ESE 619 Model Predictive Control **Homework 3: Optimization and Unconstrained Control**
University of Pennsylvania, Spring 2023 updated February 28, 2023
Student Name: Wei-Cheng (Rory) Huang *Penn ID: 55353592*

The solutions of this homework are entirely my own. I have discussed these problems with several classmates, they are: Shiming Liang and Yifei Shao

1. **Discounted LQR**

Consider the finite horizon discounted LQR problem

$$V_0^*(x_0) = \min_{x,u} \sum_{i=0}^{N-1} \alpha^i \left(x_i^{\mathrm{T}} Q x_i + u_i^{\mathrm{T}} R u_i\right)$$
$$\text{s.t.} \quad x_{i+1} = Ax_i + Bu_{\mathrm{i}}$$

with discount factor $\alpha \in (0,1)$, $Q = Q^{\mathrm{T}}, Q \succeq 0$, $R = R^{\mathrm{T}}$ and $R \succ 0$

(1) State the Bellman recursion (DP iteration) for this problem. [2 pts]

(2) Assuming that the optimal cost-to-go at time $i+1$ is $V_{i+1}(x_{i+1}) = \alpha^{i+1} x_{i+1}^{\mathrm{T}} P_{i+1} x_{i+1}$, find K_i as a function of P_{i+1}, such that the optimal input at time i is $u_i^*(x_i) = K_i x_i$ [4 pts]

(3) Given that the optimal input at time i is $u_i^*(x_i) = K_i x_i$ and the optimal cost-to-go at time $i+1$ is $V_{i+1}(x_{i+1}) = \alpha^{i+1} x_{i+1}^{\mathrm{T}} P_{i+1} x_{i+1}$, compute the matrix P_i such that the optimal cost-to-go at time i is $V_i^*(x_i) = \alpha^i x_i^{\mathrm{T}} P_i x_i$ [4 pts]

Solution:
a. Starting from 1-step problem: solved at time $N-1$

$$V_{N-1}^*(x_{N-1}) = \min_{u_{N-1}} \alpha^{N-1} \left(x_{N-1}^{\mathrm{T}} Q x_{N-1} + u_{N-1}^{\mathrm{T}} R u_{N-1}\right)$$
$$\text{s.t.} \quad x_N = Ax_{N-1} + Bu_{N-1}$$

The optimal control input is derived by setting gradient of $V_{N-1}^*(x_{N-1})$ to be zero:

$$2Ru_{N-1} = 0 \Rightarrow u_{N-1}^* = 0 = K_{N-1} x_{N-1}$$

Then the cost-to-go at time $N-1$, according to our notation convention, is:

$$V_{N-1}^*(x_{N-1}) = \alpha^{N-1} x_{N-1}^{\mathrm{T}} Q_{N-1} x_{N-1} \triangleq \alpha^{N-1} x_{N-1}^{\mathrm{T}} P_{N-1} x_{N-1}$$

Also note that $P_N = 0$ in this notation convention
b. Next, consider time step $N-2$

$$V_{N-2}^*(x_{N-2}) = \min_{u_{N-1}, u_{N-2}} \sum_{i=N-2}^{N-1} \alpha^{i+1} \left(x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k\right)$$
$$\text{s.t.} \quad x_{i+1} = Ax_i + Bu_i, \quad i = N-2, N-1$$

According to principle of optimality, since the solution at time $N-1$ is givem, the above formulation is equivalent to:

$$V_{N-2}^*(x_{N-2}) = \min_{u_{N-2}} \alpha^{N-2} \left(x_{N-2}^{\mathrm{T}} Q x_{N-2} + u_{N-2}^{\mathrm{T}} R u_{N-2}\right) + \alpha^{N-1} x_{N-1}^{\mathrm{T}} P_{N-1} x_{N-1}$$
$$\text{s.t.} \quad x_{N-1} = Ax_{N-2} + Bu_{N-2}$$

The optimal control input is derived by setting gradient of $V_{N-2}^*(x_{N-2})$ to be zero:

$$2\left(\alpha^{N-1} B P_{N-1} B + \alpha^{N-2} R\right) u_{N-2} + 2\alpha^{N-1} B^{\mathrm{T}} P_{N-1} A x_{N-2} = 0$$
$$\Rightarrow u_{N-2}^* = -\alpha \left(\alpha B^{\mathrm{T}} P_{N-1} B + R\right)^{-1} B^{\mathrm{T}} P_{N-1} A x_{N-2} = K_{N-2} x_{N-2}$$

Substituting u_{N-2}^* back into the expression, we can get that

$$\begin{aligned}
V_{N-2}^*\left(x_{N-2}\right) &= \alpha^{N-2}\left(x_{N-2}^{\mathrm{T}} Q x_{N-2}+u_{N-2}^{\mathrm{T}} R u_{N-2}\right)+\alpha^{N-1}\left(A x_{N-2}+B u_{N-2}\right)^{\mathrm{T}} P_{N-1}\left(A x_{N-2}+B u_{N-2}\right) \\
&= \alpha^{N-2} x_{N-2}^{\mathrm{T}}\left(\alpha A^{\mathrm{T}} P_{N-1} A+Q\right) x_{N-2}+2 \alpha^{N-2} x_{N-2}^{\mathrm{T}}\left(\alpha A^{\mathrm{T}} P_{N-1} B\right) u_{N-2} \\
&\qquad\qquad\qquad\qquad\qquad\qquad +\alpha^{N-2} u_{N-2}^{\mathrm{T}}\left(\alpha B^{\mathrm{T}} P_{N-1} B+R\right) u_{N-2}^{\mathrm{T}} \\
&= \alpha^{N-2} x_{N-2}^{\mathrm{T}}\Big[\alpha A^{\mathrm{T}} P_{N-1} A+Q+\alpha^2 A^{\mathrm{T}} P_{N-1} B\left(\alpha B^{\mathrm{T}} P_{N-1} B+R\right)^{-1} B^{\mathrm{T}} P_{N-1} A \\
&\qquad\qquad\qquad -2 \alpha^2 A^{\mathrm{T}} P_{N-1} B\left(\alpha B^{\mathrm{T}} P_{N-1} B+R\right)^{-1} B^{\mathrm{T}} P_{N-1} A\Big] x_{N-2} \\
&= \alpha^{N-2} x_{N-2}^{\mathrm{T}}\left[Q+\alpha A^{\mathrm{T}} P_{N-1} A-\alpha^2 A^{\mathrm{T}} P_{N-1} B\left(\alpha B^{\mathrm{T}} P_{N-1} B+R\right)^{-1} B^{\mathrm{T}} P_{N-1} A\right] x_{N-2} \\
&\triangleq \alpha^{N-2} x_{N-2}^{\mathrm{T}} P_{N-2} x_{N-2}
\end{aligned}$$

c. Keep propagate backward, For any time step i, we can get the similar results as stated above. Therefore:

(1) The Bellman recursion (DP iteration) for this problem is:

$$V_i^*\left(x_i\right)=\min_{u_i} \ \alpha^i\left(x_i^{\mathrm{T}} Q_i x_i+u_i^{\mathrm{T}} R u_i\right)+V_{i+1}^*\left(x_i\right)$$

More specifically, starting from time $i=N-1$, for any time step $i=n-1, \ldots, 1,0$:
Given P_{i+1}, calculate the optimal control input:

$$u_i^*=K_i x_i=-\alpha\left(\alpha B^{\mathrm{T}} P_{i+1} B+R\right)^{-1} B^{\mathrm{T}} P_{i+1} A x_i$$

and the optimal value cost-to-go:

$$V_i^*=\alpha^i x_i^{\mathrm{T}} P_i x_i \qquad \text{where} \quad P_i=Q+\alpha A^{\mathrm{T}} P_{i+1} A-\alpha^2 A^{\mathrm{T}} P_{i+1} B\left(\alpha B^{\mathrm{T}} P_{i+1} B+R\right)^{-1} B^{\mathrm{T}} P_{i+1} A$$

Then keep propagate backward until time 0 (initial time)

(2) As is mentioned above, the expression of optimal gain matrix K_i as a function of P_{i+1} is:

$$K_i=-\alpha\left(\alpha B^{\mathrm{T}} P_{i+1} B+R\right)^{-1} B^{\mathrm{T}} P_{i+1} A$$

(3) As is mentioned above, the expression of P_i as a function of P_{i+1} is:

$$P_i=Q+\alpha A^{\mathrm{T}} P_{i+1} A-\alpha^2 A^{\mathrm{T}} P_{i+1} B\left(\alpha B^{\mathrm{T}} P_{i+1} B+R\right)^{-1} B^{\mathrm{T}} P_{i+1} A$$

2. **Finite Horizon Optimal Control**

Consider the discrete-time dynamic system with the following state space representation:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 0.77 & -0.35 \\ 0.49 & 0.91 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0.04 \\ 0.15 \end{bmatrix} u(k) \tag{1}$$

We want to design a linear quadratic optimal control for this system with a finite horizon N = 50. We initially set the following cost matrices:

$$Q = \begin{bmatrix} 500 & 0 \\ 0 & 100 \end{bmatrix}, R = 1, P = \begin{bmatrix} 1500 & 0 \\ 0 & 100 \end{bmatrix}$$

nd assume that the initial state is $x(0) = \begin{bmatrix} 1 & -1 \end{bmatrix}^{\mathrm{T}}$

(1) Determine the optimal set of inputs

$$U_0 = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

through the Batch Approach, i.e. by writing the dynamic equations as follows:

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} I \\ A \\ A^2 \\ \vdots \\ A^N \end{bmatrix} x(0) + \begin{bmatrix} 0 & \cdots & \cdots & 0 \\ B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & \ddots & \ddots & 0 \\ A^{N-1}B & \cdots & AB & B \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

and using the formula:

$$U^*(x(0)) = -\left(\mathcal{S}^{u\mathrm{T}} \bar{Q} \mathcal{S}^u + \bar{R}\right)^{-1} \mathcal{S}^{u\mathrm{T}} \bar{Q} \mathcal{S}^x x(0)$$

and calculate the optimal cost $J_0^*(x(0))$:

$$J_0^*(x(0)) = x(0)^{\mathrm{T}} \left[\mathcal{S}^{x\mathrm{T}} \bar{Q} \mathcal{S}^x x - \mathcal{S}^{x\mathrm{T}} \bar{Q} \mathcal{S}^u \left(\mathcal{S}^{u\mathrm{T}} \bar{Q} \mathcal{S}^u + \bar{R}\right)^{-1} \mathcal{S}^{u\mathrm{T}} \bar{Q} \mathcal{S}^x\right] x(0)$$

(2) Verify the results of the previous point by solving an optimization problem. In fact, the cost can be written as a function of U_0 as follows:

$$\begin{aligned} J_0(x(0), U_0) &= (\mathcal{S}^x x(0) + \mathcal{S}^u U_0)^{\mathrm{T}} \bar{Q} (\mathcal{S}^x x(0) + \mathcal{S}^u U_0) + U_0{}^{\mathrm{T}} \bar{R} U_0 \\ &= U_0{}^{\mathrm{T}} H U_0 + 2x(0)^{\mathrm{T}} F U_0 + x(0)^{\mathrm{T}} \mathcal{S}^{x\mathrm{T}} \bar{Q} \mathcal{S}^x x(0) \end{aligned}$$

where $H := \mathcal{S}^{u\mathrm{T}} \bar{Q} \mathcal{S}^u + \bar{R}$ and $F := \mathcal{S}^{x\mathrm{T}} \bar{Q} \mathcal{S}^u$, and then minimized by solving an unconstrained minimization problem. Check that the optimizer U_0^* and the optimum $J_0^*(x(0), U_0^*)$ correspond to the ones determined analytically in the previous point.

Note: Computing the optimal control trajectory in this way is a very basic version of model predictive control. More in the later lectures. [5 pts]

(3) Design the optimal controller through the recursive approach and determine the optimal state-feedback matrices F_k Start from the Riccati Di˙erence Equations, assuming that $P_N = P$, and compute recursively the P_k:

$$P_k = A^{\mathrm{T}} P_{k+1} A + Q - A^{\mathrm{T}} P_{k+1} B \left(B^{\mathrm{T}} P_{k+1} B + R\right)^{-1} B^{\mathrm{T}} P_{k+1} A \tag{2}$$

and then calculate F_k s a function of P_{k+1}:

$$F_k = -\left(B^{\mathrm{T}} P_{k+1} B + R\right)^{-1} B^{\mathrm{T}} P_{k+1} A x_k$$

Compare the optimal cost $J_0^*(x(0)) = x(0)^{\mathrm{T}} P_0 x(0)$ with point (1) and check that they are equal. [5 pts]

(4) We want to compare the robustness of the two approaches in simulations. Hence, let us add a process disturbance $Dw(k)$ to the right-hand side of Equation (1). Assume the matrix D to be:

$$D = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix}$$

while the process $w(k)$ is a Gaussian white noise, with mean $m = 0$ and variance $\sigma^2 = 10$
Consider the simulation length to be equal to N time steps and that the input u_k to the system are defined as follows:

$$u_k = \begin{cases} U_0[k+1] & \text{for Batch Approach} \\ F_k x_k & \text{for Recursive Approach} \end{cases}$$

where $U_0[k]$ is the k-th component of the vector U_0.
Plot a graph of the state evolution over time. What is the di˙erence in the dynamic evolution?What happens if you modify the variance of the disturbance? Motivate your answer. [5 pts]

(5) We assume now the horizon to be $N = 5$, and want to refine the definition of P. From the lectures we have seen two possible choices, corresponding to P solution of the ARE and of the Lyapunov equation.
Calculate the two possible final cost weight, P_{Ric} and P_{Lyap} or the recursive approach, then design two new optimal controllers. Simulate the dynamic behavior of the controller for N. steps with the code, considering no disturbance. Plot the evolution of the state variables and compare the two cases, motivating the results.

Solution:
The MATLAB code for solvng all the problems are shown below, see code comments for more details

```
%% This is the script for ESE 6190 Homework 3 Ex2
clear all
clc

%% Basic Deifinition
% Define system matrices:
A = [0.77, -0.35; 0.49, 0.91];
% Define input matrices:
B = [0.04;0.15];
% Horizon Length
N = 50;
% Define weight matrices:
Q = [500,0; 0,100];
R = 1;
P_N = [1500, 0; 0, 100];
% Define initial state:
x0 = [1;-1];

%% (1) Solution via Batch appoach
[U_b,J_b] = UFHLQOC_Batch(A, B, N, Q, R, P_N, x0,"analytical");

%% (2) Solution via Batch appoach, but use quadrprog
[U_bopt,J_bopt] = UFHLQOC_Batch(A, B, N, Q, R, P_N, x0,"optimize");
%% verify the result
flag_U = norm(U_b-U_bopt)≤1e-8;
flag_J = norm(J_b-J_bopt)≤1e-8;

%% (3) Solution via Recursive approach
[U_r,J_r,F_r] = UFHLQOC_Recursive(A, B, N, Q, R, P_N, x0);
%% verify the result
flag_U_br = norm(U_b-U_r)≤1e-8;
flag_J_br = norm(J_b-J_r)≤1e-8;

%% (4) Simulate the disturbed dynamics
```

```
% Define disturbance process
rng(0)
n = size(A, 1);
D = [0.1;0.1];
w = sqrt(10)*randn(N);
x_b = zeros(n,N);
x_r = zeros(n,N);
% Simulate the system
for i = 0:N-1
    if i == 0
        x_b(:, i+1) = disturbed_system(x0, U_b(1), w(1), A, B, D);
        x_r(:, i+1) = disturbed_system(x0, F_r(1,:)*x0, w(1), A, B, D);
    else
        x_b(:, i+1) = disturbed_system(x_b(:,i), U_b(i+1), w(i+1), A, B, D);
        x_r(:, i+1) = disturbed_system(x_r(:,i), F_r(i+1,:)*x_r(:,i), w(i+1), A, ...
            B, D);
    end
end
x_b = [x0,x_b];
x_r = [x0,x_r];
t = 0:1:N;
% plot the figure
figure
plot(t, x_b(1,:), '-b', 'LineWidth', 2)
hold on
plot(t, x_b(2,:), '-r', 'LineWidth', 2)
legend('x_1b','x_2b',FontSize=12)
title('state evolution using bacth approach',FontSize=16)
xlabel('timestep k',FontSize=16)
ylabel('state',FontSize=16)
figure
plot(t, x_r(1,:), '-b', 'LineWidth', 2)
hold on
plot(t, x_r(2,:), '-r', 'LineWidth', 2)
legend('x_1r','x_2r',FontSize=12)
title('state evolution using recursive approach',FontSize=16)
xlabel('timestep k',FontSize=16)
ylabel('state',FontSize=16)

%% (5) Redesign the controller
% choose the terminal weight to be the solution of ARE
N = 5;
[P_ric,L_ric,G_ric] = idare(A,B,Q,R,0,eye(size(A)));
P_lya = dlyap(A,Q);
[U_ric,J_ric] = UFHLQOC_Batch(A, B, N, Q, R, P_ric, x0,"analytical");
[U_lya,J_lya] = UFHLQOC_Batch(A, B, N, Q, R, P_lya, x0,"analytical");
n = size(A, 1);
x_ric = zeros(n,N);
x_lya = zeros(n,N);
% Simulate the system
for i = 0:N-1
    if i == 0
        x_ric(:, i+1) = disturbed_system(x0, U_ric(1), 0, A, B, 0);
        x_lya(:, i+1) = disturbed_system(x0, U_lya(1), 0, A, B, 0);
    else
        x_ric(:, i+1) = disturbed_system(x_ric(:,i), U_ric(i+1), 0, A, B, 0);
        x_lya(:, i+1) = disturbed_system(x_lya(:,i), U_lya(i+1), 0, A, B, 0);
    end
end
x_ric = [x0,x_ric];
x_lya = [x0,x_lya];
t = 0:1:N;
% plot the figure
figure
plot(t, x_ric(1,:), '-b', 'LineWidth', 2)
hold on
plot(t, x_ric(2,:), '-r', 'LineWidth', 2)
legend('x^{ric}_1','x^{ric}_2',FontSize=12)
```

```
title('state evolution with solution of ARE',FontSize=16)
xlabel('timestep k',FontSize=16)
ylabel('state',FontSize=16)
figure
plot(t, x_lya(1,:), '-b', 'LineWidth', 2)
hold on
plot(t, x_lya(2,:), '-r', 'LineWidth', 2)
legend('x^{lya}_1','x^{lya}_2',FontSize=12)
title('state evolution with solution of Lyapunov Equation',FontSize=16)
xlabel('timestep k',FontSize=16)
ylabel('state',FontSize=16)

function [U,J] = UFHLQOC_Batch(A, B, N, Q, R, P_N, x0, mode)
    %% This is the function used to solve the unconstrained finitie time
    %% linear quadratic optimal control via batch approach
    %% input:
    %  A: system matrix; B: Input matrix; N: control horizon;
    %  Q,R: weight matrices for states and inputs, respectively;
    %  P_N: terminal cost weight matrix; x0: initial condition
    %  mode: whether to use anylytical method or optimization method.
    %% output:
    %  U: the optimal control sequence U=[u0; u1; ...; uN-1];
    %  J: the optimal cost J=x0'*P0*x0

    % first, rollout the dynamics to get Sx and Su:
    n = size(A, 1);
    m = size(B, 2);
    Sx = zeros((N+1)*n, n);
    Su = zeros((N+1)*n, N*m);
    for i = 0:N
        Sx(i*n+1:(i+1)*n, 1:n) = A^i;
        for j = 0:N-1
            Su(i*n+1:(i+1)*n, j*m+1:(j+1)*m) = (i-j-1≥0)*A^(i-j-1)*B;
        end
    end
    % Then, get the stack-up weight matrices and rearrage as quadratic form
    I = eye(N);
    Q_bar = kron(I, Q);
    Q_bar(end+1:end+n, end+1:end+n) = P_N;
    R_bar = kron(I, R);
    H = Su'*Q_bar*Su + R_bar;
    F = Sx'*Q_bar*Su;
    % Calculte the optimal controller and optimal cost
    if mode == "analytical"
        U = -H^-1*F'*x0;
        J = U'*H*U + 2*x0'*F*U + x0'*Sx'*Q_bar*Sx*x0;
    elseif mode == "optimize"
        [U,J] = quadprog(2*H,2*F'*x0);
        J = J + x0'*Sx'*Q_bar*Sx*x0;
    end
end

%% functions for recursive approach
function [U,J,F] = UFHLQOC_Recursive(A, B, N, Q, R, P_N, x0)
    %% This is the function used to solve the unconstrained finitie time
    %% linear quadratic optimal control via recursive approach
    %% input:
    %  A: system matrix; B: Input matrix; N: control horizon;
    %  Q,R: weight matrices for states and inputs, respectively;
    %  P_N: terminal cost weight matrix; x0: initial condition
    %% output:
    %  U: the optimal control sequence U=[u0; u1; ...; uN-1];
    %  J: the optimal cost J=x0'*P*x0
    %  F: the optimal gain matrices sequence F = [F0; F1; ...; FN-1]

    % Start from the last time step k=N-1 to the initial state;
    n = size(A, 1);
    m = size(B, 2);
```

```
        U = zeros(N, m);
        F = zeros(N*m,n);
        P = zeros(N*n,n);
        J = zeros(N,1);
        for i = N-1:-1:0
            if i == N-1
                P_prev = P_N;
                F_curr = -(B'*P_prev*B+R)^(-1)*B'*P_prev*A;
                F(i*m+1:(i+1)*m,1:n) = F_curr;
                P_curr = A'*P_prev*A + Q + A'*P_prev*B*F_curr;
                P(i*n+1:(i+1)*n,1:n) = P_curr;
            else
                P_prev = P((i+1)*n+1:(i+2)*n,1:n);
                F_curr = -(B'*P_prev*B+R)^(-1)*B'*P_prev*A;
                F(i*m+1:(i+1)*m,1:n) = F_curr;
                P_curr = A'*P_prev*A + Q + A'*P_prev*B*F_curr;
                P(i*n+1:(i+1)*n,1:n) = P_curr;
            end
        end
        for i = 0:N-1
            if i == 0
                U(i*m+1:(i+1)*m) = F(i*m+1:(i+1)*m,1:n)*x0;
                x = disturbed_system(x0,F(i*m+1:(i+1)*m,1:n)*x0,0,A,B,0);
            else
                U(i*m+1:(i+1)*m) = F(i*m+1:(i+1)*m,1:n)*x;
                x = disturbed_system(x,F(i*m+1:(i+1)*m,1:n)*x,0,A,B,0);
            end
        end
        J = x0'*P(1:n,1:n)*x0;
    end

    function xp = disturbed_system(xk, uk, wk, A, B, D)
        %% This is the function used to simulate single-step dynamics of the diturbed ...
            system
        %% input:
        %  A: system matrix; B: Input matrix; D: Disturbance matrix;
        %  xk: current state, uk: current input, wk: guassian white noise
        xp = A*xk + B*uk + D*wk;
    end
```

(1) The optimal controller and optimal cost calculated using bacth approach with analytical method:

```
    U_b =
        4.0818
        -3.8039
        -3.0017
        -1.6477
        -0.8106
        -0.3813
        -0.1757
        -0.0801
        -0.0364
        -0.0165
        -0.0075
        -0.0034
        -0.0015
        -0.0007
        -0.0003
        -0.0001
        -0.0001
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
```

```
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
    J_b =
        1.8723e+03
```

(2) The optimal controller and optimal cost calculated using bacth approach with optimization:

```
    U_bopt =
        4.0818
        -3.8039
        -3.0017
        -1.6477
        -0.8106
        -0.3813
        -0.1757
        -0.0801
        -0.0364
        -0.0165
        -0.0075
        -0.0034
        -0.0015
        -0.0007
        -0.0003
        -0.0001
        -0.0001
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
        -0.0000
```

```
            -0.0000
            -0.0000
            -0.0000
            -0.0000
            -0.0000
            -0.0000
            -0.0000
            -0.0000
            -0.0000
            -0.0000
            -0.0000
            -0.0000
            -0.0000
            -0.0000
            -0.0000
        J_bopt =
            1.8723e+03
```

We can see that the results we get are the same as what we get in (1) using batch approach with analytical method.

(3) The optimal cost using recursive method (dynamic programming):

```
        J_r =
            1.8723e+03
```

We can see that it is exactly the same as what we get in (1) and (2) using batch approach.

(4) Plot of the state evolution over time of the disturbed system:

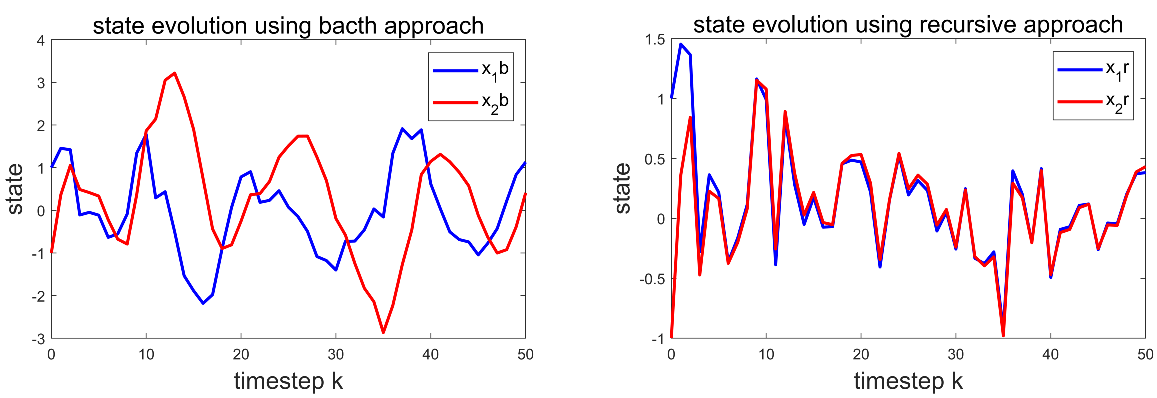

Figure 1: Evolution of the disturbed system using different approaches

From the figure we can see that recursive approach is much more robust than the batch approach. It still tend to converge even under disturbance. This is because batch approach generates the open loop control that relies on the prediction at the initial state, while recursive approach gives a closed-loop feedback policy at every time step. Therefore, when disturbance exists and largely influence the prediction of the state, batch approach can perform much better.
I also modify the variance of the disturbance, basically the reuslts are very similar to the case where variance is 10 as shown above. Only when the variance become very small (near 0, i.e. constant disturbance rather than stochastic disturbance) could batch approach converges.

(5) Plot of the state evolution over time with different terminal cost weight matrices:

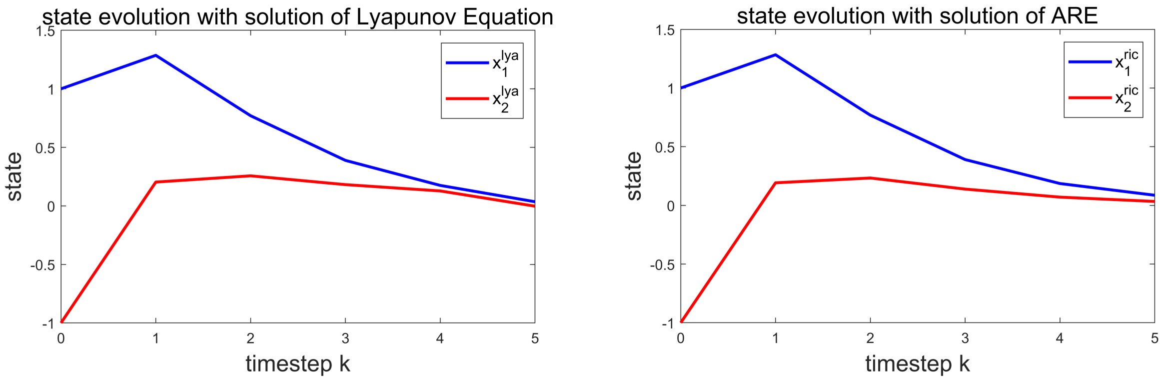

Figure 2: Evolution of the system with different terminal cost matrix

There is some differences between the two cases. Observing the matrices, final states and final control input, we can find that P_{Lyap} leads to smaller terminal state but larger control command compared to P_{Ric}. It is partly because using P_{Lyap} only make sense if the system is asmptotically stable, and hence it would hope to drive the state to original more aggresively, while using P_{Ric} is based on the assumption that no contraints will be active after the end of horizon, which is not that strict. I would say that these two cases don't show significant differences in this scenario.

3. **Stability of LQR**

Prove stability of an Infinite-Horizon LQR. Refer the slides of MPC chapter 5. [5 pts]

For inifinite-horizon LQR, we assume that we already had a convergent Riccati Difference Equation with solution P_∞:

$$P_\infty = A^{\mathrm{T}} P_\infty A + Q - A^{\mathrm{T}} P_\infty B \left(B^{\mathrm{T}} P_\infty B + R\right)^{-1} B^{\mathrm{T}} P_\infty A \tag{1}$$

and hence an valid optimal control law to be:

$$u_k^* = -\left(B^{\mathrm{T}} P_\infty B + R\right)^{-1} B^{\mathrm{T}} P_\infty A x_k = F_\infty x_k \tag{3}$$

Now, choose the candidate Lyapunov function $V(x_k) = x_k^{\mathrm{T}} P_\infty x_k$

① $V(0) = 0^{\mathrm{T}} P_\infty 0 = 0$

② $P_\infty \succ 0 \Rightarrow x_k^{\mathrm{T}} P_\infty x_k > 0, \quad \forall x_k \in \mathbb{R}^n \setminus \{0\}$

③ $V(x_{k+1}) - V(x_k) = x_{k+1}^{\mathrm{T}} P_\infty x_{k+1} - x_k^{\mathrm{T}} P_\infty x_k = (Ax_k + Bu_k)^{\mathrm{T}} P_\infty (Ax_k + Bu_k) - x_k^{\mathrm{T}} P_\infty x_k$

$$\xRightarrow{(2)} = x_k^{\mathrm{T}} \left(A^{\mathrm{T}} P_\infty A + A^{\mathrm{T}} P_\infty B F_\infty + F_\infty^{\mathrm{T}} B^{\mathrm{T}} P_\infty A + F_\infty^{\mathrm{T}} B^{\mathrm{T}} P_\infty B F_\infty - P_\infty\right) x_k$$

$$\xRightarrow{(1),(2)} = x_k^{\mathrm{T}} \left(-Q \cancel{+ P_\infty} \cancel{- A^{\mathrm{T}} P_\infty B F_\infty + A^{\mathrm{T}} P_\infty B F_\infty} + F_\infty^{\mathrm{T}} B^{\mathrm{T}} P A + F_\infty^{\mathrm{T}} B^{\mathrm{T}} P_\infty B F_\infty \cancel{- P_\infty}\right) x_k$$

$$= x_k^{\mathrm{T}} \left[-Q + F_\infty^{\mathrm{T}} B^{\mathrm{T}} P_\infty A + F_\infty^{\mathrm{T}} \left(B^{\mathrm{T}} P_\infty B + R\right) F_\infty - F_\infty^{\mathrm{T}} R F_\infty\right]$$

$$\xRightarrow{(1)} = x_k^{\mathrm{T}} \left[-Q + F_\infty^{\mathrm{T}} B^{\mathrm{T}} P_\infty A - F_\infty^{\mathrm{T}} \cancel{\left(B^{\mathrm{T}} P_\infty B + R\right) \left(B^{\mathrm{T}} P_\infty B + R\right)^{-1}} B^{\mathrm{T}} P_\infty A x_k - F_\infty^{\mathrm{T}} R F_\infty\right] x_k$$

$$= x_k^{\mathrm{T}} \left(-Q - F_\infty^{\mathrm{T}} R F_\infty\right) x_k \xRightarrow{R \succ 0, Q \succeq 0} \prec 0, \forall x_k \in \mathbb{R}^n \setminus \{0\}$$

Therefore, we prove that $V(x_k) = x_k^{\mathrm{T}} P_\infty x_k$ (the infinite horizon cost) is the true Lyapunov function of the closed loop system and the system is stable.

4. **Constrained Least Squares**

Consider the least squares problem subject to linear constraints

$$\min \frac{1}{2} x^{\mathrm{T}} Q x \quad \text{s.t.} \quad Ax = b$$

in which $x \in \mathbb{R}^n, b \in \mathbb{R}^p, Q \in \mathbb{R}^{n \times n}, Q \succeq 0, A \in \mathbb{R}^{p \times n}$. Show that this problem has a unique solution for every b and the solution is unique if and only if:

$$\operatorname{rank}(A) = p, \ \operatorname{rank}\begin{bmatrix} Q \\ A \end{bmatrix} = n$$

[5 pts]

Solution:

This is obviously a convex quadratic program, we can use KKT condition to find the least square solution we want, note that there are only equality constraints so the dual feasibility and complementarity slackness are not needed, we only need the stationary condition and primal feasibility, i.e.:

$$\begin{cases} Qx + A^{\mathrm{T}}\nu = 0 \\ Ax = b \end{cases} \Rightarrow \begin{bmatrix} Q & A^{\mathrm{T}} \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ \nu \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix}$$

And the coefficient matrix is also called the KKT matrix. A unique solution for every $b \in \mathbb{R}^p$ exists is equivalent to the KKT matrix to be nonsingular. Therefore, we are going to show that:

$$\begin{bmatrix} Q & A^{\mathrm{T}} \\ A & 0 \end{bmatrix} \text{nonsingular} \Longleftrightarrow \operatorname{rank}(A) = p, \ \operatorname{rank}\begin{bmatrix} Q \\ A \end{bmatrix} = n$$

(Sufficiency $\Rightarrow$):

$$\begin{aligned}
\begin{bmatrix} Q & A^{\mathrm{T}} \\ A & 0 \end{bmatrix} \text{ nonsingular} &\Longleftrightarrow \text{ columns of } \begin{bmatrix} Q & A^{\mathrm{T}} \\ A & 0 \end{bmatrix} \text{ linearly independent} \\
&\Rightarrow \text{ columns of } \begin{bmatrix} A^{\mathrm{T}} \\ 0 \end{bmatrix} \text{ linearly independent and} \\
&\quad \text{ columns of } \begin{bmatrix} Q \\ A \end{bmatrix} \text{ linearly independent, respectively} \\
&\Longleftrightarrow \operatorname{rank}\begin{bmatrix} A^{\mathrm{T}} \\ 0 \end{bmatrix} = p, \ \operatorname{rank}\begin{bmatrix} Q \\ A \end{bmatrix} = n \Longleftrightarrow \operatorname{rank}(A) = p, \ \operatorname{rank}\begin{bmatrix} Q \\ A \end{bmatrix} = n
\end{aligned}$$

(Necessity $\Leftarrow$):

Firstly, the rank condition given is equivalent to the conditions as follows:

$$\begin{aligned}
&\operatorname{rank}\begin{bmatrix} Q \\ A \end{bmatrix} = n \Longleftrightarrow \ x \in \mathbb{R}^n, \ \text{if} \begin{cases} Qx = 0 \\ Ax = 0 \end{cases}, x = 0 \\
&\operatorname{rank}(A) = p \Longleftrightarrow \operatorname{rank}\left(A^{\mathrm{T}}\right) = p \Longleftrightarrow y \in \mathbb{R}^p, \ \text{if } A^{\mathrm{T}} y = 0, y = 0
\end{aligned}$$

Now assume $\exists \ \begin{bmatrix} w^{\mathrm{T}} & v^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}, w \in \mathbb{R}^n, v \in \mathbb{R}^p$ such that:

$$\begin{bmatrix} Q & A^{\mathrm{T}} \\ A & 0 \end{bmatrix} \begin{bmatrix} w \\ v \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Rightarrow \begin{cases} Qw + A^{\mathrm{T}} v = 0 & ① \\ Aw = 0 & ② \end{cases}$$

Left-multiply w^{T} on both sides of ①, we get:

$$w^{\mathrm{T}} Q w + w^{\mathrm{T}} A^{\mathrm{T}} v = 0 \xrightarrow{②} \begin{cases} w^{\mathrm{T}} Q w = 0 \\ Aw = 0 \end{cases} \xrightarrow{Q \succeq 0} \begin{cases} Qw = 0 \\ Aw = 0 \end{cases}$$

And consider one of the given rank condition, we actually can make inference as follows:

$$\left.\begin{array}{r} \exists \begin{bmatrix} w \\ v \end{bmatrix}, \begin{bmatrix} Q & A^{\mathrm{T}} \\ A & 0 \end{bmatrix} \begin{bmatrix} w \\ v \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Rightarrow \begin{bmatrix} Q \\ A \end{bmatrix} w = 0 \\ \operatorname{rank} \begin{bmatrix} Q \\ A \end{bmatrix} = n \end{array}\right\} \Rightarrow w = 0$$

Still consider ①, given $w = 0$ we have:

$$Qw + A^{\mathrm{T}} v = 0 \xRightarrow{w=0} A^{\mathrm{T}} v = 0$$

Now consider the second given rank condition, the whole inference is as follows:

$$\left.\begin{array}{r} \left.\begin{array}{r} \exists \begin{bmatrix} w \\ v \end{bmatrix}, \begin{bmatrix} Q & A^{\mathrm{T}} \\ A & 0 \end{bmatrix} \begin{bmatrix} w \\ v \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Rightarrow \begin{bmatrix} Q \\ A \end{bmatrix} w = 0 \\ \operatorname{rank} \begin{bmatrix} Q \\ A \end{bmatrix} = n \end{array}\right\} \Rightarrow w = 0 \Rightarrow \quad A^{\mathrm{T}} v = 0 \\ \operatorname{rank} \begin{bmatrix} A^{\mathrm{T}} \\ 0 \end{bmatrix} = n \end{array}\right\} \Rightarrow v = 0$$

Therefore, in summary, this is stating that:

$$\left.\begin{array}{r} \exists \begin{bmatrix} w \\ v \end{bmatrix}, \begin{bmatrix} Q & A^{\mathrm{T}} \\ A & 0 \end{bmatrix} \begin{bmatrix} w \\ v \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Rightarrow \begin{bmatrix} Q \\ A \end{bmatrix} w = 0 \\ \operatorname{rank} \begin{bmatrix} Q \\ A \end{bmatrix} = n, \operatorname{rank} \begin{bmatrix} A^{\mathrm{T}} \\ 0 \end{bmatrix} = n \end{array}\right\} \Rightarrow \begin{bmatrix} w \\ v \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Rightarrow \begin{bmatrix} Q & A^{\mathrm{T}} \\ A & 0 \end{bmatrix} \text{ nonsingular}$$

With sufficiency and necessity proved, we hence prove that the KKT matrix is nonsingular if and only if the given rank condition holds.

5. **Lagrange Multipliers**

Consider the objective function $V(x) = \frac{1}{2}x^{\mathrm{T}}Hx + h^{\mathrm{T}}x$ and the optimizaiton problem is:

$$\min_{x} \; V(x)$$
$$\text{subj. to } Dx = d$$

in which $H \succ 0, x \in \mathbb{R}^m, d \in \mathbb{R}^m, m < n$. i.e. fewer constraints than decisions. Rather than partially sovling for x using the constraint and eliminating it, we make use of the method of Lagrange multipliers for treating the equality constraints.

(1) Show that the necessary and suÿcient conditions are equivalent to the matrix equation,

$$\begin{bmatrix} H & -D^{\mathrm{T}} \\ -D & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = -\begin{bmatrix} h \\ d \end{bmatrix}$$

The solution then provides the solution to the original problem.[6 pts]

Solution:

Since $H \succ 0$. This is obviosuly a strictly convex QP and the optimal solution can be decided by the KKT condition, note that there are only equality constraints so the dual feasibility and complementarity slackness are not needed, we only need the stationary condition and primal feasibility, i.e.:

$$\begin{cases} \nabla f - \nabla g^{\mathrm{T}}\lambda = 0 \\ g = 0 \end{cases} \Rightarrow \begin{cases} Hx + h^{\mathrm{T}} - D^{\mathrm{T}}\lambda = 0 \\ Dx - d = 0 \end{cases} \Rightarrow \begin{bmatrix} H & -D^{\mathrm{T}} \\ -D & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = -\begin{bmatrix} h \\ d \end{bmatrix}$$

(2) We note one other important feature of the Lagrange multiplier, their relationship to the ptimal cost of the purely quadratic case. For $h = 0$ the cost is given by:

$$V^0 = \frac{1}{2}\left(x^0\right)^{\mathrm{T}} Hx^0$$

Show that this can also be expressed in terms of λ^0 by the following:

$$V^0 = \frac{1}{2}d^{\mathrm{T}}\lambda^0$$

Solution:

According to (1), with $h = 0$, we know that x^0 and λ_0 satisfy that:

$$\begin{cases} Hx^0 - D^{\mathrm{T}}\lambda^0 = 0 \\ Dx^0 - d = 0 \end{cases} \Rightarrow \begin{cases} Hx^0 = D^{\mathrm{T}}\lambda^0 \\ Dx^0 = d \end{cases}$$

Therefore, substituting the above relation into the objective, we get:

$$V^0 = \frac{1}{2}\left(x^0\right)^{\mathrm{T}} Hx^0 = \frac{1}{2}\left(x^0\right)^{\mathrm{T}} D^{\mathrm{T}}\lambda^0 = \frac{1}{2}d^{\mathrm{T}}\lambda^0$$