F.L. Lewis
National Academy of Inventors

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
NSF

Talk available online at http://www.UTA.edu/UTARI/acs
Thanks to Dongbin Zhao S. Jagannathan
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
Derivation of Linear Quadratic Regulator

Rudolph Kalman 1960

System  
$$\dot{x} = Ax + Bu$$

Full system dynamics must be known

Off-line solution

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

Differentiate using Leibniz’ formula

$$\dot{V} = -(x^T Q x + u^T R u)$$

Differential equivalent is the Bellman equation

$$0 = H(x, \frac{\partial V}{\partial x}, u) = \dot{V} + x^T Q x + u^T R u = 2 \left( \frac{\partial V}{\partial x} \right)^T \dot{x} + x^T Q x + u^T R u = 2 x^T P(A x + B u) + x^T Q x + u^T R u$$

Given any stabilizing FB policy  
$$u = -K x$$

Scalar equation

Lyapunov equation

Stationarity condition  
$$0 = \frac{\partial H}{\partial u} = 2 R u + 2 B^T P x$$

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

HJB equation  
$$0 = 2 x^T P (A x - B R^{-1} B^T P x) + x^T Q x + x^T P B R^{-1} B^T P x$$

Riccati equation  
$$0 = x^T P A x + x^T A^T P x + x^T Q x - x^T P B R^{-1} B^T P x$$
Optimal Control- The Linear Quadratic Regulator (LQR)

User prescribed optimization criterion

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

\((Q, R)\)

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

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

Control

\[ K \]

System

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

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

System modeling is expensive, time consuming, and inaccurate

Off-line Design Loop Using ARE

On-line real-time Control Loop
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!
Books


New Chapters on:
- Reinforcement Learning
- Differential Games


Complex human-engineered systems involve an intersection of multiple decision makers (or agents) whose collective behavior depends on a compilation of local decisions that are based on partial information about each other and the state of the environment [1]-[4]. Strategic interactions among agents in these systems can be modeled as a multiplayer simultaneous-move game [5]-[8]. The agents involved can have conflicting objectives, and it is natural to make decisions based upon optimizing individual payoffs or costs.

Game theory has been mostly pioneered in the field of economics; [9] considered a finite win/lose game with perfect information between two players, and this classic example of computable economics stands in the long and distinguished tradition of game theory that goes back to [10] and [11]. Reference [12] discusses game theory in algorithmic modes but not in what is today referred to as algorithmic game theory after realizing the futility of
RL ADP has been developed for Discrete-Time Systems

Discrete-Time System Hamiltonian Function

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

- 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

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

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?
Four ADP Methods proposed by Paul Werbos

Critic NN to approximate:

Heuristic dynamic programming

\[ V(x_k) \]

Dual heuristic programming

\[ \frac{\partial V}{\partial x} \]

AD Heuristic dynamic programming (Watkins Q-Learning)

\[ Q(x_k, u_k) \]

AD Dual heuristic programming

\[ \frac{\partial Q}{\partial x}, \frac{\partial Q}{\partial u} \]

Action NN to approximate the Control

Bertsekas- Neurodynamic Programming

Barto & Bradtke- Q-learning proof (Imposed a settling time)
Integral Reinforcement Learning for CT systems
Synchronous IRL
Multi-player Non-zero Sum Games
Graphical Games on Networks
Zero-Sum Games
Q Learning for CT Systems
Experience Replay
Off-Policy IRL
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
\[ V(x(t)) = \int_t^\infty r(x, u) \, dt = \int_t^\infty (Q(x) + u^T R u) \, dt \]

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

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
\[
0 = \left( \frac{\partial V}{\partial x} \right)^T f(x, u(x)) + r(x, u(x)) \equiv H(x, \frac{\partial V}{\partial x}, u)
\]
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 \[ V(x(t)) = \int_{t}^{\infty} (x^T Q x + u^T R u) \, d\tau = x^T (t) P x(t) \]

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 \( u = -Kx \)

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

Optimal Control is
\[ u = -R^{-1} B^T P x = -Kx \]

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:

$$
0 = A_j^T P_j + P_j A_j + Q + K_j^T R K_j
$$

Bellman eq. = Lyapunov eq.

$$
A_j = A - BK_j
$$

Matrix equation

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

Key Idea

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

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

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

Value function \[ V(x(t)) = x^T(t)Px(t) \]

Lemma 1 - D. Vrabie - LQR case

Lyapunov equation
\[ A_c^T P + P A_c + L^T RL + Q = 0 \]

is equivalent to

Integral Reinforcement Learning form
\[ x^T(t)Px(t) = \int_t^{t+T} x^T(\tau)(Q + L^T RL)x(\tau)d\tau + x^T(t+T)Px(t+T) \]

Solves Lyapunov equation without knowing A or B

Proof:
\[ \frac{d(x^T P x)}{dt} = x^T(A_c^T P + PA_c)x = -x^T(L^T RL + Q)x \]

\[ \int_t^{t+T} x^T(Q + L^T RL)x d\tau = -\int_t^{t+T} d(x^T P x) = x^T(t)Px(t) - x^T(t+T)Px(t+T) \]
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.

\[ f(x) \text{ and } g(x) \text{ do not appear} \]

Equivalent to:

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

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

**Policy improvement**

Control gain update:

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

\[ g(x) \text{ needed for control update} \]

Initial stabilizing control is needed

Converges to solution to HJB eq.

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

D. Vrabie proved convergence to the optimal value and control

*Automatica 2009, Neural Networks 2009*
Nonlinear Case - Approximate Dynamic Programming

Value Function Approximation (VFA) to Solve Bellman Equation
- Paul Werbos (ADP), Dimitri Bertsekas (NDP)

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

Approximate value by Weierstrass Approximator Network

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

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

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

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 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 \[ V(x(t)) = x^T(t)Px(t) \]

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

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

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

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

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

Now solve by Batch least-squares
Solving the IRL Bellman Equation

Solve for value function parameters

\[
\begin{bmatrix}
    p_{11} & p_{12} \\
    p_{12} & p_{22}
\end{bmatrix}
\]

\[
W^T = \begin{bmatrix}
p_{11} & p_{12} & p_{22}
\end{bmatrix}
\]

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

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

\[
W_k^T \left[ \Delta \phi(x(t + T)) \right] = W_k^T \left[ \phi(x(t + T)) - \phi(x(t + 2T)) \right] = \int_t^{t+2T} \left( Q(x) + u_k^T Ru_k \right) dt \equiv \rho(t + T)
\]

\[
W_k^T \left[ \Delta \phi(x(t + 2T)) \right] = W_k^T \left[ \phi(x(t + 2T)) - \phi(x(t + 3T)) \right] = \int_t^{t+3T} \left( Q(x) + u_k^T Ru_k \right) dt \equiv \rho(t + 2T)
\]

Put together

\[
W_k^T \left[ \Delta \phi(x(t)) \: \Delta \phi(x(t + T)) \: \Delta \phi(x(t + 2T)) \right] = \begin{bmatrix}
\rho(t) & \rho(t + T) & \rho(t + 2T)
\end{bmatrix}
\]

Now solve by Batch least-squares

Or can use Recursive Least-Squares (RLS)
Integral Reinforcement Learning (IRL)

Solve Bellman Equation - Solves Lyapunov eq. without knowing dynamics

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

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

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

observe \(x(t)\)

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

observe \(x(t+T)\)

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

apply \(u^k = L_k x\)

apply \(u^k = L_k x\)

apply \(u^k = L_k x\)

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

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

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

update \(P\)

update \(P\)

update \(P\)

Do RLS until convergence to \(P_k\)

Or use batch least-squares

A is not needed anywhere

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

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

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

Update FB gain after Critic has converged

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

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

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

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

A new structure of adaptive controllers
The Bottom Line about Integral Reinforcement Learning

Off-line ARE Solution

The Optimal Control Solution

\[ u = -R^{-1}B^TPx = -Kx \]

Algebraic Riccati equation

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

Full system dynamics must be known
Off-line solution

Data-driven On-line ARE Solution

ARE solution can be found online in real-time by using the IRL Algorithm without knowing A matrix

Iterate for \( k=0,1,2,\ldots \)

CT Bellman eq.

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

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

Only B is needed
On line solution in real time
Uses data measurements along system trajectory

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

\[ (x(t), \rho(t,t+T), x(t+T)) \]
Optimal Control- The Linear Quadratic Regulator (LQR)

User prescribed optimization criterion

\[ J = (Q, R) \]

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

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

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

System modeling is expensive, time consuming, and inaccurate
Data-driven Online Adaptive Optimal Control
DDO

User prescribed optimization criterion

\[ J = (Q, R) \]

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

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

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

On-line Performance Loop

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

\((x(t), \rho(t, t+T), x(t+T))\)
Simulation 1- F-16 aircraft pitch rate controller

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

Stevens and Lewis 2003

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

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

ARE

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

Select quadratic NN basis set for VFA

Exact solution

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

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

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

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

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

ARE

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

ARE solution using full dynamics model \((A,B)\)

\[
P_{ARE} = \begin{bmatrix}
0.4750 & 0.4766 & 0.0601 & 0.4751 \\
0.4766 & 0.7831 & 0.1237 & 0.3829 \\
0.0601 & 0.1237 & 0.0513 & 0.0298 \\
0.4751 & 0.3829 & 0.0298 & 2.3370
\end{bmatrix}
\]
\[
0 = PA + A^T P + Q - PB R^{-1} B^T P
\]
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 = 0.1s.

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

Hence, the value estimate was updated every 1.5s.
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

\[ J = \frac{1}{2} \int_0^\infty \left( Q(x) + \int_0^u \sigma^{-1}(v) \, dv \right) \, dt \]

Approximate minimum time with smooth control inputs

\[ J = \frac{1}{2} \int_0^\infty \left( \tanh(x^T Q x) + \rho \int_0^u \sigma^{-1}(v) \, dv \right) \, dt \]
Control constrained by saturation function $\sigma(.)$

Encode constraint into Value function

\[
J(u, d) = \int_0^\infty \left( Q(x) + 2\int_0^u \sigma^{-T}(v)dv \right) dt
\]

\[
\|u\|_q^2 = 2\int_0^u \sigma^{-T}(v)dv
\]

(Used by Lyshevsky for $H_2$ control)

This is a quasi-norm

Weaker than a norm – homogeneity property is replaced by the weaker symmetry property

\[
\|x\|_q = \|-x\|_q
\]

Then \[ u = -\sigma\left( R^{-1}g(x)^T \frac{\partial V}{\partial x} \right) \] is BOUNDED
Near Minimum-Time Control

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

State Evolution for both controllers

Do not need to find switching surface
Issues with Nonlinear ADP

LS local smooth solution for Critic NN update

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

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

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

Batch LS

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

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

Selection of NN Training Set

Recursive Least-Squares RLS

Take sample points along a single trajectory
**IRL Value Iteration - Draguna Vrabie**

### IRL Policy iteration
Initial stabilizing control is needed

<table>
<thead>
<tr>
<th>Policy evaluation- IRL Bellman Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost update [ V_k(x(t)) = \int_t^{t+T} r(x,u_k) , dt + V_k(x(t+T)) ]</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>Value evaluation- IRL Bellman Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost update [ V_{k+1}(x(t)) = \int_t^{t+T} r(x,u_k) , dt + V_k(x(t+T)) ]</td>
</tr>
</tbody>
</table>

**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
Pythagoras 500 BC  
Natural Philosophy, Ethics, and Mathematics

Music, Mathematics, Gymnastics, Astronomy, Medicine

The School of Pythagoras  
esoterikoi and exoterikoi

Mathematikoi - learners
Akoustiki- listeners

Translate music to mathematical equations
Ratios in music
Numbers and the harmony of the spheres
Serenity and Self-Possession

Patterns in Nature

Fire, air, water, earth
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

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

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.

Spinal cord

Motor control 200 Hz


Deliberative evaluation

Limbic system

Motor control 200 Hz

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
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

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

Kyriakos Vamvoudakis

Critic Network

Take VFA as \( V(x) = \hat{W}_1^T \phi_1(x) + \varepsilon(x) \), \( \nabla V(x) = \nabla \phi_1^T \hat{W}_1 \)

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 \( f(x) \)

Vamvoudakis & Vrabie

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

\[
\hat{W}_1 = -a_1 \frac{\Delta \phi(x(t))}{\left(1 + \Delta \phi(x(t))^T \Delta \phi(x(t))\right)^2} \left[ \Delta \phi(x(t))^T \hat{W}_1 + \int_{t-T}^{t} \left( Q(x) + \frac{1}{4} \hat{W}_2^T D_1 \hat{W}_2 \right) d\tau \right]
\]

Learning the Value

Tune actor NN weights as

\[
\hat{W}_2 = -a_2 \left( F_2 \hat{W}_2 - F_1 \Delta \phi(x(t))^T \hat{W}_1 \right) - \frac{1}{4} a_2 D_1(x) \hat{W}_2 \frac{\Delta \phi(x(t))^T}{\left(1 + \Delta \phi(x(t))^T \Delta \phi(x(t))\right)^2} \hat{W}_1
\]

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 \( \tilde{W}_1 = W_1 - \hat{W}_1 \)

and the actor NN error \( \tilde{W}_2 = W_1 - \hat{W}_2 \) are UUB bounded.

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

\[
(x(t), \rho(t-T,t), x(t-T))
\]
Lyapunov energy-based Proof:

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

\( V(x) = \) Unknown solution to HJB eq.

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

Guarantees stability

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

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

\( W_1 = \) Unknown LS solution to Bellman equation for given \( N \)

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


A new form of Adaptive Control with TWO tunable networks

Adaptive Critic structure

Reinforcement learning

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

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

Two Learning Networks
Tune them Simultaneously

A new structure of adaptive controllers
A New Adaptive Control Structure with Multiple Tuned Loops

**Adaptive Critics**

The Adaptive Critic Architecture

![Diagram showing the Adaptive Critic Architecture](image)

- **Value update:** solve Bellman eq.
  \[ V(x) = W_1^T \phi_1(x) \]

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

- **Critic and Actor tuned simultaneously**
  Leads to ONLINE FORWARD-IN-TIME implementation of optimal control

**Optimal Adaptive Control**
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

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

\( Q = I, \quad R = I \)

Stevens and Lewis 2003

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

Solves ARE online

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

Select quadratic NN basis set for VFA

Exact solution

\( W_1^* = [p_{11} \quad 2p_{12} \quad 2p_{13} \quad p_{22} \quad 2p_{23} \quad p_{33}]^T \)

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

Must add probing noise to get PE

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

(exponentially decay n(t))

Algorithm converges to

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

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

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

System states
Simulation 2. – Nonlinear System

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

\[ f(x) = \begin{bmatrix} -x_1 + x_2 \\ -0.5x_1 - 0.5x_2 (1 - (\cos(2x_1) + 2)^2) \end{bmatrix} \]

\[ g(x) = \begin{bmatrix} 0 \\ \cos(2x_1) + 2 \end{bmatrix}. \]

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

Optimal Value \[ V^*(x) = \frac{1}{2} x_1^2 + x_2^2 \]

Optimal control \[ u^*(x) = -(\cos(2x_1) + 2)x_2. \]

Select VFA basis set \[ \phi_1(x) = [x_1^2, x_1x_2, x_2^2]^T, \]

Algorithm converges to

\[ \hat{W}_1(t_f) = [0.5017, -0.0020, 1.0008]^T. \]

\[ \hat{W}_2(t_f) = [0.5017, -0.0020, 1.0008]^T. \]

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

States

Optimal value fn.

Value fn. approx. error

Control approx error
New RL Structures Inspired by Experimental Neurocognitive Psychology
There Appear to be Multiple Reinforcement Learning Loops in the Brain

Multiple Time scales

theta rhythms 4-10 Hz

Limbic System

Known work by Doya

Reinforcement Learning

Basal ganglia-Dopamine neurons

Cerebellum-direct and inverse models

Hippocampus-Spatial maps & Task context

Brainstem

Spinal cord

Exteroceptive receptors

Interoceptive receptors

Muscle contraction and movement

Cerebral Cortex

DLPFC

ACC

OFC

gamma rhythms 30-100 Hz

Rapid skilled response using gist

Work by Dan Levine and others

Motor control 200 Hz

Timing pulses

New actor-critic Reinforcement Learning loop

Reset

Inf. olive

DLPFC

OFC

Amygdala- Emotion, intuitive
Neural dynamics of affect, gist, probability, and choice

Amygdala and OFC

1. Amygdala processes environmental cues
   Using fast heuristics, emotional triggers, and past experience

2. OFC classifies data quickly into pre-learned categories of responses
   Intuition and Limited data

3. Fast skilled response using gist and SATISFICING

4. ACC brings in slower deliberative feedback
   If there is risk

Long-term Memory

Short-term Memory
The human brain operates with MULTIPLE REINFORCEMENT LOOPS

**ACC and DLPFC**
Deliberative decision & control based on more real-time data

**Amygdala and OFC**
Intuition and Limited data
Fast skilled response based on stored memories using gist and SATISFICING

If there is mismatch or dissonance
ACC sends reset to OFC to form new category

If there is risk or stress, ACC recruits DLPFC
For deliberative decision & control based on real-time data
Perhaps optimality?

Hippocampus is invoked for detailed task knowledge

Dan Levine, Neural Dynamics of affect, gist, probability, and choice, Cognitive Sys Research, 2012

Longer-term Memory

Basal ganglia-Dopamine neurons
Basal ganglia select actual actions taken
New Fast Decision Structure Using Shunting Inhibition and Multiple Actor-Critic Learning

No System Dynamics Model needed
No Model Identification needed

Shunting Inhibition NN

\[ s_j = \frac{g_h \left( w_j^T x + w_{j0} \right) + b_j}{a_j + f_h(c_j^T x + c_{j0})} \]

Standard Value Function approximation is based only on excitatory NN synapses

\[ \hat{V}_i(x) = \hat{W}_i^T \phi_i(x) = \sum w_{i\ell} \phi_{i\ell}(x) \]

We use shunting inhibition- it is faster

New Value Function Approximation (VFA) Structures Can Encode Risk, Gist, and Emotional Salience

New Fast Decision Structure Using Adaptive Self-organizing Map and Multiple Actor-Critic Learning

1. Adaptive Self-Organizing Map

Self-organizing map (SOM) neural network is a well-known network for data clustering which has been widely used in many applications such as pattern recognition, biological modeling, signal processing, and data mining. This is known as the Kohonen learning algorithm [32]-[33]. It uses an unsupervised learning approach that divides a set of given input data into groups or clusters. In the SOM, not only are

2. New Actor-Critic ADP structure

Standard NN approximation is based on excitatory NN synapses

\[ \hat{V}_i(x) = \hat{W}_i^T \phi_i(x) = \sum w_{i\ell} \phi_{i\ell}(x) \]

Our New VFA structure is

\[ V(X(k)) = \sum_{j=1}^{J} \bar{I}_j(k) \sum_{i=1}^{I} (w_{ij} \phi_i(X(k))) \]

Indicator function depends on ASOM Classification

**New Value Function Approximation (VFA) Structures**
**Can Encode Risk, Gist, and Emotional Salience**

Data-driven Online Solution of Differential Games
Synchronous Solution of Multi-player Non Zero-sum Games
Multi-player Differential Games

500 BC

Sun Tz bin fa
Manufacturing as the Interactions of Multiple Agents

Each machine has its own dynamics and cost function
Neighboring machines influence each other most strongly
There are local optimization requirements as well as global necessities

Each process has its own dynamics
\[ \dot{\delta}_i = A\delta_i + (d_i + g_i)B_i u_i - \sum_{j \in N_i} e_{ij} B_j u_j \]

And cost function
\[ J_i(\delta_i(0),u_i,u_{-i}) = \frac{1}{2} \int_0^\infty (\delta_i^T Q_{ii} \delta_i + u_i^T R_{ii} u_i + \sum_{j \in N_i} u_j^T R_{ij} u_j) \, dt \]

Each process helps other processes achieve optimality and efficiency.
Real-Time Solution of Multi-Player NZS Games

Kyriakos Vamvoudakis, Automatica 2011

Multi-Player Nonlinear Systems

\[ \dot{x} = f(x) + \sum_{j=1}^{N} g_j(x) u_j \quad \text{Continuous-time, } N \text{ players} \]

Optimal control

\[ V_i^*(x(0), \mu_1, \mu_2, \ldots, \mu_N) = \min_{\mu_i} \int_0^{\infty} \left( Q_i(x) + \sum_{j=1}^{N} \mu_i^T R_{ij} \mu_i \right) dt; \quad i \in N \]

Nash equilibrium

\[ V_i^* = V_i(\mu_1^*, \mu_2^*, \ldots, \mu_N^*); \quad i \in N \]

Requires Offline solution of coupled Hamilton-Jacobi–Bellman eqs.

\[ 0 = (\nabla V_i)^T \left( f(x) - \frac{1}{2} \sum_{j=1}^{N} g_j(x) R_{jj}^{-1} g_j^T(x) \nabla V_j \right) + Q_i(x) + \frac{1}{4} \sum_{j=1}^{N} \nabla V_j^T g_j(x) R_{jj}^{-1} g_j^T(x) \nabla V_j, \quad V_i(0) = 0 \]

Control policies

\[ \mu_i(x) = -\frac{1}{2} R_{ii}^{-1} g_i^T(x) \nabla V_i, \quad i \in N \]

Linear Quadratic Regulator Case- coupled AREs

\[ 0 = P_i A_c + A_c^T P_i + Q_i + \sum_{j=1}^{N} P_j B_j R_{jj}^{-1} R_{ij} R_{jj}^{-1} B_j^T P_j, \quad i \in N \]

These are hard to solve
In the nonlinear case, HJB generally cannot be solved
Real-Time Solution of Multi-Player Games

Non-Zero Sum Games – Synchronous Policy Iteration

Kyriakos Vamvoudakis

Value functions
\[ V_i(x(0), \mu_1, \mu_2, \ldots, \mu_N) = \int_0^\infty (Q_i(x) + \sum_{j=1}^N \mu_i^T R_{ij} \mu_i) \, dt; \quad i \in N \]

Differential equivalent gives coupled Bellman eqs.

\[ 0 = Q_i(x) + \sum_{j=1}^N u_j^T R_{ij} u_j + (\nabla V_i)^T (f(x) + \sum_{j=1}^N g_j(x) u_j) \equiv H_i(x, \nabla V_i, u_1, \ldots, u_N), \quad i \in N \]

Policy Iteration Solution:

Solve Bellman eq.
\[ 0 = r(x, \mu_1^k, \ldots, \mu_N^k) + (\nabla V_i^k)^T \left( f(x) + \sum_{j=1}^N g_j(x) \mu_j^k \right), \quad V_i^k(0) = 0 \quad i \in N \]

Policy Update
\[ \mu_i^{k+1}(x) = -\frac{1}{2} R_{ii}^{-1} g_i^T(x) \nabla V_i^k, \quad i \in N \]

Convergence has not been proven
Hard to solve Hamiltonian equation
But this gives the structure we need for online Synchronous PI Solution
Real-Time Solution of Multi-Player Games
Kyriakos Vamvoudakis

Online Synchronous PI Solution for Multi-Player Games

Each player needs 2 NN – a Critic and an Actor

2-player case

Player 1

\[
\dot{V}_1(x) = \hat{W}_1^T \phi_1(x),
\]

Player 2

\[
\dot{V}_2(x) = \hat{W}_2^T \phi_2(x)
\]

\[\begin{align*}
V_{11}(x) &= \frac{1}{2} R_{11}^{-1} g_1^T(x) \nabla \phi_1^T \hat{W}_3, \\
V_{22}(x) &= \frac{1}{2} R_{22}^{-1} g_2^T(x) \nabla \phi_2^T \hat{W}_4
\end{align*}\]

On-Line Learning – for Player 1:

\[
\dot{\hat{W}}_1 = -a_1 \frac{\sigma_1}{(\sigma_1^T \sigma_1 + 1)^2} [\sigma_1^T \hat{W}_1 + Q_1(x) + u_1^T R_{11} u_1 + u_2^T R_{12} u_2]
\]

Learns Bellman eq. solution

\[
\dot{\hat{W}}_3 = -\alpha_3 \{(F_2 \hat{W}_3 - F_1 \sigma_3^T \hat{W}_1) - \frac{1}{4} \nabla \phi_1 g(x) R_{11}^{-1} R_{21} R_{11}^{-1} g(x) \nabla \phi_1 \hat{W}_3 m_2 \hat{W}_2 - \frac{1}{4} \nabla \phi_1 \hat{W}_3 m_1 \hat{W}_1 \}
\]

Learns control policy

Convergence is proven using Lyapunov functions

Multi-Agent Learning Guaranteed Convergence Proof

Lyapunov energy-based Proof:

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

V(x) = Unknown solution to HJB eq.

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

Guarantees stability

\[ \tilde{W}_1 = W_1 - \hat{W}_1 \]
\[ \tilde{W}_2 = W_1 - \hat{W}_2 \]

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

\[ H(x,W_1,u) = W_1^T \nabla \phi_1(f + gu) + Q(x) + u^T Ru = \varepsilon_H \]
Multi-player Games for Multi-Process Optimal Control
Data-Driven Optimization (DDO)

Optimal Performance of Each Process Depends on the Control of its Neighbor Processes

Control Policy of Each Process Depends on the Performance of its Neighbor Processes
Games on Communication Graphs

Sun Tzu's THE ART OF WAR

500 BC

孙子兵法

Sun Tz bin fa

**Key Point**

Lyapunov Functions and Performance Indices Must depend on graph topology


Hongwei Zhang, F.L. Lewis, and Abhijit Das  
Graphical Games
Synchronization- Cooperative Tracker Problem

Node dynamics \[ \dot{x}_i = Ax_i + B_i u_i, \quad x_i(t) \in \mathbb{R}^n, \quad u_i(t) \in \mathbb{R}^{m_i} \]

Target generator dynamics \[ \dot{x}_0 = Ax_0 \]

Synchronization problem \[ x_i(t) \to x_0(t), \forall i \]

Local neighborhood tracking error (Lihua Xie)
\[ \delta_i = \sum_{j \in N_i} e_{ij}(x_i - x_j) + g_i(x_i - x_0), \]

Local nbhd. tracking error dynamics
\[ \dot{\delta}_i = A\delta_i + (d_i + g_i)B_i u_i - \sum_{j \in N_i} e_{ij}B_j u_j \]

Local agent dynamics driven by neighbors’ controls

Define Local nbhd. performance index
\[ J_i(\delta_i(0), u_i, u_{-i}) = \frac{1}{2} \int_0^\infty (\delta_i^T Q_i \delta_i + u_i^T R_i u_i + \sum_{j \in N_i} u_j^T R_{ij} u_j) \, dt \]
\[ \equiv \frac{1}{2} \int_0^\infty L_i(\delta_i(t), u_i(t), u_{-i}(t)) \, dt \]


New Differential Graphical Game

Control action of player $i$

State dynamics of agent $i$
\[
\dot{\delta}_i = A\delta_i + (d_i + g_i)B_i u_i - \sum_{j \in N_i} e_{ij} B_j u_j
\]

Value function of player $i$
\[
J_i(\delta_i(0), u_i, u_{-i}) = \frac{1}{2} \int_{0}^{\infty} (\delta_i^T Q_{ii} \delta_i + u_i^T R_{ii} u_i + \sum_{j \in N_i} u_j^T R_{jj} u_j) \, dt
\]

Local Dynamics
Local Value Function
Only depends on graph neighbors
Standard Multi-Agent Differential Game

Central Dynamics

Value function of player $i$

$$J_i(z(0),u_i,u_{-i}) = \frac{1}{2} \int_0^\infty (z^T Q z + \sum_{j=1}^N u_j^T R_{ij} u_j) \, dt$$
New Definition of Nash Equilibrium for Graphical Games

Def. Local Best response. $u_i^*$ is said to be agent $i$’s local best response to fixed policies $u_{-i}$ of its neighbors if

$$J_i (u_i^*, u_{-i}) \leq J_i (u_i, u_{-i}), \ \forall u_i$$

Def: Interactive Nash equilibrium

$$\{u_1^*, u_2^*, ..., u_N^*\}$$ are in Interactive Nash equilibrium if

1. $J_i^* \triangleq J_i (u_i^*, u_{G-i}^*) \leq J_i (u_i, u_{G-i}^*), \ \forall i \in N$ i.e. they are in Nash equilibrium

2. There exists a policy $u_j$ such that

$$J_i (u_j, u_{G-j}^*) \neq J_i (u_j^*, u_{G-j}^*), \ \forall i, j \in N$$

That is, every player can find a policy that changes the value of every other player.

A restriction on what sorts of performance indices can be selected in multiplayer graph games.

A condition on the reaction curves (Basar and Olsder) of the agents

This rules out the disconnected counterexample.
Graphical Game Solution Equations

Value function
\[ V_i(\delta_i(t)) = \frac{1}{2} \int_0^\infty (\delta_i^T Q_{ii} \delta_i + u_i^T R_i u_i + \sum_{j \in N_i} u_j^T R_{ij} u_j) \, dt \]

Differential equivalent (Leibniz formula) is Bellman’s Equation
\[ H_i(\delta_i, \frac{\partial V_i}{\partial \delta_i}, u_i, u_{-i}) = \frac{\partial V_i}{\partial \delta_i} \left( A \delta_i + (d_i + g_i) B_i u_i - \sum_{j \in N_i} e_{ij} B_j u_j \right) + \frac{1}{2} \delta_i^T Q_{ii} \delta_i + \frac{1}{2} u_i^T R_i u_i + \frac{1}{2} \sum_{j \in N_i} u_j^T R_{ij} u_j = 0 \]

Stationarity Condition
\[ 0 = \frac{\partial H_i}{\partial u_i} \Rightarrow u_i = -(d_i + g_i) R_{ii}^{-1} B_i^T \frac{\partial V_i}{\partial \delta_i} \]

1. Coupled HJ equations
\[ \frac{\partial V_i}{\partial \delta_i} A_i^c + \frac{1}{2} \delta_i^T Q_{ii} \delta_i + \frac{1}{2} (d_i + g_i)^2 \frac{\partial V_i}{\partial \delta_i} B_i R_{ii}^{-1} B_i^T \frac{\partial V_i}{\partial \delta_i} + \frac{1}{2} \sum_{j \in N_i} (d_j + g_j)^2 \frac{\partial V_j}{\partial \delta_j} B_j R_{jj}^{-1} R_{ij} B_j^T \frac{\partial V_j}{\partial \delta_j} = 0, \ i \in N \]

\[ H_i(\delta_i, \frac{\partial V_i}{\partial \delta_i}, u_i^*, u_{-i}^*) = 0 \]

where \[ A_i^c = A \delta_i - (d_i + g_i) R_{ii}^{-1} B_i^T \frac{\partial V_i}{\partial \delta_i} + \sum_{j \in N_i} e_{ij} (d_j + g_j) B_j R_{jj}^{-1} B_j^T \frac{\partial V_j}{\partial \delta_j}, \ i \in N \]

Now use Synchronous PI to learn optimal Nash policies online in real-time as players interact
Online Solution of Graphical Games

Kyriakos Vamvoudakis

Use Reinforcement Learning

POLICY ITERATION

Algorithm 1. Policy Iteration (PI) Solution for N-player distributed games.

Step 0: Start with admissible initial policies $u_i^0$, $\forall i$.

Step 1: (Policy Evaluation) Solve for $V_i^k$ using (14)

$$H_i(\delta_i, \frac{\partial V_i^k}{\partial \delta_i}, u_i^k, u_{-i}^k) = 0, \forall i = 1, \ldots, N$$

(38)

Step 2: (Policy Improvement) Update the N-tuple of control policies using

$$u_{i}^{k+1} = \arg\min_{u_i} H_i(\delta_i, \frac{\partial V_i^k}{\partial \delta_i}, u_i, u_{-i}^k), \forall i = 1, \ldots, N$$

which explicitly is

$$u_{i}^{k+1} = -(d_i + g_i)R^{i^{-1}}B_i^T \frac{\partial V_i^k}{\partial \delta_i}, \forall i = 1, \ldots, N.$$  

(39)

Go to step 1.

On convergence End

Multi-agent Learning

Convergence Results

Theorem 3. Convergence of Policy Iteration algorithm when only $i^{th}$ agent updates its policy and all players $u_{-i}$ in the neighborhood do not change. Given fixed neighbors policies $u_{-i}$, assume there exists an admissible policy $u_i$. Assume that agent $i$ performs Algorithm 1 and the its neighbors do not update their control policies. Then the algorithm converges to the best response $u_i$ to policies $u_{-i}$ of the neighbors and to the solution $V_i$ to the best response HJ equation (36).

The next result concerns the case where all nodes update their policies at each step of the algorithm. Define the relative control weighting as $\rho_{ij} = \overline{\sigma}(R_{ij}^{-1}R_{ij})$, where $\overline{\sigma}(R_{ij}^{-1}R_{ij})$ is the maximum singular value of $R_{ij}^{-1}R_{ij}$.

Theorem 4. Convergence of Policy Iteration algorithm when all agents update their policies. Assume all nodes $i$ update their policies at each iteration of PI. Then for small enough edge weights $e_{ij}$ and $\rho_{ij}$, $\mu_i$ converges to the global Nash equilibrium and for all $i$, and the values converge to the optimal game values $V_i^k \rightarrow V_i^\ast$.
Data-driven Online Solution of Differential Games
Zero-sum 2-Player Games and H-infinity Control
H-Infinity Control Using Neural Networks

Disturbance Rejection

System

\[
\begin{align*}
\dot{x} &= f(x) + g(x)u + k(x)d \\
y &= h(x) \\
z &= \begin{bmatrix} y^T \\ u^T \end{bmatrix}^T \\
u &= l(x)
\end{align*}
\]

Performance output

disturbance

L₂ Gain Problem

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

\[
\begin{align*}
\int_0^\infty \|z(t)\|^2 dt &= \int_0^\infty (h^T h + \|u\|^2) dt \\
\int_0^\infty \|d(t)\|^2 dt &\leq \gamma^2
\end{align*}
\]

For all L₂ disturbances. And a prescribed gain \(\gamma^2\)

Zero-Sum differential game

Nature as the opposing player
Define 2-player zero-sum game as

\[ V^*(x(0)) = \min_u \max_d V(x(0), u, d) = \min_u \max_d \int_0^\infty \left( h^T(x)h(x) + u^T R u - \gamma^2 \|d\|^2 \right) dt \]

The game has a unique value (saddle-point solution) iff the Nash condition holds

\[ \min_u \max_d V(x(0), u, d) = \max_d \min_u V(x(0), u, d) \]

A necessary condition for this is the Isaacs Condition

\[ \min_u \max_d H(x, \nabla V, u, d) = \max_d \min_u H(x, \nabla V, u, d) \]

Stationarity Conditions

\[ 0 = \frac{\partial H}{\partial u}, \quad 0 = \frac{\partial H}{\partial d} \]
Online Zero-Sum Differential Games

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

Cost
\[ V(x(t), u, d) = \int_{t}^{\infty} \left( h^T h + u^T Ru - \gamma^2 \|d\|^2 \right) dt \equiv \int_{t}^{\infty} r(x, u, d) \, dt \]

Game saddle point solution found from Hamiltonian - ZS Game BELLMAN EQUATION
\[ H(x, \frac{\partial V}{\partial x}, u, d) = h^T h + u^T Ru - \gamma^2 \|d\|^2 + (\nabla V)^T (f(x) + g(x)u + k(x)d) = 0 \]

Optimal control/dist. policies found by stationarity conditions
\[ 0 = \frac{\partial H}{\partial u}, \quad 0 = \frac{\partial H}{\partial d} \]
\[ u = - \frac{1}{2} R^{-1} g^T (x) \nabla V \quad \quad d = \frac{1}{2\gamma^2} k^T (x) \nabla V \]

HJI equation
\[ 0 = H(x, \nabla V, u^*, d^*) \]
\[ = h^T h + \nabla V^T (x) f(x) - \frac{1}{4} \nabla V^T (x) g(x) R^{-1} g^T (x) \nabla V (x) + \frac{1}{4\gamma^2} \nabla V^T (x) kk^T \nabla V (x) \]


Double Policy Iteration Algorithm to Solve HJI

Add inner loop to solve for available storage

1. For a given control policy \( u_j(x) \) solve for the value \( V_{j+1}(x(t)) \)

2. Set \( d^0 = 0 \). For \( i=0,1,... \) solve for \( V^i_j(x(t)), d^{i+1} \)

\[
0 = h^T h + \nabla V^i_j(x)(f + gu_x + kd^i) + u^T_j Ru_j - \gamma^2 \left\| d^i \right\|^2
\]

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

On convergence set \( V_{j+1}(x) = V^i_j(x) \)

3. Improve policy:

\[
u_{j+1}(x) = -\frac{1}{2} R^{-1} g^T(x) \nabla V_{j+1}
\]

- Convergence proved by Van der Schaft if can solve nonlinear Lyapunov equation exactly
- Abu Khalaf & Lewis used NN to approximate \( V \) for nonlinear systems and proved convergence

ONLINE SOLUTION- Use IRL to solve the ZS Bellman eq. and update the actors at each step
Actor-Critic structure - three time scales

\[ P^{i(k)}_u = P^{i-1}_u + Z^{i(k)}_u \]

\[ \dot{V} = x^T C^T Cx + \hat{u}^T \hat{u}, \text{ if } i=1 \]

\[ \dot{V} = \hat{w}^T \hat{w} + \hat{u}^T \hat{u}, \text{ if } i>1 \]

Controller/Player 1

\[ u = -B_i^T P^{i(k-1)}_u x \]

System

\[ \dot{x} = Ax + B_2 u + B_1 w; \ x_0 \]

Disturbance/Player 2

\[ w = B_1^T P^{i-1}_w x \]
New Developments in IRL for CT Systems
Q Learning for CT Systems
Experience Replay
Off-Policy IRL
Q learning for CT Systems

Biao Luo, Derong Liu, T. Huang, 2014

\[
Q^\mu(x(t), \bar{u}[t:t+T]) = \int_t^{t+T} e^{-\gamma(t-\tau)} \left( x^T S x + \mu^T R \mu \right) d\tau + e^{-\gamma T} V^\mu(x(t+T))
\]

Algorithm 3 Q-Learning Algorithm for Continuous-Time Systems

1. **Q-function update.** Solve for Q-function using

\[
Q^{\mu+1}(x(t), \bar{u}[t:t+T]) = \int_t^{t+T} e^{-\gamma(t-\tau)} \left( x^T S x + u^T R u \right) d\tau + e^{-\gamma T} Q^\mu(x(t+T), \bar{u}[t+T:\infty]).
\]

(25)

2. **Policy improvement.** Update the policy using

\[
\mu^{j+1}(x(t)) = \arg \min_{\bar{u}[t:t+T]} Q^{\mu+1}(x(t), \bar{u}[t:t+T])
\]

(26)

CT Q function is quadratic in \( x, u \), and the derivatives of \( u \) -

\[
\int_t^{t+T} e^{-\gamma(t-\tau)} \left( x(\tau)^T S x(\tau) + u(\tau)^T R u(\tau) \right) d\tau = Z_{k-1}^T(t) \left( M_{k-1}^T S_k M_{k-1} + \tilde{R}_k \right) Z_{k-1}(t) + E_k(t + T)
\]

\[
E_k(t + T) = e^{-\gamma T} V^\mu(x(t+T))
\]

\[
Z_k(t) = \left[ \begin{array}{c} x(t) \\ u(t) \\ \vdots \\ u^{(k)}(t) \end{array} \right]
\]
IRL with Experience Replay

Humans use memories of past experiences to tune current policies

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

Value function:

\[ v(x(t)) = \int_{t-T}^{\infty} \left( Q(x(\tau)) + 2 \int_{0}^{u} (\lambda \tanh^{-1}(v/\lambda))^T R \, dv \right) d\tau \]

\[ p(t) = \int_{t-T}^{t} \left( Q + 2 \int_{0}^{u} (\lambda \tanh^{-1}(v/\lambda))^T R \, dv \right) d\tau \]

IRL Bellman equation:

\[ V(x(t - T)) = p(t) + V(x(t)) \]

VFA:

\[ \hat{V}(x) = \hat{\phi}(x) \]

\[ \hat{\phi}^T \left[ \phi(x(t)) - \phi(x(t - T)) \right] + p(t) = 0 \]

\[ \Delta \phi(x(t)) = \phi(x(t)) - \phi(x(t - T)) \]

The samples are stored in a history stack. To collect data in the history stack, consider \( \Delta \phi_j \) and \( p_j \) as evaluated values of \( \Delta \phi(t) \) and \( p(t) \) (see (17) and (26)) at the recorded time \( t_j \). That is,

\[ \Delta \phi_j = \Delta \phi(t_j) = \phi(x(t_j)) - \phi(x(t_j - T)) \quad (27) \]

and

\[ p_j = p(t_j) = \int_{t_j - T}^{t_j} \left( Q + 2 \int_{0}^{u} (\lambda \tanh^{-1}(v/\lambda))^T R \, dv \right) d\tau \quad (28) \]

NN weight tuning uses past samples

\[ \hat{\phi}_1(t) = -\alpha_1 \frac{\Delta \phi(t)}{(1 + \Delta \phi(t)^T \Delta \phi(t))^2} \left( p(t) + \Delta \phi(t)^T \hat{\phi}_1(t) \right) \]

\[ -\alpha_1 \sum_{j=1}^{i} \frac{\Delta \phi_j}{(1 + \Delta \phi_j^T \Delta \phi_j)^2} \left( p_j + \Delta \phi_j^T \hat{\phi}_1(t) \right) \]
Off-Policy Reinforcement Learning

Humans can learn optimal policies while actually playing suboptimal policies

On-policy RL

**Target policy**: The policy that we are learning about.

**Behavior policy**: The policy that generates actions and behavior

Target policy and behavior policy are the same

Sutton and Barto Book
Off-policy RL

Humans can learn optimal policies while actually applying suboptimal policies

Target policy and behavior policy are different


**Off-policy IRL**

Yu Jiang & Zhong-Ping Jiang, Automatica 2012

Humans can learn optimal policies while actually applying suboptimal policies

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

**Value**

\[ J(x) = \int_t^\infty r(x(\tau), u(\tau)) d\tau \]

**On-policy IRL**

\[
J^{[i]}(x(t)) - J^{[i]}(x(t-T)) = -\int_{t-T}^t Q(x)d\tau - \int_{t-T}^t u^{[i]T} Ru^{[i]} d\tau
\]

\[
u^{[i+1]} = -\frac{1}{2} R^{-1} g^T J_x^{[i]}
\]

**Off-policy IRL**

\[
\dot{x} = f(x) + gu^{[i]} + g(u - u^{[i]})
\]

\[
J^{[i]}(x(t)) - J^{[i]}(x(t-T)) = -\int_{t-T}^t Q(x)d\tau - \int_{t-T}^t u^{[i]T} Ru^{[i]} d\tau + 2\int_{t-T}^t R(u^{[i]} - u) d\tau
\]

**DDO**

This is a linear equation for \( J^{[i]} \) and \( u^{[i+1]} \)

They can be found simultaneously online using measured data using Kronecker product and VFA

1. Completely unknown system dynamics
2. Can use applied \( u(t) \) for –
Off-policy for Multi-player NZS Games

\[ \dot{x} = f(x) + \sum_{j=1}^{N} g(x)u_j \]

\[ V_i(x(t)) = \int_{t}^{\infty} (r_i(x,u_1,u_2,\ldots,u_N))d\tau = \int_{t}^{\infty} (Q_i(x) + \sum_{j=1}^{N} u_j^T R_j u_j) d\tau \]

1. Solve online using measured data for \( V_i^{[k]}, u_i \)
2. Completely unknown dynamics
3. Add exploring noise with no bias

Algorithm 1:

Step 1: Start with stabilizing initial policies \( u_1^{[0]}, u_2^{[0]}, \ldots, u_N^{[0]} \)

Step 2: Given the \( N \)-tuple of policies \( u_1^{[k]}, u_2^{[k]}, \ldots, u_N^{[k]} \), solve for the \( N \)-tuple of costs \( V_1^{[k]}(x(t)), V_2^{[k]}(x(t)), \ldots, V_N^{[k]}(x(t)) \)

using

\[ 0 = \nabla V_i^{[k]}(f(x) + \sum_{j=1}^{N} g(x)u_j^{[k]} + r_i(x,u_1^{[k]},u_2^{[k]},\ldots,u_N^{[k]}) \]

with \( V_i^{[k]}(0) = 0 \).

Step 3: Update the \( N \)-tuple of control policies using:

\[ u_i^{[k+1]} = \arg \min_{u_i} [H_i(x, \nabla V_i, u_1, \ldots, u_N)] \]

which explicitly is

\[ u_i^{[k+1]} = -\frac{1}{2} R_i^{-1} g^T(x) \nabla V_i^{[k]} \]
Off-policy IRL for ZS Games – H-infinity Control

Optimal tracker

Extended state
\[ X(t) = [e(t)^T \ r(t)^T] \in \mathbb{R}^{2n} \]

Including ref. traj. dynamics
\[ \dot{X}(t) = F(X(t)) + G(X(t))u(t) + K(X(t))d(t) \]

On-policy

Off-policy
\[ \dot{X} = F + G u + K d + G (u - u_i) + K (d - d_i) \]

DDO

1. Solve online using data for \( V_i, u_{i+1}, d_{i+1} \)
2. Completely unknown dynamics
3. Disturbance does not need to be specified
4. Add exploring noise with no bias

Algorithm 1: Offline RL algorithm for solving the tracking HJI equation

Initialization: Start with an admissible stabilizing control policy \( u_0 \)

1. For a control input \( u_i \) and disturbance policy \( d_i \), find \( V_i \) using the following Bellman equation
\[ e^{-\alpha T}V_i(X(t + T)) - V_i(X(t)) = \int_t^{t+T} e^{-\alpha(\tau-t)} ( - X^T Q X - u_i^T R u_i + \gamma^2 d_i^T d_i ) d\tau \]
2. Update the disturbance using
\[ d_{i+1} = \text{arg max}_d \left[ H(V_i, u_i, d) = \frac{1}{2\gamma^2} K^T V_{xi} \right] \]
and the control policy using
\[ u_{i+1} = \text{arg min}_u \left[ H(V_i, u, d) = -\frac{1}{2} R^{-1} G^T V_{xi} \right] \]

Algorithm 2: Online Off-policy RL algorithm for solving tracking HJI equation

1. Initialization: Start with a control policy \( u_0 \)
2. Solve the following Bellman equation for \( V_i \) and \( u_{i+1} \) simultaneously
\[ e^{-\alpha T}V_i(X(t + T)) - V_i(X(t)) = \int_t^{t+T} e^{-\alpha(\tau-t)} (- X^T Q X - u_i^T R u_i + \gamma^2 d_i^T d_i ) d\tau \]
\[ + \int_t^{t+T} e^{-\alpha(\tau-t)} (-2u_i^T R (u - u_i) + 2\gamma^2 d_{i+1}^T (d - d_i)) d\tau \]

H. Li, Derong Liu, D. Wang, IEEE TASE 2015
Output Synchronization of Heterogeneous MAS
Heterogeneous Multi-Agents

\[ \dot{x}_i = A_i x_i + B_i u_i \]
\[ y_i = C_i x_i \]

Leader

\[ \dot{\zeta}_0 = S \zeta_0 \]
\[ y_0 = R \zeta_0 \]

Output regulation error

\[ \eta_i(t) = y_i(t) - y_0(t) \to 0 \]

Output regulator equations

\[ A_i \Gamma_i + B_i \Gamma_i = \Pi_i S \]
\[ C_i \Pi_i = R \]

Dynamics are different, state dimensions can be different
o/p reg eqs capture the common core of all the agents dynamics
And define a synchronization manifold
Pioneered by Jie Huang

MAS
\[ \dot{x}_i = A_i x_i + B_i u_i \]
\[ y_i = C_i x_i \]

Leader
\[ \dot{\zeta}_0 = S \zeta_0 \]
\[ y_0 = R \zeta_0 \]

Output regulator equations
\[ y_i(t) - y_0(t) \to 0, \forall i \]

A standard o/p regulation Controller
\[ \dot{\zeta}_i = S \zeta_i + c \left[ \sum_{j=1}^{N} a_{ij} (\zeta_j - \zeta_i) + g (\zeta_0 - \zeta_i) \right] \]
\[ u_i = K_{1i} (x_i - \Pi_i \zeta_i) + \Gamma_i \zeta_i = K_{1i} x_i + (\Gamma_i - K_{1i} \Pi_i) \zeta_i \equiv K_{1i} x_i + K_{2i} \zeta_i \]

Must know leader’s dynamics \( S, R \)
And solve the o/p regulator equations
Optimal Output Synchronization of Heterogeneous MAS Using Off-policy IRL
Nageshrao, Modares, Lopes, Babuska, Lewis

\[ \dot{x}_i = A_i x_i + B_i u_i \quad \text{Leader} \]
\[ y_i = C_i x_i \]

**Optimal Tracker Problem**

Augmented Systems

\[ X(t) = \left[ x_i(t)^T \ z_0^T \right]^T \in \mathbb{R}^{n_i+p} \]
\[ \dot{X}_i = T_i X_i + B_{i1} u_i \]
\[ T_i = \begin{bmatrix} A_i & 0 \\ 0 & S \end{bmatrix}, B_{i1} = \begin{bmatrix} B_i \\ 0 \end{bmatrix} \]

Performance index

\[ V(X_i(t)) = \int_t^\infty e^{-\gamma(t-\tau)} X_i^T (C_{i1}^T Q_i C_{i1} + K_i^T W_i K_i) X_i \ d\tau \]
\[ = X_i(t)^T P_i X_i(t) \]

Control

\[ u_i = K_{i1} x_i + K_{i2} \zeta_0 = K_i x_i \]
Optimal Tracker Solution by Reinforcement Learning

\[ K_i = [K_{1i}, K_{2i}] = -W_i^{-1}B_{1i}^T P_i \]

Tracker ARE

\[ T_i^T P_i + T_i P_i - \gamma_i P_i + C_{1i}^T Q_i C_{1i} - P_i B_{2i} W_i^{-1} B_{2i}^T P_i = 0 \]

**Algorithm 1. On-policy IRL State-feedback algorithm**

Policy Evaluation- Solve IRL Bellman equation

\[ e^{-\gamma_i \delta t} X_i (t + \delta t)^T P_i \gamma X_i (t + \delta t) - X_i (t)^T P_i \gamma X_i (t) = -\int_t^{t+\delta t} e^{-\gamma_i (\tau-t)} (y_i - y_0)^T Q_i (y_i - y_0) d\tau \]

Policy Update-

\[ K_i^{\kappa+1} = [K_{1i}^{\kappa+1}, K_{2i}^{\kappa+1}] = -W_i^{-1}B_{1i}^T P_i^\kappa \]

Theorem- Algorithm 1 converges to the solution to the ARE

Bellman equation is solved using RLS or batch LS
It requires a Persistence of Excitation (PE) condition that may be hard to satisfy

Must know \( B_{1i} \)
Off-Policy RL

Tracker dynamics

\[ \dot{X}_i = T_i X_i + B_i u_i \]

Rewrite as

\[ \dot{X}_i = (T_i + B_i K_i^\kappa) X_i + B_i (u_i - K_i^\kappa X_i) \equiv \bar{T}_i X_i + B_i (u_i - K_i^\kappa X_i) \]

Now the Bellman equation becomes

\[
e^{-\gamma_i t} X_i(t + \delta t)^T P_i^\kappa X_i(t + \delta t) - X_i(t)^T P_i^\kappa X_i(t) = -\int_t^{t+\delta t} e^{-\gamma_i (\tau-t)} (y_i - y_0)^T Q_i (y_i - y_0) d\tau \\
+ 2\int_t^{t+\delta t} e^{-\gamma_i (\tau-t)} (u_i - K_i^\kappa X_i)^T W_i K_i^{\kappa+1} X_i d\tau
\]

Extra term containing $K_i^{\kappa+1}$

**Algorithm 2.** *Off-policy IRL Data-based algorithm*

Iterate on this equation and solve for $P_i^\kappa, K_i^{\kappa+1}$ simultaneously at each step

Note about probing noise  
If $u_i = K_i^\kappa X_i + e$ then $(u_i - K_i^\kappa X_i) = e$

Do not have to know any dynamics

agent \[ \dot{x}_i = A_i x_i + B_i u_i \]
\[ y_i = C_i x_i \]

Or leader \[ \dot{\zeta}_0 = S \zeta_0 \]
\[ y_0 = R \zeta_0 \]
Theorem- Off-policy Algorithm 2 converges to the solution to the ARE

\[ T_i^T P_i + T_i P_i - \gamma_i P_i + C_{i1}^T Q_i C_{i1} - P_i B_{i1} W_i^{-1} B_{i1}^T P_i = 0 \]

Theorem- o/p reg eq solution

Let

\[ P_i = \begin{bmatrix} P_{11}^i & P_{12}^i \\ P_{21}^i & P_{22}^i \end{bmatrix} \]

Then the solution to the output regulator equations

\[ A_i \Pi_i + B_i \Gamma_i = \Pi_i S \]
\[ C_i \Pi_i = R \]

Is given by

\[ \Pi_i = -(P_{11}^i)^{-1} P_{12}^i \]
\[ \Gamma_i = K_{i2} - K_{i1} (P_{11}^i)^{-1} P_{12}^i \]

Do not have to know the
Agent dynamics or the leader’s dynamics (S,R)
Applications of Reinforcement Learning

Human-Robot Interactive Learning
Industrial process control- Mineral grinding in Gansu, China
Resilient Control to Cyber-Attacks in Networked Multi-agent Systems
Decision & Control for Heterogeneous MAS (different dynamics)
Intelligent Operational Control for Complex Industrial Processes

Jinliang Ding

Professor Chai Tianyou

State Key Laboratory of Synthetical Automation for Process Industries
Northeastern University

May 20, 2013


Manufacturing as the Interactions of Multiple Agents

Each machine has its own dynamics and cost function.

Neighboring machines influence each other most strongly.

There are local optimization requirements as well as global necessities.
Production line for mineral processing plant

Mineral Processing Plant in Gansu China
Existing Manual Control for
Plant production indices, unit operational indices, and unit process control for a production line
Automated online reinforcement learning for determining operational indices

Implemented by Jingliang Ding and Chai Tianyou’s group in biggest mineral processing factory of hematite iron ore in China, Gansu Province.

Savings of 30.75 million RMB per year were realized by implementing this automated optimization procedure instead of the standard industry practice of human operator selection of process operational indices.
RL for Human-Robot Interaction (HRI)


PR2 meets Isura
Impedance Control

Robot dynamics

The rigid body dynamics in task coordinates [5] is:
\[ \Lambda(x, \dot{x}) \ddot{x} + \mu(x, \dot{x}) \dot{x} + J^{-T} g(x) = J^{-T} \tau + f, \] (2)
\[ \Lambda(x, \dot{x}) = J^{-T} M J^{-1}, \quad \mu(x, \dot{x}) = J^{-1} (C - M J^{-1} J) J^{-1}. \]

Prescribed Error system

Impedance control in the task space consists of the following control objective [6]:
\[ \Lambda_d \ddot{e}_x + D_d \dot{e}_x + K_d e_x = e_f \] (3)
where \( e_x = x - x_d \) is the position error between the actual position \( x \) and the reference position \( x_d \); \( e_f = f - f_d \) measures

Control torque depends on Impedance model parameters

\[ \tau = u + J^T \tilde{K} e_x + J^T \tilde{D} \dot{e}_x + J^T \tilde{\Lambda}_d e_f, \] (4)
where \( u = g + J^T (\Lambda \ddot{x}_d + \mu \dot{x}_d) \) (5)
\[ \tilde{K}_d = \Lambda \Lambda_d^{-1} K_d, \quad \tilde{D}_d = \Lambda \Lambda_d^{-1} D_d + \mu, \quad \tilde{\Lambda}_d = \Lambda \Lambda_d^{-1} - I. \]
Human Performance Factors Studies

Human task learning has 2 components:
1. Human learns a robot dynamics model to compensate for robot nonlinearities
2. Human learns a task model to properly perform a task

Inner Robot Specific Control Loop
INDEPENDENT OF TASK

Outer Task Specific Control Loop
INDEPENDENT OF ROBOT DETAILS

This paper develops a two-loop design approach for adaptive admittance control for human-robot interaction. This approach follows the human factors studies [44], [45] that indicate human learning in task performance has two components: a robot-specific component whereby a robot dynamics model is learned to compensate for robot nonlinearities, and a component where task-related details are learned. Here, an inner
RL for Human-Robot Interactions


Robot-specific inner control loop

Task-specific outer-loop control design
Using IRL Optimal Tracking

Inspired by work of Sam S. Ge

Implemented on PR2 robot