F.L. Lewis, NAI

Moncrief-O’Donnell Chair, UTA Research Institute (UTARI)
The University of Texas at Arlington, USA

and

Qian Ren Consulting Professor, State Key Laboratory of Synthetical Automation for Process Industries
Northeastern University, Shenyang, China

Reinforcement Learning and Adaptive Dynamic Programming (ADP) for Discrete-Time Systems

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

Supported by:
NSF
ONR

Supported by:
China NNSF
China Project 111
Invited by Tianyou Chai
Kung Tz  500 BC
Confucius

Tian xia da tong
Harmony under heaven

Archery
Chariot driving

Music
Rites and Rituals

Poetry
Mathematics

Man’s relations to
Family
Friends
Society
Nation
Emperor
Ancestors
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.
SHORT COURSE OVERVIEW
Day 1- Markov Decision Processes and Discrete-time Approximate Dynamic Programming (ADP)
Day 2- ADP and Online Differential Games for Continuous-time Systems
Day 3- Data-driven Real-time Process Optimization, Supervisory Control, and Structured Controllers
Books


New Chapters on:
- Reinforcement Learning
- Differential Games


Importance of Feedback Control

Darwin- FB and natural selection
Volterra- FB and fish population balance
Adam Smith- FB and international economy
James Watt- FB and the steam engine
FB and cell homeostasis

The resources available to most species for their survival are meager and limited

Nature uses Optimal control
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

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

Objectives
Get to orbit in minimum time
Use minimum fuel

http://microsat.sm.bmstu.ru/e-library/Launch/Dnepr_GEO.pdf
Adaptive Control is generally not Optimal

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

We want ONLINE DIRECT ADAPTIVE OPTIMAL Control
For any performance cost of our own choosing

Reinforcement Learning turns out to be the key to this!


Different methods of learning

Reinforcement learning
Ivan Pavlov 1890s

We want OPTIMAL performance
- ADP- Approximate Dynamic Programming

Actor-Critic Learning

Desired performance

Reinforcement signal

Critic

Tune actor

Adaptive Learning system

System

Control Inputs

outputs

environment
Outline
Data-Based Online Optimal Control for Discrete-Time Systems
   Policy Iteration
   Value Iteration (HDP- Heuristic Dynamic Programming)
   Q learning
Discrete-time Linear Systems Quadratic cost (LQR)

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

cost
\[ V(x_k) = \sum_{i=k}^{\infty} x_i^T Q x_i + u_i^T R u_i \]

Hamiltonian gives the Bellman equation
\[ V(x_k) = x_k^T Q x_k + u_k^T R u_k + V(x_{k+1}) \]

Fact. The cost is quadratic
\[ V(x_k) = x_k^T P x_k \]

for some symmetric matrix \( P \)

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

Optimal control comes from stationarity condition
\[ \min_{u_k} \left( x_k^T Q x_k + u_k^T R u_k + V(x_{k+1}) \right) \]

\[ 2Ru_k + 2(Bu_k)^T P(Ax_k + Bu_k) = 0 \]

Optimal control
\[ u_k = -(R + B^T PB)^{-1} B^T PA x_k \]

DT Riccati equation
\[ 0 = A^T PA - P + Q - A^T PB(R + B^T PB)^{-1} B^T PA \]
Optimal Control- The Linear Quadratic Regulator (LQR)

User prescribed optimization criterion

\[ J = (Q, R) \]

Off-line Design Loop
Using ARE

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

\[ L = (R + B^T PB)^{-1} B^T PA \]

Control Loop

\[ u \]
\[ x \]

On-line real-time
Control Loop

System

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

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

System modeling is expensive, time consuming, and inaccurate
We want to find optimal control solutions
Data-Based Method
Online in real-time
Using adaptive control techniques
Without knowing the full dynamics

For nonlinear systems and general performance indices

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

system

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

cost

\[ V(x_k) = \sum_{i=k}^{\infty} x_i^T Q x_i + u_i^T R u_i \]

Bellman equation

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

The cost is quadratic

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

for some symmetric matrix P

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

\[ x_k^T P x_k = x_k^T Q x_k + u_k^T R u_k + (Ax_k + Bu_k)^T P (Ax_k + Bu_k) \]

Assume ANY stabilizing fixed feedback policy

\[ u_k = -K x_k \]

Bellman eq= Lyapunov equation

\[ 0 = x_k^T ((A - BK)^T P (A - BK) - P + Q + K^T R K) x_k \]

Then

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

is the cost of using the SVFB

\[ u_k = -K x_k \]
Discrete-Time Systems Optimal Adaptive Control

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

cost \[ V_h(x_k) = \sum_{i=k}^{\infty} \gamma^{i-k} r(x_i, u_i) \]

Example \[ r(x_k, u_k) = x_k^T Q x_k + u_k^T R u_k \]

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

Bellman equation \[ V_h(x_k) = r(x_k, h(x_k)) + \gamma V_h(x_{k+1}) \quad , \quad V_h(0) = 0 \]
\[ V_h(x_k) = x_k^T Q x_k + u_k^T R u_k + \gamma V_h(x_{k+1}) \]

Control policy \[ u_k = h(x_k) = \text{the prescribed control input function} \]

Example \[ u_k = -Kx_k \quad \text{Linear state variable feedback} \]
Discrete-Time Optimal Adaptive Control

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

cost: \( V_h(x_k) = \sum_{i=k}^{\infty} \gamma^{i-k} r(x_i, u_i) \)

Bellman eq.:
\[
V_h(x_k) = r(x_k, h(x_k)) + \gamma V_h(x_{k+1})
\]

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

Optimal cost:
\[
V^*(x_k) = \min_h \left( r(x_k, h(x_k)) + \gamma V_h(x_{k+1}) \right)
\]

Bellman’s Opt. Principle
Gives DT HJB equation

Optimal Control:
\[
h^*(x_k) = \arg \min_{u_k} \left( r(x_k, u_k) + \gamma V^*(x_{k+1}) \right)
\]

Focus on these two eqs
**Discrete-Time Optimal Control**

Solutions by Comp. Intelligence Community

Bellman eq

\[
V_h(x_k) = r(x_k, h(x_k)) + \gamma V_h(x_{k+1}), \quad V_h(0) = 0
\]

\[
u_k = h(x_k) = \text{the prescribed control policy}
\]

Policy must be stabilizing

Theorem: Let \( V_h(x_k) \) solve the Bellman equation. Then

\[
V_h(x_k) = \sum_{i=k}^{\infty} \gamma^{i-k} r(x_i, h(x_i))
\]

Gives value for any prescribed control policy

Policy Evaluation for any given current policy

Dynamics Does Not Appear
Optimal Control $h^*(x_k) = \arg \min_{u_k} (r(x_k, u_k) + \gamma V^*(x_{k+1}))$

Bellman’s result

What about? -

$h'(x_k) = \arg \min_{u_k} (r(x_k, u_k) + \gamma V_h(x_{k+1}))$ for a given policy $h(.)$?

Theorem. Bertsekas.
Let $V_h(x_k)$ be the value of any given policy $h(x_k)$.

Then

$V_{h'}(x_k) \leq V_h(x_k)$

Policy Improvement

One step improvement property of Rollout Algorithms
DT Policy Iteration

Cost for any given control policy $h(x_k)$ satisfies the recursion

$$V_h(x_k) = r(x_k, h(x_k)) + \gamma V_h(x_{k+1})$$

Bellman eq.

Recursive solution - Actor/Critic Structure

Pick stabilizing initial control

Policy Evaluation

$$V_{j+1}(x_k) = r(x_k, h_j(x_k)) + \gamma V_{j+1}(x_{k+1})$$

f(.) and g(.) do not appear

Policy Improvement

$$h_{j+1}(x_{k+1}) = \arg\min_{u_k} (r(x_k, u_k) + \gamma V_{j+1}(x_{k+1}))$$

Howard (1960) proved convergence for MDP

Temporal difference

$$e_k = -V_{j+1}(x_k) + r(x_k, h_j(x_k)) + \gamma V_{j+1}(x_{k+1})$$

e.g. Control policy = SVFB

$$h(x_k) = -Lx_k$$
DT Policy Iteration – Linear Systems Quadratic Cost- LQR

\[ x_{k+1} = Ax_k + Bu_k = (A - BL)x_k, \quad u_k = -Lx_k \]

For any stabilizing policy, the cost is

\[ V(x_k) = \sum_{i=k}^{\infty} x_i^T Q x_i + u_i^T R u_i(x_i) \]

LQR value is quadratic \( V(x) = x^T P x \)

DT Policy iterations

\[ V_{j+1}(x_k) = x_k^T Q x_k + u_j^T R u_j(x_k) + V_{j+1}(x_{k+1}) \]

\[ u_{j+1}(x_{k+1}) = -\frac{1}{2} R^{-1} g(x_k)^T \frac{dV_{j+1}(x_{k+1})}{dx_{k+1}} \]

Equivalent to an Underlying Problem- DT LQR:

\[ (A - BL_j)^T P_{j+1}(A - BL_j) - P_{j+1} = -Q - L_j^T R L_j \]

\[ L_{j+1} = (R + B^T P_{j+1} B)^{-1} B^T P_{j+1} A \]

Hewer proved convergence in 1971

Policy Iteration Solves Lyapunov equation WITHOUT knowing System Dynamics
DT Policy Iteration – Linear Systems Quadratic Cost- LQR

DT Policy iterations

\[ V_{j+1}(x_k) = x_k^T Q x_k + u_j^T(x_k) R u_j(x_k) + V_{j+1}(x_{k+1}) \]

\[ u_{j+1}(x_{k+1}) = -\frac{1}{2} R^{-1} g(x_k)^T \frac{dV_{j+1}(x_{k+1})}{dx_{k+1}} \]

How to implement online?
DT Policy Iteration – How to implement online?

Linear Systems Quadratic Cost- LQR

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

\[ V(x_k) = \sum_{i=k}^{\infty} x_i^T Q x_i + u(x_i)Ru(x_i) \]

LQR cost is quadratic \[ V(x) = x^T P x \] for some matrix P

DT Policy iterations

\[ V_{j+1}(x_k) = x_k^T Q x_k + u_j^T (x_k)Ru_j(x_k) + V_{j+1}(x_{k+1}) \]

\[ x_k^T P_{j+1} x_k - x_{k+1}^T P_{j+1} x_{k+1} = x_k^T Q x_k + u_j^T Ru_j \]

\[ = \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} \begin{bmatrix} x_1^2 \\ x_2^2 \end{bmatrix} \begin{bmatrix} x_1^1 \\ x_2^1 \end{bmatrix} - \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} \begin{bmatrix} x_{k+1}^2 \\ x_{k+1}^1 \end{bmatrix} \begin{bmatrix} x_{k+1}^1 \\ x_{k+1}^2 \end{bmatrix} \]

\[ W_{j+1}^T [\phi(x_k) - \phi(x_{k+1})] = x_k^T Q x_k + u_j^T (x_k)Ru_j(x_k) \]

Then update control using

\[ h_j(x_k) = L_j x_k = (R + B^T P_j B)^{-1} B^T P_j Ax_k \]

Need to know A AND B for control update

Quadratic basis set

Solves Lyapunov eq. without knowing A and B
Implementation- DT Policy Iteration
Nonlinear Case

Value Function Approximation (VFA)

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

weights \quad basis functions

LQR case- \( V(x) \) is quadratic

\[ V(x) = x^T Px = W^T \varphi(x) \]

\[ \varphi(x) = [x_1^2, \ldots, x_1 x_n, x_2^2, \ldots, x_2 x_n, \ldots, x_n^2]' \quad \text{Quadratic basis functions} \]

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

Nonlinear system case- use Neural Network
Implementation - DT Policy Iteration

Value function update for given control – Bellman equation

\[ V_{j+1}(x_k) = r(x_k, h_j(x_k)) + \gamma V_{j+1}(x_{k+1}) \]

Assume measurements of \( x_k \) and \( x_{k+1} \) are available to compute \( u_{k+1} \)

\[
V_j(x_k) = W_j^T \varphi(x_k)
\]

Then

\[
W_{j+1}^T [\varphi(x_k) - \gamma \varphi(x_{k+1})] = r(x_k, h_j(x_k))
\]

Indirect Adaptive control with identification of the optimal value

Solve for weights in real-time using RLS
or, batch LS- many trajectories with different initial conditions over a compact set

Then update control using

\[
u_{j+1}(x_{k+1}) = -\frac{1}{2} R^{-1} g(x_k)^T dV_{j+1}(x_{k+1}) = -\frac{1}{2} R^{-1} g(x_k)^T \nabla \varphi^T(x_{k+1}) W_{j+1}^T
\]

Need to know \( g(x_k) \) for control update
Solving the IRL Bellman Equation

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

Solve for value function parameters

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

\[
W_{j+1}^T \left[ \phi(x_k) - \gamma \phi(x_{k+1}) \right] = r(x_k, h_j(x_k))
\]

\[
W_{j+1}^T \left[ \phi(x_{k+1}) - \gamma \phi(x_{k+2}) \right] = r(x_{k+1}, h_j(x_{k+1}))
\]

\[
W_{j+1}^T \left[ \phi(x_{k+2}) - \gamma \phi(x_{k+3}) \right] = r(x_{k+2}, h_j(x_{k+2}))
\]

Now solve by Batch least-squares
Reinforcement Learning (IRL) – A DATA-BASED APPROACH

Solve Bellman Equation - Solves Lyapunov eq. without knowing dynamics

\[ W_{j+1}^T [\varphi(x_k) - \gamma \varphi(x_{k+1})] = r(x_k, h_j(x_k)) \]

Data set at time \([k,k+1)\)
\[ (x_k, r(x_k, h_j(x_k)), x_{k+1}) \]

observe \(x_k\)  
observe \(x_{k+1}\)  
observe \(x_{k+2}\)

apply  
apply  
apply
observe cost  
observe cost  
observe cost
\[ r(x_k, h_j(x_k)) \]
\[ r(x_{k+1}, h_j(x_{k+1})) \]
\[ r(x_{k+2}, h_j(x_{k+2})) \]
update \(W_{j+1}\)
update \(W_{j+1}\)
update \(W_{j+1}\)

Do RLS until convergence to \(W_{j+1}\)

Or use batch least-squares

\[ u_{j+1}(x_{k+1}) = -\frac{1}{2} R^{-1} g(x_k)^T \nabla \varphi^T(x_{k+1}) W_{j+1}^T \]

This is a data-based approach that uses measurements of \(x_k, u_k\) instead of the plant dynamical model.
Continuous control feedback gain with discrete gain updates

Gain update (Policy)

Control

\[ u_k(x_k) = -L_j x_k \]

Update intervals \( j \) may not be the same
They can be selected on-line in real time

Continuous control feedback gain with discrete gain updates
Persistence of Excitation

\[ W_{j+1}^T [\varphi(x_k) - \gamma \varphi(x_{k+1})] = r(x_k, h_j(x_k)) \]

Regression vector must be PE
Adaptive Critics

The Adaptive Critic Architecture

Value update

\[ V_{j+1}(x_k) = r(x_k, h_j(x_k)) + \gamma V_{j+1}(x_{k+1}) \]

Control policy update

\[ h_{j+1}(x_k) = \arg\min_{u_k} (r(x_k, u_k) + \gamma V_{j+1}(x_{k+1})) \]

Action network

System

Policy Evaluation (Critic network)

Cost

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

\[ V(x) = W^T \varphi(x) \]
Simulation Example

- Linear system – Aircraft longitudinal dynamics

\[
A = \begin{bmatrix}
1.0722 & 0.0954 & 0 & -0.0541 & -0.0153 \\
4.1534 & 1.1175 & 0 & -0.8000 & -0.1010 \\
0.1359 & 0.0071 & 1.0 & 0.0039 & 0.0097 \\
0 & 0 & 0 & 1.0 & 0.0039 \\
0 & 0 & 0 & 0 & 1.0
\end{bmatrix}
\]

\[
B = \begin{bmatrix}
0.0075 \\
0.0134 \\
0.8647 \\
0.8647
\end{bmatrix}
\]

Unstable, Two-input system

- The HJB, i.e. ARE, Solution

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

\[
P = \begin{bmatrix}
55.8348 & 7.6670 & 16.0470 & -4.6754 & -0.7265 \\
7.6670 & 2.3168 & 1.4987 & -0.8309 & -0.1215 \\
16.0470 & 1.4987 & 25.3586 & -0.6709 & 0.0464 \\
-4.6754 & -0.8309 & -0.6709 & 1.5394 & 0.0782 \\
-0.7265 & -0.1215 & 0.0464 & 0.0782 & 1.0240
\end{bmatrix}
\]

\[
L = \begin{bmatrix}
4.1136 & -0.7170 & -0.3847 & 0.5277 & 0.0707 \\
-0.6315 & -0.1003 & 0.1236 & 0.0653 & 0.0798
\end{bmatrix}
\]
• Simulation
• The Cost function approximation – quadratic basis set

\[ \hat{V}_{i+1}(x_k, W_{V_{i+1}}) = W_{V_{i+1}}^T \phi(x_k) \]

\[ \phi^T(x) = \begin{bmatrix} x_1^2 & x_1 x_2 & x_1 x_3 & x_1 x_4 & x_1 x_5 & x_2^2 & x_2 x_3 & x_2 x_4 & x_2 x_5 & x_3^2 & x_3 x_4 & x_3 x_5 & x_4^2 & x_4 x_5 & x_5^2 \end{bmatrix} \]

\[ W_{V_{i+1}}^T = \begin{bmatrix} w_{V_{11}} & w_{V_{12}} & w_{V_{13}} & w_{V_{14}} & w_{V_{15}} \end{bmatrix} \]

• The Policy approximation – linear basis set

\[ \hat{u}_i = W_{u_i}^T \sigma(x_k) \]

\[ \sigma^T(x) = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{bmatrix} \]

\[ W_{u_i}^T = \begin{bmatrix} w_{u_{11}} & w_{u_{12}} & w_{u_{13}} & w_{u_{14}} & w_{u_{15}} \\ w_{u_{21}} & w_{u_{22}} & w_{u_{23}} & w_{u_{24}} & w_{u_{25}} \end{bmatrix} \]
• Simulation

The convergence of the value

\[ W^T_v = [55.5411 \ 15.2789 \ 31.3032 \ -9.3255 \ -1.4536 \ 2.3142 \ 2.9234 \ -1.6594 \ -0.2430 \ 24.8262 \ -1.3076 \ 0.0920 \ 1.5388 \ 0.1564 \ 1.0240] \]

\[ T = \begin{bmatrix} 11 & 12 & 13 & 14 & 15 & 1 & 2 & 3 & 4 & 5 \\ 21 & 22 & 23 & 24 & 25 & 2 & 6 & 7 & 8 & 9 \\ 31 & 32 & 33 & 34 & 35 & 3 & 7 & 10 & 11 & 12 \\ 41 & 42 & 43 & 44 & 45 & 4 & 8 & 11 & 13 \\ 51 & 52 & 53 & 54 & 55 \end{bmatrix} \]

\[ V = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 & 0 \end{bmatrix} \]

Actual ARE soln:

\[ P = \begin{bmatrix} 55.8348 & 7.6670 & 16.0470 & -4.6754 & -0.7265 \\ 7.6670 & 2.3168 & 1.4987 & -0.8309 & -0.1215 \\ 16.0470 & 1.4987 & 25.3586 & -0.6709 & 0.0464 \\ -4.6754 & -0.8309 & -0.6709 & 1.5394 & 0.0782 \\ -0.7265 & -0.1215 & 0.0464 & 0.0782 & 1.0240 \end{bmatrix} \]

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

Solves ARE online using real-time data measurements \((x_k, r(x_k, h_j(x_k)), x_{k+1})\)

Without knowing A matrix
Discrete-time nonlinear HJB solution using Approximate dynamic programming : Convergence Proof

- Simulation

The convergence of the control policy

\[ W_u = \begin{bmatrix} 4.1068 & 0.7164 & 0.3756 & -0.5274 & -0.0707 \\ 0.6330 & 0.1005 & -0.1216 & -0.0653 & -0.0798 \end{bmatrix} \]

\[
\begin{bmatrix}
L_{11} & L_{12} & L_{13} & L_{14} & L_{15} \\
L_{21} & L_{22} & L_{23} & L_{24} & L_{25}
\end{bmatrix}
= -
\begin{bmatrix}
w_{u11} & w_{u12} & w_{u13} & w_{u14} & w_{u15} \\
w_{u21} & w_{u22} & w_{u23} & w_{u24} & w_{u25}
\end{bmatrix}
\]

Actual optimal ctrl. \[ L = \begin{bmatrix} -4.1136 & -0.7170 & -0.3847 & 0.5277 & 0.0707 \\
-0.6315 & -0.1003 & 0.1236 & 0.0653 & 0.0798 \end{bmatrix} \]

Note- In this example, drift dynamics matrix A is NOT Needed. Riccati equation solved online without knowing A matrix.
Greedy Value Fn. Update - Approximate Dynamic Programming

Value Iteration = Heuristic Dynamic Programming (HDP)

Paul Werbos

Policy Iteration

\[ V_{j+1}(x_k) = r(x_k, h_j(x_k)) + \gamma V_{j+1}(x_{k+1}) \]

\[ h_{j+1}(x_k) = \arg \min_{u_k} (r(x_k, u_k) + \gamma V_{j+1}(x_{k+1})) \]

For LQR

\[ (A - BL_j)^T P_{j+1} (A - BL_j) - P_{j+1} = -Q - L_j^T R L_j \]

Hewer 1971

Underlying RE

\[ L_j = -(R + B^T P_j B)^{-1} B^T P_j A \]

Initial stabilizing control is needed

Value Iteration

Two occurrences of cost allows def. of greedy update

\[ V_{j+1}(x_k) = r(x_k, h_j(x_k)) + \gamma V_{j+1}(x_{k+1}) \]

\[ h_{j+1}(x_k) = \arg \min_{u_k} (r(x_k, u_k) + \gamma V_{j+1}(x_{k+1})) \]

For LQR

\[ P_{j+1} = (A - BL_j)^T P_j (A - BL_j) + Q + L_j^T R L_j \]

Lancaster & Rodman proved convergence

\[ L_j = -(R + B^T P_j B)^{-1} B^T P_j A \]

Initial stabilizing control is NOT needed
Motivation for Value Iteration

PI Policy Evaluation Step

\[ V_{j+1}(x_k) = r(x_k, h(x_k)) + \gamma V_{j+1}(x_{k+1}) \]

Needs stabilizing gain

\[ (A - BL_j)^T P_{j+1}(A - BL_j) - P_{j+1} = -Q - \dot{L}_j^T RL_j \]

LE= Lyapunov equation

VI Policy Evaluation Step

\[ V_{j+1}(x_k) = r(x_k, h(x_k)) + \gamma V_j(x_{k+1}) \]

Does not need stabilizing gain

\[ P_{j+1} = (A - BL_j)P_i (A - BL_j) + Q + \dot{L}_j^T RL_j \]

MR= Matrix recursion

Theorem

Let gain L be fixed and (A-BL) stable.
Let \( P_0 \geq 0 \) in MR

Then \( P_i \rightarrow P_{j+1} \)

i.e. repeated application of the VI policy evaluation step
is the same as one application of the PI policy evaluation step
IF THE CONTROL POLICY IS NOT UPDATED

Idea of GPI
Implementation - DT HDP – Value Iteration

Policy Iteration was

\[ V_{j+1}(x_k) = r(x_k, h_j(x_k)) + \gamma V_{j+1}(x_{k+1}) \]

Value function update for given control

\[ V_{j+1}(x_k) = r(x_k, h_j(x_k)) + \gamma V_j(x_{k+1}) \]

Assume measurements of \( x_k \) and \( x_{k+1} \) are available to compute \( u_{k+1} \)

\[ V_j(x_k) = W_j^T \varphi(x_k) \]

VFA

Then

\[ W_{j+1}^T [\varphi(x_k)] = r(x_k, h_j(x_k)) + \gamma W_j^T [\varphi(x_{k+1})] \]

Solve for weights using RLS

or, many trajectories with different initial conditions over a compact set

Then update control using

\[ h_j(x_k) = L_j x_k = -(R + B^T P_j B)^{-1} B^T P_j A x_k \]

Need to know \( f(x_k) \) AND \( g(x_k) \) for control update
Example 11.3-4: Policy Iteration and Value Iteration for the DT LQR

1. Policy Iteration: Hewer’s Algorithm

Value Update

\[ V^{j+1}(x_k) = \frac{1}{2} \left( x_k^T Q x_k + u_k^T R u_k \right) + V^{j+1}(x_{k+1}) , \]

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

\[ 0 = (A - BK^j)^T P^{j+1}(A - BK^j) - P^{j+1} + Q + (K^j)^T R K^j . \]

Policy Improvement

\[ \mu^{j+1}(x_k) = K^{j+1} x_k = \arg \min (x_k^T Q x_k + u_k^T R u_k + x_{k+1}^T P^{j+1} x_{k+1}) , \]

\[ K^{j+1} = -(B^T P^{j+1} B + R)^{-1} B^T P^{j+1} A . \]

2. Value Iteration: Lyapunov recursions

\[ V_{j+1}(x) = \sum_u \pi_j(x,u) \sum_{x'} P_{xx'}^{u} \left[ R_{xx'} + \gamma V_j(x') \right] \]

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

\[ P^{j+1} = (A - BK^j)^T P^j(A - BK^j) + Q + (K^j)^T R K^j . \]

3. Iterative Policy Evaluation

\[ P^{j+1} = (A - BK)^T P^j(A - BK) + Q + K^T R K . \]

this recursion converges to the solution of the Lyapunov equation

All algorithms have a model-free scalar form based on measuring states, and a model-based matrix form based on cancelling states
Adaptive (Approximate) Dynamic Programming

Four ADP Methods proposed by Paul Werbos

Critic NN to approximate:

Heuristic dynamic programming

Value Iteration

Value \( V(x_k) \)

Dual heuristic programming

Gradient \( \frac{\partial V}{\partial x} \)

AD Heuristic dynamic programming

(Watkins Q Learning)

Q function \( Q(x_k, u_k) \)

AD Dual heuristic programming

Gradients \( \frac{\partial Q}{\partial x}, \frac{\partial Q}{\partial u} \)

Action NN to approximate the Control

Bertsekas- Neurodynamic Programming

Barto & Bradtke- Q-learning proof (Imposed a settling time)
A problem with DT Policy Iteration and VI

Policy Evaluation

Assume measurements of $x_k$ and $x_{k+1}$ are available to compute $u_{k+1}$

$$V_j(x_k) = W_j^T \phi(x_k)$$

Then

$$W_{j+1}^T [\phi(x_k) - \gamma \phi(x_{k+1})] = r(x_k, h_j(x_k))$$

Since $x_{k+1}$ is measured, do not need knowledge of $f(x)$ or $g(x)$ for value fn. update

Policy Improvement

$$u_{j+1}(x_{k+1}) = -\frac{1}{2} R^{-1} g(x_k)^T \frac{dV_{j+1}(x_{k+1})}{dx_{k+1}}$$

LQR case

$$h_j(x_k) = L_j x_k = (R + B^T P_j B)^{-1} B^T P_j A x_k$$

Need to know $f(x_k)$ AND $g(x_k)$ for control update

Easy to fix – use 2 NN
Standard Neural Network VFA for On-Line Implementation

**NN for Value - Critic**
\[ \hat{V}_i(x_k, W_{V_i}) = W_{V_i}^T \phi(x_k) \]

**NN for control action**
\[ \hat{u}_i(x_k, W_{ui}) = W_{ui}^T \sigma(x_k) \]

**HDP**
\[ V_{i+1}(x_k) = x_k^T Q x_k + u^T R u + V_i(x_{k+1}) \]
\[ u_i(x_k) = \arg\min_u (x_k^T Q x_k + u^T R u + V_i(x_{k+1})) \]
\[ x_{k+1} = f(x_k) + g(x_k)u(x_k) \]

**Define target cost function**
\[ d(\phi(x_k), W_{V_i}^T) = x_k^T Q x_k + \hat{u}_i^T (x_k)R\hat{u}_i (x_k) + \hat{V}_i (x_{k+1}) \]
\[ = x_k^T Q x_k + \hat{u}_i^T (x_k)R\hat{u}_i (x_k) + W_{V_i}^T \phi(x_{k+1}) \]

**Explicit equation for cost – use LS for Critic NN update or RLS**
\[ W_{V_i+1} = \arg\min_{W_{V_i+1}} \left\{ \int_\Omega \left| W_{V_i+1}^T \phi(x_k) - d(\phi(x_k), W_{V_i}^T) \right|^2 \ dx_k \right\} \]
\[ W_{V_i+1} = \left( \int_\Omega \phi(x_k) \phi(x_k)^T \ dx \right)^{-1} \left( \int_\Omega \phi(x_k) d^T (\phi(x_k), W_{V_i}^T, W_{ui}^T) \ dx \right) \]
\[ \text{OR} \quad W_{V_i+1|m+1} = W_{V_i+1|m} + \beta \phi^T (x_k) \left( -W_{V_i+1|m} \phi(x_k) + r(x_k, u_k) + W_{V_i}^T \phi(x_{k+1}) \right) \]

**Implicit equation for DT control- use gradient descent for action update**
\[ W_{ui} = \arg\min_{W} \left( x_k^T Q x_k + \hat{u}_i^T (x_k, W) R\hat{u}_i (x_k, W) + \hat{V}_i (f(x_k) + g(x_k)\hat{u}_i (x_k, W)) \right) \]
\[ W_{ui(j+1)} = W_{ui(j)} - \alpha \frac{\partial (x_k^T Q x_k + \hat{u}_i^T (x_k)R\hat{u}_i (x_k) + \hat{V}_i (x_{k+1}))}{\partial W_{ui(j)}} \]
\[ W_{ui}^{j+1} = W_{ui}^j - \alpha \sigma(x_k)(2R\hat{u}_i (x_k) + g(x_k)\frac{\partial \phi^T (x_{k+1})}{\partial x_{k+1}} W_{V_i})^T \]

Backpropagation- P. Werbos
Oscillation is a fundamental property of neural tissue

Brain has multiple adaptive clocks with different timescales

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

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

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

Spinal cord

Motor control 200 Hz


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.
Summary of Motor Control in the Human Nervous System

- Cerebral cortex
- Motor areas
- Thalamus
- Basal ganglia
- Cerebellum
- Brainstem
- Spinal cord
- Exteroceptive receptors
- Interoceptive receptors
- Muscle contraction and movement

Memory functions:
- Long term
- Short term

Learning:
- Reinforcement Learning: dopamine
- Supervised learning
- Unsupervised learning

Rhythms:
- gamma rhythms 30-100 Hz
- theta rhythms 4-10 Hz

Hierarchy of multiple parallel loops

Motor control 200 Hz
Adaptive Actor-Critic structure

Reinforcement learning

Theta waves 4-8 Hz

Motor control 200 Hz

Paul Werbos
Q Learning
Adaptive (Approximate) Dynamic Programming

Four ADP Methods proposed by Paul Werbos

Critic NN to approximate:

<table>
<thead>
<tr>
<th>Heuristic dynamic programming</th>
<th>AD Heuristic dynamic programming (Watkins Q Learning)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value Iteration</td>
<td></td>
</tr>
<tr>
<td>Value $V(x_k)$</td>
<td>$Q(x_k, u_k)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dual heuristic programming</th>
<th>AD Dual heuristic programming</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gradient $\frac{\partial V}{\partial x}$</td>
<td>$\frac{\partial Q}{\partial x}, \frac{\partial Q}{\partial u}$</td>
</tr>
</tbody>
</table>

Action NN to approximate the Control

Bertsekas- Neurodynamic Programming
Barto & Bradtke- Q-learning proof (Imposed a settling time)
Q Learning - Action Dependent ADP

Value function recursion for given policy \( h(x_k) \)

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

Define Q function

\[
Q_h(x_k, u_k) = r(x_k, u_k) + \gamma V_h(x_{k+1})
\]

Note

\[
Q_h(x_k, h(x_k)) = V_h(x_k)
\]

Recursion for Q

\[
Q_h(x_k, u_k) = r(x_k, u_k) + \gamma Q_h(x_{k+1}, h(x_{k+1}))
\]

Simple expression of Bellman’s principle

\[
V^*(x_k) = \min_{u_k}(Q^*(x_k, u_k)) \quad h^*(x_k) = \arg\min_{u_k}(Q^*(x_k, u_k))
\]

Optimal Adaptive Control for completely unknown DT systems
Q Function Definition

Specify a control policy \( u_j = h(x_j) \); \( j = k, k+1, \ldots \).

Define Q function
\[
Q_h(x_k, u_k) = r(x_k, u_k) + \gamma V_h(x_{k+1})
\]
\( u_k \) arbitrary
\( \text{policy } h(.) \) used after time \( k \)

Note
\[
Q_h(x_k, h(x_k)) = V_h(x_k)
\]

Bellman equation for Q
\[
Q_h(x_k, u_k) = r(x_k, u_k) + \gamma Q_h(x_{k+1}, h(x_{k+1}))
\]

Optimal Q function
\[
Q^*(x_k, u_k) = r(x_k, u_k) + \gamma V^*(x_{k+1})
\]
\[
Q^*(x_k, u_k) = r(x_k, u_k) + \gamma Q^*(x_{k+1}, h^*(x_{k+1}))
\]

Optimal control solution
\[
V^*(x_k) = Q^*(x_k, h^*(x_k)) = \min_h Q_h(x_k, h(x_k))
\]
\[
h^*(x_k) = \arg \min_h Q_h(x_k, h(x_k))
\]

Simple expression of Bellman’s principle
\[
V^*(x_k) = \min_{u_k} Q^*(x_k, u_k)
\]
\[
h^*(x_k) = \arg \min_{u_k} Q^*(x_k, u_k)
\]
Q Function HDP – Action Dependent HDP

Q function for any given control policy \( h(x_k) \) satisfies the Bellman equation

\[
Q_h(x_k, u_k) = r(x_k, u_k) + \gamma Q_h(x_{k+1}, h(x_{k+1}))
\]

Recursive solution

Pick stabilizing initial control policy

Find Q function

\[
Q_{j+1}(x_k, u_k) = r(x_k, u_k) + \gamma Q_j(x_{k+1}, h_j(x_{k+1}))
\]

Update control

\[
h_{j+1}(x_k) = \arg\min_{u_k} (Q_{j+1}(x_k, u_k))
\]

Now \( f(x_k, u_k) \) not needed

Bradtke & Barto (1994) proved convergence for LQR
Q Learning does not need to know f(x_k) or g(x_k)

For LQR
\[ V(x) = W^T \varphi(x) = x^T Px \]

\[ Q_h(x_k, u_k) = r(x_k, u_k) + V_h(x_{k+1}) \]
\[ = x_k^T Q x_k + u_k^T R u_k + (Ax_k + Bu_k)^T P (Ax_k + Bu_k) \]
\[ = \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} Q + A^T PA & A^T PB \\ B^T PA & R + B^T PB \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} \equiv \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T H \begin{bmatrix} x_k \\ u_k \end{bmatrix} = \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} H_{xx} & H_{xu} \\ H_{ux} & H_{uu} \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} \]

Q is quadratic in x and u

Control update is found by
\[ 0 = \frac{\partial Q}{\partial u_k} = 2 [B^T P A x_k + (R + B^T P B) u_k] = 2 [H_{ux} x_k + H_{uu} u_k] \]
\[ \text{so} \quad u_k = -(R + B^T P B)^{-1} B^T P A x_k = -H_{uu}^{-1} H_{ux} x_k = L_{j+1} x_k \]

Control found only from Q function
A and B not needed
Q function update for control $u_k = L_j x_k$ is given by

$$Q_{j+1}(x_k, u_k) = r(x_k, u_k) + \gamma Q_{j+1}(x_{k+1}, L_j x_{k+1})$$

Assume measurements of $u_k$, $x_k$ and $x_{k+1}$ are available to compute $u_{k+1}$

QFA – Q Fn. Approximation

$$Q(x, u) = W^T \varphi(x, u)$$

Now $u$ is an input to the NN- Werbos- Action dependent NN

Then

$$W_{j+1}^T [\varphi(x_k, u_k) - \gamma \varphi(x_{k+1}, L_j x_{k+1})] = r(x_k, L_j x_k)$$

Solve for weights using RLS or backprop.

For LQR case

$$\varphi(x) = \left[x_1^2, \ldots, x_1 x_n, x_2^2, \ldots, x_2 x_n, \ldots, x_n^2\right]'$$
Model-free policy iteration

Q Policy Iteration

\[ Q_{j+1}(x_k, u_k) = r(x_k, u_k) + \gamma Q_{j+1}(x_{k+1}, L_j x_{k+1}) \]

Bradtke, Ydstie, Barto

\[ W_{j+1}^T \left[ \phi(x_k, u_k) - \gamma \phi(x_{k+1}, L_j x_{k+1}) \right] = r(x_k, L_j x_k) \]

Control policy update

Stable initial control needed

\[ h_{j+1}(x_k) = \arg \min_{u_k} Q_{j+1}(x_k, u_k) \]

\[ u_k = -H_{uu}^{-1} H_{ux} x_k = L_{j+1} x_k \]

Greedy Q Fn. Update - Approximate Dynamic Programming

ADP Method 3. Q Learning

Action-Dependent Heuristic Dynamic Programming (ADHDP)

Greedy Q Update

Model-free HDP

Paul Werbos

Stable initial control NOT needed

\[ Q_{j+1}(x_k, u_k) = r(x_k, u_k) + \gamma Q_j(x_{k+1}, h_j(x_{k+1})) \]

\[ W_{j+1}^T \phi(x_k, u_k) = r(x_k, L_j x_k) + W_{j}^T \gamma \phi(x_{k+1}, L_j x_{k+1}) \equiv \text{target}_{j+1} \]

Update weights by RLS or backprop.
Q learning actually solves the Riccati Equation WITHOUT knowing the plant dynamics

Model-free ADP

Direct OPTIMAL ADAPTIVE CONTROL

Works for Nonlinear Systems

Proofs?
Robustness?
Comparison with adaptive control methods?
ADHDP Application for Power system

- System Description

\[
x(t) = \begin{bmatrix} \Delta f(t) & \Delta P_g(t) & \Delta X_g(t) & \Delta F(t) \end{bmatrix}^T
\]

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

\[
B^T = \begin{bmatrix} 0 & 0 & 1/T_G & 0 \end{bmatrix}
\]

\[
E^T = \begin{bmatrix} 1 - K_p/T_p & 0 & 0 & 0 \end{bmatrix}
\]

- The Discrete-time Model is obtained by applying ZOH to the CT model.

\[
1/T_p \in [0.033, 0.1] \\
K_p/T_p \in [4,12] \\
1/T_T \in [2.564, 4.762] \\
1/T_G \in [9.615, 17.857] \\
1/RT_G \in [3.081, 10.639]
\]
ADHDP Application for Power system

• The system state
  \( \Delta f \) _incremental frequency deviation (Hz)
  \( \Delta P_g \) _incremental change in generator output (p.u. MW)
  \( \Delta X_g \) _incremental change in governor position (p.u. MW)
  \( \Delta F \) _incremental change in integral control.
  \( \Delta P_d \) _is the load disturbance (p.u. MW); and

• The system parameters are:
  \( T_G \) _the governor time constant
  - \( T_T \) _turbine time constant
  - \( T_P \) _plant model time constant
  - \( K_p \) _planet model gain
  - \( R \) _speed regulation due to governor action
  - \( K_E \) _integral control gain.
ADHDP Application for Power system

- ADHDP policy tuning

The convergence of $P_{11}$, $P_{12}$, $P_{13}$, $P_{22}$, $P_{23}$, $P_{33}$, $P_{34}$, and $P_{44}$ converges to Optimal GARE solution and optimal game control.
ADHDP Application for Power system

- Comparison

The ADHDP controller design

The design from [1]

- The maximum frequency deviation when using the ADHDP controller is improved by 19.3% from the controller designed in [1]
