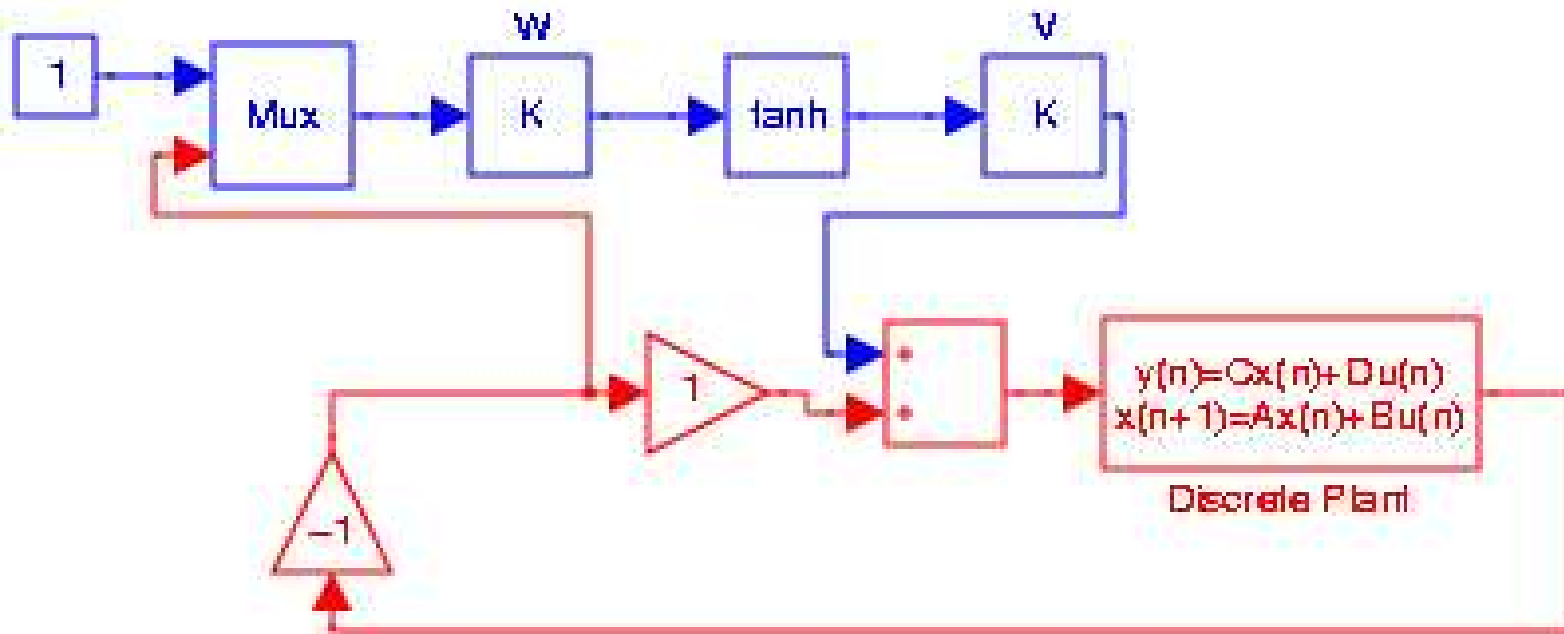
$$\begin{bmatrix} QA + A^T Q & QB \\ B^T Q & 0 \end{bmatrix} + \sum_{i=1}^n p_i P_i < 0 \quad \text{Kalman-Yakubovich-} \\ \text{Popov Lemma}$$

a finite-dimensional LMI feasibility problem in variables p_i and Q .

Neural Net for Learning Agent



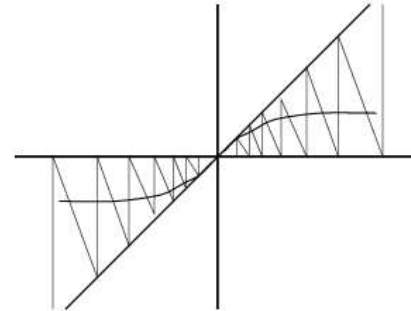
Actor Network (Critic Network not shown)



IQCs for Neural Network as RL Agent

Nonlinear part: \tanh

replace with odd, bounded-slope
IQC



Time-varying part: weight updates

replace with slowly time-varying IQC

Replace with IQCs only for stability analysis, not during
operation

IQC for the tanh Nonlinearity



$$\tanh(-x) = -\tanh(x)$$

odd condition

$$0 \leq (\tanh(x_1) - \tanh(x_2))(x_1 - x_2) \leq (x_1 - x_2)^2$$

$$0 \leq (\tanh(x_1) - \tanh(x_2)) \leq (x_1 - x_2)$$

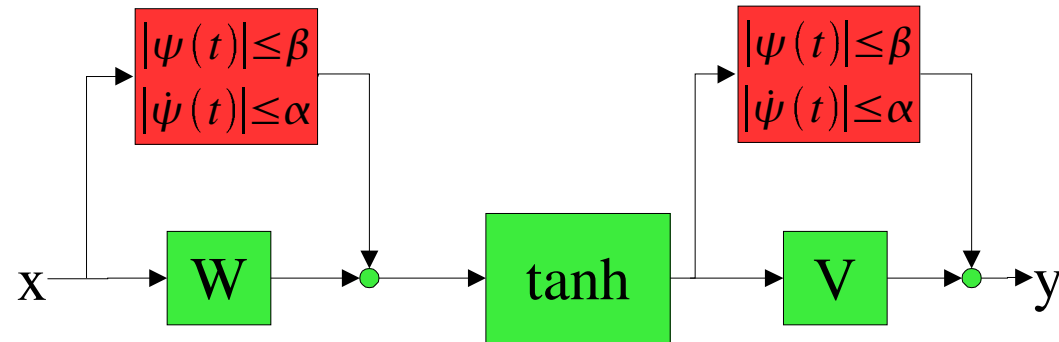
bounded slope condition

$$\Pi(j\omega) = \begin{bmatrix} 0 & 1 + \frac{p}{j\omega + 1} \\ 1 + \frac{p}{-j\omega + 1} & -2 \left(1 + \operatorname{Re} \left(\frac{p}{j\omega + 1} \right) \right) \end{bmatrix}$$

$$|p| \leq 1$$

using a scaling of $1/(s+1)$
and free parameter p

IQC for the Slowly Time-Varying Weights



$\psi(t)$ represents the changes to weights determined by learning algorithm

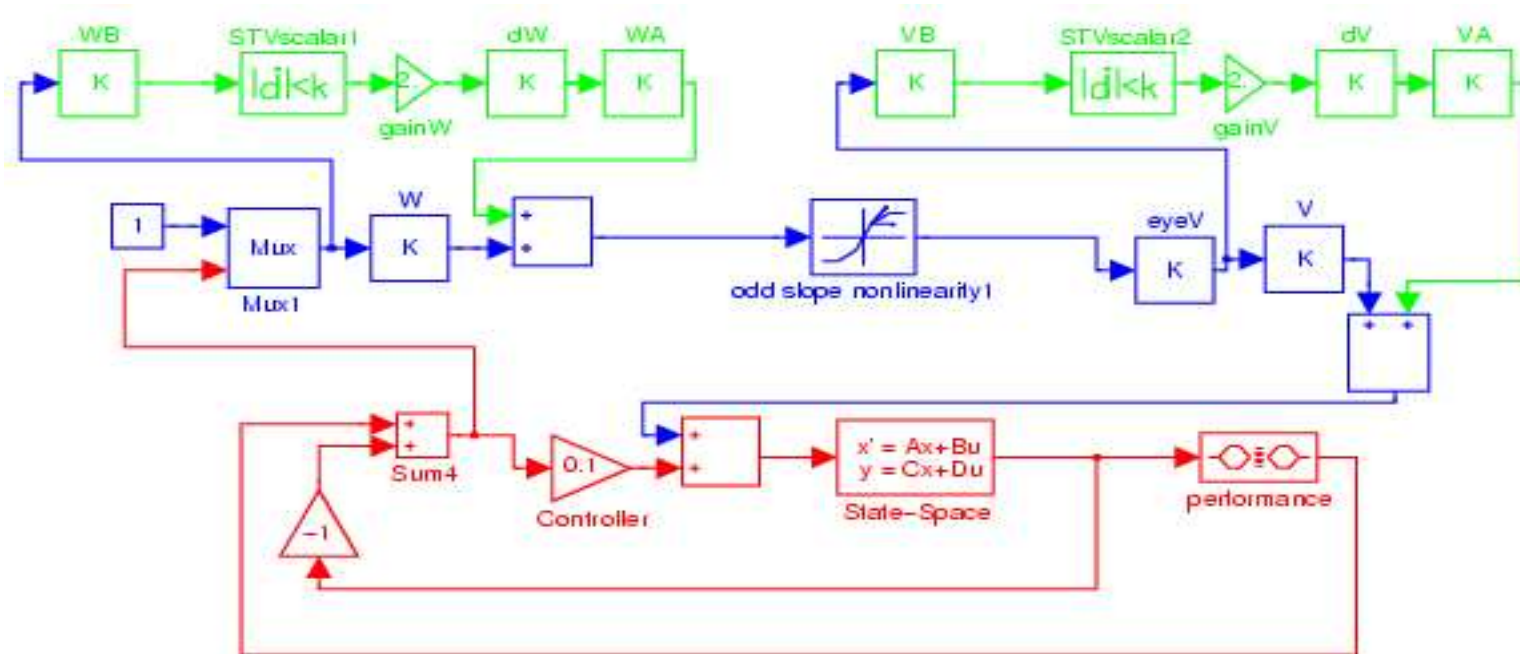
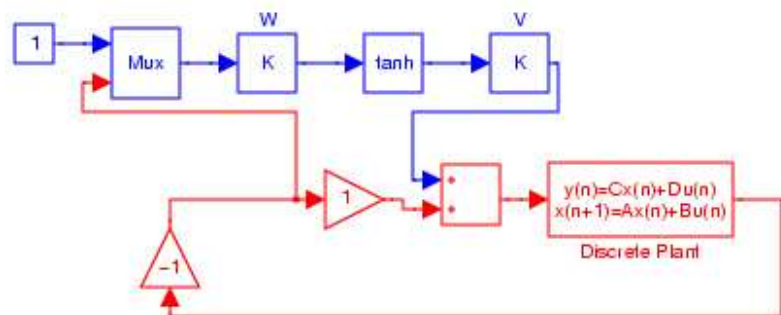
$$\begin{array}{ll} |\psi(t)| \leq \beta & \text{change in weights is bounded,} \\ |\dot{\psi}(t)| \leq \alpha & \text{and rate of change is bounded} \end{array}$$

β specifies bounds on weights for which stability analysis is valid.
 α specifies bounds on the learning rate used to adjust the weights.

IQCs for Neural Network as RL Agent

Two-layer neural net as actor,
in parallel with controller.

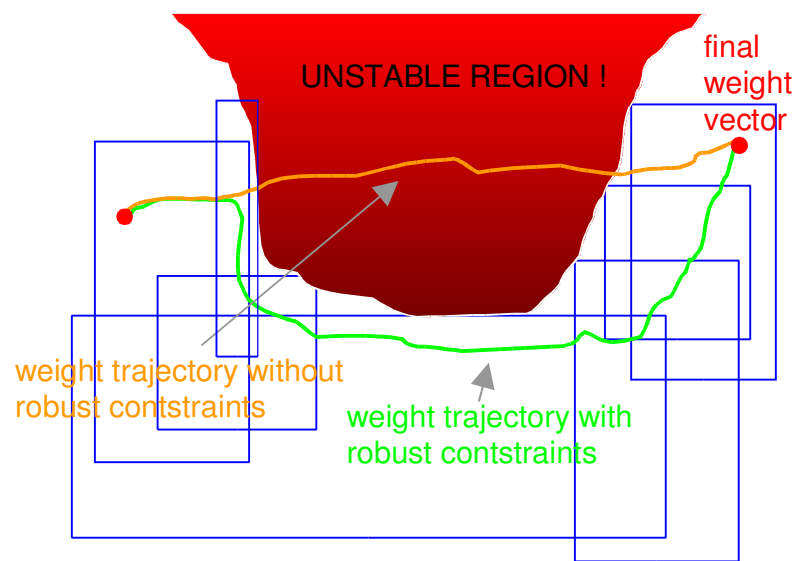
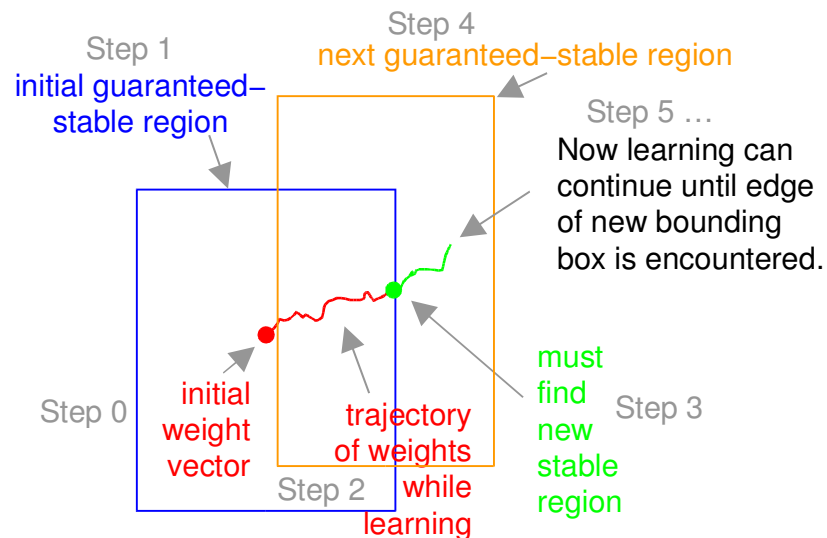
Now with tanh and varying
parameters “covered” by
IQCs.



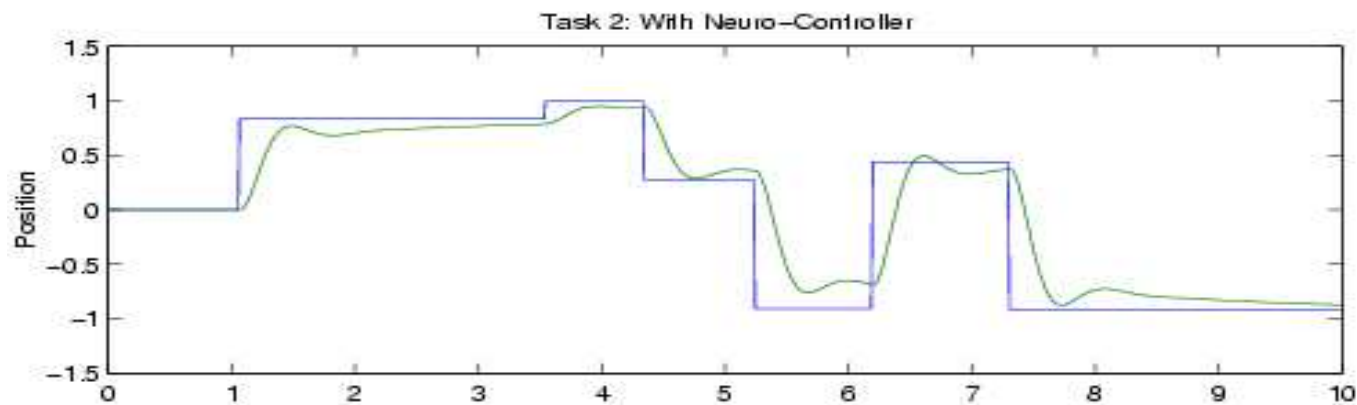
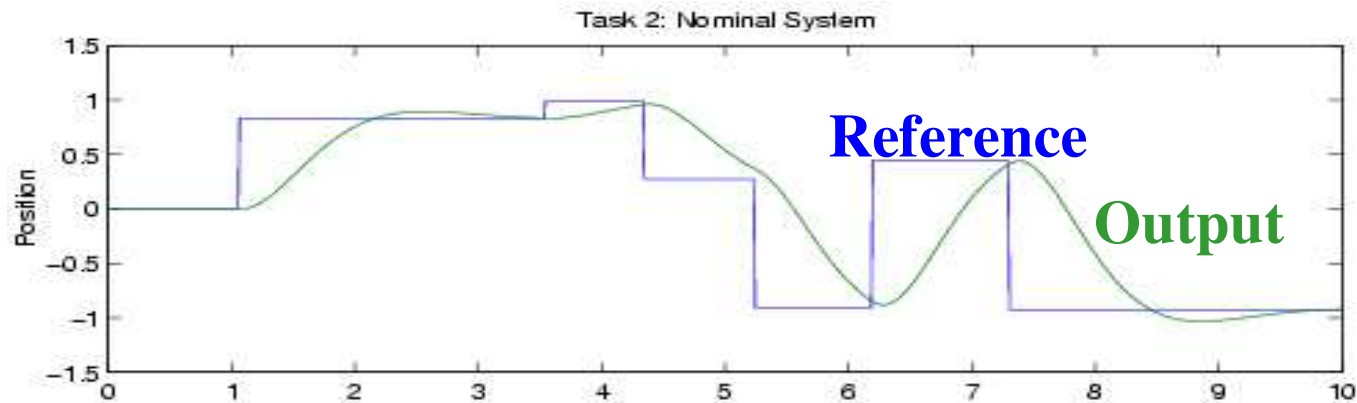
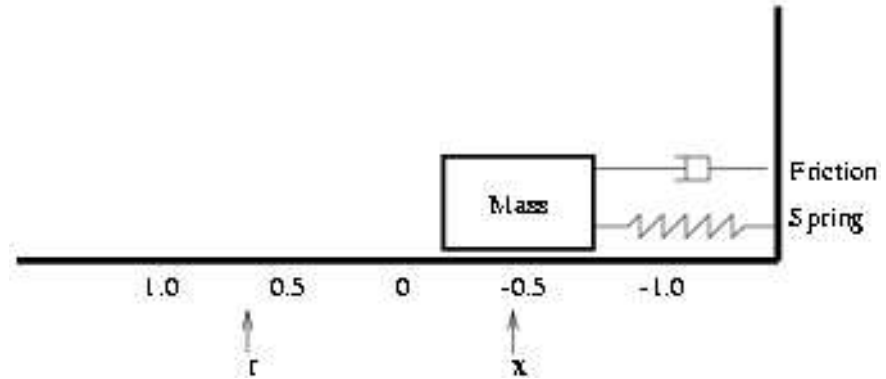
Incorporating Time-Varying IQC in Reinforcement Learning

Reinforcement learning algorithm guides adjustment of actor's weights.

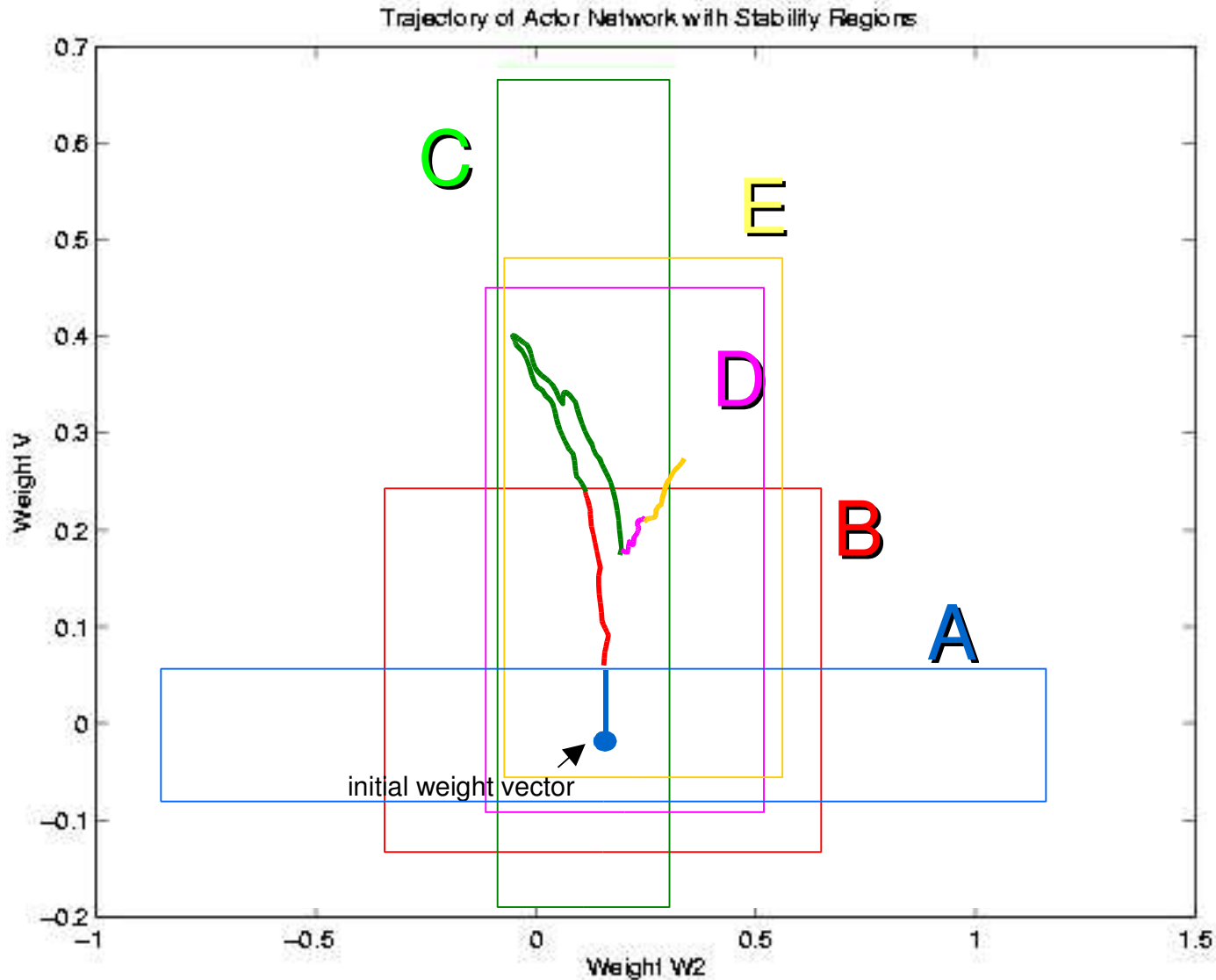
IQC places bounding box in weight space, beyond which stability has not been verified.



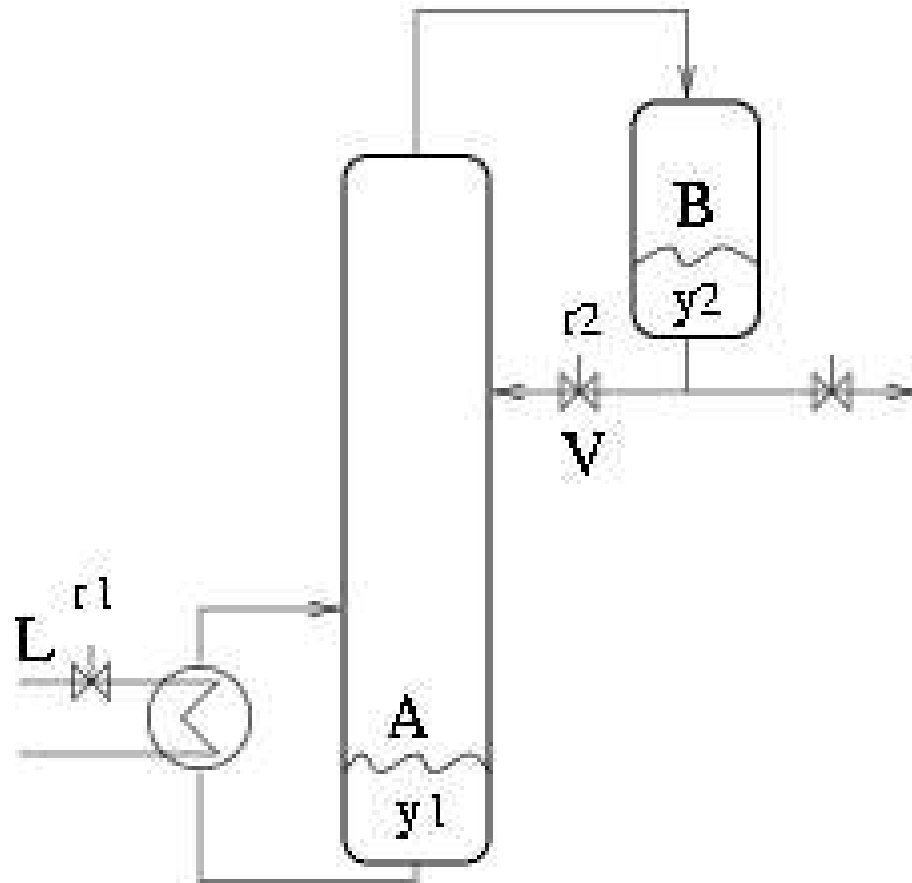
Test on Simple Simulated Task



Trajectory of Weights and Bounds on Regions of Stability



Distillation Column

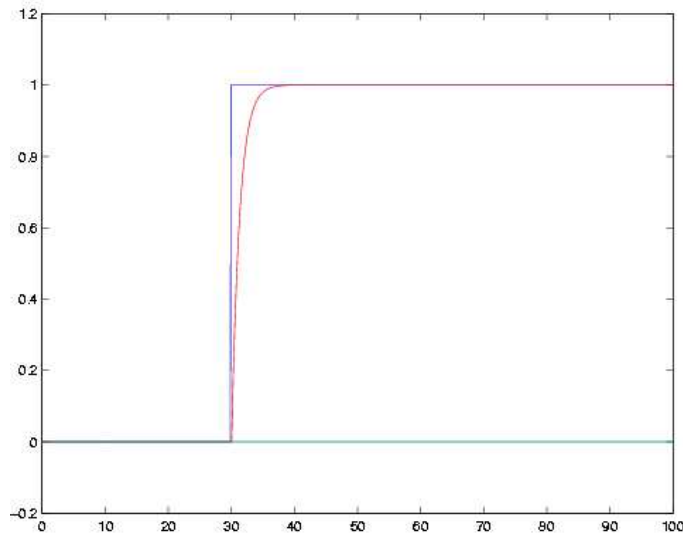


Example of task for which control variables interact in complex way.

Decoupling Controller

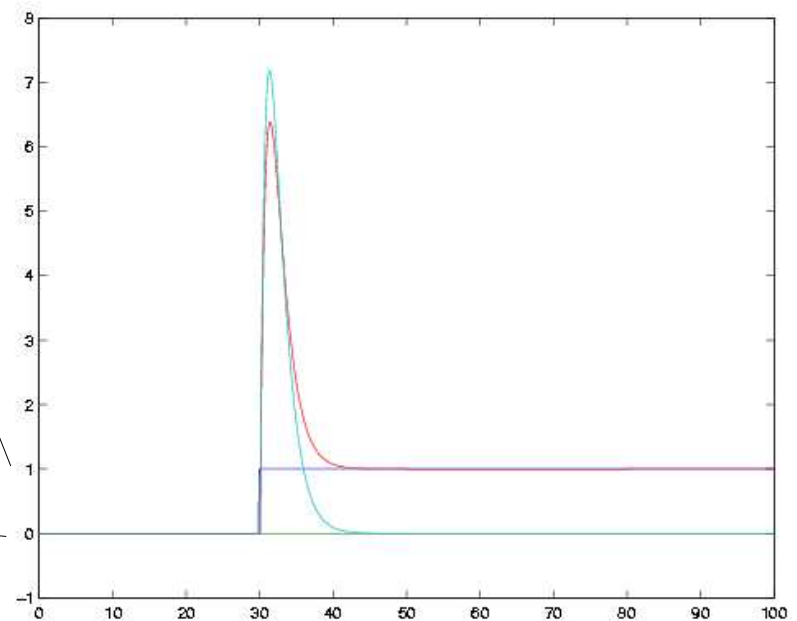
Nominal

Good response



Perturbed

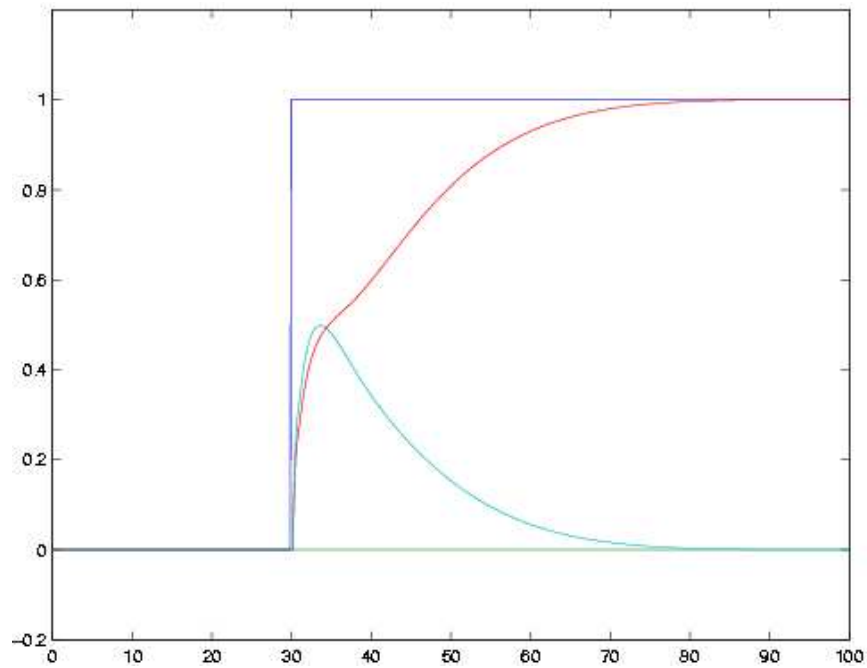
Terrible response



Robust Controller

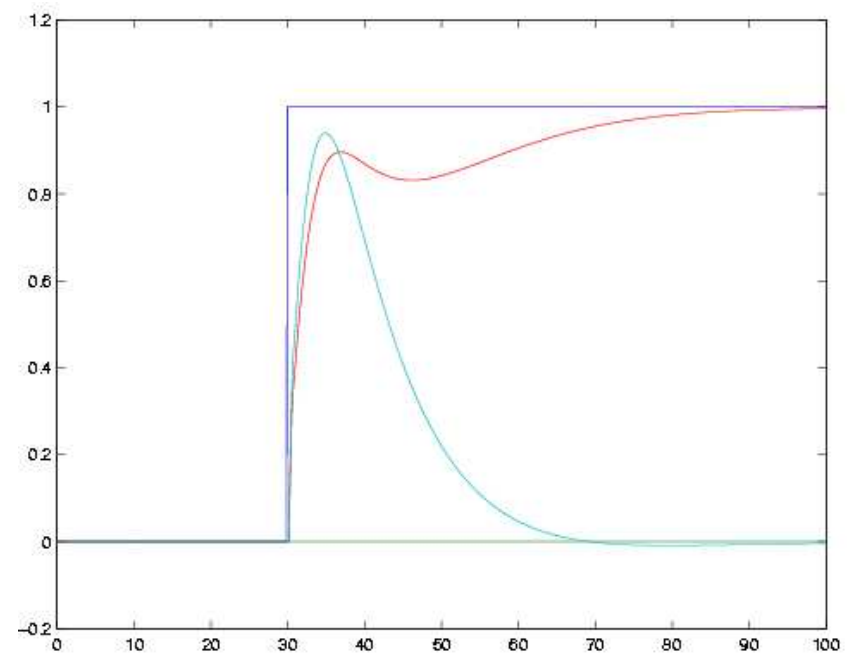
Nominal

Less aggressive response



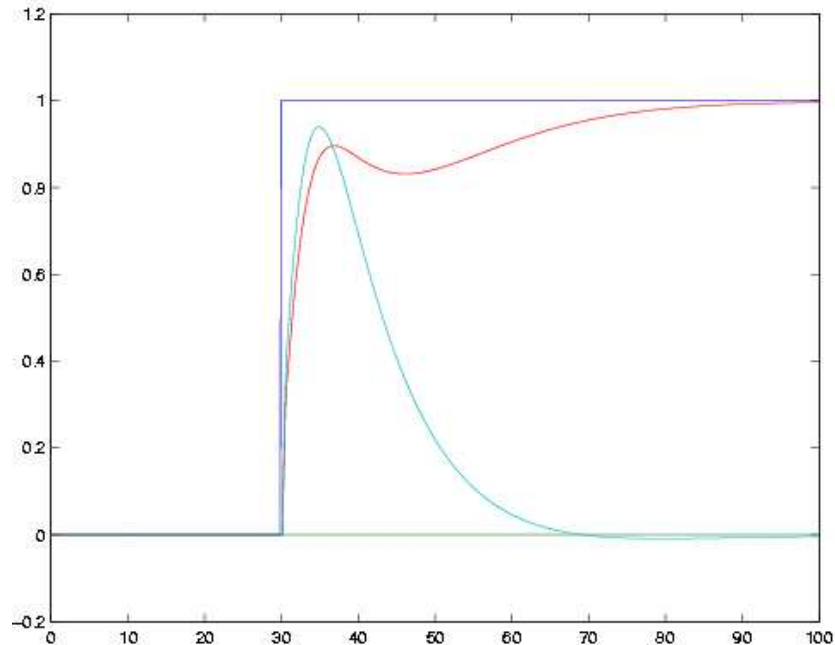
Perturbed

Much improved response

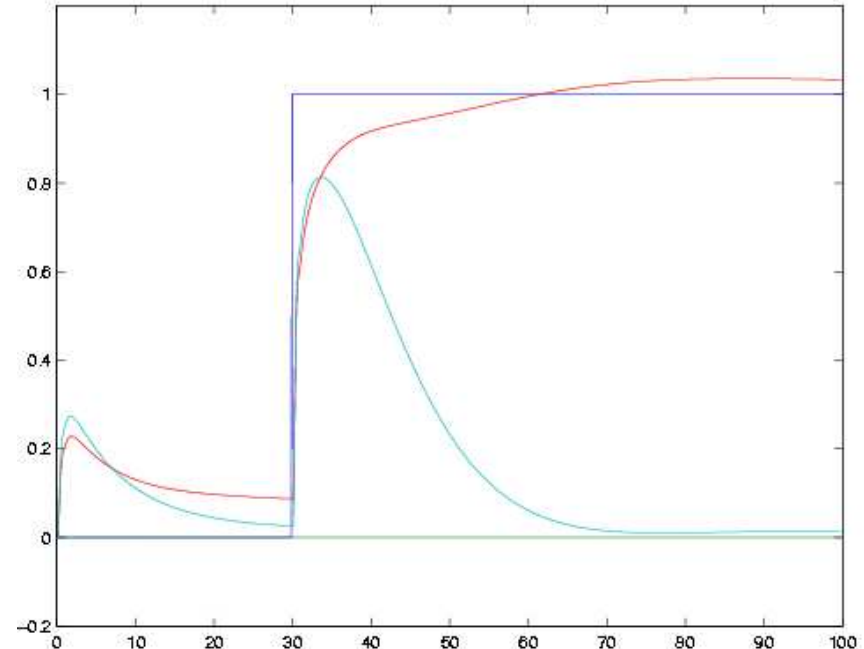


Robust Reinforcement Learning

Perturbed case, no learning
(from previous slide)



Perturbed case, with learning



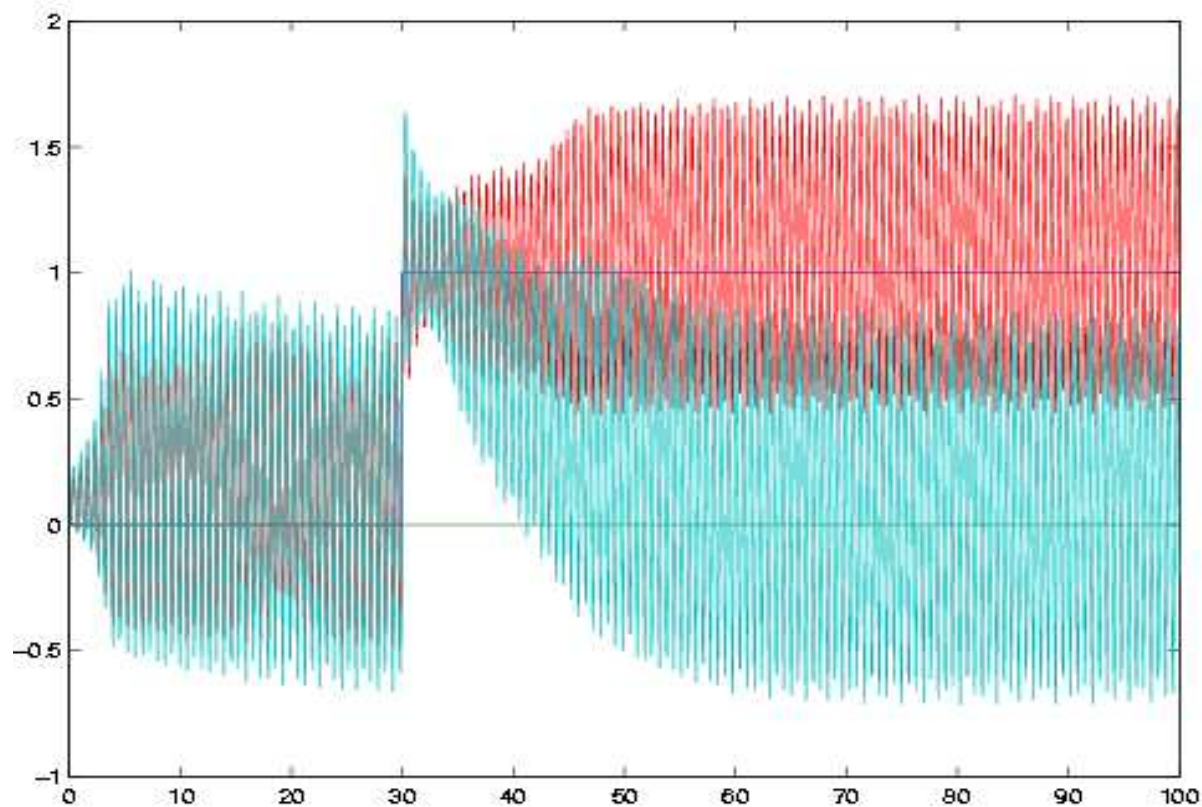
Sum Squared Error

Nominal Controller	0.646
Robust Controller	0.286
Robust RL Controller	0.243

With learning, controller has been fine-tuned to actual dynamics of real plant without losing guarantee of stability.

Reinforcement Learning without IQCs

Ultimately achieves same good performance, but during learning periods of instability occur.



Application to HVAC System: Preliminary Steps

Characteristics of Typical HVAC Systems

Energy Transfer via Heating/Cooling Coils

Air flow Regulation to Maintain Static Air Pressure

Central Water Supply Servicing Multiple Units

- **Current HVAC Systems Perform Poorly**

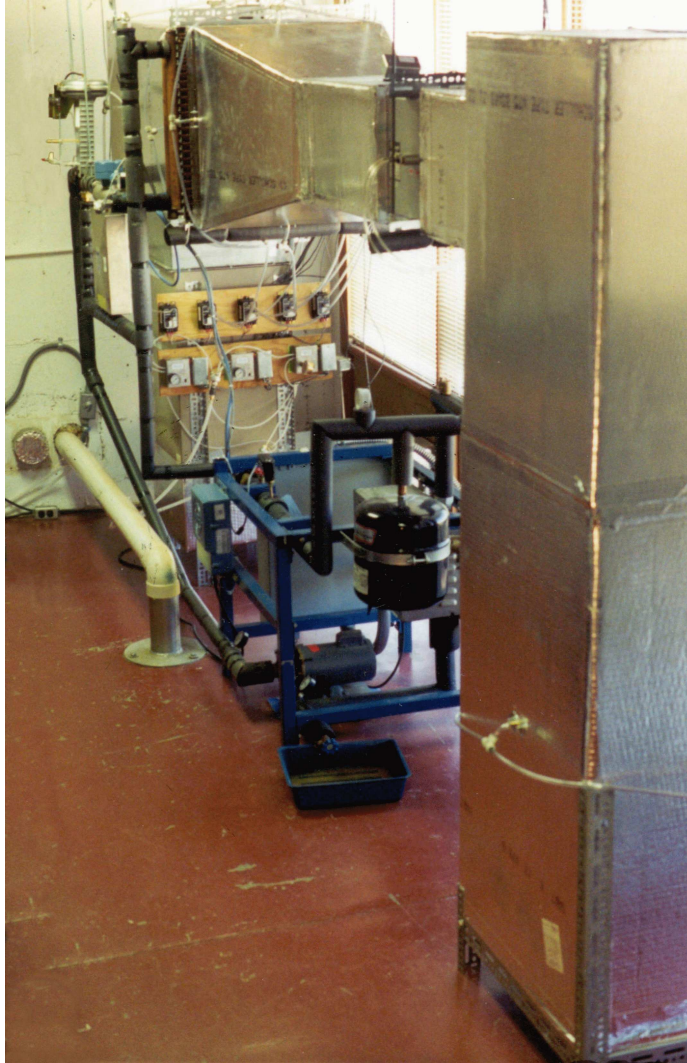
Complex Nonlinear Time-Varying System

Highly Uncertain System Dynamics

Interaction of Controlled Variables

Controlled via Multiple SISO PID Control Loops

Experimental HVAC System



Simple HVAC System

Counter Flow Hot Water
to Air Heat-Exchanger

Variable Air Volume

Mixing Box

Electric Hot Water Heater

Controlled Variables:

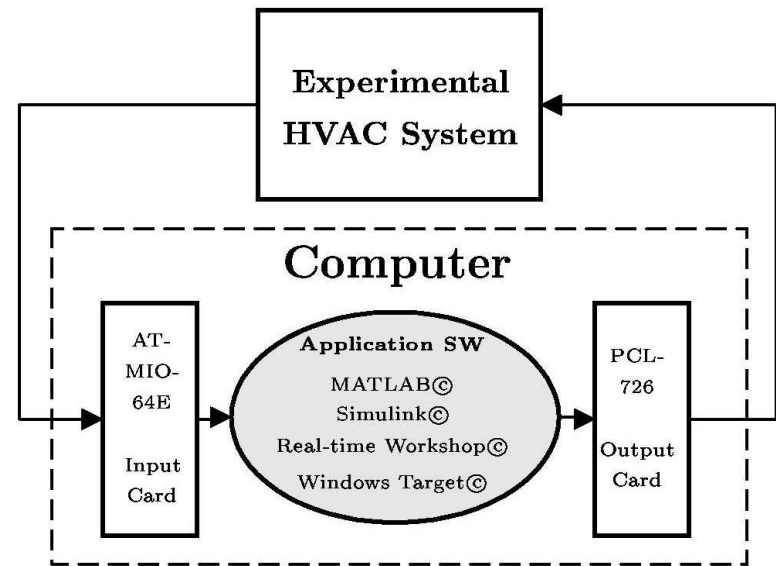
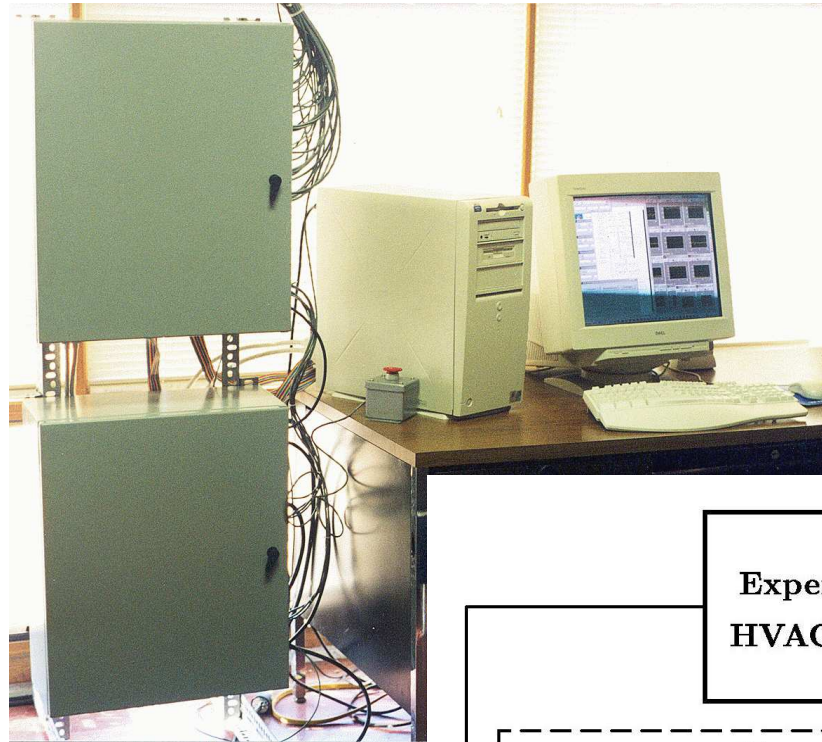
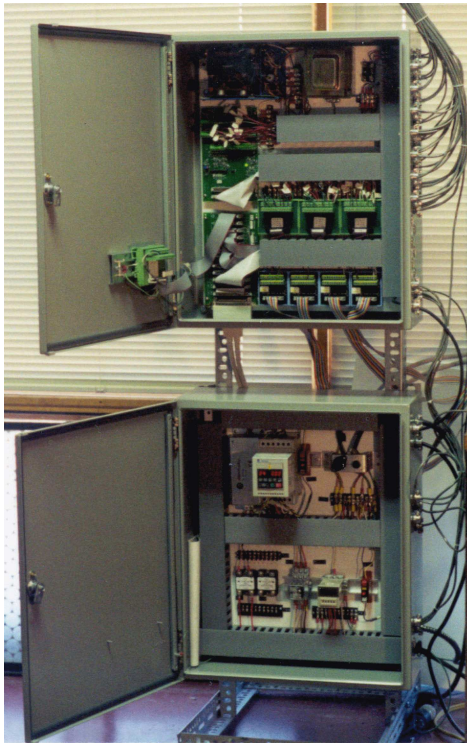
Discharge Air Temperature

Mixed Air Temperature

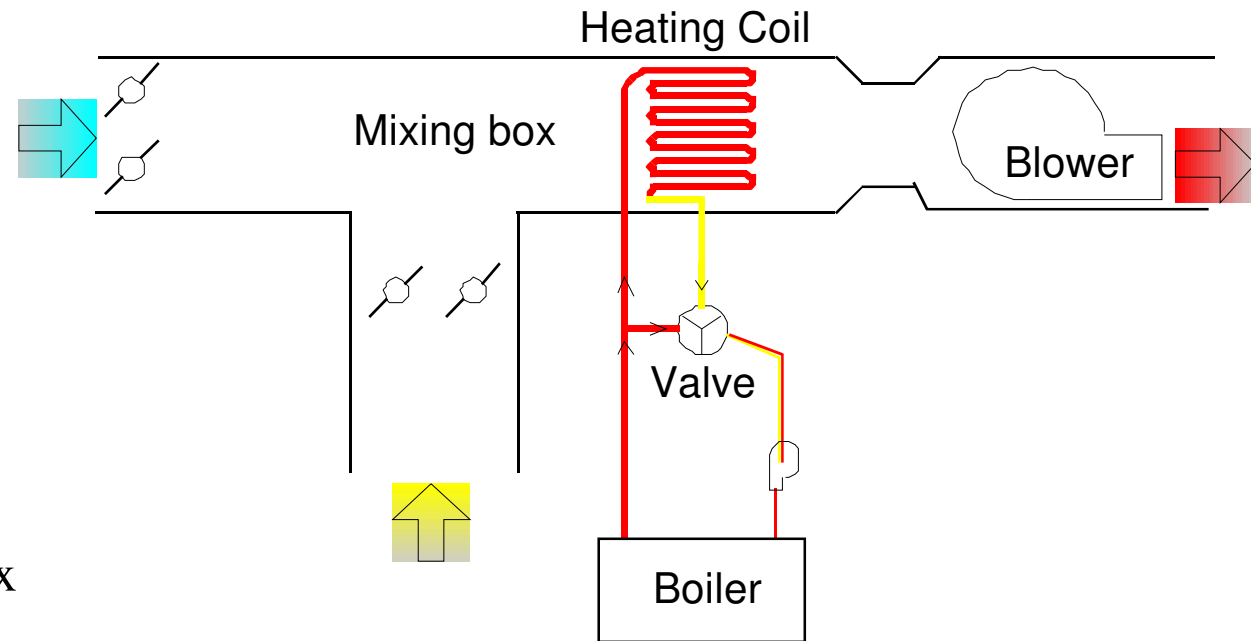
Air Flow Rate

Hot Water Temperature

PC/MATLAB Based Control System



Modeling the Experimental HVAC System



Subsystems:

Blower

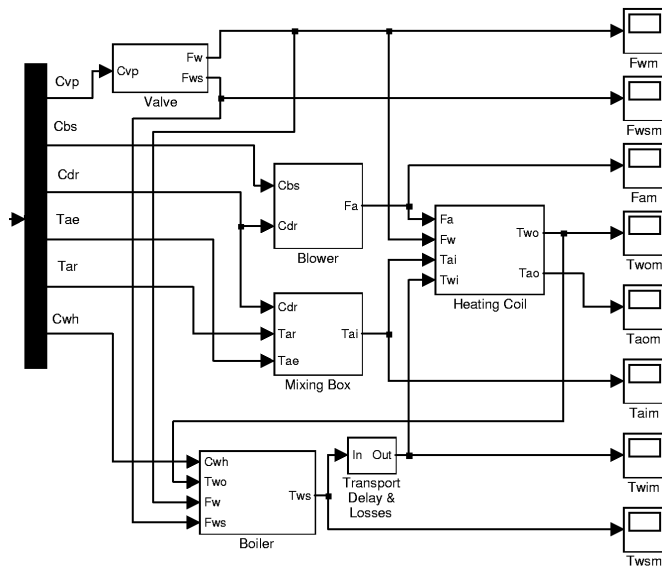
Mixing Box

Heating Coil

3-way Mixing Valve

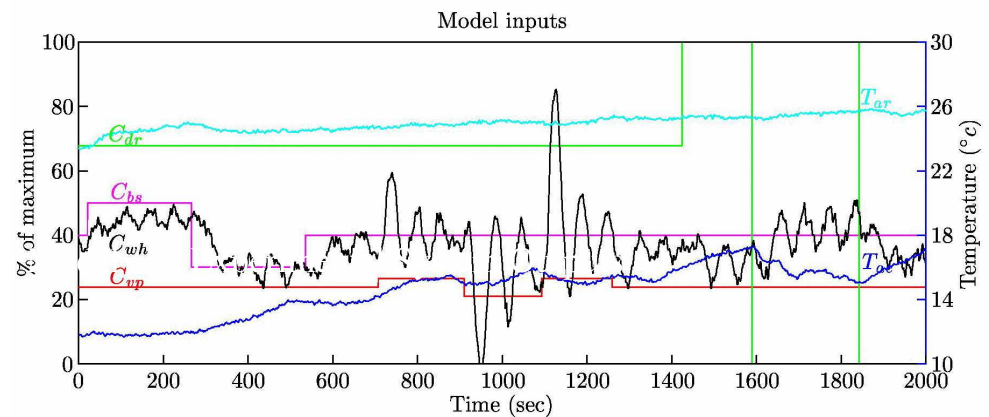
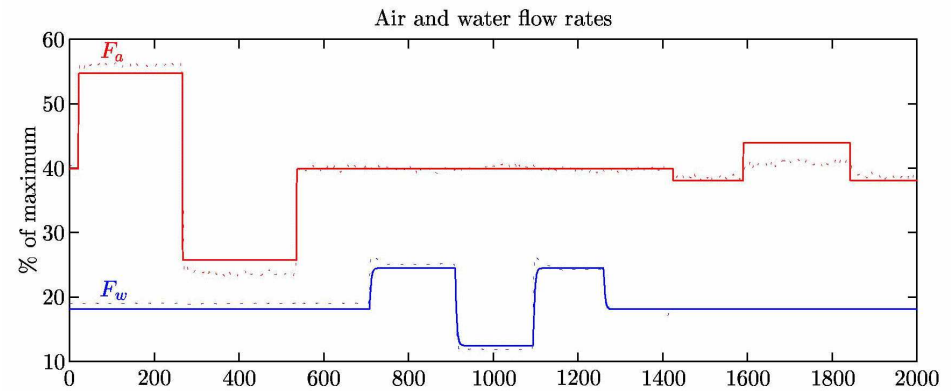
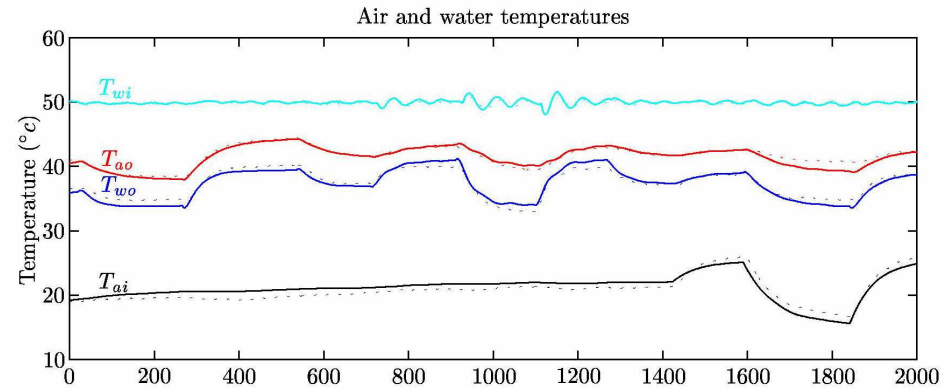
Boiler

HVAC System Model



Dynamic Model for:
 Controller Design
 Simulation Testing

Nonlinear Subsystems
 Linearization for Design
 1st Principles and Data fitting



Controller Design

- Basic Design Goals:
 - MIMO Stability and Robustness
 - Independent Control of Key System Variables
 - Discharge Air Temperature and Flow Rate

Reference Conventional PI Controller

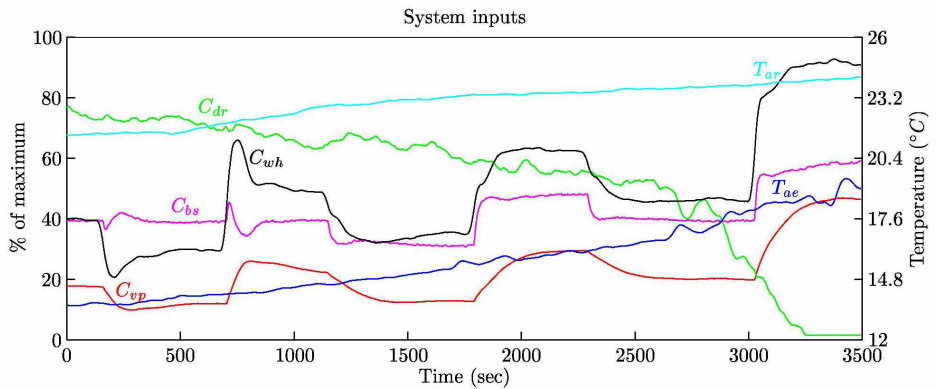
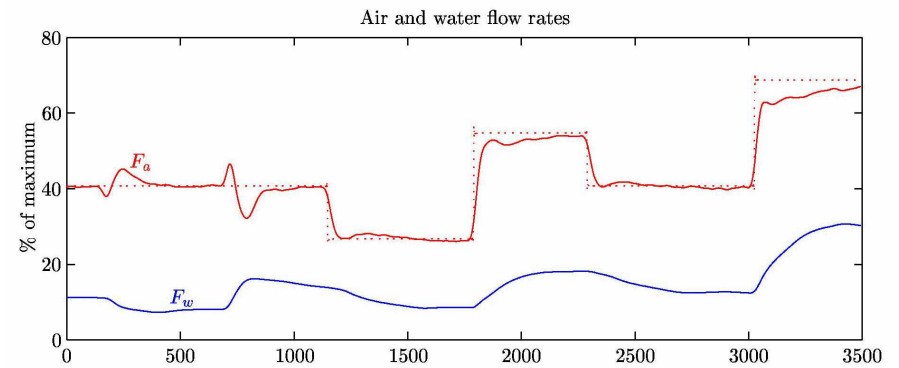
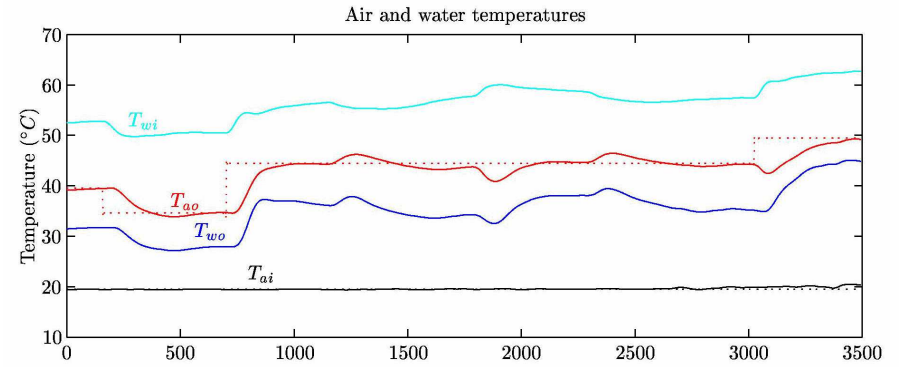
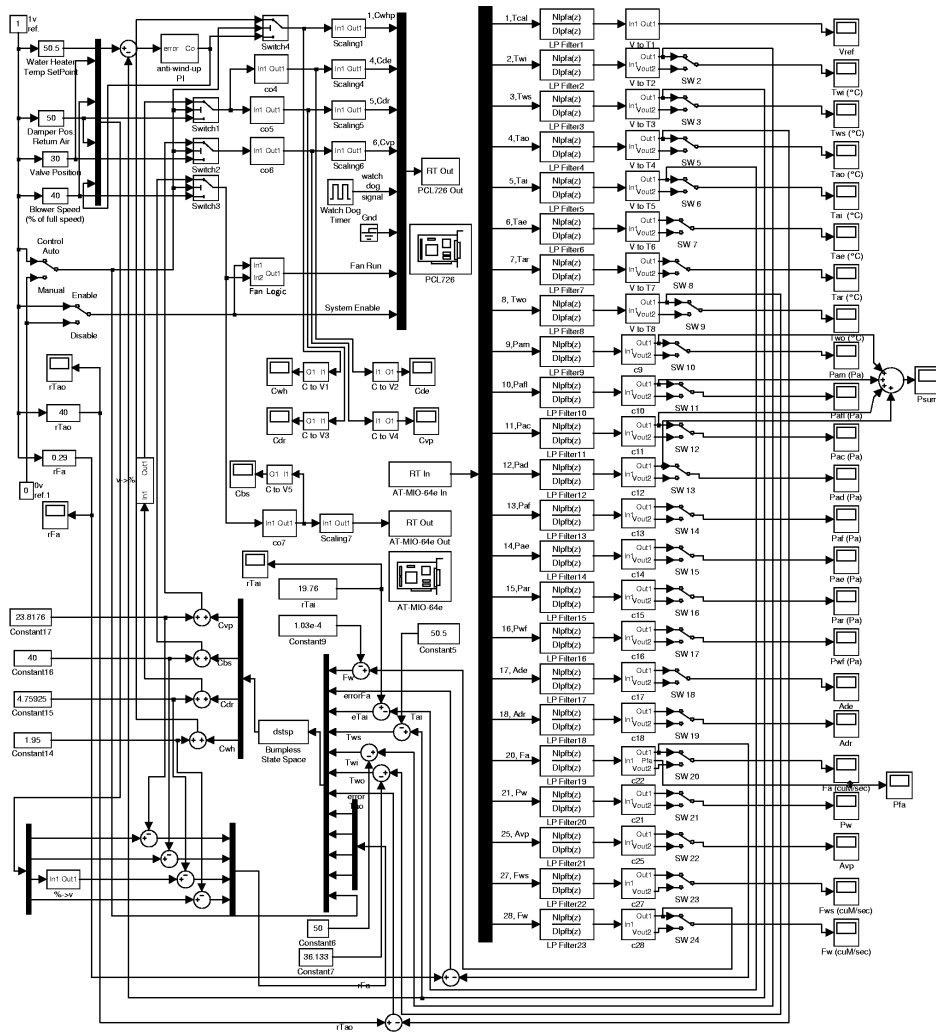
MIMO Robust Controllers:

“Minimal” (3x6) ~ T_{WS} Externally Controlled

“Constrained” (4x7) ~ T_{WS} Integrated

“Full” (4x7) MIMO

Controller K_{R3} Experimental



Conclusions

IQC bounds on parameters of tanh and sigmoid networks exist for which the combination of a reinforcement learning agent and feedback control system satisfy the requirements of robust stability theorems, for static and dynamic stability.

Resulting robust reinforcement learning algorithm improves control performance while avoiding instability on several simulated problems.

Current Work

Applying robust reinforcement learning to HVAC model and real HVAC system.

Developing continuous versions of reinforcement learning.
Continuous state, action needed for high-dimensional control problems

Investigating value-gradient method (based on Werbos' heuristic dynamic programming, 1987).
Uses known or learned model of system dynamics.
May result in faster learning.

Planned Work

Can similar bounds be placed on other activation functions?

Directly add robust constraints to function being optimized by reinforcement learning.

Extend theory and algorithms to include dynamic, recurrent neural network as actor. (Barabanov and Prokhorov, 2002)

Measured variables from system may not fully represent state of the system.

Recurrent net can learn a state representation.

Investigate alternative ways of quickly adapting the internal representation of the neural network.

Evaluate with more complex control systems.

Future Directions

Dissemination into Industry

Implementation of Robust Learning Control on MIMO HVAC System

Large Scale Experimental Platform

Gain–Scheduled Controllers

Nonlinear Modeling – PDE Approach

Robust Reinforcement Learning Control Theoretical Advances

Advanced Robust Learning Algorithms

Other Topics to Discuss

Direct–gradient policy learning (no value function)

Multigrid approach to learning value function.

Learning neighborhoods of temporally–related states.

Hierarchical policies based on recurrent neural networks.

(Additional slides for further discussion.

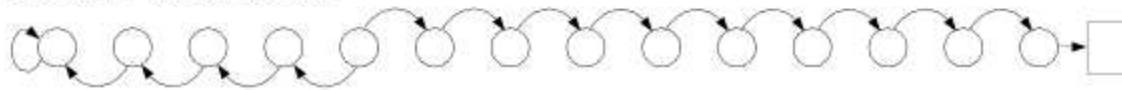
Details can be found at www.cs.colostate.edu/~anderson)

Approximating a Policy Can Be Easier Than Approximating a Value Function

Action A Transitions



Action B Transitions



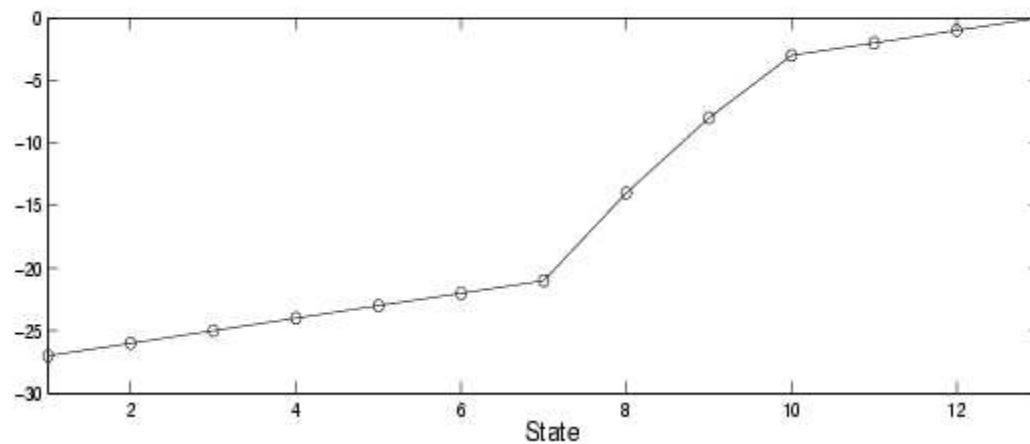
State Reinforcement

-1 -1 -1 -1 -1 -1 -1 -7 -6 -5 -1 -1 -1 -1 0

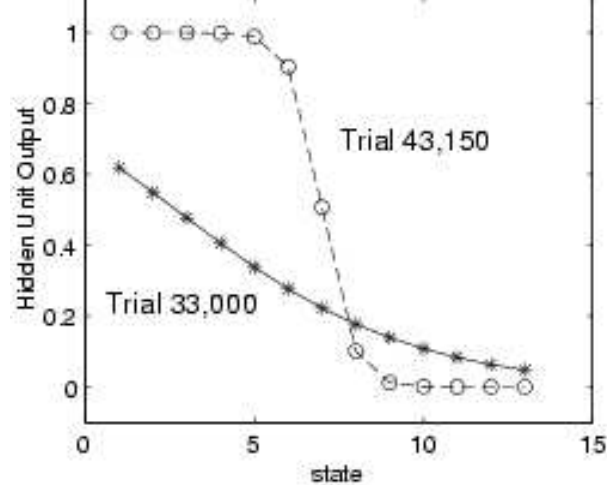
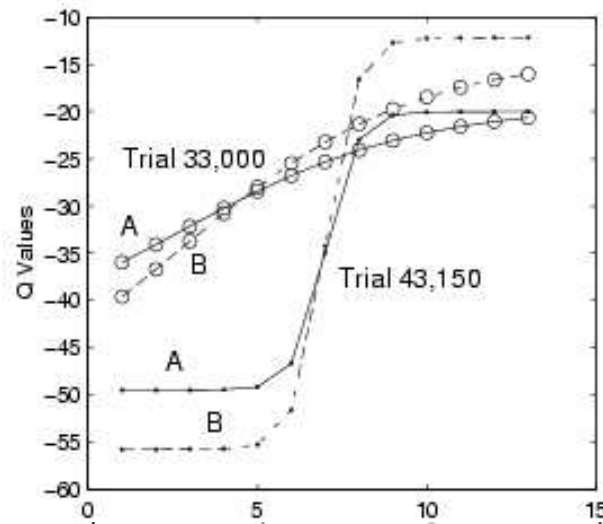
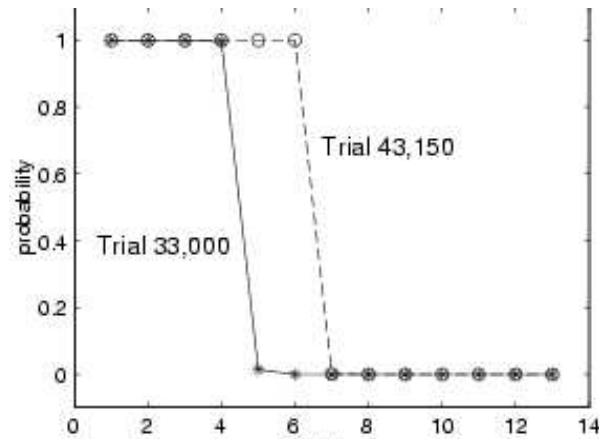
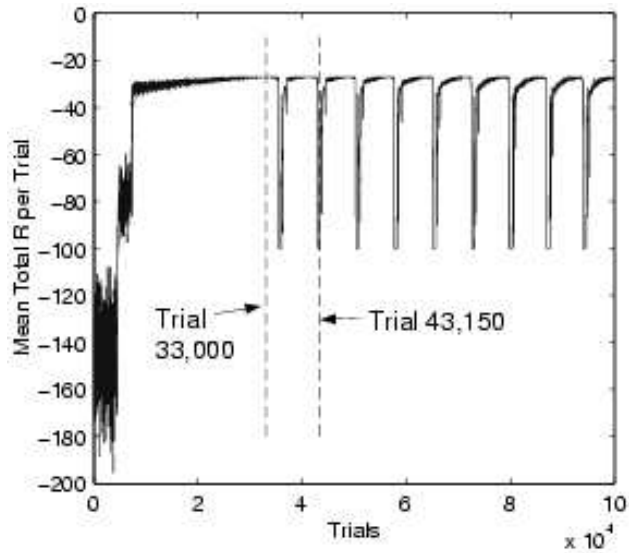
Optimal Policy

A A A A B B B B B B B B B

Max State-Action Value for Optimal Policy



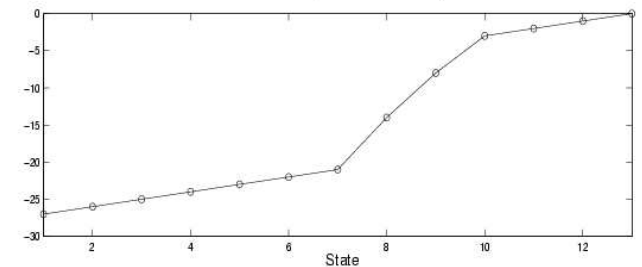
Q-Learning with One Hidden Unit



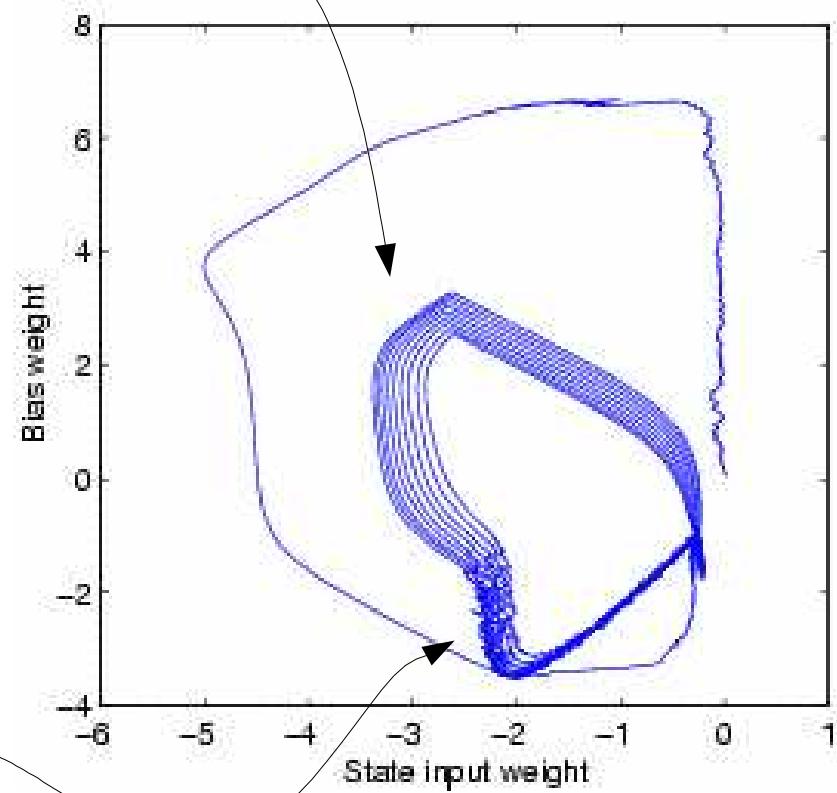
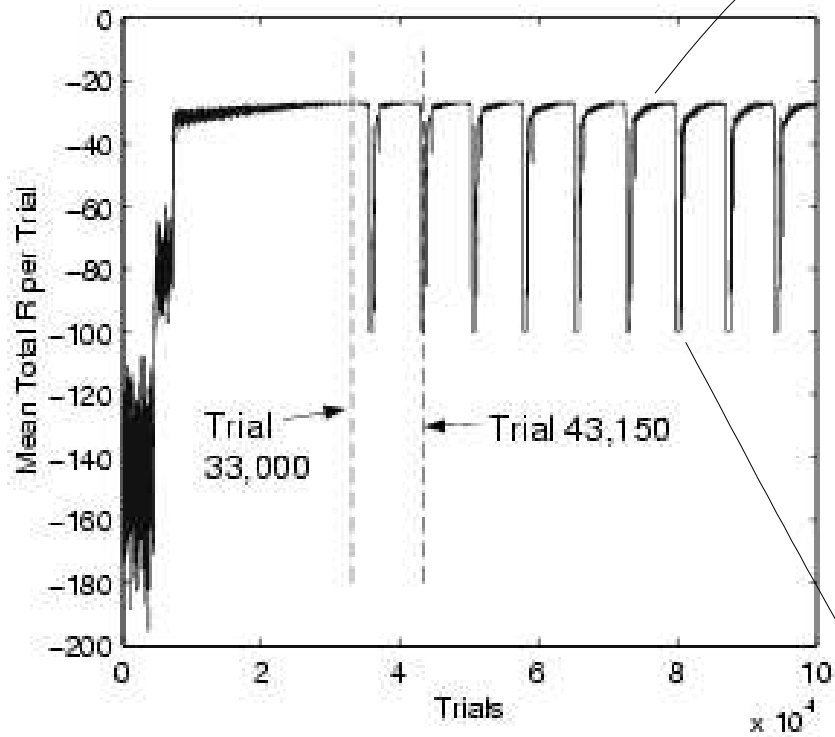
Optimal Policy

A A A A B B B B B B B B B

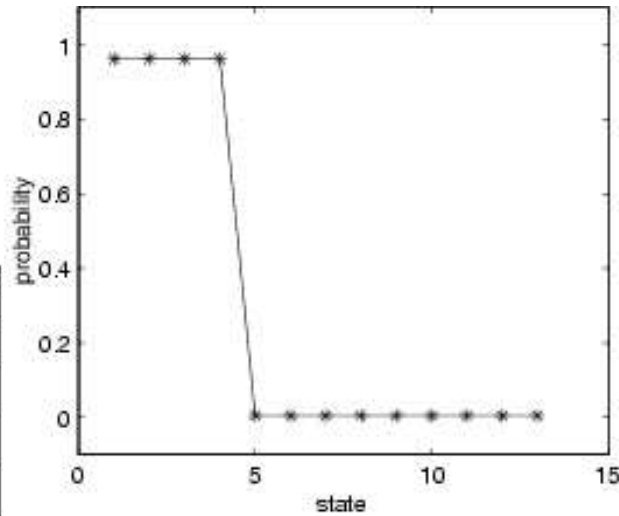
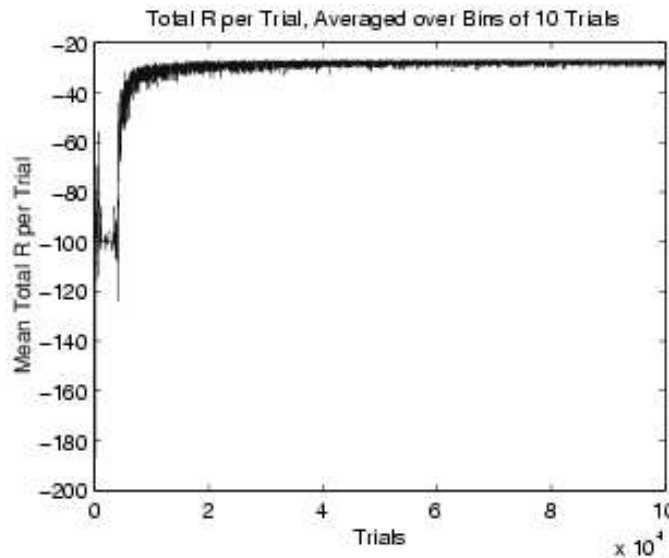
Max State-Action Value for Optimal Policy



Oscillation of Weights in Hidden Unit



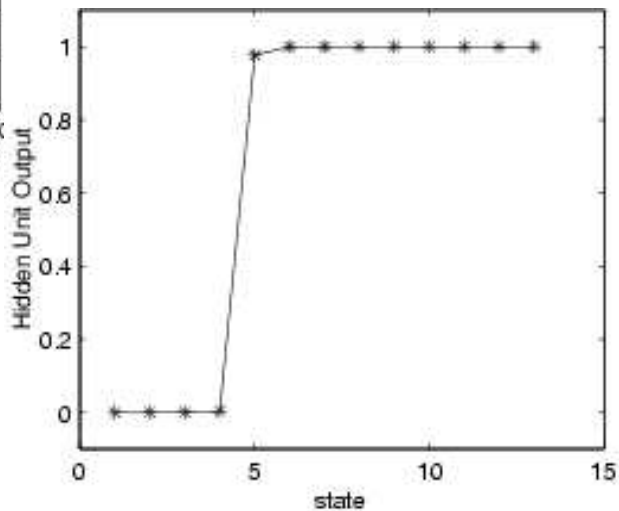
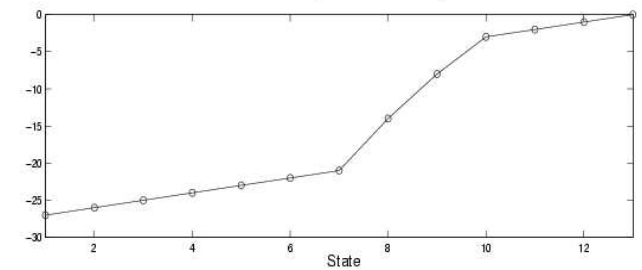
Direct-Gradient Policy Learning with One Hidden Unit (Baxter and Bartlett)



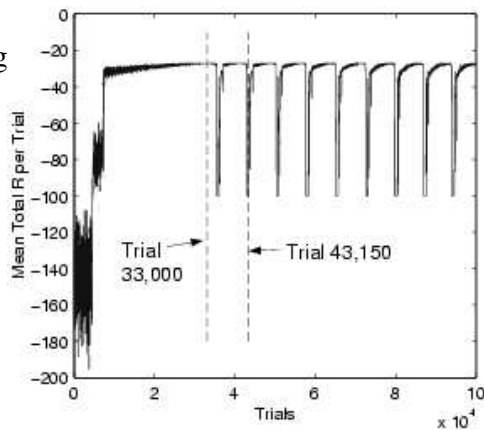
Optimal Policy

A A A A B B B B B B B B B

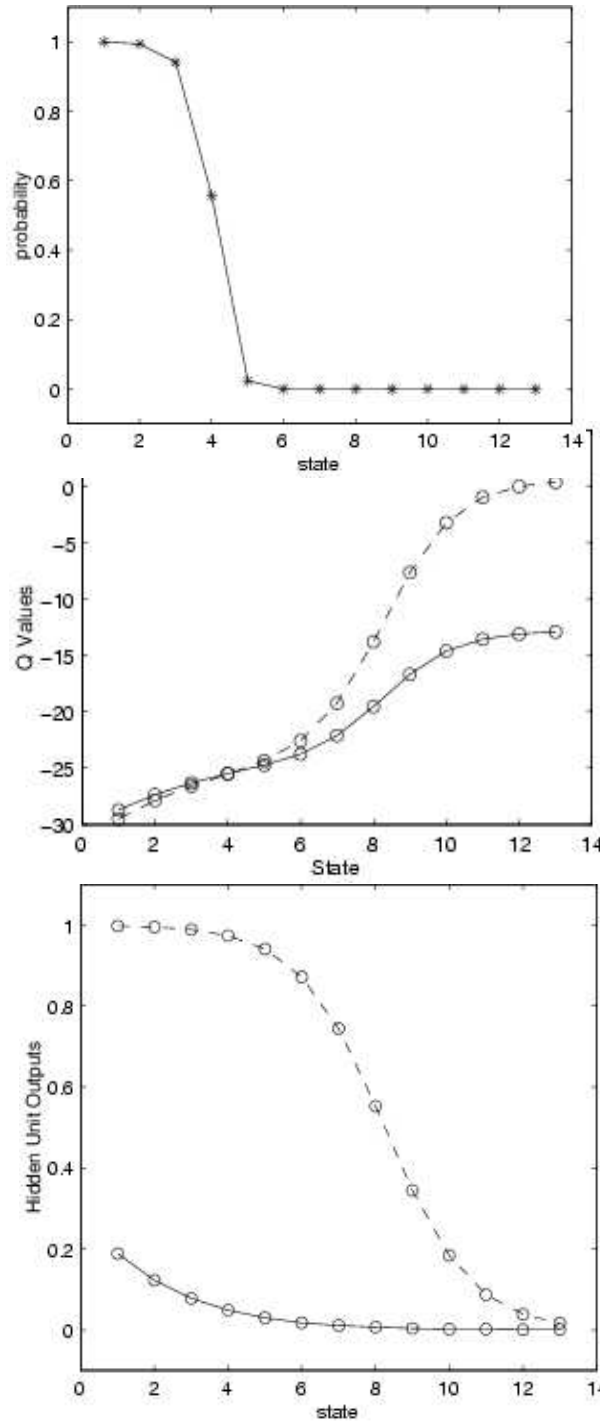
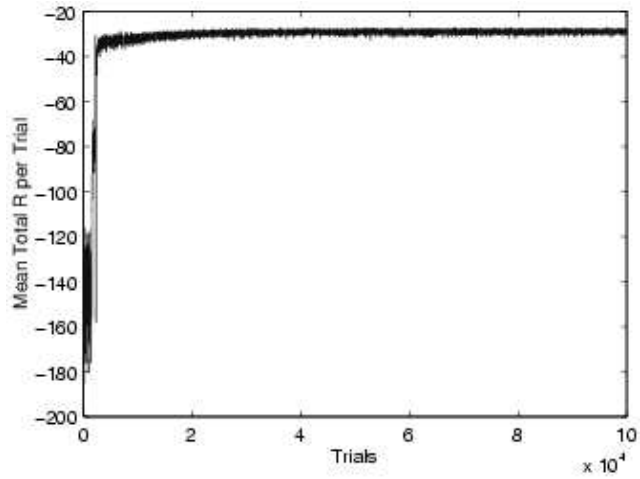
Max State-Action Value for Optimal Policy



Q-learning



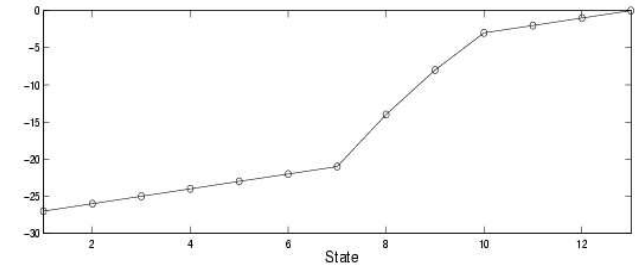
Q-Learning with Two Hidden Units



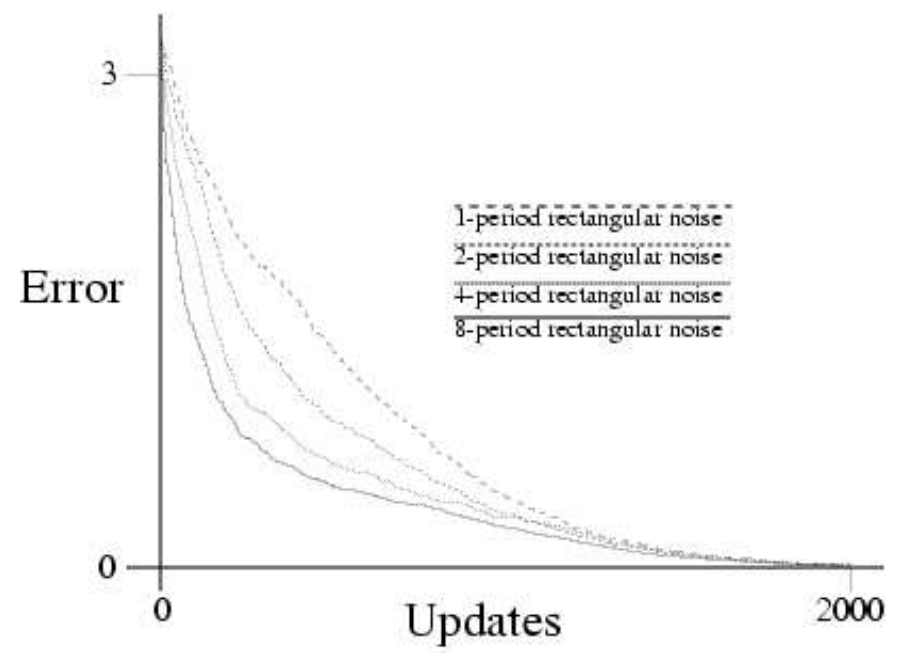
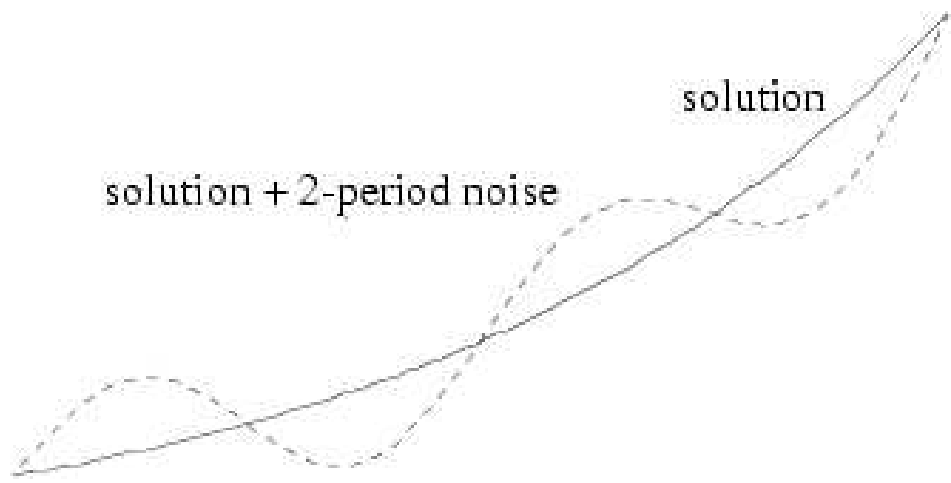
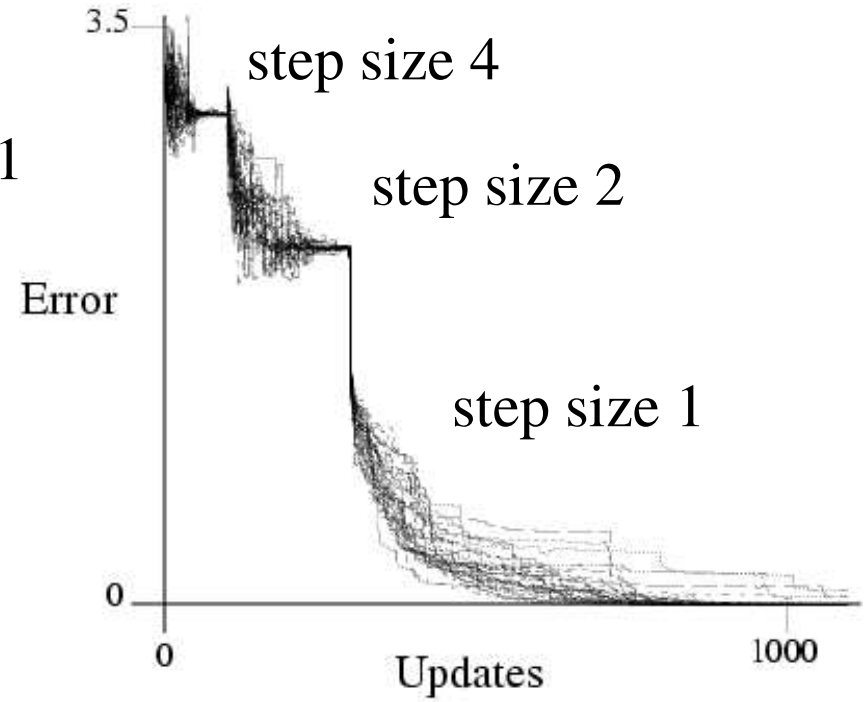
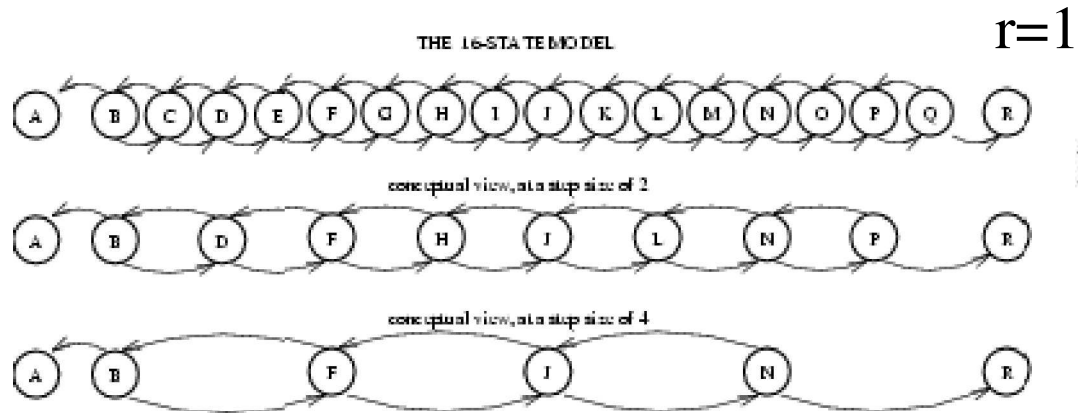
Optimal Policy

A A A A B B B B B B B B B B

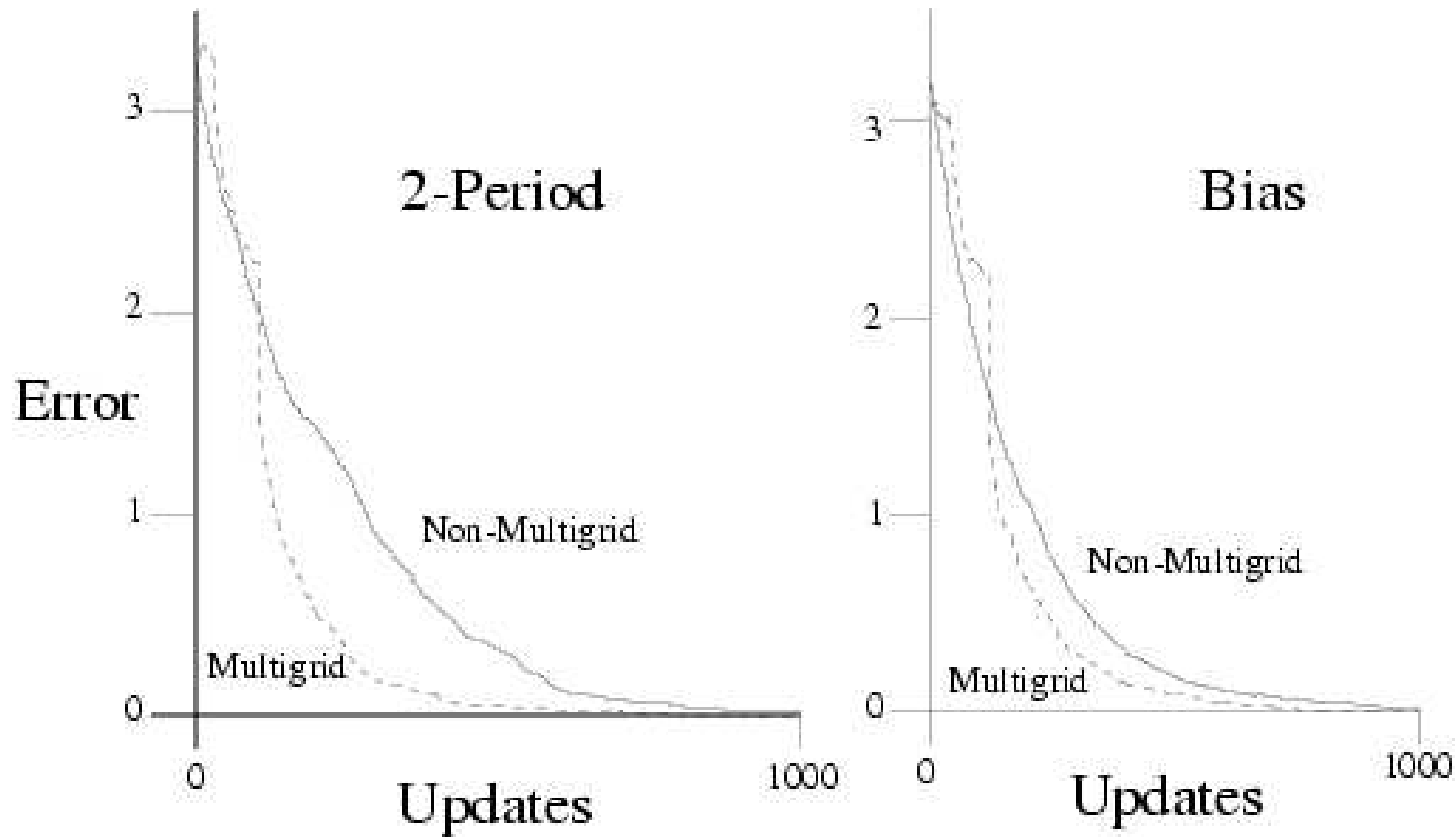
Max State-Action Value for Optimal Policy



Multigrid: Simple Markov Chain



Multigrid (with 4/2/1 schedule) reduces error faster than non-multigrid. Effect is stronger with more variation in sign of value function error.



Multigrid Value Iteration Applied to the Mountain Car Task

State space discretized into 32×32 disjoint cells.

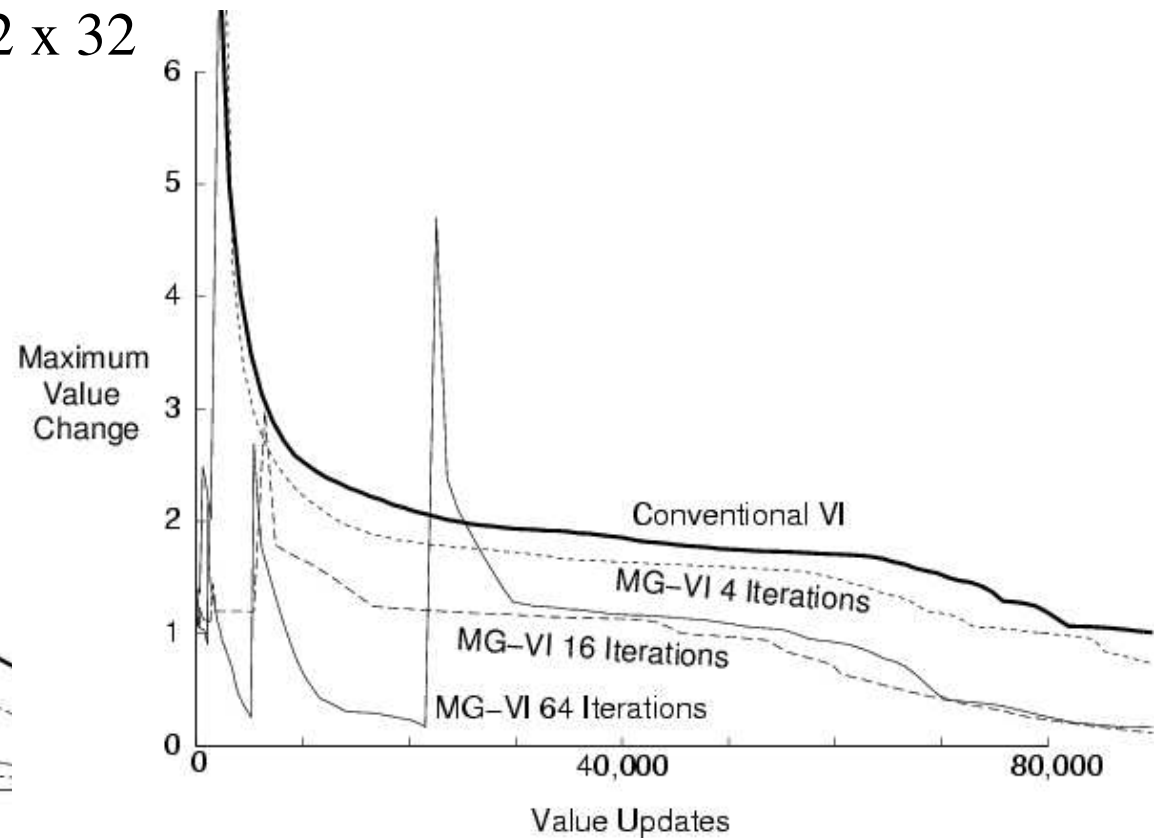
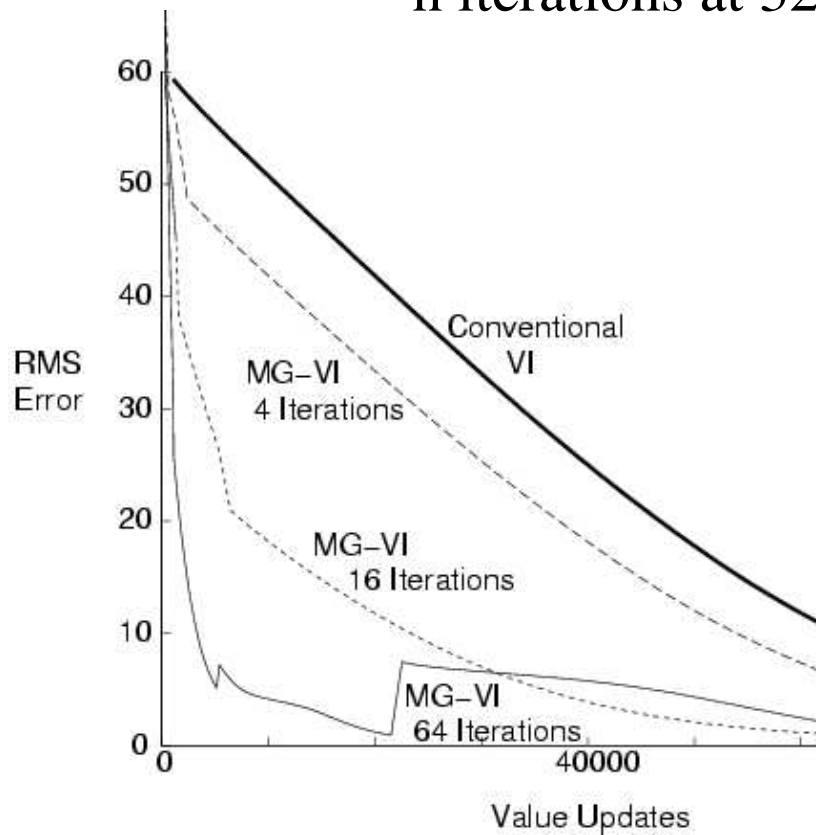
Schedule: n iterations at 2×2

n iterations at 4×4

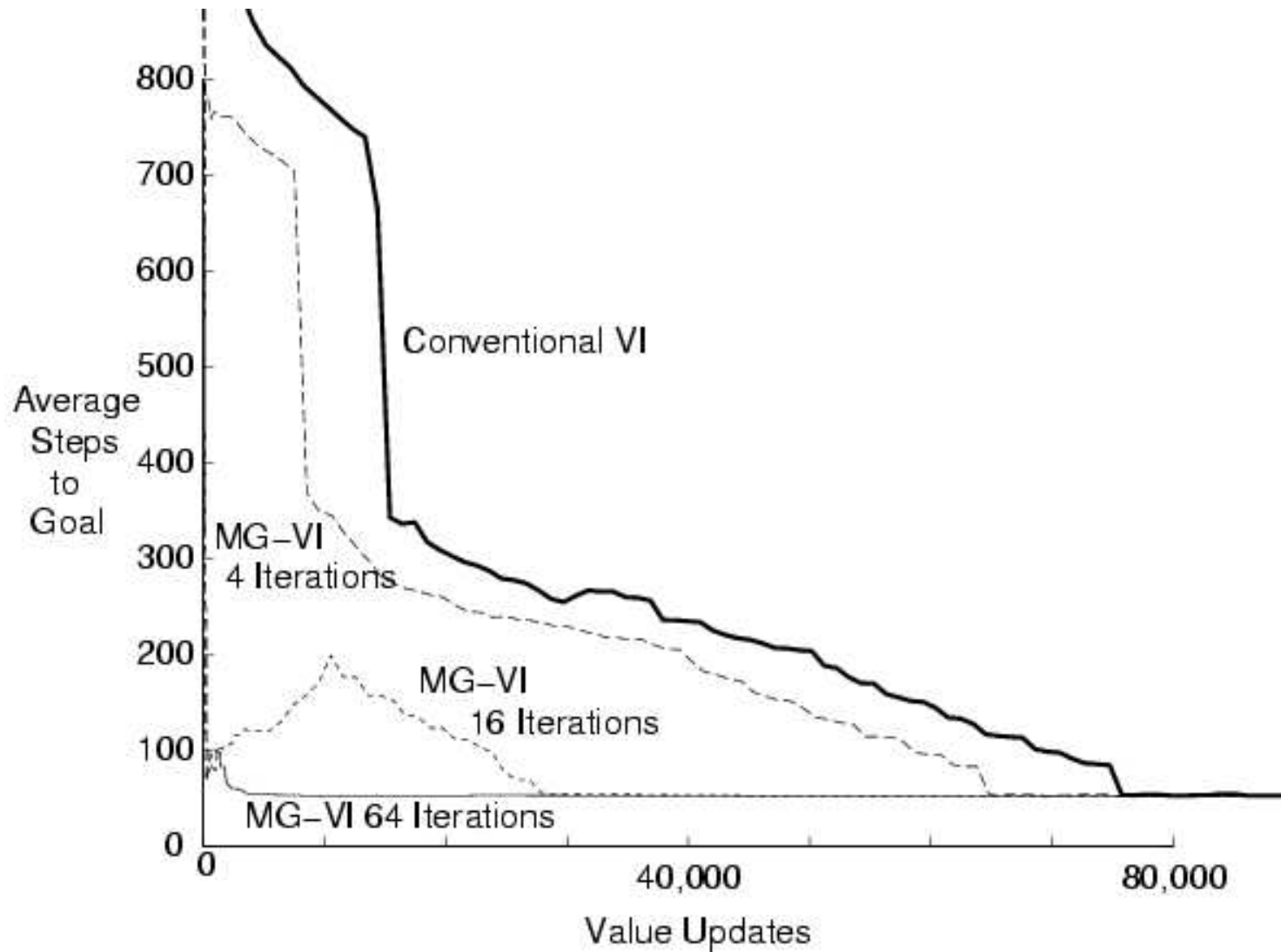
n iterations at 8×8

n iterations at 16×16

n iterations at 32×32



Steps to Reach Goal versus Updates



Multigrid Relies on Discretization of Space

Does not scale well to higher dimensions.

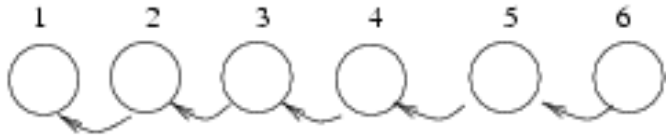
Need way to develop discretizations at coarse and fine levels based on experience.

How should experienced states be grouped?

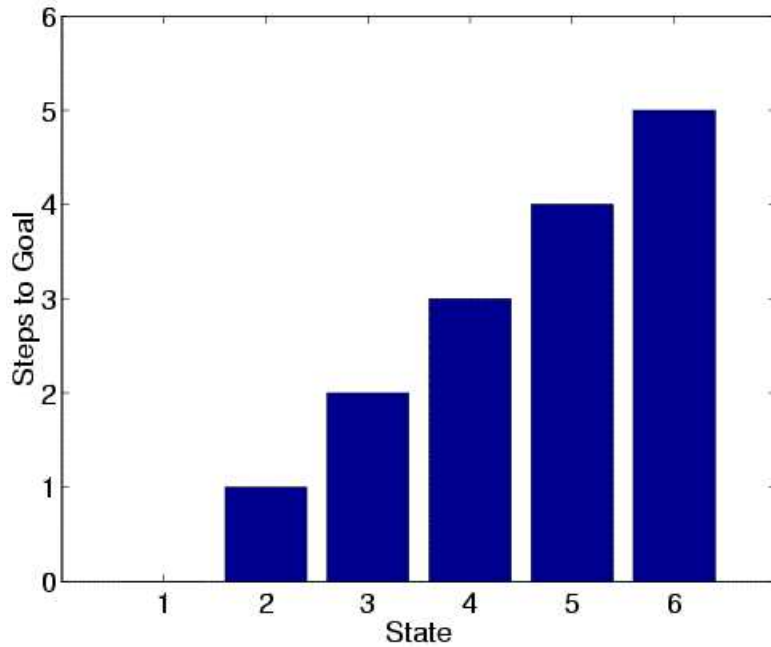
spatially?

temporally?

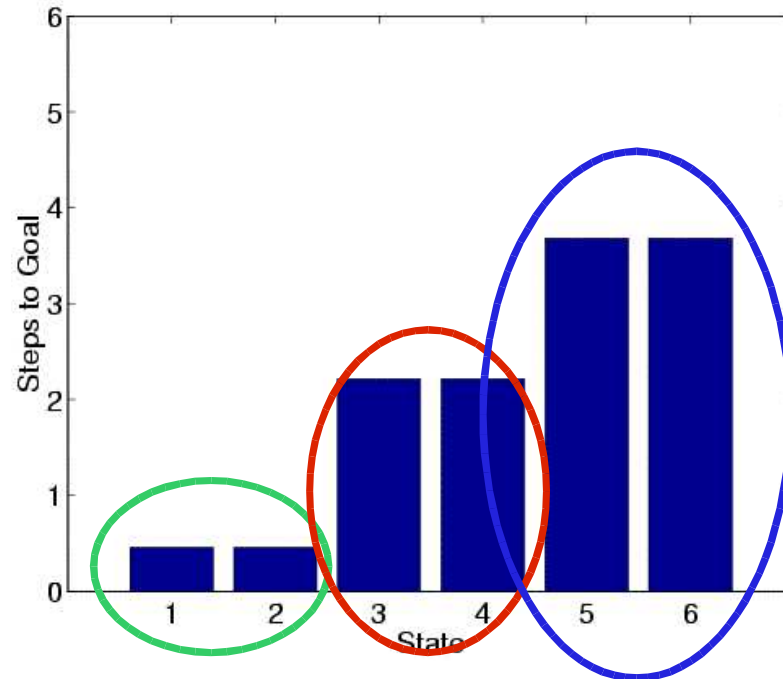
Grouping States Spatially versus Temporally

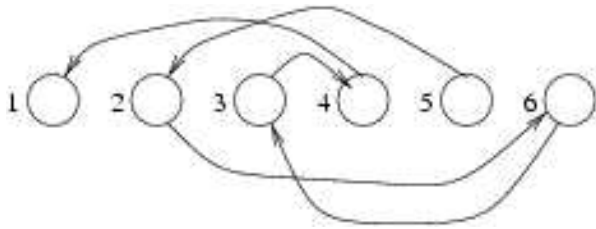


Optimal Value Function

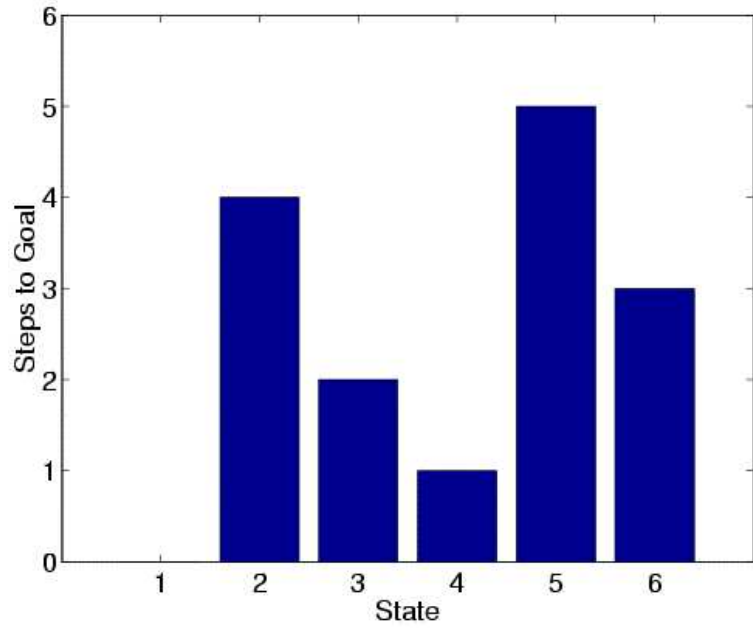


Value Function: FFA-3

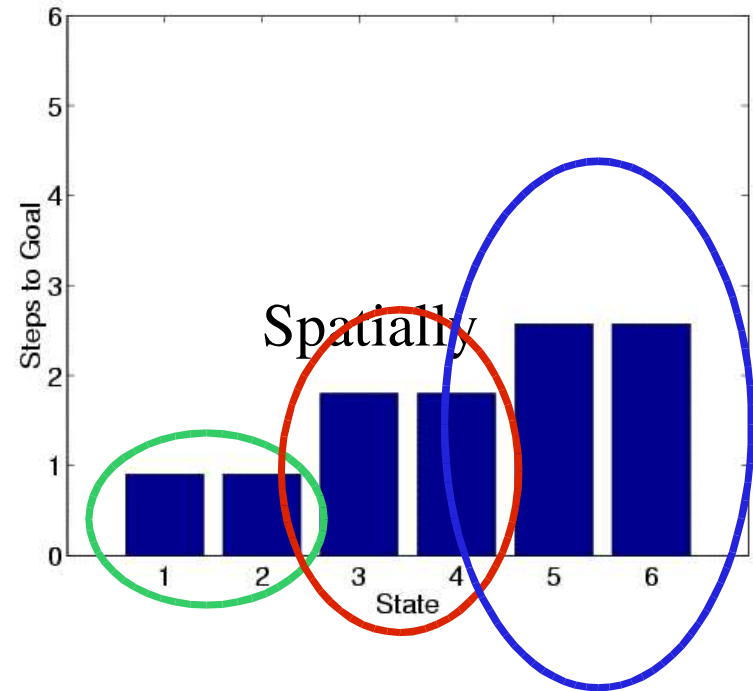




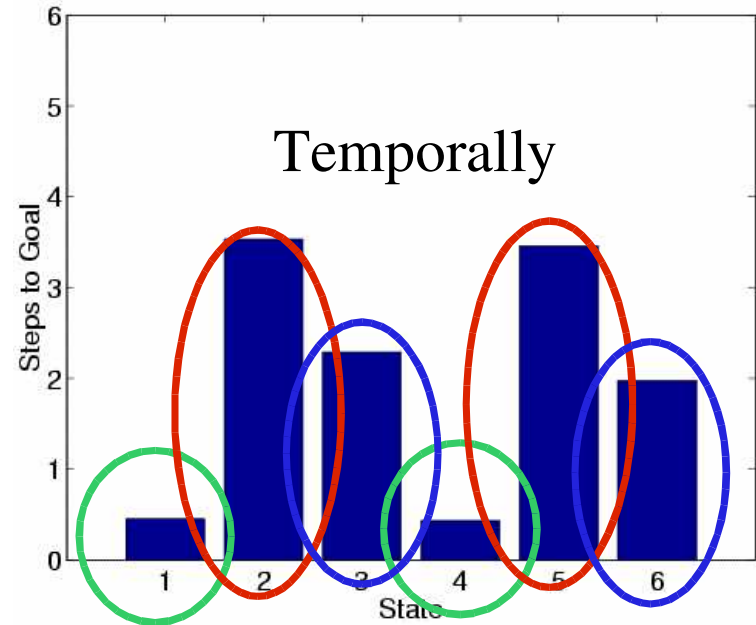
True Value Function

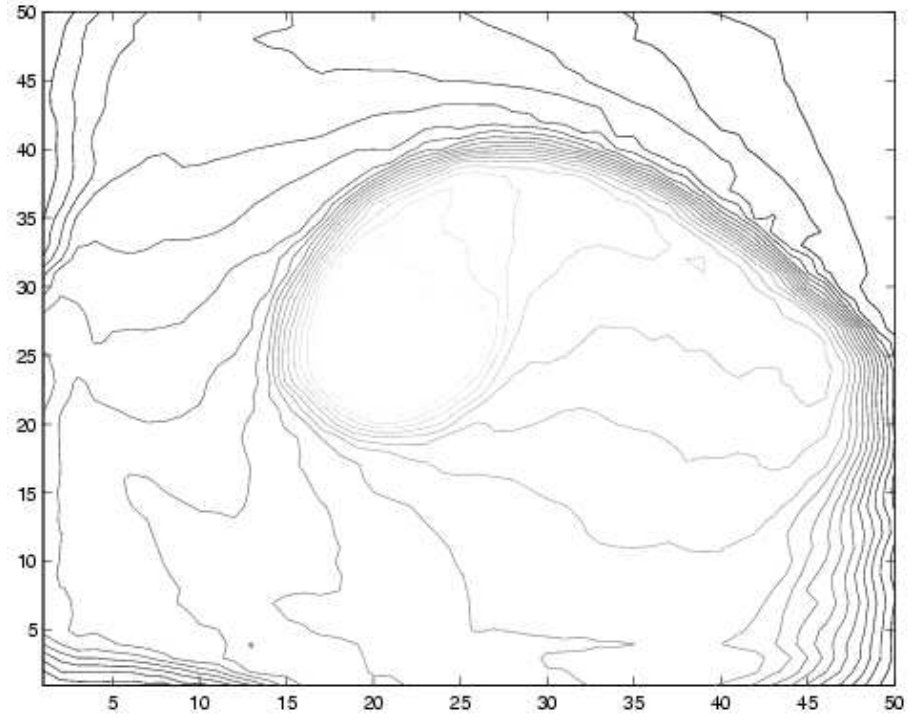
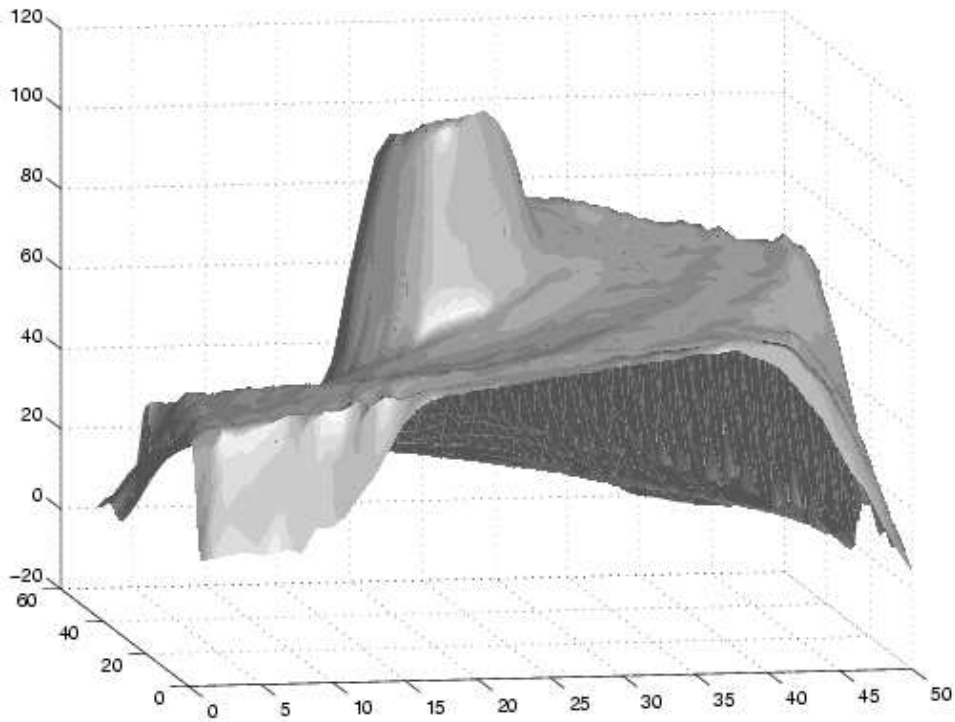
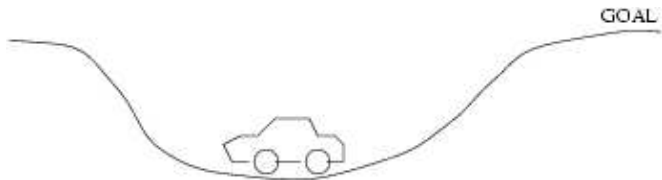


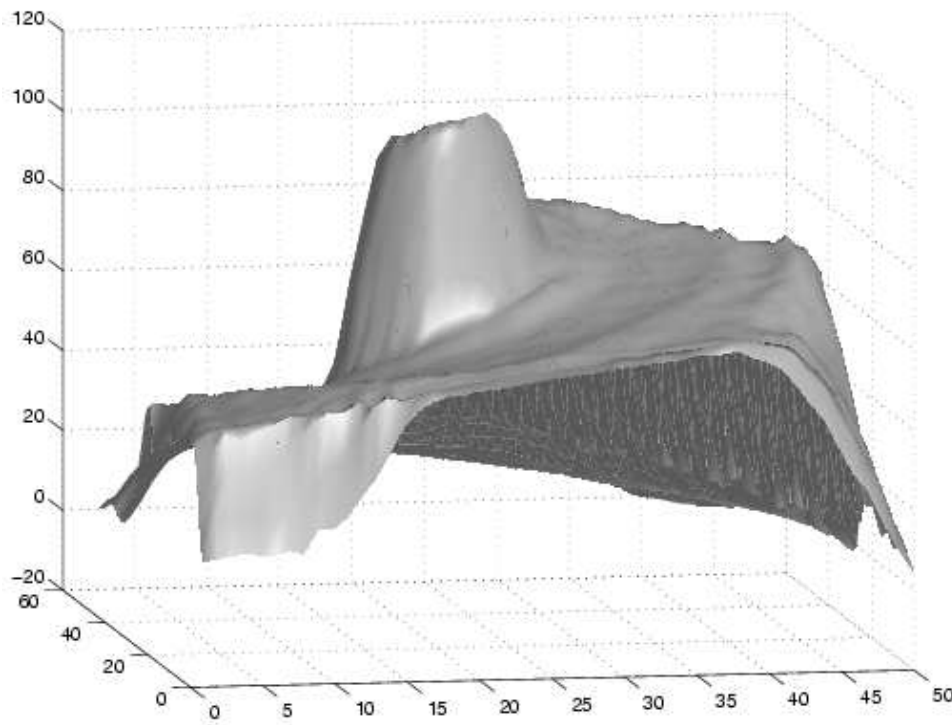
Value Function: FFA-3



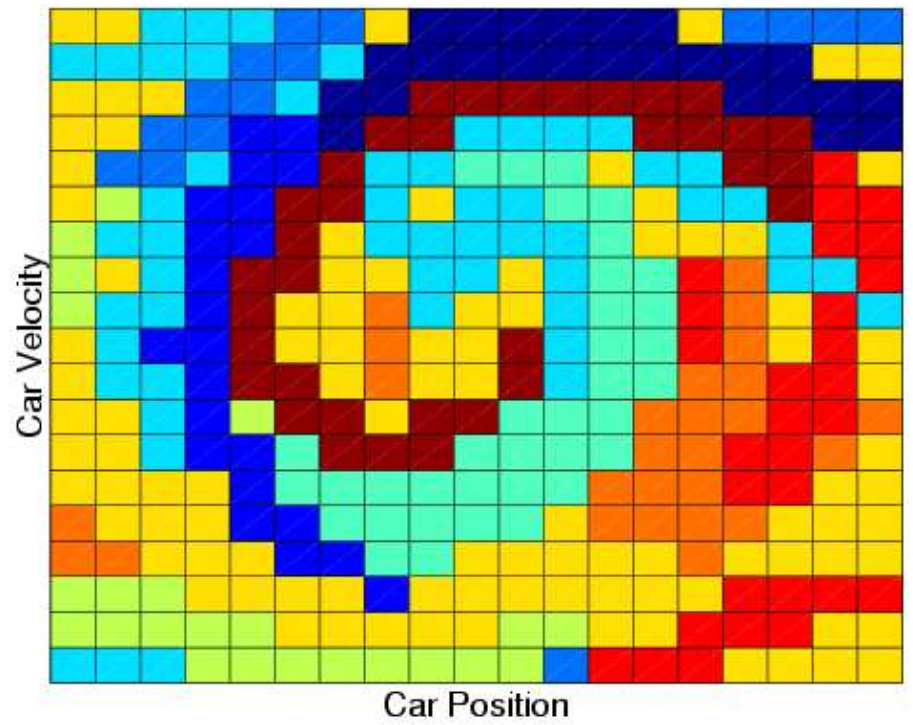
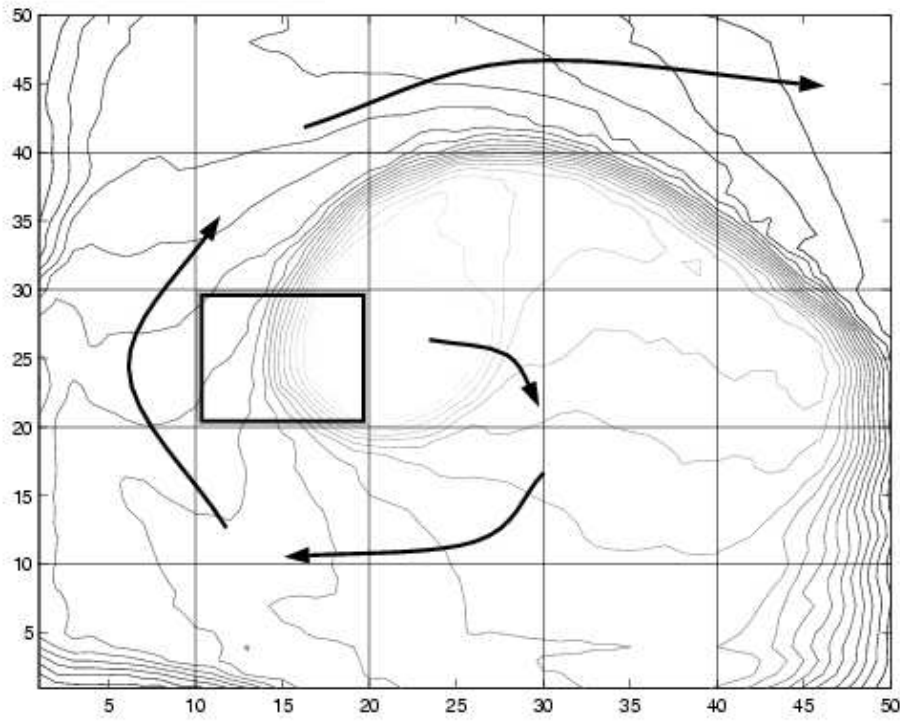
Value Function: TNA-3



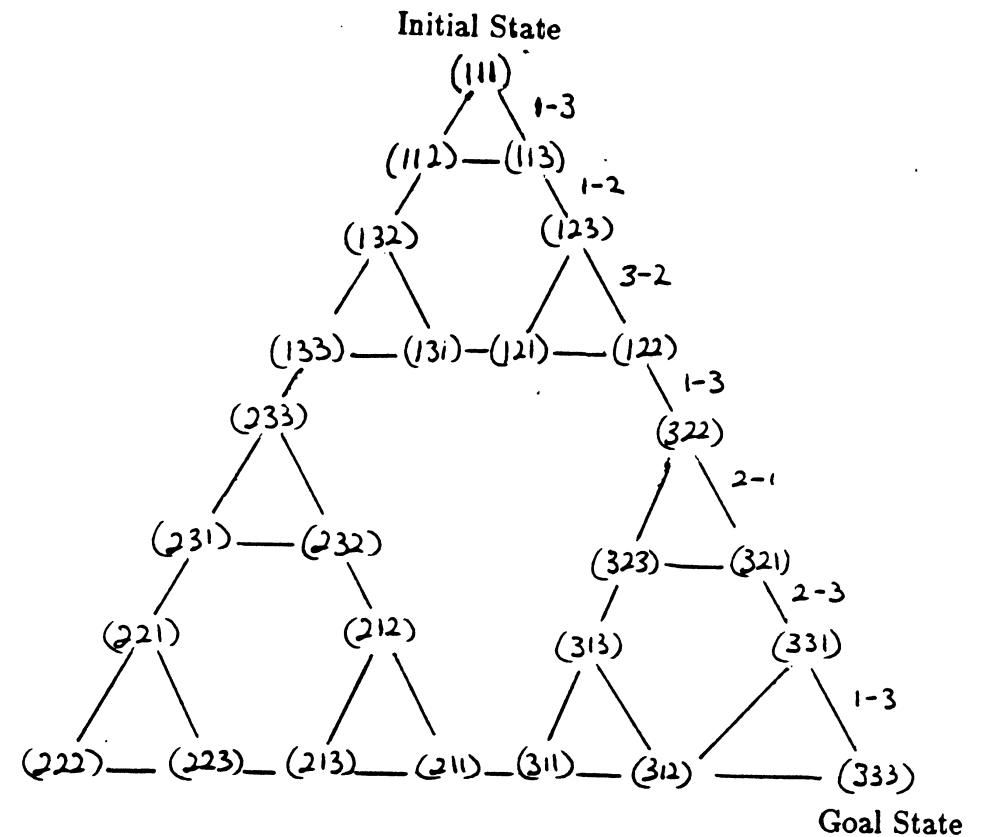
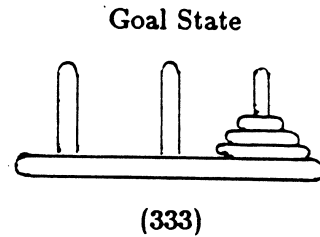
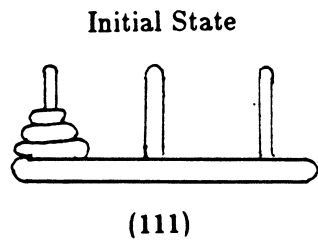


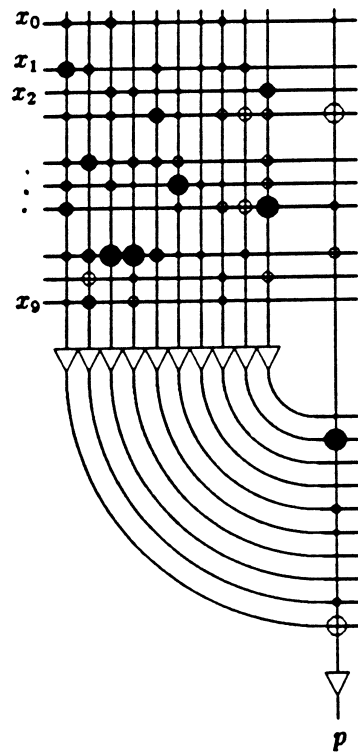
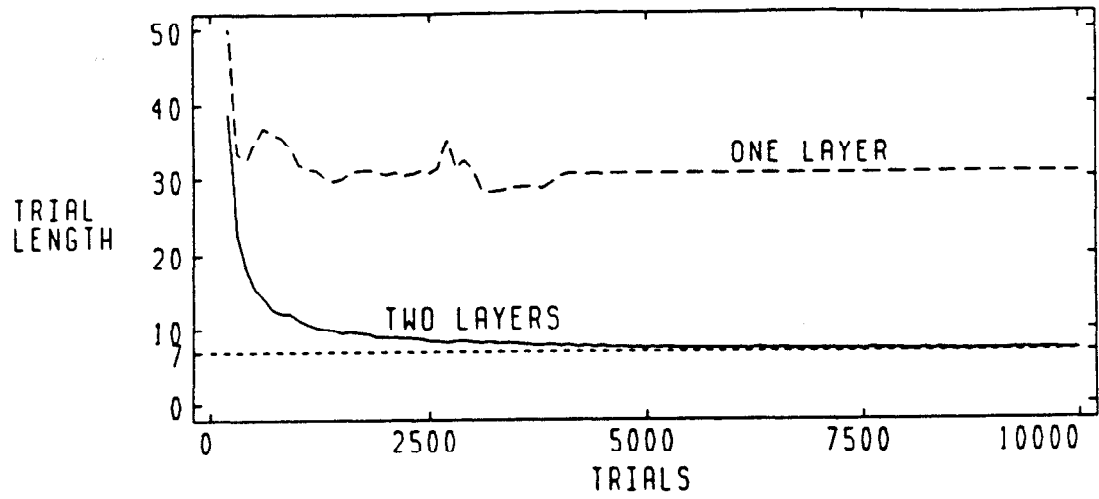


Temporal Neighborhoods

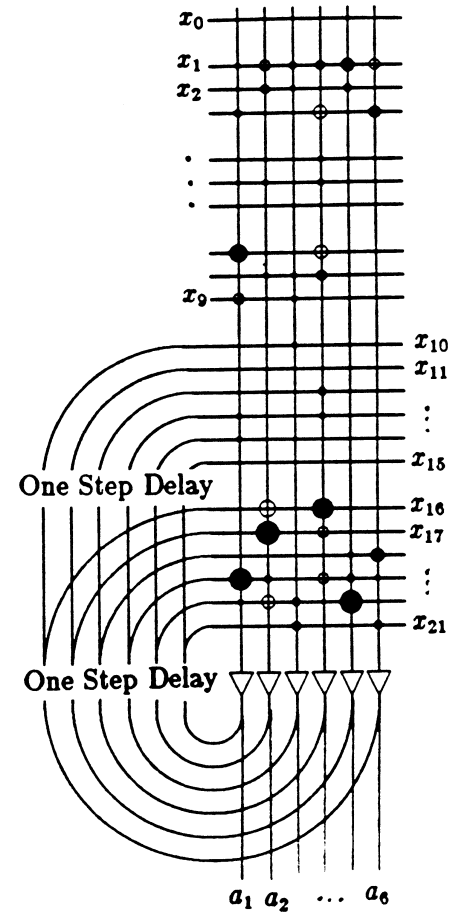


Recurrent Connections Form Macro Actions

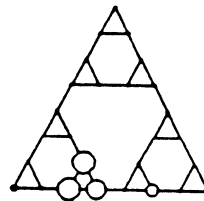




a.

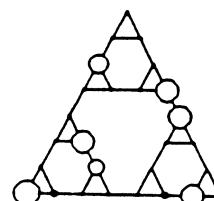


Functions learned
by hidden units.



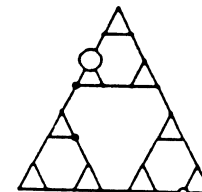
Feature 1

1.0



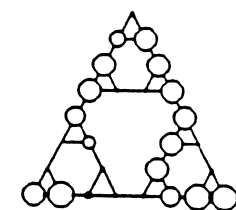
Feature 2

0.4



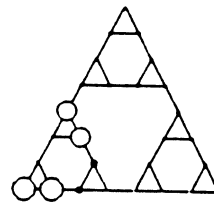
Feature 3

0.2



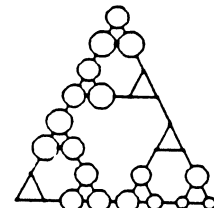
Feature 4

0.2



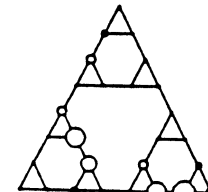
Feature 5

0.3



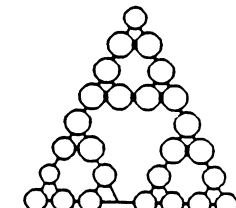
Feature 6

- 0.5



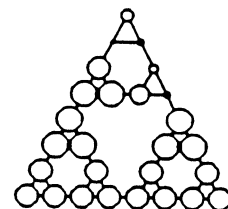
Feature 7

0.1



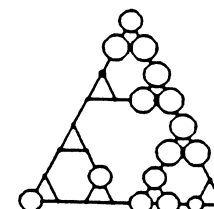
Feature 8

0.1



Feature 9

- 1.1



Feature 10

- 0.3