Q-Learning and Coarse Models

Sean Meyn

Department of Electrical and Computer Engineering
and the Coordinated Science Laboratory
University of Illinois

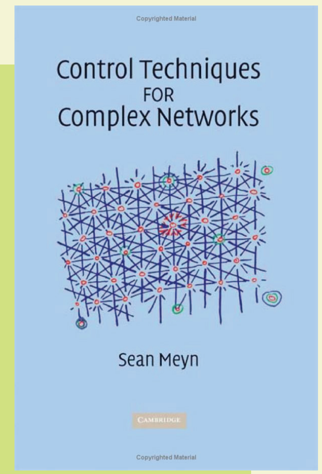
Joint work with Prashant Mehta
NSF support: ECS 0523620 and CMS 05-56352

ECE Illinois
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

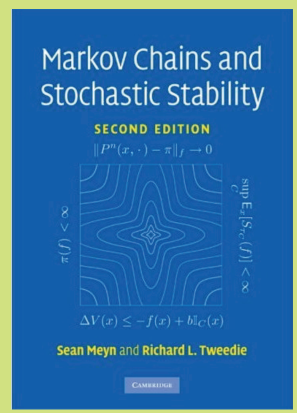
CSL

Coarse Models: *A rich collection of model reduction techniques*

Many of today's participants have contributed to this research.
A biased list:

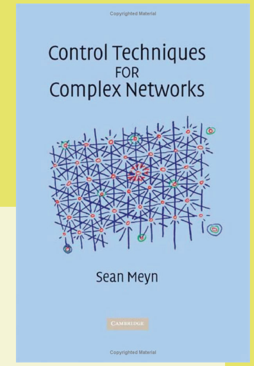


- *Fluid models*: Law of Large Numbers scaling,
  most likely paths in large deviations
- *Workload relaxation* for networks
  Heavy-traffic limits



- *Clustering*: spectral graph theory
  Markov spectral theory
- Singular perturbations
- *Large population limits*: Interacting particle systems

Workload Relaxations



An example from CTCN:

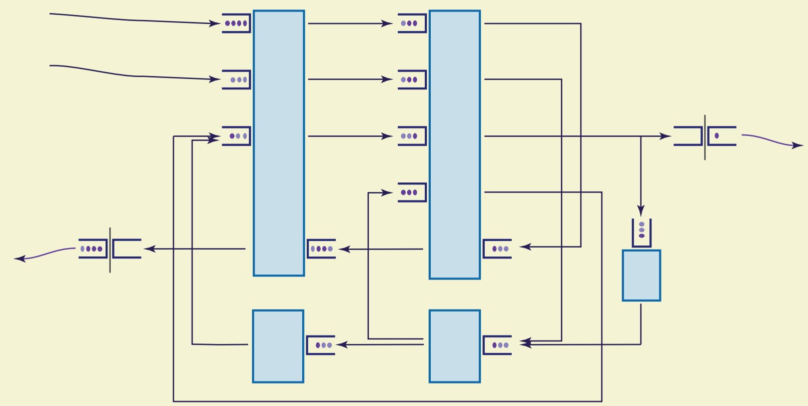


Figure 7.1: Demand-driven model with routing, scheduling, and re-work.

Workload at two stations evolves as a two-dimensional system
Cost is projected onto these coordinates:

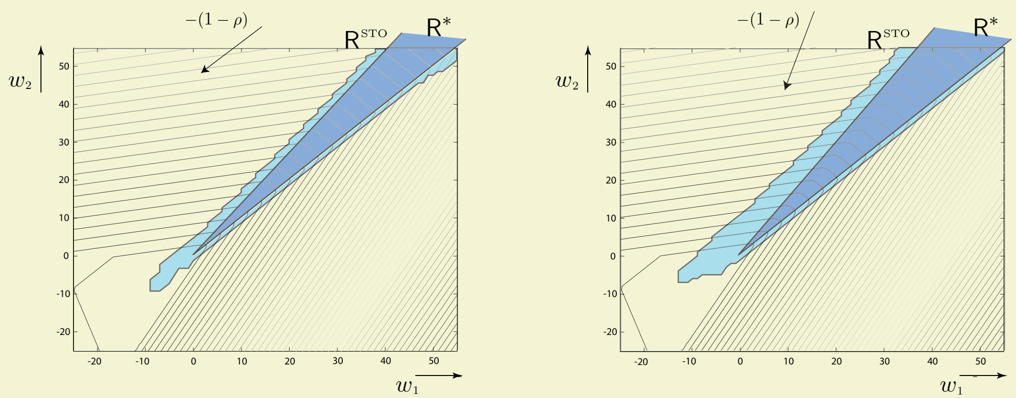
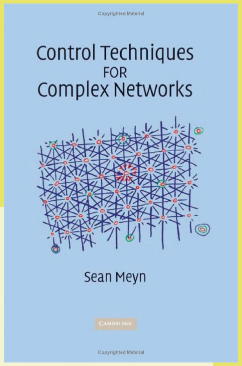


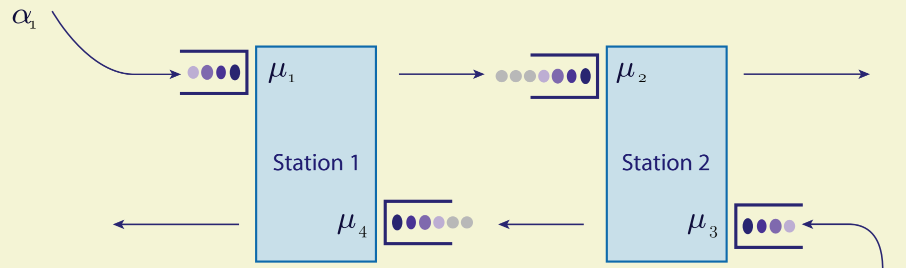
Figure 7.2: Optimal policies for two instances of the network shown in Figure 7.1. In each figure the optimal stochastic control region $\mathsf{R}^{\rm STO}$ is compared with the optimal region R^* obtained for the two dimensional fluid model.

Optimal policy for relaxation = hedging policy for full network

Workload Relaxations and Simulation



An example from CTCN:

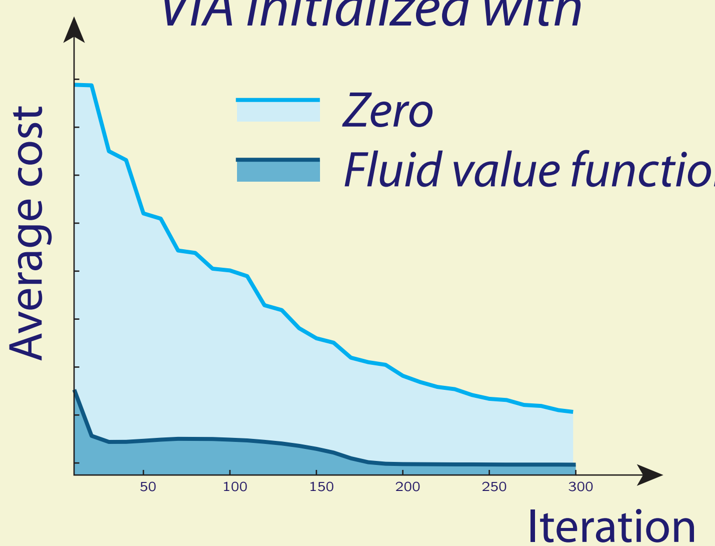


Decision making at stations 1 & 2
e.g., setting safety-stock levels

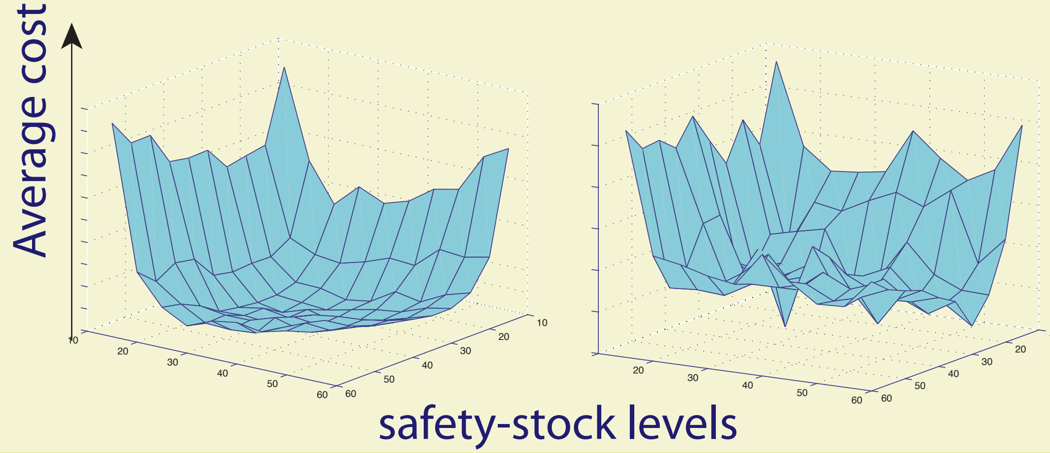
α_3

DP and simulations accelerated using *fluid value function* for *workload relaxation*

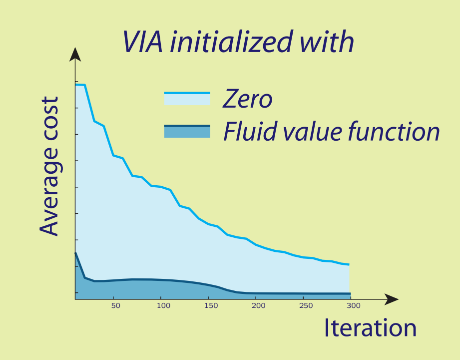
VIA initialized with



Simulated mean with and without control variate:



What To Do With a Coarse Model?

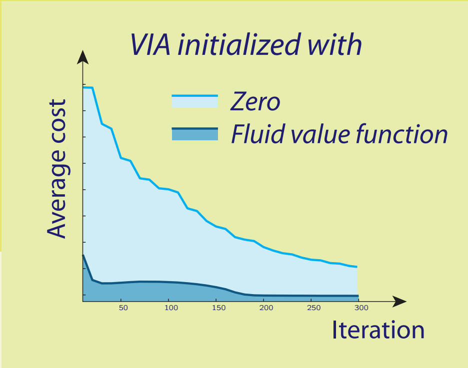


Setting: we have qualitative or partial quantitative insight regarding optimal control

The network examples relied on specific network structure

What about other models?

What To Do With a Coarse Model?



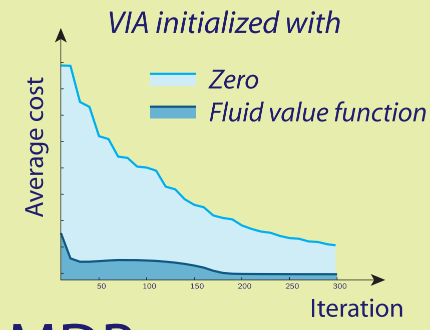
Setting: we have qualitative or partial quantitative insight regarding optimal control

The network examples relied on specific network structure

What about other models?

An answer lies in a new formulation of Q-learning

What is Q learning?



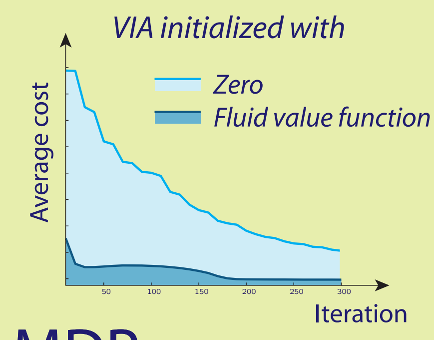
Watkin's 1992 formulation applied to finite state space MDPs

Idea is similar to Mayne & Jacobson's *differential dynamic programming*

Q-Learning
C. J. C. H. Watkins and P. Dayan
Machine Learning, 1992

Differential dynamic programming
D. H. Jacobson and D. Q. Mayne
American Elsevier Pub. Co. 1970

What is Q learning?



Watkin's 1992 formulation applied to finite state space MDPs

Idea is similar to Mayne & Jacobson's *differential dynamic programming*

Q-Learning
C. J. C. H. Watkins and P. Dayan
Machine Learning, 1992

Differential dynamic programming
D. H. Jacobson and D. Q. Mayne
American Elsevier Pub. Co. 1970

Deterministic formulation: Nonlinear system on Euclidean space,

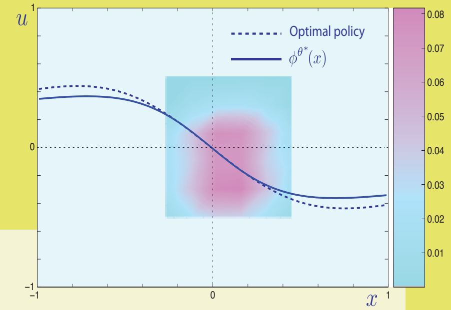
$$\frac{d}{dt}x(t) = f(x(t), u(t)), \qquad t \geq 0$$

Infinite-horizon discounted cost criterion,

$$J^*(x) = \inf \int_0^\infty e^{-\gamma s} c(x(s), u(s))\, ds, \qquad x(0) = x$$

with c a non-negative cost function.

What is Q learning?



Deterministic formulation: Nonlinear system on Euclidean space,

$$\frac{d}{dt}x(t) = f(x(t), u(t)), \qquad t \geq 0$$

Infinite-horizon discounted cost criterion,

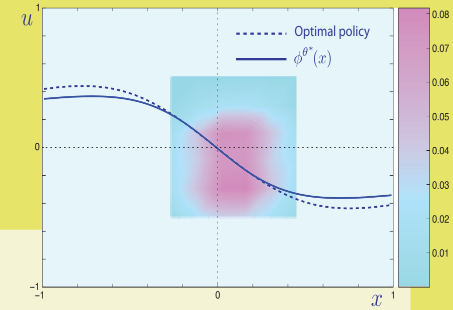
$$J^*(x) = \inf \int_0^\infty e^{-\gamma s} c(x(s), u(s))\, ds, \qquad x(0) = x$$

with c a non-negative cost function.

Differential generator: For any smooth function h,

$$\mathcal{D}_u h\,(x) := (\nabla h\,(x))^{\intercal} f\,(x, u)$$

What is Q learning?



Deterministic formulation: Nonlinear system on Euclidean space,

$$\frac{d}{dt}x(t) = f(x(t), u(t)), \qquad t \geq 0$$

Infinite-horizon discounted cost criterion,

$$J^*(x) = \inf \int_0^\infty e^{-\gamma s} c(x(s), u(s))\, ds, \qquad x(0) = x$$

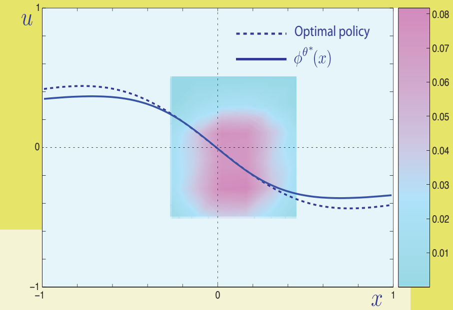
with c a non-negative cost function.

Differential generator: For any smooth function h,

$$\mathcal{D}_u h\,(x) := (\nabla h\,(x))^{\intercal} f(x, u)$$

HJB equation: $\quad \min_u \big(c(x, u) + \mathcal{D}_u J^*\,(x) \big) = \gamma J^*(x)$

What is Q learning?



Deterministic formulation: Nonlinear system on Euclidean space,

$$\frac{d}{dt}x(t) = f(x(t), u(t)), \qquad t \geq 0$$

Infinite-horizon discounted cost criterion,

$$J^*(x) = \inf \int_0^\infty e^{-\gamma s} c(x(s), u(s))\, ds, \qquad x(0) = x$$

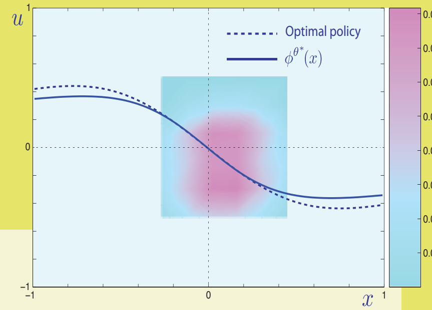
with c a non-negative cost function.

Differential generator: For any smooth function h,

$$\mathcal{D}_u h\,(x) := (\nabla h\,(x))^{T} f(x, u)$$

HJB equation: $$\min_u \big(c(x, u) + \mathcal{D}_u J^*\,(x) \big) = \gamma J^*(x)$$

The *Q-function* of Q-learning is this function of two variables

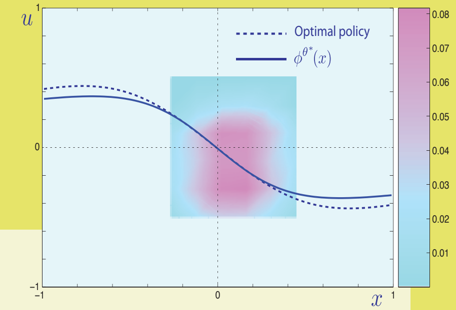


Q learning - Steps towards an algorithm

Sequence of five steps:

Step 1: Recognize fixed point equation for the Q-function
Step 2: Find a stabilizing policy that is ergodic
Step 3: Optimality criterion - minimize Bellman error
Step 4: Adjoint operation
Step 5: Interpret and simulate!

Q learning - Steps towards an algorithm



Step 1: Recognize fixed point equation for the Q-function

Q-function: $$H^*(x,u) \;=\; c(x,u) + \mathcal{D}_u J^*(x)$$

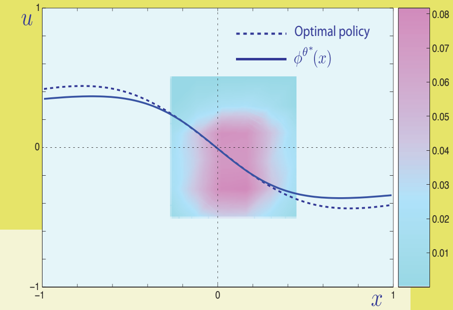
Its minimum: $$\underline{H}^*(x) := \min_{u \in \mathsf{U}} H^*(x,u) = \gamma J^*(x)$$

Fixed point equation:

$$\mathcal{D}_u \underline{H}^*(x) = -\gamma(c(x,u) - H^*(x,u))$$

Step 1: Recognize fixed point equation for the Q-function
Step 2: Find a stabilizing policy that is ergodic
Step 3: Optimality criterion - minimize Bellman error
Step 4: Adjoint operation
Step 5: Interpret and simulate!

Q learning - Steps towards an algorithm



Step 1: Recognize fixed point equation for the Q-function

Q-function: $$H^*(x,u) = c(x,u) + \mathcal{D}_u J^*(x)$$

Its minimum: $$\underline{H}^*(x) := \min_{u \in \mathsf{U}} H^*(x,u) = \gamma J^*(x)$$

Fixed point equation:

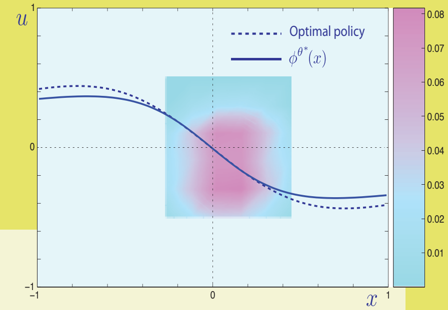
$$\mathcal{D}_u \underline{H}^*(x) = -\gamma\big(c(x,u) - H^*(x,u)\big)$$

Key observation for learning: For any input-output pair,

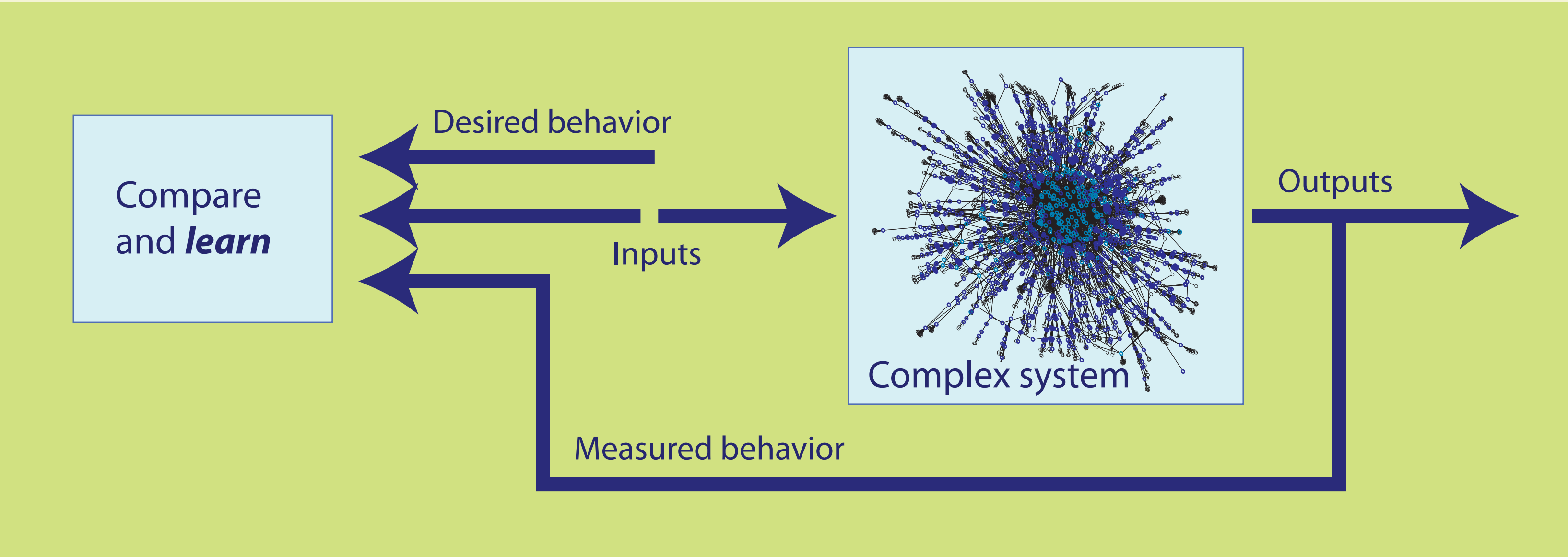
$$\mathcal{D}_u \underline{H}^*(x) = \frac{d}{dt}\underline{H}^*(x(t))\Big|_{\substack{x=x(t)\\ u=u(t)}}$$

Step 1: Recognize fixed point equation for the Q-function
Step 2: Find a stabilizing policy that is ergodic
Step 3: Optimality criterion - minimize Bellman error
Step 4: Adjoint operation
Step 5: Interpret and simulate!

Q learning - Steps towards an algorithm

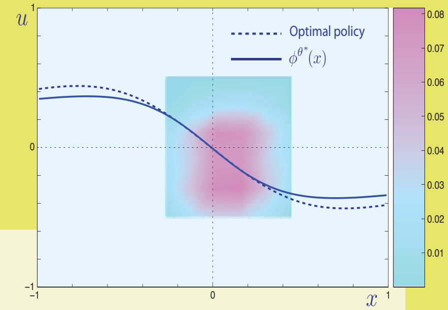


After Step 5: Not quite adaptive control:



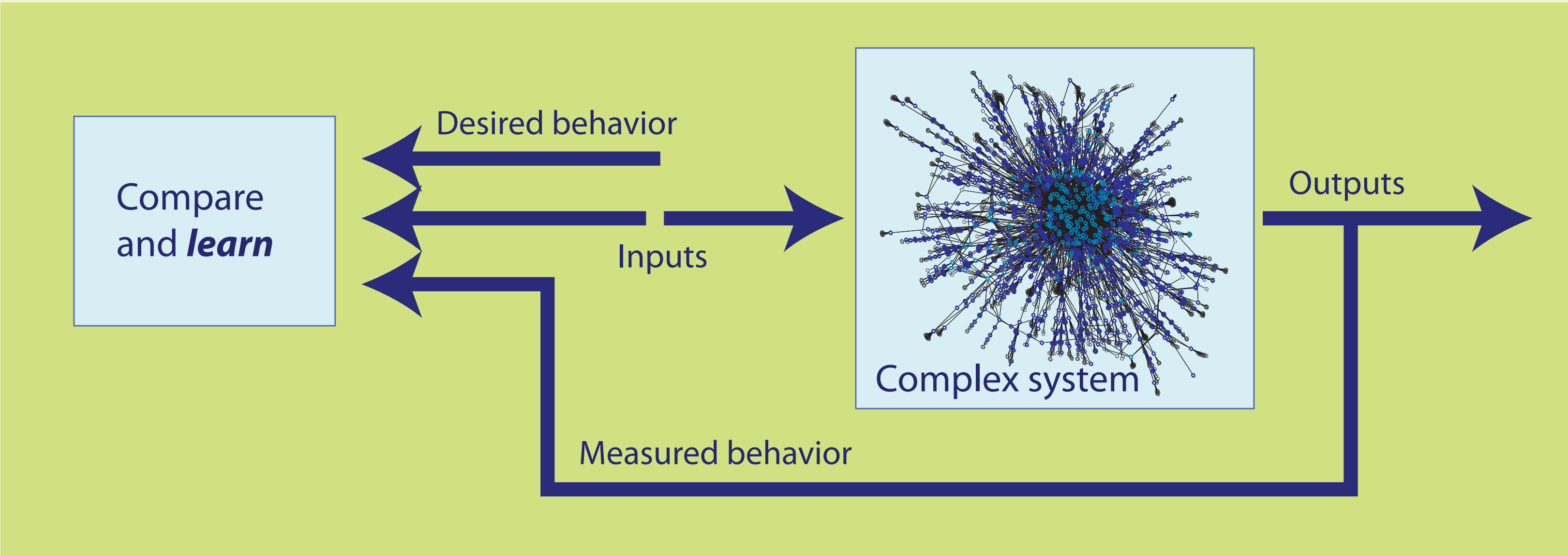
Ergodic input applied

Step 1: Recognize fixed point equation for the Q-function
Step 2: Find a stabilizing policy that is ergodic
Step 3: Optimality criterion - minimize Bellman error
Step 4: Adjoint operation
Step 5: Interpret and simulate!



Q learning - Steps towards an algorithm

After Step 5: Not quite adaptive control:



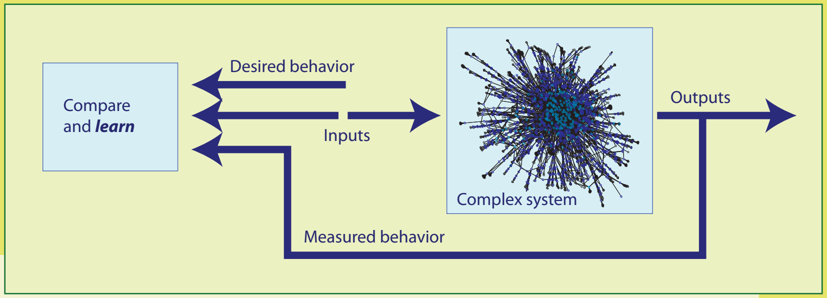
Ergodic input applied

Based on observations minimize the mean-square Bellman error:

$$\mathcal{E}_{\rm Bell}(\theta) := \int \big[\mathcal{L}^\theta\big]^2 \varpi(dx, du)$$

$$\mathcal{L}^\theta(x,u) := \mathcal{D}_u \underline{H}^\theta(x) + \gamma(c - H^\theta), \qquad \theta \in \mathbb{R}^d$$

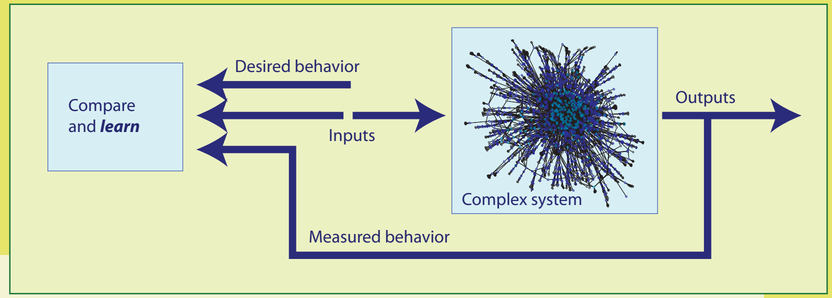
Q learning - Local Learning



Cubic nonlinearity:

$$\frac{d}{dt}x = -x^3 + u, \qquad c(x,u) = \frac{1}{2}x^2 + \frac{1}{2}u^2$$

Q learning - Local Learning

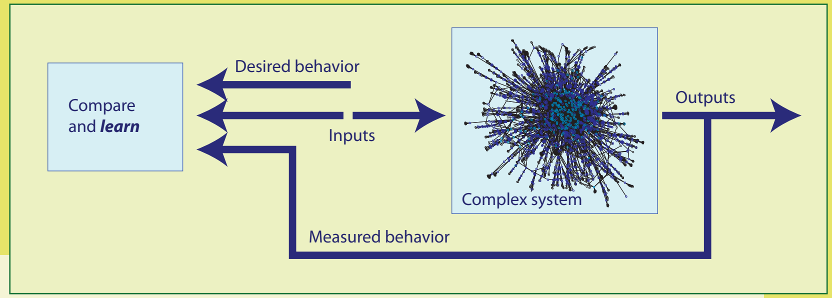


Cubic nonlinearity: $\frac{d}{dt}x = -x^3 + u, \qquad c(x,u) = \frac{1}{2}x^2 + \frac{1}{2}u^2$

HJB:

$$\min_u \Big(\tfrac{1}{2}x^2 + \tfrac{1}{2}u^2 + (-x^3 + u)\nabla J^*(x) \Big) = \gamma J^*(x)$$

Q learning - Local Learning



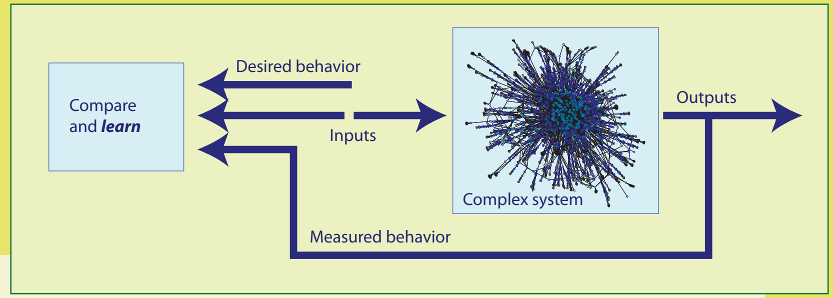
Cubic nonlinearity: $\frac{d}{dt}x = -x^3 + u, \qquad c(x,u) = \frac{1}{2}x^2 + \frac{1}{2}u^2$

HJB: $\min_u \Big(\frac{1}{2}x^2 + \frac{1}{2}u^2 + (-x^3 + u)\nabla J^*(x) \Big) = \gamma J^*(x)$

Basis:

$$H^\theta(x,u) = c(x,u) + \theta^x x^2 + \theta^{xu} \frac{x}{1+2x^2} u$$

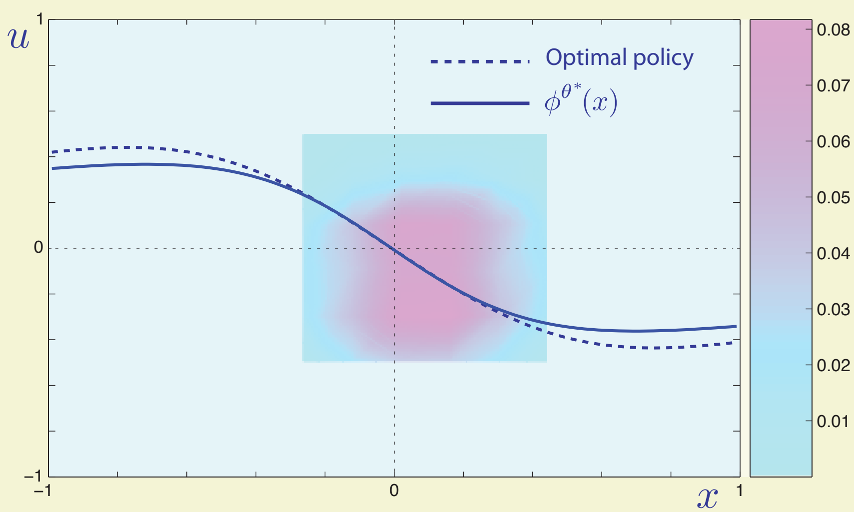
Q learning - Local Learning



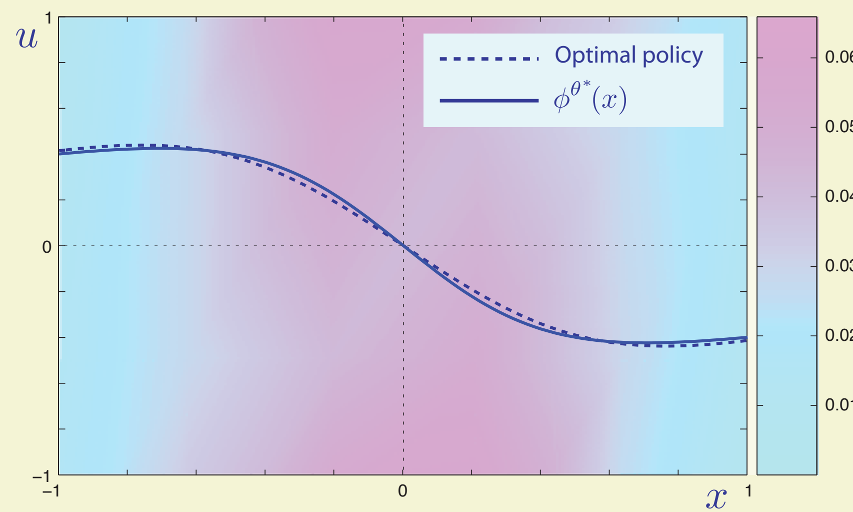
Cubic nonlinearity: $\frac{d}{dt}x = -x^3 + u, \qquad c(x,u) = \frac{1}{2}x^2 + \frac{1}{2}u^2$

HJB: $\min_u \Big(\frac{1}{2}x^2 + \frac{1}{2}u^2 + (-x^3 + u)\nabla J^*(x) \Big) = \gamma J^*(x)$

Basis: $H^\theta(x,u) = c(x,u) + \theta^x x^2 + \theta^{xu} \frac{x}{1+2x^2} u$



Low amplitude input



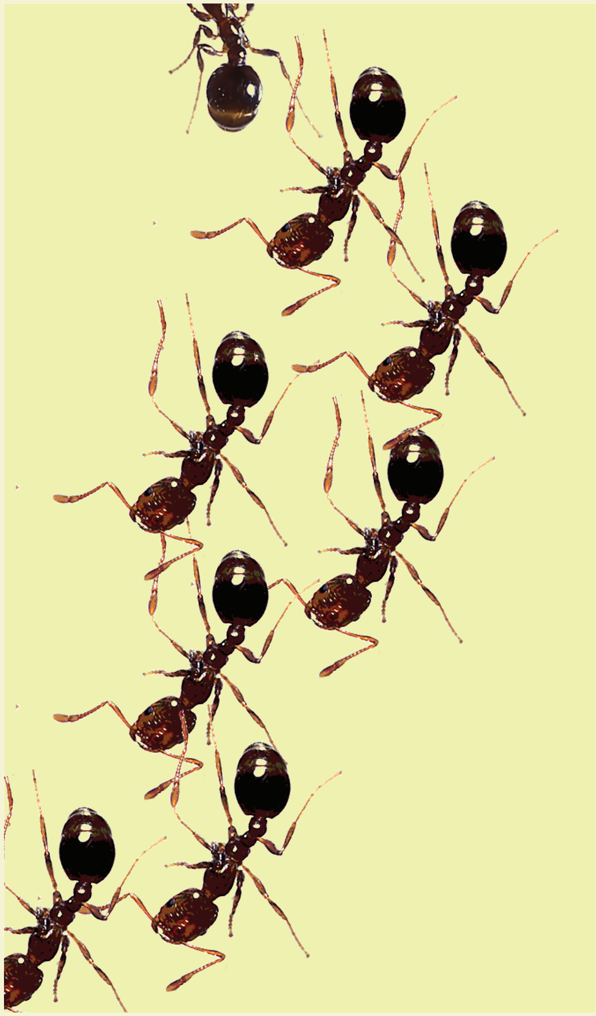
High amplitude input

$$u(t) = A(\sin(t) + \sin(\pi t) + \sin(et))$$

Multi-agent model

M. Huang, P. E. Caines, and R. P. Malhame. Large-population cost-coupled LQG problems with nonuniform agents: Individual-mass behavior and decentralized ε-Nash equilibria. *IEEE Trans. Auto. Control*, 52(9):1560–1571, 2007.

Huang et. al. Local optimization for global coordination



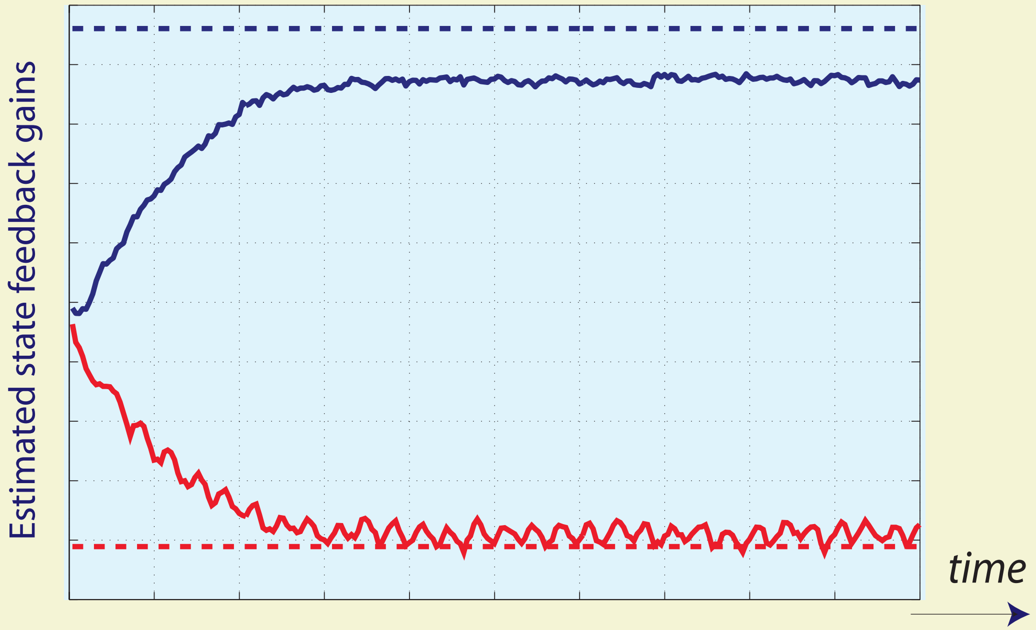
Multi-agent model

Model: Linear autonomous models - global cost objective

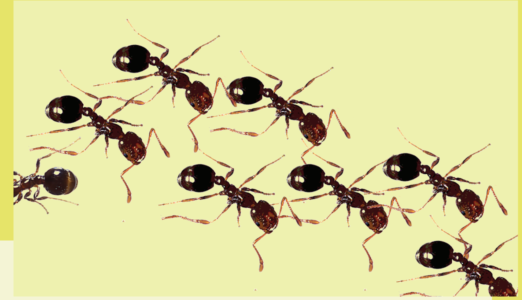
HJB: Individual state + global average

Basis: Consistent with low dimensional LQG model

Results from five agent model:



Multi-agent model



Model: Linear autonomous models - global cost objective

HJB: Individual state + global average

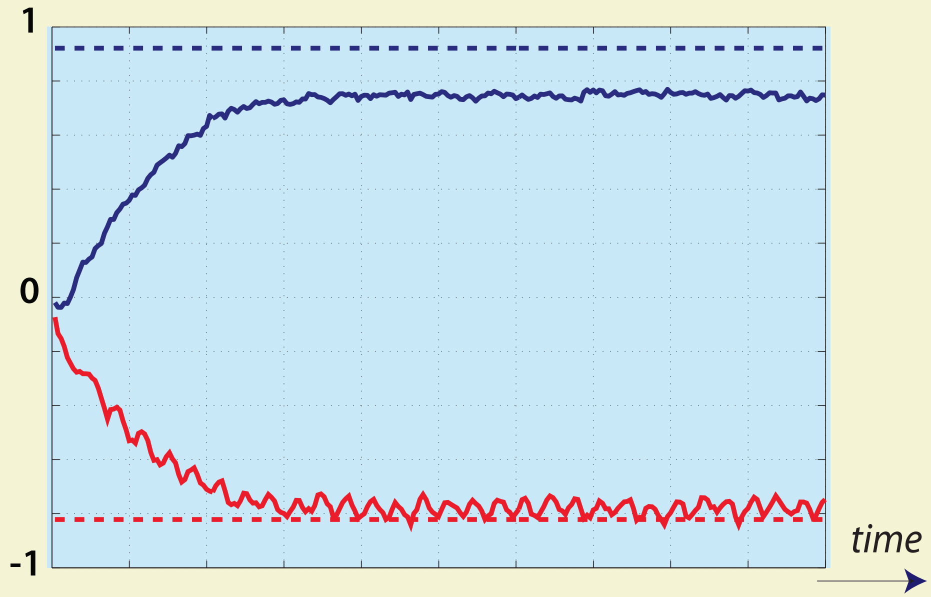
Basis: Consistent with low dimensional LQG model

Results from five agent model:

Estimated state feedback gains

—— k_x^i *(individual state)*

—— k_z^i *(ensemble state)*



Gains for agent 4: Q-learning sample paths and gains predicted from ∞-agent limit

Conclusions

Coarse models give tremendous insight

They are also tremendously useful
for design in approximate dynamic programming algorithms

Conclusions

Coarse models give tremendous insight

They are also tremendously useful
for design in approximate dynamic programming algorithms

Q-learning is as fundamental as the Riccati equation - this
should be included in our first-year graduate control courses

Conclusions

Coarse models give tremendous insight

They are also tremendously useful
for design in approximate dynamic programming algorithms

Q-learning is as fundamental as the Riccati equation - this
should be included in our first-year graduate control courses

Current research: Algorithm analysis and improvements
Applications in biology and economics
Analysis of game-theoretic issues
in coupled systems