Adaptive Learning Structures for Real-Time Optimal Control and Dynamic Games

Talk available online at http://ARRI.uta.edu/acs
He who exerts his mind to the utmost knows nature’s pattern.
The way of learning is none other than finding the lost mind.

Man’s task is to understand patterns in nature and society.

Meng Tz
500 BC

Mencius
- Optimal Control
- Reinforcement learning
- Integral Reinforcement Learning for Continuous-time systems
- Policy Iteration for ZS games
- Synchronous Policy Iteration
- PI for Solving Dynamic Games online
It is man’s obligation to explore the most difficult questions in the clearest possible way and use reason and intellect to arrive at the best answer.

Man’s task is to understand patterns in nature and society.

The first task is to understand the individual problem, then to analyze symptoms and causes, and only then to design treatment and controls.

Ibn Sina 1002-1042
(Avicenna)
Cell Homeostasis

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

Permeability control of the cell membrane

Rocket Orbit Injection

Dynamics

\[ \dot{r} = w \]

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

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

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

Objectives

Get to orbit in minimum time
Use minimum fuel
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

\[ J = \frac{1}{2} \int_0^\infty \left( \tanh(x^T Q x) + \rho \int_0^u \sigma^{-1}(v) \, dv \right) \, dt \]
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:
- 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
We want to find optimal control solutions online in real-time using adaptive control techniques.

Adaptive Control Structures for:

A. Optimal control  B. Zero-sum games  C. Non zero-sum games

1. System dynamics
2. Value/cost function
3. Bellman equation
4. HJ solution equation (Riccati eq.)
5. Policy iteration – gives the structure we need
Books


New Chapters on:
- Reinforcement Learning
- Differential Games

Continuous-Time Optimal Control

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

Cost/value \[ V(x(t)) = \int_t^\infty r(x,u) \, dt = \int_t^\infty (Q(x) + u^TRu) \, dt \]

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

HJB equation

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

Off-line solution

HJB hard to solve. May not have smooth solution.

Dynamics must be known
Optimal Control: 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
\[ 0 = H(x, \frac{\partial V}{\partial x}, u) = \dot{V} + x^T Q x + u^T R u = 2 \left( \frac{\partial V}{\partial x} \right)^T \dot{x} + x^T Q x + u^T R u = 2x^T P (A x + B u) + x^T Q x + u^T R u \]

Given any stabilizing FB policy \[ u = -K x \]

The cost value is found by solving \textbf{Lyapunov equation} = \textbf{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
\[ 0 = PA + A^T P + Q - P B R^{-1} B^T P \]

Full system dynamics must be known
Off-line solution
We want to find optimal control solutions
Online in real-time
Using adaptive control techniques
Without knowing the full dynamics
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.

Reinforcement Learning

1. Apply a control. Evaluate the benefit of that control.
2. Improve the control policy.

RL finds optimal policies by evaluating the effects of suboptimal policies.
Joao's chart

Optimal control

RL?
Different methods of learning

Reinforcement learning
Ivan Pavlov 1890s

We want OPTIMAL performance
- ADP- Approximate Dynamic Programming

Actor-Critic Learning
Desired performance

Adaptive Learning system
Actor

Tune actor
Reinforcement signal
Critic

System
Control Inputs
Environment
outputs

Sutton & Barto book

RL 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?
Continuous-Time Optimal Control

To find online methods for optimal control

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
\[ 0 = \left( \frac{dV^*}{dx} \right)^T f + Q(x) - \frac{1}{4} \left( \frac{dV^*}{dx} \right)^T g R^{-1} g^T \frac{dV^*}{dx} \]
\[ , \quad V(0) = 0 \]
Optimal Control: 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
\[ 0 = H(x, \frac{\partial V}{\partial x}, u) = \dot{V} + x^T Q x + u^T R u = 2 \left( \frac{\partial V}{\partial x} \right)^T \dot{x} + x^T Q x + u^T R u = 2 x^T P (A x + B u) + x^T Q x + u^T R u \]

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
CT Policy Iteration – a Reinforcement Learning Technique

To avoid solving HJB equation

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

Find cost for any given admissible \( u(x) \)

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

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

CT Bellman equation

Scalar equation

---

**Policy Iteration Solution**

Pick stabilizing initial control policy

**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

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

---

- 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
LQR Policy iteration = Kleinman algorithm

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

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

\[
A_j = A - BK_j
\]

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
Lemma 1 – Draguna Vrabie

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

Is equivalent to

Integral reinf. form for the CT Bellman eq.

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

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

Proof:

\[ \frac{d(V(x))}{dt} = \left( \frac{\partial V}{\partial x} \right)^T f(x,u) = -r(x,u) \]

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

Allows definition of temporal difference error for CT systems

\[ e(t) = -V(x(t)) + \int_{t}^{t+T} r(x,u) \, d\tau + V(x(t+T)) \]
Lemma 1 – Draguna Vrabie

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

Is equivalent to \textbf{Integral reinf. form for the CT Bellman eq.}

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

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

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

Allows definition of temporal difference error for CT systems
Lemma 1 - D. Vrabie- LQR case

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

\[ A_c = A - BL \]

is equivalent to

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

Solves Lyapunov equation without knowing A or B

Proof:

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

\[
\int_t^{t+T} x^T (Q + L^T RL) xd\tau = - \int_t^{t+T} d(x^T P x) = x^T (t)Px(t) - x^T (t + T)Px(t + T)
\]
Lemma 1 - D. Vrabie- LQR case

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

is equivalent to \( A_c = A - BL \)

\[ x^T (t) P x(t) = \int_{t}^{t+T} x^T (\tau)(Q+L^T RL)x(\tau)d\tau + x^T (t+T) P x(t+T) \]

Solves Lyapunov equation without knowing A or B
Integral Reinforcement Learning (IRL)- Draguna Vrabie

**IRL Policy iteration**

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

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

D. Vrabie proved convergence to the optimal value and control
Integral Reinforcement Learning (IRL) - Draguna Vrabie

CT LQR Case

Value function is quadratic

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

\[ u_k(t) = -L_k x(t) \]

**CT Bellman eq.**

\[
x^T(t)P_k x(t) = \int_t^{t+T} x^T(\tau) \left( Q + L_k^T R L_k \right) x(\tau) d\tau + x^T(t+T)P_k x(t+T)
\]

is equivalent to

\[
A_k^T P_k + P_k A_k + L_k^T R L_k + Q = 0
\]

\[
A_k = A - B L_k
\]

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

Converges to solution to ARE

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

**Theorem - D. Vrabie**

This algorithm converges and is equivalent to Kleinman's Algorithm

This is a data-based approach that uses measurements of x(t), u(t)
Instead of the plant dynamical model.
Another View- Bellman Optimality Equation Is a Fixed Point Equation

\[ V^*(x(t)) = \min_{u(t)} \left\{ \int_{t}^{t+\Delta t} r(x(\tau), u(\tau))d\tau + V^*(x(t+\Delta t)) \right\} \]

or

\[ 0 = \min_{u(t)} \left\{ \int_{t}^{t+\Delta t} r(x(\tau), u(\tau))d\tau + V^*(x(t+\Delta t)) - V^*(x(t)) \right\} \]

Policy must be stabilizing to solve this eq.

Define Contraction map

Bellman Eq.

\[ 0 = \int_{t}^{t+\Delta t} r(x(\tau), u_k(\tau))d\tau + V_k(x(t+\Delta t)) - V_k(x(t)) \]

\[ u_{k+1}(x(t)) = \arg\min_{u(\tau)} \left\{ \int_{t}^{t+\Delta t} r(x(\tau), u(\tau))d\tau + V_k(x(t+\Delta t)) - V_k(x(t)) \right\} \]

Recall

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

\[
x^T(t)P_k x(t) = \int_{t}^{t+T} x^T(\tau)(Q+L_k^T RL_k)x(\tau) \, d\tau + x^T(t+T)P_k x(t+T)
\]

\[
x^T(t)P_k x(t) - x^T(t+T)P_k x(t+T) = \int_{t}^{t+T} x^T(\tau)(Q+L_k^T RL_k)x(\tau) \, d\tau
\]

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

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

\[
= \overline{p}_k \left[ \overline{x}(t) - \overline{x}(t+T) \right]
\]

Quadratic basis set
Algorithm Implementation

Critic update

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

Use Kronecker product

\[ \text{vec}(ABC) = \left(C^T \otimes A\right) \text{vec}(B) \]

To set this up as

\[ \bar{x}(t) = x(t) \otimes x(t) \quad \text{is the quadratic basis set} \]

\[ \bar{p}_k^T \bar{x}(t) = \int_t^{t+T} x(\tau)^T (Q + L_k^T R L_k) x(\tau) d\tau + \bar{p}_k^T \bar{x}(t+T) \]

\( \text{c.f. Linear in the parameters system ID} \)

\[ \bar{p}_k^T \phi(t) = \bar{p}_k^T \left[ \bar{x}(t) - \bar{x}(t+T) \right] = \int_t^{t+T} x(\tau)^T (Q + L_k^T R L_k) x(\tau) d\tau \]

\[ \equiv \rho(t, t+T) \quad \text{Reinforcement on time interval } [t, t+T] \]

Quadratic regression vector

Regression matrix

Same form as standard System ID problems

Solve using RLS or batch LS

Need \( n(n+1)/2 \) data points along the system trajectory

Unknown parameters
Nonlinear Case- Approximate Dynamic Programming

Value Function Approximation (VFA) to Solve Bellman Equation
– Paul Werbos (ADP), Dmitri 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 \([t, t+T] \)

Now use RLS along the trajectory to get new weights \( W_{k+1} \)

Then find updated FB

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

Direct Optimal Adaptive Control for Partially Unknown CT Systems
Integral Reinforcement Learning (IRL)

1. Select initial control policy

2. Find associated cost
\[
\overline{p}_k^T [\overline{x}(t) - \overline{x}(t+T)] = \int_t^{t+T} x(\tau)^T (Q + L_k^T R L_k) x(\tau) d\tau = \rho(t, t+T)
\]

3. Improve control
\[
L_{k+1} = R^{-1} B^T P_k
\]

Bellman Equation
Solves Lyapunov eq. without knowing dynamics

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

Data set at time \([t, t+T)\)
\[
\left( x(t), \rho(t, t+T), x(t+T) \right)
\]

A is not needed anywhere

observe \(x(t)\)
apply \(u^k(t) = L_k x(t)\)
observe cost integral
observe \(x(t+T)\)
update P
update control gain to \(L_{k+1}\)
do RLS until convergence to \(P_k\)
Batch LS Algorithm Implementation

Or use Recursive Least-Squares solution along the trajectory

The Critic update

\[ x^T(t)P_k x(t) = \left( \int_{t}^{t+T} x^T(\tau)(Q+L_k^T R L_k)x(\tau) d\tau + x^T(t+T)P_k x(t+T) \right) \]

can be setup as

\[ \bar{p}_k^T \varphi(t) = \bar{p}_k^T [\bar{x}(t) - \bar{x}(t+T)] = \int_{t}^{t+T} x(\tau)^T (Q+L_k^T R L_k)x(\tau) d\tau \equiv d(\bar{x}(t), L_k) \]

\[ \bar{x}(t) = x(t) \otimes x(t) \text{ is the quadratic basis set} \]

Evaluating \(d(\bar{x}(t), L_k)\) for \(N=n(n+1)/2\) trajectory points, one can setup a least squares problem to solve

\[ \bar{p}_k = (X X^T)^{-1} X Y \]

\[ X = [\varphi(t) \varphi(t+T) \ldots \varphi(t+NT)] \]

\[ Y = [d(\bar{x}(t), K_i) \ d(\bar{x}(t+T), K_i) \ldots \ d(\bar{x}(t+NT), K_i)]^T \]
Persistence of Excitation

\[ p_k^T \phi(t) \equiv p_k^T [\bar{x}(t) - x(t+T)] = \int_t^{t+T} x(\tau)^T (Q + L_k^T R L_k) x(\tau) d\tau \]

Regression vector must be PE

Relates to choice of reinforcement interval T
Implementation

Policy evaluation
Need to solve online

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

Add a new state = Integral Reinforcement

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

This is the controller dynamics or memory
Draguna Vrabie

Direct Optimal Adaptive Controller

Solves Riccati Equation Online without knowing A matrix

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

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

Dynamic Control System w/ MEMORY

Update FB gain after Critic has converged
Continuous-time control with discrete gain updates

Gain update (Policy)

\[ L_k \]

Control

\[ u_k(t) = -L_k x(t) \]

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

Continuous-time control with discrete gain updates
Optimal Control Design Allows a Lot of Design Freedom

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

Tailor controls design by choosing utility \( r(x,u) \)
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

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

State Evolution for both controllers

Encode into Value Function

Murad Abu-Khalaf
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^TP + Q - PBR^{-1}B^TP

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

= \begin{bmatrix} 1.4245 & 1.1682 & -0.1352 & 1.4349 & -0.1501 & 0.4329 \end{bmatrix}^T

Stevens and Lewis 2003

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

A matrix not needed

Converge to SS Riccati equation soln

Solves ARE online without knowing A

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

Take sample points along a single trajectory
Recursive Least-Squares RLS

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
Sun Tzu

500 BC

Sun Tzu's
ART OF WAR

孙子兵法

Sun Tz bin fa
2. H-Infinity Control Using Neural Networks

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

\[
\begin{align*}
x &\quad \rightarrow \quad z \\
z &\quad \rightarrow \quad y \\
y &\quad \rightarrow \quad d \\
d &\quad \rightarrow \quad u \\
u &\quad \rightarrow \quad \text{control}
\end{align*}
\]

L₂ Gain Problem

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

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

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

Zero-Sum differential game

Nature as the opposing player
2. Online Zero-Sum Differential Games

H-infinity Control

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

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

Differential equivalent is ZS game Bellman equation
\[ 0 = r(x,u,d) + \dot{V} = r(x,u,d) + (\nabla V)^T (f(x) + g(x)u + k(x)d) \equiv H(x, \frac{\partial V}{\partial x},u,d) \]
\[ V(0) = 0 \]

Given any stabilizing control and disturbance policies \( u(x), d(x) \)

the cost value is found by solving this nonlinear Lyapunov equation
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

\[ 0 = \frac{\partial H}{\partial u}, \quad 0 = \frac{\partial H}{\partial d} \]
Game saddle point solution found from Hamiltonian - 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

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

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

HJI equation

\[ 0 = H(x, \nabla V, u^*, d^*) \]

\[ = h^T h + \nabla V^T(x) f(x) - \frac{1}{4} \nabla V^T(x) g(x) R^{-1} g^T(x) \nabla V(x) + \frac{1}{4\gamma^2} \nabla V^T(x) k k^T \nabla V(x) \]

\[ V(0) = 0 \]

(‘Nonlinear Game Riccati’ equation)
Linear Quadratic Zero-Sum Games

\[
\dot{x} = Ax + B_1 u_1 + B_2 u_2
\]
\[
y = C x
\]

\[-J_2(x(t), u_1, u_2) = J_1(x(t), u_1, u_2) = \frac{1}{2} \int_t^\infty (x^T Q x + u_1^T R_{11} u_1 - u_2^T R_{12} u_2) \, d\tau, \quad Q = C^T C\]

Game Algebraic Riccati Equation

\[0 = A^T P + PA + Q - PB_1 R_{11}^{-1} B_1^T P + PB_2 R_{12}^{-1} B_2^T P\]

\[u_1 = -K_1 x \equiv -R_{11}^{-1} B_1^T P x, \quad u_2 = K_2 x \equiv R_{12}^{-1} B_2^T P x\]
Policy Iteration Algorithm to Solve HJI

Start with stabilizing initial control policy $u_0(x)$

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

$$0 = h^T h + \nabla V_{j+1}^T(x) \left( \dot{f}(x) + g(x)u_j(x) \right) + u_j^T(x)Ru_j(x) + \frac{1}{4\gamma^2} \nabla V_{j+1}^T(x)kk^T \nabla V_{j+1}(x)$$

$V_{j+1}(0)=0$

2. Improve policy:

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

Minimal nnd solution of HJ equation is the Available Storage for $u_j(x)$

Off-line solution
Nonlinear HJ equation must be solved at each step
Double Policy Iteration Algorithm to Solve HJI

Add inner loop to solve for available storage
Start with stabilizing initial policy $u_0(x)$

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 $V_j^i(x(t)), d^{i+1}$

   \[
   0 = h^T h + \nabla V_j^i(x)(f + gu_j + kd^i) + u_j^T Ru_j - \gamma^2 \left\| d^i \right\|^2
   \]

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

   On convergence set $V_{j+1}(x) = V_j^i(x)$

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

Off-line solution
Nonlinear Lyapunov equation must be solved at each step
Online Learning for Two-Player Zero-Sum Games

Online game solutions without knowing $A$ matrix

- **System dynamics**
  \[
  \dot{x} = Ax + B_1 w + B_2 u, \quad x \in \mathbb{R}^n, \; u \in \mathbb{R}^m, \; w \in \mathbb{R}^q
  \]

- **Cost function**
  \[
  V(x_0, u, w) = \int_0^\infty (x^T C^T C x + u^T u - w^T w) dt
  \]

- **Goal: saddle point**
  \[
  V(x_0, \tilde{u}, w^*) \geq V(x_0, u^*, w^*) \geq V(x_0, u^*, \tilde{w})
  \]

- **State-feedback stabilizing solution**
  \[
  u^* = -B_2^T P x, \quad w^* = B_1^T P x, \quad V(x_0, u^*, w^*) = x_0^T P x_0
  \]
  \[
  0 = A^T P + PA + C^T C - P(B_2 B_2^T - B_1 B_1^T) P
  \]
Online Policy Iteration for 2-player ZS games

Options:
1. Both players learn online (two critics) to optimize their behavior policies
   a) simultaneously
   b) taking turns – while one is learning the other player maintains a fixed policy
2. Only one player learns online => single critic
   - the other player uses a fixed policy and only updates it at discrete moments based on information on the policy of his opponent

Parameters that define the policies of the players
Online Nash equilibrium Learning

The game is played as follows:
1. The game starts while Player 2 (the disturbance) does not play.
2. Player 1
   a. plays the game without opponent and
   b. uses reinforcement learning to find the optimal behavior which
      minimizes its value;
   c. then informs Player 2 on his new optimized value fn.
3. Player 2 starts playing using the value fn. of his opponent.
4. Player 1
   a. corrects iteratively his own behavior using reinforcement learning such
      that its value is again minimized;
   b. then informs Player 2 on his new optimized value fn.
5. Go to step 3 until the two policies are characterized by the
   same parameter values.
Policy Iteration for Online Zero-Sum Games

The game is played as follows:

1. \( i = 1; \quad P_{u}^{i-1} = P_{u}^{0} = 0; \quad w_{1} = B_{1}^{T} P_{u}^{0} x = 0 \)

2. Player 1 solves online, using HDP, the Riccati equation

\[
P_{u}^{1} A + A^{T} P_{u}^{1} - P_{u}^{1} B_{2} B_{2}^{T} P_{u}^{1} + C^{T} C = 0
\]

\[
u_{1} = -B_{2}^{T} P_{u}^{1} x
\]

then informs Player 2 on \( P_{u}^{1} \)

3. Player 2 uses the value \( P_{u}^{i} \) of Player 1. Computes his policy \( w_{i} = B_{1}^{T} P_{u}^{i} x \)

4. Player 1 solves online, using HDP, the Riccati equation;

\[
Z_{u}^{i} A_{u}^{i-1} + A_{u}^{i-1T} Z_{u}^{i} - Z_{u}^{i} B_{2} B_{2}^{T} Z_{u}^{i} + Z_{u}^{i-1} B_{1} B_{1}^{T} Z_{u}^{i-1} = 0
\]

\[
P_{u}^{i} = Z_{u}^{i} + P_{u}^{i-1}
\]

\[
u_{i} = -B_{2}^{T} P_{u}^{i} x
\]

then informs Player 2 on \( P_{u}^{i} \)

5. Set \( i = i + 1 \). Go to step 3 until the two policies are characterized by the same parameter values.

Riccati equations can be solved using HDP without knowledge of the A matrix

Convergence proven by Lanzon, Feng, Brian Anderson 2009

Draguna Vrabie
Integral Reinforcement Learning (IRL) to solve ARE - Draguna Vrabie

CT Bellman eq.

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

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

Solves Lyapunov equation without knowing A or B

Only B is needed

Converges to solution to ARE

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

This is a data-based approach that uses measurements of x(t), u(t) Instead of the plant dynamical model.
Actor-Critic structure - three time scales

\[ P_u^{i(k)} = P_u^{i-1} + Z_u^{i(k)} \]

\[ P_u^{i(k-1)} = P_u^{i-1} + Z_u^{i(k-1)} \]

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

Controller/Player 1:
\[ u = -B_T^i P_u^{i(k-1)} x \]

Critic Learning procedure:
\[ \dot{V} = x^T C^T C x + \hat{u}^T \hat{u}, \text{ if } i = 1 \]
\[ \dot{V} = \hat{w}^T \hat{w} + \hat{u}^T \hat{u}, \text{ if } i > 1 \]

Disturbance/Player 2:
\[ w = B_1^T P_w^{i-1} x \]
Comparison of CT IRL ADP to Discrete-Time ADP

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

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

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

a. Use discrete-time ADP

\[ A = \begin{bmatrix}
-0.0665 & 8 & 0 & 0 \\
0 & -3.663 & 3.663 & 0 \\
-6.86 & 0 & -13.736 & -13.736 \\
0.6 & 0 & 0 & 0 \\
\end{bmatrix}, \quad B = [0 \ 0 \ 13.7355 \ 0]. \]

sampling period of T = 0.01s.

\[ P_{\text{critic NN}} = \begin{bmatrix}
0.4802 & 0.4768 & 0.0603 & 0.4754 \\
0.4768 & 0.7887 & 0.1239 & 0.3834 \\
0.0603 & 0.1239 & 0.0567 & 0.0300 \\
0.4754 & 0.3843 & 0.0300 & 2.3433 \\
\end{bmatrix}. \]

\[ A^T PA - P + Q - A^T PB (B^T PB + R)^{-1} B^T PA = 0. \]

\[ P_{\text{DARE}} = \begin{bmatrix}
0.4750 & 0.4766 & 0.0601 & 0.4751 \\
0.4766 & 0.7831 & 0.1237 & 0.3829 \\
0.0601 & 0.1237 & 0.0513 & 0.0298 \\
0.4751 & 0.3829 & 0.0298 & 2.3370 \\
\end{bmatrix}. \]
b. Use continuous-time IRL ADP

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

Solves ARE online without knowing A

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

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.

Less computation is needed using CT IRL

In DT ADP sampling period is 0.01s and the critic parameter estimates were updated every 0.15s.

Yet, the parameter estimates for the P matrix entries almost overlay each other.
Simulation- H-inf control for Electric Power Plant- LFC

\[ \dot{x} = Ax + B_2u + B_1d \]

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

\[ A = \begin{bmatrix} -1/T_p & K_p/T_p & 0 & 0 \\ 0 & -1/T_r & 1/T_r & 0 \\ -1/RT_G & 0 & -1/T_G & -1/T_G \\ K_e & 0 & 0 & 0 \end{bmatrix}, \quad B_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \]

\[ B_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \]

\[ A = \begin{bmatrix} -0.0665 & 8 & 0 & 0 \\ 0 & -3.663 & 3.663 & 0 \\ -6.86 & 0 & -13.736 & -13.736 \\ 0.6 & 0 & 0 & 0 \end{bmatrix}, \quad B = [0 \quad 0 \quad 13.7355 \quad 0]^T, \quad B_1 = [-8 \quad 0 \quad 0 \quad 0]^T \]

\[ 0 = A^T P + PA + C^T C - P(B_2B_2^T - B_1B_1^T)P \]

A is unknown
B_1, B_2 are known
Simulation result – Electric Power Plant LFC

- System – Power plant - internally stable system;
  - system state \( x = [\Delta f(t) \ \Delta P_g(t) \ \Delta X_g(t) \ \Delta E(t)] \)
    (incremental changes of: frequency deviation, generator output, governor position and integral control)
  - Player 1 - controller; Player 2 – load disturbance

- Nash equilibrium solution
  \[
P_u^\infty = \Pi = \begin{bmatrix} 0.6036 & 0.7398 & 0.0609 & 0.5877 \\ 0.7398 & 1.5438 & 0.1702 & 0.5978 \\ 0.0609 & 0.1702 & 0.0502 & 0.0357 \\ 0.5877 & 0.5978 & 0.0357 & 2.3307 \end{bmatrix}
  \]

- Online learned solution using ADP – after 5 updates of the parameters of Player 2
  \[
P_u^5 = \begin{bmatrix} 0.6036 & 0.7399 & 0.0609 & 0.5877 \\ 0.7399 & 1.5440 & 0.1702 & 0.5979 \\ 0.0609 & 0.1702 & 0.0502 & 0.0357 \\ 0.5877 & 0.5979 & 0.0357 & 2.3307 \end{bmatrix}
  \]

Solves GARE online without knowing \( A \)
\[
0 = A^T P + PA + C^T C - P(B_2 B_2^T - B_1 B_1^T)P
\]
Parameters of the cost function of the game

- Cost function learning using least squares
- Sampling integration time $T=0.1\, s$
- The policy of Player 1 is updated every 2.5\, s
- The policy of Player 2 is updated only when the policy of Player 1 has converged
- Number of updates of Player 1 before an update of Player 2

moments when Player 2 is updated
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

Limbic System

Hippocampus

Cerebellum

Brainstem

Spinal cord

Exteroceptive receptors

Interoceptive receptors

Muscle contraction and movement

Long term
Memory functions

Short term

Reinforcement learning - dopamine

Supervised learning

Unsupervised learning

Gamma rhythms 30-100 Hz

Theta rhythms 4-10 Hz

Motor control 200 Hz

Hierarchy of multiple parallel loops

picture by E. Stingu
D. Vrabie
Adaptive Critic structure

Reinforcement learning

Theta waves 4-8 Hz

Motor control 200 Hz
Cerebral cortex
Motor areas

Basal ganglia

Thalamus

Hippocampus

Cerebellum

Brainstem

Spinal cord

Inf. olive

Exteroceptive receptors

Interoceptive receptors

Muscle contraction and movement

**gamma rhythms** 30-100 Hz

**theta rhythms** 4-10 Hz

Intense processing
due to the amounts of
information data to be processed
Cognitive map of the environment
- place cells -

**theta rhythms** 4-10 Hz

Behavior reference
Information sent to the
lower processing levels

Motor control 200 Hz
2. Synchronous
Online Solution of Optimal Control for Nonlinear Systems

Optimal Adaptive Control

Policy Iteration gives the structure needed for online optimal solution

A new structure of adaptive controllers
CT Policy Iteration – a Reinforcement Learning Technique

To avoid solving HJB equation

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

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

Cost for any given admissible \( u(x) \)

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

Policy Iteration Solution

Pick stabilizing initial control policy

**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

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

- 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
Synchronous
Online Solution of Optimal Control for Nonlinear Systems

Optimal Adaptive Control

Policy Iteration gives the structure needed for online optimal solution

Need to solve online:

Bellman eq. for Value

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

Control update

\[ h(x) = -\frac{1}{2} R^{-1} g^T(x) \nabla V \]
Solve by parameterizing value $V(x)$

*Value Function Approximation* – *Paul Werbos*

converts Bellman PDE into algebraic equation

**Critic NN**

Take VFA as $V(x) = W_1^T \phi_1(x) + \varepsilon(x)$, \quad \nabla V(x) = \nabla \phi_1^T W_1$

Then Bellman eq

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

becomes

$$H(x, W_1, u) = W_1^T \nabla \phi_1 (f + gu) + Q(x) + u^T Ru = \varepsilon_H$$

$W_1 =$ LS solution to this eq for given $N$. Unknown.

**Action NN for Control Approximation**

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

Comes from $u(x) = -\frac{1}{2} R^{-1} g^T (x) \nabla V$

$$\nabla V(x) = \nabla \phi_1^T W_1$$
Online Synchronous Policy Iteration

Theorem (Kyriakos Vamvoudakis)- Online Learning of Nonlinear Optimal Control

Let $\sigma_1 \equiv \nabla \phi_1(f + gu)$ be PE. Tune critic NN weights as

$$\dot{\hat{W}}_1 = -a_1 \frac{\partial E_1}{\partial \hat{W}_1} = -a_1 \frac{\sigma_1}{(\sigma_1^T \sigma_1 + 1)^2} [\sigma_1^T \hat{W}_1 + Q(x) + u^T Ru]$$

Learning the Value

Tune actor NN weights as

$$\dot{\hat{W}}_2 = -\alpha_2 \{(F_2 \hat{W}_2 - F_1 \sigma_1^T \hat{W}_1) - \frac{1}{4} \overline{D}_1(x)\hat{W}_2 m^T(x)\hat{W}_1\}$$

Learning the control policy

where $\overline{D}_1(x) \equiv \nabla \phi_1(x)g(x)R^{-1}g^T(x)\nabla \phi_1^T(x)$, $m \equiv \frac{\sigma_1}{(\sigma_1^T \sigma_1 + 1)^2}$

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 $\tilde{W}_2 = W_1 - \hat{W}_1$ are UUB bounded.
Summary Nota Bene

Control policy

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

Tune critic NN weights as

\[ \dot{\hat{W}}_1 = -a_1 \frac{\partial E_1}{\partial \hat{W}_1} = -a_1 \frac{\sigma_1}{(\sigma_1^T \sigma_1 + 1)^2} [\sigma_1^T \hat{W}_1 + Q(x) + u^T Ru] \]

Tune actor NN weights as

\[ \dot{\hat{W}}_2 = -\alpha_2 \{(F_2 \hat{W}_2 - F_1 \sigma_1^T \hat{W}_1) - \frac{1}{4} D_1(x)\hat{W}_2 m^T (x) \hat{W}_1\} \]

Note, it does not work to simply set

\[ u(x) = -\frac{1}{2} R^{-1} g^T (x) \nabla \phi_1^T \hat{W}_1 \]

Must have TWO NNs
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

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

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

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

\[ H(x, W_1, u) = W_1^T \nabla \phi_1(f + gu) + Q(x) + u^T Ru = \varepsilon_H \]
ONLINE solution
Does not require solution of HJB or nonlinear Lyapunov eq.

Does require system dynamics to be known

Finds approximate local smooth solution to NONLINEAR HJB equation online

An optimal adaptive controller
‘indirect’ because it identifies parameters for VFA
‘direct’ because control is directly found from value function
A New Adaptive Control Structure with Multiple Tuned Loops

Adaptive Critics

The Adaptive Critic Architecture

Value update - solve Bellman eq.

\[ V(x) = W_1^T \phi_1(x) \]

Cost

Policy Evaluation (Critic network)

Control policy update

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

Action network

System

Critic and Actor tuned \textit{simultaneously}

Leads to ONLINE FORWARD-IN-TIME implementation of optimal control

Optimal Adaptive Control
Adaptive Control

Identify the performance value-Optimal Adaptive

Identify the system model-Indirect Adaptive

Identify the Controller-Direct Adaptive

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

\[
W_1^* = \begin{bmatrix}
p_{11} & 2p_{12} & 2p_{13} & p_{22} & 2p_{23} & p_{33}
\end{bmatrix}^T
\]

\[
= \begin{bmatrix}
1.4245 & 1.1682 & -0.1352 & 1.4349 & -0.1501 & 0.4329
\end{bmatrix}^T
\]

Solves ARE online

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

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

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

\[
\hat{W}_2(t_f) = \begin{bmatrix}
1.4279 & 1.1612 & -0.1366 & 1.4462 & -0.1480 & 0.4317
\end{bmatrix}^T.
\]

\[
\hat{u}_2(x) = -\frac{1}{2}R^{-1}B^T P x = \begin{bmatrix}
2x_1 & 0 & 0 \\
x_2 & x_1 & 0 \\
x_3 & 0 & x_1 \\
0 & 2x_2 & 0 \\
0 & x_3 & x_2 \\
0 & 0 & 2x_3
\end{bmatrix}^T \begin{bmatrix}
1.4279 \\
1.1612 \\
-0.1366 \\
1.4462 \\
-0.1480 \\
0.4317
\end{bmatrix}.
\]
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 \[ \phi_1(x) = [x_1^2 \quad x_1x_2 \quad x_2^2]^T, \]

Algorithm converges to
\[ \hat{W}_1(t_f) = [0.5017 \quad -0.0020 \quad 1.0008]^T. \]

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

\[ \hat{u}_2(x) = -\frac{1}{2} R^{-1} \begin{bmatrix} 0 \\ \cos(2x_1) + 2 \end{bmatrix} \begin{bmatrix} 2x_1 & 0 \\ x_2 & x_1 \\ 0 & 2x_2 \end{bmatrix}^T \begin{bmatrix} 0.5017 \\ -0.0020 \\ 1.0008 \end{bmatrix} \]
Online Synchronous Policy Iteration using IRL

Does not need to know \( f(x) \)

Replace \( \sigma_1 \equiv \nabla \phi_1(f + gu) \) by \( \Delta \phi(x(t)) \equiv \phi(x(t)) - \phi(x(t - T)) \)

**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

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

Learning the Value

Tune actor NN weights as

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

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.
Can avoid knowledge of drift term $f(x)$ by using Integral Reinforcement Learning (IRL)

Draguna Vrabie
Double Policy Iteration Algorithm to Solve HJI

Add inner loop to solve for available storage
Start with stabilizing initial policy \( u_0(x) \)

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

\[
2. \text{Set } d^0 = 0. \text{ For } i=0,1,... \text{ solve for } V^i_j(x(t)), d^{i+1} \\
0 = h^T h + \nabla V^i_j(x)(f + gu_j + kd^i) + u^T_j Ru_j - \gamma^2 \| d^i \|^2 \\
d^{i+1} = \frac{1}{2\gamma^2} k^T(x) \nabla V^i_j \\
\text{On convergence set } V_{j+1}(x) = V^i_j(x) \\
\]

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

Off-line solution
Nonlinear Lyapunov equation must be solved at each step
Online Solution of ZS Games for Nonlinear Systems

Optimal (Game) Adaptive Control

Policy Iteration gives the structure needed for online solution

Need to solve online these 3 equations:

ZS game Bellman eq. for Value

$$0 = h^T h + \nabla V^T (x)(f + gu + kd) + u^T Ru - \gamma^2 \|d\|^2$$

Disturbance update

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

Control update

$$u(x) = -\frac{1}{2} R^{-1} g^T (x) \nabla V$$
Use Three Neural Networks

**Critic NN for VFA**

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

Bellman eq becomes algebraic eq.

\[ H(x, \hat{W}_1, u) = \hat{W}_1^T \nabla \phi_1 (f + gu + kd) + h^T h + u^T Ru - \gamma^2 \| d \|^2 = e_1 \]

**Control Actor NN**

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

Comes from

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

**Disturbance actor NN**

\[ d(x) = \frac{1}{2\gamma^2} k^T (x) \nabla \phi_1^T \hat{W}_3, \]

Comes from

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

Simultaneously:

a. Solve Bellman eq.

and

b. update \( u(x), d(x) \)
Online Synchronous Policy Iteration for ZS games

Theorem (Kyriakos Vamvoudakis)- Online Gaming

Let \( \sigma_2 = \nabla \phi(f + gu + kd) \) be PE. Tune critic NN weights as

\[
\dot{\hat{W}}_1 = -a_1 \frac{\sigma_2}{(\sigma_2 \sigma_2 + 1)^2} \left[ \sigma_2^T \hat{W}_1 + h^T Qh - \gamma^2 \|d\|^2 + u^T Ru \right]
\]

Learning the Value

Tune actor NN weights as

\[
\begin{align*}
\dot{\hat{W}}_2 &= -\alpha_2 \left\{ (F_2 \hat{W}_2 - F_1 \sigma_2^T \hat{W}_1) - \frac{1}{4} \bar{D}_1(x) \hat{W}_2 m^T(x) \hat{W}_1 \right\} \\
\dot{\hat{W}}_3 &= -\alpha_3 \left\{ (F_4 \hat{W}_3 - F_3 \sigma_2^T \hat{W}_1) + \frac{1}{4\gamma^2} \bar{E}_1(x) \hat{W}_3 m^T \hat{W}_1 \right\}
\end{align*}
\]

Learning the control policies

where

\[
\begin{align*}
\bar{D}_1(x) &= \nabla \phi_1(x) g(x) R^{-1} g^T(x) \nabla \phi_1^T(x), \\
\bar{E}_1(x) &= \nabla \phi_1(x) kk^T \nabla \phi_1^T(x),
\end{align*}
\]

\[
m = \frac{\sigma_2}{(\sigma_2^T \sigma_2 + 1)^2}
\]

Then there exists an \( N_0 \) such that, for the number of hidden layer units

the closed-loop system state, the critic NN error \( \hat{W}_1 = W_1 - \hat{W}_1 \)

and the actor NN errors \( \hat{W}_2 = W_1 - \hat{W}_2, \quad \hat{W}_3 = W_1 - \hat{W}_3 \)

are UUB bounded.
Actor-Critic structure –
A New Adaptive Controller with three tuned loops

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

**Critic NN**
Learning procedure

**Controller state-memory**
\[ \dot{V} = h^T h + u^T \frac{R}{T} u - \frac{\gamma^2}{T} \|d\|^2 \]

**Controller/Player 1**
\[ u(x) = -\frac{T}{2} R \cdot g^T (x) \nabla \phi^T \dot{\hat{W}} \]

**Disturbance/Player 2**
\[ d(x) = \frac{T}{2\gamma^2} R^T (x) \nabla \phi^T \dot{\hat{W}}_3 \]

A novel form of Hybrid Controller
ONLINE solution
Does not require solution of HJI eq, HJ eq, or nonlinear Lyapunov eq.

Does require system dynamics to be known

Finds approximate local smooth solution to NONLINEAR HJI equation online

An optimal adaptive controller
   ‘indirect’ because it identifies parameters for VFA
   ‘direct’ because control is directly found from value function
Simulation 1- F-16 aircraft pitch rate controller

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

\[
y = C^T x
\]

\[
Q = C^T C = I, \quad R = I
\]

Solves GARE online

\[
A^T P + PA + Q - PBR^{-1}B^T P + \frac{1}{\gamma^2} PKK^T P = 0
\]

Stevens and Lewis 2003

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

Wind gust

Exact solution

\[
W^*_1 = \begin{bmatrix} p_{11} & 2p_{12} & 2p_{13} & p_{22} & 2p_{23} & p_{33} \end{bmatrix}^T
\]

\[
= [1.6573 \quad 1.3954 \quad -0.1661 \quad 1.6573 \quad -0.1804 \quad 0.4371]^T
\]

Must add probing noise to \(u(x)\) and \(d(x)\) to get PE

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

(exponentially decay \(n(t)\))

Algorithm converges to

\[
\hat{W}_1(t_f) = [1.7090 \quad 1.3303 \quad -0.1629 \quad 1.7354 \quad -0.1730 \quad 0.4468]^T.
\]

\[
\hat{W}_2(t_f) = \hat{W}_3(t_f) = \hat{W}_4(t_f)
\]

\[
\dot{\mathbf{u}}_2(x) = -\frac{1}{2} R^{-1} \begin{bmatrix}
2x_1 & 0 & 0 & 1.7090 \\
0 & x_2 & x_1 & 1.3303 \\
x_3 & 0 & x_1 & -0.1629 \\
0 & 2x_2 & 0 & 1.7354 \\
0 & x_3 & x_2 & -0.1730 \\
0 & 0 & 2x_3 & 0.4468
\end{bmatrix} \mathbf{g}
\]

\[
\dot{\mathbf{d}}(x) = \frac{1}{2\gamma^2} \begin{bmatrix}
2x_1 & 0 & 0 & 1.7090 \\
x_2 & x_1 & 0 & 1.3303 \\
x_3 & 0 & x_1 & -0.1629 \\
0 & 2x_2 & 0 & 1.7354 \\
0 & x_3 & x_2 & -0.1730 \\
0 & 0 & 2x_3 & 0.4468
\end{bmatrix} \mathbf{g}
\]

Wind gust
Critic NN parameters

System states
F-16 aircraft pitch rate controller

Critic NN parameters
With disturbance

Critic NN parameters
Without disturbance

Converges FASTER with an opponent
One learns faster with an adversary
Simulation 3. – Nonlinear System

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

\[
\begin{align*}
    f(x) &= \begin{bmatrix} -x_1 + x_2 \\ -x_1^3 - x_2^3 + 0.25x_2(\cos(2x_1) + 2) - 0.25x_2 \frac{1}{\gamma^2}(\sin(4x_1) + 2)^2 \end{bmatrix} \\
    g(x) &= \begin{bmatrix} 0 \\ \cos(2x_1) + 2 \end{bmatrix}, \quad k(x) = \begin{bmatrix} 0 \\ (\sin(4x_1) + 2) \end{bmatrix}.
\end{align*}
\]

\[ Q = I, \quad R = I, \quad \gamma = 8 \]

Optimal Value

\[ V^*(x) = \frac{1}{4}x_1^4 + \frac{1}{2}x_2^2 \]

Saddle point solution

\[ u^*(x) = -(\cos(2x_1) + 2)x_2, \quad d^*(x) = \frac{1}{\gamma^2}(\sin(4x_1) + 2)x_2 \]

Solves HJI eq. online

\[
0 = h^T h + \nabla V^T(x)f(x) - \frac{1}{4} \nabla V^T(x)g(x)R^{-1}g^T(x)\nabla V(x) + \frac{1}{4\gamma^2} \nabla V^T(x)kk^T\nabla V(x)
\]

Select VFA basis set

\[ \phi_1(x) = [x_1^2 \quad x_2^2 \quad x_1^4 \quad x_2^4] \]

Algorithm converges to

\[ \hat{W}_1(t_f) = [0.0008 \quad 0.4999 \quad 0.2429 \quad 0.0032]^T \]

\[ \hat{W}_2(t_f) = \hat{W}_3(t_f) = \hat{W}_1(t_f) \]

\[
\begin{align*}
    \hat{u}_2(x) &= -\frac{1}{2}R^{-1}\begin{bmatrix} 0 \\ \cos(2x_1) + 2 \\ 4x_1^3 \\ 0 \end{bmatrix}^T \begin{bmatrix} 2x_1 & 0 \\ 0 & 2x_2 \\ 4x_1^3 & 0 \\ 0 & 4x_2^3 \end{bmatrix} \begin{bmatrix} 0.0008 \\ 0.4999 \\ 0.2429 \\ 0.0032 \end{bmatrix} \\
    d(x) &= \frac{1}{2\gamma^2}\begin{bmatrix} 0 \\ \sin(4x_1) + 2 \\ 4x_1^3 \\ 0 \end{bmatrix}^T \begin{bmatrix} 2x_1 & 0 \\ 0 & 2x_2 \\ 4x_1^3 & 0 \\ 0 & 4x_2^3 \end{bmatrix} \begin{bmatrix} 0.0008 \\ 0.4999 \\ 0.2429 \\ 0.0032 \end{bmatrix}
\end{align*}
\]
3. Real-Time Solution of Multi-Player NZS Games

Kyriakos Vamvoudakis

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

Nash equilibrium
\[ V_i^* \equiv V_i (\mu_i^*, \mu_2^*, \ldots, \mu_N^*) \leq V_1 (\mu_1^*, \mu_2^*, \ldots, \mu_N^*), i \in N \]

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

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

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

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

These are hard to solve
In the nonlinear case, HJB generally cannot be solved
Team Interest vs. Self Interest

The objective functions of each player can be written as a team average term plus a conflict of interest term:

\[
J_1 = \frac{1}{3} (J_1 + J_2 + J_3) + \frac{1}{3} (J_1 - J_2) + \frac{1}{3} (J_1 - J_3) \equiv J_{\text{team}} + J_{\text{coi}}^1
\]

\[
J_2 = \frac{1}{3} (J_1 + J_2 + J_3) + \frac{1}{3} (J_2 - J_1) + \frac{1}{3} (J_2 - J_3) \equiv J_{\text{team}} + J_{\text{coi}}^2
\]

\[
J_3 = \frac{1}{3} (J_1 + J_2 + J_3) + \frac{1}{3} (J_3 - J_1) + \frac{1}{3} (J_3 - J_2) \equiv J_{\text{team}} + J_{\text{coi}}^3
\]

For N-players

\[
J_i = \frac{1}{N} \sum_{j=1}^{N} J_j + \frac{1}{N} \sum_{j=1}^{N} (J_i - J_j) \equiv J_{\text{team}} + J_{\text{coi}}^i , \quad i = 1, N
\]

For \textit{N-player zero-sum games}, the first term is zero, i.e. the players have no goals in common.
Real-Time Solution of Multi-Player Games

Non-Zero Sum Games – Synchronous Policy Iteration

Kyriakos Vamvoudakis

Value functions

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

Differential equivalent gives Bellman eqs.

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

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
Policy Iteration gives the structure needed for online solution

Need to solve online:

Coupled Bellman eqs.

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

Control policies

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

Each player needs 2 NN – a Critic and an Actor
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
\[ \hat{V}_1(x) = \hat{W}_1^T \phi_1(x), \]
\[ u_1(x) = -\frac{1}{2} R_{11}^{-1} g_1(x) \nabla \phi_1^T \hat{W}_3, \]

Player 2
\[ \hat{V}_2(x) = \hat{W}_2^T \phi_2(x), \]
\[ u_2(x) = -\frac{1}{2} R_{22}^{-1} g_2(x) \nabla \phi_2^T \hat{W}_4. \]

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

\[ \dot{\hat{W}}_3 = -\alpha_3 \{ (F_2 \hat{W}_3 - F_1 \sigma_3^T \hat{W}_1) - \frac{1}{4} \nabla \phi_3 g(x) R_{11}^{-1} R_{21} R_{11}^{-1} g_1(x) \nabla \phi_3^T \hat{W}_3 m_2^T \hat{W}_2 - \frac{1}{4} D_1(x) \hat{W}_3 m_1^T \hat{W}_1 \} \]

Learns control policy
Lyapunov energy-based Proof:

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

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

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

Guarantees stability

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

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

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

\[ H(x, W_1, u) = W_1^T \nabla \phi_1 (f + gu) + Q(x) + u^T Ru = \varepsilon_H \]
Simulation. – Nonlinear System – 2-player game

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

\[
f(x) = \begin{bmatrix} x_2 \\
-\frac{1}{3}x_1 + \frac{1}{4}x_2(\cos(2x_1) + 2) + \frac{1}{4}x_2(\sin(4x_1^2) + 2) \end{bmatrix}
\]

\[
g(x) = \begin{bmatrix} 0 \\
\cos(2x_1) + 2 \end{bmatrix}, \quad k(x) = \begin{bmatrix} 0 \\
\sin(4x_1) + 2 \end{bmatrix}.
\]

\[ Q_1 = 2Q_2 = 2I, \quad R_{11} = 2R_{22} = 2I, \quad R_{12} = 2R_{21} = 2I \]

Optimal Value

\[ V_1^*(x) = \frac{1}{2}x_1^2 + x_2^2 \]

\[ V_2^*(x) = \frac{1}{4}x_1^2 + \frac{1}{2}x_2^2 \]

Optimal Policies

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

\[ d^*(x) = -(\sin(4x_1^2) + 2)x_2 \]

Solves HJB equations online

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

Select VFA basis set

\[ \varphi_1(x) = \varphi_2(x) = [x_1^2 \; x_1x_2 \; x_2^2] \]

Algorithm converges to

\[ \hat{W}_1(t_f) = [0.5015 \; 0.0007 \; 1.0001]^T = \hat{W}_3(t_f) \]

\[ \hat{W}_2(t_f) = [0.2514 \; 0.0006 \; 0.5001]^T = \hat{W}_4(t_f) \]

\[ \hat{u}(x) = -\frac{1}{2}R_{11}^{-1} \begin{bmatrix} 0 \\
\cos(2x_1) + 2 \end{bmatrix}^T \begin{bmatrix} 2x_1 \\
x_2 \\
0 \; 2x_2 \end{bmatrix} \begin{bmatrix} 0.5015 \\
0.0007 \\
1.0001 \end{bmatrix} \]

\[ \hat{d}(x) = -\frac{1}{2}R_{22}^{-1} \begin{bmatrix} 0 \\
\sin(4x_1^2) + 2 \end{bmatrix}^T \begin{bmatrix} 2x_1 \\
x_2 \\
0 \; 2x_2 \end{bmatrix} \begin{bmatrix} 0.2514 \\
0.0006 \\
0.5001 \end{bmatrix} \]
Critic 1 NN parameters

Critic 2 NN parameters

Evolution of the States

3D approximation error value for player 1.

3D approximation error of control for player 1.
Online Synchronous Policy Iteration using IRL

Does not need to know \( f(x) \)

Replace \( \sigma_1 \equiv \nabla \phi_1(f + gu) \) by \( \Delta \phi(x(t)) \equiv \phi(x(t)) - \phi(x(t - T)) \)

**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

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

Learning the Value

Tune actor NN weights as

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

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.
Graphical Coalitional Games

500 BC

Sun Tzu

Sun Tzu's

The Art

of War

孙子兵法

Sun Tzu bin fa
Games on Communication Graphs

500 BC

Sun Tz bin fa
Games on Communication Graphs

http://ARRI.uta.edu/acs
Books Coming


Key Point

Lyapunov Functions and Performance Indices
Must depend on graph topology

Hongwei Zhang, F.L. Lewis, and Abhijit Das
“Optimal design for synchronization of cooperative systems: state feedback, observer and output feedback,”
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)
\[ \delta_i = \sum_{j \in N_i} e_{ij} (x_i - x_j) + g_i (x_i - x_0), \quad \text{Pinning gains } g_i \geq 0 \quad (\text{Ron Chen}) \]

K. Vamvoudakis and F.L. Lewis, “Graphical Games for Synchronization”
Automatica, to appear.
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
\[ x_i(t) \rightarrow x_0(t), \forall i \]

Local neighborhood tracking error (Lihua Xie)
\[ \delta_i = \sum_{j \in N_i} e_{ij} (x_i - x_j) + g_i (x_i - x_0), \quad \text{Pinning gains } g_i \geq 0 \quad \text{(Ron Chen)} \]

Standard way =

Global neighborhood tracking error
\[ \delta = \begin{bmatrix} \delta_1^T \\ \delta_2^T \\ \vdots \\ \delta_N^T \end{bmatrix} \in \mathbb{R}^{nN}, \quad x = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_N^T \end{bmatrix} \in \mathbb{R}^{nN}, \quad x_0 = I x_0 \in \mathbb{R}^{nN} \]
\[ \delta = \left( (L + G) \otimes I_n \right) (x - x_0) = \left( (L + G) \otimes I_n \right) \zeta, \quad \zeta = (x - x_0) \in \mathbb{R}^{nN} \]

Lemma. Let graph be strongly connected and at least one pinning gain nonzero. Then
\[ \| \zeta \| \leq \| \delta \| / \sigma (L + G) \]
and agents synchronize iff \[ \delta(t) \rightarrow 0 \]
Graphical Game: Games on Graphs

Local nbhd. tracking error dynamics

\[ \dot{\delta}_i = \sum_{j \in N_i} e_{ij} (\dot{x}_i - \dot{x}_j) + g_i (\dot{x}_i - \dot{x}_0) \]

\[ \delta_i = A \delta_i + (d_i + g_i) B_i u_i - \sum_{j \in N_i} e_{ij} B_j u_j \]

Define Local nbhd. performance index

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

\[ u_{-i}(t) = \{ u_j : j \in N_i \} \]

Local value functions for fixed policies \( u_i \)

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

Static Graphical Game

\( (G, U, v) \)

\[ G = (V, E), \quad v = [v_1 \ldots v_N]^T \]

\[ v_i(U_i, \{U_j : j \in N_i\}) \in R \]

Standard N-player differential game

\[ \dot{z} = Az + \sum_{i=1}^N B_i u_i \]

\[ J_i(z(0), u_i, u_{-i}) = \frac{1}{2} \int_0^\infty (z^T Qz + \sum_{j=1}^N u_j^T R_{ij} u_j) dt \]

Kyriakos Vamvoudakis
**Graphical Game: Games on Graphs**

Kyriakos Vamvoudakis

Local nbhd. tracking error dynamics

$$\dot{\delta}_i = \sum_{j \in N_i} e_{ij}(\dot{x}_i - \dot{x}_j) + g_i(\dot{x}_i - \dot{x}_0)$$

$$\dot{\delta}_i = A\delta_i + (d_i + g_i)B_i u_i - \sum_{j \in N_i} e_{ij} B_j u_j$$

Define Local nbhd. performance index

$$J_i(\delta_i(0), u_i, u_{-i}) = \frac{1}{2} \int_0^\infty (\delta^T Q_ii \delta_i + u_i^T R_i u_i + \sum_{j \in N_i} u_j^T R_j u_j) \, dt \equiv \frac{1}{2} \int_0^\infty L_i(\delta_i(t), u_i(t), u_{-i}(t)) \, dt$$

$$u_{-i}(t) = \{u_j : j \in N_i\}$$

Values driven by neighbors’ controls

Local value functions for fixed policies $u_i$

$$V_i(\delta_i(t)) = \frac{1}{2} \int_t^\infty (\delta^T Q_ii \delta_i + u_i^T R_i u_i + \sum_{j \in N_i} u_j^T R_j u_j) \, dt$$

Static Graphical Game

$$(G, U, v)$$

$$G = (V, E), \quad v = [v_1 \cdots v_N]^T$$

$$v_i(U_i, \{U_j : j \in N_i\}) \in R$$

Value depends only on neighbors
New Differential Graphical Game

State dynamics of agent $i$

$$
\dot{\delta}_i = A\delta_i + (d_i + g_i)B_iu_i - \sum_{j \in N_i} e_{ij}B_ju_j
$$

Value function of player $i$

$$
J_i(\delta_i(0), u_i, u_{-i}) = \frac{1}{2} \int_0^\infty (\delta_i^TQ_{ii}\delta_i + u_i^TR_{ii}u_i + \sum_{j \in N_i} u_j^TR_{ij}u_j) \, dt
$$

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

Central Dynamics

\[ \dot{z} = Az + \sum_{i=1}^{N} B_i u_i \]

Local Value Function

\[ J_i(z(0), u_i, u_{-i}) = \frac{1}{2} \int_{0}^{\infty} (z^T Q z + \sum_{j=1}^{N} u_j^T R_{ij} u_j) \, dt \]

Value function of player \( i \)

Control action of player \( i \)

depends on ALL other control actions
Team Interest vs. Self Interest

Cooperation vs. Collaboration

The objective functions of each player can be written as a team average term plus a conflict of interest term:

\[ J_1 = \frac{1}{3} (J_1 + J_2 + J_3) + \frac{1}{3} (J_1 - J_2) + \frac{1}{3} (J_1 - J_3) \equiv J_{\text{team}} + J_{1}^{\text{coi}} \]

\[ J_2 = \frac{1}{3} (J_1 + J_2 + J_3) + \frac{1}{3} (J_2 - J_1) + \frac{1}{3} (J_2 - J_3) \equiv J_{\text{team}} + J_{2}^{\text{coi}} \]

\[ J_3 = \frac{1}{3} (J_1 + J_2 + J_3) + \frac{1}{3} (J_3 - J_1) + \frac{1}{3} (J_3 - J_2) \equiv J_{\text{team}} + J_{3}^{\text{coi}} \]

For N-players

\[ J_i = \frac{1}{N} \sum_{j=1}^{N} J_j + \frac{1}{N} \sum_{j=1}^{N} (J_i - J_j) \equiv J_{\text{team}} + J_{i}^{\text{coi}}, \quad i = 1, N \]

For N-player zero-sum games, the first term is zero, i.e. the players have no goals in common.
Problems with Nash Equilibrium Definition on Graphical Games

Game objective

\[ V_i^*(\delta_i(t)) = \min_{u_i} \int_0^\infty \left( \frac{1}{2}(\delta_i^T Q_i \delta_i + u_i^T R_i u_i + \sum_{j \in N_i} u_j^T R_j u_j) \right) dt \]

Define \( u_{-i}(t) \) \( \{u_j : j \in N_i\} \) Neighbors of node \( i \)

\[ u_{G-i} = \{u_j : j \in N, j \neq i\} \]

All other nodes in graph

Def: Nash equilibrium

\( \{u_1^*, u_2^*, \ldots, u_N^*\} \) are in Nash equilibrium if

\[ J_i^* \equiv J_i(u_i^*, u_{G-i}^*) \leq J_i(u_i, u_{G-i}^*), \quad \forall i \in N \]

Counterexample. Disconnected graph

Then, each agent’s cost does not depend on any other agent

\[ J_i(u_i) = J_i(u_i, u_{G-i}) = J_i(u_i, u_{G-i}^*), \quad \forall i \]

Let each node play his optimal control

\[ J_i^* = J_i(u_i^*) \]

Then all agents are in Nash equilibrium

Note- this Nash is also coalition-proof
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

\[
J_i (u_i^*, u_{-i}) \leq J_i (u_i, u_{-i}), \quad \forall u_i
\]

Def: Interactive Nash equilibrium

\( \{u_1^*, u_2^*, \ldots, u_N^*\} \) are in Interactive Nash equilibrium if

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

2. There exists a policy \( u_j \) such that

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

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.
Theorem 3. Let \((A,B)\) be reachable for all \(i\).
Let agent \(i\) be in local best response
\[ J_i (u_i^*,u_{-i}) \leq J_i (u_i,u_{-i}), \quad \forall i \]
Then \(\{u_1^*,u_2^*,...,u_N^*\}\) are in global Interactive Nash iff the graph is strongly connected.

\[ u_i = u_i(V_i) \equiv -(d_i + g_i)R_{ii}^{-1}B_i^T \frac{\partial V_i}{\partial \delta_i} \equiv -K_i p_i \]

\[ u_k = -K_k p_k - v_k \]

Hamiltonian System

\[
\begin{bmatrix}
\dot{\delta} \\
\dot{p}
\end{bmatrix} = 
\begin{bmatrix}
(I_N \otimes A) & ((L + G) \otimes I_n) \text{diag}(B_i K_i) \\
- \text{diag}(Q_{ii}) & -(I_N \otimes A^T)
\end{bmatrix}
\begin{bmatrix}
\delta \\
p
\end{bmatrix} + 
\begin{bmatrix}
((L + G) \otimes I_n)B_k \\
0
\end{bmatrix}
\begin{bmatrix}
v_k \\
p
\end{bmatrix} 
= A \begin{bmatrix}
\delta \\
p
\end{bmatrix} + Bv_k
\]

\[
\begin{bmatrix}
\bar{B} & \bar{AB} & \bar{A^2B} & \cdots
\end{bmatrix}
\]

Picks out the shortest path from node \(k\) to node \(i\)
Graphical Game Solution Equations

Value function

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

Differential equivalent (Leibniz formula) is Bellman’s Equation

\[ H_i(\delta_i, \frac{\partial V_i}{\partial \delta_i}, u_i, u_{-i}) = \frac{\partial V_i}{\partial \delta_i} \left( A \delta_i + (d_i + g_i) B_i u_i - \sum_{j \in N_i} e_{ij} B_j u_j \right) + \frac{1}{2} \delta_i^T Q_{ii} \delta_i + \frac{1}{2} u_i^T R_{ii} u_i + \frac{1}{2} \sum_{j \in N_i} u_j^T R_{ij} u_j = 0 \]

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

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

where

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

2. Best Response HJ Equations – other players have fixed policies \( u_j \)

\[ 0 = H_i(\delta_i, \frac{\partial V_i}{\partial \delta_i}, u_i^*, u_{-i}) = \frac{\partial V_i}{\partial \delta_i} A_i^c + \frac{1}{2} \delta_i^T Q_{ii} \delta_i + \frac{1}{2} (d_i + g_i)^2 \frac{\partial V_i}{\partial \delta_i} B_i R_{ii}^{-1} B_i^T \frac{\partial V_i}{\partial \delta_i} + \frac{1}{2} \sum_{j \in N_i} u_j^T R_{ij} u_j \]

where

\[ A_i^c = A \delta_i - (d_i + g_i)^2 B_i R_{ii}^{-1} B_i^T \frac{\partial V_i}{\partial \delta_i} - \sum_{j \in N_i} e_{ij} B_j u_j \]
Let $V_i > 0 \in C^1, i \in N$ be smooth solutions to HJ equations (23) and control policies $u^*_i, i \in N$ be given by (22) in terms of these solutions $V_i$. Then

\begin{enumerate}
\item Systems (8) are asymptotically stable.
\item $u^*_i, u^-_i$ are in cooperative Nash equilibrium and the corresponding game values are
\end{enumerate}

\[ J^*_i (\delta_i (0)) = V_i, i \in N \]  \hfill (34)

---

Theorem 2. Solution for Best Response Policy
Given fixed neighbor policies $u^-_i = \{u_j : j \in N_i\}$, assume there is an admissible policy $u_i$. Let $V_i > 0 \in C^1$ be a smooth solution to the best response HJ equation (36) and let control policy $u^*_i$ be given by (22) in terms of this solution $V_i$. Then

\begin{enumerate}
\item System (8) is asymptotically stable.
\item $u^*_i$ is the best response to the fixed policies $u^-_i$ of its neighbors.
\end{enumerate}
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)

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

(38)

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

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

which explicitly is

$$u_i^{k+1} = -(d_i + e_i)R_{ii}^{-1}B_i^T \frac{\partial V_i^k}{\partial \delta_i}, \forall i = 1, ..., N.$$  

(39)

Go to step 1.

On convergence End

---

**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).

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 $V_i^k \rightarrow V_i^*$. 
Online Solution of Graphical Games

New Structure of Adaptive Controller

Reinforcement Learning Adaptive Critic

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

Do not need to know system drift dynamics

\[ \dot{x}_i = f(x_i) + g(x_i)u_i \]


6. ADP Using Reduced State Information (Output Feedback) (Partially Observable Markov Decision Processes)

**DT Linear Quadratic Regulator Optimal Control**

**DT System**
\[ x_{k+1} = Ax_k + Bu_k \]
\[ y_k = Cx_k \]

**Performance measure**
\[ V^\mu(x_k) = \sum_{i=k}^{\infty} \left( y_i^T Q y_i + u_i^T R u_i \right) = \sum_{i=k}^{\infty} r_i \]

**Utility**
\[ r_k = y_k^T Q y_k + u_k^T R u_k \]

Value is quadratic in the state
\[ V^\mu(x_k) = x_k^T P x_k \]

**Algebraic Riccati Equation**
\[ 0 = A^T P A - P + C^T Q C - A^T P B (R + B^T P B)^{-1} B^T P A \]

**Optimal feedback gain (policy)**
\[ u_k = -K x_k = -(R + B^T P B)^{-1} B^T P A x_k \]

**Off-line solution**
Requires knowledge of system dynamics \( A, B, C \)

We want online solution of ARE using only measured input/output data
Bellman equation

\[ x_k^T P x_k = y_k^T Q y_k + u_k^T R u_k + x_{k+1}^T P x_{k+1} \]

Value function is Quadratic in the state

Policy Iteration Algorithm

i. Start with stabilizing control policy

ii. Value Update

\[ 0 = -x_k^T P^{j+1} x_k + y_k^T Q y_k + (u_k^i)^T R u_k^j + x_{k+1}^T P^{j+1} x_{k+1} \]

Lyapunov Equation

iii. Policy Improvement

\[ u_{k}^{j+1} = -(R + B^T P^{j+1} B)^{-1} B^T P^{j+1} A x_k \]

Value Iteration Algorithm

i. Start with ANY control policy

ii. Value Update

\[ x_k^T P^{j+1} x_k = y_k^T Q y_k + (u_k^i)^T R u_k^j + x_{k+1}^T P^{j+1} x_{k+1} \]

Lyapunov recursion

iii. Policy Improvement

\[ u_{k}^{j+1} = -(R + B^T P^{j+1} B)^{-1} B^T P^{j+1} A x_k \]

Requires state measurements
**Expanded State Equation (ESE)**

Express state in terms of inputs and outputs

System

\[ x_{k+1} = Ax_k + Bu_k \]
\[ y_k = Cx_k \]

Expanded State Equation

\[ x_k = A^N x_{k-N} + \begin{bmatrix} B & AB & A^2B & \cdots & A^{N-1}B \end{bmatrix} \begin{bmatrix} u_{k-1} \\ u_{k-2} \\ \vdots \\ u_{k-N} \end{bmatrix} \]

Reachability matrix \( U_N \)

Observability matrix \( V_N \)

Invertibility matrix of Markov Parameters \( T_N \)

\[ x_k = A^N x_{k-N} + U_N \bar{u}_{k-1,k-N} \]

\[ \bar{y}_{k-1,k-N} = V_N x_{k-N} + T_N \bar{u}_{k-1,k-N} \]
Express State in Terms of Previous Inputs & Outputs

Observable implies $V_N$ has full column rank $n$

$$A^N = MV_N$$

for $N$ greater than the observability index

$$M = A^N V_N^+ + Z(I - V_N V_N^+)$$

$$M \equiv M_0 + M_1$$

MP pseudoinverse is $$V_N^+ = (V_N^T V_N)^{-1} V_N^T$$

Projection on range perp. $V_N$ is $$P(R^\perp(V_N)) = I - V_N V_N^+$$

1. From $$\overline{y}_{k-1,k-N} = V_N x_{k-N} + T_N \overline{u}_{k-1,k-N}$$

$$A^N x_{k-N} = MV_N x_{k-N} = M\overline{y}_{k-1,k-N} - M T_N \overline{u}_{k-1,k-N}$$

$$(M_0 + M_1)V_N x_{k-N} = (M_0 + M_1)\overline{y}_{k-1,k-N} - (M_0 + M_1) T_N \overline{u}_{k-1,k-N}$$

but $$M_1 V_N = 0$$

so $$0 = M_1 \overline{y}_{k-1,k-N} - M_1 T_N \overline{u}_{k-1,k-N}, \quad \forall M_1 \text{ s.t. } M_1 V_N = 0$$

Then $$A^N x_{k-N} = M_0 V_N x_{k-N} = M_0 \overline{y}_{k-1,k-N} - M_0 T_N \overline{u}_{k-1,k-N}$$

2. From $$x_k = A^N x_{k-N} + U_N \overline{u}_{k-1,k-N}$$

Then state in terms of inputs & outputs is

$$x_k = M_0 \overline{y}_{k-1,k-N} + (U_N - M_0 T_N) \overline{u}_{k-1,k-N} \equiv M_y \overline{y}_{k-1,k-N} + M_u \overline{u}_{k-1,k-N}$$

Markov parameters

So

$$x_k = \begin{bmatrix} M_u & M_y \end{bmatrix} \begin{bmatrix} \overline{u}_{k-1,k-N} \\ \overline{y}_{k-1,k-N} \end{bmatrix}$$

$$\overline{y}_{k-1,k-N} = \begin{bmatrix} y_{k-1} \\ y_{k-2} \\ \vdots \\ y_{k-N} \end{bmatrix}, \quad \overline{u}_{k-1,k-N} = \begin{bmatrix} u_{k-1} \\ u_{k-2} \\ \vdots \\ u_{k-N} \end{bmatrix}$$
Prior Work

\[ x_k = M_0 \bar{y}_{k-1,k-N} + (U_N - M_0 T_N) \bar{u}_{k-1,k-N} \equiv M_y \bar{y}_{k-1,k-N} + M_u \bar{u}_{k-1,k-N} \]

But this needs to know dynamics \( A, B, C \) to compute \( M_y \) and \( M_u \)

A lot of work has been done to

express the optimal control policy

\[ u_k = -Kx_k = -(R + B^T PB)^{-1} B^T Pax_k \]

in terms of inputs and outputs

and identify the Markov Parameters online

We can avoid all this by using Reinforcement Learning techniques

\( RL \) Can learn the System Parameters online
Express Bellman Equation in Terms of Inputs & Outputs

Bellman Equation

\[ x_k^T P x_k = y_k^T Q y_k + u_k^T R u_k + x_{k+1}^T P x_{k+1} \]

Quadratic in state

We know

\[ x_k = \begin{bmatrix} M_u & M_y \end{bmatrix} \begin{bmatrix} \bar{u}_{k-1,k-N} \\ \bar{y}_{k-1,k-N} \end{bmatrix} \]

Value in terms of i/o

\[ V^u(x_k) = x_k^T P x_k = \bar{z}_{k-1,k-N}^T P \begin{bmatrix} M_u & M_y \end{bmatrix} \bar{z}_{k-1,k-N} \]

\[ V^u(x_k) = \bar{z}_{k-1,k-N}^T \begin{bmatrix} M_u^T P M_u & M_u^T P M_y \\ M_y^T P M_u & M_y^T P M_y \end{bmatrix} \bar{z}_{k-1,k-N} = \bar{z}_{k-1,k-N}^T \bar{P} \bar{z}_{k-1,k-N} \]

ARE solution and Markov Parameters

Bellman Equation

\[ \bar{z}_{k-1,k-N}^T \bar{P} \bar{z}_{k-1,k-N} = y_k^T Q y_k + u_k^T R u_k + \bar{z}_{k,k-N+1}^T \bar{P} \bar{z}_{k,k-N+1} \]

TD error

\[ e_k = -\bar{z}_{k-1,k-N}^T \bar{P} \bar{z}_{k-1,k-N} + y_k^T Q y_k + u_k^T R u_k + \bar{z}_{k,k-N+1}^T \bar{P} \bar{z}_{k,k-N+1} \]

Quadratic in previous inputs & outputs

We can use either PI or VI to learn the parameter matrix \( \bar{P} \)

ONLINE in Real-Time using measurements of inputs/outputs

Along the system trajectories

System parameters are not needed for Value Update Step
**Policy Update Step with no System Information**

**Bellman Equation**

\[
\overline{z}_{k-1,k-N}^T P \overline{z}_{k-1,k-N} = y_k^T Q y_k + u_k^T R u_k + \overline{z}_{k,k-N+1}^T P \overline{z}_{k,k-N+1}
\]

**Policy Update**

\[
u(x_k) = \arg \min_{u_k} \left( y_k^T Q y_k + u_k^T R u_k + \overline{z}_{k,k-N+1}^T P \overline{z}_{k,k-N+1} \right)
\]

**Value is quadratic in previous i/o**

\[
V^u(x_k) = \overline{z}_{k-1,k-N}^T \begin{bmatrix} M_u^T P M_u & M_u^T P M_y \\ M_y^T P M_u & M_y^T P M_y \end{bmatrix} \overline{z}_{k-1,k-N} = \overline{z}_{k-1,k-N}^T P \overline{z}_{k-1,k-N}
\]

\[
\overline{z}_{k-1,k-N} = \begin{bmatrix} \overline{u}_{k-1,k-N} \\ \overline{y}_{k-1,k-N} \end{bmatrix}
\]

**Partition as**

\[
\overline{z}_{k,k-N+1}^T P \overline{z}_{k,k-N+1} = \begin{bmatrix} u_k \\ \overline{u}_{k-1,k-N+1} \\ \overline{y}_{k,k-N+1} \end{bmatrix}^T \begin{bmatrix} p_0 & p_u & p_y \\ p_u^T & p_{u2} & p_{u3} \\ p_y^T & p_{y2} & p_{y3} \end{bmatrix} \begin{bmatrix} u_k \\ \overline{u}_{k-1,k-N+1} \\ \overline{y}_{k,k-N+1} \end{bmatrix}
\]

**Quadratic in inputs and outputs. Has same form as Q learning.**

**Policy Update Step**

\[
u(x_k) = \arg \min_{u_k} \left( y_k^T Q y_k + u_k^T R u_k + \overline{u}_{k-1,k-N+1}^T \begin{bmatrix} p_0 & p_u & p_y \\ p_u^T & p_{u2} & p_{u3} \\ p_y^T & p_{y2} & p_{y3} \end{bmatrix} \begin{bmatrix} u_k \\ \overline{u}_{k-1,k-N+1} \\ \overline{y}_{k,k-N+1} \end{bmatrix} \right)
\]

**Differentiate wrt \( u_k \)**

\[
0 = R u_k + p_0 u_k + p_u \overline{u}_{k-1,k-N+1} + p_y \overline{y}_{k,k-N+1}
\]

**Policy Update**

\[
u_k = -(R + p_0)^{-1} \left( p_u \overline{u}_{k-1,k-N+1} + p_y \overline{y}_{k,k-N+1} \right)
\]

**Do NOT NEED A or B!**

**Compare to**

\[
u_k^{j+1} = -(R + B^T P^{j+1} B)^{-1} B^T P^{j+1} A x_k
\]
Policy Iteration using output measurements

Policy Evaluation- solve Bellman equation for $\bar{P}$

$$\bar{z}_{k-1,k-N}^T \bar{P} \bar{z}_{k-1,k-N} = y_k^T Q y_k + u_k^T R u_k + \bar{z}_{k,k-N+1}^T \bar{P} \bar{z}_{k,k-N+1}$$

Unpack parameters into matrix form

$$\bar{z}_{k,k-N+1}^T \bar{P} \bar{z}_{k,k-N+1} = \begin{bmatrix} u_k \\ \bar{u}_{k-1,k-N+1}^T \\ \bar{y}_{k,k-N+1}^T \end{bmatrix}^T \begin{bmatrix} p_0 & p_u & p_y \\ p_u^T & P_{22} & P_{23} \\ p_y^T & P_{32} & P_{33} \end{bmatrix} \begin{bmatrix} u_k \\ \bar{u}_{k-1,k-N+1} \\ \bar{y}_{k,k-N+1} \end{bmatrix}$$

Control Update

$$u_k = -(R + p_0)^{-1} \left( p_u \bar{u}_{k-1,k-N+1} + p_y \bar{y}_{k,k-N+1} \right)$$

Does not need ANY system dynamics
Looks a lot like Q learning – but Q needs states
The Controller is in ARMA Polynomial Regulator Form!

\[ u_k = - (R + p_0)^{-1} \left( p_u \bar{u}_{k-1,k-N+1} + p_y \bar{y}_{k,k-N+1} \right) \]

An ARMA Controller that is equivalent to the optimal SVFB gain

Compare to the Optimal Polynomial regulator in Lewis and Syrmos, Optimal Control, 1995
Simulation Example

$x_{k+1} = \begin{bmatrix} 1.1 & -0.3 \\ 1 & 0 \end{bmatrix} x_k + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u_k$

$y_k = \begin{bmatrix} 1 & -0.8 \end{bmatrix} x_k$

Actual $P_{\text{bar}}$ matrix

$\hat{P} = \begin{bmatrix} M_u^T P M_u & M_u^T P M_y \\ M_y^T P M_u & M_y^T P M_y \end{bmatrix}$

Learned $P_{\text{bar}}$ matrix

$\hat{P} = \begin{bmatrix} 1.1340 & -0.8643 & 1.1571 & -0.3161 \\ -0.8643 & 0.7942 & -1.0348 & 0.2966 \\ 1.1571 & -1.0348 & 1.3609 & -0.3850 \\ -0.3161 & 0.2966 & -0.3850 & 0.1102 \end{bmatrix}$

Solves ARE online
Without knowing $A$, $B$ and without measuring states

Also works for unstable systems

$P = \begin{bmatrix} 1.0150 & -0.8150 \\ -0.8150 & 0.6552 \end{bmatrix}$

$\hat{P} = \begin{bmatrix} 1.0150 & -0.8440 & 1.1455 & -0.3165 \\ -0.8440 & 0.7918 & -1.0341 & 0.2969 \\ 1.1455 & -1.0341 & 1.3667 & -0.3878 \\ -0.3165 & 0.2969 & -0.3878 & 0.1113 \end{bmatrix}$

Convergence of $p_0$, $p_u$, $p_y$

State trajectories

$0 = A^T P A - P + C^T Q C - A^T P B (R + B^T P B)^{-1} B^T P A$
Our revels now are ended. These our actors,
As I foretold you, were all spirits, and
Are melted into air, into thin air.

The cloud-capped towers, the gorgeous palaces,
The solemn temples, the great globe itself,
Yea, all which it inherit, shall dissolve,
And, like this insubstantial pageant faded,
Leave not a rack behind.

We are such stuff as dreams are made on,
and our little life is rounded with a sleep.

Prospero, in The Tempest, act 4, sc. 1, l. 152-6, Shakespeare
The way that can be told is not the Constant Way
The name that can be named is not the Constant Name

For nameless is the true way
Beyond the myriad experiences of the world

To experience without intention is to sense the world

All experience is an arch
wherethrough gleams that untravelled land
whose margins fade forever as we move

Dao ke dao feichang dao
Ming ke ming feichang ming