Moncrief-O’Donnell Chair, UTA Research Institute (UTARI)
The University of Texas at Arlington, USA
and
Qian Ren Consulting Professor, State Key Laboratory of Synthetical Automation for Process Industries
Northeastern University, Shenyang, China

New Developments in Integral Reinforcement Learning:
Continuous-time Optimal Control and Games

Supported by:
ONR
US NSF

Talk available online at
http://www.UTA.edu/UTARI/acs

Supported by:
China NNSF
China Project 111
Invited by
Zhongping Jiang
Wen Changyun
Yang Guanghong
New Research Results
Integral Reinforcement Learning for Online Optimal Control
IRL for Online Solution of Multi-player Games
Multi-Player Games on Communication Graphs
Off-Policy Learning
Experience Replay
Bio-inspired Multi-Actor Critics
Output Synchronization of Heterogeneous MAS

Applications to:
Microgrid
Robotics
Industry Process Control
Optimality and Games

Optimal Control is Effective for:
  Aircraft Autopilots
  Vehicle engine control
  Aerospace Vehicles
  Ship Control
  Industrial Process Control

Multi-player Games Occur in:
  Networked Systems Bandwidth Assignment
  Economics
  Control Theory disturbance rejection
  Team games
  International politics
  Sports strategy

But, optimal control and game solutions are found by
  Offline solution of Matrix Design equations
  A full dynamical model of the system is needed
Optimal Control - The Linear Quadratic Regulator (LQR)

User prescribed optimization criterion

\[ V(x(t)) = \int_{t}^{\infty} (x^T Q x + u^T R u) \, d\tau \]

\[ 0 = PA + A^T P + Q - PBR^{-1} B^T P \]

\[ K = R^{-1} B^T P \]

An Offline Design Procedure
that requires Knowledge of system dynamics model (A,B)

System modeling is expensive, time consuming, and inaccurate
Adaptive Control is online and works for unknown systems. Generally not Optimal

Optimal Control is off-line, and needs to know the system dynamics to solve design eqs.

We want to find optimal control solutions
   Online in real-time
   Using adaptive control techniques
   Without knowing the full dynamics

For nonlinear systems and general performance indices

Bring together Optimal Control and Adaptive Control

Reinforcement Learning turns out to be the key to this!
Optimality in Biological Systems

Every living organism improves its control actions based on rewards received from the environment.

The resources available to living organisms are usually meager. Nature uses optimal control.

We want OPTIMAL performance
- ADP- Approximate Dynamic Programming

Actor-Critic Learning

Desired performance

Reinforcement learning
Ivan Pavlov 1890s

Optimality in Biological Systems

Every living organism improves its control actions based on rewards received from the environment.

The resources available to living organisms are usually meager. Nature uses optimal control.

We want OPTIMAL performance
- ADP- Approximate Dynamic Programming

Actor-Critic Learning

Desired performance

Reinforcement learning
Ivan Pavlov 1890s
Books


New Chapters on:
- Reinforcement Learning
- Differential Games

Reinforcement Learning and Adaptive Dynamic Programming for Feedback Control


Multi-player Game Solutions
IEEE Control Systems Magazine, Feb. 2017

RL for Markov Decision Processes \((X, U, P, R)\)

- **X** = states,
- **U** = controls
- **P** = Probability of going to state \(x'\) from state \(x\) given that the control is \(u\)
- **R** = Expected reward on going to state \(x'\) from state \(x\) given that the control is \(u\)

**Expected Value** of a policy \(\pi(x, u)\)

\[
V^\pi_k(x) = E_\pi \{J_{k,T} \mid x_k = x\} = E_\pi \{\sum_{i=k}^{k+T} \gamma^{i-k} r_i \mid x_k = x\}
\]

**Optimal control problem**

- Determine a policy \(\pi(x, u)\) to minimize the expected future cost

**optimal policy**

\[
\pi^* (x, u) = \arg \min_{\pi} V^\pi_k (s) = \arg \min_{\pi} E_\pi \{\sum_{i=k}^{k+T} \gamma^{i-k} r_i \mid x_k = x\}.
\]

**optimal value**

\[
V^*_k(x) = \min_{\pi} V^\pi_k (x) = \min_{\pi} E_\pi \{\sum_{i=k}^{k+T} \gamma^{i-k} r_i \mid x_k = x\}.
\]

**Policy Iteration**

- Policy evaluation by Bellman eq.

\[
V_j(x) = \sum_{x'} \pi_j (x, u) \sum_{x'} P_{x'x}^u [R_{x'x}^u + \gamma V_j(x')] \quad \text{for all } x \in X.
\]

- Policy Improvement

\[
\pi_{j+1}(x, u) = \arg \min_u \sum_{x'} P_{x'x}^u [R_{x'x}^u + \gamma V_j(x')] \quad \text{for all } x \in X.
\]

Policy Evaluation equation is a system of \(N\) simultaneous linear equations, one for each state.

Policy Improvement makes

\[
V^\pi'(x) \leq V^\pi (x)
\]

---

Discrete-Time Systems Optimal Adaptive Control

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

cost  \[ V_h(x_k) = \sum_{i=k}^{\infty} \gamma^{i-k} r(x_i, u_i) \quad \text{Example} \quad r(x_k, u_k) = x_k^T Q x_k + u_k^T R u_k \]

Difference eq equivalent  \[ V_h(x_k) = r(x_k, u_k) + \gamma \sum_{i=k+1}^{\infty} \gamma^{i-(k+1)} r(x_i, u_i) \]

Bellman equation  \[ V_h(x_k) = x_k^T Q x_k + u_k^T R u_k + \gamma V_h(x_{k+1}) \]

Hamiltonian  \[ H(x_k, \nabla V(x_k), u_k) = r(x_k, u_k) + \gamma V_h(x_{k+1}) - V_h(x_k) \]

System dynamics does not appear

Continuous-time Systems Nonlinear Optimal Regulator

Nonlinear System dynamics  \[ \dot{x} = f(x,u) = f(x) + g(x)u \]

Cost/value  \[ V(x(t)) = \int_{t}^{\infty} r(x,u) \, dt = \int_{t}^{\infty} (Q(x) + u^T R u) \, dt \]

Bellman Equation, in terms of the Hamiltonian function  \[ H(x, \frac{\partial V}{\partial x}, u) = \dot{V} + r(x,u) = \left( \frac{\partial V}{\partial x} \right)^T \dot{x} + r(x,u) = \left( \frac{\partial V}{\partial x} \right)^T (f(x) + g(x)u) + r(x,u) = 0 \]

STABILITY ?!

Leibniz gives
Differential equivalent

Problem- System dynamics shows up in Hamiltonian
RL ADP has been developed for Discrete-Time Systems

Discrete-Time System Hamiltonian Function

\[ x_{k+1} = f(x_k, u_k) \]

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

- Directly leads to temporal difference techniques
- System dynamics does not occur
- Two occurrences of value allow APPROXIMATE DYNAMIC PROGRAMMING methods

Continuous-Time System Hamiltonian Function

\[ \dot{x} = f(x, u) \]

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

Leads to off-line solutions if system dynamics is known
Hard to do on-line learning

- How to define temporal difference?
- System dynamics DOES occur
- Only ONE occurrence of value gradient

How can one do Policy Iteration for Unknown Continuous-Time Systems?
What is Value Iteration for Continuous-Time systems?
How can one do ADP for CT Systems?
Discrete-Time Systems
Adaptive (Approximate) Dynamic Programming

Four ADP Methods proposed by Paul Werbos

Critic NN to approximate:

Heuristic dynamic programming
Value Iteration
Value \( V(x_k) \)

AD Heuristic dynamic programming
(Watkins Q Learning)
Q function \( Q(x_k, u_k) \)

Dual heuristic programming
Gradient
\( \frac{\partial V}{\partial x} \)

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)
CT Systems- Derivation of Nonlinear Optimal Regulator

To find online methods for optimal control

Nonlinear System dynamics
\[ \dot{x} = f(x,u) = f(x) + g(x)u \]

Cost/value
\[ V(x(t)) = \int_{t}^{\infty} r(x,u) \, dt = \int_{t}^{\infty} (Q(x) + u^T Ru) \, dt \]

Bellman Equation, in terms of the Hamiltonian function
\[ H(x, \frac{\partial V}{\partial x}, u) = \dot{V} + r(x,u) = \left( \frac{\partial V}{\partial x} \right)^T \dot{x} + r(x,u) = \left( \frac{\partial V}{\partial x} \right)^T (f(x) + g(x)u) + r(x,u) = 0 \]

Stationarity condition
\[ \frac{\partial H}{\partial u} = 0 \]

Stationary Control Policy
\[ u = h(x) = -\frac{1}{2} R^{-1} g^T (x) \frac{\partial V}{\partial x} \]

HJB equation
\[ 0 = \left( \frac{dV^*}{dx} \right)^T f + Q(x) - \frac{1}{4} \left( \frac{dV^*}{dx} \right)^T g R^{-1} g^T \frac{dV^*}{dx} \quad , \quad V(0) = 0 \]

Off-line solution
HJB hard to solve. May not have smooth solution.
Dynamics must be known
CT Policy Iteration – a Reinforcement Learning Technique

Given any admissible policy \( u(x) = h(x) \)

The cost is given by solving the CT Bellman equation

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

Utility \( r(x,u) = Q(x) + u^T Ru \)

Policy Iteration Solution

Pick stabilizing initial control policy \( h_0(x) \)

**Policy Evaluation** - Find cost, Bellman eq.

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

\( V_j(0) = 0 \)

**Policy improvement** - Update control

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

Converges to solution of HJB

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

- Convergence proved by Leake and Liu 1967,
- Saridis 1979 if Lyapunov eq. solved exactly
- Beard & Saridis used Galerkin Integrals to solve Lyapunov eq.
- Abu Khalaf & Lewis used NN to approx. \( V \) for nonlinear systems and proved convergence

Full system dynamics must be known

Off-line solution

Policy Iterations for the Linear Quadratic Regulator

System \( \dot{x} = Ax + Bu \)

Cost \( V(x(t)) = \int_{t}^{\infty} (x^T Q x + u^T R u) \, d\tau \quad = x^T (t) P x(t) \)

Differential equivalent is the Bellman equation
\[ 0 = H(x, \frac{\partial V}{\partial x}, u) = \dot{V} + x^T Q x + u^T R u = 2 \left( \frac{\partial V}{\partial x} \right)^T \dot{x} + x^T Q x + u^T R u = 2x^T P (Ax + Bu) + x^T Q x + u^T R u \]

Given any stabilizing FB policy \( u = -K x \)

The cost value is found by solving Lyapunov equation = Bellman equation
\[ 0 = (A - BK)^T P + P (A - BK) + Q + K^T R K \]

Optimal Control is
\( u = -R^{-1} B^T P x = -K x \)

Algebraic Riccati equation
\[ 0 = PA + A^T P + Q - PBR^{-1} B^T P \]

Full system dynamics must be known
Off-line solution
LQR Policy iteration = Kleinman algorithm

1. For a given control policy \( u = -K_j x \) solve for the cost:

\[
0 = A_j^T P_j + P_j A_j + Q + K_j^T R K_j
\]

\[
A_j = A - B K_j
\]

Bellman eq. = Lyapunov eq.

2. Improve policy:

\[
K_{j+1} = R^{-1} B^T P_j
\]

- If started with a stabilizing control policy \( K_0 \) the matrix \( P_j \) monotonically converges to the unique positive definite solution of the Riccati equation.
- Every iteration step will return a stabilizing controller.
- The system has to be known.

OFF-LINE DESIGN
MUST SOLVE LYAPUNOV EQUATION AT EACH STEP.  

Kleinman 1968
Integral Reinforcement Learning

Work of Draguna Vrabie

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

Can Avoid knowledge of drift term \( f(x) \)

Policy iteration requires repeated solution of the CT Bellman equation

\[ 0 = \dot{V} + r(x,u(x)) = \left( \frac{\partial V}{\partial x} \right)^T \dot{x} + r(x,u(x)) = \left( \frac{\partial V}{\partial x} \right)^T f(x,u(x)) + Q(x) + u^T Ru \equiv H(x, \frac{\partial V}{\partial x}, u(x)) \]

This can be done online without knowing \( f(x) \)
using measurements of \( x(t), u(t) \) along the system trajectories

Integral Reinforcement Learning

Work of Draguna Vrabie 2009

value

\[ V(x(t)) = \int_{t}^{\infty} r(x,u) \, d\tau = \int_{t}^{t+T} r(x,u) \, d\tau + \int_{t+T}^{\infty} r(x,u) \, d\tau \]

Key Idea= US Patent

Lemma 1 – Draguna Vrabie

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

Bad Bellman Equation

Is equivalent to

Integral reinf. form (IRL) for the CT Bellman eq.

\[ V(x(t)) = \int_{t}^{t+T} r(x,u) \, d\tau + V(x(t+T)), \quad V(0) = 0 \]

Good Bellman Equation

Solves Bellman equation without knowing \( f(x,u) \)

Allows definition of temporal difference error for CT systems

\[ e(t) = -V(x(t)) + \int_{t}^{t+T} r(x,u) \, d\tau + V(x(t+T)) \]
Integral Reinforcement Learning (IRL)- Draguna Vrabie

**IRL Policy iteration**

**Policy evaluation** - IRL Bellman Equation

Cost update:  
\[ V_k(x(t)) = \int_t^{t+T} r(x, u_k) \, dt + V_k(x(t+T)) \]

CT Bellman eq.

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

Equivalent to:  
\[ 0 = \left( \frac{\partial V}{\partial x} \right)^T f(x, u) + r(x, u) \equiv H(x, \frac{\partial V}{\partial x}, u) \]

Solves Bellman eq. (nonlinear Lyapunov eq.) without knowing system dynamics

**Policy improvement**

Control gain update:  
\[ u_{k+1} = h_{k+1}(x) = -\frac{1}{2} R^{-1} g^T(x) \frac{\partial V_k}{\partial x} \]

\( g(x) \) needed for control update

Initial stabilizing control is needed

Converges to solution to HJB eq.  
\[ 0 = \left( \frac{dV^*}{dx} \right)^T f + Q(x) - \frac{1}{4} \left( \frac{dV^*}{dx} \right)^T gR^{-1} g^T \frac{dV^*}{dx} \]

CT Policy Iteration – How to implement online?
Linear Systems Quadratic Cost - LQR

Value function is quadratic \( V(x(t)) = x^T(t)Px(t) \)

Policy evaluation - solve IRL Bellman Equation

\[
x^T(t)P_k x(t) = \int_t^{t+T} x^T(\tau)(Q+K_k^T RK_k)x(\tau) \, d\tau + x^T(t+T)P_k x(t+T)
\]

\[
x^T(t)P_k x(t) - x^T(t+T)P_k x(t+T) = \int_t^{t+T} x^T(\tau)(Q+K_k^T RK_k)x(\tau) \, d\tau
\]

\[
= \begin{bmatrix} x^1(t) & x^2(t) \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} \begin{bmatrix} x^1(t) \\ x^2(t) \end{bmatrix} - \begin{bmatrix} x^1(t+T) & x^2(t+T) \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} \begin{bmatrix} x^1(t+T) \\ x^2(t+T) \end{bmatrix}
\]

\[
= \begin{bmatrix} p_{11} & p_{12} & p_{22} \end{bmatrix} \begin{bmatrix} (x^1)^2 \\ 2x^1x^2 \\ (x^2)^2 \end{bmatrix} - \begin{bmatrix} p_{11} & p_{12} & p_{22} \end{bmatrix} \begin{bmatrix} (x^1)^2 \\ 2x^1x^2 \\ (x^2)^2 \end{bmatrix}
\]

\[
= \bar{p}_k^T [\bar{x}(t) - \bar{x}(t+T)]
\]

\[
\bar{p}_k^T \phi(t) \equiv \bar{p}_k^T [\bar{x}(t) - \bar{x}(t+T)] = \int_t^{t+T} x(\tau)^T(Q+L_k^T RL_k)x(\tau) \, d\tau = \rho(t, t+T)
\]

Same form as standard System ID problems
Approximate Dynamic Programming Implementation

Value Function Approximation (VFA) to Solve Bellman Equation

- Paul Werbos (ADP), Dimitri Bertsekas (NDP)

\[ V_k(x(t)) = \int_t^{t+T} \left( Q(x) + u_k^T R u_k \right) dt + V_k(x(t+T)) \]

Approximate value by Weierstrass Approximator Network

\[ V = W^T \phi(x) \]

\[ W_k^T \phi(x(t)) = \int_t^{t+T} \left( Q(x) + u_k^T R u_k \right) dt + W_k^T \phi(x(t+T)) \]

\[ W_k^T \left[ \phi(x(t)) - \phi(x(t+T)) \right] = \int_t^{t+T} \left( Q(x) + u_k^T R u_k \right) dt \]

Scalar equation with vector unknowns

regression vector

Reinforcement on time interval \([t, t+T]\)

Same form as standard System ID problems in Adaptive Control

Now use RLS or batch least-squares along the trajectory to get new weights \(W_k\)

Then find updated FB

\[ u_{k+1} = h_{k+1}(x) = -\frac{1}{2} R^{-1} g^T(x) \frac{\partial V_k}{\partial x} = -\frac{1}{2} R^{-1} g^T(x) \left[ \frac{\partial \phi(x(t))}{\partial x(t)} \right]^T W_k \]

Direct Optimal Adaptive Control for Partially Unknown CT Systems
Solving the IRL Bellman Equation- LQR case

LQR case \( V(x(t)) = x^T(t)Px(t) \)

Solve for value function parameters
\[
\begin{bmatrix}
p_{11} & p_{12} \\
p_{12} & p_{22}
\end{bmatrix}
\]

\[ W^T = [p_{11} \quad p_{12} \quad p_{22}] \]

Need data from 3 time intervals to get 3 equations to solve for 3 unknowns

\[
W_k^T \left[ \phi(x(t)) - \phi(x(t + T)) \right] = \int_{t}^{t+T} \left( Q(x) + u_k^T R u_k \right) dt
\]

\[
W_k^T \left[ \phi(x(t + T)) - \phi(x(t + 2T)) \right] = \int_{t+T}^{t+2T} \left( Q(x) + u_k^T R u_k \right) dt
\]

\[
W_k^T \left[ \phi(x(t + 2T)) - \phi(x(t + 3T)) \right] = \int_{t+2T}^{t+3T} \left( Q(x) + u_k^T R u_k \right) dt
\]

Now solve by Batch least-squares
Integral Reinforcement Learning (IRL)

Solve Bellman Equation - Solves Lyapunov eq. without knowing dynamics

\[ W_k^T \left[ \phi(x(t)) - \phi(x(t+T)) \right] = \int_t^{t+T} x(\tau)^T \left( Q + K_k^T R K_k \right) x(\tau) d\tau = \rho(t, t+T) \]

Data set at time \([t, t+T)\)

\[(x(t), \rho(t, t+T), x(t+T))\]

Observe \(x(t)\)

Apply \(u^k = K_k x\)

Observe cost integral \(\rho(t, t+T)\)

Update \(P\)

Observe \(x(t+T)\)

Apply \(u^k = K_k x\)

Observe cost integral \(\rho(t+T, t+2T)\)

Update \(P\)

Observe \(x(t+2T)\)

Apply \(u^k = K_k x\)

Observe cost integral \(\rho(t+2T, t+3T)\)

Update \(P\)

Do RLS until convergence to \(P_k\)

Or use batch least-squares

Update control gain

\[ K_{k+1} = R^{-1} B^T P_k \]

This is a data-based approach that uses measurements of \(x(t), u(t)\) Instead of the plant dynamical model.
Gain update (Policy)

$$K_k$$

Interval $$T$$ can vary

Control

$$u_k(t) = -K_k x(t)$$

Reinforcement Intervals $$T$$ need not be the same
They can be selected on-line in real time

Continuous-time control with discrete gain updates
Persistence of Excitation

\[ W_k^T \left[ \phi(x(t)) - \phi(x(t + T)) \right] = \int_t^{t+T} \left( Q(x) + u_k^T R u_k \right) dt \]

Regression vector must be PE

Relates to choice of reinforcement interval T
Implementation

Policy evaluation
Need to solve online

\[ W_k^T \left[ \phi(x(t)) - \phi(x(t+T)) \right] = \int_{t}^{t+T} x(\tau)^T (Q + K_k^T R K_k^T) x(\tau) d\tau = \rho(t, t+T) \]

Add a new state = Integral Reinforcement

\[ \dot{\rho} = x^T Q x + u^T R u \]

This is the controller dynamics or memory
Direct Optimal Adaptive Controller

Solves Riccati Equation Online without knowing A matrix

CT time Actor-Critic Structure

Run RLS or use batch L.S. To identify value of current control

Update FB gain after Critic has converged

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

Reinforcement interval $T$ can be selected on line on the fly – can change
Optimal Adaptive IRL for CT systems

Actor / Critic structure for CT Systems

Reinforcement learning

\[ V_k(x(t)) = \int_{t}^{t+T} r(x,u_k) \, dt + V_k(x(t+T)) \]

Theta waves 4-8 Hz

\[ u_{k+1} = h_{k+1}(x) = -\frac{1}{2} R^{-1} g^T(x) \frac{\partial V_k}{\partial x} \]

A new structure of adaptive controllers

D. Vrabie, 2009
Data-driven Online Adaptive Optimal Control

DDO

User prescribed optimization criterion

\[ J = (Q, R) \]

\[ x^T(t)P_kx(t) = \int_{t}^{t+T} x^T(\tau)(Q+K_k^TRK_k)x(\tau)d\tau + x^T(t+T)P_kx(t+T) \]

\[ K_{k+1} = R^{-1}B^TP_k \]

An Online Supervisory Control Procedure that requires no Knowledge of system dynamics model A

Automatically tunes the control gains in real time to optimize a user given cost function

Uses measured data \((u(t), x(t))\) along system trajectories
Simulation 1- F-16 aircraft pitch rate controller

\[
\dot{x} = \begin{bmatrix}
-1.01887 & 0.90506 & -0.00215 \\
0.82225 & -1.07741 & -0.17555 \\
0 & 0 & -1
\end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u
\]

\[
Q = I, \quad R = I
\]

ARE \quad 0 = PA + A^T P + Q - PBR^{-1} B^T P

Select quadratic NN basis set for VFA

Exact solution \quad W_1^* = [p_{11}, 2p_{12}, 2p_{13}, p_{22}, 2p_{23}, p_{33}]^T

\[
= [1.4245, 1.1682, -0.1352, 1.4349, -0.1501, 0.4329]^T
\]

Stevens and Lewis 2003

\[
x = [\alpha \quad q \quad \delta_e]
\]
Simulations on: F-16 autopilot

A matrix not needed

Converge to SS Riccati equation soln

Solves ARE online without knowing $A$

\[ 0 = PA + A^T P + Q - PBR^{-1}B^T P \]
Simulation 2: Load Frequency Control of Electric Power system

\[ \dot{x} = Ax + Bu \]

\[ x(t) = [\Delta f(t) \Delta P_g(t) \Delta X_g(t) \Delta E(t)]^T \]

\[ A = \begin{bmatrix} -1/T_p & K_p/T_p & 0 & 0 \\ 0 & -1/T_r & 1/T_r & 0 \\ -1/R T_G & 0 & -1/T_G & -1/T_G \\ K_E & 0 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ 1/T_G \\ 0 \end{bmatrix} \]

ARE

\[ 0 = PA + A^T P + Q - PBR^{-1}B^T P \]

ARE solution using full dynamics model (A,B)

\[ P_{ARE} = \begin{bmatrix} 0.4750 & 0.4766 & 0.0601 & 0.4751 \\ 0.4766 & 0.7831 & 0.1237 & 0.3829 \\ 0.0601 & 0.1237 & 0.0513 & 0.0298 \\ 0.4751 & 0.3829 & 0.0298 & 2.3370 \end{bmatrix} \]
\[
0 = PA + A^T P + Q - PBR^{-1}B^T P
\]

Solves ARE online without knowing \( A \)

\[
P_{\text{ARE}} = \begin{bmatrix}
0.4750 & 0.4766 & 0.0601 & 0.4751 \\
0.4766 & 0.7831 & 0.1237 & 0.3829 \\
0.0601 & 0.1237 & 0.0513 & 0.0298 \\
0.4751 & 0.3829 & 0.0298 & 2.3370
\end{bmatrix}
\]

IRL period of \( T = 0.1 \)s.

Fifteen data points \( (x(t), x(t+T), \rho(t:t+T)) \)

Hence, the value estimate was updated every 1.5s.

\[
P_{\text{critic NN}} = \begin{bmatrix}
0.4802 & 0.4768 & 0.0603 & 0.4754 \\
0.4768 & 0.7887 & 0.1239 & 0.3834 \\
0.0603 & 0.1239 & 0.0567 & 0.0300 \\
0.4754 & 0.3843 & 0.0300 & 2.3433
\end{bmatrix}
\]
Optimal Control Design Allows a Lot of Design Freedom

The Power of Optimal Design

Once you can do optimal design that minimizes a performance index, many sorts of designs are immediately possible.

Minimum energy

\[ J = \frac{1}{2} \int_0^\infty x^T Q x + u^T R u \, dt \]

Minimum fuel

\[ J = \frac{1}{2} \int_0^\infty x^T Q x + \rho |u| \, dt \]

Minimum time

\[ J = \int_0^T 1 \, dt = T \]

Constrained control inputs

\[ J = \frac{1}{2} \int_0^\infty \left( Q(x) + \int_0^n \sigma^{-1}(\nu) d\nu \right) dt \]

Approximate minimum time with smooth control inputs

\[ J = \frac{1}{2} \int_0^\infty \left( \tanh(x^T Q x) + \rho \int_0^n \sigma^{-1}(\nu) d\nu \right) dt \]

\[ \tanh(p) \]

\[ \begin{array}{c}
-1 \\
1 \\
\end{array} \]

\[ p \]
Issues with Nonlinear ADP

LS local smooth solution for Critic NN update

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

$$V(x(t)) = \int_{t}^{t+T} r(x,u) \, d\tau \quad + \quad V(x(t+T)), \quad V(0) = 0$$

Integral over a region of state-space
Approximate using a set of points

Batch LS

Selection of NN Training Set

Set of points over a region vs. points along a trajectory

Take sample points along a single trajectory

Recursive Least-Squares RLS

For Linear systems- these are the same

For Nonlinear systems

Persistence of excitation is needed to solve for the weights
But EXPLORATION is needed to identify the complete value function
- PE Versus Exploration
IRL Value Iteration - Draguna Vrabie

**IRL Policy iteration**  Initial stabilizing control is needed

**Policy evaluation- IRL Bellman Equation**

Cost update

$$V_k(x(t)) = \int_t^{t+T} r(x,u_k) \, dt + V_k(x(t+T))$$

**Policy improvement**

Control gain update

$$u_{k+1} = h_{k+1}(x) = -\frac{1}{2} R^{-1} g^T(x) \frac{\partial V_k}{\partial x}$$

Converges to solution to HJB eq.

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

---

**IRL Value iteration**  Initial stabilizing control is NOT needed

**Value evaluation- IRL Bellman Equation**

Cost update

$$V_{k+1}(x(t)) = \int_t^{t+T} r(x,u_k) \, dt + V_k(x(t+T))$$

**Policy improvement**

Control gain update

$$u_{k+1} = h_{k+1}(x) = -\frac{1}{2} R^{-1} g^T(x) \frac{\partial V_{k+1}}{\partial x}$$

Converges if $T$ is small enough

CT PI Bellman eq. = Lyapunov eq.

CT VI Bellman eq.
Kung Tz  500 BC
Confucius

Tian xia da tong
Harmony under heaven

Archery
Chariot driving

Music
Rites and Rituals

Poetry
Mathematics

Man's relations to
Family
Friends
Society
Nation
Emperor
Ancestors
Optimal Adaptive IRL for CT systems

D. Vrabie, 2009

Actor / Critic structure for CT Systems

Reinforcement learning

\[ V_k(x(t)) = \int_{t}^{t+T} r(x,u_k) \, dt + V_k(x(t+T)) \]

Theta waves 4-8 Hz

\[ u_{k+1} = h_{k+1}(x) = -\frac{1}{2} R^{-1}(x) g^T(x) \frac{\partial V_k}{\partial x} \]

A new structure of adaptive controllers
Oscillation is a fundamental property of neural tissue

Brain has multiple adaptive clocks with different timescales

*gamma rhythms* 30-100 Hz, hippocampus and neocortex
high cognitive activity.
  • consolidation of memory
  • spatial mapping of the environment – place cells

The high frequency processing is due to the large amounts of sensorial data to be processed

*theta rhythm*, Hippocampus, Thalamus, 4-10 Hz
sensory processing, memory and voluntary control of movement.


**Figure 1.** Learning-oriented specialization of the cerebellum, the basal ganglia, and the cerebral cortex [1], [2]. The cerebellum is specialized for supervised learning based on the error signal encoded in the climbing fibers from the inferior olive. The basal ganglia are specialized for reinforcement learning based on the reward signal encoded in the dopaminergic fibers from the substantia nigra. The cerebral cortex is specialized for unsupervised learning based on the statistical properties of the input signal.

Doya, Kimura, Kawato 2001
Summary of Motor Control in the Human Nervous System

Cerebral cortex
Motor areas

Thalamus

Basal ganglia

Cerebellum

Brainstem

Exteroceptive receptors

Interoceptive receptors

Muscle contraction and movement

Long term
Memory functions

Unsupervised learning

Reinforcement Learning- dopamine

Hierarchy of multiple parallel loops

Muscle contraction and movement

theta rhythms 4-10 Hz

gamma rhythms 30-100 Hz

picture by E. Stingu D. Vrabie

Kenji Doya

Short term

Reinforcement Learning- dopamine

Supervised learning

reflex

Motor control 200 Hz
Synchronous Real-time Data-driven Optimal Control
Actor / Critic structure for CT Systems

\[ V_k(x(t)) = \int_{t}^{t+T} r(x, u_k) \, dt + V_k(x(t+T)) \]

Theta waves 4-8 Hz

\[ u_{k+1} = h_{k+1}(x) = -\frac{1}{2} R^{-1} g^T(x) \frac{\partial V_k}{\partial x} \]

Policy Iteration gives the structure needed for online optimal solution

A new structure of adaptive controllers
Synchronous Online Solution of Optimal Control for Nonlinear Systems

Kyriakos Vamvoudakis

Critic Network

Take VFA as

\[ V(x) = \hat{W}_1^T \phi_1(x) + \varepsilon(x), \quad \nabla V(x) = \nabla \phi_1^T \hat{W}_1 \]

Then IRL Bellman eq

\[ V(x(t)) = \int_t^{t+T} \left( Q(x) + u_k^T R u_k \right) dt + V(x(t+T)) \]

becomes

\[ \hat{W}_1^T \phi(x(t-T)) = \int_{t-T}^t \left( Q(x) + u_k^T R u_k \right) dt + \hat{W}_1^T \phi(x(t)) \]

Action Network for Control Approximation

\[ u(x) = -\frac{1}{2} R^{-1} g^T (x) \nabla \phi_1^T \hat{W}_2, \]

Define \( \Delta \phi(x(t)) \equiv \phi(x(t)) - \phi(x(t-T)) \)

Bellman eq becomes

\[ \Delta \phi(x(t))^T \hat{W}_1 + \int_{t-T}^t \left( Q(x) + \frac{1}{4} \hat{W}_2^T D_1 \hat{W}_2 \right) = 0 \]

Data-driven Online Synchronous Policy Iteration using IRL

Does not need to know $f(x)$

Theorem (Vamvoudakis & Vrabie)- Online Learning of Nonlinear Optimal Control

Let $\Delta \phi(x(t)) \equiv \phi(x(t)) - \phi(x(t - T))$ be PE. Tune critic NN weights as

$$\hat{W}_1 = -a_1 \frac{\Delta \phi(x(t))}{\left(1 + \Delta \phi(x(t))^T \Delta \phi(x(t))\right)^2} \left(\Delta \phi(x(t))^T \hat{W}_1 + \int_{t-T}^{t} \left(Q(x) + \frac{1}{4} \hat{W}_2^T \overline{D}_1 \hat{W}_2\right) d\tau\right)$$

Learning the Value

Tune actor NN weights as

$$\hat{W}_2 = -a_2 \left(F_2 \hat{W}_2 - F_1 \Delta \phi(x(t))^T \hat{W}_1\right) - \frac{1}{4} a_2 \overline{D}_1(x) \hat{W}_2 \frac{\Delta \phi(x(t))^T}{\left(1 + \Delta \phi(x(t))^T \Delta \phi(x(t))\right)^2} \hat{W}_1$$

Learning the control policy

Then there exists an $N_0$ such that, for the number of hidden layer units $N > N_0$

the closed-loop system state, the critic NN error $\hat{W}_1 = W_1 - \hat{W}_1$ and the actor NN error $\hat{W}_2 = W_1 - \hat{W}_2$ are UUB bounded.

Data set at time $[t,t+T)$

$$\left(x(t), \rho(t - T, t), x(t - T)\right)$$
Lyapunov energy-based Proof:

\[ L(t) = V(x) + \frac{1}{2} \text{tr}(\tilde{W}_1^T a_1^{-1}\tilde{W}_1) + \frac{1}{2} \text{tr}(\tilde{W}_2^T a_2^{-1}\tilde{W}_2). \]

\[ V(x) = \text{Unknown solution to HJB eq.} \]

\[ 0 = \left( \frac{dV}{dx} \right)^T f(x) + Q(x) - \frac{1}{4} \left( \frac{dV}{dx} \right)^T g R^{-1} g^T \frac{dV}{dx} \]

Guarantees stability

\[ \tilde{W}_1 = W_1 - \hat{W}_1 \]

\[ \tilde{W}_2 = W_1 - \hat{W}_2 \]

\[ W_1 = \text{Unknown LS solution to Bellman equation for given } N \]

\[ H(x,W_1,u) = W_1^T \nabla \phi_1(f + gu) + Q(x) + u^T R u = \varepsilon_H \]
Synchronous Online Solution of Optimal Control for Nonlinear Systems


A new form of Adaptive Control with TWO tunable networks

Adaptive Critic structure

Reinforcement learning

\[
\dot{\hat{W}}_1 = -a_1 \frac{\Delta \phi(x(t))}{(1 + \Delta \phi(x(t))^T \Delta \phi(x(t)))^2} \left( \Delta \phi(x(t))^T \hat{W}_1 + \int_{t-\tau}^{t} \left( Q(x) + \frac{1}{4} \hat{W}_2^T \dot{D}_1 \hat{W}_2 \right) d\tau \right)
\]

\[
\dot{\hat{W}}_2 = -a_2 \left( F_2 \hat{W}_2 - F_1 \Delta \phi(x(t))^T \hat{W}_1 \right) - \frac{1}{4} a_2 \overline{D}_1(x) \hat{W}_2 \frac{\Delta \phi(x(t))^T}{(1 + \Delta \phi(x(t))^T \Delta \phi(x(t)))^2} \hat{W}_1
\]

Desired behavior/Reference trajectory

Actor (control policy) -> Control signal

System

Output/State

Two Learning Networks
Tune them Simultaneously

A new structure of adaptive controllers
A New Class of Adaptive Control

Identify the performance value - Optimal Adaptive

Identify the system model - Indirect Adaptive

Identify the Controller - Direct Adaptive

\[ V(x) = W^T \varphi(x) \]
Simulation 1- F-16 aircraft pitch rate controller

\[
\dot{x} = \begin{bmatrix}
-1.01887 & 0.90506 & -0.00215 \\
0.82225 & -1.07741 & -0.17555 \\
0 & 0 & -1
\end{bmatrix} x + \begin{bmatrix}
0 \\
0 \\
1
\end{bmatrix} u
\]

\[Q = I, \quad R = I\]

Select quadratic NN basis set for VFA

Exact solution \[W_1^* = [p_{11} 2p_{12} 2p_{13} p_{22} 2p_{23} p_{33}]^T \]

\[= [1.4245 \quad 1.1682 \quad -0.1352 \quad 1.4349 \quad -0.1501 \quad 0.4329]^T\]

Must add probing noise to get PE

\[u(x) = - \frac{1}{2} R^{-1} g^T (x) \nabla \phi^T \hat{W}_2 + n(t) \quad \text{(exponentially decay } n(t))\]

Algorithm converges to

\[\hat{W}_1(t_f) = [1.4279 \quad 1.1612 \quad -0.1366 \quad 1.4462 \quad -0.1480 \quad 0.4317]^T.\]

\[\hat{W}_2(t_f) = [1.4279 \quad 1.1612 \quad -0.1366 \quad 1.4462 \quad -0.1480 \quad 0.4317]^T\]

\[\hat{u}_2(x) = - \frac{1}{2} R^{-1} B^T P x = - \frac{1}{2} R^{-1} \begin{bmatrix}
2x_1 & 0 & 0 \\
x_2 & x_1 & 0 \\
x_3 & 0 & x_1 \\
0 & 2x_2 & 0 \\
0 & x_3 & x_2 \\
0 & 0 & 2x_3
\end{bmatrix} \begin{bmatrix}
1.4279 \\
1.1612 \\
-0.1366 \\
1.4462 \\
-0.1480 \\
0.4317
\end{bmatrix}\]

Stevens and Lewis 2003

\[x = [\alpha \quad q \quad \delta_e]\]

Solves ARE online

\[0 = PA + A^T P + Q - PBR^{-1} B^T P\]
Critic NN parameters-Converge to ARE solution

System states
Simulation 2. – Nonlinear System

\[ \dot{x} = f(x) + g(x)u, \quad x \in \mathbb{R}^2 \]

\[
\begin{align*}
 f(x) &= \begin{bmatrix} -x_1 + x_2 \\ -0.5x_1 - 0.5x_2(1-(\cos(2x_1) + 2)^2) \end{bmatrix} \\
 g(x) &= \begin{bmatrix} 0 \\ \cos(2x_1) + 2 \end{bmatrix}.
\end{align*}
\]

\[\begin{aligned} Q = I, & \quad R = I, \\
\text{Optimal Value} & \quad V^*(x) = \frac{1}{2} x_1^2 + x_2^2, \\
\text{Optimal control} & \quad u^*(x) = -(\cos(2x_1) + 2)x_2. \end{aligned}\]

Select VFA basis set \[ \phi_1(x) = \begin{bmatrix} x_1^2 & x_1 & x_2 \end{bmatrix}^T \],

Algorithm converges to

\[ \hat{W}_1(t_f) = \begin{bmatrix} 0.5017 & -0.0020 & 1.0008 \end{bmatrix}^T. \]

\[ \hat{W}_2(t_f) = \begin{bmatrix} 0.5017 & -0.0020 & 1.0008 \end{bmatrix}^T. \]

\[ \hat{u}_2(x) = -\frac{1}{2} R^{-1} \begin{bmatrix} 0 & 2x_1 & 0 \cr x_2 & x_1 & 0 \cr 0 & 2x_2 & 0 \end{bmatrix} \begin{bmatrix} 0.5017 \\ -0.0020 \\ 1.0008 \end{bmatrix} \]
Critic NN parameters

System States

Optimal value fn.

Value fn. approx. error

Control approx error