Mec6313 Homework 1

State feedback controller and observer design for a pick and place robot

Stefan Bracher

Version 3

Presented to: El-Kébir Boukas

November 28, 2007

Content

1. The pick and place robot

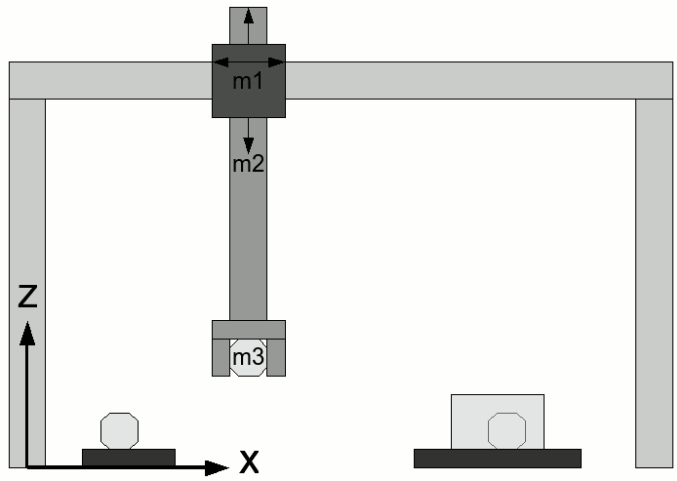

Figure 1: The pick and place robot

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:
$m_1 = 1\,kg$
$m_2 = 1\,kg$
$m_3 = 0 - 0.5\,kg$
$k_1 = k_2 = 0.1\,kg/s\,(friction\,coefficients)$

Kinematic chain:

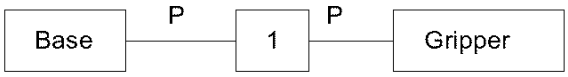

Figure 2: Kinematic chain

2. Model

The mass of the object to be moved (load mass m_3) as well as the gravity are not included in the model but regarded as disturbances to the system.

The Newton-Euler equations for the system without gravity

$$F_x = \ddot{x} \cdot (m_1 + m_2) + \dot{x} \cdot k_1 \tag{2.1}$$

$$F_z = \ddot{z} \cdot m_2 + \dot{z} \cdot k_2 \tag{2.2}$$

with the slip coefficients k_1, k_2

State Model

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

$$\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u}$$
$$\boldsymbol{y} = C\boldsymbol{x} + D$$

(2.3)

with

$$A=\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \dfrac{-k_1}{(m_1+m_2)} & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \dfrac{-k_2}{m_2} \end{bmatrix}=\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -0.05 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -0.1 \end{bmatrix} \tag{2.4}$$

$$B=\begin{bmatrix} 0 & 0 \\ \dfrac{1}{(m_1+m_2)} & 0 \\ 0 & 0 \\ 0 & \dfrac{1}{m_2} \end{bmatrix}=\begin{bmatrix} 0 & 0 \\ 0.5 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.5}$$

$$C=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

3. Open loop system properties

Observability

Theory	*Our system (Eq. 2.3)*
n: Number of states $O=[C^T, A^T C^T, ...(A^{n-1})^T C^T]$ The system is observable if rank(O)=n.	$O=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ ... & ... & ... & ... \end{bmatrix}$ -> rank(O)=4. (3.1) Thus the system is observable.

Controllability

Theory	*Our system (Eq. 2.3)*
n: Number of states $CT=[B, AB, ... A^{(n-1)}B]$ The system is controllable if rank(CT)=n.	$CT=\begin{bmatrix} 0 & 0 & 0.5 & 0 & ... \\ 1 & 0 & -0.025 & 0 & ... \\ 0 & 0 & 0 & 1 & ... \\ 0 & 0.5 & 0 & -0.1 & ... \end{bmatrix}$ (3.2) -> rank(CT)=4 Thus the system is controllable

Stability

Theory	*Our system (Eq. 2.3)*
Lyapunov theorem: A system is stable if there are two positive-definite and symmetric matrices P and Q so that $A^T P + PA = -Q$	As the matrix A has the form $\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -0.05 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -0.1 \end{bmatrix}$, it can be said directly that the system is not stable, as if deplaced, the system will not come back to it's initial state x_0, y_0

4. State Feedback stabilization

In order to stabilize the system, a state feedback controller is used:

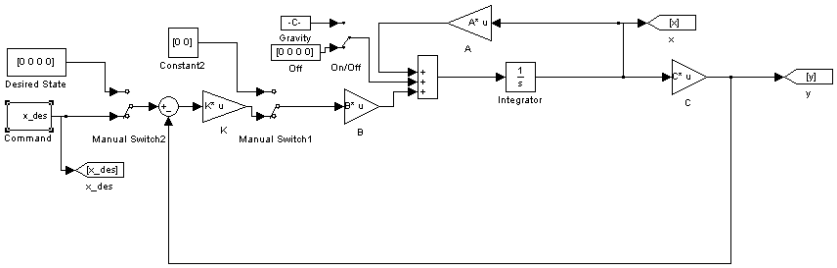

Figure 3: Block Model of the System mit state feedback

Closed Loop System:

$$\boldsymbol{u} = K(\boldsymbol{x}_{des} - \boldsymbol{x}_{real})$$
$$\dot{\boldsymbol{x}} = [A - BK]\boldsymbol{x} + BK\,\boldsymbol{x}_{des}$$
$$\dot{\boldsymbol{y}} = C\,\boldsymbol{x} \qquad (4.1)$$

Desired system properties

Reaction time T_r(5%)=0.2s

Maximal Overshoot d=2%

The gain matrix K is determined by pole placement in order to obtain the desired system properties.

The characteristic equation for the dominant poles is [1]:

$$s^2 + 2\xi\omega_n \cdot s + \omega_n^2 = 0 \qquad (4.2)$$

with $\omega_n = \dfrac{3}{\xi \cdot T_r}$, $d = 100\,e^{\frac{-\xi\pi}{\sqrt{(1-\xi^2)}}}$

The dominant poles become: $p_{1,2} = -\omega_n \xi \pm i \cdot \omega_n \cdot \sqrt{(1-\xi^2)}$ [2] (4.3)

The remaining poles are chosen to be $p_{3,4} = 2 * p_{1,2}$ (4.4)

Which gives the following desired poles: $p_{1,2} = -1.5000 \pm 1.2046\text{i}$

$p_{3,4} = -3.0000 \pm 2.4092\text{i}$ (4.5)

Solving* the characteristic equation of the closed loop system, $det(sI - A + BK) = 0$ with with Matlab* (4.6)

gives: $$K = \begin{bmatrix} 14.7895 & 8.8986 & -0.0093 & -2.4137 \\ -0.0047 & 1.2023 & 7.4094 & 4.4007 \end{bmatrix} \qquad (4.7)$$

However, by looking at the Matrix **A,** we know that x must z be independent, an error in x direction should not cause a movement in z direction. Tthus a matrix

$$K_{man} = \begin{bmatrix} 14.7895 & 8.8986 & 0 & 0 \\ 0 & 0 & 7.4094 & 4.4007 \end{bmatrix} \qquad (4.8)$$

is to be preferred.

*Matlab command “place(A, B, poles)”

5. Observer

Until now we had access to all states. But in order to save money on sensors, or because it is not possible to access the states, this might not be the case.

Our Matrix C could become: $C=\begin{bmatrix}0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$ (5.1)

But luckily, as the positions x and z of the system are just an integral of the velocities, an observer can be built to estimate the full state.

The observer has the form: $\dot{\hat{\boldsymbol{x}}}=A\hat{\boldsymbol{x}}+B\boldsymbol{u}+L[\boldsymbol{y}-C\hat{\boldsymbol{x}}]=[A-LC]\hat{\boldsymbol{x}}+B\boldsymbol{u}+L\boldsymbol{y}$ (5.2)

with characteristic equation: $det(sI-A+LC)=0$ (5.3)

As the observer should be faster than the controller itself, its poles should be more negative. We chose $P_{obs}=5*P$ (5.4)

Solving the characteristic equation (5.3) gives:

$$L=\begin{bmatrix}184.3681 & 1.6082\\ 22.4139 & -5.9353\\ 1.6012 & 185.7524\\ 6.1094 & 22.4361\end{bmatrix}$$ (5.5)

Again, because of the independence of x and z, L is changed in order to decouple the system:

$$L_{man}=\begin{bmatrix}184.3681 & 0\\ 22.4139 & 0\\ 0 & 185.7524\\ 0 & 22.4361\end{bmatrix}$$ (5.6)

*Matlab command “place(A, C', poles)'”

6. Simulation

State Feedback Control

The state feedback control from section 4 is simulated with Simulink.

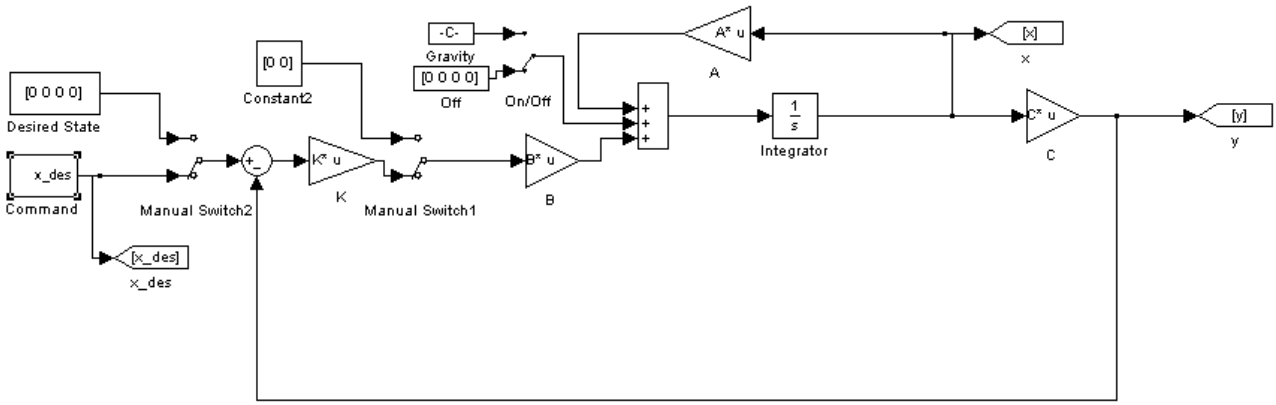

Figure 4: Simulink model of the closed loop system with state feedback (matrix C is the identity matrix, thus all states can be found in the output)

To test the system, a command change in x at t=2s and in z at t=6s sent.

In a first trial, the original K (4.7) is used and gravity turned on:

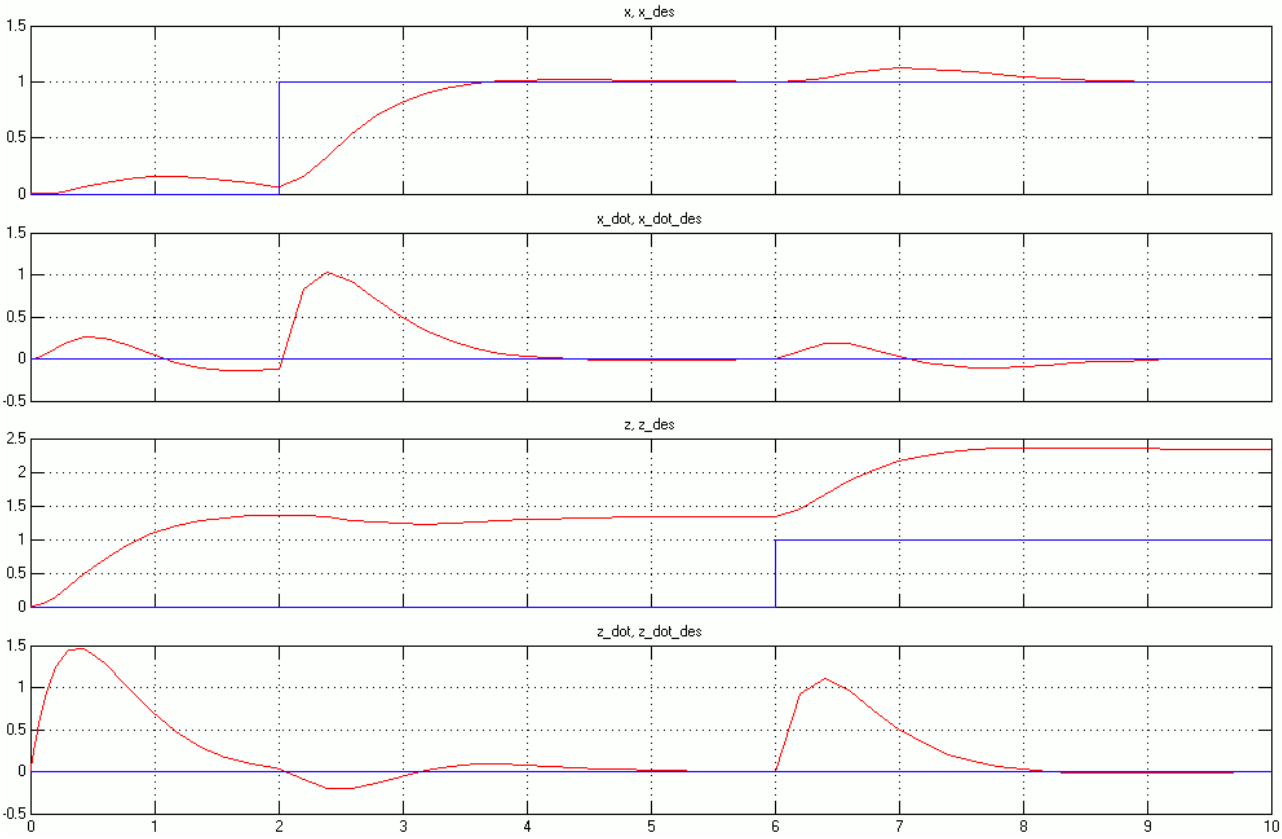

Figure 5: Commanded states (blue) and System states (red), K from Eq. 4.7, gravity on

From Figure 5 it can be seen that the controller is not able to eliminate the accumulated error in z due to gravity. An integral part would be necessary to do so. A solution to that problem is given in part 7, but for the following Simulations, gravity is turned off, to look at the performance of the PD-control.
Turning off the gravity corresponds to rotate the robot in the x-y plane.

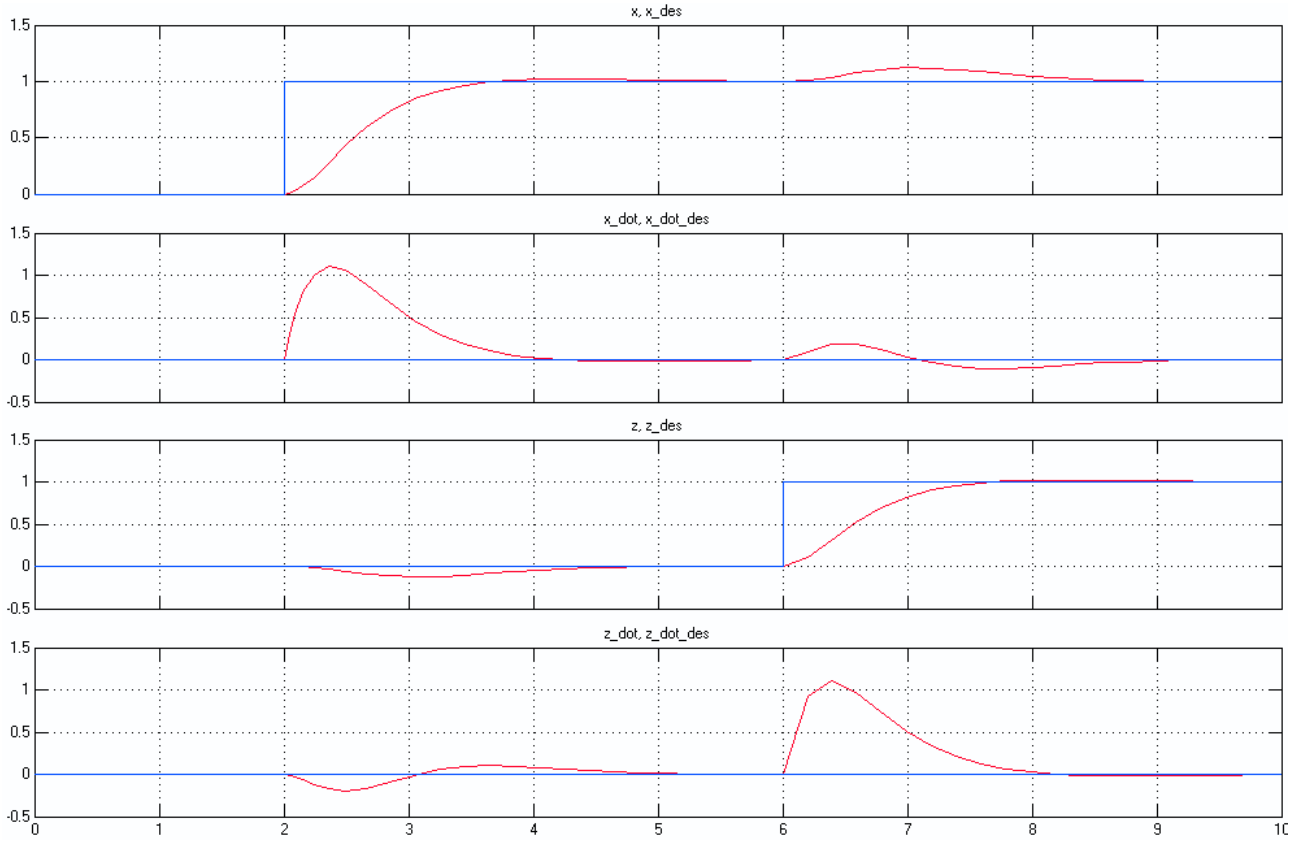

Figure 6: Commanded states (blue) and System states (red), K from Eq. 4.7, gravity off

With gravity off, the system can be stabilized, but as K (from Eq. 4.7) couples the error in z with x and vice versa, the controller introduces itself some errors.

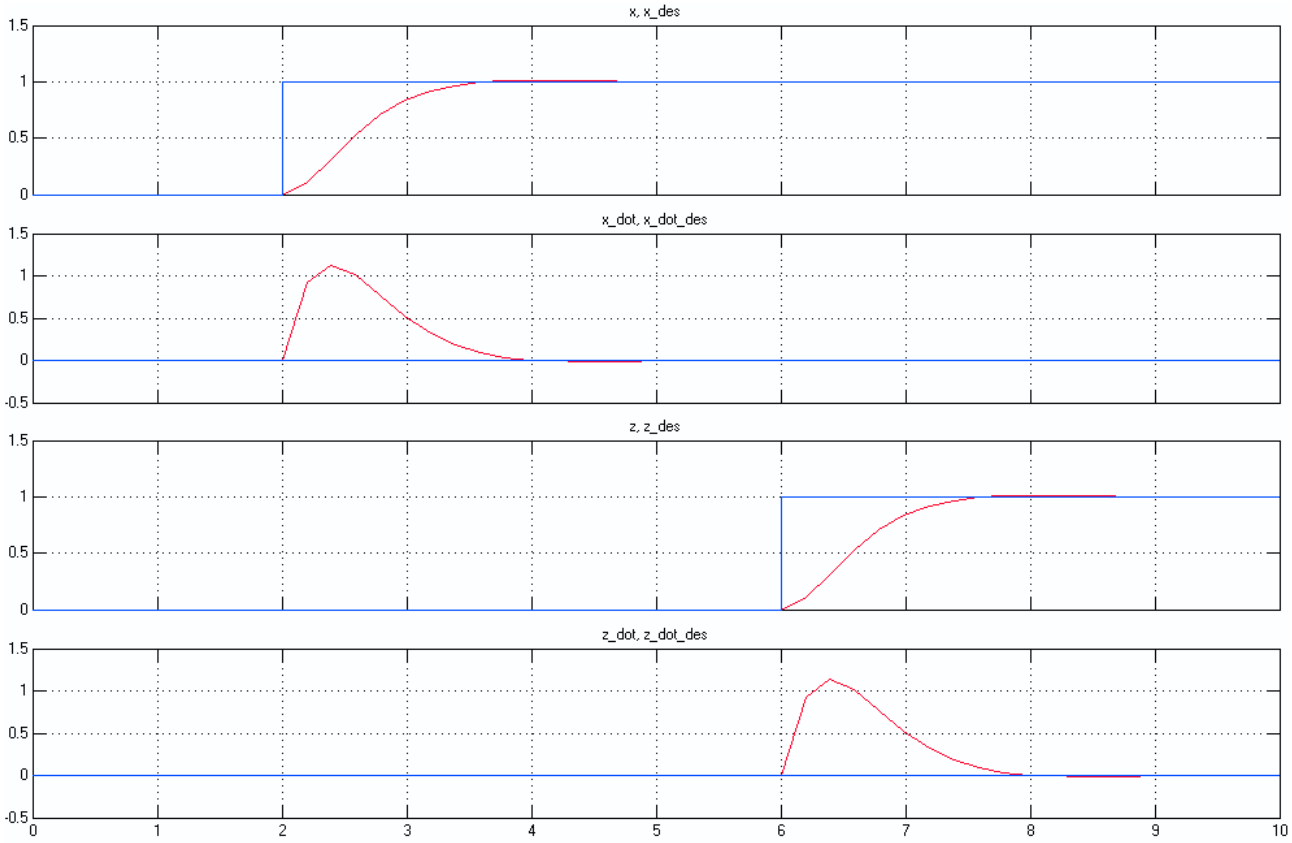

Figure 7: Commanded states (blue) and System states (red), K from Eq. 4.8, gravity off

Finally, by using K from equation 4.8, the system can be controlled with the desired properties specified in section 4 and produces no unnecessary errors by itself.

State Feedback Control + Observer

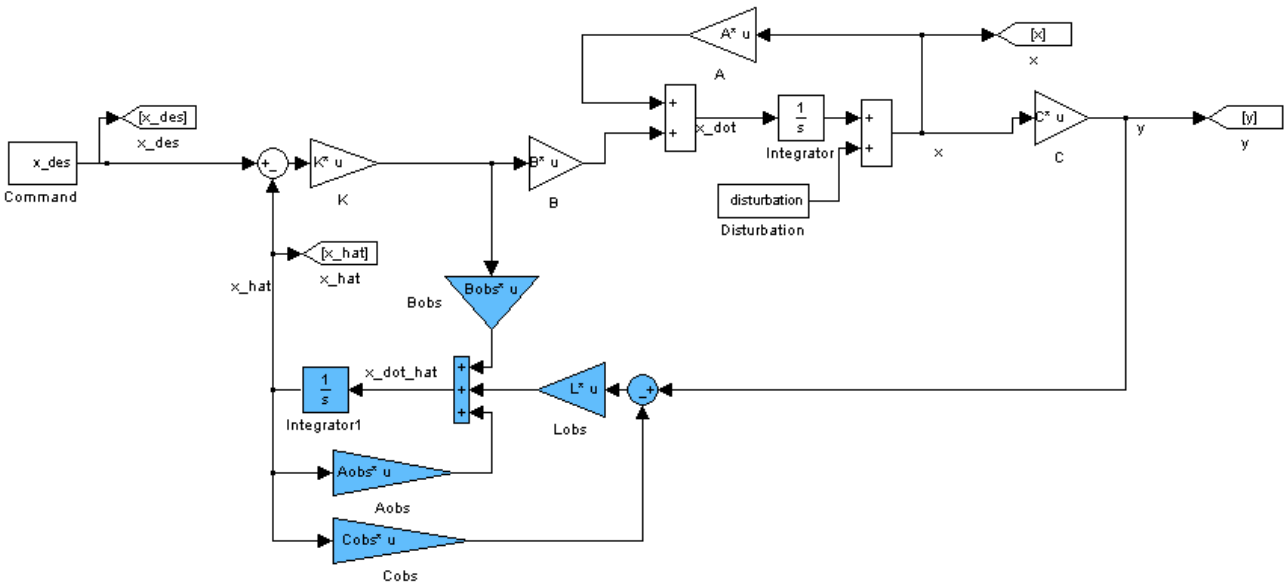

Figure 8: State feedback control with observer (blue blocks)

The simulation is repeated for the system with observer from section 5.

To omit the errors the proposed controller and estimator can not handle, gravity is turned off from the beginning and L from equation 5.6 is used. The gravity problem will be solved later on in section 7,

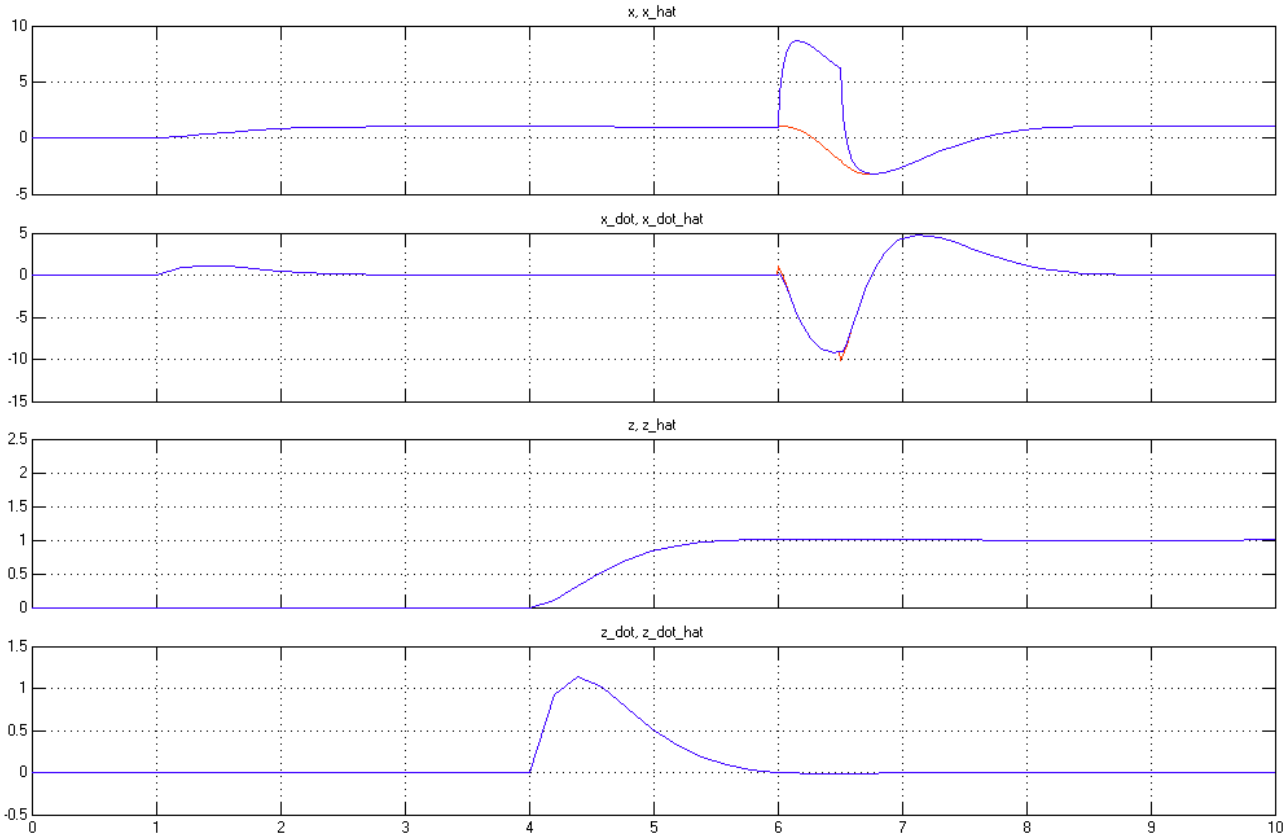

Figure 9: Estimated states (blue) and System states (red), L from Eq. 5.6, gravity off

It is clear that by using the exact system matrices A, B and C for the observer, the estimated states correspond perfectly with the system states (t<6s). At t=6s an error of 1 is introduced in $\dot{\boldsymbol{x}}$ for 0.5s. The observer is able to compensate this error. However this is only possible because it is not a continuous error, as then an integral part would be needed.

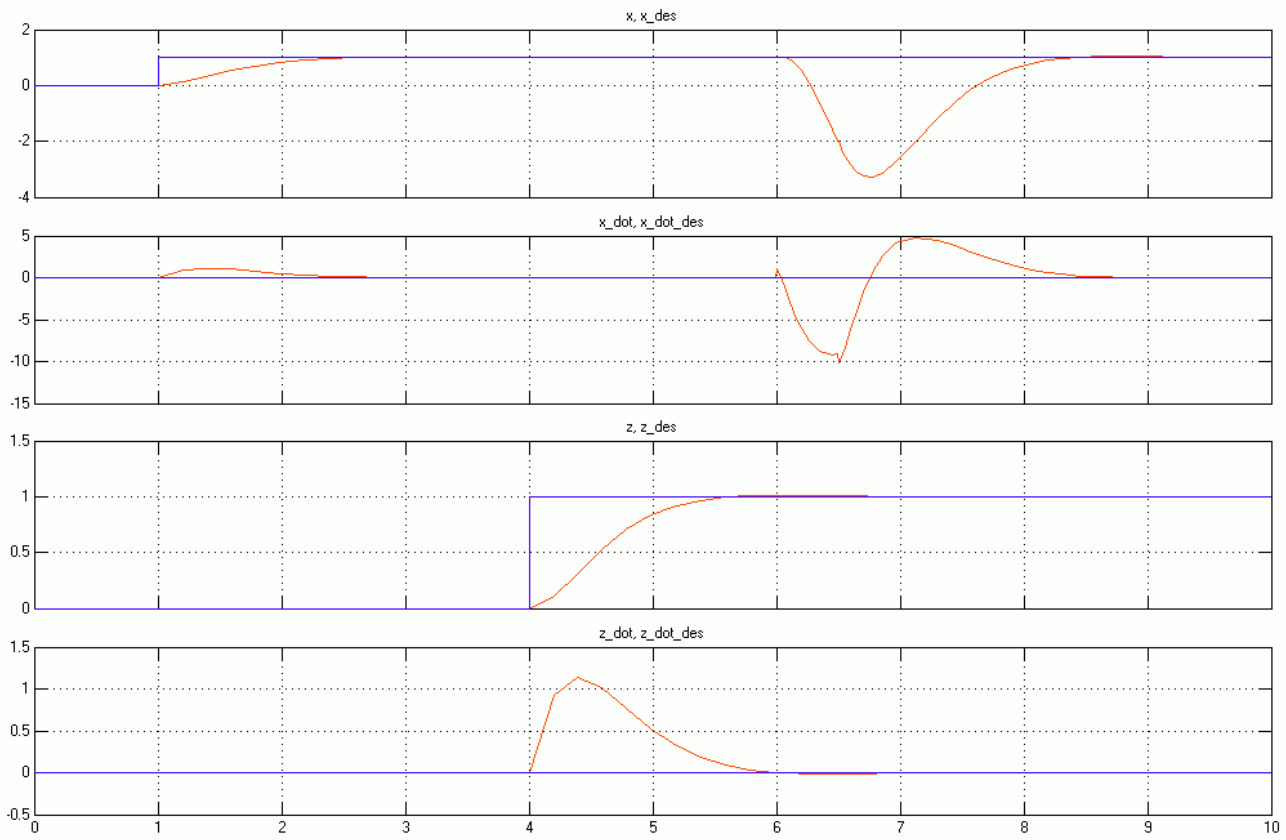

Figure 10: Desired states (blue) and System states (red), L from Eq. 5.6, gravity off

The control works as desired.

7. Control with integral part

In the simulation section it became clear that the designed control in sections 4 and 5 does not fit well for the pick and place robot problem, as it lacks integral parts and is thus unable to compensate for gravity.
Instead of simply using a PID controller, we will try to use the state feedback controller by adding two virtual states, the integral of the x and z position.

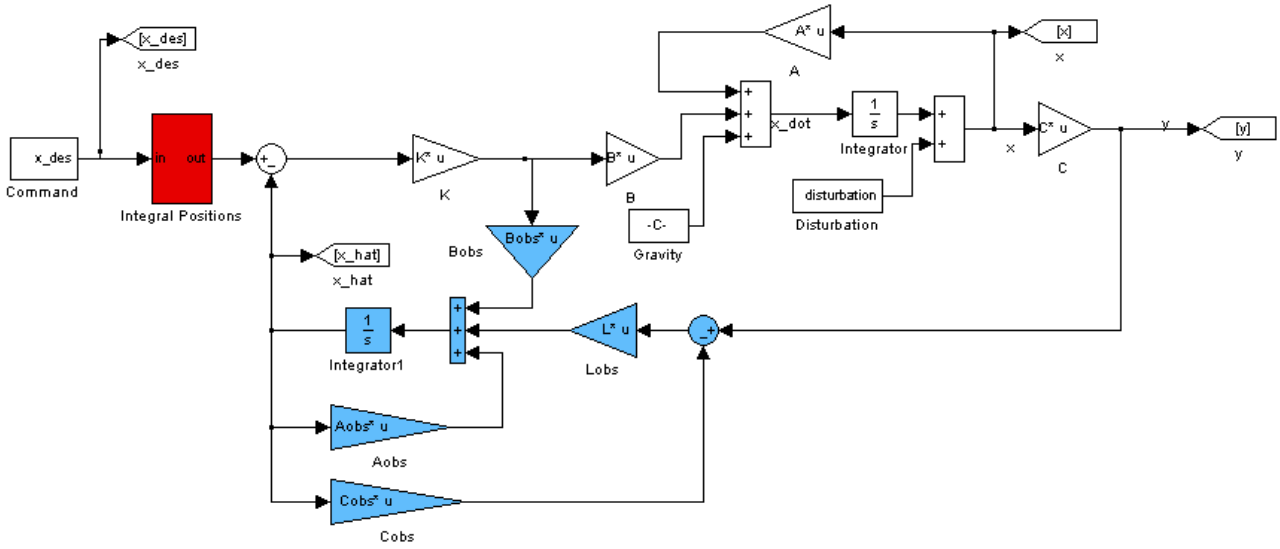

Figure 11: State feedback control with additional virtual states

From the command, the integrals of the positions are calculated and given to the controller as desired states (red box). By knowing only about the velocities, we would miss some errors, thus matrix C has to be changed so that as well information about x and z are given back. The matrices A and B of the system itself remain untouched.

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.1}$$

To calculate the controller gains, the virtual matrices Avirt and Bvirt are introduced.

$$A_{virt} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -0.05 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -0.1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}, \quad B_{virt} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{7.2}$$

In addition to 4.3, 2 additional poles are introduced as: $p_{5,6}=3*p_{1,2}$

(7.3)

The characteristic equation $det(sI-A_{virt}+B_{virt}K)=0$ is solved to get the controller gains.

After some manual changes, we use:

$$K_{man}=\begin{bmatrix} 65.0713 & 18.0375 & 0 & 0 & 69.9271 & 0 \\ 0 & 0 & 31.4651 & 8.8313 & 0 & 31.64217 \end{bmatrix}$$

(7.4)

The same is done for the observer:

$$P_{obs}=5*P$$

(7.5)

Characteristic equation: $det(sI-A_{virt}+LC_{virt})=0$

(7.6)

The L used is:

$$L_{man}=\begin{bmatrix} 22.1232 & 0 & 0 & 0 \\ 0 & 17.6718 & 0 & 0 \\ 0 & 0 & 26.9776 & 0 \\ 0 & 0 & 0 & 23.0775 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

(7.8)

The simulation result with gravity turned **ON** and at t=6s an error of “1” in $\dot{x}$ for 0.5s is showed in the following pictures. Unlike in section 6, the controller is now able to compensate the gravity.

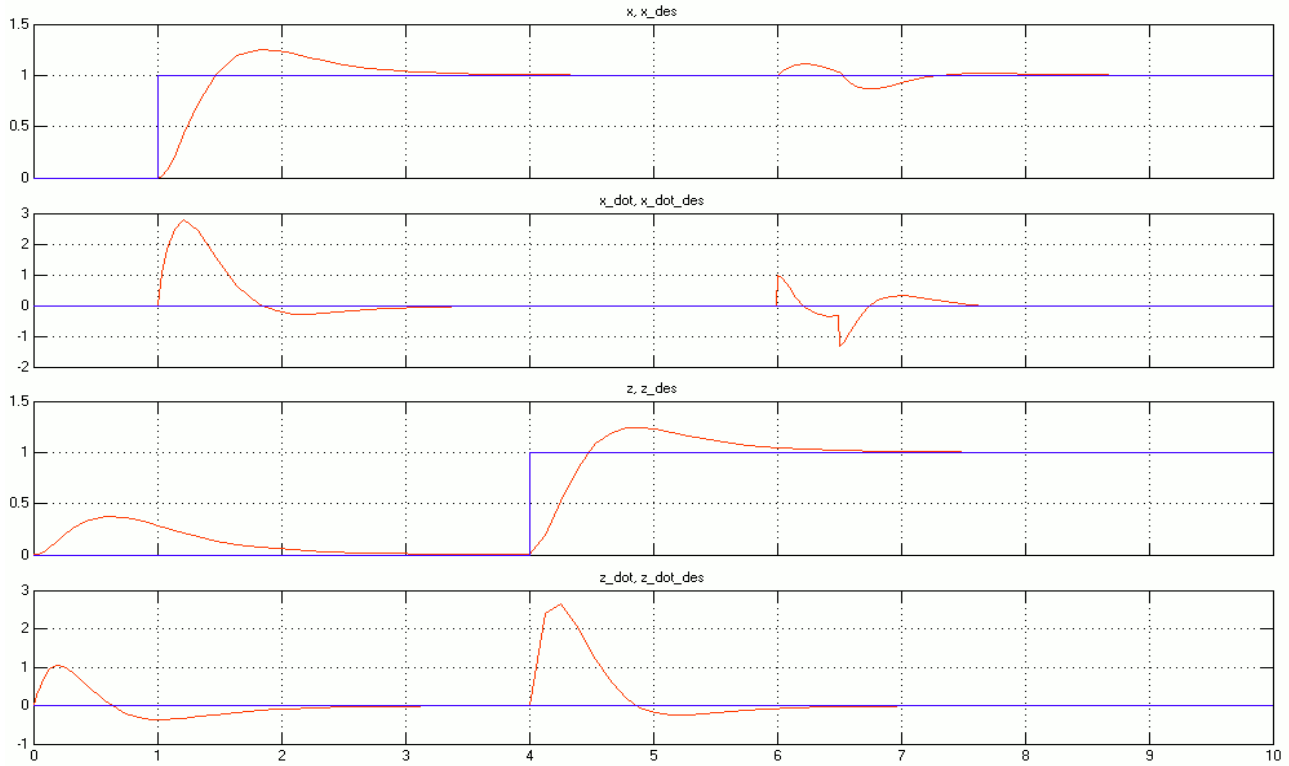

Figure 12: Desired states (blue) and System states (red), K and L from calculated with virtual states, gravity on, additional error in x at t=6s

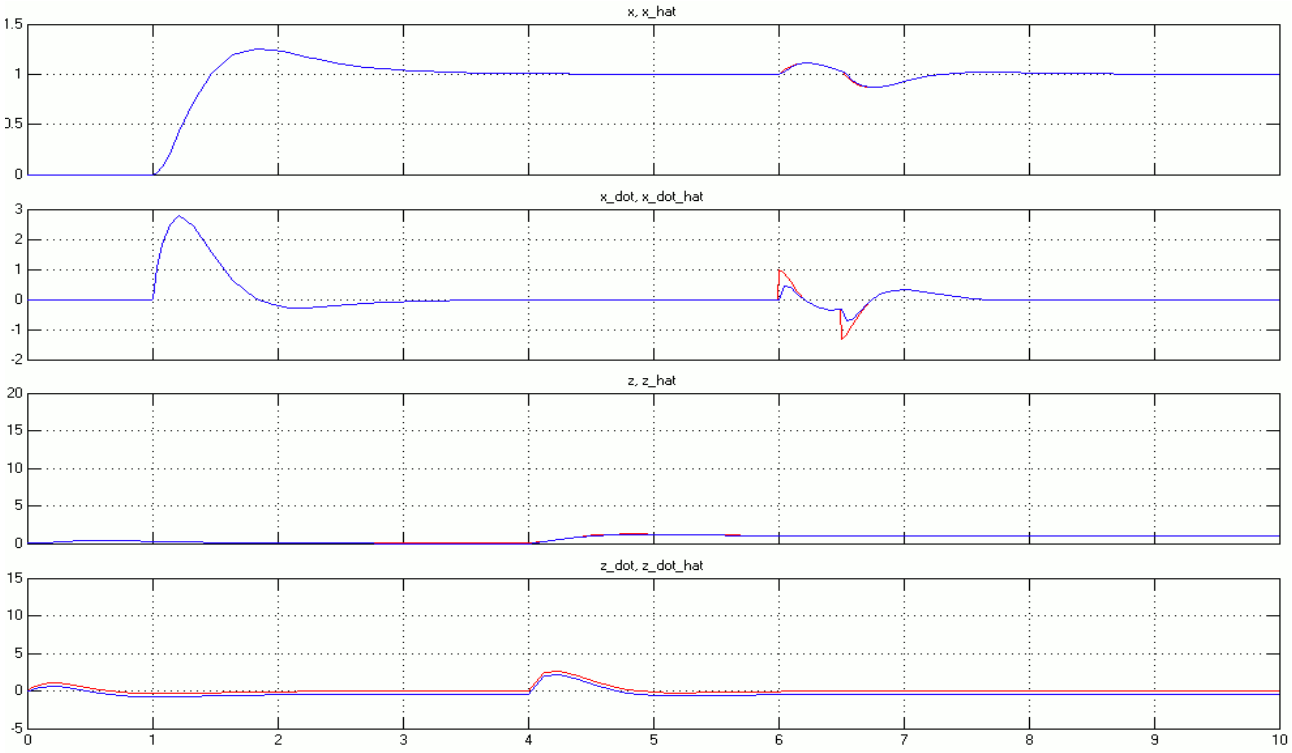

Figure 13: Estimated states (blue) and System states (red)

8. Conclusion

The simple state feedback controller with observer allows to stabilize the command of an open loop system, as long as there are no external static errors present. The pole placement method gave a good idea what the gain matrices K and L should look like. But manual optimization (decoupling) was necessary in order to have a good control.
For the robot presented, there are gravity, the additional weight of the load as well as mechanical imperfections present, not to speak about possible modeling errors. But to correct continuous errors like gravity, an integral part is necessary. An possible approach for a controller able to compensate the gravity effects was presented in section 7.

References

[1] *„Class Notes MEC6313“*, Ecole Polytechnique de Montréal, 2007

[2] *„Class Notes ELE6207“*, Ecole Polytechnique de Montréal, 2007

Appendix

p_p_init.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                % Script:       MEC 6313 Homework 1, Initialisation code
                % Description:  Analysis of a P-P Pick and place system,
            %               Initialisation of Simulink parameters for simulation
                    % Author:       Stefan Bracher
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                            %%%% Some cleaning up %%%
                                    clear all;
                                       clc;

                                %%% 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

                                d=2;        % Overshoot [%]
                            tr=2;       % Response Time(5%) [s]

                            %%% Open loop system matrixes %%%%%
                              A=[ 0    1               0    0;
                                   0   -k1/(m1+m2)    0    0;
                                  0   0                0    1;
                                 0   0               0    -k2/m2];

                                    B=[ 0               0;
                                         1/(m1+m2)    0;
                                        0               0;
                                       0              1/m2];

                                         C=[1 0 0 0;
                                            0 1 0 0;
                                            0 0 1 0;
                                           0 0 0 1];

                                %%% Open Loop Contrability %%%
                                        CT=ctrb(A, B)
                                      CT_rank=rank(CT)

                                %%% Open Loop Observability %%%
                                        O=obsv(A, C)
                                      O_rank=rank(O)

                                        % Controller
                          xi=abs(log(d/100)/sqrt(pi^2+log(d/100)^2));
                                        wn=3/(xi*tr);
                                p1=-wn*xi+i*wn*sqrt(1-xi^2);
                                p2=-wn*xi-i*wn*sqrt(1-xi^2);
                                   P=[p1, p2, 2*p1, 2*p2]

                                      %K=place(A, B, P)
                    %K=[14.7895    8.8986   -0.0093   -2.4137; % Tje K calculated by place
                         %   -0.0047    1.2023    7.4094    4.4007]
                        K=[14.7895    8.8986   0    0;  % Manual K that is much better
                                       0    0    7.4094 4.4007]
```

p_p2_init.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                % Script:       MEC 6313 Homework 1, Initialisation code, Part 2
          % Description:  Analysis of a P-P Pick and place system that needs an observer,
               %               Initialisation of Simulink parameters for simulation
                                  % Author:       Stefan Bracher
             %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                                      %%%% Some cleaning up %%%
                                             clear all;
                                                clc;

                                          %%% 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
                                       d=2;        % Overshoot [%]
                                   tr=2;       % Response Time(5%) [s]

                                   %%% Open loop system matrixes %%%%%
                                     A=[ 0    1               0    0;
                                          0   -k1/(m1+m2)    0    0;
                                         0   0                0    1;
                                       0   0               0    -k2/m2];

                                          B=[ 0                0;
                                               1/(m1+m2)     0;
                                             0                0;
                                           0               1/m2];

                                               C=[ 0 1 0 0;
                                                   0 0 0 1];

                                      %%% Open Loop Contrability %%%
                                             CT=ctrb(A, B)
                                           CT_rank=rank(CT)

                                      %%% Open Loop Observability %%%
                                             O=obsv(A, C)
                                           O_rank=rank(O)

                                            % Controller
                             xi=abs(log(d/100)/sqrt(pi^2+log(d/100)^2));
                                             wn=3/(xi*tr);
                                       p1=-wn*xi+i*wn*sqrt(1-xi^2);
                                       p2=-wn*xi-i*wn*sqrt(1-xi^2);
                                          P=[p1, p2, 2*p1, 2*p2]

                  %K=place(A, B, P)  % Solving the characteristic equation
               %K=[14.7895    8.8986   -0.0093   -2.4137; % THe K calculated by place
                            %   -0.0047    1.2023    7.4094    4.4007]
                    K=[14.7895    8.8986   0    0;  % Manual K that is much better
                                       0    0    7.4094 4.4007]

                                       % Observer
               Aobs=A;     % Observator matrices, here some errors could be included
                           Bobs=B;     % to make it more realistic
                                         Cobs=C;

              Pobs=5*P        % Observator Poles must be faster than controller poles
                   % L=place(A, C', Pobs)' % Solving the characteristic equation
                                           %
                          % L =[184.3681    1.6082;% L obtained by place
                                       %    22.4139   -5.9353;
                                       %     1.6012  185.7524;
                                       %     6.1094   22.4361]
                              L =[184.3681    0;% L modified manually
                                           22.4139   0;
                                            0  185.7524;
                                            0   22.4361]
```

p_p3_init.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Script:       MEC 6313 Homework 1, Initialisation code, Part 3
% Description:  Analysis of a P-P Pick and place system that needs an observer,
%               Initialisation of Simulink parameters for simulation
%              Realisation of integral controller with virtual states
%                % Author:       Stefan Bracher
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                        %%%% Some cleaning up %%%
                                clear all;
                                   clc;

                              %%% 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
                         d=2;        % Overshoot [%]
                     tr=2;       % Response Time(5%) [s]

                     %%% Open loop system matrixes %%%%%
                       A=[ 0    1              0    0;
                            0   -k1/(m1+m2)    0    0;
                           0   0                0    1;
                          0   0               0    -k2/m2];
                           B=[ 0                0;
                                1/(m1+m2)     0;
                               0              0;
                              0              1/m2];
                                  C=[ 1 0 0 0;
                                      0 1 0 0;
                                      0 0 1 0;
                                       0 0 0 1];

                    %%% Virtual open loop system matrixes %%%
                                Avirt=zeros(6, 6);
                                Avirt(1:4, 1:4)=A;
                              Avirt(5:6, 1:6)=[1 0 0 0 0 0; 0 0 1 0 0 0];
                                Bvirt=zeros(6, 2);
                                Bvirt(1:4, 1:2)=B;
                                Cvirt=zeros(4, 6);
                                Cvirt(1:4, 1:4)=C;

                                % Controller
                             xi=abs(log(d/100)/sqrt(pi^2+log(d/100)^2));
                                    wn=3/(xi*tr);
                               p1=-wn*xi+i*wn*sqrt(1-xi^2);
                               p2=-wn*xi-i*wn*sqrt(1-xi^2);
                            P=[p1, p2, 2*p1, 2*p2, 3*p1, 3*p2]

          %K=place(Avirt, Bvirt, P)  % Solving the characteristic equation
        %K=[  65.0713   18.0375   21.6412    4.8093   69.9271   53.0123; % K by
        %  -10.9072   -2.4197   31.4651    8.8313  -27.1154   31.6421]  %place
                K=[  65.0713   18.0375   0    0   69.9271   0; % Manual
                       0   0   31.4651    8.8313  0   31.6421]  %K

                                % Observer
      Aobs=Avirt;     % Observator matrices, here some errors could be included
                     Bobs=Bvirt;     % to make it more realistic
                                Cobs=Cvirt;
          Pobs=5*P; % Observator Poles must be faster than controller poles
          %L=place(Aobs, Cobs', Pobs)' % Solving the characteristic equation
                                      %
              % L =[22.1232   -3.7372    8.0311    2.6379; % L by place
                     %     0.5819   17.6718   -4.3702   13.5896;
                     %    -5.5426    5.4021   26.9776   -6.4295;
                     %     3.4222  -20.3625   -1.3682   23.0775;
                     %   174.8495  -37.4266   36.1641   23.0089;
                     %    42.3615   20.9822  260.7304  -45.2737]
                    L =[22.1232   0       0         0;   % L manual
                              0         17.6718 0         0;
                              0         0       26.9776   0;
                              0         0       0   23.0775;
                                 1         0       0   0;
                                 0         0       1   0]
```