```
clear
close all
clc
%{ The following program will determine the stability of the
% Linearized System with Eigenvalues and the Lyapunov equation.
% - The linearized Jacobian Matrices are converted to Jordan Form,
% Transfer functions, and Discrete State Space
% - The controllability/observability are checked for this system.
%}

% Model parameters
J = 0.0475; %kg m^2
m = 4; %kg
r = 0.25; %m
g = 9.81; % m/s^2
c = 0.05; %Ns/m

% Linearized Jacobian Matrices evaluated at the equilibrium point
A = [0 0 0 1 0 0;
    0 0 0 0 1 0;
    0 0 0 0 0 1;
    0 0 -g -c/m 0 0;
    0 0 0 0 -c/m 0;
    0 0 0 0 0 0];

B = [0 0;
    0 0;
    0 0;
    1/m 0;
    0 1/m;
    r/J 0];
C = [1 0 0 0 0 0;
    0 1 0 0 0 0];
D = [0 0;
    0 0];
% Convert Jacobian Matrices to State-Space Form
stateSpace = ss(A,B,C,D);
% Visualize the State-Space Form
disp('Linearized State-Space Form:')
display(stateSpace)
```

```
Linearized State-Space Form:

stateSpace =

  A =
            x1       x2       x3       x4       x5       x6
   x1        0        0        0        1        0        0
   x2        0        0        0        0        1        0
   x3        0        0        0        0        0        1
   x4        0        0    -9.81  -0.0125        0        0
   x5        0        0        0        0  -0.0125        0
```

```
  x6         0         0         0         0         0         0

 B =
         u1     u2
  x1      0      0
  x2      0      0
  x3      0      0
  x4   0.25      0
  x5      0   0.25
  x6  5.263      0

 C =
      x1  x2  x3  x4  x5  x6
  y1   1   0   0   0   0   0
  y2   0   1   0   0   0   0

 D =
      u1  u2
  y1   0   0
  y2   0   0

Continuous-time state-space model.
```

Stability Analysis Eigenvalue Analysis (A-lambda*I) = 0

```
EigValues = eig(A);
disp(' ')
disp('Eigenvalues of A')
disp(EigValues)

%Lyapunvo Analysis
% A^T*P + P*A= Q
Q = eye(6);
disp('Q Matrix')
disp(Q)
try
    P = lyap(A',Q);
    disp('P Matrix')
    disp(P)

    disp('Display P')
    disp(P)
    disp('Check if P Transpose = P')
    disp(P')
    disp("these matrices are the same")

    disp(' ')
    %check positive definite matrices
    x = [1; -1; 1];
    Vy = x'*P*x;
    derV = x'*(A'*P +P*A)*x;
    disp('Check if lyapunov function is > 0')
    disp(Vy)
    disp('Check if derivative of lyapunov function is < 0')
```

```
    disp(derVy)
    disp('Check eigenvalues of P are > 0')
    disp(eig(P))
    disp('Check determinant of P is > 0')
    disp(det(P))
    disp('Check P_11 > 0')
    disp(P(1,1))
    if (P(1,1) > 0 && Vy > 0 && det(P) > 0 && P(1,1) > 0 && derVy < 0)
        for i = eig(P)
            if i < 0
                break
            end
        end
       disp('This is asymptotically stable system because P is positive-
definite ')
    else
      disp('This is asymptotically stable system because P is not positive-
definite ')
    end
catch error
    % if error with lyapunov does not exist
    disp(error.message)
end
```

```
Eigenvalues of A
            0
            0
      -0.0125
            0
      -0.0125
            0

Q Matrix
     1     0     0     0     0     0
     0     1     0     0     0     0
     0     0     1     0     0     0
     0     0     0     1     0     0
     0     0     0     0     1     0
     0     0     0     0     0     1

The solution of this Lyapunov equation does not exist or is not unique.
```

Jordan Form J = V^-1*A*V

```
[V_g, J_A] = jordan(A);
disp(' ')
disp('Generalized eigenvector Matrix, V_g')
format shortG
disp(round(V_g))

J_B = inv(V_g)*B;
J_C = C* V_g;
J_D = D;
```

```
JordForm_ss = ss(J_A,J_B,J_C,J_D);
disp('Jordan form State-Space')
display(JordForm_ss)
```

```
Generalized eigenvector Matrix, V_g
        -785        62784     -5022720      5022720            0            0
           0            0            0            0          -80           80
           0            1            0            0            0            0
           0         -785        62784       -62784            0            0
           0            0            0            0            1            0
           0            0            1            0            0            0

Jordan form State-Space

JordForm_ss =

  A =
               x1        x2        x3        x4        x5        x6
   x1           0         1         0         0         0         0
   x2           0         0         1         0         0         0
   x3           0         0         0         0         0         0
   x4           0         0         0   -0.0125         0         0
   x5           0         0         0         0   -0.0125         0
   x6           0         0         0         0         0         0

  B =
                u1         u2
   x1    -0.02548          0
   x2           0          0
   x3       5.263          0
   x4       5.263          0
   x5           0       0.25
   x6           0       0.25

  C =
                  x1           x2           x3           x4           x5           x6
   y1         -784.8    6.278e+04   -5.023e+06    5.023e+06            0            0
   y2              0            0            0            0          -80           80

  D =
       u1   u2
   y1   0    0
   y2   0    0

Continuous-time state-space model.
```

```
%Transfer functions

% Convert Jacobian State-Space Form to local X Transfer Function
[X_num, X_den] = ss2tf(A,B,C,D,1);
X_tf = tf(X_num(1,:),X_den);
disp('Local X Transfer Function:')
```

```
display(X_tf)

% Visualize local X Transfer Function Poles
disp('X Transfer Function Poles:')
display(pole(X_tf));

% Visualize local X Transfer Function Zeros
disp('local X Transfer Function Zeros:')
display(zero(X_tf));


% Convert State-Space Form to Y Transfer Function
[Y_num, Y_den] = ss2tf(A,B,C,D,2);
Y_tf = tf(Y_num(2,:),Y_den);
disp('Local Y Transfer Function:')
display(Y_tf)

% Visualize Y Transfer Function Poles
disp('Y Transfer Function Poles:')
disp(pole(Y_tf));

% Visualize Y Transfer Function Zeros
disp('Y Transfer Function Zeros:')
disp(zero(Y_tf));
```

```
Local X Transfer Function:

X_tf =

  0.25 s^4 + 0.003125 s^3 - 51.63 s^2 - 0.6454 s
  ----------------------------------------------
         s^6 + 0.025 s^5 + 0.0001563 s^4

Continuous-time transfer function.
X Transfer Function Poles:
            0 +          0i
            0 +          0i
            0 +          0i
            0 +          0i
      -0.0125 + 1.4908e-10i
      -0.0125 - 1.4908e-10i

local X Transfer Function Zeros:
            0
       14.371
      -14.371
      -0.0125

Local Y Transfer Function:

Y_tf =

      0.25 s^4 + 0.003125 s^3
  -------------------------------
```

```
  s^6 + 0.025 s^5 + 0.0001563 s^4

Continuous-time transfer function.
Y Transfer Function Poles:
            0 +          0i
            0 +          0i
            0 +          0i
            0 +          0i
      -0.0125 + 1.4908e-10i
      -0.0125 - 1.4908e-10i

Y Transfer Function Zeros:
            0
            0
            0
      -0.0125
```

```
%Discrete Jacobian State Space
dt = 0.1; % Sampling time
SS_Discr = c2d(stateSpace,dt,'zoh');
disp('Discrete Jacobian State-Space')
display(SS_Discr)

eigsDiscrete = eig(SS_Discr.A);
disp('Eigenvalues of Discrete Jacobian State-Space')
disp(eigsDiscrete)
disp('Discrete Eigenvalues are enclosed in an unit circle.')
disp(' ')
```

```
Discrete Jacobian State-Space

SS_Discr =

  A = 
              x1         x2         x3         x4         x5         x6
   x1          1          0   -0.04903    0.09994          0  -0.001634
   x2          0          1          0          0    0.09994          0
   x3          0          0          1          0          0        0.1
   x4          0          0    -0.9804     0.9988          0   -0.04903
   x5          0          0          0          0     0.9988          0
   x6          0          0          0          0          0          1

  B = 
             u1        u2
   x1  0.001034         0
   x2         0  0.001249
   x3   0.02632         0
   x4   0.01638         0
   x5         0   0.02498
   x6    0.5263         0

  C = 
       x1  x2  x3  x4  x5  x6
```

```
   y1   1   0   0   0   0   0
   y2   0   1   0   0   0   0

  D =
        u1  u2
   y1   0   0
   y2   0   0

Sample time: 0.1 seconds
Discrete-time state-space model.
Eigenvalues of Discrete Jacobian State-Space
            1
            1
      0.99875
            1
      0.99875
            1

Discrete Eigenvalues are enclosed in an unit circle.


%Check controllability & observability

%SISO Controllable Canonical State-Space
%Note X and Y use the same A,B,D matrices
A_cntrl = [0 1 0 0 0 0;
0 0 1 0 0 0;
0 0 0 1 0 0;
0 0 0 0 1 0;
0 0 0 0 0 1;
0 0 0 0 -0.0001563 -0.025];

B_cntrl = [0;
0;
0;
0;
0;
1];

%1st row is X State-Space & 2nd row is Y State-Space
C_cntrl= [0 -0.6454 -51.63 0.003125 0.25 0; 0 0 0 0.003125 0.25 0];
D_cntrl = [0; 0];

%SISO Canonical controllable & observable check:
%Note: A is 6x6 matrix and full rank is 6
if rank(ctrb(A_cntrl,B_cntrl)) == size(A_cntrl,2) && ...
    rank(obsv(A_cntrl,C_cntrl)) ~= size(C_cntrl,2)
    disp('This SISO Canoncial Form is controllable, not observable')
end

%SISO Observable Canonical State-Space
%Note X and Y use the same A,C,D matrices
A_obs = [-0.025 1 0 0 0 0;
-0.000156 0 1 0 0 0;
```

```
0 0 0 1 0 0;
0 0 0 0 1 0;
0 0 0 0 0 1;
0 0 0 0 0 0];

B_obs_x = [0;
0.25;
0.003125;
-51.63;
-0.645;
0];

B_obs_y = [0;
0.25;
0.003125;
0;
0;
0];

C_obs= [1 0 0 0 0 0];
D_obs = 0;


%SISO Observable Canonical State-Space for X
if rank(ctrb(A_obs,B_obs_x)) ~= size(A_obs,2) && ...
   rank(obsv(A_obs,C_obs)) == size(C_obs,2)
   disp('The x SISO Observable Form is observable, not controllable')
end

%SISO Observable Canonical State-Space for Y
if rank(ctrb(A_obs,B_obs_y)) ~= size(A_obs,2) && ...
   rank(obsv(A_obs,C_obs)) == size(C_obs,2)
   disp('The y SISO Observable Form is observable, not controllable')
end


% Jacobian Matrices Controllability & Observability:
if rank(ctrb(A,B)) == size(A,2) && rank(obsv(A,C)) == size(C,2)
    disp('These Jacobian Matrices are both observable and controllable')
end
```

```
This SISO Canoncial Form is controllable, not observable
The x SISO Observable Form is observable, not controllable
The y SISO Observable Form is observable, not controllable
These Jacobian Matrices are both observable and controllable
```

Published with MATLAB® R2023a