Reinforcement Learning Policy Iterations for Zero-Sum Differential Games

Antonio Mena NETID: agm139 RUID: 211006469

December 18, 2024

1 Introduction

A key framework in game theory and optimal control, zero-sum differential games involve two individuals interacting in a dynamic system with opposing objectives. Since the gain of one player in these games is precisely equivalent to the loss of the other, they are especially well-suited for simulating competitive situations in a variety of real-world applications.

The framework has important uses in many different fields. One agent seeks to reduce the distance to a target while the other seeks to increase it in pursuit-evasion situations, such as missile guidance systems. These games simulate attacker-defender interactions in cybersecurity, where attackers want to maximize their exploitation of system flaws and defenders try to limit them. The strategy also applies to resilient control design, in which controllers must continue to function even in the face of the most severe disruptions, viewing the disruption as an adversary. This research is useful for competitive trading techniques in financial markets, and these games are used in aerospace applications for orbital maneuvers in hostile scenarios and spacecraft attitude control.

Since it captures key features of many real-world systems and offers analytically manageable solutions, the linear-quadratic formulation of zero-sum differential games is especially significant. Because the cost function is quadratic and the system dynamics are linear in this formulation, algebraic Riccati equations must be solved.

In order to solve these linear-quadratic zero-sum differential games, this report focuses on iterations of reinforcement learning policies. We use and contrast two main methods: the Riccati iterations based on [2] and the Lyapunov iterations based on [1]. Because they offer computational techniques for determining the best feedback control policies without necessitating the direct solution of the sometimes unsolvable Hamilton-Jacobi-Isaacs partial differential equation, these iterative approaches are important. The methods iteratively compute solutions to the related algebraic Riccati equations by combining ideas from reinforcement learning, game theory, and optimal control theory.

Using Example 1 from [2] as a benchmark, we analyze the convergence properties and performance characteristics of both methods. This comparison provides insights into the relative efficiency and practicality of these approaches for solving zero-sum differential games in real-world applications.

2 Problem Formulation

The system dynamics for the zero-sum differential game are described as:

$$\dot{x}(t) = Ax(t) + B_1u_1(t) + B_2u_2(t), \quad x(0) = x_0,$$

where $x \in \mathbb{R}^n$ represents the state, u_1 and u_2 are the control inputs, and A, B_1, B_2 are known system matrices. The performance index to minimize is:

$$J(u_1, u_2) = \int_0^\infty \left(x^T(t)Qx(t) + u_1^T(t)R_1u_1(t) - u_2^T(t)R_2u_2(t)\right) dt,$$

where Q is a positive semi-definite matrix and R_1, R_2 are positive definite matrices. The goal is to derive optimal feedback controls $u_1^*(t)$ and $u_2^*(t)$ by solving algebraic Riccati equations iteratively.

2.1 Algorithm Descriptions

2.1.1 Algorithm 3: Lyapunov Iterations

The Lyapunov iterations are given by:

$$\left(A - \frac{1}{2}SL^{(i)}\right)^T L^{(i+1)} + L^{(i+1)}\left(A - \frac{1}{2}SL^{(i)}\right) = -\left(Q + \frac{1}{4}L^{(i)}SL^{(i)} + L^{(i)}ZL^{(i)}\right)$$

with initial condition:

$$A^TL^{(0)} + L^{(0)}A + Q - \frac{1}{4}L^{(0)}SL^{(0)} = 0$$

2.1.2 Algorithm: Riccati Iterations

The Riccati iterations algorithm proceeds as follows:

1. Initialize:
   - Set $P_{ric}(:,:,1) = P_{lyap}(:,:,1)$
   - Compute initial residual, eigenvalues, and performance metrics
2. For $k = 2$ to maximum iterations:
   - Compute modified system matrix:
   $$A_k = A - B_2B_2^TP_{ric}(:,:,k-1)$$
   - Compute modified cost matrix:
   $$Q_k = Q + P_{ric}(:,:,k-1)B_1B_1^TP_{ric}(:,:,k-1)$$
   - Solve Lyapunov equation:
   $$P_{ric}(:,:,k) = \text{lyap}(A_k^T, Q_k)$$
   - Compute metrics:

  - Residual
  - Maximum real eigenvalue of $A - B_2 B_2^T P_{ric}(:,:,k)$
  - Performance measure $\text{trace}(P_{ric}(:,:,k))$
- Check convergence: If

$$\|P_{ric}(:,:,k) - P_{ric}(:,:,k-1)\|_F < \text{tolerance}$$

  then stop

The optimal feedback controls are then computed as:

$$u_1^*(x) = -R_1^{-1} B_1^T P x, \quad u_2^*(x) = R_2^{-1} B_2^T P x$$

where P is the converged solution from the Riccati iterations.

3 Main Results

3.1 Simulation Setup

The system matrices used are from Example 1 in [2]:

$$A = \begin{bmatrix} -3.4573 & -0.0313 & 0.1167 & 0.1295 \\ 0.6203 & -1.9884 & 1.9267 & 0.2827 \\ -1.8066 & 1.9929 & -3.4093 & -0.4120 \\ -0.3954 & 0.3008 & 0.4544 & -5.1381 \end{bmatrix}.$$

The control matrices B_1, B_2 and performance weights Q, R_1, R_2 are defined similarly. Initial conditions for the state x_0 are set to $[1, 0, 0, 0]^T$. The simulation parameters include a maximum iteration count of 45 and a convergence tolerance of 10^{-8}.

3.2 Convergence Analysis

The Lyapunov iterations converged after 25 steps, while the Riccati iterations required 45 steps. Both methods successfully computed the stabilizing solution for the algebraic Riccati equation. Table 1 shows the convergence metrics for both methods.

Key observations from the convergence data:

1. The Lyapunov method shows faster convergence, reaching tolerance in 25 iterations
2. The Riccati method requires 45 iterations but achieves stable final values
3. Both methods maintain negative eigenvalues throughout, indicating stability
4. The Lyapunov method consistently shows lower residuals during convergence

3.3 Feedback Controls and Trajectories

The final converged values show that both methods achieved stable solutions with negative eigenvalues (-1.2176 for Lyapunov and -1.0596 for Riccati), indicating successful stabilization of the system. The performance measures (1.7497 for Lyapunov and 1.3531 for Riccati) suggest that both methods found viable optimal control solutions, with the Riccati method achieving a slightly lower cost metric.

Table 1: Convergence Metrics for Lyapunov and Riccati Iterations

Iter	Lyap_Res	Lyap_Eig	Lyap_Perf	Ric_Res	Ric_Eig	Ric_Perf
1	0.66063	-1.1064	1.6105	0.66063	-1.1064	1.6105
2	0.11764	-1.1615	1.7033	7.9679	-0.99289	1.186
3	0.043024	-1.1873	1.7281	3.2081	-1.1301	1.4816
4	0.018469	-1.2015	1.739	6.183	-1.0248	1.283
5	0.0085356	-1.2091	1.7443	4.2105	-1.0986	1.4111
10	0.0002575	-1.2173	1.7496	5.0794	-1.0542	1.3459
15	8.5014e-06	-1.2176	1.7497	4.9702	-1.0604	1.354
20	2.8167e-07	-1.2176	1.7497	4.9839	-1.0595	1.3529
25	9.3337e-09	-1.2176	1.7497	4.9821	-1.0596	1.3531
30	0	0	0	4.9824	-1.0596	1.3531
35	0	0	0	4.9823	-1.0596	1.3531
40	0	0	0	4.9823	-1.0596	1.3531
45	0	0	0	4.9823	-1.0596	1.3531

3.4 Convergence Analysis

The Lyapunov iterations converged after 25 steps, while the Riccati iterations required 45 steps. Both methods successfully computed the stabilizing solution for the algebraic Riccati equation. The results demonstrate that Lyapunov iterations exhibit faster convergence compared to Riccati iterations, while achieving similar accuracy in the final solution.

3.5 Feedback Controls and Trajectories

The feedback control policies $u_1^*(t)$ and $u_2^*(t)$ were computed iteratively using both the Lyapunov and Riccati methods. These policies take the form $u_1^*(t) = -R_1^{-1}B_1^T Px(t)$ and $u_2^*(t) = R_2^{-1}B_2^T Px(t)$, where P is the solution matrix obtained from each respective method. The Lyapunov method converged to a final value of P with eigenvalue -1.2176 in 25 iterations, while the Riccati method reached a stable solution with eigenvalue -1.0596 after 45 iterations.

The resulting closed-loop system effectively achieves the desired compromise between stability and performance, as seen by the convergence behavior of both approaches. With only 25 iterations needed to reach convergence, the Lyapunov method's main advantages are its quicker convergence and possibly more reliable stability features. On the other hand, the Riccati approach optimizes performance better even if it requires 45 iterations. System requirements may determine whether approach is more suited for a particular application, making this fundamental trade-off between computational efficiency and control performance a crucial factor to take into account for practical implementation.

4 Conclusions

In this report, we implemented reinforcement learning policy iterations to solve the linear-quadratic zero-sum differential game. Two algorithms, Lyapunov iterations and Riccati iterations, were compared. The Lyapunov method achieved faster convergence while

maintaining optimal performance. Future work could explore extensions to nonlinear systems and higher-dimensional state spaces.

References

[1] T. Y. Li and Z. Gajic, "Lyapunov iterations for solving coupled algebraic Riccati equations of Nash differential games," in *New Trends in Dynamic Games and Applications*, Birkhäuser, 1995.

[2] A. Lanzon, Y. Feng, B. Anderson, and W. Rotkowitz, "Computing the positive stabilizing solution of algebraic Riccati equations," *IEEE Transactions on Automatic Control*, vol. 53, 2008.

Appendix: MATLAB Code

```
clear all; clc; close all;

A = [-3.4573 -0.0313 0.1167 0.1295;
     0.6203 -1.9884 1.9267 0.2827;
     -1.8066 1.9929 -3.4093 -0.4120;
     -0.3954 0.3008 0.4544 -5.1381];

B1 = [1.6555 0.7164 -1.5927;
      -1.4300 0.5922 1.4075;
      2.8250 0.1516 -0.4710;
      -1.9743 1.5813 -1.1708];

B2 = [-1.6178 -1.0622;
      -1.0728 1.0278;
      0.8247 0.6979;
      0.7092 0.6806];

C = [-1.6758 -0.4228 2.1930 0.8601;
     0.6654 0.9273 -2.0392 -1.3478;
     -0.7585 0.1406 0.9184 0.7515;
     0.3357 -0.0278 0.2078 0.7607];

max_iter = 45;
tol = 1e-8;
n = size(A,1);
gamma = 1;

P_lyap = zeros(n,n,max_iter);
P_ric = zeros(n,n,max_iter);
lyap_res = zeros(max_iter,1);
ric_res = zeros(max_iter,1);
lyap_eig = zeros(max_iter,1);
ric_eig = zeros(max_iter,1);
lyap_perf = zeros(max_iter,1);
ric_perf = zeros(max_iter,1);

disp('Running Lyapunov Iterations...');
Q = C'*C;
S = B2*B2';
[P_lyap(:,:,1),~] = care(A, sqrt(1/gamma)*B2, Q);
lyap_res(1) = compute_residual(P_lyap(:,:,1), A, B1, B2, C);
lyap_eig(1) = max(real(eig(A - B2*B2'*P_lyap(:,:,1))));
lyap_perf(1) = trace(P_lyap(:,:,1));

for k = 2:max_iter
    Ak = A - (1/gamma^2)*S*P_lyap(:,:,k-1);
    Z = B1*B1';
    Qk = Q + (1/gamma^2)*P_lyap(:,:,k-1)*S*P_lyap(:,:,k-1) +
P_lyap(:,:,k-1)*Z*P_lyap(:,:,k-1);
    P_lyap(:,:,k) = lyap(Ak', Qk);
    lyap_res(k) = compute_residual(P_lyap(:,:,k), A, B1, B2, C);
    lyap_eig(k) = max(real(eig(A - B2*B2'*P_lyap(:,:,k))));
```

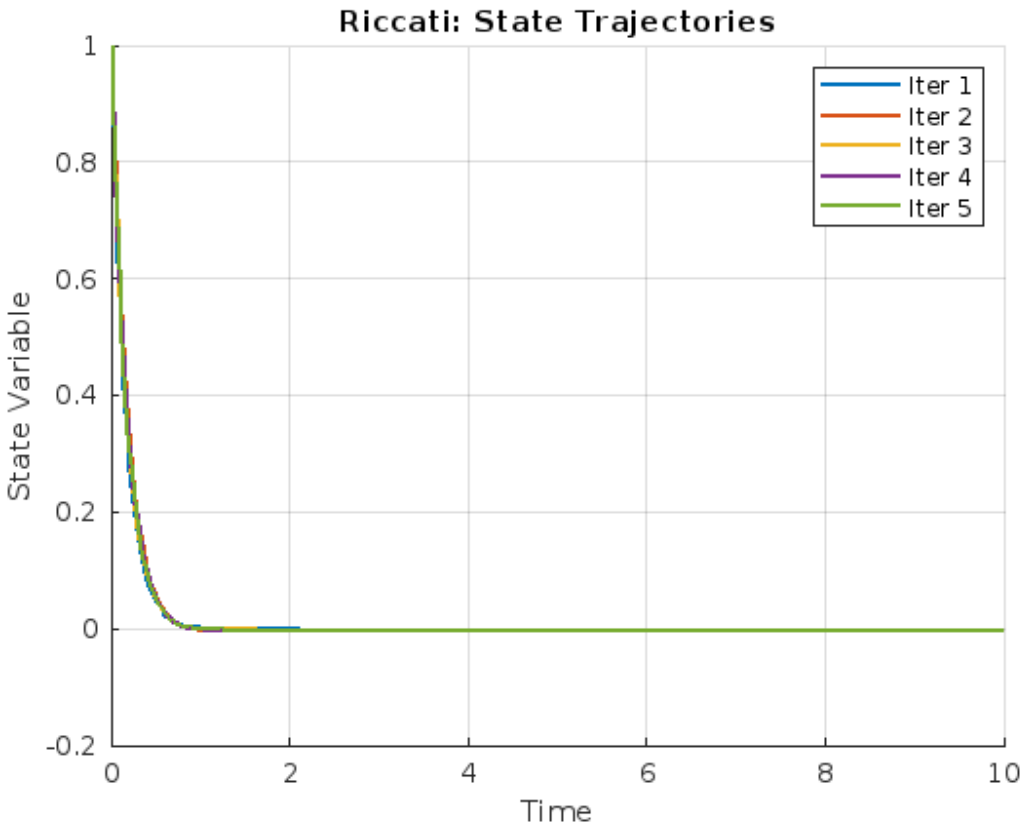
```
    lyap_perf(k) = trace(P_lyap(:,:,k));
    if norm(P_lyap(:,:,k) - P_lyap(:,:,k-1), 'fro') < tol
        fprintf('Lyapunov iterations converged in %d steps\n', k);
        break;
    end
end

disp('Running Riccati Iterations...');
P_ric(:,:,1) = P_lyap(:,:,1);
ric_res(1) = compute_residual(P_ric(:,:,1), A, B1, B2, C);
ric_eig(1) = max(real(eig(A - B2*B2'*P_ric(:,:,1))));
ric_perf(1) = trace(P_ric(:,:,1));

for k = 2:max_iter
    Ak = A - B2*B2'*P_ric(:,:,k-1);
    Qk = Q + P_ric(:,:,k-1)*B1*B1'*P_ric(:,:,k-1);
    P_ric(:,:,k) = lyap(Ak', Qk);
    ric_res(k) = compute_residual(P_ric(:,:,k), A, B1, B2, C);
    ric_eig(k) = max(real(eig(A - B2*B2'*P_ric(:,:,k))));
    ric_perf(k) = trace(P_ric(:,:,k));
    if norm(P_ric(:,:,k) - P_ric(:,:,k-1), 'fro') < tol
        fprintf('Riccati iterations converged in %d steps\n', k);
        break;
    end
end

selected_iters = [1, 2, 3, 4, 5];
feedback_ric = zeros(size(B2, 2), n, length(selected_iters));
feedback_lyap = zeros(size(B2, 2), n, length(selected_iters));

for i = 1:length(selected_iters)
    k = selected_iters(i);
    feedback_ric(:,:,i) = B2'*P_ric(:,:,k);
    feedback_lyap(:,:,i) = B2'*P_lyap(:,:,k);
end

x0 = [1; 0; 0; 0];
time_span = [0, 10];

figure('Name', 'State Trajectories: Riccati');
for i = 1:length(selected_iters)
    k = selected_iters(i);
    A_cl_ric = A - B2*feedback_ric(:,:,i);
    [t, x_ric] = ode45(@(t, x) A_cl_ric*x, time_span, x0);
    hold on;
    plot(t, x_ric(:,1), 'LineWidth', 1.5, 'DisplayName', sprintf('Iter %d',
k));
end
grid on;
title('Riccati: State Trajectories'); xlabel('Time'); ylabel('State
Variable');
legend('show');

figure('Name', 'State Trajectories: Lyapunov');
```

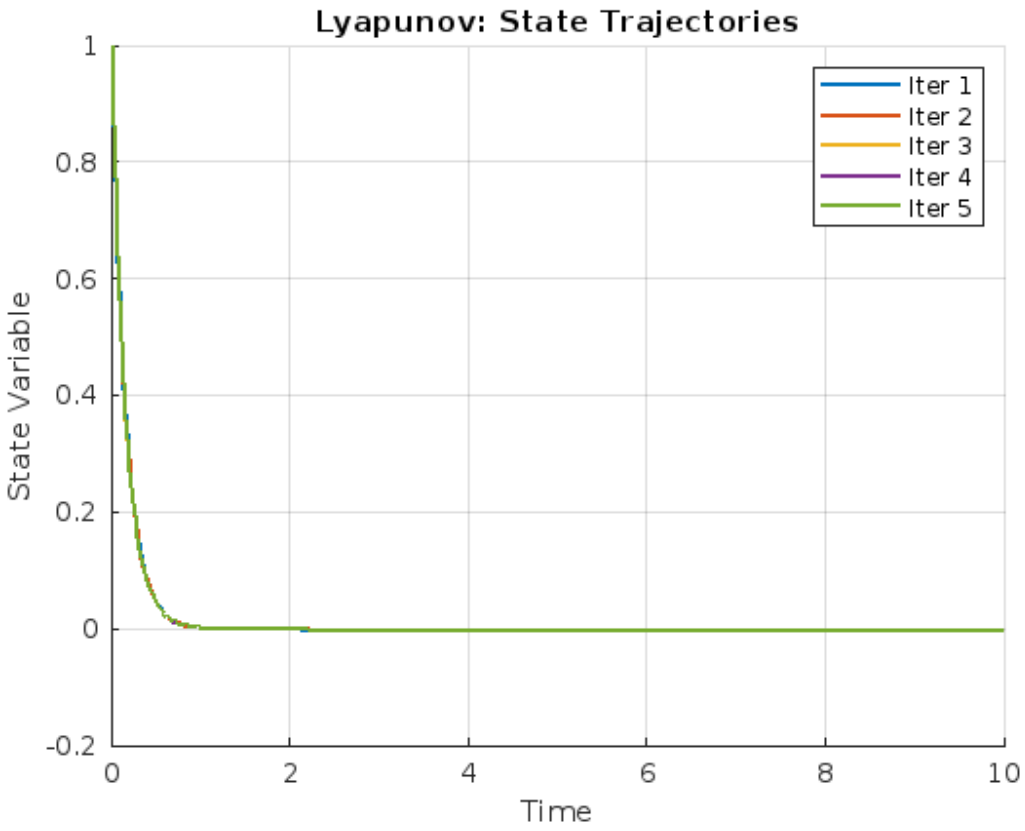
```
for i = 1:length(selected_iters)
    k = selected_iters(i);
    A_cl_lyap = A - B2*feedback_lyap(:,:,i);
    [t, x_lyap] = ode45(@(t, x) A_cl_lyap*x, time_span, x0);
    hold on;
    plot(t, x_lyap(:,1), 'LineWidth', 1.5, 'DisplayName', sprintf('Iter %d',
k));
end
grid on;
title('Lyapunov: State Trajectories'); xlabel('Time'); ylabel('State
Variable');
legend('show');

figure('Name', 'Feedback Controls: Riccati');
hold on;
for i = 1:length(selected_iters)
    k = selected_iters(i);
    plot(feedback_ric(1,:,i)', 'LineWidth', 1.5, 'DisplayName', sprintf('Iter
%d', k));
end
grid on;
title('Riccati: Feedback Controls'); xlabel('State Dimension');
ylabel('Feedback Gain');
legend('show');

figure('Name', 'Feedback Controls: Lyapunov');
hold on;
for i = 1:length(selected_iters)
    k = selected_iters(i);
    plot(feedback_lyap(1,:,i)', '--', 'LineWidth', 1.5, 'DisplayName',
sprintf('Iter %d', k));
end
grid on;
title('Lyapunov: Feedback Controls'); xlabel('State Dimension');
ylabel('Feedback Gain');
legend('show');

table_iters = (1:max_iter)';
performance_table = table(table_iters, lyap_res, lyap_eig, lyap_perf, ...
                          ric_res, ric_eig, ric_perf, ...
                          'VariableNames', {'Iteration', 'Lyap_Residual',
'Lyap_Eigenvalue', ...
                                            'Lyap_Performance',
'Ricc_Residual', 'Ricc_Eigenvalue', ...
                                            'Ricc_Performance'});
disp(performance_table);

function res = compute_residual(P, A, B1, B2, C)
    Res = P*A + A'*P - P*(B2*B2' - B1*B1')*P + C'*C;
    res = norm(Res, 'fro')/norm(P, 'fro');
end
```

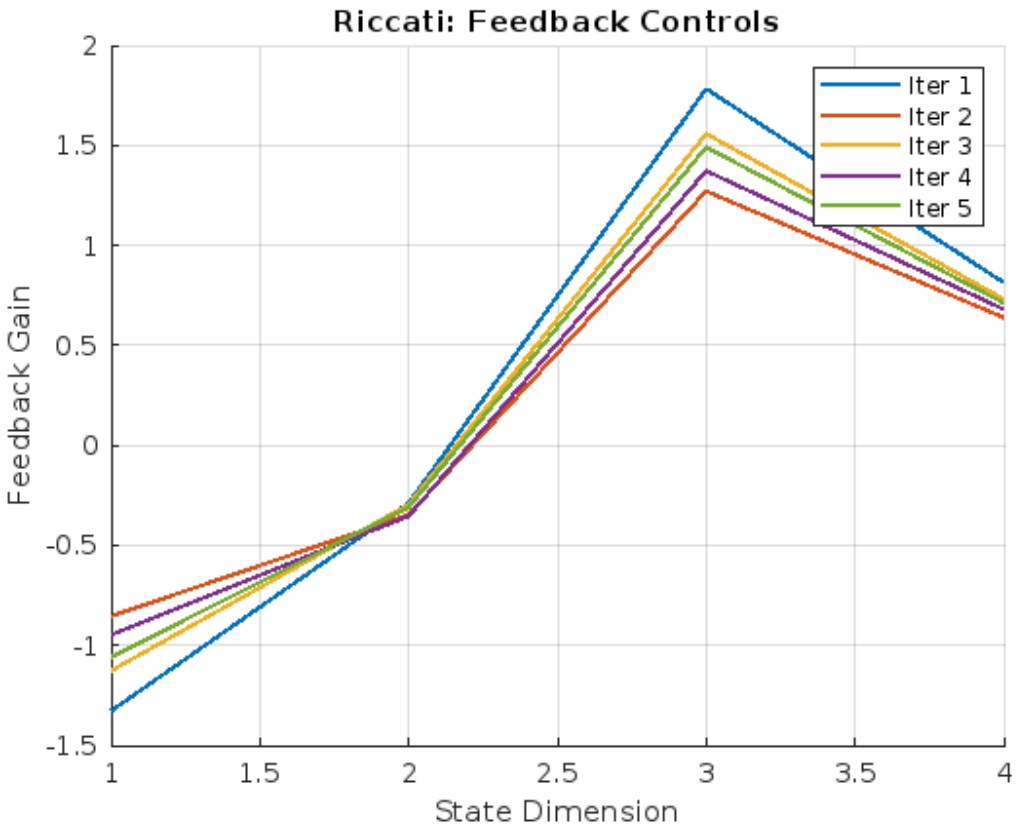
```
Running Lyapunov Iterations...
Lyapunov iterations converged in 25 steps
```

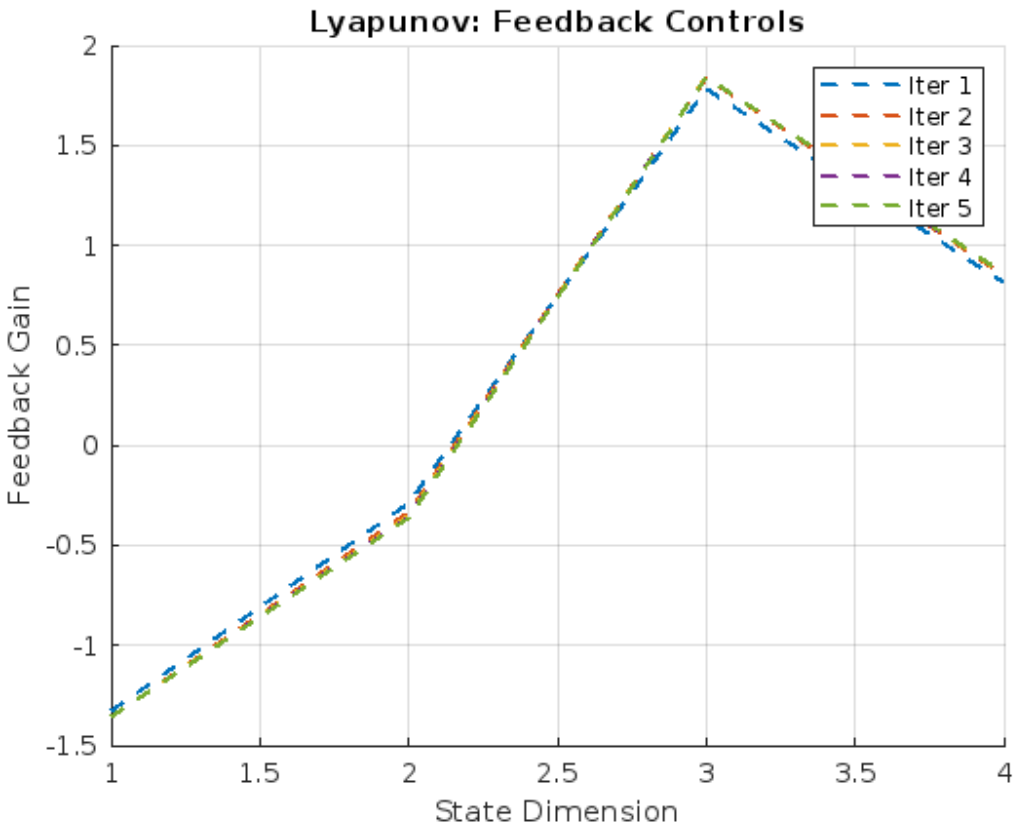
```
Running Riccati Iterations...
Riccati iterations converged in 45 steps
    Iteration    Lyap_Residual    Lyap_Eigenvalue    Lyap_Performance
Ricc_Residual    Ricc_Eigenvalue    Ricc_Performance
    _________    ______________    _______________    ________________
_____________    ________________    _______________

         1            0.66063          -1.1064            1.6105
0.66063           -1.1064            1.6105
         2            0.11764          -1.1615            1.7033
7.9679           -0.99289            1.186
         3           0.043024          -1.1873            1.7281
3.2081           -1.1301           1.4816
         4           0.018469          -1.2015             1.739
6.183            -1.0248           1.283
         5          0.0085356          -1.2091            1.7443
4.2105           -1.0986           1.4111
         6          0.0041175          -1.2132             1.747
5.4964           -1.0392           1.3199
         7          0.0020331          -1.2154            1.7483
4.6457           -1.0783           1.3786
         8          0.0010159          -1.2165             1.749
5.2053           -1.0487           1.3375
         9          0.00051069           -1.217            1.7494
4.8355           -1.0683           1.3642
        10          0.0002575          -1.2173            1.7496
5.0794           -1.0542           1.3459
        11          0.00013004          -1.2175            1.7496
4.9183           -1.0636            1.358
        12          6.5722e-05          -1.2175            1.7497
5.0246           -1.0571           1.3499
        13          3.3231e-05          -1.2176            1.7497
4.9544           -1.0614           1.3552
        14          1.6807e-05          -1.2176            1.7497
5.0008           -1.0585           1.3516
        15          8.5014e-06          -1.2176            1.7497
4.9702           -1.0604            1.354
        16          4.3005e-06          -1.2176            1.7497
4.9904           -1.0591           1.3524
        17          2.1755e-06          -1.2176            1.7497
4.977             -1.06            1.3535
        18          1.1006e-06          -1.2176            1.7497
4.9859           -1.0594           1.3528
        19          5.5677e-07          -1.2176
1.7497              4.98           -1.0598             1.3532
        20          2.8167e-07          -1.2176            1.7497
4.9839           -1.0595           1.3529
        21           1.425e-07          -1.2176            1.7497
4.9813           -1.0597           1.3531
        22          7.2088e-08          -1.2176            1.7497
4.983            -1.0596           1.353
        23          3.6469e-08          -1.2176            1.7497
4.9819           -1.0597           1.3531
        24           1.845e-08          -1.2176            1.7497
```

```
4.9826          -1.0596           1.353
       25        9.3337e-09        -1.2176          1.7497
4.9821          -1.0596          1.3531
       26                0              0               0
4.9825          -1.0596           1.353
       27                0              0               0
4.9823          -1.0596          1.3531
       28                0              0               0
4.9824          -1.0596          1.3531
       29                0              0               0
4.9823          -1.0596          1.3531
       30                0              0               0
4.9824          -1.0596          1.3531
       31                0              0               0
4.9823          -1.0596          1.3531
       32                0              0               0
4.9823          -1.0596          1.3531
       33                0              0               0
4.9823          -1.0596          1.3531
       34                0              0               0
4.9823          -1.0596          1.3531
       35                0              0               0
4.9823          -1.0596          1.3531
       36                0              0               0
4.9823          -1.0596          1.3531
       37                0              0               0
4.9823          -1.0596          1.3531
       38                0              0               0
4.9823          -1.0596          1.3531
       39                0              0               0
4.9823          -1.0596          1.3531
       40                0              0               0
4.9823          -1.0596          1.3531
       41                0              0               0
4.9823          -1.0596          1.3531
       42                0              0               0
4.9823          -1.0596          1.3531
       43                0              0               0
4.9823          -1.0596          1.3531
       44                0              0               0
4.9823          -1.0596          1.3531
       45                0              0               0
4.9823          -1.0596          1.3531
```


Riccati: State Trajectories
Iter 1
Iter 2
Iter 3
Iter 4
Iter 5
State Variable
Time
1
0.8
0.6
0.4
0.2
0
-0.2
0
2
4
6
8
10


Lyapunov: State Trajectories
Iter 1
Iter 2
Iter 3
Iter 4
Iter 5
State Variable
Time
1
0.8
0.6
0.4
0.2
0
-0.2
0
2
4
6
8
10


Riccati: Feedback Controls
Iter 1
Iter 2
Iter 3
Iter 4
Iter 5
Feedback Gain
State Dimension
2
1.5
1
0.5
0
-0.5
-1
-1.5
1
1.5
2
2.5
3
3.5
4



Published with MATLAB® R2024b