/MATLAB Drive/hw4.mlx

```
% Define system matrices
A = [-1, 1; 0, -2];
B = [1; 0.1];
C = [1, 0.1];
D = 0;

% (a) Controllability and Observability Matrices
Co = ctrb(A, B); % Controllability matrix
Ob = obsv(A, C); % Observability matrix

% Check rank
rank_Co = rank(Co); % Rank of controllability matrix
rank_Ob = rank(Ob); % Rank of observability matrix

disp('Controllability Matrix:');
disp(Co);
disp(['Rank: ', num2str(rank_Co)]);
disp('Observability Matrix:');
```

```
Controllability Matrix:
    1.0000   -0.9000
    0.1000   -0.2000
Rank: 2
Observability Matrix:
    1.0000    0.1000
   -1.0000    0.8000
Rank: 2
Balanced Realization System:
  ss with properties:

              A: [2x2 double]
              B: [2x1 double]
              C: [-1.0102 -0.1028]
              D: 0
              E: []
        Offsets: []
         Scaled: 0
      StateName: {2x1 cell}
      StatePath: {2x1 cell}
```

/MATLAB Drive/hw4.mlx

```
    disp(Ob);
    disp(['Rank: ', num2str(rank_Ob)]);

    % (c) Balanced Realization
    sys = ss(A, B, C, D);
    [sys_bal, G] = balreal(sys);
    disp('Balanced Realization System:');
    disp(sys_bal);
    % Extract and display the balanced realization matrices
    A_bal = sys_bal.A;
    B_bal = sys_bal.B;
    C_bal = sys_bal.C;
    D_bal = sys_bal.D;

    disp('Balanced A Matrix:');
    disp(A_bal);
    disp('Balanced B Matrix:');
    disp(B_bal);
```

```
        InputGroup: [1x1 struct]
        OutputName: {''}
        OutputUnit: {''}
        OutputGroup: [1x1 struct]
            Notes: [0x1 string]
          UserData: []
              Name: ''
                Ts: 0
          TimeUnit: 'seconds'
      SamplingGrid: [1x1 struct]
Balanced A Matrix:
   -0.9626   -0.1969
    0.1969   -2.0374
Balanced B Matrix:
   -1.0102
    0.1028
Balanced C Matrix:
   -1.0102   -0.1028
Balanced D Matrix:
    0
```

/MATLAB Drive/hw4.mlx

```
    disp('Balanced C Matrix:');
    disp(C_bal);
    disp('Balanced D Matrix:');
    disp(D_bal);
    disp('Gramian for controllability and observability 相等:');
    % after balance realization,controllability gramian and observability gramian 相等
    Wc_bal = gram(sys_bal, 'c');
    Wo_bal = gram(sys_bal, 'o');

    disp('Balanced Controllability Gramian Wc:');
    disp(Wc_bal);

    disp('Balanced Observability Gramian Wo:');
    disp(Wo_bal);
    disp('由 A 的 diagonal elements 可以看到:');
    disp('mode 1 eigenvalue = -0.9626 小於 mode 2 eigenvalue = -2.0374.');
    disp('因此他的 exponential decay 較慢,');
    disp('所以最終 overall system dynamics 只會留下他, 可以做 order reduction.');
```

```
Gramian for controllability and observability 相等:
Balanced Controllability Gramian Wc:
    0.5301   -0.0000
   -0.0000    0.0026
Balanced Observability Gramian Wo:
    0.5301    0.0000
    0.0000    0.0026
由 A 的 diagonal elements 可以看到:
mode 1 eigenvalue = -0.9626 小於 mode 2 eigenvalue = -2.0374.
因此他的 exponential decay 較慢,
所以最終 overall system dynamics 只會留下他, 可以做 order reduction.
而 B 和 C matrices 則留下對應的 first mode element.
Reduced-Order System:
  ss with properties:

              A: -0.9626
              B: -1.0102
              C: -1.0102
```

```
    disp('而 B 和 C matrices 則留下對應的 first mode element.');

    % (d) Order Reduction
    sys_red = modred(sys_bal, [2], 'truncate'); % Reduce to order 1
    disp('Reduced-Order System:');
    disp(sys_red);

    % (e) Bode Plots
    disp('由Bode plot可以看到做order reduction後的system dynamic');
    disp('和原系統幾乎一模一樣');
    disp('因此這個reduction成功的降階,並保留了原系統的大部分特性');
    figure;
    bode(sys, 'b', sys_red, 'r--');
    legend('Original System', 'Reduced System');
    title('Bode Plot Comparison');


% Define system matrices
```

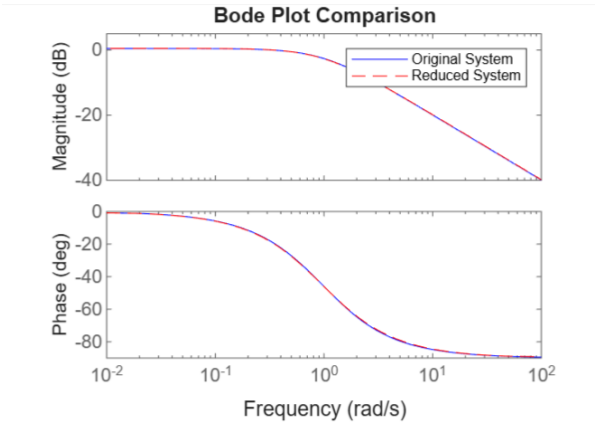
```
Reduced-Order System:
  ss with properties:

              A: -0.9626
              B: -1.0102
              C: -1.0102
              D: 0
              E: []
        Offsets: []
         Scaled: 0
      StateName: {''}
      StatePath: {''}
      StateUnit: {''}
  InternalDelay: [0x1 double]
     InputDelay: 0
    OutputDelay: 0
      InputName: {''}
      InputUnit: {''}
     InputGroup: [1x1 struct]
     OutputName: {''}
     OutputUnit: {''}
    OutputGroup: [1x1 struct]
```

由Bode plot可以看到做order reduction後的system dynamic

和原系統幾乎一模一樣

因此這個reduction成功的降階,並保留了原系統的大部分特性



/MATLAB Drive/hw4.mlx

```
% Define system matrices
A = [0, 1; 2, 3];
B = [0; 1];
C = [1, 1];
D = 0;

% (a) State Feedback Gain (Pole Placement)
desired_poles_state_feedback = [-10, -15];
K = place(A, B, desired_poles_state_feedback); % Compute state feedback gain
disp('State Feedback Gain K:');
disp(K);

% (b) Observer Gain (Pole Placement)
desired_poles_observer = [-20, -20];
L = acker(A', C', desired_poles_observer)'; % Compute observer gain
disp('Observer Gain L:');
disp(L);

% (c) Closed-loop and Observer Dynamics
A_cl = A - B*K; % Closed-loop dynamics
A_obs = A - L*C; % Observer dynamics
disp('由題目給的matrix A可以知原本系統open loop有pole在RHP不穩定');
```

```
State Feedback Gain K:
   152.0000   28.0000
Observer Gain L:
  -179.5000
   222.5000
由題目給的matrix A可以知原本系統open loop有pole在RHP不穩定
```

/MATLAB Drive/hw4.mlx

```
disp('close loop 後')
disp('Closed-loop A-BK:');
disp(A_cl);
disp('Observer A-LC:');
disp(A_obs);
disp('由step response可以看到系統close loop後穩定下來了,no blow up')
% (d) Simulate Open-Loop and Closed-Loop Step Responses
sys_open = ss(A, B, C, D); % Open-loop system
sys_closed = ss(A_cl, B, C, D); % Closed-loop system

t = 0:0.01:5; % Time vector
[y_open, t] = step(sys_open, t); % Open-loop step response
[y_closed, t] = step(sys_closed, t); % Closed-loop step response

% Plot Step Responses
figure;
plot(t, y_open, 'r', 'LineWidth', 1.5); hold on;
plot(t, y_closed, 'b--', 'LineWidth', 1.5);
xlabel('Time (s)');
ylabel('Response');
title('Step Response: Open-Loop vs Closed-Loop');
legend('Open-Loop', 'Closed-Loop');
grid on;
```

```
close loop 後
Closed-loop A-BK:
         0    1.0000
 -150.0000  -25.0000
Observer A-LC:
  179.5000  180.5000
 -220.5000 -219.5000
由step response可以看到系統close loop後穩定下來了,no blow up
```