Optimal Control and Online Game Solutions Using ADP

ADP Using Output Feedback

Supported by:
NSF - PAUL WERBOS

http://ARRI.uta.edu/acs
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

Outline
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

- Online synchronous optimal adaptive control - Vamvoudakis
- Integral Reinforcement Learning- Vrabie
- IRL for synchronous optimal adaptive control

D. ADP with OPFB
Continuous-Time Optimal Control

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

Cost/value
\[ V(x(t)) = \int_{t_0}^{\infty} r(x,u) \, dt = \int_{t_0}^{\infty} (Q(x) + u^T R u) \, dt \]

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
\[ 0 = \left( \frac{dV^*}{dx} \right)^T f(x) + Q(x) - \frac{1}{2} \left( \frac{dV^*}{dx} \right)^T g R^{-1} g^T \frac{dV^*}{dx} \quad , \quad V(0) = 0 \]

Off-line 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
\[ 0 = H(x, \frac{\partial V}{\partial x}, u) = \dot{x} + x^T Q x + u^T R u = 2 \left( \frac{\partial V}{\partial x} \right)^T \dot{x} + x^T Q x + u^T R u = 2 x^T P (Ax + Bu) + x^T Q x + u^T R u \]

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

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

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
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 greedy value iteration 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?

Small Time-Step Approximate Tuning for Continuous-Time Adaptive Critics

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

Use Euler’s forward approximation
\[ H(x, \frac{\partial V}{\partial x}, u) = \dot{V}(x) + r(x,u) \approx \frac{V_{t+1} - V_t}{\Delta t} + r(x,u) \approx \frac{V_{t+1} - V_t}{\Delta t} + \frac{r^D(x_t, u_t)}{\Delta t} \]

- This defines a temporal difference error.
- System dynamics does NOT appear
- Has TWO occurrences of value V and so can be used to define greedy Value Iteration
- NOTE that the UTILITY must also be discretized or it does not work

Baird’s Advantage function
\[ A^*_i(x_t, u_t) = \frac{r^D(x_t, u_t) + V(x_{t+1}) - V^*(x_t)}{\Delta t} \]

Problems:
- Only an approximation
- Must select a constant sampling period up front
- Physics-based CT nonlinearities are lost
CT Policy Iteration

To avoid solving HJB equation
\[ 0 = \left( \frac{dV^*}{dx} \right)^T f + Q(x) + \frac{1}{2} \left( \frac{dV^*}{dx} \right)^T g R^{-1} g^T \frac{dV^*}{dx} \]

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

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

**CT Bellman equation**

**Policy Iteration Solution**

Pick stabilizing initial control policy

**Policy Evaluation** - Find cost, Bellman eq.
\[ 0 = \left( \frac{\partial V}{\partial x} \right)^T f(x, h_j(x)) + r(x, h_j(x)) \]
\[ V_j(0) = 0 \]

**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
1. Online Solution of Optimal Control for Nonlinear Systems

Optimal Adaptive Control

Policy Iteration gives the structure needed for online 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

converts Bellman PDE into algebraic equation

Critic NN

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

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

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

Comes from \( h(x) = -\frac{1}{2} R^{-1} g^T (x) \nabla V \), \( \nabla V(x) = \nabla \phi_1^T W_1 \)
Online Synchronous Policy Iteration

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

Let $\sigma_i = \nabla \phi_i (f + gu)$ be PE. Tune critic NN weights as

\[
\hat{\theta}_1 = -\alpha_1 \frac{\partial E_1}{\partial \hat{\theta}_1} = -\alpha_1 \frac{\sigma_i}{(\sigma_i^T \sigma_i + 1)^2} \left[ \sigma_1^T \hat{\theta}_1 + \phi(x) + u^T R u \right]
\]

Learning the Value

Tune actor NN weights as

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

Learning the control policy

where $D_1(x) = \nabla \phi_1 (x) g(x) R^{-1} g^T (x) \nabla \phi_1^T (x)$, $m = \frac{\sigma_1}{(\sigma_1^T \sigma_1 + 1)^2}$

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 $\hat{W}_1 = W_1 - \hat{W}_1$

and the actor NN error $\hat{W}_2 = W_1 - \hat{W}_2$ 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

\[
\hat{\theta}_1 = -\alpha_1 \frac{\partial E_1}{\partial \hat{\theta}_1} = -\alpha_1 \frac{\sigma_1}{(\sigma_1^T \sigma_1 + 1)^2} \left[ \sigma_1^T \hat{\theta}_1 + \phi(x) + u^T R u \right]
\]

Tune actor NN weights as

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

Extra terms needed for stability

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:

\[ L(t) = V(x) + \frac{1}{2} tr(W_1^T a_1^{-1} \tilde{W}_1) + \frac{1}{2} tr(W_2^T a_2^{-1} \tilde{W}_2). \]

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

\[ 0 = \left(\frac{dV}{dx}\right)^T f + Q(x) - \frac{1}{2} \left(\frac{dV}{dx}\right)^T g R^{-1} g^T \frac{dV}{dx} \]

Guarantees stability

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

\( W_1 \) = Unknown LS solution to Lyap. eq for given \( N \)

\[ H(x,W_1,u) = W_1^T \nabla \phi(f + gu) + Q(x) + u^T Ru = \varepsilon_H \]

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
Simulation 1- F-16 aircraft pitch rate controller

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

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

Select quadratic NN basis set for VFA

Exact solution

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

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

Must add probing noise to get PE

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

(Exponentially decay \( n(t) \))

Algorithm converges to

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

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

\[
\dot{w}_2(x) = -\frac{1}{2} R^{-1} g^T P_x = -\frac{1}{2} R^{-1}
\]

Critic NN parameters- Converge to ARE solution

System states
Simulation 2. – Nonlinear System

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

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

\[ g(x) = \begin{bmatrix} 0 \\ \cos(2x_2) + 2 \end{bmatrix} \]

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

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 \[ \phi(x) = [x_1^2 \ x_1 \ x_2 \ x_2^2]^T. \]

Algorithm converges to

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

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

\[ \hat{u}_2(x) = -\frac{1}{2}R^{-1} \begin{bmatrix} 0 \\ \cos(2x_1) + 2 \end{bmatrix} \begin{bmatrix} 2x_1 & 0 \\ x_2 & x_1 \\ 0 & 2x_2 \end{bmatrix} \begin{bmatrix} 0.5017 \\ -0.0020 \\ 1.0008 \end{bmatrix} \]
2. H-Infinity Control Using Neural Networks

**System**

\[
x = f(x) + g(x)u + k(x)d
\]

\[
y = x
\]

\[
z = \psi(x, u)
\]

\[
u = I(y)
\]

*where*

\[
\|y\|^2 = h^T h + \|u\|^2
\]

**L₂ Gain Problem**

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

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

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

2 players

Cost
\[
V(x(t), u, d) = \int_0^\infty \left( h^T h + u^T Ru - \gamma^2 \|d\|^2 \right) dt = \int_0^\infty r(x, u, d) dt
\]

Differential equivalent is ZS game Bellman equation

\[
0 = r(x, u, d) + \dot{V} = r(x, u, d) + (\nabla V)^T (f(x) + g(x)u + k(x)d) = H(x, \frac{\partial V}{\partial x}, u, d)
\]

\[
V(0) = 0
\]

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

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

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

\[ H(x, \frac{\partial V}{\partial x}, u, d) = h^T h + u^T R u - \gamma^2 \|d\|^2 + (\nabla V)^T (f(x) + g(x)u + k(x)d) \]

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**

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

\[ y = C x \]

\[ -J_2(x(t), u_1, u_2) = J_1(x(t), u_1, u_2) = \frac{1}{2} \int_{t}^{\infty} \left( x^T Q x + u_1^T R_1 u_1 - u_2^T R_2 u_2 \right) d\tau \]

, \quad Q = C^T C

Game Algebraic Riccati Equation

\[ 0 = A^T P + PA + Q - PB_1 R_1^{-1} B_1^T P + PB_2 R_2^{-1} B_2^T P \]

\[ u_1 = -K_1 x = -R_1^{-1} B_1^T P x, \quad u_2 = K_2 x = R_2^{-1} B_2^T P x \]
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 $V_{j+1}(x(t))$

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

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

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 $V_{j+1}(x(t))$

$$0 = h^T h + \nabla V_{j}^T(x) \left( f + gu_j + k d^i \right) + u_j^T(x)Ru_j - \gamma^2 \left\| d^i \right\|^2$$

$$d^{i+1} = \frac{1}{2\gamma^2} k^T(x) \nabla V_{j}$$

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

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 approx. 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

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

Use Three Neural Networks

Critic NN

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

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

\[ u(x) = -\frac{1}{2} R^{-1} g^T(x) \nabla \phi_1^T \hat{W}_2 \quad \text{Comes from} \quad \mu(x) = -\frac{1}{2} R^{-1} g^T(x) \nabla V \]

Disturbance actor NN

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

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

\[ \hat{\theta}_2 = -\alpha_2 \frac{\sigma_2}{(\sigma_2^T \sigma_2 + 1)^2} \left[ \sigma_2^T \hat{\theta}_1 + \sigma_2^T \theta' \right] \]

Learning the Value

Tune actor NN weights as

\[ \begin{align*}
\dot{\hat{\theta}}_2 &= -\alpha_2 \left( (F_2 \hat{\theta}_2 - F_1 \sigma_2^T \hat{\theta}_1) - \frac{1}{4} D_1(x) \hat{\theta}_2 m^T(x) \hat{\theta}_1 \right) \\
\dot{\hat{\theta}}_3 &= -\alpha_3 \left( (F_4 \hat{\theta}_3 - F_3 \sigma_2^T \hat{\theta}_1) + \frac{1}{4} \sigma_2^T \sigma_2 \hat{\theta}_1^T m(x) \right)
\end{align*} \]

Learning the control policies

where \( D_1(x) = \nabla \phi(x) g(x) R^{-1} g^T(x) \nabla \phi^T(x) \),
\( E_1(x) = \nabla \phi(x) k_k^T \nabla \phi^T(x) \),
\[ m = \frac{\sigma_2}{(\sigma_2^T \sigma_2 + 1)^2} \]

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

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

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

are UUB bounded.

---

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

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

\[y = C^T x\]

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

Solves GARE online \(A^T P + PA + Q - PBR^{-1}B^T P + \frac{1}{\gamma^2}PKK^T P = 0\)

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

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

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

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

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

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

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

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

\[
\dot{\hat{W}}_1(x) = -\frac{1}{2}R^{-1}g^T (x)\nabla \phi^T \hat{W}_1 + n(t)
\]

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

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

\[ f(x) = \begin{bmatrix} -x_1 + x_2 \\ -x_1^3 - x_2^3 + 0.25x_1\cos(2x_1) + x_1^2 - 0.25x_2 \frac{1}{x_1}(\sin(4x_1) + 2x_1) \end{bmatrix} \]

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

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

Optimal Value

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

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

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

Select VFA basis set

\[ \phi_i(x) = [x_1^i, x_2^i, x_1^4, x_2^4] \]

Algorithm converges to

\[ \hat{W}_1(t_f) = \begin{bmatrix} 0.0008 \\ 0.4999 \\ 0.2429 \\ 0.0032 \end{bmatrix} \]

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

\[ \hat{u}_2(x) = -\frac{1}{2}R^{-1}\begin{bmatrix} 0 \\ \cos(2x_1) + 2 \end{bmatrix}^T \begin{bmatrix} 2x_1 \\ 0 \\ 2x_2 \\ 4x_1^2 \\ 4x_2^2 \\ 0 \end{bmatrix} \begin{bmatrix} 0.0008 \\ 0.4999 \\ 0.2429 \\ 0.0032 \end{bmatrix} \]

\[ \hat{d}(x) = \frac{1}{2\gamma^2}\begin{bmatrix} 0 \\ \sin(4x_1) + 2 \end{bmatrix}^T \begin{bmatrix} 2x_1 \\ 0 \\ 2x_2 \\ 4x_1^3 \\ 4x_2^3 \\ 0 \end{bmatrix} \begin{bmatrix} 0.0008 \\ 0.4999 \\ 0.2429 \\ 0.0032 \end{bmatrix} \]
Critic NN parameters

Value fn. approx. error

Control approx error

Dist. approx error
3. Real-Time Solution of Multi-Player NZS Games

Kyriakos Vamvoudakis

Multi-Player Nonlinear Systems

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

Optimal control

$$V^*_i(x(0), \mu_1, \mu_2, \ldots, \mu_N) = \min_{\mu_i} \int_{0}^{\infty} \left( Q_i(x) + \sum_{j=1}^{N} \mu_j^T R_{ij} \mu_j \right) dt; \quad i \in N$$

Nash equilibrium

$$V^*_i \triangleq V_i(\mu_1^*, \mu_2^*, \ldots, \mu_N^*) \leq V_i(\mu_1, \mu_2, \ldots, \mu_N), \quad i \in N$$

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

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

Control policies

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

Linear Quadratic Regulator Case–coupled AREs

$$0 = R_iA_i + A_i^T R_i + Q_i + \sum_{j=1}^{N} P_j B_j R_{ij} R_j^{-1} B_j^T P_j, \quad i \in N$$

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

$$V_i(x(0), \mu_1, \mu_2, \ldots, \mu_N) = \int_{0}^{\infty} \left( Q_i(x) + \sum_{j=1}^{N} \mu_j^T R_{ij} \mu_j \right) dt; \quad i \in N$$

Differential equivalent gives Bellman eqs.

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

Policy Iteration Solution:

Solve Bellman eq.

$$0 = r(x, \mu^k_1, \ldots, \mu^k_N) + (V^k)^T \left( f(x) + \sum_{j=1}^{N} g_j(x)\mu^k_j \right), \quad V^k_i(0) = 0 \quad i \in N$$

Policy Update

$$\mu^k_{i+1}(x) = -\frac{1}{2} R_{ii}^{-1} g_i^T(x)V^k_i, \quad i \in N$$

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.**

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

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

<table>
<thead>
<tr>
<th>2-player case</th>
<th>Player 1</th>
<th>Player 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>N Critic Neural Networks for VFA</td>
<td>( \hat{V}_1(x) = \hat{W}_1^T \phi_1(x) )</td>
<td>( \hat{V}_2(x) = \hat{W}_2^T \phi_2(x) )</td>
</tr>
<tr>
<td>N Actor Neural Networks</td>
<td>( u_1(x) = -\frac{1}{2} R_{11}^{-1} g_1^T(x)\nabla \phi_1^T \hat{W}_3 )</td>
<td>( u_2(x) = -\frac{1}{2} R_{22}^{-1} g_2^T(x)\nabla \phi_2^T \hat{W}_4 )</td>
</tr>
</tbody>
</table>

On-Line Learning – for Player 1:

\[
\dot{\hat{W}}_1 = -a_1 \frac{\sigma_i}{(\sigma_i^2 + 1)^2} [\sigma_i^T \hat{W}_1 + Q_1(x) + u_1^T R_1 u_1 + u_2^T R_2 u_2]
\]

Learns Bellman eq. solution

\[
\dot{\hat{W}}_3 = -\alpha_3 \{ F_3 \hat{W}_3 - F_3 \hat{W}_3^T \hat{W}_3 \} - \frac{1}{4} \nabla \phi \hat{g}(x) R_1^{-1} R_2 R_2^{-1} g(x) \nabla \phi^T \hat{W}_3 \hat{m}_3 \hat{W}_3 - \frac{1}{4} I_3(x) \hat{W}_3 \hat{m}_3^T \hat{W}_3
\]

Learns control policy
Simulation. – Nonlinear System – 2-player game

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

\[
f(x) = \begin{bmatrix} x_2 \\ -x_2 - \frac{1}{2}x_1 + \frac{1}{2}x_3(\cos(2x_1) + 2) + \frac{1}{2}x_2(\sin(4x_1) + 2) \end{bmatrix}
\]

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

\( Q_1 = 2Q_2 = 2I, \quad R_{11} = 2R_{22} = 2I, \quad R_{12} = 2R_{21} = 2I \)

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

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

\[
d^*(x) = -(\sin(4x_1^3) + 2)x_2
\]

Solves HJB equation online

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

Select VFA basis set

\[
\phi_1(x) = \phi_2(x) = [x_1^2, x_1x_2, x_2^2]
\]

Algorithm converges to

\[
\hat{W}_i(t_f) = \begin{bmatrix} 0.5015 & 0.0007 & 1.0001 \end{bmatrix} = \hat{W}_3(t_f)
\]

\[
\hat{W}_2(t_f) = \begin{bmatrix} 0.2514 & 0.0006 & 0.5001 \end{bmatrix} = \hat{W}_4(t_f)
\]

\[
\hat{u}(x) = -\frac{1}{2}R_{11}^{-1} \begin{bmatrix} 0 & 2x_1 \\ \cos(2x_1) + 2 & x_1 \\ 0 & 2x_2 \end{bmatrix}^T \begin{bmatrix} 0.5015 \\ 0.0007 \\ 1.0001 \end{bmatrix} = \hat{W}_1(t_f)
\]

\[
\hat{d}(x) = -\frac{1}{2}R_{22}^{-1} \begin{bmatrix} 0 & 2x_1 \\ \sin(4x_1^3) + 2 & x_1 \\ 0 & 2x_2 \end{bmatrix}^T \begin{bmatrix} 0.2514 \\ 0.0006 \\ 0.5001 \end{bmatrix}
\]

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.
4. 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

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

This can be done online without knowing \( f(x) \)
using measurements of \( x(t), u(t) \) along the system trajectories
Lemma 1 – Draguna Vrabie

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

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 - D. Vrabie- LQR case

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

\[ A_c = A - BL \]

is equivalent to

Integral reinf. 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:

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

\[ \int_t^{t+T} x^T(Q+L^T RL)xd\tau = -\int_t^{t+T} d(x^T P x) = x^T(t)Px(t) - x^T(t+T)Px(t+T) \]
Integral Reinforcement Learning (IRL)- Draguna Vrabie

**Policy iteration**

<table>
<thead>
<tr>
<th>Policy evaluation</th>
<th>CT Bellman eq.</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>

$f(x)$ 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 nonlinear Lyapunov eq. without knowing system dynamics

**Policy improvement**

<table>
<thead>
<tr>
<th>Control gain update</th>
<th>$u_{k+1} = h_{k+1}(x) = -\frac{1}{2} R^{-1} g(x) \frac{\partial V}{\partial x}$</th>
</tr>
</thead>
</table>

$g(x)$ needed for control update

Initial stabilizing control is needed

Converges to solution to HJB eq.

$$0 = \left( \frac{dV^*}{dx} \right)^T f + Q(x) - \frac{1}{2} \left( \frac{dV^*}{dx} \right)^T g R^{-1} g^T \frac{dV^*}{dx}$$

D. Vrabie proved convergence to the optimal value and control

---

**Integral Reinforcement Learning (IRL)- Draguna Vrabie**

**CT LQR Case**

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

**CT Bellman eq.**

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

is equivalent to

$$A_k^T P_k + P_k A_k + L_k^T R L_k + Q = 0$$

$$A_k = A - BL_k$$

$L_{k+1} = R^{-1} B^T P_k$

Only $B$ is needed

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

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

Lewis and Syrmos 1995

Policy must be stabilizing to solve this eq.

Define Contraction map

Bellman Eq.

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

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

Recall

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

Recall equivalent CT Policy Iteration – How to implement online?

Linear Systems Quadratic Cost- LQR

Policy evaluation

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

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

\[ \begin{bmatrix} x^T(\tau) & x^T(\tau) \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} \begin{bmatrix} x^T(\tau) \\ x^T(\tau) \end{bmatrix} - \begin{bmatrix} x^T(\tau) \\ x^T(\tau) \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} \begin{bmatrix} x^T(\tau+T) \\ x^T(\tau+T) \end{bmatrix} \]

\[ = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} \begin{bmatrix} 2x^T \dot{x}^T \\ 2x^T \dot{x}^T \end{bmatrix} - \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} \begin{bmatrix} (x^T)^2 \\ (x^T)^2 \end{bmatrix} \]

\[ = \tilde{p}_{ij} \begin{bmatrix} \bar{x}(\tau) - \bar{x}(t+T) \end{bmatrix} \]

Quadratic basis set
Algorithm Implementation

Critic update

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

Use Kronecker product

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

To set this up as

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

c.f. Linear in the parameters system ID

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

Quadric regression vector

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

Same form as standard System ID problems

Solve using RLS or batch LS

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

Nonlinear Case- Approximate Dynamic Programming

Value Function Approximation (VFA) – 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 Neural Network

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

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

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

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

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

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

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

Batch LS Algorithm Implementation

Or use Recursive Least-Squares solution along the trajectory

The Critic update

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

can be setup as

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

\[\overline{x}(t) = x(t) \otimes x(t)\]

is the quadratic basis set

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

\[\overline{p}_k = (XX^T)^{-1} XY\]

\[X = [\phi(t) \quad \phi(t+T) \quad ... \quad \phi(t+NT)]\]

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

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

Regression vector must be PE

Relates to choice of reinforcement interval \( T \)

Implementation

Policy evaluation
Need to solve online

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

Add a new state

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

This is the controller dynamics or memory
Direct Optimal Adaptive Controller

Solves Riccati Equation Online without knowing A matrix

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

Update FB gain after Critic has converged

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

Reinforcement interval T can be selected on line on the fly – can change

Gain update (Policy)

Control

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

Continuous-time control with discrete gain updates
Simulation 1- F-16 aircraft pitch rate controller

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

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

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

Select quadratic NN basis set for VFA

Exact solution

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

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

Simulations on: F-16 autopilot

Load frequency control for power system

A matrix not needed

Converge to SS Riccati equation soln

Solves ARE online without knowing A

\[0 = PA + A^T P + Q - PBR^{-1} B^T P\]
Comparison of CT IRL ADP to Discrete-Time ADP

\[ \dot{x} = Ax + Bu \]
\[ x(t) = \begin{bmatrix} \Delta \rho(t) & \Delta P_e(t) & \Delta X(t) & \Delta E(t) \end{bmatrix}^T \]

\[
A = \begin{bmatrix}
-1/T_p & \frac{1}{T_e} & 0 & 0 \\
0 & -1/T_p & 1/T_e & 0 \\
-1/RT_e & 0 & -1/T_e & -1/T_i \\
K_e & 0 & 0 & 0
\end{bmatrix},
B = \begin{bmatrix} 0 \\
0 \\
1/T_i \\
0
\end{bmatrix}
\]

Frequency Generator output
Governor position
Integral control

a. Use discrete-time ADP

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

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

b. Use continuous-time IRL ADP

\[ 0 = PA + A^T P + Q - PBR^{-1}B^T P \]
Solves ARE online without knowing A

IRL period of \( T = 0.1 \) s.
Fifteen data points \( (x(t), x(t+T), \rho(t:t+T)) \)
Hence, the value estimate was updated every 1.5 s.

Less computation is needed using CT IRL
In DT ADP sampling period is 0.01 s and the critic parameter estimates were updated every 0.15 s.
Yet, the parameter estimates for the P matrix entries almost overlay each other.
Issues with Nonlinear ADP

LS local smooth solution for Critic NN update

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

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

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

Recursive Least-Squares RLS

Selection of NN Training Set

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
5. Online Learning for Two-Player Zero-Sum Games

Online game solutions without knowing $A$ matrix

- **System dynamics**
  \[
  \dot{x} = Ax + B_1w + B_2u, \quad x \in R^n, \quad u \in R^m, \quad w \in R^q
  \]

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

- **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 costs;
   c. then informs Player 2 on his new optimized behavior policy.
3. Player 2 starts playing using the policy of his opponent.
4. Player 1
   a. corrects iteratively his own behavior using reinforcement learning such that its cost is again minimized;
   b. then informs Player 2 on his new optimized behavior policy.
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. \( i = 1; \quad P_{u}^{i-1} = P_{u}^{0} = 0; \quad w_{i} = B_{1}^{T} P_{u}^{0} x = 0 \)
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 \( P_{u}^{1} \) 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;
   \[
   Z_{u}^{i} A_{u}^{i-1} + A_{u}^{i-1}^{T} Z_{u}^{i} - Z_{u}^{i} B_{2} B_{2}^{T} Z_{u}^{i} + Z_{u}^{i-1} - B_{1} B_{1}^{T} Z_{u}^{i-1} = 0 \\
   P_{u}^{i} = Z_{u}^{i} + P_{u}^{i-1} \\
   u_{i} = -B_{2}^{T} P_{u}^{i} x 
   \]
   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.

Convergence proven by Lanzon, Feng, Anderson 2009

Riccati equations can be solved using HDP without knowledge of the A matrix.
Integral Reinforcement Learning (IRL) to solve ARE - Draguna Vrabie

CT Bellman eq.

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

Solves Lyapunov equation without knowing A or B

Only B is needed

Converges to solution to ARE

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

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
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
  \[
  P^\infty_u = \Pi = \begin{bmatrix}
  0.6036 & 0.7398 & 0.0609 & 0.5877 \\
  0.7398 & 1.5438 & 0.1702 & 0.5978 \\
  0.0609 & 0.1702 & 0.0502 & 0.0357 \\
  0.5877 & 0.5978 & 0.0357 & 2.3307 \\
  \end{bmatrix}
  \]
- Online learned solution using ADP – after 5 updates of the parameters
  \[
  P^5_u = \begin{bmatrix}
  0.6036 & 0.7399 & 0.0609 & 0.5877 \\
  0.7399 & 1.5440 & 0.1702 & 0.5979 \\
  0.0609 & 0.1702 & 0.0502 & 0.0357 \\
  0.5877 & 0.5979 & 0.0357 & 2.3307 \\
  \end{bmatrix}
  \]
  Solves GARE online without knowing \( A \)

Parameters of the critic – ARE Solution elements

- 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
6. Online Synchronous Policy Iteration using IRL

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

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

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

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

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

**Learning the Value**

Tune actor NN weights as

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

**Learning the control policy**

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

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

and the actor NN error \( \hat{W}_2 = W_2 - \dot{\hat{W}}_2 \) are UUB bounded.

See the talk Thursday morning at 9am
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**

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

**Utility**

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

Value is quadratic in the state

\[ V^H(x_k) = x_k^T P x_k \]

**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)**

\[ u_k = -Kx_k = -(R + B^T P B)^{-1} B^T P A x_k \]

**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**

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

**Quadratic in the state**

**Policy Iteration Algorithm**

i. Start with stabilizing control policy

ii. Value Update

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

**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

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

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

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
\[ M = A^N V_N + Z(I - V_N V_N^T) = M_0 + M_1 \]

MP pseudoinverse is
\[ V_N^+ = (V_N^T V_N)^{-1} V_N^T \]

Projection on range perp. \( V_N \)
\[ \text{is } \quad P(R^c(V_N)) = I - V_N V_N^T \]

1. From
\[ \bar{y}_{k-1,k-N} = V_N x_{k-N} + T_N \bar{u}_{k-1,k-N} \]
\[ A^N x_{k-N} = MV_N x_{k-N} = M\bar{y}_{k-1,k-N} - MT_N \bar{u}_{k-1,k-N} \]
\[ (M_0 + M_1) V_N x_{k-N} = (M_0 + M_1) \bar{y}_{k-1,k-N} - (M_0 + M_1) T_N \bar{u}_{k-1,k-N} \]

but \[ M_1 V_N = 0 \]

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

Then
\[ A^N x_{k-N} = M_0 V_N x_{k-N} = M_0 \bar{y}_{k-1,k-N} - M_0 T_N \bar{u}_{k-1,k-N} \]

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

Then state in terms of inputs & outputs is
\[ x_k = M_0 \bar{y}_{k-1,k-N} + (U_N - M_0 T_N) \bar{u}_{k-1,k-N} = M_1 \bar{y}_{k-1,k-N} + M_u \bar{u}_{k-1,k-N} \]

Markov parameters
Prior Work

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

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

in terms of inputs and outputs

and identify the Markov Parameters online

We can avoid all this by using Reinforcement Learning techniques

\textit{RL Can learn the System Parameters online}

Express Bellman Equation in Terms of Inputs & Outputs

\textbf{Bellman Equation}
\[ x_k^T P x_k = y_k^T Q y_k + u_k^T R u_k + x_{k+1}^T P x_{k+1} \quad \text{Quadratic in state} \]

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

Value in terms of i/o
\[ V^\mu(x_k) = x_k^T P x_k = \overline{x}_{k-1,k-N}^T \begin{bmatrix} M_u^T & M_y^T \end{bmatrix} \begin{bmatrix} M_u & M_y \end{bmatrix} \overline{x}_{k-1,k-N} \]

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

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

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

\textbf{Quadratic in previous inputs & outputs}

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

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

Policy Update
\[ \mu(x_k) = \arg\min_{u_k} \left( y_k^T Q y_k + u_k^T R u_k + T_{k-1,N+k} \right) \]

Value is quadratic in previous i/o
\[ V^P(x_k) = T_{k-1,N+k} \begin{bmatrix} T_{u} & T_p & T_{y} \end{bmatrix} u_k = T_{k-1,N+k} \]

Partition as \[ T_{k-1,N+k} = \begin{bmatrix} u_k & \bar{u}_{k-1,N+1} & \bar{y}_{k,N+1} \end{bmatrix} \]

Policy Update Step
\[ \mu(x_k) = \arg\min_{u_k} \left( y_k^T Q y_k + u_k^T R u_k + T_{k-1,N+k} \right) \]

Differentiate wrt \( u_k \)
\[ 0 = R u_k + p_0 u_k + p_u \bar{u}_{k-1,N+1} + p_y \bar{y}_{k,N+1} \]

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

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

Unpack parameters into matrix form
\[ \bar{T}_{k-1,N+k} = \begin{bmatrix} u_k & \bar{u}_{k-1,N+1} & \bar{y}_{k,N+1} \end{bmatrix} \begin{bmatrix} T_{u} & T_p & T_{y} \end{bmatrix} \]

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

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

Also works for unstable systems

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

\[
\begin{bmatrix} 1.0150 & -0.8440 & 1.1455 & -0.3165 \\ -0.8440 & 0.7918 & -1.0341 & 0.2969 \\ 1.1455 & -1.0341 & 1.3667 & -0.3878 \\ -0.3165 & 0.2969 & -0.3878 & 0.1113 \end{bmatrix}
\]

Actual Pbar matrix

\[
\bar{P} = \begin{bmatrix} M_u^T P M_u & M_u^T P M_y \\ M_y^T P M_u & M_y^T P M_y \end{bmatrix}
\]

Learned Pbar matrix

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

Convergence of \(p_u, p_y, p_v\)

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

Solves ARE online

Without knowing \(A, B\) and without measuring states
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