Mec6313 Homework 2

State feedback controller design for a system with uncertainties using linear matrix inequalities

Stefan Bracher

Version 2

Presented to: El-Kébir Boukas

November 28, 2007

Index

1. Introduction

To design a state feedback controller, there exist several different methods. One of them, the Linear Matrix Inequality (LMI) approach is used to stabilize a pick-and-place robot system. It is used because it promises that a robust controller is obtained, even if the physical properties of the system, for example the load mass, change. The result is then successfully verified with a simulation in SIMULINK.
The same system has already been stabilized in a previous work [1] using the pole placement method, which did not result in completely satisfying results, as the controller gains had to be fine tuned manually in order to avoid coupling of the control in x- and z-directions (Physically the two directions are completely independent, however, the pole-placement method provided a solution that contained some small gains that transmitted an error in x in a correction in z and vice versa, for details, see [1]).

2. The system to study: A pick and place robot

2.1 The system

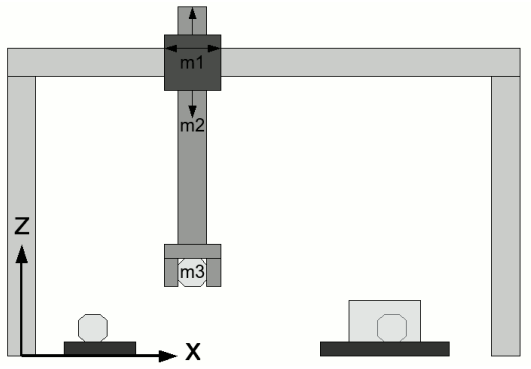

Figure 1: The pick and place robot

The system studied is a serial pick and place robot with two prismatic joints. It can move in the x-z plane to pick objects on one conveyor belt and place them on another.
Possible applications of such a robot are packaging or assembly.

Its parameters are:
Nominal mass fist member: $m_1 = 1\,kg$
Nominal mass second member: $m_2 = 1\,kg$
Load: $m_3 = 0 - 0.1\,kg$
Nominal friction coefficients: $k_1 = k_2 = 0.1\,kg/s$

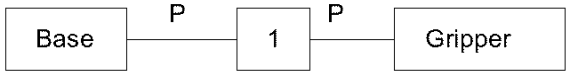

Figure 2: Kinematic chain

2.2 Basic Nominal Model

The mass of the object to be moved (m_3) as well as the gravity are not included in the basic model. Gravity is regarded as perturbation to the system while the object mass will be added as an uncertainty to m_2 in section 3. The system states are considered to be fully observable.

The Newton-Euler Equations of the system are:

$$F_x = \ddot{x}\cdot(m_1+m_2) + \dot{x}\cdot k_1$$
$$F_z = \ddot{z}\cdot m_2 + \dot{z}\cdot k_2$$

with the slip coefficients k_1, k_2

(2.2.1)

Which gives the following state model:

States: $\boldsymbol{x} = [x, \dot{x}, z, \dot{z}]'$
Inputs: $\boldsymbol{u} = [F_x, F_z]'$
Output: $\boldsymbol{y} = [x, \dot{x}, z, \dot{z}]'$

$$\dot{\boldsymbol{x}} = A_b\boldsymbol{x} + B_b\boldsymbol{u}$$
$$\boldsymbol{y} = C_b\boldsymbol{x} + D_b$$

with

$$A_b = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-k_1}{(m_1+m_2)} & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \frac{-k_2}{m_2} \end{bmatrix},\quad B_b = \begin{bmatrix} 0 & 0 \\ \frac{1}{(m_1+m_2)} & 0 \\ 0 & 0 \\ 0 & \frac{1}{m_2} \end{bmatrix},\quad C_b = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},\quad D_b = 0$$

(2.2.2)

2.3 Nominal Model with additional states

To be able to compensate for a continuous perturbation, a state feedback controller needs an integral part [1]. Thus the basic model is enriched by the states $\xi = \int x$ and $\zeta = \int z$ which gives:

States:	$\boldsymbol{x} = [\xi, x, \dot{x}, \zeta, z, \dot{z}]'$	$\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u}$
Inputs:	$\boldsymbol{u} = [F_x, F_z]'$	$\boldsymbol{y} = C\boldsymbol{x} + D$
Output:	$\boldsymbol{y} = [\xi, x, \dot{x}, \zeta, z, \dot{z}]'$	

with

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \frac{-k_1}{(m_1+m_2)} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & \frac{-k_2}{m_2} \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{1}{(m_1+m_2)} & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & \frac{1}{m_2} \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad D = 0 \tag{2.3.1}$$

The system states are still considered to be fully observable, which demands that the robot in question will be equipped with an integrator for ξ and ζ.

3. Uncertainties

Real systems always contain uncertainties. This can be included in the state model as [2]:

$$\dot{\boldsymbol{x}} = (A + \Delta A)\,\boldsymbol{x} + (B + \Delta B)\,\boldsymbol{u}$$
$$\boldsymbol{y} = C\,\boldsymbol{x} \tag{3.1}$$

The uncertainty in our system (2.3.1) is the load mass m_3 which has to be added to m_2. As the maximal load mass $m_{3\max}$ is known to be 0.1kg, the resulting uncertainty of m_2 is a norm bounded uncertainty of the form $m_2 = m_2 + \Delta$ with $\Delta < m_{3\max}$. To include this, according to [2], A and B can be written as:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -k_1 * \left(\frac{1}{(m_1+m_2)} + \Delta_1\right) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -k_2 * \left(\frac{1}{m_2} + \Delta_2\right) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \frac{-k_1}{(m_1+m_2)} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & \frac{-k_2}{m_2} \end{bmatrix} + D_A F_A E_A$$

with $D_A = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{1}{(m_1+m_2+m_{3\max})} - \frac{1}{(m_1+m_2)} & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & \frac{1}{(m_2+m_{3\mathrm{mac}})} - \frac{1}{m_2} \end{bmatrix}$, $1 \geqslant F_A(1,1) \geqslant 0$, $E_A = \begin{bmatrix} 0 & 0 & -k_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -k_2 \end{bmatrix}$

(3.2)

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{1}{(m_1+m_2)} + \Delta_3 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & \frac{1}{m_2} + \Delta_4 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{1}{(m_1+m_2)} & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & \frac{1}{m_2} \end{bmatrix} + D_B * F_B * E_B$$

with $D_B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{1}{(m_1+m_2+m_{3\max})} - \frac{1}{(m_1+m_2)} & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & \frac{1}{(m_2+m_{3\mathrm{mac}})} - \frac{1}{m_2} \end{bmatrix}$, $1 \geqslant F_B(1,1) \geqslant 0$, $E_B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

(3.3)

Where F_A and F_B are variables that can change between zero and one.

4. Stability: Linear matrix inequality (LMI) equivalent for the Lyapunov theorem

4.1 Nominal case

The Lyapunov theorem (see box on the right) to determine the stability of a system $\dot{\boldsymbol{x}} = A\,\boldsymbol{x}$ can be transformed in the following matrix inequalities (LMIs) [2]:

$$A^T P + PA < 0$$
$$P > 0 \qquad (4.1.1)$$

This LMI can be solved with Yalmip [3] and Sedumi [4], two Matlab-Add-ons.

Lyapuov stability theorem [2]
A system $\dot{\boldsymbol{x}} = A\,\boldsymbol{x}$ is stable if for any symmetric positiv-definite Q>0, there exists a unique symmetric positive matrix P>0 that solves: $A^T P + PA = -Q$

LMI stability theorem for nominal systems [2]
A system $\dot{\boldsymbol{x}} = A\,\boldsymbol{x}$ is stable if there exists a unique symmetric positive matrix P>0 that solves the linear matrix inequality: $A^T P + PA < 0$

For our nominal system (2.3.1), Yalmip [3] is not able to find a matrix P that solves the LMI** (This is indicated by a negative primary residual of the solver algorithm.).
Thus the stability can not be determined by the LMI (The theorem is not reciprocal), but looking at the physics it is clear that the system is instable as, given a little poke, it would never come back to its initial position.

**See Appendix stability_nom

4.2 Systems with norm-bounded uncertainties

Based on (4.1.1), a stability theorem using LMI can be elaborated for systems with norm-bounded uncertainties [2]:

LMI stability theorem for systems with norm bounded uncertainties [2]
The uncertain system $\dot{\boldsymbol{x}} = (A + \Delta A)\,\boldsymbol{x}$ is stable if there exist a symmetric and positive-definite matrix P > 0 such that the following LMI holds: $\begin{bmatrix} A^T P + PA + {E_A}^T E_A & P D_A \\ {D_A}^T P & -I \end{bmatrix} < 0$

Yalmip [3] does again not find a suitable matrix P**. It is clear that by adding a load to the pick-and-place robot, th system will not become stable.

**See Appendix stability_uncertain

5. Design of a state feedback controller

5.1 LMI Theorem for state-feedback control of norm-bounded uncertain systems

Similar to the theorems in the previous section for system stability, there is a theorem that can be used to determine if a system is stabilizable with a state-feedback controller:

State-Feedback Stabilization theorem, norm-bounded uncertain systems [2]

There exists a state feedback controller $\dot{x}=(A+BK)\boldsymbol{x}+B\boldsymbol{u}$ that stabilizes the uncertain system if there exist a symmetric and positive-definite matrix X > 0, a matrix Y and positive scalars ϵ_A > 0 and ϵ_B > 0 that satisfy the following LMI's:

X>0

$$\begin{bmatrix} J & X E_A^T & Y^T E_B^T \\ E_A X & -\epsilon_A I & 0 \\ E_B Y & 0 & -\epsilon_B I \end{bmatrix} < 0$$

where $J = X A^T + A X + B Y + Y^T B^T + \epsilon_A D_A D_A^T + \epsilon_B D_B D_B^T$.

The controller gain is given by $K = Y X^{-1}$

5.2 Application of the LMI Theorem to design the state-feedback controller

The idea is that if Yalmip and Sedumi [3, 4] find an X and a Y that solve the above theorem, we have found the gains K for a state feedback controller that stabilizes the system.

A numerical solution for K to stabilize the system can be obtained**:

$$K = \begin{bmatrix} -0.9673 & -2.9042 & -4.2310 & -0.0000 & -0.0000 & -0.0000 \\ 0.0000 & -0.0000 & -0.0000 & -0.6652 & -2.0772 & -1.8859 \end{bmatrix} \tag{5.2.1}$$

**See Appendix stability_uncertain

As desired, the system is decoupled, what means that an error in X will not produce a correction in Z and vice versa.

6 Simulation

To verify the theoretical results of the previous chapters, a simulation of the stabilized system with different load masses is done in Simulink, using K from (5.2.1).

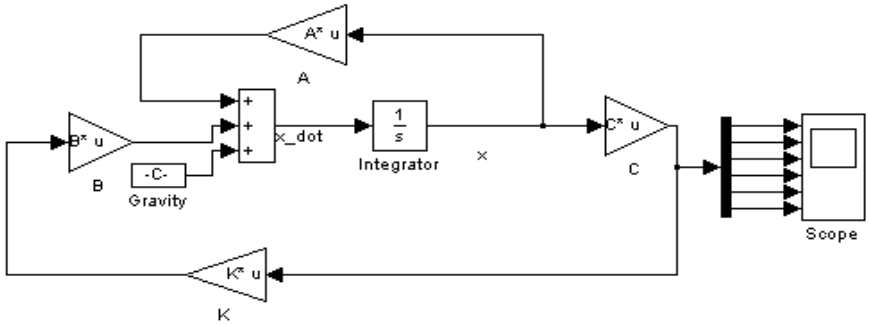

Figure 3: Simulink model

In order to test the robustness, two simulation runs are done, one with no load and one with a load of 0.1kg. Both simulations are started at initial positions x=1 and z=1 and include gravity.

No demanded trajectory was given to the system (Figure 3), thus there are no reference values. The only requirement to the system is to stabilize (return to its initial position).

6.1 No load

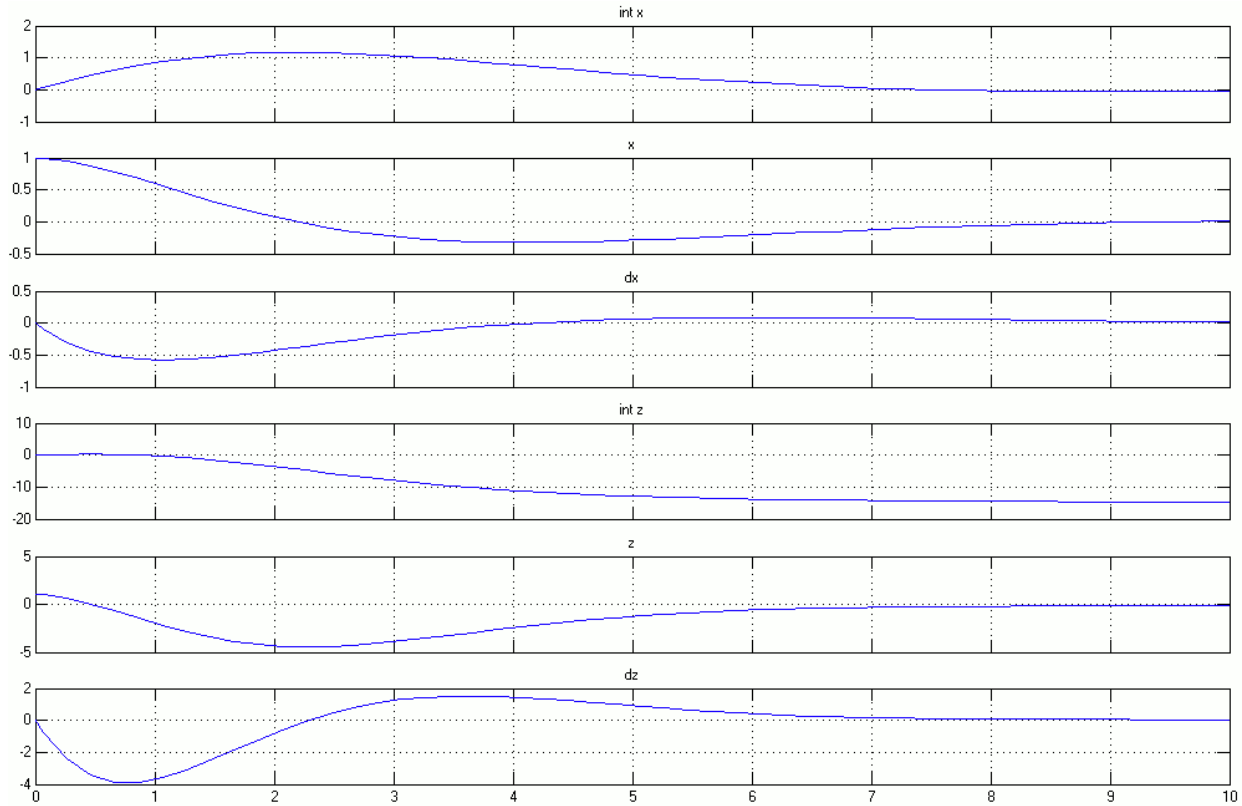

Figure 4: Simulation result with no load: The system stabilizes at x=0 and y=0

6.2 Load of 0.1kg

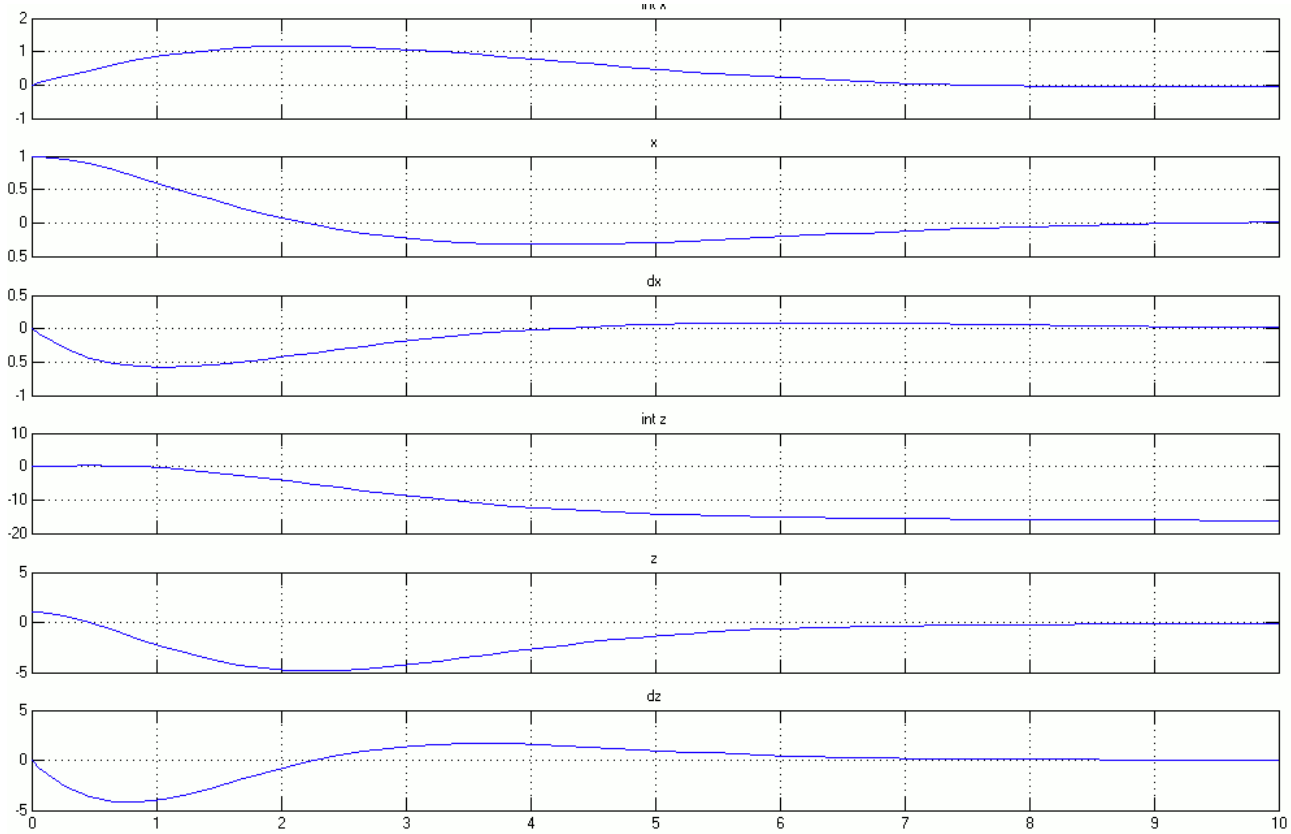

Figure 5: Simulation result with max load (0.1kg): The system stabilizes at x=0 and y=0. There is a small visible difference in the response of dz (compared to Figure 4)

7. Conclusion

The LMI method proved to be working for the analyzed pick-and-place robot system. The solution takes in account that the system consists of “two decoupled” systems, one for the x and one for the z direction and that a change on the x variables thus should not change any z variable and vice versa. This was not the case at a previous attempt to design a state feedback controller using the pole placement method [1].

The designed controller is successfully stabilizing the system for no load as well as maximum load. The resulting response time however is a bit slow. This is because the only requirement given in the LMI's, was to stabilize the system. To add performance, an additional $\Delta A = a \cdot I$ would have to be added to the system matrix A .

References

[1] Bracher Stefan,“State feedback controller and observer design for a pick and place robot”, Homeworks MEC6313, Ecole Polytechnique de Montréal, 2007

[2] Boukas El-Kebir*, “Class Notes MEC6313“*, Ecole Polytechnique de Montréal, 2007

[3] Löfberg J.*, “YALMIP : A Toolbox for Modeling and Optimization in MATLAB*.” In Proceedings of the CACSD Conference, Taipei, Taiwan, 2004.

[4] Sedumi: http://sedumi.mcmaster.ca

Appendix

stability_nom.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Script:       MEC 6313 Homework 2, stability_nom.m
% Description:  Checks the stability of the system A with LMIs
% Author:       Stefan Bracher
% Requirements:  Yalmip (http://control.ee.ethz.ch/~joloef/yalmip.php)
%                Sedumi (http://sedumi.mcmaster.ca)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Some cleaning up %%%
clear all;
clc;
yalmip('clear');

%%% Parameters %%%
m1=1;        % Mass of the first moving member, including motor
m2=1;        % Mass of the second moving member
k1=0.1;      % Friction coefficient member 1
k2=0.1;      % Friction coefficient member 2

%%% System %%%
A=[ 0   1    0            0     0    0;
    0   0    1            0     0    0;
    0   0   -k1/(m1+m2)   0     0    0;
    0   0    0            0     1    0;
    0   0    0            0     0    1;
    0   0    0            0     0    -k2/m2];

%%% LMI Variables %%%
n=size(A, 1);
P=sdpvar(n, n, 'symmetric'); % P symetric

%%% LMI %%%
F=set(P>0);              % P positive definit
F=F+set(A'*P+P*A<0);     % Lyaponov equivalent
Sol=solvesdp(F);         % Solving it

%%% Extract Data %%%
P=double(P)
[primal, dual]=checkset(F);   % Show primary and dual constraint residuals

if ((primal(1)>0)&&(primal(2)>0)&&(abs(dual(1))<1)&&(abs(dual(2))<1))
% The system is stable if the primal residuals are positive and the dual
% residuals are small
disp('The system is stable');
else
disp('No information about the stability of the system can be given');
end
```

stability_uncertain.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Script:       MEC 6313 Homework 2, stability_uncertain.m
% Description:  Checks the stability of the norm-bounded uncertain
%               system A with LMIs
% Author:       Stefan Bracher
%
% Requirements:  Yalmip (http://control.ee.ethz.ch/~joloef/yalmip.php)
%                Sedumi (http://sedumi.mcmaster.ca)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Some cleaning up %%%
clear all;
clc;
yalmip('clear');

%%% Parameters %%%
m1=1;       % Mass of the first moving member, including motor
m2=1;       % Mass of the second moving member
m3max=0.1;  % Maximal Load Mass
k1=0.1;     % Friction coefficient member 1
k2=0.1;     % Friction coefficient member 2

%%% System %%%
A=[ 0   1    0            0     0    0;
    0   0    1            0     0    0;
    0   0   -k1/(m1+m2)   0     0    0;
    0   0     0           0     1    0;
    0   0     0           0     0    1;
    0   0     0           0     0    -k2/m2];
EA=[0 0 -k1 0 0 0;
    0 0 0   0 0 -k2];
DA=[0                                   0;
    0                                   0;
    1/(m1+m2+m3max)-1/(m1+m2)           0;
    0                                   0;
    0                                   0;
    0                                   1/(m2+m3max)-1/m2];

%%% LMI Variables %%%
n=size(A, 1);
P=sdpvar(n, n, 'symmetric'); % P symetric

%%% LMI %%%
F=set(P>0);             % P positive definit
F=F+set([A'*P+P*A+EA'*EA     P*DA;
         DA'*P              -eye(2, 2)]<0);    % Lyaponov equivalent

Sol=solvesdp(F);       % Solving it

%%% Extract Data %%%
P=double(P)

[primal, dual]=checkset(F);   % Show primary and dual constraint residuals

if ((primal(1)>0)&&(primal(2)>0)&&(abs(dual(1))<1)&&(abs(dual(2))<1))
% The system is stable if the primal residuals are positive and the dual
% residuals are small
disp('The system is stable');
else
disp('No information about the stability of the system can be given');
end
```

state_feedback_uncertain.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Script:       MEC 6313 Homework 2, state_feedback_uncertain.m

% Description:  Finds the gains K to stabilize the system dx=(A+BK)x+Bu
%               with norm-bounded uncertainties.
% Author:       Stefan Bracher
% Requirements: Yalmip (http://control.ee.ethz.ch/~joloef/yalmip.php)
%               Sedumi (http://sedumi.mcmaster.ca)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Some cleaning up %%%
clear all;
clc;
yalmip('clear');

%%% Parameters %%%
m1=1;        % Mass of the first moving member, including motor
m2=1;        % Mass of the second moving member
m3max=0.1;   % Maximal Load Mass
k1=0.1;      % Friction coefficient member 1
k2=0.1;      % Friction coefficient member 2

%%% System %%%
A=[ 0   1    0             0     0    0;
    0   0    1             0     0    0;
    0   0   -k1/(m1+m2)    0     0    0;
    0   0     0            0     1    0;
    0   0     0            0     0    1;
    0   0     0            0     0    -k2/m2];
EA=[0 0 -k1 0 0 0;
    0 0 0   0 0 -k2];
DA=[0                                   0;
    0                                   0;
    1/(m1+m2+m3max)-1/(m1+m2)           0;
    0                                   0;
    0                                   0;
    0                                   1/(m2+m3max)-1/m2];
B=[0            0;
   0            0;
   1/(m1+m2)    0;
   0            0;
   0            0;
   0            1/m2];
EB=eye(2, 2);
DB=DA;
C=eye(6, 6);


%%% LMI Variables %%%
n=size(A, 1);
m=size(B, 2);
X=sdpvar(n, n, 'symmetric');
Y=sdpvar(m, n, 'full');
epA=sdpvar(1, 1);
epB=sdpvar(1,1);

%%% LMI %%%
 F=set(X>0);             % X positive definit
 F=F+set(epA>0);           % epA positive
 F=F+set(epB>0);           % epB positive
 F=F+set([X*A'+A*X+B*Y+Y'*B'+epA*DA*DA'+epB*DB*DB'      X*EA'          Y'*EB';
          EA*X  -epA*eye(2,2)   zeros(2, 2);
          EB*Y  zeros(2, 2)     -epB*eye(2,2)
            ]<0);

Sol=solvesdp(F);       % Solving it

%%% Extract Data %%%
X=double(X)
Y=double(Y);
K=Y*inv(X)
[primal, dual]=checkset(F);   % Show primary and dual constraint residuals
```

```
if ((primal(1)>0)&&(primal(2)>0)&&(abs(dual(1))<1)&&(abs(dual(2))<1))
% The system is stable if the primal residuals are positive and the dual
% residuals are small
disp('The system is stable');
else
disp('No information about the stability of the system can be given');
end
```

simulation_init.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Script:       MEC 6313 Homework 2, simulation_init.m
% Description:  Initializes the simulation
% Author:       Stefan Bracher
%
% Requirements:  state_feedback_uncertain.m
%                Yalmip (http://control.ee.ethz.ch/~joloef/yalmip.php)
%                Sedumi (http://sedumi.mcmaster.ca)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Calculate robust controller
state_feedback_uncertain

%%% Parameters %%%
m3=0;   % The load, 0 to 0.1 kg

%%% Update real mass for system %%%
m2=m2+m3;   % Use real Mass
```