دانشکده مهندسی برق

یادگیری تقویتی در کنترل

تمرین ششم: یادگیری تقویتی در کنترل بهینه

استاد: دکتر سعید شمقدری

دانشجو: سیده ستاره خسروی

زمستان ۱۴۰۳

چکیده

در تمرین سری ششم یادگیری تقویتی در کنترل با ۳ سوال از مبحث کنترل بهینه مواجه هستیم، که در هر فصل به سوال و یا سوالات مطرح شده پاسخ داده شده است.

واژه‌های کلیدی: یادگیری تقویتی، کنترل بهینه

فهرست مطالب

عنوان	صفحه

فهرست تصاویر و نمودارها

عنوان صفحه

فصل ۱:   کنترل بهینه

۱.۱ مقدمه

در این فصل به ۳ سوال مربوط به این فصل پاسخ داده می‌شود.

۱.۲ سوال اول

صورت سوال قسمت الف:

۱. سیستم زمان گسسته زیر را در نظر بگیرید :

$$x(k+1) = \begin{bmatrix} 0.9065 & 0.0816 & -0.0005 \\ 0.0743 & 0.9012 & -0.0007 \\ 0 & 0 & 0.1327 \end{bmatrix} x(k) + \begin{bmatrix} -0.0027 \\ -0.0068 \\ 1 \end{bmatrix} u(k), \qquad x(0) = \begin{bmatrix} 10 \\ -10 \\ -3 \end{bmatrix}$$

فرض کنید $Q = I_3$ و $R = 1$ هستند.

الف) (با فرض دانستن مدل) تابع مقدار بهینه و پالیسی فیدبک حالت بهینه را با استفاده از دستور dare در متلب به دست آورید.

پاسخ:

در این قسمت با توجه به پیشنهاد خود وبسایت MATLAB، از دستور idare بجای dare استفاده کردیم.

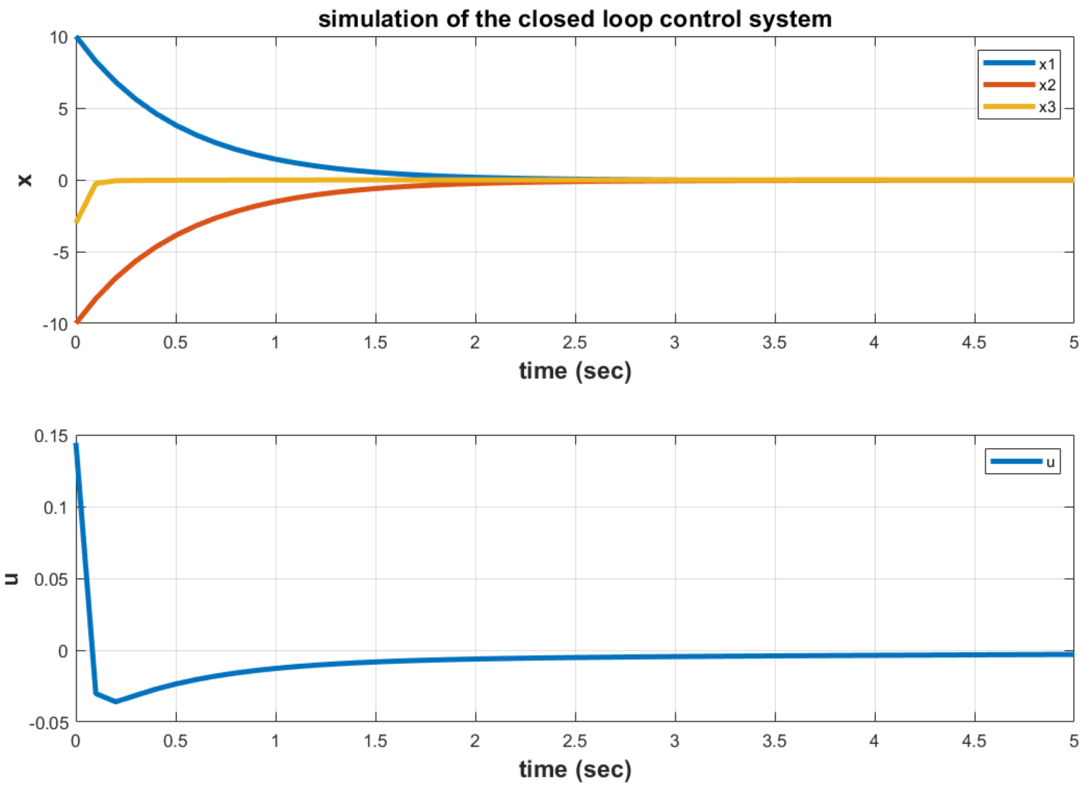
برای حل این سوال ابتدا دینامیک مسئله مطابق زیر تعریف گردید:

```
%%
A = [0.9065 0.0816 -0.0005;0.0743 0.9012 -0.0007;0 0 0.1327];
B = [-0.0027 -0.0068 1]';

n = size(A , 1) ;

R = 1 ;
Q = eye(n);
E = eye(n);
S = zeros(n , 1) ;

[P_lqr , K_lqr , L] = idare(A , B , Q , R , S , E);
```

در انتهای کد نیز با دستور مذکور مقدار K و P حاصل بدست آمد که K حاصل از LQR برابر است با:

```
K_lqr =

   -0.0643    -0.0699     0.0667
```

در ادامه‌ی کد سوال اول بخش الف، سیگنال‌ها و متغیرهای حالت را درحضور این پالیسی بدست آمده رسم کردیم که به صورت زیر است:



شکل ۱: سیگنال **u** و متغیرهای حالت پس از اعمال پالیسی حاصل از **idare**

صورت سوال قسمت ب:

ب) (با فرض دانستن مدل) کنترل بهینه را با استفاده از الگوریتم PI به‌دست آورید. (برای حل معادله ماتریسی میتوانید از راهنمای پیوست کمک بگیرید.)

$$\left(A - BK_j\right)^T P_{j+1}\left(A - BK_j\right) - P_{j+1} + Q + K_j^T R K_j = 0$$

$$K_{j+1} = \left(R + B^T P_{j+1} B\right)^{-1} B^T P_{j+1} A$$

پاسخ:

برای حل این قسمت ابتدا تابعی تعریف کردیم که با استفاده از آن مقدار P را بدست آوریم. در اینجا از بهینه سازی استفاده می‌کنیم. فرض می‌کنیم معادله‌ای که قرار است از روی آن P محاسبه شود (معادله ریکاتی صورت سوال)، برابر است با M و لازم است این M برابر با صفر باشد فلذا یک P باید انتخاب شود که این M را کمینه کند. پیاده سازی این تابع به صورت زیر است:

```
%% Optimization Function
function z = PI(P, A, B, K, Q, R)
    P = reshape(P, size(A));
    M = (A - B * K)' * P * (A - B * K) - P + Q + K' * R * K;
    z = sum(abs(M(:)));
end
```

حال با لحاظ دینامیک سیستم، تنظیمات الگوریتم را به صورت زیر قرار می‌دهیم:

```
%% Policy Iteration Algorithm
nP = 100; % Number of policy iterations
K = zeros(nP, n); % Storing policies
K(1, :) = [0.4, 0.5, 0.6]; % Initial policy (should be feasible)
P = cell(nP, 1);
P{1} = zeros(n);

% Convergence metrics
delta_K_values = zeros(nP, 1);
delta_P_values = zeros(nP, 1);

options = optimoptions('fmincon', 'Display', 'off');
```

می‌خواهیم در ۱۰۰ تکرار مسئله حل شود، تمامی سیاست‌های بدست آمده را نیز می‌خواهیم در K ذخیره کنیم. برای اینکه الگوریتم استفاده شود، یک سیاست اولیه نیز مطابق کد بالا تعریف می‌کنیم. فقط باید حواسمان باشد که این سیاست اولیه شدنی باشد. در نهایت برای ذخیره سازی Pها نیز متغیر P که قرار است سلولی باشد و هر سلول آن شامل ۹ پارامتر باشد تعریف می‌گردد. برای رسم نموداری که در بخش آخر سوال اول خواسته شده نیز لازم است متغیرهایی تعریف کنیم. تنظیمات بهینه سازی را نیز باید لحاظ کنیم. این تنظیمات مربوط به fmincon است که برای حل تابع بهینه سازی که نوشتیم به کار می‌رود.

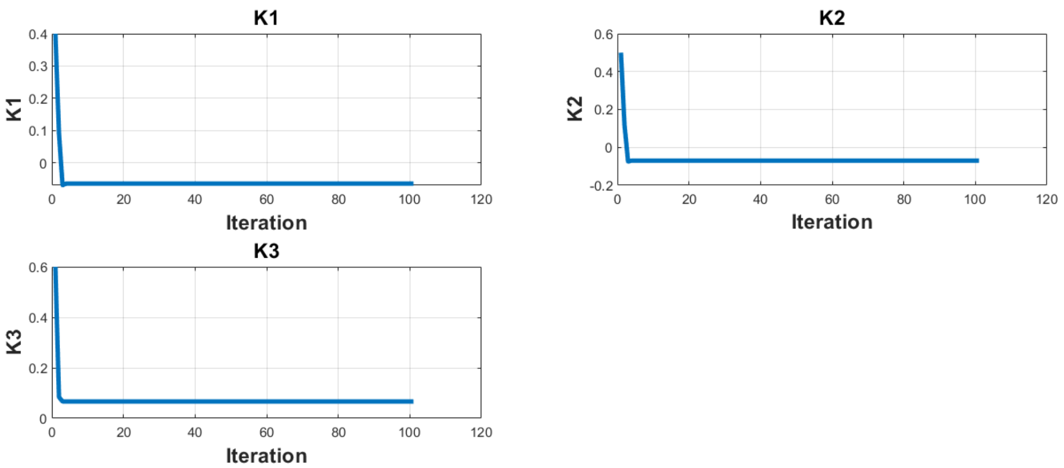
حلقه الگوریتم نیز به صورت زیر است، ابتدا در هر تکرار P را بدست می‌آوریم و سپس سیاست را بهبود می‌دهیم.

```
for j = 1:nP
    % Step 1: Solve for P_{j+1}
    cost = @(P) PI(P, A, B, K(j, :), Q, R);
    [Ps, ~] = fmincon(cost, P{j}(:), [], [], [], [], [], [], [], options);
    P{j+1} = reshape(Ps, size(A));

    % Step 2: Update Policy (K_{j+1})
    K(j+1, :) = (R + B' * P{j+1} * B)^(-1) * (B' * P{j+1} * A);

    % Compute Convergence Metrics
    delta_K = norm(K(j+1, :) - K(j, :));
    delta_P = norm(P{j+1} - P{j}, 'fro'); % Frobenius norm for matrices
    delta_K_values(j) = delta_K;
    delta_P_values(j) = delta_P;

    disp(['Iteration(', num2str(j), ')']);
end
```

نتیجه به صورت زیر است:

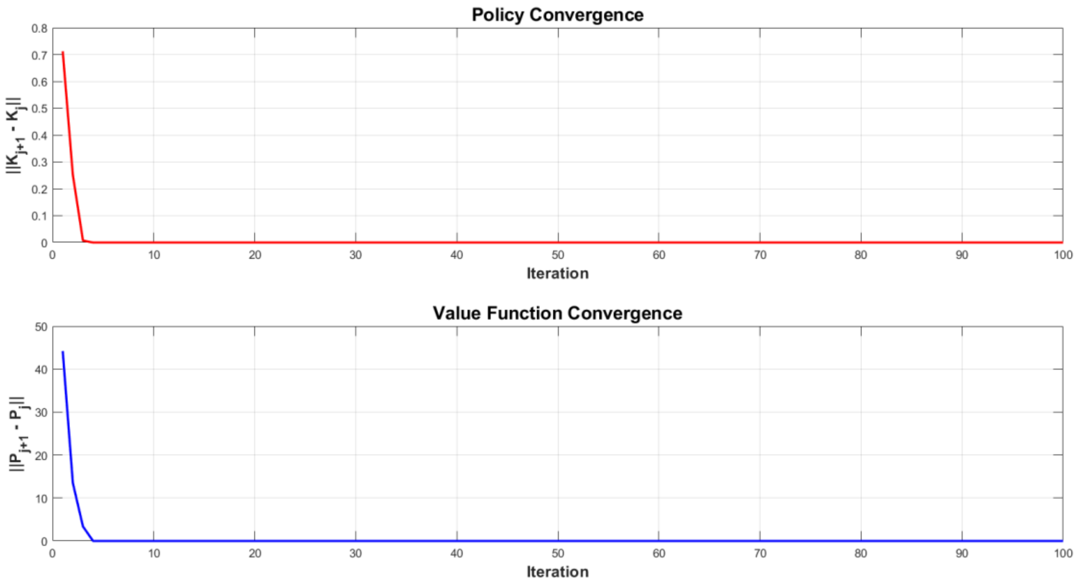
```
K LQR = -0.064318    -0.069887     0.066683
K PI = -0.064325    -0.069894     0.066683
```

در طی تکرارهای مختلف نمودار سیاست به صورت زیر خواهد بود:



شکل ۲: بهبود سیاست در PI

نمودارهای خواسته شده در بخش آخر این سوال نیز به صورت زیر می‌گردد:



شکل ۳: نمودار همگرایی در PI

نکته قابل توجه این است که نیاز نیست حتما ۱۰۰ تکرار انجام شود، الگوریتم در تعداد تکرارهای کمتر نیز به همگرایی می‌رسد.

صورت سوال قسمت پ:

پ) (با فرض دانستن مدل) کنترل بهینه را با استفاده از الگوریتم VI به‌دست آورید.

$$K_j = \left(R + B^T P_j B\right)^{-1} B^T P_j A$$

$$P_{j+1} = \left(A - BK_j\right)^T P_j \left(A - BK_j\right) + Q + K_j^T R K_j$$

پاسخ:

در این بخش نیز کد مشابه قبل است فقط تنظیمات به صورت زیر تغییر می‌کند، در اینجا برای به روز رسانی P از تکرار سیاست تعمیم یافته نیز استفاده می‌کنیم، سایر تنظیمات مشابه قبل است.

```
%% Value Iteration Algorithm
nP = 100; % Number of policy iterations
nGPI = 10; % Number of gradient policy iterations for P update
K = zeros(nP, n);
K(1, :) = [0.4, 0.5, 0.6]; % Initial policy (should be feasible)
P = cell(nP, 1);
P{1} = zeros(n);

% Convergence metrics
delta_K_values = zeros(nP, 1);
delta_P_values = zeros(nP, 1);
```

در ادامه حلقه‌ی تکرار به صورت زیر است:

```
for j = 1:nP
    % Step 1: Update Value Function (P_{j+1})
    PP = P{j};
    for i = 1:nGPI
        PP = (A - B * K(j, :))' * PP * (A - B * K(j, :)) + Q + K(j, :)' * R * K(j, :);
    end
    P{j+1} = PP;

    % Step 2: Update Policy (K_{j+1})
    K(j+1, :) = (R + B' * P{j+1} * B)^(-1) * (B' * P{j+1} * A);

    % Compute Convergence Metrics
    delta_K = norm(K(j+1, :) - K(j, :));
    delta_P = norm(P{j+1} - P{j}, 'fro'); % Frobenius norm for matrices
    delta_K_values(j) = delta_K;
    delta_P_values(j) = delta_P;

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

    % Convergence Check
    if delta_K < 1e-6
        break;
    end
end
disp(['Elapsed Time = ', num2str(toc)]);

disp(['K LQR = ', num2str(K_lqr)]);
disp(['K VI = ', num2str(K(j+1, :))]);
```

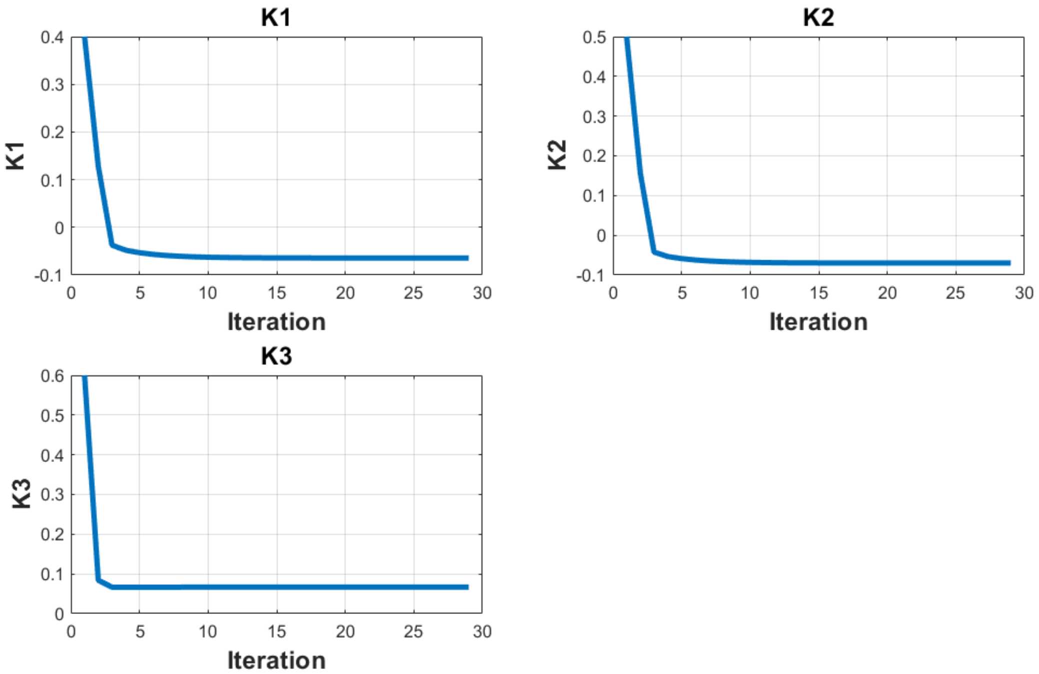
با استفاده از GPI، تابع ارزش را بروز کرده و P محاسبه می‌گردد، و از این P برای بروز رسانی سیاست استفاده می‌شود. در اینجا برخلاف کد قسمت قبل، شرط توقف را نیز لحاظ کردیم، که دیگر نیازی به انجام تکرارهای بیشتر در صورت همگرایی نباشد.

بقیه قسمت‌ها کد توضیح خاصی ندارد و صرفا رسم نمودار است.

سیاست بدست آمده به صورت زیر است:

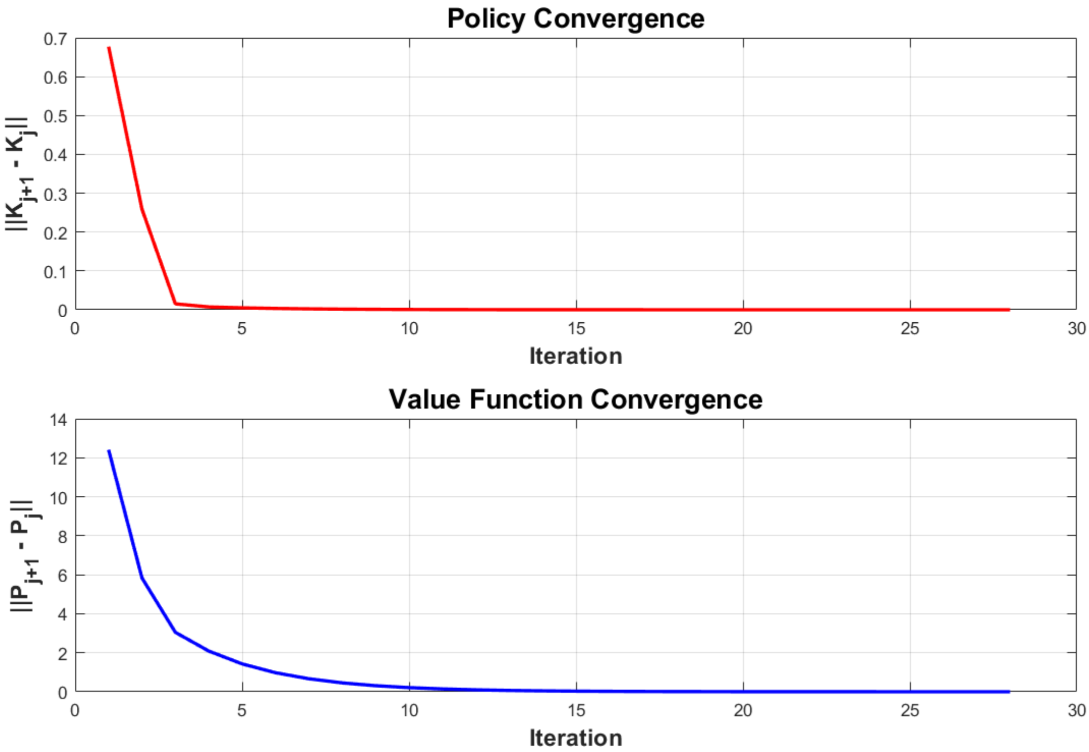
```
K LQR = -0.064318    -0.069887     0.066683
K VI = -0.064317    -0.069886     0.066683
```

نمودار مقادیر K نیز به صورت زیر خواهد بود، برخلاف قسمت قبلی اینجا تا ۲۸ تکرار انجام شده است.



شکل ۴: روند تغییرات K در VI

نمودار خواسته شده در بخش آخر سوال نیز در شکل ۵ قابل مشاهده است. در اینجا نیز با استفاده از VI به مقادیر بهینه که توسط idare بدست آمده بود همگرا شدیم.



شکل ۵: همگرایی در VI

صورت سوال قسمت ت:

ت) (با فرض دانستن مدل) کنترل بهینه را با استفاده از الگوریتم λ_PI به دست آورید. این الگوریتم بدین‌گونه است که ابتدا پالیسی آپدیت شده و سپس تابع مقدار به‌روز می‌شود. این قسمت به ازای $\lambda = 0.5$ و $\lambda = 0.9$ حل شود.

$$K_j = \left(R + B^T P_j B\right)^{-1} B^T P_j A$$

$$P_{j+1} = (1-\lambda)\left(A - BK_j\right)^T P_j \left(A - BK_j\right) + \lambda\left(A - BK_j\right)^T P_{j+1}\left(A - BK_j\right) + K_j^T R K_j + Q$$

پاسخ:

برای پاسخ به این بخش نیز مشابه الگوریتم PI از تابع بهینه سازی استفاده کردیم، فقط متناسب با تغییرات صورت سوال در معادله‌ی ریکاتی تغییرات لازم را ایجاد کردیم که به صورت زیر است:

```
%% Lambda Policy Iteration Cost Function
function z = lambda_PI(P_vec, A, B, K, Q, R, P_j, lambda)
    P = reshape(P_vec, size(A)); % Reshape vector back into matrix form
    K_mat = reshape(K, size(B, 2), size(A, 1));

    % Compute the lambda-weighted equation
    M = (1 - lambda) * (A - B * K_mat)' * P_j * (A - B * K_mat) + ...
        lambda * (A - B * K_mat)' * P * (A - B * K_mat) + ...
        K_mat' * R * K_mat + Q - P;

    z = M(:); % Flatten the matrix for optimization
end
```

سایر بخش‌های کد مانند قبل است و فقط حلقه تکرار به صورت زیر می‌گردد:

```
for j = 1:nP
    % Step 1: Policy Improvement (Update K)
    K_j = K(j, :);
    P_j = P{j};
    K(j+1, :) = (R + B' * P_j * B)^(-1) * (B' * P_j * A);

    % Step 2: Value Function Update (Solve for P_{j+1})
    options = optimoptions('fsolve', 'Display', 'off', 'TolFun', 1e-8);
    cost = @(P_vec) lambda_PI(P_vec, A, B, K(j+1, :), Q, R, P_j, lambda);
    P_next_vec = fsolve(cost, P_j(:), options);
    P{j+1} = reshape(P_next_vec, size(A));

    % Compute Convergence Metrics
    delta_K = norm(K(j+1, :) - K(j, :));
    delta_P = norm(P{j+1} - P{j}, 'fro'); % Frobenius norm for matrices
    delta_K_values(j) = delta_K;
    delta_P_values(j) = delta_P;

    fprintf('Iteration %d: ||K_{j+1} - K_j|| = %.8f, ||P_{j+1} - P_j|| = %.8f\n', ...
        j, delta_K, delta_P);

    % Convergence Check
    if delta_K < tol && delta_P < tol
        converged = true;
        break;
    end
end
```

حاصل به ازای مقدار lambda برابر با ۰.۵ و در ۵۰ تکرار به صورت زیر است:

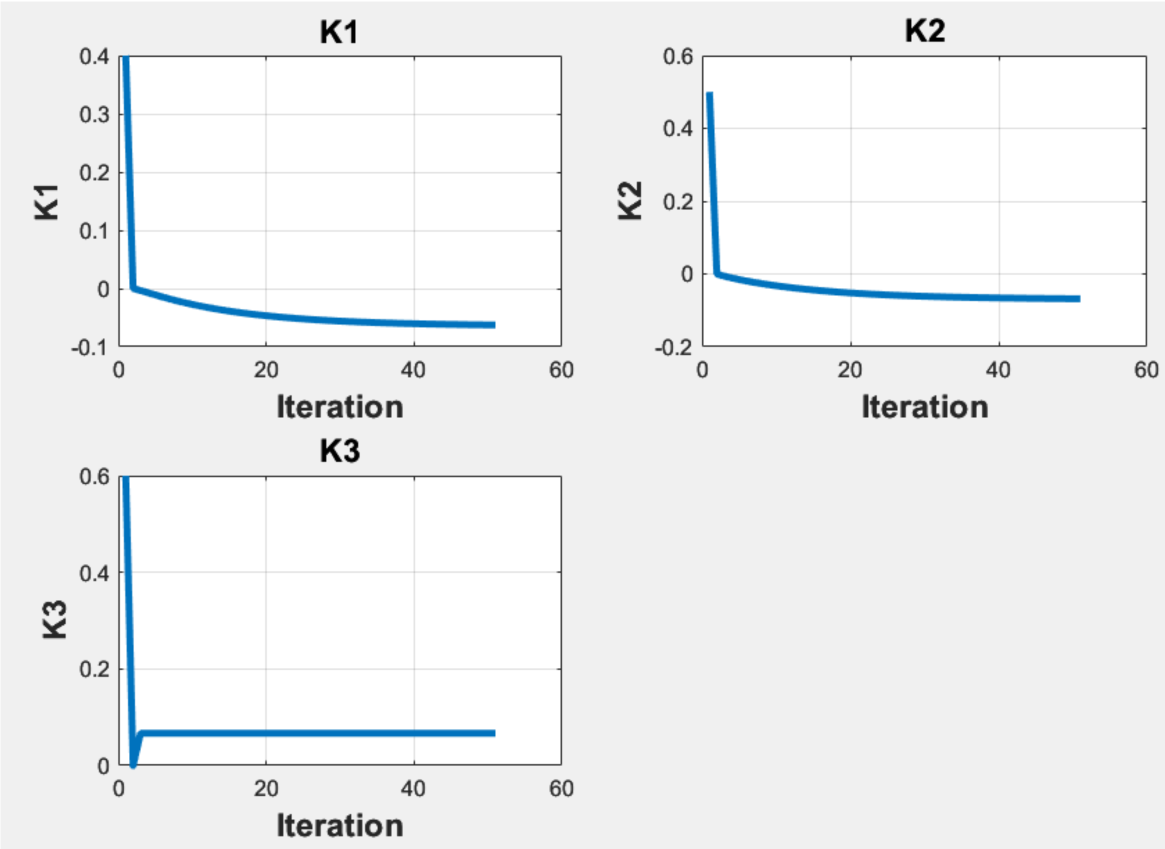
```
K Algorithm = -0.062591    -0.068137    0.066682
K LQR = -0.064318    -0.069887    0.066683
```

و برای lambda با مقدار ۰.۹ می‌شود:

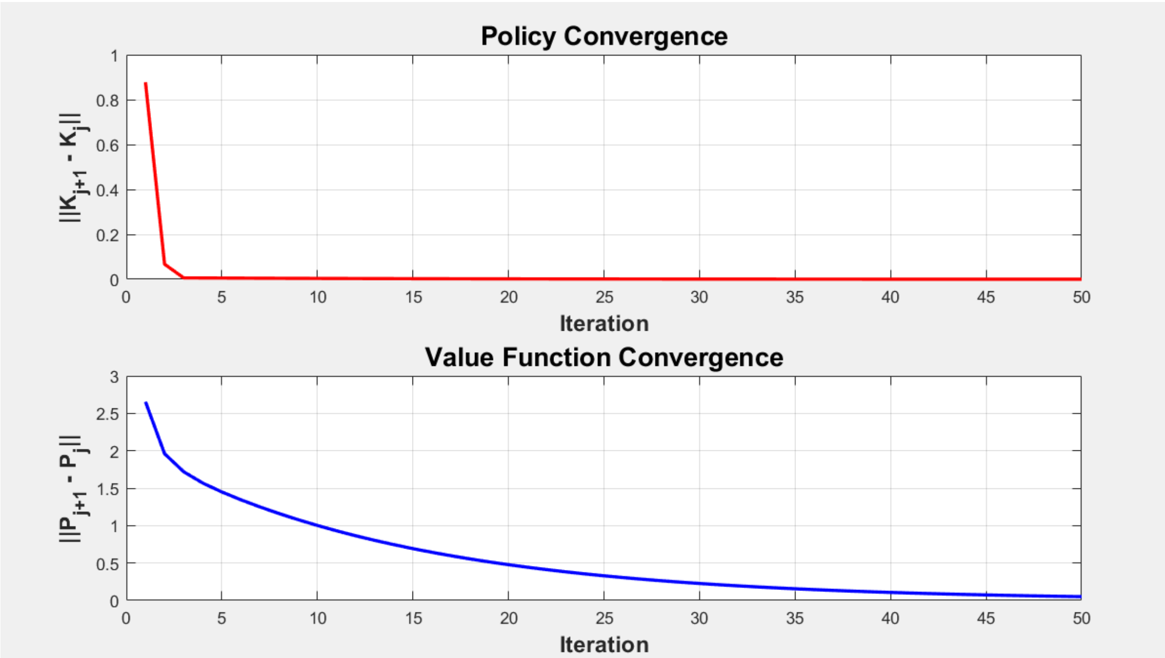
```
K Algorithm = -0.064318    -0.069887     0.066683
K LQR = -0.064318    -0.069887     0.066683
```

که نشان می‌دهد با این مقدار بهتر به جواب بهینه همگرا می‌شویم تا با اعمال مقدار ۰.۵.

نمودارها نیز به صورت زیر است، برای مقدار لاندای ۰.۵:

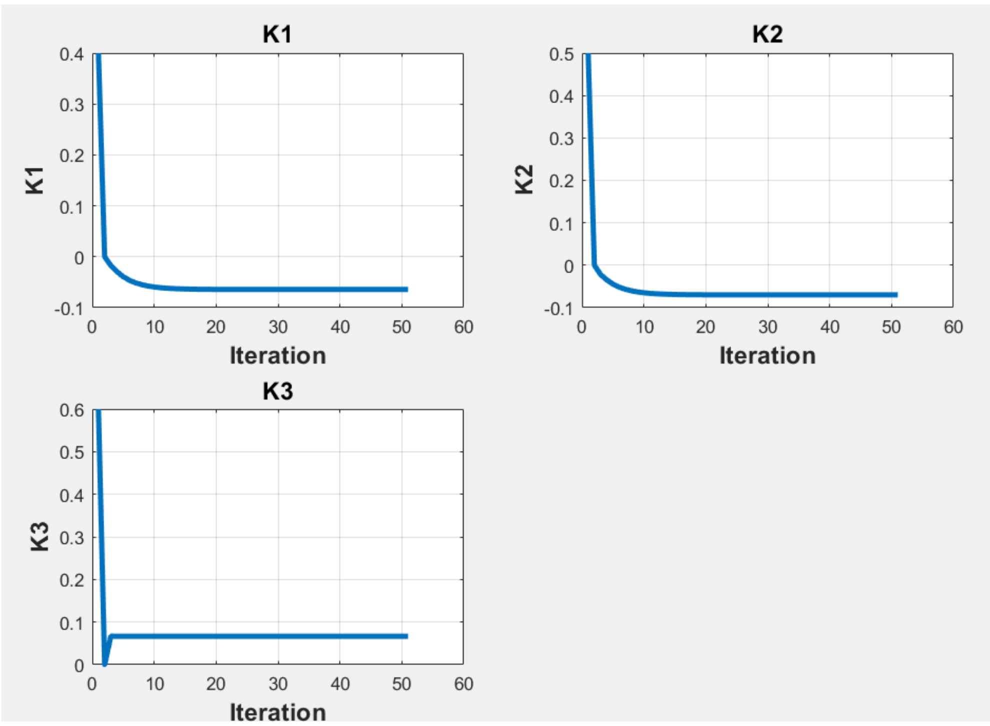


شکل ۶: مقادیر K در لاندای ۰.۵



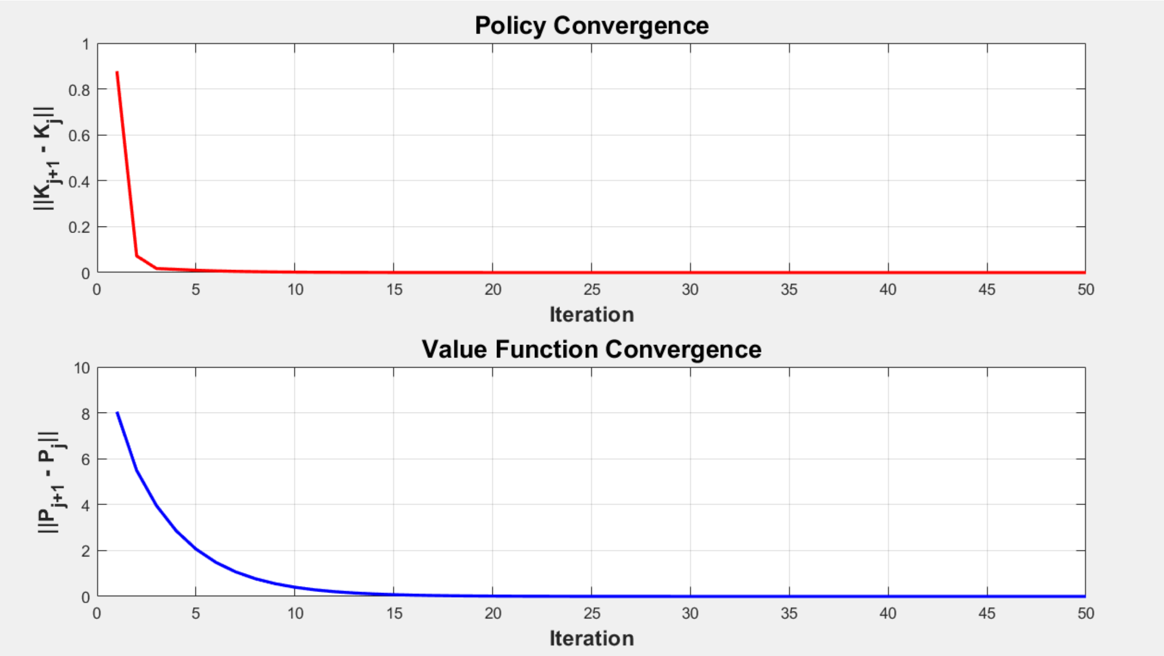
شکل ۸: همگرایی در لاندای ۰.۵

و سپس برای لاندای ۰.۹:



شکل ۷: مقادیر K در لاندای ۰.۹

و برای همگرایی داریم:



شکل ۹: همگرایی در لاندای ۰.۹

همانطور که مشاهده می‌گردد در حالتی که لاندا برابر با ۰.۹ است، وضعیت همگرایی الگوریتم بهتر است.

صورت سوال قسمت ث:

ث) در این قسمت نمودارهای $\|P_{j+1} - P_j\|$ و $\|K_{j+1} - K_j\|$ را برای نتایج حاصل از قسمت‌های ب، پ و ت (دو شکل و در هر شکل ۴ نمودار)، نسبت به شماره تکرارها رسم نموده و نتایج را بررسی نمایید. توضیح دهید که الگوریتم λ_PI چه ارتباطی با دیگر الگوریتم‌ها دارد.

پاسخ:

نمودارهای خواسته شده در بخش‌های قبلی رسم گردید.

روش lambda-PI با ارائه‌ی پارامتری به نام lambda باعث می‌شود که VI یا PI به شکلی تعمیم یافته تبدیل شوند. که تعیین می‌کند چه مقدار از تابع ارزش فعلی و بعدی در ارزیابی سیاست استفاده شود.

هرچقدر مقدار لاندا کمتر باشد و به صفر نزدیک شود، به سمت VI و هرچقدر مقدار لاندا افزایش یابد و به یک نزدیک شود، الگوریتم به سمت PI، می‌رود. که در نتایج نیز مشهود است. انتظار می‌رود PI

همگرایی سریعتر داشته باشد، در اینجا نیز وقتی مقدار لاندا را با ۰.۹ جایگزین کردیم، همگرایی سریعتر و بهتر شد.

۱.۳ سوال دوم

صورت سوال:

۲. برای مدل سوال قبل کنترل بهینه را با استفاده از الگوریتم PI زیر به دست آورید. توجه کنید که این الگوریتم در گام ارزیابی سیاست به مدل نیاز ندارد. (یک دستگاه معادلات به شکل $Ax = b$ که x مجهول باشد را می‌توانید با دستور $x = A\backslash b$ در متلب حل کنید. دقت کنید که رنک A کمتر از تعداد مجهول‌ها نباشد.)

$$(\bar{x}_k^T - \bar{x}_{k+1}^T)\bar{p}_{j+1} = r(x_k, u_k)$$

$$K_{j+1} = (R + B^T P_{j+1} B)^{-1} B^T P_{j+1} A$$

که

$$\bar{p} = [p_{11}, 2p_{12}, \cdots, 2p_{1n}, p_{22}, 2p_{23}, \cdots, 2p_{n-1,n}, p_{nn}]^T$$

$$\bar{x} = [x_1^2, x_1x_2, \cdots, x_1x_n, x_2^2, x_2x_3, \cdots, x_{n-1}x_n, x_n^2]^T$$

پاسخ:

در این سوال، برای حل مسئله نیاز داریم داده تولید کنیم. به همین ترتیب با استفاده از مدل اصلی سیستم و کد زیر داده تولید می‌کنیم:

```
%% System Definition
n = 3; % State dimension
m = 1; % Control dimension

% Generate synthetic data (replace with real sampled data if available)
num_samples = 1000;
A_true = [0.9065 0.0816 -0.0005; 0.0743 0.9012 -0.0007; 0 0 0.1327]; % True A
B_true = [-0.0027; -0.0068; 1]; % True B
x_samples = randn(n, num_samples); % Random states
u_samples = randn(m, num_samples); % Random controls
x_next_samples = A_true * x_samples + B_true * u_samples + 0.01 * randn(n, num_samples);
```

```
%% Step 1: Estimate Dynamics (A, B) from Sampled Data
% Solve x_{k+1} = A*x_k + B*u_k using least squares
X = x_samples(:, 1:end-1); % Current states x_k
U = u_samples(:, 1:end-1); % Control inputs u_k
X_next = x_next_samples(:, 1:end-1); % Next states x_{k+1}

% Construct regression problem
Theta = [X; U]; % Combine state and control inputs
W = X_next; % Target next state

% Solve for [A, B] using least squares
AB = W / Theta; % [A, B] = W * pinv(Theta)
A_est = AB(:, 1:n); % Extract A
B_est = AB(:, n+1:end); % Extract B

disp('Estimated A:');
disp(A_est);
disp('Estimated B:');
disp(B_est);
```

سپس با استفاده از کد بالا، و همانطور که در صورت سوال خواسته شده بود معادله را حل می‌کنیم.

و سپس با استفاده از حل بالا حلقه تکرار را به صورت زیر می‌نویسیم، که در آن از دینامیک تخمین زده شده استفاده می‌گردد:

```
%% Step 2: Policy Iteration with Estimated Dynamics
R = 1; % Control cost
Q = eye(n); % State cost

nP = 200; % Number of iterations
K = zeros(nP, n); % Policy (control gains)
K(1, :) = [0.4, 0.5, 0.6]; % Initial policy guess

P = cell(nP, 1); % Cost-to-go matrices
P{1} = zeros(n);

for j = 1:nP
    % Step 1: Solve for P_{j+1} using Riccati equation
    P{j+1} = (Q + K(j, :)' * R * K(j, :) + (A_est - B_est * K(j, :))' * P{j} * (A_est - B_est * K(j, :)

    % Step 2: Update Policy
    K(j+1, :) = (R + B_est' * P{j+1} * B_est)^(-1) * (B_est' * P{j+1} * A_est);

    % Display iteration progress
    disp(['Iteration ', num2str(j), ' Complete']);
end

disp('Final Policy:');
disp(K(end, :));
```

حاصل پس از ۵۰ تکرار به صورت زیر می‌شود:

```
Final Policy:
   -0.0569   -0.0631    0.0665
```

۱.۴ سوال سوم

صورت سوال:

۳. سیستم زمان پیوسته زیر را در نظر بگیرید :

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y(t) = Cx(t)$$

که در آن هدف کنترلی دنبال کردن خروجی مطلوب y_d توسط خروجی سیستم بوده و دینامیک پایدار آن به صورت زیر می‌باشد.

$$\dot{y}_d(t) = Fy_d(t)$$

برای مینیمم نمودن تابع مقدار با افق محدود، نیاز داریم تابع مقدار را به صورت کلی زیر تعریف نماییم. در این رابطه $\gamma > 0$ همان discount factor است که اثر توابع پاداش قدیمی‌تر را کم ارزش می‌نماید.

$$V(x, y_d) = \int_t^{\infty} e^{-\gamma(\tau - t)} r(x, y_d) d\tau$$

الف) برای این منظور یک ساختار مناسب برای تابع پاداش و سیگنال کنترلی پیشنهاد دهید.

ب) با تشکیل معادله حالت افزوده برای بردار حالت افزوده $X = [x^T \ \ y_d^T]^T$ و با الهام از LQR، رابطه بلمن LQT(Linear Quadratic Tracking) و تابع همیلتنین LQT را بدست آورید. و درنهایت بهره فیدبک حالت بهینه را محاسبه نمایید. (تابع مقدار دارای ساختاری کوادراتیک است.)

پ) با در نظر گرفتن ساختار تابع Q به فرم $Q = \frac{1}{2}[X^T \ \ u^T]H[X^T \ \ u^T]^T$، با استفاده از رابطه بلمن بدست آمده در قسمت ب، الگوریتم Q-learning را برای این مسئله بدست آورید.

ت) الگوریتم حاصل از قسمت پ را به فرم LS تبدیل نمایید.

قسمت الف:

با فرض رابطه مقدار که در صـــورت ســـوال آمده اســـت و مطالعه مقاله‌ی پیوســـت که خودمان یافتیم، داریم:

$$V(x, y_d) = \int_t^{\infty} e^{-\gamma(\tau - t)} r(x, y_d)\, d\tau$$

شکل ۱۰: تابع مقدار

می‌توانیم تابع پاداش را با توجه به میزان تلاش کنترلی و خطای ردیابی به صورت زیر تعریف کنیم:

$$r(x, y_d) = -\left[(Cx - y_d)^T Q_y (Cx - y_d) + u^T R u\right]$$

شکل ۱۱: تابع پاداش

که در آن Q_y و R مثبت معین هستند، Cx نیز خروجی سیستم و y_d خروجی مطلوب است. به همین ترتیب داریم:

$$V(x, y_d) = \int_t^{\infty} e^{-\gamma(\tau - t)} \left[(Cx - y_d)^T Q_y (Cx - y_d) + u^T R u\right] d\tau$$

شکل ۱۲: تابع ارزش

البته در مقاله‌ای که یافتیم نیز تعریف زیر موجود است:

$$J(x, \bar{y}_d) = \frac{1}{2} \int_t^{\infty} e^{-\gamma(\tau - t)} \left[(Cx - y_d)^T Q (Cx - y_d) + u^T R u\right] d\tau$$

where $\gamma > 0$ is the discount factor.

شکل ۱۳: تابع ارزش

جواب استاندارد مسئله LQT به صورت زیر است:

$$u = -R^{-1}B^{T}S\,x + R^{-1}B^{T}\,v_{SS}$$

شکل ۱۴: جواب استاندارد مسئله LQT

ولی برای اینکه بتوانیم با لحاظ تابع ارزشـــی که به آن رســـیدیم، این مســـئله را به فرم *quadratic* در بیاوریم سیاست را به صورت زیر تعریف می‌کنیم:

$$u = Kx + K'y_d$$

شکل ۱۵: سیگنال کنترلی

که این سیگنال *admissible fixed control policy* است. به همین ترتیب با اثبات ارائه شده در مقاله می‌توان تابع ارزش را به صورت زیر بازنویسی نمود:

$$J\left(x(t),\bar{y}_d\right) = V\left(x(t),y_d(t)\right) = \frac{1}{2}\left[x(t)^{T}\;y_d(t)^{T}\right]P\left[x(t)^{T}\;y_d(t)^{T}\right]^{T}$$

شکل ۱۶: فرم کوادراتیک تابع ارزش

قسمت پ:

با تعریف زیر:

$$X(t) = \left[x(t)^{T}\;\;y_d(t)^{T}\right]^{T}$$

شکل ۱۷: بردار حالت افزوده

و فضای حالت زیر:

$$\begin{aligned}\dot{x} &= Ax + Bu\\ y &= Cx\end{aligned}$$

شکل ۱۸: فضای حالت سیستم

و همچنین:

$$\dot{y}_d = F\, y_d$$

شکل ۱۹: دینامیک ردیابی

خواهیم داشت:

$$\dot{X} = \begin{bmatrix} A & \mathbf{0} \\ \mathbf{0} & F \end{bmatrix} X + \begin{bmatrix} B \\ \mathbf{0} \end{bmatrix} u \equiv T\,X + B_1\,u$$

شکل ۲۰: سیستم افزوده

با فرض رابطه کوادراتیک تابع ارزش به صورت زیر:

$$V\big(X(t)\big) = \frac{1}{2} X(t)^T P\, X(t)$$

شکل ۲۱: فرم کوادراتیک تابع ارزش

با اســتفاده از رابطه شــکل ۲۱ و قرار دادن آن در ســمت چپ رابطه شــکل ۱۶، و لحاظ رابطه شــکل ۲۰ داریم:

$$0 = (T\,X + B_1\,u)^T\, P\,X + X^T P(T\,X + B_1\,u) - \gamma X^T P X + X^T C_1^T Q C_1\, X + u^T R u$$

where

$$C_1 = [C \text{ - } I]$$

شکل ۲۲: ساده سازی

سیگنال کنترلی را نیز به صورت زیر اگر لحاظ کنیم:

$$u = K\,x + K' y_d = K_1\,X$$

شکل ۲۳: سیگنال کنترلی

که در آن:

$$K_1 = [K \;\; K']$$

شکل ۲۵: ضریب سیگنال کنترلی

با قرار دادن تابع شکل ۲۱ و سیگنال ۲۳ در رابطه ۲۲، رابطه بلمن LQT معادله لیاپانوف LQT افزوده را نتیجه می‌دهد:

$$(T + B_1 K_1)^T P + P(T + B_1 K_1) - \gamma P + C_1^T Q C_1 + K_1^T R K_1 = 0$$

شکل ۲۴: معادله لیاپانوف

براساس معادله شکل ۲۲، رابطه همیلتونین به صورت زیر تعریف می‌شود:

$$H(X,u,P) = (TX + B_1 u)^T PX + X^T P(TX + B_1 u) - \gamma X^T PX \\ + X^T C_1^T Q C_1 X + u^T R u$$

شکل ۲۶: همیلتونین

با فرض شکل ۲۳ و مشتق گیری از همیلتونین شکل ۲۶ داریم:

$$\frac{\partial H}{\partial u} = B_1^T PX + Ru = 0$$

شکل ۲۷: مشتق همیلتونین

پاسخ زیر بدست می‌آید:

$$K_1 = -R^{-1} B_1^T P$$

شکل ۲۸: جواب مسئله

که P در رابطه زیر صدق می‌کند:

$$0 = T^T P + PT - \gamma P + PB_1 R^{-1} B_1^T P + C_1^T Q C_1$$

شکل ۲۹: معادله ریکاتی

قسمت پ و ت:

مطابق فرضی که داشتیم یعنی:

$$V(x) = \frac{1}{2}x^T P x$$

شکل ۳۰: تابع ارزش

برای تابع ارزش حالت عمل نیز خواهیم داشت:

$$Q(x,u) = \frac{1}{2}\left(x^T Q_1 x + u^T R u + \dot{x}^T P \dot{x}\right)$$

شکل ۳۱: تابع ارزش حالت عمل

با لحاظ فضای حالت داریم:

$$Q(x,u) = \frac{1}{2}\left(x^T Q_1 x + u^T R u + (Ax+Bu)^T P (Ax+Bu)\right)$$

شکل ۳۲: تابع ارزش حالت عمل

که رابطه فوق را گسترش می‌دهیم:

$$Q(x,u) = \frac{1}{2}\left(x^T Q_1 x + u^T R u + x^T A^T P A x + 2x^T A^T P B u + u^T B^T P B u\right)$$

شکل ۳۳: تابع ارزش حالت عمل

و سپس داریم:

$$Q(x,u) = \frac{1}{2}\begin{bmatrix} x \\ u \end{bmatrix}^T \begin{bmatrix} Q_1 + A^T P A & A^T P B \\ B^T P A & R + B^T P B \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix}$$

شکل ۳۴: فرم نهایی تابع ارزش حالت عمل

که در آن:

$$H = \begin{bmatrix} H_{xx} & H_{xu} \\ H_{ux} & H_{uu} \end{bmatrix}$$

شکل ۳۵: ماتریس H

و درایه‌های آن برابرند با:

$$H_{xx} = Q_1 + A^T P A, \quad H_{xu} = H_{ux}^T = A^T P B, \quad H_{uu} = R + B^T P B$$

شکل ۳۶: درایه‌های ماتریس H

با فرض زیر:

$$Q(x(t), u(t)) = \frac{1}{2} x(t)^T Q x(t) + \frac{1}{2} u(t)^T R u(t) + \gamma \int_t^{\infty} e^{-\gamma(\tau - t)} Q(x(\tau), u(\tau)) d\tau$$

شکل ۳۸: تابع ارزش حالت عمل

و تعریف زیر:

$$z(t) = \begin{bmatrix} x(t) \\ u(t) \end{bmatrix}$$

شکل ۳۷: نعریف جدید برای تبدیل به

least square

خواهیم داشت:

$$Q(x(t), u(t)) = \frac{1}{2} z(t)^T H z(t)$$

شکل ۳۹: تعریف جدید تابع ارزش حالت عمل

بر اساس تعاریف بالا داریم:

$$z(t)^T Hz(t) = x(t)^T Qx(t) + u(t)^T Ru(t) + \gamma \int_t^{\infty} e^{-\gamma(\tau - t)} z(\tau)^T Hz(\tau) d\tau$$

شکل ۴۰: معادله نهایی

حال براساس روابطی که رسیدیم بخش ارزیابی سیاست به صورت زیر تعریف می‌شود که براساس LS قابل حل است:

$$z(t)^T H^{j+1} z(t) = x(t)^T Qx(t) + (u^j(t))^T Ru^j(t) + \gamma \int_t^{\infty} e^{-\gamma(\tau - t)} z(\tau)^T H^j z(\tau) d\tau$$

شکل ۴۲: ارزیابی سیاست

و بخش بهبود سیاست به صورت زیر تعریف می‌شود:

$$u^{j+1}(t) = -(H^j_{uu})^{-1} H^j_{ux} x(t)$$

شکل ۴۱: بهبود سیاست

بحث LS نیز به این صورت رفع می‌شود، که این الگوریتم از تکرار سیاست با استفاده از تابع Q به‌صورت آنلاین استفاده می‌کند و می‌تواند بدون نیاز به دانش دینامیک سیستم افزوده اجرا شود. این روش بر اساس استفاده از حداقل مربعات (LS) و داده‌های جمع‌آوری‌شده از مسیرهای سیستم اجرا می‌شود.

با فرض رابطه کلی زیر:

$$u(t)^T Ru(t) + x(t)^T Qx(t) = \rho(t)$$

شکل ۴۳: رابطه کلی

با اندازه گیری ρ براساس رابطه بالا و zها می‌توان بخش ارزیابی سیاست که معادله آن در شکل ۴۲ معرفی گردید، را بر اساس LS نیز حل نمود.

۱.۵ منابع

مقاله معرفی شده در سامانه LMS:

Reinforcement Q-learning for optimal tracking control of linear discrete-time systems with unknown dynamics

مقاله دوم (اصلی):

Linear Quadratic Tracking Control of Partially-Unknown Continuous-time Systems using Reinforcement Learning