Optimal Adaptive Control of Nonlinear Discrete-time Systems

S. Jagannathan
Department of Electrical and Computer Engineering
Missouri University of Science and Technology
Rolla, MO, USA
(sarangap@mst.edu)
Outline

- Introduction
- Background on Optimal Adaptive Control
- Optimal Adaptive Control of Unknown Dynamic Systems
- Challenges of Value and/or Policy Iteration-based Schemes
- Online Optimal Adaptive Control w/o Policy and/or Value Iterations
- Online Optimal Tracking
- Hardware Implementation
- Extensions to Continuous-time
- Conclusions
Reinforcement learning (RL) / Adaptive dynamic programming (ADP) techniques with online approximators such as neural networks (NN’s) have been widely employed in the literature in the context of optimal adaptive control (Si, Barto, Powell & Wunsch, 2004)

- Policy and/or value iterations are utilized
- NN reconstruction errors are ignored
- Convergence of the NN implementations are not provided
- Full or partial knowledge of the system dynamics are needed and in some cases a model of the system is needed

The Bellman equation forms the basis for a family of RL/ADP schemes for finding optimal adaptive policies forward in time

RL/ADP based schemes do not require the system dynamics

- Convergence and/or closed-loop stability can be demonstrated by using temporal difference error (TD) method

RL/ADP-based online methods are preferred over offline-based schemes
Background

Consider discrete-time (DT) nonlinear affine system given by
\[ x_{k+1} = f(x_k) + g(x_k)u_k, \] (1)
where \( x_k \in \mathbb{R}^n \) is the system state and \( u_k \in \mathbb{R}^m \) is the control inputs.

Value function
\[ V^h(x_k) = \sum_{i=k}^{\infty} \gamma^{i-k} r(x_i, u_i) = \sum_{i=k}^{\infty} \gamma^{i-k} \left( Q(x_i) + u_i^T R u_i \right), \] (2)
with \( 0 < \gamma \leq 1 \) is a discount factor, \( Q(x_k) \geq 0, R > 0, \) \( u_k = h(x_k) \) is a feedback control input, and stage cost is \( r(x_k, u_k) = Q(x_k) + u_k^T R u_k. \)

Note: Discounting makes initial time steps more important and traditional learning technique usually learn during first few time steps, and hence discounting is not preferred for learning schemes. Value function is the positive definite solution to the Bellman equation that satisfies \( V(0) = 0. \)
Optimal Control for DT System

- Discrete-time Hamilton-Jacobi-Bellman (HJB) equation
  \[ V^*(x_k) = \min_{h(.)} \left( r(x_k, h(x_k)) + \gamma V^*(x_{k+1}) \right). \]  
  \( (3) \)

- Optimal control policy:
  \[ h^*(x_k) = \arg\min_{h(.)} \left( r(x_k, h(x_k)) + \gamma V^*(x_{k+1}) \right) \]
  \[ = -\frac{\gamma}{2} R^{-1} g^T(x_k) \nabla V^*(x_{k+1}) \]  
  \( (4) \)

**Note:** For solving optimal control \( h^*(x_k) \), DT HJB equation \( (3) \) needs to be solved. However, due to nonlinear nature of DT HJB equation, finding its solution is difficult or impossible.
Hamiltonian

- Discrete-time Hamiltonian

\[ H(x_k, h(x_K), \Delta V_k) = r(x_k, h(x_k)) + \gamma V^h(x_{k+1}) - V^h(x_k), \]

where \( \Delta V_k = \gamma V^h(x_{k+1}) - V^h(x_k) \) is the forward difference operator. It is important to note Hamiltonian is the temporal difference (TD) error for prescribed control policy \( u_k = h(x_k) \).

- Iterative based methods will be shown to obtain the value function and optimal control from the DT HJB.
Traditional Optimal Control of Linear Systems

- **Linear Quadratic Regulator**
  - DT linear time-invariant system dynamics:
    \[ x_{k+1} = Ax_k + Bu_k, \] (6)
  - Infinite horizon value function:
    \[ V(x_k) = \sum_{i=k}^{\infty} (x_i^T Q x_i + u_i^T R u_i) \] (7)
  - Bellman equation:
    \[ V(x_k) = (x_k^T Q x_k + u_k^T R u_k) + V(x_{k+1}) \] (8)
Bellman Equation for DT LQR: ARE

- Value function in quadratic form of state: $V(x_k) = x_k^T P x_k$ with some $P$.
- DT Hamiltonian:
  \[ H(x_k, u_k) = x_k^T Q x_k + u_k^T R u_k + (A x_k + B u_k)^T P (A x_k + B u_k) - x_k^T P x_k. \]  
  \[ (9) \]
- Use stationarity condition $\frac{\partial H(x_k, u_k)}{\partial u_k} = 0$ to obtain optimal control
  \[ u_k = -K x_k = -(B^T P B + R)^{-1} B^T P A x_k. \]  
  \[ (10) \]

Substitute (10) into (9) to obtain DT Algebraic Riccati Equation (DARE)
\[ A^T P A - P + Q - A^T P B (B^T P B + R)^{-1} B^T P A = 0. \]  
\[ (11) \]
DT ARE is exactly the Bellman optimality equation for the DT LQR.
**Value Function Approximation (VFA)**

- VFA is the key for implementing PI and VI online for dynamical systems with infinite state and action spaces.

- **Linear system:**
  Since value function is quadratic of state in LQR case, it can be
  \[
  V(x_k) = \frac{1}{2} x_k^T P x_k = \frac{1}{2} (\text{vec}(P))^T (x_k \otimes x_k) \equiv \bar{p}^T \bar{x}_k \equiv \bar{p}^T \phi(x_k), \quad (12)
  \]

- **Nonlinear system:**
  According to Weierstrass Higher-Order Approximation Theorem, there exists a dense basis set \{\varphi_i(x)\} such that
  \[
  V(x) = \sum_{i=1}^{\infty} w_i \varphi_i(x) = \sum_{i=1}^{L} w_i \varphi_i(x) + \sum_{i=L+1}^{\infty} w_i \varphi_i(x) \equiv W^T \phi(x) + \varepsilon_L(x), \quad (13)
  \]
  with basis vector \(\phi(x) = [\varphi_1(x) \quad \varphi_2(x) \quad \cdots \quad \varphi_L(x)] : R^n \rightarrow R^L\) and \(\varepsilon_L(x)\) converges uniformly to zero when \(L \rightarrow \infty\).
Example: Solving DT LQR online by using PI and VI

- System dynamics:
  \[ x_{k+1} = Ax_k + Bu_k , \]  
  \[ (14) \]

- Value function:
  \[ V(x_k) = \sum_{i=k}^{\infty} (x_i^T Q x_i + u_i^T R u_i) = x_k^T P x_k \]  
  \[ (15) \]

- Optimal control:
  \[ u_k = -K x_k = -(B^T P B + R)^{-1} B^T P A x_k . \]  
  \[ (16) \]

**Note:** Solving optimal control needs \( P \) which is the solution of Riccati equation. Next, PI and VI will be used to solve \( P \) online.
Optimal Adaptive Control via PI

- PI for DT LQR:
  - Initialization: select any admissible control $h_0(x_k)$.
  - Start iteration $j = 0$ until convergence
    - Policy evaluation step:
      \[
      \bar{p}_{j+1}^T (\bar{x}_{k+1} - \bar{x}_k) = r(x_k, h_j(k)) = x_k^T (Q + K_j^T R K_j) x_k, \tag{17}
      \]
      where $\bar{p}_{j+1} = vec(P^{j+1})$
    - Policy improvement step:
      \[
      u_{k}^{j+1} = h_{j+1}(x_k) = -(B^T P^{j+1} B + R)^{-1} B^T P^{j+1} A x_k \tag{18}
      \]
Optimal Adaptive Control via VI

- VI for DT LQR:
  - Initialization: select any control \( h_0(x_k) \).
  - Start iteration \( j = 0 \) until convergence
    - Value update step:
      \[
      p_{j+1}^T \bar{x}_{k+1} = r(x_k, h_j(k)) + p_j^T \bar{x}_k = x_k^T (Q + K_j R K_j) x_k + p_j^T \bar{x}_k ,
      \]
      where \( p_{j+1} = vec(P^{j+1}) \)
    - Control policy update:
      \[
      u_{k+1} = h_{j+1}(x_k) = -(B^T P^{j+1} B + R)^{-1} B^T P^{j+1} A x_k
      \]

Note: The control policy update (18),(20) requires system dynamics \( A, B \) which cannot be known beforehand.
Policy (PI) and Value Iteration (VI) Flowchart

**Start Policy Iteration using TD learning**

**Initialization**
Admissible control \( u^0 = h_0(x_0) \)

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

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

or equivalently,
\[ h_{j+1}(x_k) = -\frac{1}{\gamma} R^T g^T (x_k) \nabla V_{j+1}(x_{k+1}) \]

**Update time interval**
\[ j \rightarrow j + 1 \]

\[ |V_{j+1} - V_j|_{F} < \epsilon \]

**Start Value Iteration using TD learning**

\[ u^0 = h_0(x_0) \]

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

**Control policy update**
\[ h_{j+1}(x_k) = \arg \min_{h_j(x_k)} (r(x_k, h_j(x_k)) + \gamma V_{j+1}(x_{k+1})) \]

or equivalently,
\[ h_{j+1}(x_k) = -\frac{1}{\gamma} R^T g^T (x_k) \nabla V_{j+1}(x_{k+1}) \]

**Update time interval**
\[ j \rightarrow j + 1 \]

\[ |V_{j+1} - V_j|_{F} < \epsilon \]

**Note:** Difference between PI and VI: 1) VI does not require initial admissible control; 2) Value update is different with policy evaluation.
Second ‘Actor’ Neural Network:
For relaxing the requirement of full system dynamics knowledge, a second neural network is introduced for control policy.

- **Actor NN**
  
  \[ u_k = h(x_k) = U^T \sigma(x_k), \]  
  \[ (21) \]

  with \( \sigma(x): R^n \rightarrow R^m \) is activation or basis function and \( U \in R^{M \times m} \) is weights or unknown parameter.

- Update law is required for the NN weights

**Note:** After convergence of critic NN parameters to \( W_{j+1} \) in PI or VI, actor NN is used to update the control policy based on \( W_{j+1} \). Actor NN avoids the need of state drift dynamics \( f(\bullet) \) (or \( A \) in linear system).
Fig 1. Temporal Difference Learning using Policy Iteration.

The value function is estimated by observing the current and the next state, and the cost incurred. Based on the new value, the action is updated.
Nonlinear Optimal Control via Offline

- Discrete-time unknown nonlinear affine system
  \[ x_{k+1} = f(x_k) + g(x_k)u(x_k) = f_k + g_ku_k \]  
  (1)

- Infinite horizon cost function
  \[ V(x_k) = \sum_{n=k}^{\infty} Q(x_n) + u_n^T R u_n \]
  \[ = Q(x_k) + u_k^T R u_k + V(x_{k+1}) \]  
  (2)

where \( Q(x_k) \geq 0 \), \( R > 0 \in \mathbb{R}^{m \times m} \).

- The control input \( u_k \) must be admissible (Chen & Jagannathan, 2008):
  - \( u_k \) is continuous on a compact set;
  - \( u_k \) stabilizes (1) and \( u_k|_{x_k=0} = 0 \);
  - \( V(x_0) \) is finite for all \( x_0 \in \Omega \).

The optimal control input for (1) is found by solving
$$\frac{\partial V(x_k)}{\partial u_k} = 0$$
and revealed to be
$$u^*_k = -\frac{1}{2} R^{-1} g(x_k)^T \frac{\partial V(x_{k+1})}{\partial x_{k+1}}$$  \hspace{1cm} (3)

- Optimal control (3) is generally not implementable for unknown nonlinear systems due to $g(x_k)$ and $x_{k+1}$

- This work relaxes the need for explicit knowledge of (1) while ensuring convergence and stability
  - Online NN system identification scheme
  - Offline optimal control using only the learned NN model of the system
Identifier Design

- The nonlinear system (1) is rewritten as

\[
x_{k+1} = f_k + g_k u_k = h(x_k, u_k) \\
= w_s^T \phi(v_s^T z_k) + \varepsilon(x_k) = w_s^T \phi(\tilde{z}(x_k)) + \varepsilon_s(x_k)
\]

(4)

- The proposed nonlinear identifier is written as

\[
\hat{x}_{k+1} = \hat{w}_{s,k}^T \phi(\tilde{z}_k) - v_k
\]

(5)

where

\[
v_k = \frac{\hat{\lambda}_k \tilde{x}_k}{\tilde{x}_k^T \tilde{x}_k + C_s}
\]

(6)

and \( \tilde{x}_k = x_k - \hat{x}_k \) is the identification error, \( \hat{\lambda}_k \in \mathbb{R} \) is an additional tunable parameter, and \( C_s > 0 \) is a constant
**Assumption:** The term $\varepsilon_s(x_k)$ is assumed to be upper bounded by a function of estimation error such that

$$\varepsilon_s^T(x_k)\varepsilon_s(x_k) \leq \varepsilon_M(x_k) = \lambda^*\tilde{x}_k^T\tilde{x}_k$$  \hspace{1cm} (7)

where $\|\lambda^*\| \leq \lambda_M$

The proposed tuning laws for $\hat{w}_{s,k}$ and $\hat{\lambda}_k$ are given by

$$\hat{w}_{s,k+1} = \hat{w}_{s,k} + \alpha_s \phi(\bar{z}_k)\tilde{x}_{k+1}^T$$ \hspace{1cm} (8)

and

$$\hat{\lambda}_k = \hat{\lambda}_k - \gamma_s \tilde{x}_k^T\tilde{x}_k / (\tilde{x}_k^T\tilde{x}_k + C_s)$$ \hspace{1cm} (9)

The asymptotical error convergence can be demonstrated using

$$L_k = \tilde{x}_k^T\tilde{x}_k + tr[\hat{w}_{s,k}^T\hat{w}_{s,k}] / \alpha_s + \hat{\lambda}_k^2 / \gamma_s$$ \hspace{1cm} (10)
Estimation of Control Coefficient Matrix

- Using the online system identification scheme (5), an estimate for $g(x_k)$ is given by
  \[ g(x_k) = \hat{w}^T_s \phi'(\bar{z}_k) v_s^T (\partial z_k / \partial u_k) \]  
  (11)
where $\phi'(<\bar{z}_k>) \in \mathbb{R}^{l \times l}$ is the derivative of the activation function with respect to $\bar{z}_k$ and $\partial z_k / \partial u_k$ is a constant matrix.

- After a sufficiently long online learning session, the robust term (6) approaches to zero, and the dynamic system can be rewritten as
  \[ \hat{x}_{k+1} = f(x_k) + g(x_k)u_k = \hat{x}_{k+1} = \hat{w}^T_{s,k} \phi(\bar{z}_k) \]  
  (12)

- Then observe $\partial x_{k+1} / \partial u_k = g(x_k) = \partial \hat{x}_{k+1} / \partial u_k$ to yield the relationship shown in (11)
Offline Heuristic Dynamic Programming

- Only the NN model learned online is used for the offline training

1) Initialize the cost function and select the initial stabilizing control input:

\[ V_0(x_k) = 0, \quad u_0(\hat{x}_k) = \arg\min_u \left( Q(x_k) + u_k^T R u_k + V_0(\hat{x}_{k+1}) \right) \]

2) Compute the value function

\[
V_1(\hat{x}_k) = Q(x_k) + u_0^T Ru_0 + V_0(\hat{x}_{k+1})
\]

\[
= Q(x_k) + u_0^T Ru_0 + V_0(\hat{x}_{k+1})
\]

3) Optimization is achieved by iterating between a sequence of control policies and value functions (Proof of converge shown in Al-Tamimi, Lewis, & Abu-Khalaf, 2008)

\[
u_i(x_k) = \arg\min_u \left( Q(x_k) + u_k^T R u_k + V_i(\hat{x}_{k+1}) \right)\]

\[V_{i+1}(x_k) = Q(x_k) + u_i^T Ru_i + V_i(\hat{x}_{k+1})\]
NN Implementation of Offline HDP

- Value function (2) and optimal control (3) are assumed to have NN representations as

\[ V_i(x_k) = W_{Vi}^T \sigma(x_k) + \varepsilon_{Vi} \]  \hspace{1cm} (13)
\[ u_i(x_k) = W_{Ai}^T \Theta(x_k) + \varepsilon_{Ai} \]  \hspace{1cm} (14)

- The NN approximations of (13) and (14) are written as

\[ \hat{V}_i(x_k) = \hat{W}_{Vi}^T \sigma(x_k) = \hat{W}_{Vi}^T \sigma_k \]  \hspace{1cm} (15)
\[ \hat{u}_i(x_k) = \hat{W}_{Ai}^T \Theta(x_k) = \hat{W}_{Ai}^T \Theta_k \]  \hspace{1cm} (16)

- Critic weights are tuned at each iteration using method of weighted residuals

\[ \hat{W}_{Vi+1} = \left( \int_{\Omega} \sigma(x_k) \sigma^T(x_k) dx \right)^{-1} \times \int_{\Omega} \sigma(x_k) d_i^T(x_k), \hat{x}_{k+1}, W_{Vi}, W_{Ai} dx \]  \hspace{1cm} (17)
Tuning the Action Network

- Define the action error to be the difference between the estimated control input (14) and the estimated optimal control

\[ e_{ai,j} = \hat{u}_{i,k} - \hat{u}_{i,k}^* = \hat{W}^T_{Ai,j} \vartheta(x_k) + \frac{1}{2} R^{-1} g^T(x_k) \frac{\partial \hat{V}_i(\hat{x}_{j,k+1})}{\partial \hat{x}_{j,k+1}} \]

\[ = \hat{W}^T_{Ai,j} \vartheta(x_k) + \frac{1}{2} R^{-1} \left( \frac{\partial z_{j,k}}{\partial \hat{u}_{ij,k}} \right)^T v_s \phi'(\bar{z}_{j,k})^T \hat{w}_{s,k} \frac{\partial \sigma(\hat{x}_{j,k+1})}{\partial \hat{x}_{j,k+1}} \hat{W}_{Vi} \] (18)

- Action network weight update

\[ \hat{W}_{Ai,j} = \hat{W}_{Ai,j} - \alpha_j \vartheta_k e_{ai,j}^T / (\vartheta_k^T \vartheta_k + 1) \] (19)

Proof of action network convergence can be shown
Remarks

- The results indicate that only a *nearly* optimal control law is possible when the NN reconstruction errors are explicitly considered.

- When the NN reconstruction errors are ignored, the NN estimate of the optimal control law converges to the optimal control uniformly.

- Although the NN reconstruction errors are often ignored in the literature, proof of convergence for the NN implementations of HDP are rarely studied.
Offline HDP Flowchart

Start Online Learning

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

Robust NN identifier
\( \hat{x}_{k+1} = \hat{W}_j^T \phi(v_j^T z_k) - v_k \)

Tune NN weights and Robust term

\( \| \hat{x}_{k+1} \| = 0 \)

No

Yes

Finish Online Learning

Start Offline HDP

Initialization
\( V_0 = 0 \)

Control policy update
\( \hat{u}_{i,j} = \hat{W}_j z_k - \alpha_j q_j^T q_j^T / (q_j^T q_j + 1) \)
\( \hat{u}_i(x_k) = \hat{W}_j^T \phi(x_k) \)
\( e_{i,j} = \hat{W}_j^T \phi(x_k) + \frac{1}{2} R^{-1}(\frac{\partial \phi(x_k)}{\partial \hat{w}_j})^T v_j^T \hat{z}_j \hat{v}_j \frac{\partial \phi(x_k)}{\partial \hat{w}_j} \hat{W}_j^T \phi(x_k) \)

Value function update
\( \hat{V}_{i+1} = \int_{\Omega} \sigma(x_k) \sigma(x_k) \hat{V}_i(x_k) d\Omega + \int_{\Omega} \sigma(x_k) \hat{V}_i(x_k) d\Omega = \hat{V}_i(x_k) \)
\( \hat{V}_i(x_k) = \hat{W}_j^T \sigma(x_k) \)

\( k = k + 1 \)

No

Yes

\( \| \hat{V}_{i+1} - \hat{V}_i \| < \varepsilon \)

\( \hat{V}_{i+1} \)

\( i = i + 1 \)
Simulation Results

- **Example 1: Linear System**
  \[
  x_{k+1} = \begin{bmatrix} x_{1,k+1} \\ x_{2,k+1} \end{bmatrix} = \begin{bmatrix} 0.3 & -0.8 \\ 0.8 & 1.8 \end{bmatrix} x_k + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_k
  \]

- Online identifier parameters:
  \[
  \alpha_s = 0.09, \quad \gamma_s = 0.0027
  \]

- Offline training set and learning parameter:
  \[
  x_1 \in [-0.5,0.5], \quad x_2 \in [-0.5,0.5], \quad \alpha_j = 0.01
  \]

- Value network activation functions
  \[
  \{x_1^2, x_1x_2, x_2^2, x_1^4, x_1^3x_2, \ldots, x_2^6\}
  \]

- Action network activation functions
  \[
  \{x_1, x_2, x_1^3, \ldots, x_2^5\}
  \]

- DARE solution
  \[
  u^*_k = -[0.691 \ 1.3405] x_k
  \]
Linear System Results

State Identification Error

\[ \text{Identification Error} \]

Iteration Index \( k \)

Fig. 2

\[ x_1 - x_2 \] Trajectories

\[ \text{DARE Solution} \]

\[ \text{NN Solution} \]

Fig. 3
Nonlinear System Example

- **Example 2:** Nonlinear system

\[
x_{k+1} = \begin{bmatrix}
x_{1,k+1} \\
x_{2,k+1}
\end{bmatrix} = \begin{bmatrix}
-sin(0.5x_{2,k}) \\
-cos(1.4x_{2,k})sin(0.9x_{1,k})
\end{bmatrix} + \begin{bmatrix}
0 \\
1
\end{bmatrix} u_k
\]

- NN identifier, value and action network parameters same as Ex. 1

- HDP algorithm described in Al-Tamimi, Lewis, and Abu-Khalaf (2008) was also implemented using full knowledge of the nonlinear system dynamics for comparison
Challenges of PI and VI–based Schemes

- Iterative-based schemes may not be suitable for hardware implementation.
- PI and VI-based schemes may not converge if suitable number of iterations are not chosen.
- In certain instances, model of the nonlinear system needs to be created before designing the PI and VI-schemes.
Consider discrete-time nonlinear affine system

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

where

\[
\begin{bmatrix}
-1.85x_{2,k} \\
\sin(0.5 \cdot x_{1,k} - x_{2,k}) + 1.5x_{2,k}
\end{bmatrix},
\begin{bmatrix}
0 \\
-x_{2,k}
\end{bmatrix}
\]

Simulation parameter

- Initial states: \( x_0 = [0.5 \ - \ 4.5]^T \)
- \( Q = I_{2 \times 2}, R = 1 \)
- Sampling interval: \( T_s = 0.1 \text{ sec} \)
Effect of Insufficient Iterations

From these figures it is obvious when number of iteration is decreased, policy iteration cannot force the regulation errors converge to zeros.
Online Near Optimal Regulation

- Optimal control
  \[ u^*_k = - \frac{1}{2} R^{-1} g(x_k)^T \partial J(x_{k+1})/\partial x_{k+1} \] (20)

- No offline scheme or internal dynamics \( f(x_k) \)

- To begin, the cost function (2) is assumed to have an NN representation written as
  \[ J(x_k) = W^T_c \sigma(x_k) + \varepsilon_c \] (21)

- The NN approximation of (4) is written as
  \[ \hat{J}(x_k) = \hat{W}^T_{c,k} \sigma(x_k) = \hat{W}^T_{c,k} \sigma_k, \] (22)

- The cost function error (residual or TD) dynamics are formed as
  \[ e_{c,k} = r_{k-1} + \hat{W}^T_{c,k} \sigma_k - \hat{W}^T_{c,k} \sigma_{k-1} \]
  \[ e_{c,k+1} = r_k + \hat{W}^T_{c,k+1} (\sigma_{k+1} - \sigma_k) \] (23)
Auxiliary Critic Error System

- Define the auxiliary error vector

\[ E_{c,k} = Y_{k-1} + \hat{W}_{c,k} X_{k-1} \in \mathbb{R}^{1x(1+j)} \]  

where

\[ Y_{k-1} = [r_k \ r_{k-2} \ ... \ r_{k-1-j}] \]

\[ X_{k-1} = [\Delta\sigma_k \ \Delta\sigma_{k-1} \ ... \ \Delta\sigma_{k-j}] \]

\[ \Delta\sigma_k = \sigma_k - \sigma_{k-1} \]

- Auxiliary error dynamics

\[ E_{c,k+1}^T = Y_k^T + X_k^T \hat{W}_{c,k+1} \]

- The system (25) can be viewed as an affine nonlinear system with control input \( \hat{W}_{c,k+1} \)
Critic Weight Update

Therefore, the critic weight update is written as

\[
\hat{W}_{c,k+1} = X_k \left( X_k^T X_k \right)^{-1} \left( \alpha_c E^T_{c,k} - Y_k^T \right)
\]  

(26)

and (24) becomes

\[
E^T_{c,k+1} = \alpha_{c,k} E^T_{c,k}
\]

(27)

We can show that \( \left( X_k^T X_k \right)^{-1} \) exists.

The weight update (26) is comparable to the least squares updates used in offline training.

- Instead of summing over a mesh of training points, the update (26) represents a sum over the system’s time history stored in \( E_c(k) \).
- Thus, the update (9) uses data collected in real time instead of data formed offline.
The optimal control (3) has a NN representation written as

\[ u(x_k) = W_A^T \vartheta(x_k) + \varepsilon_A \quad (28) \]

The NN approximation of (28) takes the form of

\[ \hat{u}_k = \hat{u}(x_k) = \hat{W}_{A,k}^T \vartheta(x_k) \quad (29) \]

The optimal control signal error is defined as

\[ e_{a,k} = \hat{W}_{A,k}^T \vartheta(x_k) + \frac{1}{2} R^{-1} g^T(x_k) \frac{\partial \sigma(x_{k+1})}{\partial x_{k+1}} \hat{W}_{c,k} \quad (30) \]

Action network weight update

\[ \hat{W}_{A,k} = \hat{W}_{A,k} - \alpha_a \frac{\vartheta_k e_{a,k}^T}{\vartheta_k \vartheta_k + 1} \quad (31) \]

In the absence of NN reconstruction errors, the action and critic errors converge to zero asymptotically and the action and critic NN weights remain bounded. That is \( \hat{u} \to u^* \).
Offline and Online Schemes: A Comparison

Control policy update
\[ \hat{W}_{a,j}^{i+1} = \hat{W}_{a,j}^i - \alpha_j \nabla J(x_k) \nabla^T \frac{1}{2} R \left( \frac{\partial x_k}{\partial a_{i,j}} \right)^T \nabla J(x_k) \nabla \frac{\partial x_k}{\partial a_{i,j}} \hat{W}_{a,j}^i \]

Value function update
\[ \hat{V}_{i+1}^j(x_k) = \hat{V}_{i}^j(x_k) \]

Yes
\[ \| \hat{V}_{i+1}^j - \hat{V}_i^j \|_F < \varepsilon \]

No
\[ i = i + 1 \]

Start Offline HDP

Initialization \( V_0 = 0 \)

Start Online Optimal Regulation Control

 Initialization \( V_0 = 0, u_{0,k} \)

Critic NN weights update
\[ \hat{W}_{c,j,k+1} = X_k (X_k^T X_k)^{-1} (\alpha_c F_{c,k}^T - Y_k^T) \]
\[ \hat{J}(x_k) = \hat{W}_{c,j,k} \sigma(x_k) \]

Action NN weights update
\[ \hat{W}_{a,k+1} = \hat{W}_{a,k} - \alpha_a \frac{\nabla \sigma(x_k)}{\nabla^T \frac{1}{2} R \left( \frac{\partial x_k}{\partial a_{i,j}} \right)^T \nabla \frac{\partial x_k}{\partial a_{i,j}} \hat{W}_{a,k}} \]
\[ e_{a,k} = \hat{W}_{a,k}^T \sigma(x_k) + \frac{1}{2} R \left( \frac{\partial x_k}{\partial a_{i,j}} \right)^T \nabla \frac{\partial x_k}{\partial a_{i,j}} \hat{W}_{a,k} \]
\[ \hat{u}(x_k) = \hat{W}_{a,k} \sigma(x_k) \]

\[ k = k + 1 \]
Simulation Results: Example 1

- Example 1: Linear System

\[ x_{k+1} = \begin{bmatrix} x_{1,k+1} \\ x_{2,k+1} \end{bmatrix} = \begin{bmatrix} 0 & -0.8 \\ 0.8 & 1.8 \end{bmatrix} x_k + \begin{bmatrix} 0 \\ -1 \end{bmatrix} u_k \]

- Results of solving the discrete time algebraic Riccati equation (DARE) that requires \( A \) to be known

\[ u_k^* = [0.6239 \ 1.2561] x_k \]

\[ J_k^* = 1.5063x_{1,k}^2 + 2.0098x_{1,k}x_{2,k} + 3.7879x_{2,k}^2 \]

- Initial stabilizing control:

\[ u_{0,k} = [0.5 \ 1.4] x_k \]

- Implementation of the proposed optimal regulator does not required the \( A \) matrix to be known
Optimal Regulation Results

- Cost approximator activation functions:
  \[ \sigma = \{x_1^2, x_1 x_2, x_2^2, x_1 x_3, x_1 x_2, \ldots, x_2^6\} \]

- Action network activation functions:
  \[ \mathcal{G} = \{x_1, x_2, x_1^3, \ldots, x_2^5\} \]

- Tuning constants: \( a_c = 10^{-6}, \ a_a = 0.1 \)

- Final cost estimator weights:
  \[
  \hat{W}_c = [1.5071 \ 2.0097 \ 3.7886 \ -0.0082 \ -0.0015 \ 0.0025 \ 0.0030 \ldots \\
  -0.0014 \ 0.0020 \ 0.0000 \ 0.0000 \ 0.0008 \ -0.0003 \ 0.0009 \ -0.0002]
  \]
  \[ (J_k^* = 1.5063x_{1,k}^2 + 2.0098x_{1,k}x_{2,k} + 3.7879x_{2,k}^2) \]

- Final weights of the action network:
  \[
  \hat{W}_A = [0.6208 \ 1.2586 \ 0.0589 \ -0.0338 \ 0.0095 \ldots \\
  0.0092 \ -0.0049 \ 0.0074 \ 0.0050 \ 0.0075. \ -0.0054]
  \]
  \[ (u^*(k) = [0.6239 \ 1.2561]x(k)) \]
Optimal Regulation Results (cont.)

Fig. 10

Neural Network Adaptive Critic Weights

Fig. 11

Action Network Weights

Fig. 12

Action Error

$x_1$-$x_2$ Trajectories
Simulation Results: Example 2

- Regulation Example 2: Nonlinear System

\[
x_{k+1} = \begin{bmatrix} x_{1,k+1} \\ x_{2,k+1} \end{bmatrix} = \begin{bmatrix} x_{1,k}x_{2,k} \\ x_{1,k}^2 + 1.8x_{2,k} \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \end{bmatrix} u_k
\]

- Tuning parameters and activation functions same as example 1

- Initial stabilizing control:

\[
u_{0,k} = \begin{bmatrix} -0.4 & 1.24 \end{bmatrix} x_k
\]

- Online results are compared to result obtained via offline training (Chen & Jagannathan, 2008)
Online Results for the Nonlinear System

Adaptive Critic Weights

Fig. 13

Action Error

Time Index 'k'

Fig. 14

Difference Between Optimal Controls

Fig. 15

x₁-x₂ Trajectories

Fig. 16
Optimal Tracking Results (cont.)

**Table I. Cost Value Comparisons**

<table>
<thead>
<tr>
<th>Control policy</th>
<th>Cost with $f_d$ known</th>
<th>Cost with $f_d$ unknown</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_{e0}$</td>
<td>5.4779</td>
<td>5.6078</td>
</tr>
<tr>
<td>$u^*_e$</td>
<td>5.3087</td>
<td>5.3813</td>
</tr>
</tbody>
</table>

Fig. 17

**Tracking Error**

- PE removed
- Probing noise
Optimal Tracking

- For regulation, the optimal control inputs are constructed so that $u_k = 0$ when $x_k = 0$

- In contrast, tracking control inputs typically consist of feedback and feedforward terms so that $u_k$ is not zero when the tracking error $e_k$ becomes zero

- In the literature, finite horizon cost functions in the form of

  $$ J_k = \sum_{j=1}^{N} (e_j^T Q e_j + u_j^T R u_j) $$

  are often considered for tracking to mitigate the fact that $u_k$ does not vanish (Lewis and Syrmos, 1995)

- **Objective:** Design the control input to ensure that the nonlinear system (1) tracks a desired trajectory in an optimal manner.

Optimal Tracker Development

- To begin, write the desired trajectory dynamics
  \[
  x_{d,k+1} = f(x_{d,k}) + g(x_k)u_{d,k}
  \]  
  (32)

- Tracking error \( e_k = x_k - x_{d,k} \) with dynamics given by
  \[
  e_k = f_{e,k} + g(x_k)u_{e,k}
  \]  
  (33)

  with \( f_{e,k} = f(x_k) - f(x_{d,k}) \) and \( u_{e,k} = u_k - u_{d,k} \)

- Infinite horizon cost function for tracking can be given by
  \[
  J_{e,k} = \sum_{i=0}^{\infty} r_{e,k+i} = r_{e,k} + \sum_{i=0}^{\infty} r_{e,k+i+1} = r_{e,k} + J_{e,k+1}
  \]  
  (34)

  with \( r_{e,k} = Q_e(e_k) + u_{e,k}^T R_e u_{e,k} \), \( Q_e(e_k) > 0 \), \( R_e \in \mathbb{R}^{m \times m} \)

- The signal \( u_{e,k} \) is admissible, thus \( J_{e,k} \) is finite.
Optimal Tracking Control Input

- Optimal feedback control found by solving $\frac{\partial J_{e,k}}{\partial u_{e,k}} = 0$

\[ u_{e,k}^* = -\frac{1}{2} R_e^{-1} g^T(x_k) \frac{\partial J_{e}^*(e_{k+1})}{\partial e_{k+1}} \]  \hspace{1cm} (35)

- The optimal control $u_{k}^*$ is then given by

\[ u_{k}^* = u_{d,k} - \frac{1}{2} R_e^{-1} g^T(x_k) \frac{\partial J_{e}^*(e_{k+1})}{\partial e_{k+1}} \]  \hspace{1cm} (36)

where the feedforward term $u_{d,k}(k)$, is found from (32) as

\[ u_{d,k}(k) = g(x_k)^{-1}(x_{d,k+1} - f_{d,k}) \]  \hspace{1cm} (37)

Optimal control requires nonlinear cost function, feedback and feedforward values which will be approximated by OLA’s.
The feedforward control input (37) is approximated using an OLA and written as

\[ \hat{u}_{d,k} = g(x_k)^{-1}(x_{d,k+1} - \hat{\Theta}_{d,k}^T \phi_d) \]  

(38)

The feedforward OLA update is derived using Lyapunov methods to be

\[ \hat{\Theta}_{d,k+1} = \hat{\Theta}_{d,k} + \alpha_k \phi_d (e_{k+1} - g(x_k) \hat{u}_{e,k})^T \]  

(39)

The estimate of the system control input (36) is then written as

\[ \hat{u}_k = \hat{u}_{d,k} + \hat{u}_{e,k} \]  

(40)
Remarks

- **Remark 1:**
  - For offline ADP, the optimal cost function is learned using a mesh of training points. In contrast, the optimal cost function is learned online in this work using the system’s time history.

- **Remark 2:**
  - By construction, the feedback control input error $e_{ea,k}$ becomes zero when the tracking error $e_k$ is zero. As a result, the feedback control estimator (41) cannot be updated once the tracking error has converged to zero.
  - To prevent the control input error from becoming zero before the cost function and nearly optimal control policy have been learned, external probing noise may be introduced into the nonlinear system dynamics to ensure the tracking error is persistently exciting.
Online Regulation and Tracking: Comparison

![Diagram](image)

- **Start Online Optimal Regulation Control**
- **Initialization**
  \[ V_0 = 0, u_{0,k} \]
- **Critic NN update**
  \[ \hat{w}_{c,k+1} = \hat{w}_{c,k} - X_k (X_k^T X_k)^{-1} (\alpha_c E_{c,k}^T - Y_k^T) \]
  \[ \hat{j}(x_k) = \hat{w}_{c,k}^T \sigma(x_k) \]
- **Action NN update**
  \[ \hat{e}_{a,k+1} = \hat{w}_{a,k}^T g(x_k) + \frac{R^{-1} g^T(x_k) \partial \sigma(x_{k+1})}{\partial x_{k+1}} \hat{w}_{c,k} \]
  \[ e_{a,k} = \hat{w}_{a,k}^T g(x_k) + \frac{R^{-1} g^T(x_k) \partial \sigma(x_{k+1})}{\partial x_{k+1}} \hat{w}_{c,k} \]
  \[ \hat{u}(x_k) = \hat{w}_{a,k}^T g(x_k) \]
  \[ k = k + 1 \]
- **Cost function update**
  \[ \tilde{\Theta}_{c,k+1} = \tilde{\Theta}_{c,k} - \alpha_c X_{c,k} E_{c,k}^T \]
  \[ \hat{j}(e_k) = \hat{j}(e_k) = \tilde{\Theta}_{c,k}^T \sigma(e_k) \]
- **Feedback control update**
  \[ \tilde{\Theta}_{a,k+1} = \tilde{\Theta}_{a,k} - \alpha_{a,k} g(e_k) \]
  \[ e_{a,k} = \tilde{\Theta}_{a,k} g(e_k) + \frac{1}{2} R^{-1} g^T(x_k) \partial \sigma(e_{k+1}) \tilde{\Theta}_{a,k} \]
  \[ \hat{u}(e_k) = \hat{u}(e_k) = \tilde{\Theta}_{a,k} g(e_k) \]
- **Feedforward control update**
  \[ \tilde{\Theta}_{d,k+1} = \tilde{\Theta}_{d,k} + \alpha_d \phi_d (e_{k+1} - g(x_k) \hat{u}(x_k))^T \]
  \[ \hat{u}_{d,k} = g(x_k)^{-1} (x_{d,k+1} - \tilde{\Theta}_{d,k} \phi_d) \]
  \[ \hat{u}(x_k) = \hat{u}(x_k) + \hat{u}(x_k) \]
  \[ k = k + 1 \]
Nonlinear System

- Consider the Nonlinear system

\[
\begin{bmatrix}
    x_{1,k+1} \\
    x_{2,k+1}
\end{bmatrix}
= \begin{bmatrix}
    \sin(0.5x_{2,k})x_{1,k}^2 \\
    \cos(1.4x_{2,k})\sin(0.9x_{1,k})
\end{bmatrix} + \begin{bmatrix}
    1 & 0
\end{bmatrix} \begin{bmatrix}
    u_{1,k}
\end{bmatrix}.
\]

- Desired trajectory:

\[
\begin{bmatrix}
    x_{d1,k+1} \\
    x_{d2,k+1}
\end{bmatrix}
= \begin{bmatrix}
    \sin(0.25 \cdot k) \\
    \cos(0.25 \cdot k)
\end{bmatrix}.
\]

- Initial stabilizing control:

\[
u_e0 = \begin{bmatrix}
    0.1 & 0 \\
    0 & -0.1
\end{bmatrix} \begin{bmatrix}
    e_{1,k} \\
    e_{2,k}
\end{bmatrix}.
\]

- No offline training is required to implement the proposed optimal tracking control scheme

- 11 past values are used to form the auxiliary cost function error
OLA and Parameter Design

- The OLA’s considered for implementation were neural networks:
  - 10 hidden layer neurons were used in both the cost function and feedback networks along with the sigmoid activation function.
  - 25 hidden layer neurons were used in the feedforward estimator along with the radial basis activation function.

- Tuning gains were selected according to the theoretical results:
  - Cost function parameter tuning gain: \( \alpha_{ec} = 0.1 \)
  - Feedback control signal parameter tuning gain: \( \alpha_{ea} = 0.1 \)
  - Feedforward control signal parameter tuning gain: \( \alpha_d = 0.05 \)

- All tunable parameters were initialized to zero.

- Probing noise was generated from a random distribution with zero mean and variance of 0.04 to ensure the tracking error was persistently exciting.
## Optimal Tracking Results

![Graph showing tracking error over time](image)

**Fig. 18**

### Table I. Cost Value Comparisons

<table>
<thead>
<tr>
<th>Control policy</th>
<th>Cost with $f_d$ known</th>
<th>Cost with $f_d$ unknown</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_{e0}$</td>
<td>5.4779</td>
<td>5.6078</td>
</tr>
<tr>
<td>$\hat{u}^*_e$</td>
<td>5.3087</td>
<td>5.3813</td>
</tr>
</tbody>
</table>
Results for Optimal Tracking Control

Fig. 19

- All parameters remain bounded and converge to constant values as stability theorem suggested

Fig. 20
Hardware Implementation
Multi-modal Engine Control

- Objective is to control an SI engine to minimize emissions and maximize fuel efficiency
  - Engine dynamics are unknown
  - States are not available
  - Nonstrict nonlinear discrete-time system
- High levels of exhaust gas recirculation (EGR) dilute cylinder contents
- Although stoichiometric equivalence ratio is maintained, partial fuel burns and misfires become frequent
- Lean combustion changes the equivalence ratio causing partial fuel burns and misfires
- Control of fuel input attempts to reduce cyclic dispersion of heat release
Lean Engine Control

- Daw model equations: states are total air and total fuel; output is heat release or acceleration

\[
\begin{align*}
    x_1(k+1) &= AF(k) + F(k)x_1(k) - R \cdot F(k)CE(k)x_2(k) + d_1(k) \\
    x_2(k+1) &= (1 - CE(k))F(k)x_2(k) + (MF(k) + u(k)) + d_2(k) \\
    y(k) &= x_2(k)CE(k)
\end{align*}
\]

\[
CE(k) = \frac{CE_{\text{max}}}{1 + 100 \cdot \frac{\varphi(k) - \varphi_m}{\varphi_u - \varphi_l}}
\]

\[
\varphi(k) = R \frac{x_2(k)}{x_1(k)}
\]

- This model represents spark engine dynamics as a nonlinear system in nonstrict feedback form.

Adaptive critic neural network controller for lean engine operation reduces NOx by over 90%, unburned hydrocarbons by 30%, and CO by 10% from stoichiometric levels.

- Fuel efficiency is enhanced by 10%.
- Misfires have been minimized.

Cyclic Dispersion in Heat Release Without (left) and With (right) Control

Engine Heat Release During Misfire
Lean Engine Control

- Time series of heat release shows more stable heat release output with controller active.
- Return maps of current cycle heat release plotted against next cycle show 64% reduced cyclic dispersion.

Fig. 21
Experimental Data

EGR = 18%

Uncontrolled Return Map at EGR=18%

Controlled Return Map at EGR=18%

Controlled Estimation Error at EGR=18%

\[ \hat{x}_1 m = -4.76 \times 10^{-6} \]
\[ \hat{x}_2 m = 3.09 \times 10^{-6} \]
\[ \hat{y} m = -0.00229 \]

Fig. 22
A wedge formation consisting of a leader and two followers was considered:

\[ L_{ijd} = 0.8 \quad \Psi_{ijd} = \pm 2\pi / 3 \]

Initial stabilizing controls for leader \( i \) and each follower:

\[ u_{i\text{eo}} = 3 g^{-1}_i(k) e_{ic}(k) \quad u_{j\text{eo}} = 4 g^{-1}_j(k) e_{jc}(k) \]

To evaluate the performance of the optimal controller, a second non-optimal NN control scheme is selected for each robot according to (Jagannathan, 2006):

\[
\dot{u}_{\text{NN}}(k) = \bar{B}^{-1}(k)(K e_c(k) + \hat{W}^T(k) \sigma(V^Tz(k)))
\]

\[
\hat{W}(k+1) = \alpha \sigma(k) e_{c}^T(k+1) - \gamma \|e_{c}(k+1)\| \hat{W}(k)
\]

Experimental Results

Primary Components
1. MST Mote
2. Microstrain Inertial Sensor
3. CMU2 Camera
4. US Digital Wheel Encoders
5. Sharp IR Sensors

Missouri S&T Robot Testbed
Experimental Results

Fig. 24

Optimal Path

leader
follower

Fig. 25

Feedback Weights

Cost Weights

Feedforward Weights
Experimental Results: With Obstacle Avoidance

Fig. 26

Optimal Path

X Position (m)

Y Position (m)

leader
follower

Fig. 27

Neural Network Path

X Position (m)

Y Position (m)

leader
follower

Leader Position Error

Neural Network
Optimal Network

Follower Position Error

Neural Network
Optimal Network

Tracking Error (m)

Time (s)

0 20 40 60 80 100 120 140 160 180 200

0 0.1 0.2 0.3 0.4 0.5 0.6 0.7

0 0.1 0.2 0.3 0.4 0.5 0.6 0.7

0 0.1 0.2 0.3 0.4 0.5 0.6 0.7

0 0.1 0.2 0.3 0.4 0.5 0.6 0.7
Experimental Results: With Obstacle Avoidance

\[ J_{eTotal} = \sum_{k=0}^{k_{final}} J_e(e_c(k)) \]

<table>
<thead>
<tr>
<th>Control Policy/Robot</th>
<th>( \hat{u}_e )</th>
<th>( u_{NN} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>126.04</td>
<td>192.71</td>
</tr>
<tr>
<td>F</td>
<td>829.18</td>
<td>2635</td>
</tr>
</tbody>
</table>
Continuous-time RL/ADP Schemes

- Dierks & Jagannathan, ACC 2010 – Single OLA to learn the HJB equation for the online optimal control of nonlinear continuous-time systems
Conclusions

- RL using Value and Policy iteration-based schemes help solve optimal control forward-in-time without needing system dynamics.
- Due to implementation challenges, online RL/ADP schemes without iterative approach for optimal control are preferred.
- Discrete-time design and implementation is preferred for hardware implementation though the Lyapunov first difference is nonlinear with respect to the states. Lyapunov theory guarantees that all signals remain bounded while under ideal conditions, the approximated control input approaches the optimal input asymptotically.
- Simulation and hardware implementation results verify the theoretical results.