دانشکده مهندسی برق

یادگیری تقویتی در کنترل

تمرین هشتم: LQR مبتنی بر Actor-Critic

استاد: دکتر سعید شمقدری

دانشجو: سیده ستاره خسروی

زمستان ۱۴۰۳

چکیده

در تمرین سری هشتم یادگیری تقویتی در کنترل با ۲ سوال از مبحث کنترل بهینه، یادگیری تقویتی مبتنی بر نقاد و بازیگر مواجه هستیم، که در هر فصل به سوال و یا سوالات مطرح شده پاسخ داده شده است.

واژه‌های کلیدی: یادگیری تقویتی، کنترل بهینه، نقاد، بازیگر

فهرست مطالب

عنوان صفحه

فهرست تصاویر و نمودارها

عنوان صفحه

فصل ۱: کنترل بهینه مبتنی بر نقاد و بازیگر

۱.۱ مقدمه

در این فصل به ۲ سوال مربوط به بادگیری تقویتی و حل مسئله کنترل بهینه مبتنی بر Actor Critic پاسخ داده می‌شود.

۱.۲ سوال اول

صورت سوال:

۱) برای سیستم غیرخطی افاین زیر،

$$\dot{x}(t) = \begin{bmatrix} x_2 \\ -x_1\left(\frac{\pi}{2} + \arctan(5x_1)\right) - \frac{5x_1^2}{2(1+25x_1^2)} + 4x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 3 \end{bmatrix} u(t), \qquad x(0) = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

فرض کنید $Q = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$ و $R = 1$ هستند. یک کنترل‌کننده با رویکرد actor-critic برای این سیستم طراحی کنید. تابع پایه مربوط به شبکه actor را به صورت $\sigma_a(x) = x_2$ در نظر گرفته و توابع پایه مرتبط با شبکه critic را به صورت $\phi_c(x) = [x_1 \;\; x_2 \;\; x_1x_2 \;\; x_1^2 \;\; x_2^2 \;\; x_1^2 \arctan(5x_1) \;\; x_1^3]$ فرض نمایید.

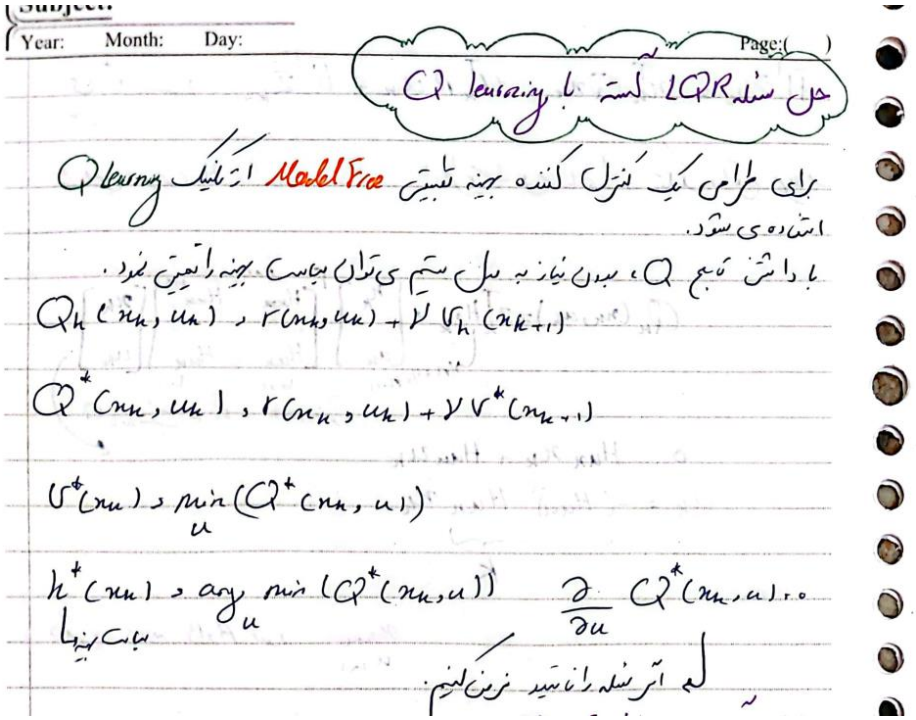
الف) الگوریتم actor-critic مطرح شده برای سیستم گسسته را به سیستم پیوسته تعمیم دهید.

ب) با استفاده از الگوریتم قسمت الف نمودار همگرایی وزن‌های شبکه‌های actor و critic، نمودار مربوط به حالات سیستم و سیگنال کنترلی متناظر را رسم نمایید.

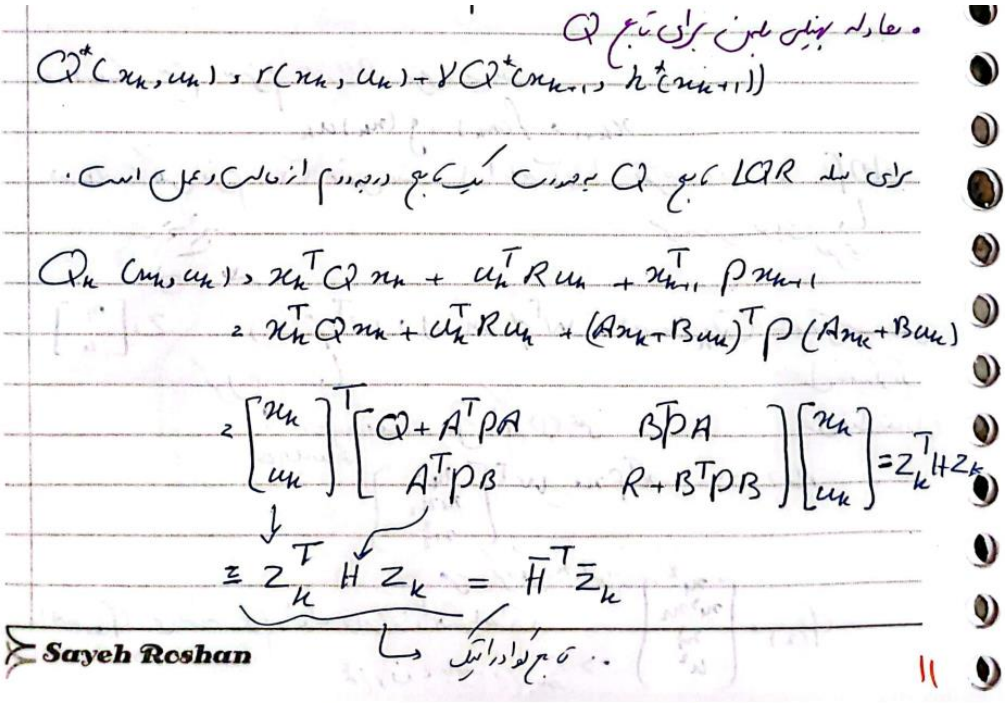
پاسخ بخش‌های الف و ب:

در این بخش برای حل سوال چون نتواستیم از روش مطرح شده در کلاس برای حل سوال پیش برویم، از روش Actor Critic مبتنی بر Q learning استفاده کردیم. ابتدا این الگوریتم را شرح داده، تعمیم آن به پیوسته را مطرح کرده و کد آن را می‌نویسیم.

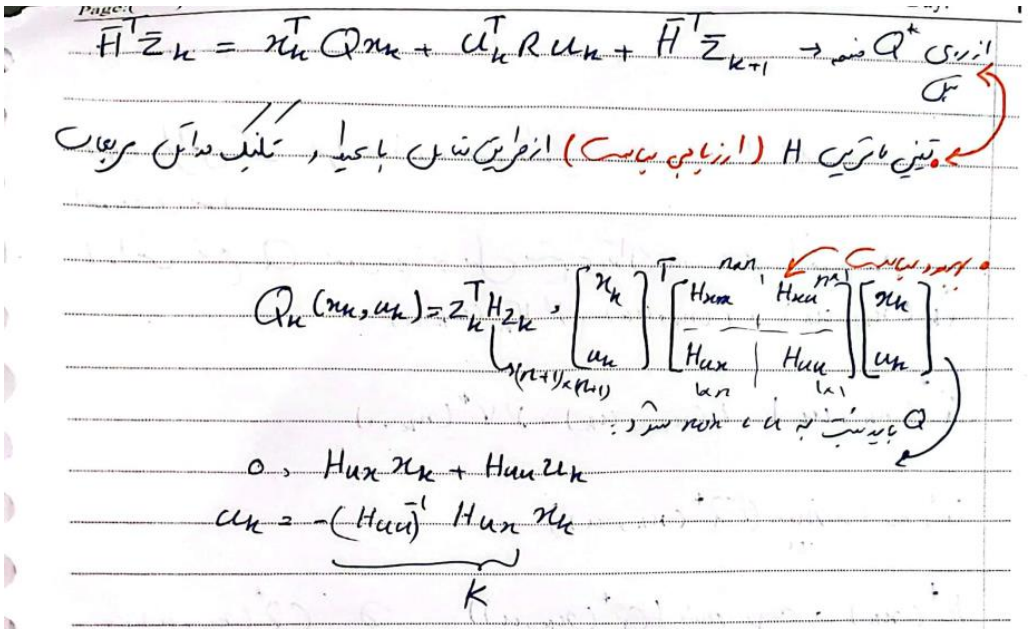
می‌دانیم که برای یک سیستم گسسته Q- learning به صورت زیر بدست می‌آید.



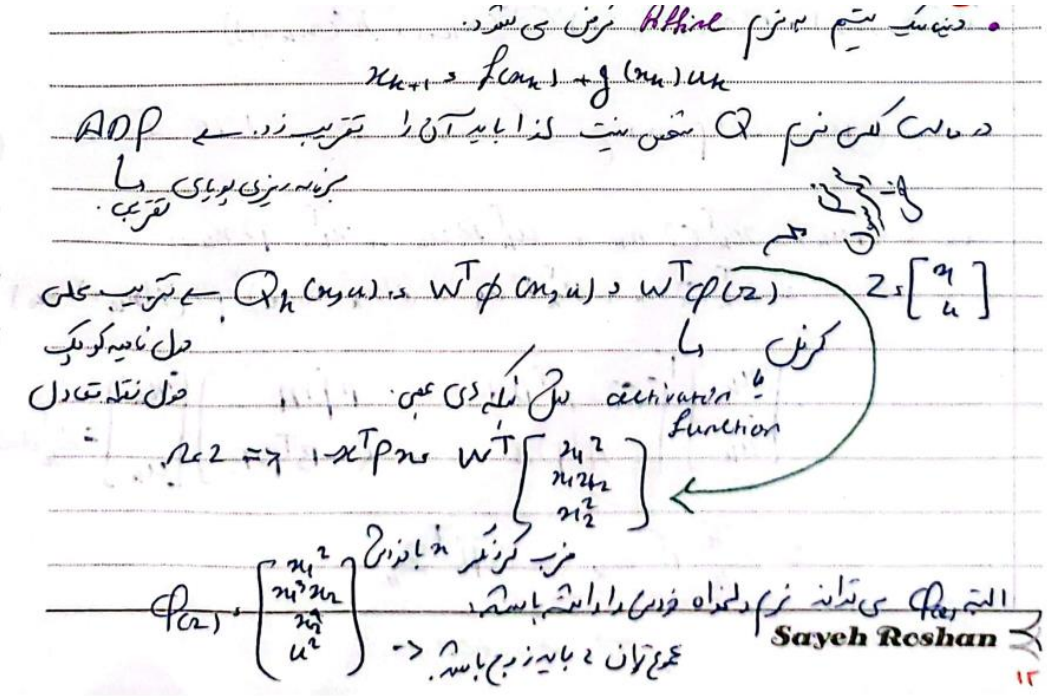
در ادامه داریم:



در نهایت بحث بهبود سیاست نیز به صورت زیر می‌گردد.



حال اگر این سیستم گسسته غیرخطی اما به صورت افاین باشد، داریم:



$$e_k = r(x_k, h(x_k)) + \gamma W^T \varphi(z_{k+1}) - W^T \varphi(z_k) = 0$$

$$\frac{\partial}{\partial u} Q_h(x_k, u) = \frac{\partial}{\partial u} W^T \varphi(x_k, u) = 0 \rightarrow u = ?$$

که نهایتا الگوریتم آن به صورت زیر خواهد بود:

Q learning Policy Iteration Algorithm الگوریتم

Initialize. Select any admissible (i.e stabilizing) control policy $h_0(x_k)$.

Policy evaluation Step. Determine the least-squares solution W_{j+1} to

$$W_{j+1}^T (\varphi(z_k) - \gamma \varphi(z_{k+1})) = r(x_k, h_j(x_k)).$$

Policy Improvement Step. Determine an improved policy using

$$h_{j+1}(x_k) = \arg\min_{u \in U} (W_{j+1}^T \varphi(x_k, u))$$

$$\frac{\partial}{\partial u} W_{j+1}^T \varphi(x_k, u) = 0 \quad \text{در کنترل بسته}$$

برای سیستم گسسته زمان که دیدیم به چه صورت می‌شود. در پیوسته زمان نیز مشابه قبل است با این تفاوت:

$$V(x) = x^T P x = \bar{x}^T \bar{P} = \bar{P}^T \bar{x} \rightarrow \phi(x)$$

$$\dot{x} = f + gu$$

$$V(x) = W^T \phi(x)$$

$$W^T (\phi(x(t)) - \phi(x(t+T))) = \int_t^{t+T} r(\tau)\, d\tau$$

که البته کدنویسی آن مشابه حالت گسسته است.

برای کدنویسی باید دقت شود، برخلاف روش ارائه شده در کلاس، در این روش باید وزن‌های مورد استفاده، برحسب u هم باشند، بعلاوه اینکه هر سطر باید توان زوج از حالت‌ها و ورودی‌ها باشد. به همین دلیل از توابع پایه‌ای صورت سوال نمی‌توانیم استفاده کنیم، و توابع دیگری را باید لحاظ کنیم.

به صورت زیر توابع پایه‌ای را تشکیل می‌دهیم.

```
%% define basis functions
syms X1 X2 U ;
assume([X1 X2 U] , 'real');

% phi(z)  z = [X ; u] for q learning
phi = [X1^2 X1*X2 X2^2 ...
       X1^4  X1^3*X2 X1^2*X2^2 X1*X2^3 X2^4 ...
       X1^6  X1^5*X2 X1^4*X2^2 X1^3*X2^3 X1^2*X2^4 X1*X2^5 X2^6 ...
       X1*U X2*U X1^3*U X2^3*U X1^5*U X2^5*U ...
       X1^4*X2*U X1*X2^4*U X1^2*X2*U X1*X2^2*U U^2]';
```

چون قرار است از استراتژی Q Learning برای حل استفاده کنیم، باید بدانیم که از فرمت z باید استفاده شود، و به همین ترتیب در تابع پایه‌ای باید U هم وجود داشته باشد، چون باید از Q نسبت به U مشتق بگیریم تا بتوانیم مسئله را حل کنیم.

در ادامه به تعداد مقادیر توابع پایه‌ای نیاز به وزن‌های حقیقی است، این وزن‌ها را تعریف و تابع Q را تشکیل می‌دهیم.

```
% Q = W'*phi for q learning
Ws = sym('Ws' , [size(phi , 1) , 1]);
assume(Ws , 'real') ;

Q = Ws'*phi ;
```

سپس می‌دانیم که با استفاده از مشتق Q بر حسب U مقدار بهینه بدست می‌آید، که به صورت زیر مشتق گرفته و مقادیر بهینه را نیز بدست آورده و ذخیره می‌کنیم و به گونه‌ای ذخیره سازی را تغییر می‌دهیم که محاسبات سریعتر انجام شود.

```
dotQ = diff(Q , U) ;
Us = solve(dotQ , U) ;

uFCN  = matlabFunction(Us);
phiFCN = matlabFunction(phi);
```

تنظیمات آموزش را به صورت زیر تعیین می‌کنیم:

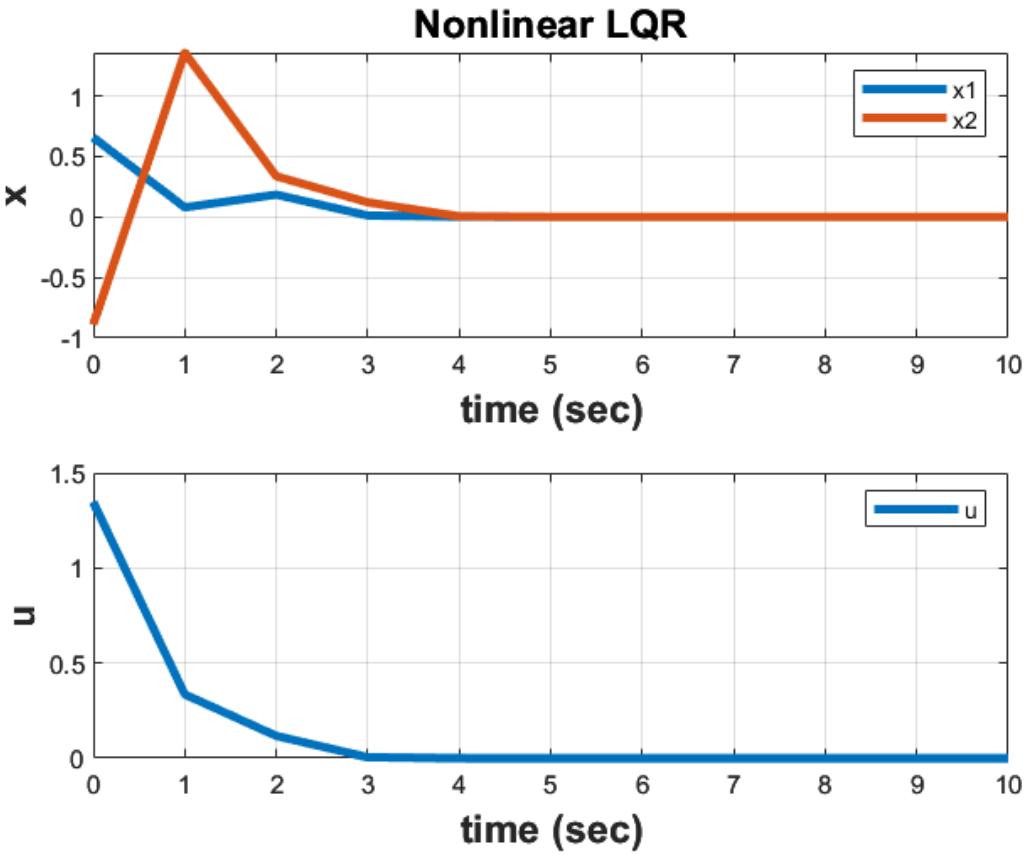
```
nP = 500 ;
M  = 100 ;
W = zeros(size(phi , 1) , nP); W(end , 1) = 1 ;
R = 1 ; Q = [0 0; 0 1] ;
```

سپس در ادامه مطابق الگوریتمی که در صفحه چهار آوردیم، آموزش را کدنویسی کردیم.

```
for j = 1:nP

    PHI = [] ;
    SAI = [] ;
    for k = 1:M
        xt = [-1; 1];
        ut = uFCN(W(6 , j),W(7 , j),W(8 , j),W(9 , j), xt(1) , xt(2));

        ut = ut + 0.01*randn ;

        xt1 = [xt(2); -1*xt(1)*((pi/2) + atan(5*xt(1))) - 5*xt(1)^2/(2*(1+25*xt(1)^2)) + 4*xt(2) + 3*ut] ;

        ut1 = uFCN(W(6 , j),W(7 , j),W(8 , j),W(9 , j), xt1(1) , xt1(2));

        PHI = [PHI ; phiFCN(ut , xt(1) , xt(2))'-phiFCN(ut1 , xt1(1) , xt1(2))'] ;
        SAI = [SAI ; xt'*Q*xt + ut'*R*ut] ;
    end
    W(: , j+1) = PHI\SAI ;

    disp(['Iteration(' num2str(j) ')']);

    if norm(W(: , j+1)-W(: , j)) < 1e-4
        break;
    end
end
```

دینامیک سیستم و شرایط اولیه را درون باکس‌ها مشخص کردیم.

پس از انجام آموزش، نتایج به صورت زیر است:



شکل ۱: نمودار حالت‌ها و سیگنال کنترلی

۱.۳ سوال دوم

صورت سوال:

۲) روند تبدیل الگوریتم ۳ به الگوریتم ۴ در مقاله مربوط به یادگیری تقویتی معکوس را توضیح دهید. (مقاله: Inverse Reinforcement Learning for Adversarial Apprentice Games)

پاسخ:

اولین گام، تقریب توابع ارزش با استفاده از شبکه‌های عصبی است، به این صورت که برای تابع ارزش، ورودی کنترلی و ورودی مخرب، به صورت زیر تقریب می‌زنیم.

$$\hat{V}_l^{ji} = \left(C_l^{ji}\right)^T \varphi(x_l) \tag{35}$$

$$\bar{u}_l^{j(i+1)} = \left(W_l^{ji}\right)^T \phi(x_l) \tag{36}$$

$$\bar{v}_l^{j(i+1)} = \left(H_l^{ji}\right)^T \rho(x_l) \tag{37}$$

where $\varphi(x_l)$ in (21), $\phi(x_l) = [\phi_1(x_l)\ \phi_2(x_l)\ \cdots\ \phi_{N_2}(x_l)]^T$, and $\rho(x_l) = [\rho_1(x_l)\ \rho_2(x_l)\ \cdots\ \rho_{N_3}(x_l)]^T$ are activation vector functions of three NNs, respectively. Moreover, $C_i^{ji} \in \mathbb{R}^{N_1}$.

در ادامه، معادله بلمن اصلی در الگوریتم ۳ گام چهارم، که معادله ۳۳ می‌باشد:

$$\begin{aligned} & V_l^{ji}(x_l(t+T)) - V_l^{ji}(x_l(t)) \\ & + \int_t^{t+T} 2\Big(\big(u_l^{j(i+1)}\big)^T R_l\big(u_l - u_l^{ji}\big) \\ & \qquad - \big(v_l^{j(i+1)}\big)^T S_l\big(v_l - v_l^{ji}\big)\Big) d\tau \\ & = \int_t^{t+T} \Big(- q^T(x_l) Q_l^j q(x_l) \\ & \qquad - (u_l^{ji})^T R_l u_l^{ji} + (v_l^{ji})^T S_l v_l^{ji}\Big) d\tau. \end{aligned} \tag{33}$$

با جایگزینی تقریب‌های شبکه عصبی به معادله ۳۸ تبدیل می‌گردد:

$$\begin{aligned} & \left(C_l^{ji}\right)^T [\varphi(x_l(t+T) - \varphi(x_l(t))] \\ & \quad + 2\sum_{h=1}^{m} r_h \left(W_{l,h}^{ji}\right)^T \int_t^{t+T} \phi(x_l)\tilde{u}_{l,h}^{ji} d\tau \\ & \quad - 2\sum_{p=1}^{k} s_p \left(H_{l,p}^{ji}\right)^T \int_t^{t+T} \rho(x_l)\tilde{v}_{1,p}^{ji} d\tau \\ & = \bar{e}_l^{ji}(t) - \int_t^{t+T} \Big(q^T(x_l)\hat{Q}_l^j q(x_l) + \left(\bar{u}_l^{ji}\right)^T R_l \bar{u}_l^{ji} \\ & \qquad - \left(\bar{v}_l^{ji}\right)^T S_l \bar{v}_l^{ji}\Big) d\tau \end{aligned} \tag{38}$$

در اینجا انتگرال‌ها و رابطه زیر:

$$\Delta\varphi = \varphi(x_l(t+T)) - \varphi(x_l(t))$$

از طریق داده‌های مشاهده شده، محاسبه می‌شوند. روش BLS نیز، وزن‌ها را با لحاظ یک معادله خطی از روی داده‌های مشاهده شده حل می‌کند.

سپس به روزرسانی وزن جریمه حالت، یا درواقع معادله ۳۴ در الگوریتم ۳:

$$\begin{aligned}&\int_t^{t+T} q^T(x_l) Q_l^{j+1} q(x_l) d\tau \qquad (34)\\ &= \int_t^{t+T} \left(u_e^{*T} R_l u_e^* - 2u_e^{*T} R_l u_l^j + (v_l^j)^T S_l v_l^j\right) d\tau\\ &\quad - V_l^j(x_l(t+T)) + V_l^j(x_l(t)).\end{aligned}$$

با جایگزینی خروجی‌های شبکه عصبی که در ۴۱ آمده تبدیل می‌شود:

$$\begin{aligned}&\int_t^{t+T} q^T(x_l) \hat{Q}_l^{j+1} q(x_l)\\ &= \int_t^{t+T} \left(u_e^{*T} R_l u_e^* - 2u_e^{*T} R_l W_l^j \phi + \rho^T H_l^j S_l \left(H_l^j\right)^T \rho\right) d\tau\\ &\quad - \left(C_l^j\right)^T (\varphi(x_l(t+T) - \varphi(x_l(t))) \qquad (41)\end{aligned}$$

وزن‌های Q نیز، از طریق رابطه ۴۳ بدست می‌آیند:

$$\mathrm{vecm}\left(\hat{Q}_l^{j+1}\right) = \left(\bar{\Gamma}_l^T \bar{\Gamma}_l\right)^{-1} \bar{\Gamma}_l^T Y_l^j.$$

باید به میزان زیر:

$$N_1 \geq N_1 + mN_2 + kN_3$$

نقطده داده جمع شود تا ماتریس‌های مورد استفاده در بدست آوردن Q، معکوس پذیر شوند.

از اینتروال‌های زمانی کوچک T باید جهت گسسته سازی استفاده شود.

در نهایت الگوریتم ۳:

Algorithm 3 Model-Free Integral Inverse RL Algorithm for Adversarial Apprentice Game

1: **Initialization**: select $Q_l^0 \geq 0$, $R_l > 0$, $S_l > 0$, initial stabilizing u_l^{00}, and small thresholds ε_1, ε_2. Set $j = 0$. Apply stabilizing u_l to the learner dynamics (31).

2: **Outer j iteration loop based on inverse optimal control**

3: **Inner i iteration loop using optimal control:** given j, set $i = 0$, and use initial stabilizing u_l^{j0}.

4: Off-policy Integral RL for solving V_l^j, u_l^j and v_l^j

$$\begin{aligned} & V_l^{ji}(x_l(t+T)) - V_l^{ji}(x_l(t)) \\ & + \int_t^{t+T} 2\Big(\big(u_l^{j(i+1)}\big)^T R_l\big(u_l - u_l^{ji}\big) \\ & \qquad - \big(v_l^{j(i+1)}\big)^T S_l\big(v_l - v_l^{ji}\big)\Big) d\tau \\ & = \int_t^{t+T} \Big(- q^T(x_l) Q_l^j q(x_l) \\ & \qquad - (u_l^{ji})^T R_l u_l^{ji} + (v_l^{ji})^T S_l v_l^{ji}\Big) d\tau. \end{aligned} \tag{33}$$

5: Stop if $\|V_l^{ji} - V_l^{j(i-1)}\| \leq \varepsilon_1$, then set $V_l^j = V_l^{ji}$, $u_l^j = u_l^{ji}$ and $v_l^j = v_l^{ji}$. Otherwise, set $i \leftarrow i + 1$ and go to Step 4.

6: **State-penalty weight Q_l^{j+1} update using the expert's demonstration u_e^***

$$\begin{aligned} & \int_t^{t+T} q^T(x_l) Q_l^{j+1} q(x_l) d\tau \\ & = \int_t^{t+T} \Big(u_e^{*T} R_l u_e^* - 2u_e^{*T} R_l u_l^j + (v_l^j)^T S_l v_l^j\Big) d\tau \\ & \quad - V_l^j(x_l(t+T)) + V_l^j(x_l(t)). \end{aligned} \tag{34}$$

7: **Stop if** $\|Q_l^{j+1} - Q_l^j\| \leq \varepsilon_2$. Otherwise, set $u_l^{(j+1)0} = u_l^j$ and $j \leftarrow j + 1$, then go to Step 3.

به الگوریتم ۴ تبدیل می‌شود:

Algorithm 4 Model-Free Integral Inverse RL Algorithm via NNs for Adversarial Apprentice Game

1: **Initialization**: select $\hat{Q}_l^0 \geq 0$, stabilizing $\bar{u}_l^{00}$, and small thresholds ε_1, ε_2. Apply stabilizing u_l to (31). Set $j = 0$.
2: **Outer j iteration loop based on inverse optimal control**
3: **Inner i iteration loop using optimal control:** given j, set $i = 0$, and use initial stabilizing $\bar{u}_l^{j0}$.
4: Off-policy integral RL for solving C_l^{ji}, W_l^{ji} and H_l^{ji} by (40).
5: Stop if $\|\hat{V}_l^{ji} - \hat{V}_l^{j(i-1)}\| \leq \varepsilon_1$, then set $(C_l^j \ W_l^j \ H_l^j)$ $=(C_l^{ji} \ W_l^{ji} \ H_l^{ji})$. Otherwise, set $i \leftarrow i + 1$ and go to Step 4.
6: **State-penalty weight $\hat{Q}_l^{j+1}$ update by** (43) **using the expert's demonstration** u_e^*.
7: **Stop if** $\|\hat{Q}_l^{j+1} - \hat{Q}_l^j\| \leq \varepsilon_2$. Otherwise, set $\bar{u}_l^{(j+1)0} = \bar{u}_l^j$ and $j \leftarrow j + 1$ and go to Step 3.