دانشکده مهندسی برق

یادگیری تقویتی در کنترل

تمرین هفتم: یادگیری تقویتی در کنترل بهینه و مقاوم

استاد: دکتر سعید شمقدری

دانشجو: سیده ستاره خسروی

زمستان ۱۴۰۳

چکیده

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

واژه‌های کلیدی: یادگیری تقویتی، کنترل بهینه، کنترل مقاوم

فهرست مطالب

فهرست تصاویر و نمودارها

عنوان	صفحه

فصل ۱:   کنترل بهینه و مقاوم

۱.۱ مقدمه

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

۱.۲ سوال اول

صورت سوال:

۱) برای سیستم خطی پیوسته-زمان زیر

$$\dot{x}(t) = \begin{bmatrix} -1.01887 & 0.90506 & -0.00215 \\ 0.82225 & -1.07741 & -0.17555 \\ 0 & 0 & -1 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u(t), \qquad x(0) = \begin{bmatrix} 10 \\ -10 \\ -3 \end{bmatrix}$$

فرض کنید $Q = I_3$ و $R = 1$ هستند. یک کنترل‌کننده با رویکرد IRL برای این سیستم طراحی کنید. نتایج هر مرحله Policy Iteration (یعنی ماتریس‌های P و K) را گزارش کنید. پاسخ زمانی حالت‌های سیستم را تا ثانیه ۱۰ رسم کنید.

پاسخ:

در این بخش برای حل سوال، ابتدا لازم است مانند تمرین قبلی تابعی برای بدست آوردن مقدار P در گام ارزیابی پالیسی ایجاد کنیم. به همین منظور از رابطه زیر که مربوط به IRL است کمک می‌گیریم.

P حل مثبت معین متقارن حقیقی معادله ماتریسی لیاپانوف زیر:

$$(A - BK)^{\mathrm{T}} P + P(A - BK) = -(K^{\mathrm{T}} R K + Q) \tag{۷}$$

شکل ۱: رابطه ارزیابی پالیسی (اسلایدهای درس)

به همین ترتیب، کد زیر که مربوط به تابع این بخش است نوشته می‌شود که درون حلقه آموزش با حل آن به وسیله fmincon مقدار P بدست می‌آید.

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

سپس دینامیک سیستم را تعریف می‌کنیم.

```
clc;
clear;
close all;

%% System Definition
A = [-1.01887 0.90506 -0.00215; 0.82225 -1.07741 -0.17555; 0 0 -1];
B = [0; 0; 1];

n = size(A, 1);

R = 1;
Q = eye(n);
```

در ادامه تنظیمات فرایند آموزش را انجام می‌دهیم، تعداد دفعاتی که حلقه آموزش قرار است اجرا شود را مشخص می‌کنیم. متغیرهایی برای ذخیره مقادیر P و K تعریف می‌کنیم و یک K ابتدایی که feasible باشد نیز انتخاب می‌کنیم. تنظیمات حل fmincon نیز در این بخش مشخص می‌شود.

```
%% 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');
```

حال حلقه اصلی برنامه را می‌نویسیم.

```
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));
```

در گام اول مطابق کد بالا با استفاده از تابعی که نوشته بودیم و دستور fmincon مقدار P را بدست می‌آوریم. سپس بر اساس رابطه‌ی آمده درون اسلایدهای درس:

$$K_{i+1} = R^{-1}B^T P_i \tag{۱۰}$$

شکل ۲: بهبود پالیسی (اسلایدهای درس)

بهبود پالیسی را درون کد انجام می‌دهیم، مقادیر خطا را محاسبه کرده و ذخیره می‌کنیم و سپس در هر تکرار مقادیر P و K را نمایش می‌دهیم.

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

% 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), ')']);
disp(['K = ', num2str(K(j+1, :))]);
disp(['P = ', num2str(K(j+1, :))]);
```

در ادامه اگر خطای K، کمتر از 1e-6 باشد، الگوریتم متوقف می‌گردد و K نهایی نمایش داده می‌شود.

```
    % Convergence Check
    if delta_K < 1e-6
        break;
    end
end

disp(['Final K PI = ', num2str(K(j+1, :))]);
```

در ادامه‌ی کد دو تابع برای رسم Kها، Pها و خطای آن‌ها نوشته شده است که توضیح آن‌ها صرف نظر می‌گردد.

در کد دوم مربوط به این سوال نیز دینامیک دیفرانسیلی سیستم در حضور K بهینه که در کد اول بدست آمد، تعریف می‌شود و با استفاده از ode45 معادله آن حل شده و متغیرهای حالت را رسم می‌کنیم. در اینجا شرایط اولیه سیستم را مانند تمرین قبلی لحاظ کردیم.

```
%% Plot System Variables
A = [-1.01887 0.90506 -0.00215; 0.82225 -1.07741 -0.17555; 0 0 -1];
B = [0; 0; 1];
n = size(A, 1);

K_lqr = [-0.13523 -0.1501 0.43293];


t = 0:0.01:10;

x0 = [10 -10 -3]';

% Function to represent the system dynamics (x'(t) = Ax(t) + Bu(t))
system = @(t, x) (A - B * K_lqr) * x;

% Solve the system of ODEs using ode45
[t, x] = ode45(system, t, x0);
```

حال به بررسی خروجی‌های سیستم می‌پردازیم.

با اجرای کد اول مقدار K بهینه نهایی پس از ۴ تکرار به صورت زیر بدست می‌آید.

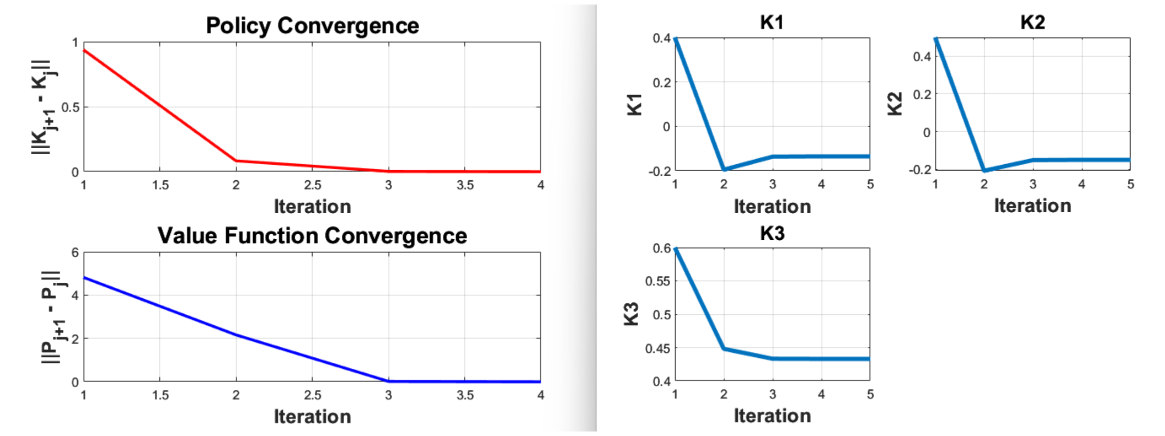
```
Final K PI = -0.13523      -0.1501      0.43293
```

شکل ۳: مقدار K بهینه

در هر تکرار نیز مقادیر K و P به صورت زیر بدست می‌آید.

```
Iteration(1)
K = -0.19517     -0.2079     0.44807
P =

ans =

    2.4647    2.2537   -0.1952
    2.2537    2.5698   -0.2079
   -0.1952   -0.2079    0.4481

Iteration(2)
K = -0.13661    -0.15148     0.43318
P =

ans =

    1.4324    1.1761   -0.1366
    1.1761    1.4428   -0.1515
   -0.1366   -0.1515    0.4332
Iteration(3)
K = -0.13523      -0.1501      0.43293
P =

ans =

    1.4245    1.1682   -0.1352
    1.1682    1.4349   -0.1501
   -0.1352   -0.1501    0.4329

Iteration(4)
K = -0.13523      -0.1501      0.43293
P =

ans =

    1.4245    1.1682   -0.1352
    1.1682    1.4349   -0.1501
   -0.1352   -0.1501    0.4329
```

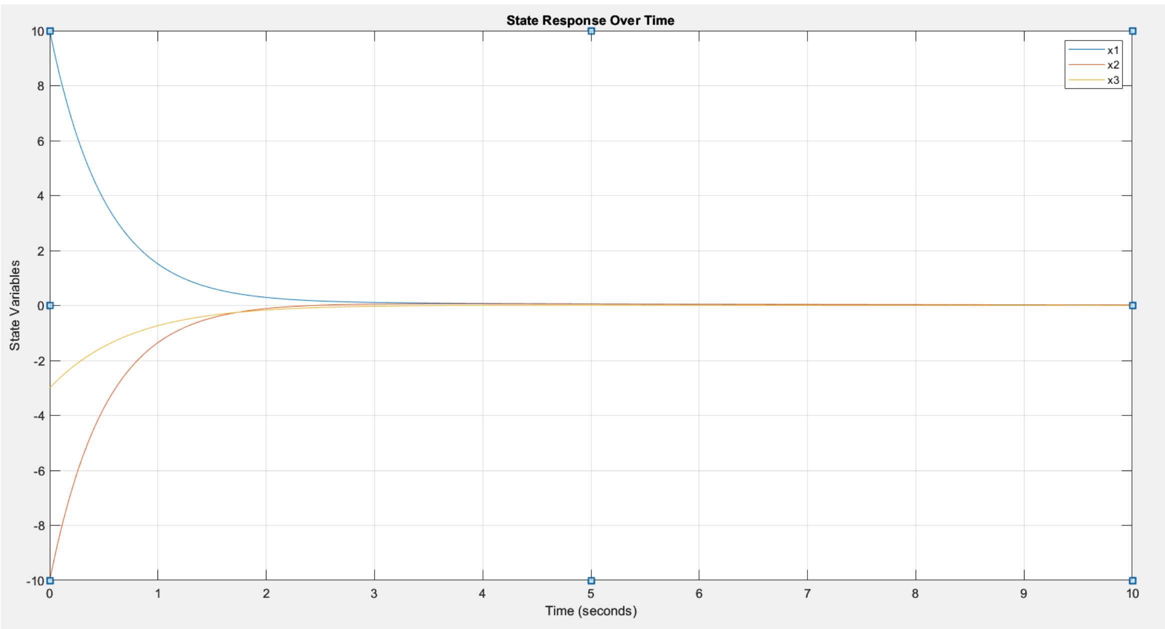
شکل ۴: مقادیر K و P در ۴ تکرار اول و همگرایی



شکل ۵: نمودار همگرایی مقادیر K و اندازه تغییرات K و P

در شکل ۵ نیز، مقادیر K در هر تکرار رسم شده و تغییرات اندازه P و K در مراحل آموزش رسم شده، که مانند آنچیزی که مقادیر عددی نشان می‌دهد همگرایی الگوریتم پس از ۴ تکرار کاملا مشهود است.

در ادامه متغیرهای حالت سیستم رسم گردیده است که به صورت شکل ۶ است.



شکل ۶: متغیرهای حالت سیستم در ۱۰ ثانیه ابتدایی

همانطور که مشاهده می‌شود متغیرهای حالت سیستم در ۳ ثانیه ابتدایی به مقدار صفر میل می‌کنند.

۱.۳ سوال دوم

صورت سوال قسمت الف:

۲) سیستم خطی پیوسته-زمان سوال قبل را با اغتشاش و به صورت زیر در نظر بگیرید

$$\dot{x}(t) = \begin{bmatrix} -1.01887 & 0.90506 & -0.00215 \\ 0.82225 & -1.07741 & -0.17555 \\ 0 & 0 & -1 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u(t) + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} d(t),$$

$$x(0) = \begin{bmatrix} 10 \\ -10 \\ -3 \end{bmatrix}$$

فرض کنید $Q = I_3$، $R = 1$ و $\beta = 5$ هستند.

الف) (با فرض دانستن مدل) ابتدا معادله ریکاتی را برای این سیستم محاسبه نموده و سپس P موجود در تابع مقدار بهینه را با استفاده از دستور care بدست آورید. این P را با P حاصل از سوال ۱ مقایسه نمایید.

پاسخ:

بر اساس مقاله پیوست برای سیستم مشابه زیر:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + B_2u(t) + B_1w(t) \\ z(t) &= Cx \end{aligned}$$

معادله GARE به صورت زیر می‌گردد:

$$A^TP + PA + Q + \gamma^{-2}PB_1B_1^TP - PB_2R^{-1}B_2^TP = 0$$

که در آن S مقدار کرنل بهینه بزرگتر مساوی صفر و H هم نماینده همان Q است.

بر اساس توضیحات سایت متلب، که در تصویر روبرو موجود است، کد مناسب را می‌نویسیم.

Solve H-infinity-like Riccati Equation

This example show how to solve the H_∞-like Riccati equation

$$A^TX + XA + X(\gamma^{-2}B_1B_1^T - B_2B_2^T)X + C^TC = 0$$

You can rewrite the equation in the format supported by `care` as follows:

$$A^TX + XA - X\underbrace{[B_1, B_2]}_{B}\underbrace{\begin{bmatrix} -\gamma^2 I & 0 \\ 0 & I \end{bmatrix}^{-1}}_{R}\begin{bmatrix} B_1^T \\ B_2^T \end{bmatrix}X + C^TC = 0$$

For any given input matrices, you can now compute the stabilizing solution X by

```
B = [B1 , B2]
m1 = size(B1,2)
m2 = size(B2,2)
R = [-g^2*eye(m1) zeros(m1,m2) ; zeros(m2,m1) eye(m2)]

X = care(A,B,C'*C,R)
```

```
%% P Calculation
A = [-1.01887 0.90506 -0.00215; 0.82225 -1.07741 -0.17555; 0 0 -1];
B2 = [0 0 1]';
B1 = [1 0 0]';
B = [B1 B2];

n = size(A , 1) ;

Q = eye(n);
beta = 5;

m1 = size(B1,2);
m2 = size(B2,2);
R = [-beta^2*eye(m1) zeros(m1,m2) ; zeros(m2,m1) eye(m2)];

[P_care,G,K]=care(A,B,Q,R);
disp('CARE P:');
disp(P_care);
```

با استفاده از کد بالا نتیجه به صورت زیر خواهد بود:

```
CARE P:
    1.6573    1.3954   -0.1661
    1.3954    1.6573   -0.1804
   -0.1661   -0.1804    0.4371
```

شکل ۷: مقادیر P براساس دستور CARE

که این مقادیر بدست آمده به مقادیر بدست آمده در سوال اول نزدیک است.

صورت سوال قسمت ب:

ب) بهره فیدبک حالت اولیه K_1^0 و K_1 را پایدارساز و بهره اولیه اغتشاش را $K_2 = K_2^0 = [0 \;\; 0 \;\; 0]$ در نظر بگیرید. نویز اکتشاف n، نویزی تصادفی در بازه $[0, 0.1]$ مفروض است. با استفاده از الگوریتم Model-free off-policy RL زیر، برای این سیستم یک کنترل کننده H_∞ طراحی نمایید.

(در این الگوریتم $Q(x) = x^T Q x$، $V_j(x) = x^T P^j x$، $u_j = K_1^j x$، $d_j = K_2^j x$ و $d = K_2 x$ و $u = K_1 x + n$ سیاست‌های رفتار هستند. شرط همگرایی را هم به ازای $\|P_{j+1} - P_j\| \leq \varepsilon$ به صورت $\varepsilon = 10^{-4}$ در نظر بگیرید.)

پاسخ:

در این قسمت ابتدا تنظیمات اولیه‌ی آموزش را تعیین می‌کنیم. (تعریف مقادیر بهره، دینامیک سیستم و ...)

```
%% Define System and Config
A = [-1.01887 0.90506 -0.00215; 0.82225 -1.07741 -0.17555; 0 0 -1];
B2 = [0 0 1]';
B1 = [1 0 0]';

Q = eye(3); R = 1; beta = 5;

% Set the number of iterations
nP = 20;

n = size(A, 1); % order
K1 = zeros(nP, n);
K2 = zeros(nP, n);

% initial values policy
K1(1, :) = [0.4 0.5 0.6];
K2(1, :) = [0 0 0];

% behavior policy
K1b = K1(1, :);
K2b = K2(1, :);
```

در ادامه تنظیمات حلقه آموزش را برای بحث انتگرال‌گیری و محاسبات P تنظیم می‌کنیم.

```
%% Learning Loop
t = 10;
dt = 0.01; % sampling time
N = t / dt; % number of samples for integral


value = 0;
xu = [0 0 0];
xd = [0 0 0];

P = cell(nP, 1);
P{1} = zeros(n);
```

در اینجا مقدار اولیه ارزش، مقدار اولیه P که البته به صورت متغیر سلولی نیز تعریف شده است برابر با صفر یا ماتریس صفر لحاظ می‌شود. (مانند تمرین قبلی و سوال اول تمرین کنونی)

سپس حلقه‌ی اصلی برنامه را می‌نویسیم.

الگوریتم صورت سوال مفروض است:

Algorithm 12 Off-Policy IRL Algorithm to Find the Solution of HJI

1: **procedure**
2: Given admissible policy u_0
3: for $j = 0, 1, \dots$ given u_j and d_j, solve for the value V_j, u_{j+1} and d_{j+1} using off-policy Bellman equation

$$V_j(x(t+T)) - V_j(x(t)) = \int_t^{t+T} \big(-Q(x) - {u_j}^T R\, u_j + \beta^2 {d_j}^T d_j - 2{u_{j+1}}^T R\,(u - u_j) + 2\beta^2 {d_{j+1}}^T\,(d - d_j)\big)d\tau,$$

on convergence, set $V_{j+1} = V_j$.
4: Go to 3.
5: **end procedure**

براساس این الگوریتم حلقه آموزش اصلی را می‌نویسیم.

ابتدا مقادیر سـای و فی که براسـاس اسـلایدهای درس لازم اسـت برای محاسـبه مقادیر بهره K1 و K2 استفاده شوند را تعریف می‌کنیم.

```
% Policy Iteration
for j = 1:nP

    x = zeros(3, N);
    x(:, 1) = [10; -10; -3];
    phi_j = [];
    sai_j = [];
```

حال حلقه محاسبه انتگرال را می‌نویسیم.

```
for t = 1:N
    ub = -K1b * x(:, t) + 0.01 * randn;
    db = K2b * x(:, t);

    % Compute target policies
    u_j = -K1(j, :) * x(:, t);
    d_j = K2(j, :) * x(:, t);

    % Compute the integrand from the Bellman equation:
    % (-Q(x) - u_j^T R u_j + β^2 d_j^T d_j - 2u_{j+1}^T R(u-u_j) + 2β^2 d_{j+1}^T(d-d_j))
    value = value + dt * (-x(:, t)' * Q * x(:, t) - u_j' * R * u_j + ...
            beta^2 * d_j' * d_j - 2 * u_j' * R * (ub - u_j) + ...
            2 * beta^2 * d_j' * (db - d_j));

    % Compute policy differences
    e1 = ub - u_j;
    e2 = db - d_j;

    % Accumulate state-action pairs for policy improvement
    xu = xu + dt * (kron(x(:, t), e1)');
    xd = xd + dt * (kron(x(:, t), e2)');

    % Update state using behavior policies
    x(:, t + 1) = x(:, t) + dt * (A * x(:, t) + B2 * ub + B1 * db);
end
```

ابتدا نویز پروب را برای محاسبه u رفتار لحاظ می‌کنیم. در مرحله بعد پالیسی هدف را محاسبه می‌کنیم. براساس الگوریتم داده شـده و رابطه انتگرالی آن، بروزرسـانی ارزش را پیاده سـازی می‌کنیم. سـپس خطا محاسبه می‌شود و در ادامه حالت‌ها براساس پالیسی‌های رفتار بروز می‌شوند.

در ادامه مقادیر فی و سای بروز شده و بر اساس آن‌ها پارامترهای L محاسبه می‌شوند.

```
phi_j = [phi_j; value];
sai_j = [sai_j; [QuadraticFeatures(x(:, 1))' - QuadraticFeatures(x(:, t))', 2 * xu * kron(eye(n),

L_params = sai_j / phi_j;
P_vector = [L_params(1) L_params(2)/2 L_params(3)/2 L_params(4)/2 L_params(5)/2 L_params(6)];
P{j+1} = VectorToSymmetricMatrix(P_vector);
delta_P = norm(P{j+1} - P{j}, 'fro');

K1(j+1, :) = [L_params(7) L_params(8) L_params(9)];
K2(j+1, :) = [L_params(10) L_params(11) L_params(12)];

if delta_P < 1e-4
    break;
end
```

و با اســـتفاده از پارامترهای L نیز بهره K برای ورودی و اغتشـــاش محاســـبه می‌شـــود. شـــرط توقف نیز براساس ماتریس P چک می‌شود.

```
%% Functions
function quadraticFeatures = QuadraticFeatures(vector)
    % Extract individual elements of the state vector
    s1 = vector(1);
    s2 = vector(2);
    s3 = vector(3);

    % Construct quadratic terms of the state vector
    quadraticFeatures = [s1^2, s1*s2, s1*s3, s2^2, s2*s3, s3^2]';
end

function symmetricMatrix = VectorToSymmetricMatrix(vector)
    % Reshape the vector into a symmetric matrix
    symmetricMatrix = [vector(1)    vector(2)/2  vector(3)/2;
                       vector(2)/2  vector(4)    vector(5)/2;
                       vector(3)/2  vector(5)/2  vector(6)];
end
```

از توابع بالا نیز برای تبدیل P در حلت برداری به P در حلت ماتریســـی متقارن و از تابع اول نیز برای محاسبه ضرب کرونکر بردار حالت در خودش استفاده می‌شود.

با اجرای کد به مقادیر زیر دست می‌یابیم.

```
P:
   -0.0889    0.0222    0.0067
    0.0222   -0.0445   -0.0067
    0.0067   -0.0067   -0.0080

K1: -0.087056    0.086511    0.026037
K2: 0   0   0
```

شکل ۸: مقادیر P و K1 و K2

صورت سوال قسمت پ:

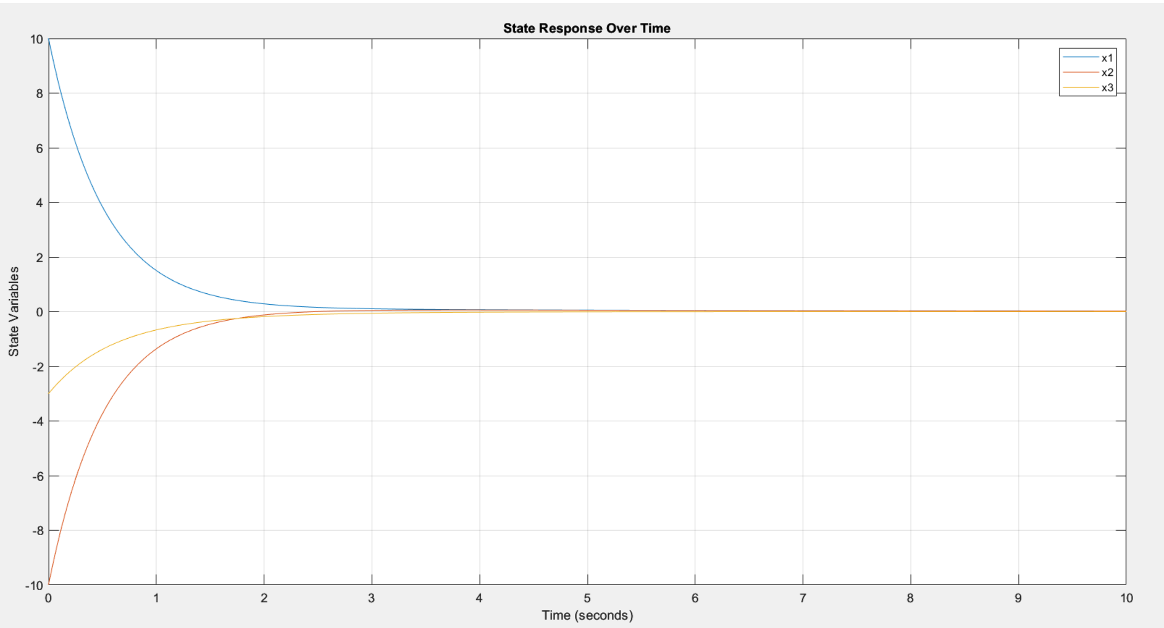
پ) پاسخ زمانی حالت‌ها را رسم کنید. پس از همگرایی نویز اکتشاف را حذف کنید.

پاسخ:

با توجه به اینکه مقدار درایه‌های بردار K2 صفر است و نویز را نیز باید حذف کنیم، بخش بعدی سوال به صورت زیر پیاده سازی گردید:

```
%% Plot System Variables
A = [-1.01887 0.90506 -0.00215; 0.82225 -1.07741 -0.17555; 0 0 -1];
B = [0; 0; 1];
n = size(A, 1);

K_1 = [-0.087052    0.086505    0.026037];


t = 0:0.01:10;

x0 = [10 -10 -3]';

% Function to represent the system dynamics (x'(t) = Ax(t) + Bu(t))
system = @(t, x) (A - B * K_1) * x;

% Solve the system of ODEs using ode45
[t, x] = ode45(system, t, x0);
```

پاسخ حالت‌ها نیز به صورت زیر است:



شکل ۹: پاسخ حالت‌ها

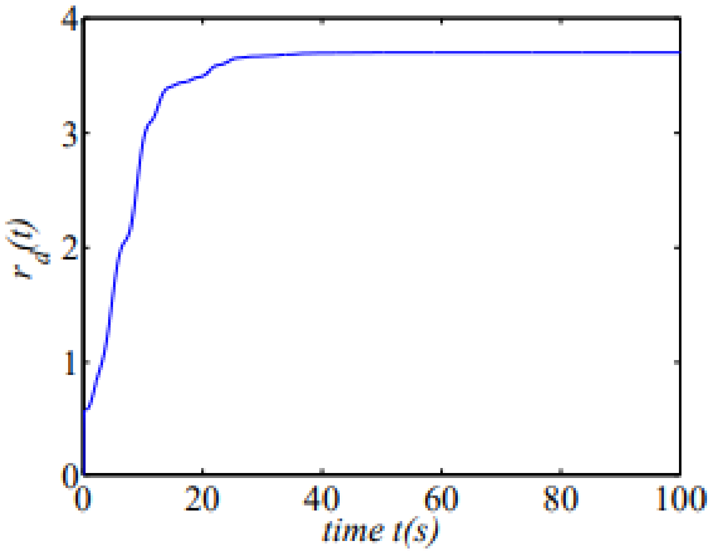
صورت سوال قسمت ت:

ت) به منظور مشاهده ضریب تضعیف واقعی، فرض کنید $w(t) = 0.2e^{-0.2t}\cos(t)$ بوده و سیگنال زیر را رسم کنید.

$$r_d(t) = \left(\frac{\int_0^t \left(\|z(\tau)\|^2 + \|u(\tau)\|_R^2\right)d\tau}{\int_0^t \|w(\tau)\|^2 d\tau}\right)^{\frac{1}{2}}$$

پاسخ:

مطابق محاسبات، نمودار این بخش به صورت زیر است و انتظار می‌رود مقدار rd به ۳.۷۰۲۴ میل کند.



شکل ۱۰: نمودار همگرایی rd

۱.۴ سوال سوم

صورت سوال:

۳) الگوریتم زیر را به صورت کامل توضیح داده، یک نام مناسب برای آن پیشنهاد داده و آن را با الگوریتم سوال ۲ مقایسه نمایید.

procedure
Given admissible policy u_0
for $j = 0, 1, \ldots$ given u_j
for $i = 0, 1, \ldots$ set $d^0 = 0$, solve for the value $V_j^{(i)}(x)$ using Bellman's equation

$$Q(x) + \left(\frac{\partial V_j^i}{\partial x}\right)^T (f(x) + g(x)u_j + h(x)d^i) + u_j^T R u_j - \beta^2 (d^i)^T d^i = 0,\ V_j^i(0) = 0,$$

$$d^{i+1} = \frac{1}{2\beta^2} h^T(x)(\frac{\partial V_j^i}{\partial x}),$$

on convergence, set $V_{j+1}(x) = V_j^i(x)$.
Update the control policy u_{j+1} using

$$u_{j+1} = -\frac{1}{2} R^{-1} g^T(x)(\frac{\partial V_{j+1}}{\partial x}).$$

Go to 3.
end procedure

پاسخ:

این الگوریتم، یک روش تکراری است و برای حل مسئله کنترلی مبتنی بر معادلات بلمن به کار می‌رود. هدف آن محاسبه تابع ارزش V(x)، کنترل بهینه و مقدار اغتشاش بهینه است. این فرآیند با استفاده از یک سیاست اولیه آغاز می‌شود و به صورت تدریجی به پاسخ‌های بهینه همگرا می‌شود.

ابتدا باید با یک سیاست اولیه مجاز و شدنی فرایند آغاز شود. مقدار اولیه اغتشاش نیز صفر لحاظ می‌شود. در این روش دو حلقه بروز رسانی داریم، در هر تکرار j از حلقه بیرونی، یک حلقه داخلی برای حل معادله بلمن وجود دارد. (معادله زیر)

$$Q(x) + \left(\frac{\partial V_j^{(i)}}{\partial x}\right)^T (f(x) + g(x)u_j + h(x)d^i) + u_j^T R u_j - \beta^2 (d^i)^T d^i = 0$$

که در آن Vj اولیه، صفر است و این معادله به صورت تحلیلی یا عددی برای تابع ارزش حل می‌شود.

سپس مرحله به روز رسانی انجام می‌گردد.
اغتشاش به صورت زیر به روز می‌گردد:

$$d^{i+1} = \frac{1}{2\beta^2} h^T(x) \left(\frac{\partial V_j^{(i)}}{\partial x} \right)$$

این به روز رسانی تا زمانی که مقدار ارزش همگرا شود ادامه می‌یابد.
سیاست کنترلی نیز از طریق رابطه زیر به روز می‌گردد:

$$u_{j+1} = -R^{-1} g^T(x) \left(\frac{\partial V_{j+1}}{\partial x} \right)$$

حلقه بیرونی نیز در حال تکرار است تا زمانی که ارزش و سیاست کنترلی همگرا شوند، به مقدار بهینه برسند.
این روش، به دلیل داشتن دو فرآیند همزمان (ارزیابی و بهبود)، به روش GPI نیز نزدیک است. می‌توان برای آن از نام Generalized Policy Iteration for H∞ Robust Control نیز استفاده نمود.

در مقایسه دو الگوریتم، الگوریتم دوم با استفاده از دو حلقه تکرار داخلی و بیرونی به دنبال یک حل دقیق‌تر و جامع‌تر است. از این رو، برای مسائلی که شامل سیستم‌های غیرخطی یا اختلالات پیچیده هستند، مناسب‌تر است. الگوریتم اول ساده‌تر است و برای حالت‌های ساده‌تر یا خطی کاربرد دارد.
در الگوریتم دوم برخلاف الگوریتم اول، معادله بلمن در یک حلقه درونی به صورت تکراری حل می‌شود، و از دو حلقه تکرار استفاده می‌گردد. رویکرد الگوریتم اول به صورت Offpolicy و IRL است.
الگوریتم دوم برخلاف قبلی از رویکرد انتگرالی اما استفاده نمی‌کند و می‌توان از آن به عنوان Offpolicy Differential RL نام برد. و همچنین احتمالا این الگوریتم برای مسئله H∞ Tracking می‌باشد.