Course Name: Reinforcement Learning for Engineers

Course Number and Section: 16:332:515:01

Instructor: Professor Zoran Gajic

Date Submitted: 12/23/2023

Submitted by: Dhruv Rana (RUID: 209001759)

Contents

Reinforcement Learning Policy Iterations for Nash Differential Games

1 Introduction

Reinforcement Learning (RL) has emerged as a powerful paradigm in artificial intelligence, allowing agents to learn optimal strategies in dynamic environments through interaction and feedback. Simultaneously, Nash Differential Games introduces a strategic layer to multi-agent systems, where each agent aims to optimize its actions considering the dynamic responses of others. Integrating these two domains, Reinforcement Learning Policy Iterations for Nash Differential Games presents a novel approach to solving complex decision-making problems involving multiple interacting agents.

In Nash Differential Games, the challenge lies in navigating the intricate interplay between agents with conflicting objectives. Traditional RL methods may struggle to capture the strategic nature of such interactions, necessitating the development of specialized techniques. This integration addresses scenarios where agents must not only adapt to an ever-changing environment but also strategically respond to the actions of other intelligent entities.

The core concept revolves around the application of policy iteration—a fundamental RL technique—in the context of Nash Differential Games. Each agent, equipped with its own policy, engages in an iterative process of refining its strategy based on the observed outcomes of the system. The overarching goal is to discover a Nash Equilibrium, a state where no agent has an incentive to unilaterally deviate from its chosen policy, given the strategies of the other agents.

This synthesis of RL and Nash Differential Games holds promise for addressing real-world problems characterized by strategic interactions among decision-making entities. Applications range from autonomous systems coordinating actions in dynamic environments to economic scenarios where multiple stakeholders seek to optimize their interests. This exploration delves into a solution to the differential non-zero sum Nash game (conflict of interest game with simultaneous decision making) using dynamic programming and solves the corresponding linear quadratic Nash differential game by employing policy iteration as a means to converge towards a stable Nash Equilibrium.

2 Problem Formulation

Consider a linear, continuous-time, time-invariant system represented in its state space form controlled by two agents

$$\frac{dx(t)}{dt} = Ax(t) + B_1u_1(t) + B_2u_2(t), \quad x(t_0) = x_0 \tag{1}$$

Where our goal is to minimize two scalar quadratic performance criteria, associated to each player and to this dynamic system, are given by

$$J_1(u_1(t), u_2(t), x(t)) = \frac{1}{2}\int_t^\infty (x(t)^TQ_1x(t) + u_1(t)^TR_{11}u_1(t) + u_2(t)^TR_{12}u_2(t))dt \tag{2}$$

$$J_2(u_1(t), u_2(t), x(t)) = \frac{1}{2}\int_t^\infty (x(t)^TQ_2x(t) + u_1(t)^TR_{21}u_1(t) + u_2(t)^TR_{22}u_2(t))dt \tag{3}$$

The optimal solution to the given optimization problem defined in (1)-(3) leads to the Nash optimal strategies u_1^* and u_2^* which satisfies

$$J_1(u_1^*, u_2^*) <= J_1(u_1, u_2^*) \quad , \quad J_1(u_1^*, u_2^*) <= J_1(u_1^*, u_2)$$

The Hamiltonian for the Nash differential games for each control agent are given by

$$H_1(x, u_1, u_2^*, \frac{\partial J_1^*}{\partial x}) = \frac{1}{2}[x^TQ_1x + u_1^TR_{11}u_1 + u_2^{*T}R_{12}u_2^*] + \left(\frac{\partial J_1^*}{\partial x}\right)^T[Ax + B_1u_1 + B_2u_2^*] \tag{4}$$

$$H_2(x, u_1^*, u_2, \frac{\partial J_2^*}{\partial x}) = \frac{1}{2}[x^TQ_2x + u_1^{*T}R_{21}u_1^* + u_2^TR_{22}u_2] + \left(\frac{\partial J_2^*}{\partial x}\right)^T[Ax + B_1u_1^* + B_2u_2] \tag{5}$$

minimizing with respect to u leads to control given by

$$u_1^* = -R_{11}B_1^T\left(\frac{\partial J_1^*}{\partial x}\right)^T \tag{6}$$

$$u_2^* = -R_{22}B_2^T\left(\frac{\partial J_2^*}{\partial x}\right)^T \tag{7}$$

To solve this optimization problem we will utilize the algorithm proposed by Li and Gajic,1995 [1]

Step 0 : Take a stabilizable control input $u_1^{(0)}(x(t))$ and $u_2^{(0)}(x(t))$, for example $u_1^{(0)}(x(t)) = -R_{11}B_1^T K_1^{(0)} x(t)$ and $u_2^{(0)}(x(t)) = -R_{22}B_2^T K_2^{(0)} x(t)$. Where $k_i, i = 1, 2$ must satisfy the coupled algebraic Riccati equations

$$0 = K_1 A + A^T K_1 + Q1 - K_1 S_{11} K_1 - K_2 S_{22} K_1 - K_1 S_{22} K_2 + K_2 S_{12} K_2 \tag{8}$$

$$0 = K_2 A + A^T K_2 + Q2 - K_2 S_{22} K_2 - K_2 S_{11} K_1 - K_1 S_{11} K_2 + K_1 S_{21} K_1 \tag{9}$$

where $S_{ij} = B_j R_{jj}^{-1} R_{ij} R_{jj}^{-1} B_j^T$

To find initial stable $K_1^{(0)}$ and $K_2^{(0)}$, either triple $(A, B_1, \sqrt{Q_1})$ or $(A, B_2, \sqrt{Q_2})$ must be stabilizable-detectable. Then a unique solution of the following Riccati equation

$$0 = K_1^{(0)} A + A^T K_1^{(0)} + Q1 - K_1^{(0)} S_{11} K_1^{(0)} \tag{10}$$

exists such that $(A - S_{11}K_1^{(0)})$ is a stable matrix. By Plugging $K_1 = K_1^{(0)}$ into (9) we get the second Riccati equation where the closed-loop matrix $\left(A - S_{11}K_1^{(0)} - S_{22}K_2^{(0)}\right)$ is stable

$$0 = K_2^{(0)}(A - S_{11}K_1^{(0)}) + (A - S_{11}K_1^{(0)})^T K_2^{(0)} + (Q2 + K_1^{(0)} S_{21} K_1^{(0)}) - K_2^{(0)} S_{22} K_2^{(0)} \tag{11}$$

Step 1 : Substitute the control inputs and evaluate the expression for the performance criterion, along the trajectories of the system

$$J_1^{(0)} = \frac{1}{2}\int_t^{\infty} x^T [Q_1 + K_1^{(0)} S_{11} K_1^{(0)} + K_2^{(0)} S_{12} K_2^{(0)}] x \quad dt \tag{12}$$

$$J_2^{(0)} = \frac{1}{2}\int_t^{\infty} x^T [Q_2 + K_1^{(0)} S_{21} K_1^{(0)} + K_2^{(0)} S_{22} K_2^{(0)}] x \quad dt \tag{13}$$

$$x\dot{(t)} = \left(A - S_{11}K_1^{(0)} - S_{22}K_2^{(0)}\right) x(t) \tag{14}$$

We can find the expression $\left(\frac{\partial J_1^{(0)}}{\partial x}(t)\right)$ and $\left(\frac{\partial J_2^{(0)}}{\partial x}(t)\right)$ using the formulas (12)-(14)

$$\begin{aligned} \frac{dJ_1^{(0)}}{dt} &= -\frac{1}{2}x^T[Q_1 + K_1^{(0)}S_{11}K_1^{(0)} + K_2^{(0)}S_{12}K_2^{(0)}]x \\ \frac{\partial J_1^{(0)}}{\partial x}\frac{dx}{dt} &= -\frac{1}{2}x^T[Q_1 + K_1^{(0)}S_{11}K_1^{(0)} + K_2^{(0)}S_{12}K_2^{(0)}]x \\ \frac{\partial J_1^{(0)}}{\partial x}\left(A - S_{11}K_1^{(0)} - S_{22}K_2^{(0)}\right) &= -\frac{1}{2}x^T[Q_1 + K_1^{(0)}S_{11}K_1^{(0)} + K_2^{(0)}S_{12}K_2^{(0)}]x \end{aligned} \tag{15}$$

The solution to this partial differential equation has the form

$$\begin{aligned} J_1^{(0)} &= -\frac{1}{2}x^T K_1^{(1)} x \\ \frac{\partial J_1^{(0)}}{\partial x} &= K_1^{(1)} x \end{aligned} \tag{16}$$

By using the fact that and the standard symmetrization technique $x^T M x = \frac{1}{2}x^T(M + M^T)x$ we get

$$x^T K_1^{(1)}\left(A - S_{11}K_1^{(0)} - S_{22}K_2^{(0)}\right)x = -\frac{1}{2}x^T[Q_1 + K_1^{(0)}S_{11}K_1^{(0)} + K_2^{(0)}S_{12}K_2^{(0)}]x \tag{17}$$

$$\begin{aligned} \left(A - S_{11}K_1^{(0)} - S_{22}K_2^{(0)}\right)^T K_1^{(1)} + K_1^{(1)}\left(A - S_{11}K_1^{(0)} - S_{22}K_2^{(0)}\right) = \\ -\left[Q_1 + K_1^{(0)}S_{11}K_1^{(0)} + K_2^{(0)}S_{12}K_2^{(0)}\right] \end{aligned} \tag{18}$$

performing operations (15)-(18) again for second control agent and (21) along (16) we get

$$\begin{aligned} \frac{\partial J_1^{(0)}}{\partial x}(t) = K_1^{(1)}x(t) \rightarrow u_1^{(1)} &= -R_{11}B_1^T K_1^{(1)} x(t) \\ \frac{\partial J_2^{(0)}}{\partial x}(t) = K_2^{(1)}x(t) \rightarrow u_2^{(2)} &= -R_{22}B_2^T K_2^{(1)} x(t) \end{aligned} \tag{19}$$

Step 2 : For the known value of $\left(\frac{\partial J_1^{(0)}}{\partial x}(t)\right)$ and $\left(\frac{\partial J_2^{(0)}}{\partial x}(t)\right)$ find a new approximation for the control law by minimizing the Hamiltonian respect to u

$$H_1(x, u_1, u_2^*, \frac{\partial J_1^{*}{}^{(0)}}{\partial x}) = \frac{1}{2}[x^T Q_1 x + u_1^T R_{11} u_1 + u_2^{*T} R_{12} u_2^*] + \left(\frac{\partial J_1^*}{\partial x}\right)^{(0)^T} [Ax + B_1 u_1 + B_2 u_2^*]$$
$$H_2(x, u_1^*, u_2, \frac{\partial J_2^{*}{}^{(0)}}{\partial x}) = \frac{1}{2}[x^T Q_2 x + u_1^{*T} R_{21} u_1^* + u_2^T R_{22} u_2] + \left(\frac{\partial J_2^*}{\partial x}\right)^{(0)^T} [Ax + B_1 u_1^* + B_2 u_2] \tag{20}$$

The minimization produces a stabilizing control given by

$$\begin{aligned} u_1^{(1)}(t) &= -R_{11} B_1^T \left(\frac{\partial J_1}{\partial x}(t)\right)^{(0)} \\ u_2^{(1)}(t) &= -R_{22} B_2^T \left(\frac{\partial J_2}{\partial x}(t)\right)^{(0)} \end{aligned} \tag{21}$$

Algorithm : Solve the following Lyapunov iterations with initial conditions $K_1^{(0)}$ and $K_2^{(0)}$ from (10)-(11) until we converge.

$$\begin{aligned} &\left(A - S_{11}K_1^{(i)} - S_{22}K_2^{(i)}\right)^T K_1^{(i+1)} + K_1^{(i+1)} \left(A - S_{11}K_1^{(i)} - S_{22}K_2^{(i)}\right) = \\ &\qquad\qquad -[Q_1 + K_1^{(i)} S_{11} K_1^{(i)} + K_2^{(i)} S_{12} K_2^{(i)}] \\ &\left(A - S_{11}K_1^{(i)} - S_{22}K_2^{(i)}\right)^T K_2^{(i+1)} + K_2^{(i+1)} \left(A - S_{11}K_1^{(i)} - S_{22}K_2^{(i)}\right) = \\ &\qquad\qquad -[Q_1 + K_1^{(i)} S_{21} K_1^{(i)} + K_2^{(i)} S_{22} K_2^{(i)}] \end{aligned} \tag{22}$$

3 Simulation Results

This section presents the results that were obtained in simulation while finding the statefeedback controllers that correspond to the Nash equilibrium solution of the differential game. We use the system from Vrabie and Lewis [2] on page 325 of the book chapter. The purpose of the design is to allow the two players to determine the control strategies that satisfy the equilibrium by means reinforcement learning techniques characterized by (22). The matrices of the model of the plant, that are used in this simulation are:

$$A = \begin{bmatrix} -0.0366 & 0.0271 & 0.0188 & -0.4555 \\ 0.0482 & -1.0100 & 0.0024 & -4.0208 \\ 0.1002 & 0.2855 & -0.7070 & 1.3229 \\ 0 & 0 & 1.0000 & 0 \end{bmatrix} \tag{23}$$

$$\begin{aligned} B_1 &= \begin{bmatrix} 0.4422 & 3.0447 & -5.52 & 0 \end{bmatrix}.T \\ B_2 &= \begin{bmatrix} 0.1761 & -7.5922 & 4.99 & 0 \end{bmatrix}.T \end{aligned} \tag{24}$$

With the following cost function parameters $R_{11} = 1$, $R_{22} = 2$, $R_{12} = 0.25$, $R_{21} = 0.6$, $Q_1 = diag(3.5; 2; 4; 5)$, $Q_2 = diag(1.5; 6; 3; 1)$. The closed loop system was excited with an initial condition, the initial state of the system being x_0 =[0, 0, 0, 1].

By using the Python code from Appendix A with 11 iterations we obtain the solutions for K_1 and K_2

$$K_1 = \begin{bmatrix} 7.65859301 & 0.64378599 & 0.63982315 & -3.08312621 \\ 0.64378599 & 0.28775293 & 0.28554716 & -0.09447086 \\ 0.63982315 & 0.28554716 & 0.56200666 & 0.22701676 \\ -3.08312621 & -0.09447086 & 0.22701676 & 6.69871617 \end{bmatrix} \tag{25}$$

$$K_2 = \begin{bmatrix} 3.45787531 & 0.15681142 & 0.20465317 & -1.84796418 \\ 0.15681142 & 0.62347817 & 0.28894605 & -0.07113316 \\ 0.20465317 & 0.28894605 & 0.40137661 & 0.07289788 \\ -1.84796418 & -0.07113316 & 0.07289788 & 3.78502749 \end{bmatrix}$$

This matches perfectly with the result obtained in [2]. Having obtained the solutions the optimal feedback controls can be now expressed in the feedback form

$$\begin{aligned} u_1^{opt}(t) &= -R_{11}B_1^T K_1 x^{opt}(t) = -F_1^{opt} x^{opt}(t) \\ u_2^{opt}(t) &= -R_{22}B_2^T K_2 x^{opt}(t) = -F_2^{opt} x^{opt}(t) \end{aligned} \tag{26}$$

where the optimal system dynamic equation is given by

$$\begin{aligned} \dot{x(t)} &= \left(A - S_{11}K_1^{(0)} - S_{22}K_2^{(0)}\right) x(t) = \left(A - B_1F_1^{opt} - B_2F_2^{opt}\right) x^{opt}(t) \\ \rightarrow x^{opt}(t) &= e^{\left(A - S_{11}K_1^{(0)} - S_{22}K_2^{(0)}\right)(t-t_0)} x_0 \end{aligned} \tag{27}$$

Again using the code from Appendix A we can get the approximate state trajectories (Figure 1) and feedback controls (Figure 2).

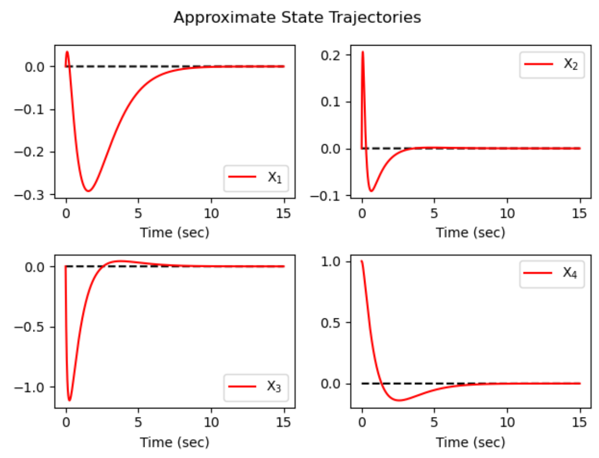

Figure 1: Approximate State Trajectories

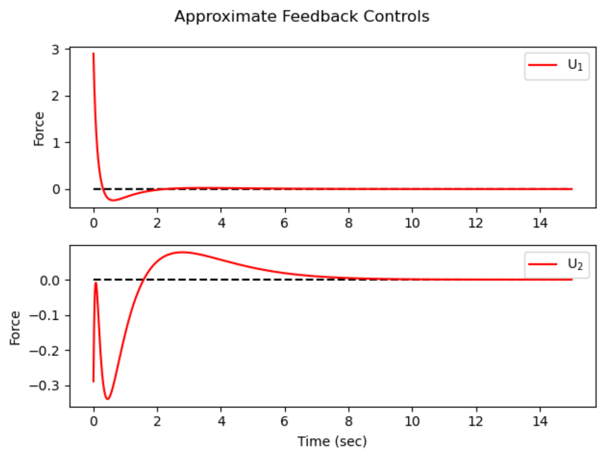

Figure 2: Approximate Feedback Controls

In order to find the optimal performance for each player we substitute their optimal controls into

$$\begin{aligned} J_1^{opt} &= \frac{1}{2}\int_t^\infty x^{opt^T}[Q_1 + K_1 S_{11} K_1 + K_2 S_{12} K_2]x \quad dt \\ J_2^{opt} &= \frac{1}{2}\int_t^\infty x^{opt^T}[Q_2 + K_1 S_{21} K_1^{(0)} + K_2 S_{22} K_2]x \quad dt \end{aligned} \tag{28}$$

$$\begin{aligned} J_1^{opt} &= \frac{1}{2}x_0^T \int_t^\infty e^{(A-S_{11}K_1-S_{22}K_2)^T(t-t_0)}[Q_1 + K_1 S_{11} K_1 + K_2 S_{12} K_2]e^{(A-S_{11}K_1-S_{22}K_2)(t-t_0)}x_0 dt \\ J_2^{opt} &= \frac{1}{2}x_0^T \int_t^\infty e^{(A-S_{11}K_1-S_{22}K_2)^T(t-t_0)}[Q_2 + K_1 S_{21} K_1 + K_2 S_{22} K_2]e^{(A-S_{11}K_1-S_{22}K_2)(t-t_0)}x_0 dt \end{aligned} \tag{29}$$

This has the solution of form as we have seen before

$$\begin{aligned} J_1^{opt} &= \frac{1}{2}x_0^T K_1 x_0 \\ J_2^{opt} &= \frac{1}{2}x_0^T K_1 x_0 \end{aligned} \tag{30}$$

Using the python code from Appendix A we can find the value of Performance Criteria Per Iteration (Table 1)

iterations	J1	J2
iteraion 0	3.467780	2.100135
iteraion 1	3.341168	1.910400
iteraion 2	3.350130	1.886189
iteraion 3	3.348513	1.892976
iteraion 4	3.349377	1.892118
iteraion 5	3.349302	1.892528
iteraion 6	3.349359	1.892492
iteraion 7	3.349355	1.892514
iteraion 8	3.349358	1.892513
iteraion 9	3.349358	1.892514
iteraion 10	3.349358	1.892514
optimal =	3.349358	1.892514

Table 1: Performance Criteria Per Iteration

4 Conclusion

In conclusion, the exploration of Lyapunov Iterations within the framework of Nash Differential Games has provided valuable insights into addressing the challenges of strategic decision-making in multi-agent systems. The integration of Lyapunov stability theory, a powerful tool in control theory, with the dynamics of Nash games presents a novel approach to achieving stability and convergence in the face of competing objectives among intelligent agents.

Through this exploration, it becomes evident that Lyapunov Iterations provide a formal and rigorous foundation for analyzing the long-term behavior of Nash equilibria. This not only enhances our theoretical understanding of strategic interactions but also opens avenues for developing robust control strategies in multi-agent systems.

References

[1] T-Y. Li and Z. Gajic. Lyapunov iterations for solving coupled algebraic riccati equations of nash differential games and algebraic riccati equations of zero-sum games. In Geert Jan Olsder, editor, *New Trends in Dynamic Games and Applications*, pages 333–351, Boston, MA, 1995. Birkhäuser Boston.

[2] Draguna Vrabie and Frank Lewis. *Integral Reinforcement Learning for Finding Online the Feedback Nash Equilibrium of Nonzero-Sum Differential Games*, pages 313–330. 01 2011.

A Python Code

```python
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt
import itertools
import pandas
from tabulate import tabulate

A = np.array([[-0.0366,  0.0271,  0.0188, -0.4555],
              [ 0.0482, -1.0100,  0.0024, -4.0208],
              [ 0.1002,  0.2855, -0.7070,  1.3229],
              [ 0     ,  0     ,  1.0000,  0     ]])

B1 = np.array([[0.4422,  3.0447, -5.52, 0]]).T
B2 = np.array([[0.1761, -7.5922,  4.99, 0]]).T

X0 = np.array([[0, 0, 0, 1]]).T

Q1_diag = [3.5,2.0,4.0,5.0]
Q2_diag = [1.5,6.0,3.0,1.0]

R11 = np.array([[1.0]])
R12 = np.array([[0.25]])
R21 = np.array([[0.6]])
R22 = np.array([[2.0]])

Q1 = np.array(np.diag(Q1_diag))
```

```python
Q2 = np.array(np.diag(Q2_diag))

S11 = np.array(B1 * np.linalg.inv(R11) * R11 * np.linalg.inv(R11) *
    B1.T)
S21 = np.array(B1 * np.linalg.inv(R11) * R21 * np.linalg.inv(R11) *
    B1.T)

S12 = np.array(B2 * np.linalg.inv(R22) * R12 * np.linalg.inv(R22) *
    B2.T)
S22 = np.array(B2 * np.linalg.inv(R22) * R22 * np.linalg.inv(R22) *
    B2.T)

# Inital K1 and K2 according to (Li and Gajic, 1995)
K1_0= sp.linalg.solve_continuous_are(
    A,
    B1,
    Q1,
    R11)

K1_0 = np.array(K1_0)

K2_0= sp.linalg.solve_continuous_are(
    (A - np.matmul(S11,K1_0)) ,
    B2 ,
    Q2 + np.matmul(K1_0,np.matmul(S21,K1_0)) , R22)

K2_0 = np.array(K2_0)

```

```
max_iterations = 11

K1 = [K1_0]
K2 = [K2_0]

J1 = np.zeros((max_iterations))
J2 = np.zeros((max_iterations))

F1 = np.zeros((max_iterations+1, len((np.linalg.inv(R11)* np.matmul(
    B1.T,(K1[0]))).flatten())))
F2 = np.zeros((max_iterations+1, len((np.linalg.inv(R22)* np.matmul(
    B2.T,(K2[0]))).flatten())))

F1[0] = np.linalg.inv(R11)* np.matmul(B1.T,(K1[0]))
F2[0] = np.linalg.inv(R22)* np.matmul(B2.T,(K2[0]))

for i in range(1,max_iterations+1):
    # Solve using Lyapunov Iterations (Li and Gajic, 1995)
    next_K1 = sp.linalg.solve_lyapunov(
        (A - np.matmul(S11,K1[i-1]) - np.matmul(S22,K2[i-1])).T,
        -(Q1 + (np.matmul(K1[i-1],np.matmul(S11,K1[i-1]))) + (np.
    matmul(K2[i-1],np.matmul(S12,K2[i-1]))) ) )
    next_K2 = sp.linalg.solve_lyapunov(
        (A - np.matmul(S11,K1[i-1]) - np.matmul(S22,K2[i-1])).T,
        -(Q2 + (np.matmul(K1[i-1],np.matmul(S21,K1[i-1]))) + (np.
    matmul(K2[i-1],np.matmul(S22,K2[i-1]))) ) )

    K1.append(np.array( next_K1 ))
    K2.append(np.array( next_K2 ))

    J1[i-1] = (1/2) * np.matmul ( X0.T , np.matmul(K1[i] , X0) )
    J2[i-1] = (1/2) * np.matmul ( X0.T , np.matmul(K2[i] , X0) )

    F1[i] = (np.linalg.inv(R11)* np.matmul(B1.T,(K1[i]))).flatten()
    F2[i] = (np.linalg.inv(R22)* np.matmul(B2.T,(K2[i]))).flatten()

print("K1 :- \n",    K1[-1], "\nK1 gain :-\n", np.linalg.inv(R11)*
    np.matmul(B1.T,(K1[-1])))
print("\nK2 :- \n", K2[-1], "\nK2 gain :-\n", np.linalg.inv(R22)* np
    .matmul(B2.T,(K2[-1])))

data= {"J1": np.concatenate((J1,np.array([J1[-1]]))), "J2":np.
    concatenate((J2,np.array([J2[-1]])))}
iteration = [f"iteraion {i}" for i in range (max_iterations)]
iteration.append("optimal = ")
df = pandas.DataFrame(data, index=iteration)
print("\n",tabulate(df, headers = 'keys', tablefmt = 'psql'))

def X_t (t, S1, K1, S2, K2, X0):
    X = sp.linalg.expm((A - (np.matmul(S1,K1)) - (np.matmul(S2,K2)))
    *t)
    return (np.matmul(X,X0))

end_time = 15 #sec
time = np.linspace(0,end_time,end_time*100)

```

```python
X = np.zeros((len(time),len(X0)))
U1 = np.zeros((len(time)))
U2 = np.zeros((len(time)))

for i in range(len(time)):
    X[i] = X_t(time[i],S11,K1[-1],S22,K2[-1],X0).T.squeeze()
    U1[i] = -1 * np.matmul( np.linalg.inv(R11)* np.matmul(B1.T,(K1
   [-1])), X[i])
    U2[i] = -1 * np.matmul( np.linalg.inv(R22)* np.matmul(B2.T,(K2
   [-1])), X[i])

plt.figure(1)
for i in range(1,len(X[0])+1):
    plt.subplot(2,2,i)
    plt.plot(time, np.zeros_like(time), 'k--')
    plt.plot(time,X[:,i-1],'r', label=f"X$_{i}$")
    plt.xlabel("Time (sec)")
    plt.legend(loc='best')

plt.suptitle("Approximate State Trajectories")
plt.tight_layout()

plt.figure(2)
plt.subplot(2,1,1)
plt.plot(time, np.zeros_like(time), 'k--')
plt.plot(time, U1, 'r', label="U$_1$")
plt.ylabel("Force")
plt.legend(loc='best')

plt.subplot(2,1,2)
plt.plot(time, np.zeros_like(time), 'k--')
plt.plot(time, U2, 'r', label="U$_2$")
plt.xlabel("Time (sec)")
plt.ylabel("Force")
plt.legend(loc='best')

plt.suptitle("Approximate Feedback Controls")
plt.tight_layout()
plt.show()
```