# Reinforcement Learning Policy Iterations for Nash Differential Games

Kanchu Kiran

Decemeber 23, 2023

## 1 Introduction

This project delves into the intricate world of Nash differential games, a subset of game theory where multiple players make decisions over time, impacting each other's outcomes. Unlike zero-sum games where one player's gain is another's loss, non-zero sum Nash differential games, as explored in this study, involve complex interdependencies where players' objectives may partially align or conflict, mirroring real-world scenarios like economic markets or strategic interactions in international relations.

Central to our analysis is the implementation of policy iteration algorithms, a method traditionally used in reinforcement learning and optimal control. Policy iteration is a recursive process where the policy (or strategy) is repeatedly improved upon until convergence to an optimal policy is achieved. This method is particularly well-suited for Nash differential games, where finding an equilibrium strategy requires iterative adjustments based on the actions of other players.

Our approach is further enriched by the application of Lyapunov iterations, following the methodology outlined by Li and Gajic (1995). Lyapunov iterations are employed for solving coupled algebraic Riccati equations, which are crucial in identifying stabilizing solutions and Nash equilibria in differential games. These iterations offer computational efficiency and robustness, especially in dealing with the high-dimensional state spaces typical in Nash games.

The model for our investigation is built upon a set of system dynamics, represented by matrices $A, B1, B2$ and cost function parameters $Q1, Q2, R11, R12, R21, R22$. These elements encapsulate the state transitions and control actions within the game, with each player aiming to optimize their respective cost functions. Our MATLAB implementation critically examines stabilizability and detectability conditions, ensuring that the iterative policy updates lead towards a feasible and stable Nash equilibrium.

In summary, this project stands at the intersection of game theory, control theory, and applied mathematics, aiming to contribute a nuanced understanding of Nash differential games through the lens of policy and Lyapunov iterations. The insights gleaned from this study are anticipated to have broader implications in areas where strategic decision-making and dynamic interactions are paramount.

## 2 Problem Formulation

### 2.1 System Dynamics

Consider a controlled linear dynamic system corresponding to the Nash differential game strategies is given by the following state equation:

$$\dot{x} = Ax + B_1u_1 + B_2u_2, \quad x(t_0) = x_0 \tag{1}$$

where $x \in \mathbb{R}^n$ is the state vector, $u_1 \in \mathbb{R}^{m_1}$ and $u_2 \in \mathbb{R}^{m_2}$ are control inputs $A$, $B_1$, and $B_2$ are constant matrices of appropriate dimensions.

## 2.2 Performance Criterion

In this Nash differential game, the performance criterion for each player is defined by individual and coupled cost functions. With each control agent a quadratic type function is associated defined as:

$$J_1(u_1, u_2, x_0) = \frac{1}{2}\int_{t_0}^{\infty} (x^T Q_1 x + u_1^T R_{11} u_1 + u_2^T R_{12} u_2)\, dt \tag{2}$$

$$J_2(u_1, u_2, x_0) = \frac{1}{2}\int_{t_0}^{\infty} (x^T Q_2 x + u_1^T R_{21} u_1 + u_2^T R_{22} u_2)\, dt \tag{3}$$

The weighting matrices are symmetric and defined as follows:

$$Q_i \geq 0, \quad R_{ii} > 0, \quad R_{ij} \geq 0, \quad i = 1, 2; \quad j = 1, 2; \quad i \neq j \tag{4}$$

where $Q_i$ are positive semidefinite, and $R_{ii}$ are positive definite matrices.
The optimal solution to the given problem leads to the so-called Nash optimal strategies $u_1^*$ and $u_2^*$ satisfying:

$$J_1(u_1^*, u_2^*) \leq J_1(u_1, u_2^*), \quad J_2(u_1^*, u_2^*) \leq J_2(u_1^*, u_2) \tag{5}$$

These equations describe the long-term costs incurred by each player, incorporating both the state variables and control actions. The matrices $Q_1, Q_2$ represent the cost of the states, while $R_{11}, R_{22}$ are the control costs, and $R_{12}, R_{21}$ quantify the interaction effects between the players' strategies. The goal of each player in the Nash game is to minimize their respective cost function, reflecting a balance between individual objectives and the impact of the other player's decisions.

## 2.3 Hamiltonian

The Hamiltonians $H_1$ and $H_2$ in the Nash game optimization problem are defined as:

$$\begin{aligned} H_1(x(t), p_1(t), u_1(t), u_2(t)) = \frac{1}{2}x(t)^T Q_1 x(t) + u_1(t)^T R_{11} u_1(t) + u_2(t)^T R_{12} u_2(t) \\ + p_1(t)^T \left(Ax(t) + B_1 u_1(t) + B_2 u_2(t)\right) \end{aligned} \tag{6}$$

$$\begin{aligned} H_2(x(t), p_2(t), u_1(t), u_2(t)) = \frac{1}{2}x(t)^T Q_2 x(t) + u_1(t)^T R_{21} u_1(t) + u_2(t)^T R_{22} u_2(t) \\ + p_2(t)^T \left(Ax(t) + B_1 u_1(t) + B_2 u_2(t)\right) \end{aligned} \tag{7}$$

These equations describe the Hamiltonian functions for each player in the Nash differential game. The Hamiltonian for each player is a composite function that includes the state $x(t)$, the co-state $p_i(t)$, and the control inputs $u_1(t)$ and $u_2(t)$. The state and control terms are weighted by the matrices $Q_1, Q_2, R_{11}, R_{12}, R_{21}$, and $R_{22}$, while the dynamics of the system are captured by the matrices $A, B_1$, and $B_2$. These Hamiltonians are central to determining the optimal control strategies for the players in the game.

## 2.4 Necessary Conditions

The necessary conditions for optimum in the Nash differential game are given by the following set of equations:

$$\frac{dx(t)}{dt} = Ax(t) + B_1 u_1(t) + B_2 u_2(t), \qquad x(t_0) = x_0, \qquad i = 1, 2 \tag{8}$$

$$\frac{dp_1(t)}{dt} = -Q_1 x(t) - A^T p_1(t) + \left(\frac{\partial H_1}{\partial u_2}\right)^T, \qquad p_1(t_f) = \frac{\partial \gamma_1(x(t_f))}{\partial x}, \tag{9}$$

$$\frac{dp_2(t)}{dt} = -Q_2 x(t) - A^T p_2(t) + \left(\frac{\partial H_2}{\partial u_1}\right)^T, \qquad p_2(t_f) = \frac{\partial \gamma_2(x(t_f))}{\partial x}, \tag{10}$$

where the Hamiltonians $H_1$ and $H_2$ are given by:

$$\frac{\partial H_1}{\partial u_1} = R_{11}u_1 + B_1^T p_1 = 0, \qquad u_1^{opt}(t) = -R_{11}^{-1}B_1^T p_1(t), \tag{11}$$

$$\frac{\partial H_2}{\partial u_2} = R_{22}u_2 + B_2^T p_2 = 0, \qquad u_2^{opt}(t) = -R_{22}^{-1}B_2^T p_2(t), \tag{12}$$

These conditions outline that the optimal control strategies in feedback form are linear functions of the state variables, characterized by the gain matrices $K_1(t)$ and $K_2(t)$. Furthermore, the conditions specify the dynamics of the co-state variables $p_1(t)$ and $p_2(t)$.
Additionally, the derivatives of the Hamiltonians with respect to the control inputs are related to the system matrices as follows:

$$\frac{\partial H_1}{\partial u_1} = R_{11}u_1 + B_1^T p_1, \quad \frac{\partial H_2}{\partial u_2} = R_{22}u_2 + B_2^T p_2 \tag{13}$$

The state and co-state equations when both players use feedback controls become:

$$\frac{dx(t)}{dt} = (A - S_1K_1(t) - S_2K_2(t))x(t), \tag{14}$$

$$\frac{dp_1(t)}{dt} = -(Q_1 + K_1(t)Z_1K_1(t))x(t) - (A - S_1K_1(t))^T p_1(t), \tag{15}$$

$$\frac{dp_2(t)}{dt} = -(Q_2 + K_2(t)Z_2K_2(t))x(t) - (A - S_2K_2(t))^T p_2(t), \tag{16}$$

where the matrices $S_1, S_2, Z_1$, and $Z_2$ are defined by:

$$S_1 = B_1R_{11}^{-1}B_1^T, \quad S_2 = B_2R_{22}^{-1}B_2^T, \quad Z_1 = B_1R_{11}^{-1}R_{12}^{-1}B_1^T, \quad Z_2 = B_2R_{22}^{-1}R_{21}^{-1}B_2^T \tag{17}$$

These equations illustrate the structure of the Nash equilibrium in terms of feedback controls and demonstrate the interdependence of the agents' strategies through the system matrices.

## 2.5 Nash Strategies and Coupled Algebraic Riccati Equations

In game theory, Nash strategies represent a set of strategies where each player's strategy is optimal given the other players' strategies. In the context of linear-quadratic differential games, Nash strategies can be obtained through the synthesis of feedback control laws that rely on solving coupled algebraic Riccati equations (CAREs). The closed-loop Nash strategy for each player is defined by:

$$u_i^* = -R_{ii}^{-1}B_i^T K_i x, \quad i = 1, 2 \tag{18}$$

where $K_i$ satisfies the coupled algebraic Riccati equations:

$$K_1A + A^TK_1 + Q_1 - K_1S_1K_1 - K_2S_2K_1 - K_1S_2K_2 + K_2Z_2K_2 = N_1(K_1, K_2) = 0 \tag{19}$$

$$K_2A + A^TK_2 + Q_2 - K_2S_2K_2 - K_2S_1K_1 - K_1S_1K_2 + K_1Z_1K_1 = N_2(K_1, K_2) = 0 \tag{20}$$

with,
$S_i = B_iR_{ii}^{-1}B_i^T$ and $Z_i = B_jR_{ij}^{-1}R_{ji}R_{jj}^{-1}B_j^T$, for $i, j = 1, 2$, $i \neq j$.

The existence of solutions to these CAREs and the subsequent derivation of Nash strategies are contingent upon the system's parameters satisfying certain detectability and stabilizability conditions. These conditions ensure that the control laws derived are not only mathematically sound but also practically implementable.

## 2.6 Lyapunov Iterations

Lyapunov iterations are instrumental in computing the solutions to the coupled algebraic Riccati equations, which are pivotal for determining the closed-loop Nash strategies in linear-quadratic differential games.

### 2.6.1 Initialization

The initialization of the Lyapunov iterations is based on the assumption that the system under consideration satisfies certain stabilizability-detectability conditions. These conditions ensure that there exists a unique positive definite solution to an auxiliary algebraic Riccati equation, which provides the initial stabilizing solutions.

**Hurwitz Condition (Stabilizable and Detectable)** Either the triple $(A, B_1, \sqrt{Q_1})$ or $(A, B_2, \sqrt{Q_2})$ is stabilizable-detectable. To guarantee the existence of a stabilizing solution, the system described by the matrices $A$, $B_i$, and the weighting matrix $Q_i$ must satisfy the Hurwitz condition, which necessitates that the system is both stabilizable and detectable. Mathematically, this is captured by the following set of auxiliary Riccati equations:

$$K_i^{(0)} A + A^T K_i^{(0)} + Q_i - K_i^{(0)} S_i K_i^{(0)} = 0, \quad i = 1, 2 \tag{21}$$

These equations serve as the initial conditions for the iterative process, where $K_i^{(0)}$ are the initial stabilizing solutions, and $S_i$ are the corresponding solutions to the Lyapunov equations:

$$S_i = B_i R_{ii}^{-1} B_i^T, \quad i = 1, 2 \tag{22}$$

The matrices $R_{ii}$ are derived from the cost functionals associated with each control agent in the Nash game, signifying the control effort penalties.

### 2.6.2 Iterative Procedure

The Lyapunov iterations for solving these equations are performed under the stabilizability-detectability assumption. The iterations are defined as:

$$(A - S_1 K_1^{(i)} - S_2 K_2^{(i)})^T K_1^{(i+1)} + K_1^{(i+1)} (A - S_1 K_1^{(i)} - S_2 K_2^{(i)}) = -Q_1^{(i)}, \tag{23}$$

$$(A - S_1 K_1^{(i)} - S_2 K_2^{(i)})^T K_2^{(i+1)} + K_2^{(i+1)} (A - S_1 K_1^{(i)} - S_2 K_2^{(i)}) = -Q_2^{(i)} \tag{24}$$

for $i, j = 1, 2$, $i \neq j$, where $Q_i^{(n)}$ are updated iteratively. The algorithm starts with initial conditions obtained from auxiliary algebraic Riccati equations and iterates towards the stabilizing solution. The iterative process is designed to converge to the unique positive definite stabilizing solution of the coupled algebraic Riccati equations, ensuring the existence and uniqueness of the solution.

## 2.7 Optimal Performance Criterion

Lyapunov equations:

$$(A - S_1 K_1 - S_2 K_2)^T V_1 + V_1 (A - S_1 K_1 - S_2 K_2) + Q_1 + K_1^T S_1 K_1 + K_2^T Z_2 K_2 = 0 \tag{25}$$

$$(A - S_1 K_1 - S_2 K_2)^T V_2 + V_2 (A - S_1 K_1 - S_2 K_2) + Q_2 + K_2^T S_2 K_2 + K_1^T Z_1 K_1 = 0 \tag{26}$$

Moreover, by setting $V_1 = K_1$ and $V_2 = K_2$, we see that $V_1$ and $V_2$ satisfy the coupled algebraic Riccati equations (31), so that the optimal performance criteria are given by:

$$J_1^{opt}(x(t_0)) = \frac{1}{2}x_0^T V_1 x_0 = \frac{1}{2}x_0^T K_1 x_0 \tag{27}$$

$$J_2^{opt}(x(t_0)) = \frac{1}{2}x_0^T V_2 x_0 = \frac{1}{2}x_0^T K_2 x_0 \tag{28}$$

These kinds of optimization problems appear in power systems, machine learning, economics, and other engineering and scientific areas.

### 2.7.1 Convergence

The convergence of the Lyapunov iterations is evaluated by examining the stability of the updated solutions and the associated cost functionals. Convergence is indicated by the successive solutions $K_i^{(k)}$ approaching a limit that satisfies the coupled algebraic Riccati equations.

**Condition for Convergence** The iterative solutions $K_i^{(k)}$ are said to converge if they satisfy the coupled algebraic Riccati equations as $k$ approaches infinity:

$$\begin{aligned}(A - S_1K_1^{(\infty)} - S_2K_2^{(\infty)})^T K_1^{(\infty)} + K_1^{(\infty)}(A - S_1K_1^{(\infty)} - S_2K_2^{(\infty)}) \\ + (Q_1 + K_1^{(\infty)}S_1K_1^{(\infty)} + K_2^{(\infty)}Z_2K_1^{(\infty)}) = 0,\end{aligned} \tag{29}$$

$$\begin{aligned}(A - S_1K_1^{(\infty)} - S_2K_2^{(\infty)})^T K_2^{(\infty)} + K_2^{(\infty)}(A - S_1K_1^{(\infty)} - S_2K_2^{(\infty)}) \\ + (Q_2 + K_1^{(\infty)}Z_1K_1^{(\infty)} + K_2^{(\infty)}S_2K_2^{(\infty)}) = 0.\end{aligned} \tag{30}$$

This indicates that the limit points $K_i^{(\infty)}$ of the sequences $K_i^{(k)}$ represent the sought solutions of the Nash strategies.

# 3 Implementation

The implementation focuses on applying the policy iteration algorithm to a non-zero sum Nash differential game. The game dynamics are defined by system matrices $A$, $B1$, and $B2$, along with the cost function parameters $Q1$, $Q2$, $R11$, $R12$, $R21$, and $R22$.

## 3.1 System Initialization

The MATLAB code initializes the system matrices and cost function parameters as follows:

```
A = [-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 = [0.4422; 3.0447; -5.52; 0];
B2 = [0.1761; -7.5922; 4.99; 0];

Q1 = diag([3.5, 2, 4, 5]);
Q2 = diag([1.5, 6, 3, 1]);
R11 = 1;
R12 = 0.25;
R21 = 0.6;
R22 = 2;

The initial state vector x0 is set to [0; 0; 0; 1].
```

### 3.2 Initialization and stabilizable-detectable check

The policy iteration begins with the computation of matrices $K1$ and $K2$ using the Algebraic Riccati Equation (ARE):

```
K1 = are(A, S1, Q1);
K2 = are(A - S1 * K1, S2, Q2 + K1 * S1 * K1);
```

The triple $(A, B_1, \sqrt{Q_1})$ or $(A, B_2, \sqrt{Q_2})$ is stabilizable-detectable. This implies weaker conditions than being controllable-observable.

At least one control agent must be capable of controlling and observing the unstable modes. Given that the context is a noncooperative game, the joint effect of the agents is assumed to address the unstable modes, albeit this is considered somewhat idealistic. To assert the first triple is stabilizable-detectable, the following conditions must be satisfied:

- The matrix $A - S_1 K_1^{(0)}$ must be stable, where $S_1$ and $K_1^{(0)}$ are matrices derived from the system parameters.
- The matrix $Q_2 + K_1^{(0)} Z_1 K_1^{(0)}$ should be positive semidefinite, indicating that the system does not respond negatively to state deviations.
- Finally, the matrix $A - S_1 K_1^{(0)} - S_2 K_2^{(0)}$ is also required to be stable, ensuring the overall system stability even after control adjustments.

These conditions are essential for the system to be not only stabilizable but also detectable, allowing for reliable control and observation of the system's dynamics.

```
Here is the code implementation to acheive above:
if all(real(eig(A - S1 * K1)) < 0)
    disp('Matrix A - S1 * K1 is stable.');
else
    disp('Matrix A - S1 * K1 is NOT stable.');
end
if all(eig(Q2 + K1' * Z1 * K1) >= 0)
    disp('The matrix Q2 + K1' * Z1 * K1 is positive semidefinite.');
else
    disp('The matrix Q2 + K1'* Z1 * K1 is NOT positive semidefinite.');
end
if all(real(eig(A - S1 * K1 - S2 * K2)) < 0)
    disp('Matrix A - S1 * K1 - S2 * K2 is stable.');
else
    disp('Matrix A - S1 * K1 - S2 * K2 is NOT stable.');
end
```

### 3.3 Policy Iteration Loop

The main loop of the policy iteration algorithm involves updating the feedback matrices $K1$ and $K2$ and computing the cost functionals $J1$ and $J2$. The iteration continues until the convergence is reached:

```
for i = 2:num_iterations
    % Update K matrices
    % Store K matrices and compute cost functionals J
    % ...
end
```

# 4 Simulation Results

## 4.1 State Trajectories

The state trajectories over time for each coordinate of the state vector $x$ are visualized through the following figures. These trajectories illustrate the evolution of the system's state as the policy iteration progresses, converging to the Nash equilibrium.

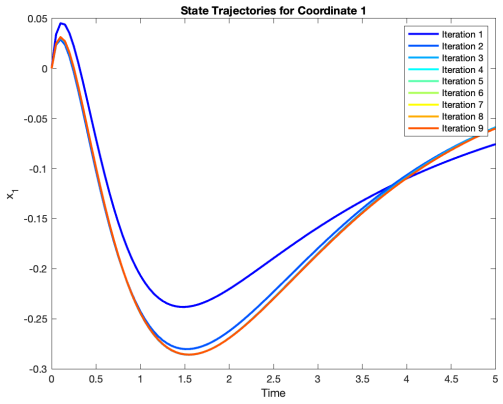


Figure 1: State trajectories for Coordinate 1.

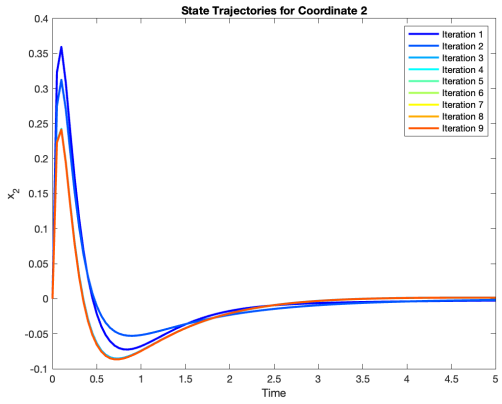


Figure 2: State trajectories for Coordinate 2.

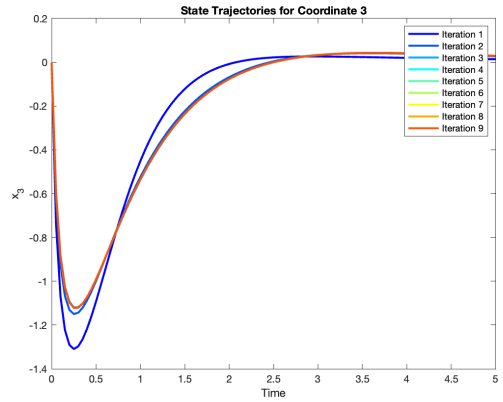


Figure 3: State trajectories for Coordinate 3.

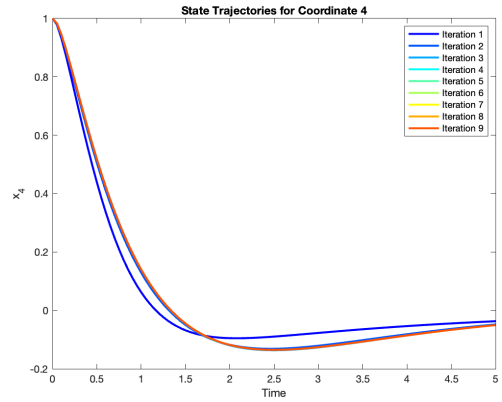


Figure 4: State trajectories for Coordinate 4.

Each pair of figures shows the state trajectories corresponding to specific coordinates over the course of seven iterations. The convergence of these trajectories towards the same path indicates the algorithm's efficacy in stabilizing the system and finding an optimal solution for both players in the Nash differential game.

## 4.2 Control Trajectories

The control trajectories for each player in the Nash differential game are depicted in Figures 5 and 6. These plots display the control inputs $u_1$ and $u_2$ over time for seven iterations of the policy iteration algorithm.

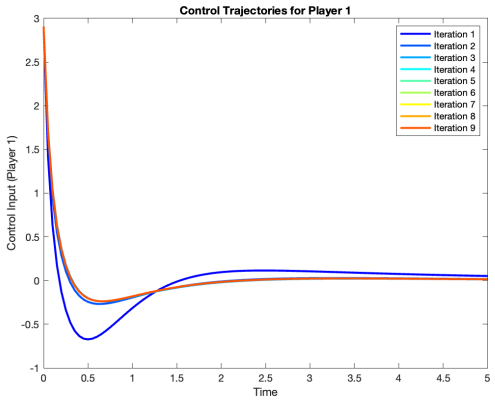


Figure 5: Control trajectories for Player 1.

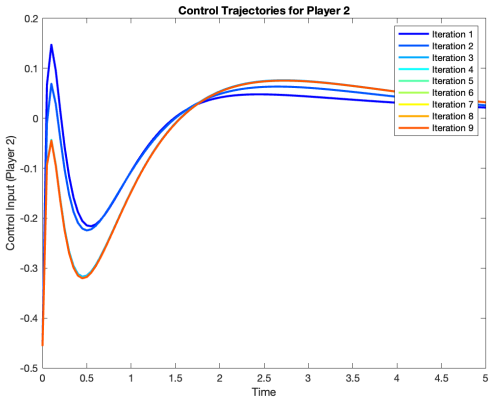


Figure 6: Control trajectories for Player 2.

In the case of Player 1, as depicted in Figure 5, the control input exhibits a rapid convergence towards the Nash equilibrium strategy after the initial iterations. The fluctuations in the control input diminish as the iterations progress, indicating a stabilization in the policy and approaching an optimal control strategy.

Similarly, the control input for Player 2, shown in Figure 6, also stabilizes over the iterations. The convergence is indicative of the players reaching a consensus on the best-response strategies, which is a characteristic of Nash equilibrium in a differential game setting.

The diminishing variance in control inputs between successive iterations for both players signifies that the equilibrium point is not only reached but also maintained, validating the effectiveness of the policy iteration algorithm. These trajectories corroborate the theoretical expectation that as the players adjust their strategies in response to each other's moves, they naturally gravitate towards a set of strategies that represent the Nash equilibrium, where no player has anything to gain by unilaterally changing their strategy.

## 4.3 Convergence and Performance Criterion

The convergence of the cost functionals $J_1$ and $J_2$ for both players is depicted in Figure 7. The graph tracks the cost associated with each player's strategy over successive iterations of the policy iteration algorithm.

The graph clearly shows that Player 1's cost functional $J_1$ experiences a significant drop between the first and the second iteration, followed by a plateau, which indicates a rapid convergence to a strategy that minimizes their cost. On the other hand, Player 2's cost functional $J_2$ starts at a lower value and remains relatively stable across the iterations, suggesting that Player 2's initial strategy was already near-optimal or that their optimal control strategy is less sensitive to the system dynamics and the actions of Player 1.

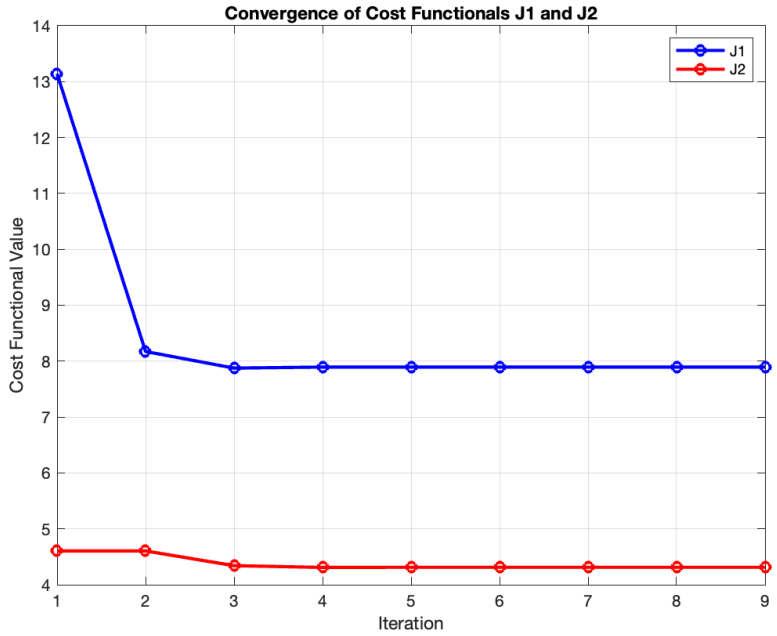


Figure 7: Convergence of Cost Functionals $J_1$ and $J_2$.

The convergence of both cost functionals from the second iteration onwards implies that the Nash equilibrium was reached quickly and maintained throughout the remaining iterations. This is indicative of the effectiveness of the policy iteration algorithm in finding stable strategies for both players where neither player can unilaterally improve their outcome, satisfying the Nash equilibrium condition in a non-zero sum game.

The stability of the cost functionals also suggests that the control strategies derived from the Nash equilibrium are robust, leading to consistent performance even as the game progresses. This robustness is crucial in dynamic environments where the strategies must adapt to evolving conditions while maintaining optimality.

Table 1: Performance Criterion Values at Each Iteration

| **Iteration** | **Player 1 (J1)** | **Player 2 (J2)** |
|---|---|---|
| 1 | 13.142 | 4.6047 |
| 2 | 8.1727 | 4.6047 |
| 3 | 7.8751 | 4.3413 |
| 4 | 7.894 | 4.3095 |
| 5 | 7.8938 | 4.3123 |
| 6 | 7.8946 | 4.3123 |
| 7 | 7.8947 | 4.3124 |
| 8 | 7.8947 | 4.3124 |
| 9 | 7.8947 | 4.3124 |

The table illustrates the progression of cost functionals for both players as the policy iteration algorithm iteratively refines their strategies. It is evident that after the initial iteration, there is a significant reduction in Player 1's cost ($J_1$), indicating the rapid convergence towards a strategy that minimizes their cost. Player 2's cost ($J_2$) remains relatively stable across iterations, suggesting that Player 2's initial strategy was near-optimal or less sensitive to changes in the game dynamics.

The stability of these cost functionals from the second iteration onwards confirms the convergence to a Nash equilibrium. This equilibrium represents a situation where neither player can unilaterally improve their outcome, signifying the effectiveness of the policy iteration algorithm in finding stable and optimal strategies in the context of a non-zero sum differential game.

# References


[1] Li, T. Y., & Gajic, Z. (1995). Lyapunov iterations for solving coupled algebraic Riccati equations of Nash differential games and the algebraic Riccati equation of zero-sum games. In *New Trends in Dynamic Games and Applications*, G. J. Olsder (Ed.), 333-351. Birkhauser.

[2] Vrabie, D., & Lewis, F. (2012). Integral Reinforcement Learning for Finding Online the Feedback Nash Equilibrium of Nonzero-Sum Differential Games. In *Advances in Reinforcement Learning*, A. Mellouk (Ed.), 313-330. IntechChina.

[3] Anderson, B. D. O., & Moore, J. B. (2007). *Optimal Control: Linear Quadratic Methods*. Dover Publications.

[4] Lewis, F. L., Vrabie, D., & Syrmos, V. L. (2012). *Optimal Control*. John Wiley & Sons.

[5] Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT Press.

[6] Kleinman, D. L. (1968). On an iterative technique for Riccati equation computations. *IEEE Transactions on Automatic Control*, 13(1), 114-115.


# Appendix

```
%% System matrices A, B1, B2, and the cost function parameters Q1, Q2, R11, R12, R21, R22
A = [-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 = [0.4422; 3.0447; -5.52; 0];
B2 = [0.1761; -7.5922; 4.99; 0];

Q1 = diag([3.5, 2, 4, 5]);
Q2 = diag([1.5, 6, 3, 1]);
R11 = 1;
R22 = 2;
R12 = 0.25;
R21 = 0.6;

S1 = B1 * inv(R11) * B1';
S2 = B2 * inv(R22) * B2';

Z1 = B1 * inv(R11) * R21 * inv(R11) * B1'
Z2 = B2 * inv(R22) * R12 * inv(R22) * B2'

x0 = [0; 0; 0; 1];

% Initialization for policy iteration
K1 = are(A, S1, Q1);
K2 = are(A - S1 * K1, S2, Q2 + K1 * S1 * K1);

% stabilizable-detectable.
if all(real(eig(A - S1 * K1)) < 0)
    disp('Matrix A - S1 * K1 is stable.');
else
    disp('Matrix A - S1 * K1 is NOT stable.');
end
```

```
if all(eig(Q2 + K1' * Z1 * K1) >= 0)
    disp('The matrix Q2 + K1''* Z1 * K1 is positive semidefinite.');
else
    disp('The matrix Q2 + K1''* Z1 * K1 is NOT positive semidefinite.');
end
if all(real(eig(A - S1 * K1 - S2 * K2)) < 0)
    disp('Matrix A - S1 * K1 - S2 * K2 is stable.');
else
    disp('Matrix A - S1 * K1 - S2 * K2 is NOT stable.');
end

% Initialize storage arrays
num_iterations = 9;
K1_array = cell(num_iterations, 1);
K2_array = cell(num_iterations, 1);
J1_array = zeros(num_iterations, 1);
J2_array = zeros(num_iterations, 1);

% Store initial values
K1_array{1} = K1;
K2_array{1} = K2;
J1_array(1) = 0.5 * trace(K1);
J2_array(1) = 0.5 * trace(K2);

% Policy Iteration Loop
for i = 2:num_iterations
    % Update K matrices
    K1 = lyap((A - S1 * K1_array{i-1} - S2 * K2_array{i-1})', Q1 + K1_array{i-1} * S1 *
    K1_array{i-1} + K2_array{i-1} * S2 * K2_array{i-1});
    K2 = lyap((A - S1 * K1_array{i-1} - S2 * K2_array{i-1})', Q2 + K2_array{i-1} * S2 *
    K2_array{i-1} + K1_array{i-1} * S1 * K1_array{i-1});

    % Store K matrices and compute cost functionals J
    K1_array{i} = K1;
    K2_array{i} = K2;
    J1_array(i) = 0.5 * trace(K1);
    J2_array(i) = 0.5 * trace(K2);
end

%% Simulations and Plots
time = 0:0.05:5;
x_traj_array = cell(num_iterations, 1);
for i = 1:num_iterations
    x_traj = zeros(length(x0), length(time));
    for j = 1:length(time)
        x_traj(:, j) = expm((A - S1 * K1_array{i} - S2 * K2_array{i}) * time(j)) * x0;
    end
    x_traj_array{i} = x_traj;
end

u1_traj_array = cell(num_iterations, 1);
u2_traj_array = cell(num_iterations, 1);
for k = 1:num_iterations
    u1_traj = -inv(R11) * B1' * K1_array{i} * x_traj_array{k, 1};
    u2_traj = -inv(R22) * B2' * K2_array{i} * x_traj_array{k, 1};
    u1_traj_array{k} = u1_traj;
    u2_traj_array{k} = u2_traj;
end


num_colors = num_iterations;
gradient_colormap = jet(num_colors);

% Plot State Trajectories
for i = 1:size(x0, 1)
    figure;
    legend_labels = cell(num_iterations, 1);
```

```
    for k = 1:num_iterations
        plot_color = gradient_colormap(k, :);
        plot(time, x_traj_array{k}(i, :), 'LineWidth', 2, 'Color', plot_color);
        hold on;
        legend_labels{k} = ['Iteration ' num2str(k)];
    end
    legend(legend_labels);
    title(['State Trajectories for Coordinate ' num2str(i)]);
    xlabel('Time');
    ylabel(['x_' num2str(i)]);
    saveas(gcf, sprintf('state_trajectory_coord%d.png', i));
end

% Plot Control Trajectories for Player 1
figure;
legend_labels1 = cell(num_iterations, 1);
for k = 1:num_iterations
    plot_color = gradient_colormap(k, :);
    plot(time, u1_traj_array{k}, 'LineWidth', 2, 'Color', plot_color);
    hold on;
    legend_labels1{k} = ['Iteration ' num2str(k)];
end
legend(legend_labels1);
title('Control Trajectories for Player 1');
xlabel('Time');
ylabel('Control Input (Player 1)');
saveas(gcf, 'control_trajectories_player1.png');

% Plot Control Trajectories for Player 2
figure;
legend_labels2 = cell(num_iterations, 1);
for k = 1:num_iterations
    plot_color = gradient_colormap(k, :);
    plot(time, u2_traj_array{k}, 'LineWidth', 2, 'Color', plot_color);
    hold on;
    legend_labels2{k} = ['Iteration ' num2str(k)];
end
legend(legend_labels2);
title('Control Trajectories for Player 2');
xlabel('Time');
ylabel('Control Input (Player 2)');
saveas(gcf, 'control_trajectories_player2.png');


% Plot J1 and J2
figure;
plot(1:num_iterations, J1_array, 'b-o', 'LineWidth', 2);
hold on;
plot(1:num_iterations, J2_array, 'r-o', 'LineWidth', 2);
legend('J1', 'J2');
title('Convergence of Cost Functionals J1 and J2');
xlabel('Iteration');
ylabel('Cost Functional Value');
grid on;
saveas(gcf, 'J1_J2_convergence.png');

% Create a table for convergence
Iterations = (1:num_iterations)';
T = table(Iterations, J1_array, J2_array, 'VariableNames', {'Iteration', 'J1', 'J2'});
disp(T);
```