## Contents



## Homework 7 Gabriel Colangelo

```
clear
close all
clc
```

## Problem 2

```
% Scaling parameters
lambda1 = 2;
lambda2 = 3;
mu1     = lambda1^2;
mu2     = lambda2^2;

% xdot  = Ax + B1*phi1 + B2*phi2
% zi    = Ci*X
C1      = [1 0 0 0];
C2      = [0 1 0 0];
B1      = [0 0 1 0]';
B2      = [0 0 0 1]';

% Initialize spring constant
K       = 0.1;

% Initialize counter
count   = 0;

% Initialize while loop logic
tfeas   = 1;

% Options for feasp - silent
opts    = [0;0;0;0;1];

% Create iterative loop
while tfeas > -.01

    % Increase counter
    count   = count + 1;

    % Create counter break
    if count > 1000
        disp('Stable spring constant not found')
        fprintf('\n')
        break
    end

    % LMI toolbox setup
    setlmis([]);

    % Constant matrix
    A       = [0 0 1 0; 0 0 0 1; -2*K K -2 1; K -K 1 -1];

    % Positive definite matrix
    P       = lmivar(1, [4,1]);

    % Create LMI -[PA+A'P+C1'C1+C2'C2 PB1 PB2; B1'P -I 0; B2'P 0 -I];
    lmi1    = newlmi;
```

```
    lmiterm([lmi1,1,1,P],1,A,'s');      % PA + A'P
    lmiterm([lmi1 1 1 0],mu1*(C1'*C1)); %mu1*C1'C1
    lmiterm([lmi1 1 1 0],mu2*(C2'*C2)); %mu2*C2'C2
    lmiterm([lmi1 1 2 P],1,B1);         %PB1
    lmiterm([lmi1 1 3 P],1,B2);         %PB2
    lmiterm([lmi1 2 2 0],-mu1);         %-mu1*I
    lmiterm([lmi1 3 3 0],-mu2);         %-mu2*I

    Plmi    = newlmi;
    lmiterm([-Plmi,1,1,P],1,1);
    lmiterm([Plmi,1,1,0],1);
    lmis    = getlmis;

    % Solve LMIS
    [tfeas, xfeas] = feasp(lmis,opts);

    % Create P matrix
    P       = dec2mat(lmis,xfeas,P);

    % If not feasible, increase K
    if tfeas > -.01
       K       = K + .1;
    end

end

% Check P
fprintf('\n')
disp('Final P is')
disp(P)

disp('Eigenvalues of P are')
disp(eig(P))

fprintf(['A spring constant value that guarantees' ...
        ' the system is globally exponentially stable about' ...
        ' the zero solution is K = %.1f \n'],K)

% Check QMI
Ctilde = [lambda1*C1;lambda2*C2];
Btilde = [lambda1^-1*B1, lambda2^-1*B2];
Q      = P*A + A'*P + P*Btilde*Btilde'*P + Ctilde'*Ctilde;

fprintf('\n')
disp('The QMI from Equation 14.37 is:')
disp(Q)
disp('With eigenvalues:')
disp(eig(Q))
```

```
Final P is
  639.3024 -384.3185    3.8190   -4.9162
 -384.3185  258.6842   -1.7025    3.3037
    3.8190   -1.7025   12.2526   -6.0652
   -4.9162    3.3037   -6.0652    6.3122

Eigenvalues of P are
    2.5066
   15.9522
   20.1819
  877.9106

A spring constant value that guarantees the system is globally exponentially stable about the zero solution is K = 20.9

The QMI from Equation 14.37 is:
 -514.4370  319.3492    2.8361    0.6291
  319.3492 -198.3229   -2.2082   -0.1117
```

```
    2.8361   -2.2082  -11.8837    7.3092
    0.6291   -0.1117    7.3092   -4.5236

With eigenvalues:
 -712.7215
  -16.3885
   -0.0450
   -0.0123
```

## Problem 4

```
% Laplace variable
s       = tf('s');

% Initial beta
beta    = 0;

% Initialize counter
count   = 0;

% Initialize Loop logic
logic   = 0;

% Define small posiitve alpha
alpha   = 1e-3;

while logic == 0

    % Increase counter
    count   = count + 1;

    % Create counter break
    if count > 1000
        disp('SPR beta not found')
        fprintf('\n')
        break
    end

    % Transfer Funtion
    g       = (beta*s + 1)/(s^2 + s + 2);

    % Transfer function to state space
    [A,B,C,D] = tf2ss(g.Numerator{1},g.Denominator{1});

    % Get size of A
    n        = length(A);

    cvx_begin sdp quiet

    % Variable definition
    variable P(n, n) symmetric

    % LMIs
    [(P*A + A'*P + 2*alpha*P), (P*B - C');...
     (B'*P - C), -(D + D')] <= -eps*eye(n+1);
    P >= eps*eye(n);
    cvx_end

    % Check in P = P' > 0
    logic = all(eig(P) > 0);

    % If logic is true, increase beta
    if logic == 0
       beta  = beta + .1;
    end
```

```
end

% Create controllability and observability gramian
Wc      = gram(ss(A,B,C,D),'c');
Wo      = gram(ss(A,B,C,D),'o');

% Check if gramian is invertible and A is Hurwitz
if det(Wc) ~= 0 && all(eig(A) < 0)
    disp('System is controllable')
end

if det(Wo)~= 0 && all(eig(A) < 0)
    disp('System is observable')
end

fprintf('The minimum beta needed for P = P'' > 0 is %.1f \n',beta)

LMI     = [(P*A + A'*P + 2*alpha*P), (P*B - C');...
           (B'*P - C), -(D + D')]

disp('The eigenvalues of the LMI are: ')
disp(eig(LMI))
```

```
System is controllable
System is observable
The minimum beta needed for P = P' > 0 is 1.1

LMI =

   -0.1978    0.4365    0.0000
    0.4365   -3.9927   -0.0000
    0.0000   -0.0000         0

The eigenvalues of the LMI are:
   -4.0423
   -0.1482
    0.0000
```

---

*Published with MATLAB® R2021b*