ELEC-E8116 Model-based control systems /exercises with solutions 8

Problem 1. Consider the system

$$\frac{Y(s)}{U(s)} = \frac{s+0.5}{s^2+2s+4}$$ and the criterion to be minimized $J = \int_0^\infty (3y^2 + 0.5u^2)\, dt$.

Write a *Matlab m-file* to do the following:

Solve the optimal control law by using the *lqr*-function in Matlab. Calculate the *damping ratio* of the closed loop system. Simulate the system by letting the reference signal be zero (regulator problem) and letting the initial states be non-zero. Then consider the tracking problem. Use a static pre-compensator to set the static gain of the closed-loop system to the value 1. Then simulate the system for a step change in the reference signal.

Solution. Note that Matlab's *lqr* function uses the cost $J = \int_0^\infty \left(x^T Q x + u^T R u \right) dt$

$$J = \int_0^\infty (3y^2 + 0.5u^2)dt = \int_0^\infty (3y^T y + 0.5u^2)dt = \int_0^\infty (3(Cx)^T Cx + 0.5u^2)dt$$

$$\int_0^\infty (x^T (3C^T C)x + 0.5u^2)dt$$

The Matlab m-code shows the solution and simulation.

The process model in state-space form is

A =

```
   -2   -2
    2    0
```

B =

```
    1
    0
```

C =

```
    1.0000    0.2500
```

D =

0

The process and optimal controller are

$$\dot{x} = Ax + Bu$$
$$y = Cx$$
$$u = -Lx$$

which leads to the closed loop representation

$$\dot{x} = (A - BL)x$$
$$y = Cx$$

The state feedback matrix, the Riccati equation solution and the closed loop poles are

L =

1.2197 0.0917

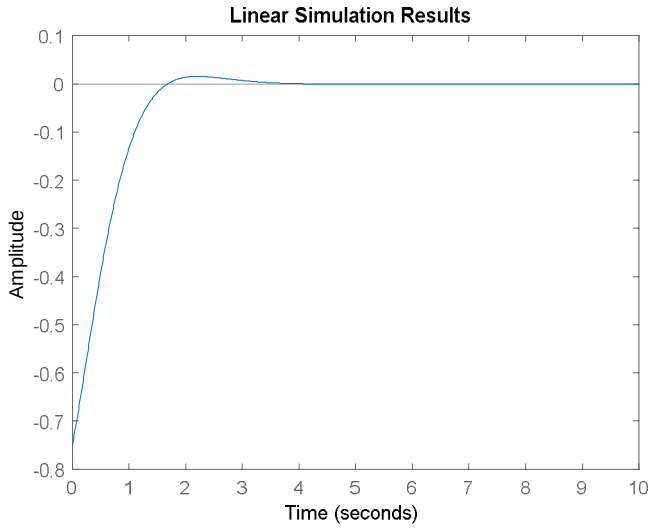
S =

0.6099 0.0458
0.0458 0.3086

E =

-1.6099 + 1.2616i
-1.6099 - 1.2616i

The damping ratio is 0.787. The simulation result using the initial condition [-1 1] is shown in the figure. Note that it shows the output signal *y*. If both states should be plotted, you should have defined the C-matrix as an identity matrix. The system would not change, but both states would be defined as outputs.



As for the servo problem, the controller is $u = -Lx + kr$ where k is a constant and r the reference. The closed loop is

$$\dot{x} = (A - BL)x + kBr$$
$$y = Cx$$

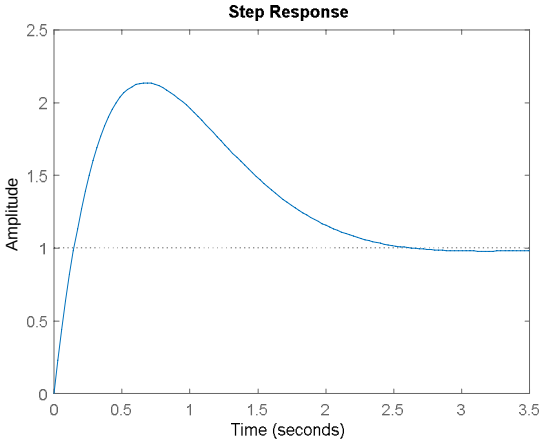
The corresponding transfer function from reference to output is

$$G_{cl}(s) = kC\left(sI - A + BL\right)^{-1} B$$

The static gain is $G_{cl}(0)$, and it must be set to 1 in order the output to follow the reference. Hence

$$k = \frac{1}{C(-A + BL)^{-1} B}$$

The simulation result for a step change in r is shown below.



```
% Model-based control systems
% Exercise 9, Problem 1
%
%Model
s=tf('s');
G=(s+0.5)/(s^2+2*s+4);
Gss=ss(G);
[A,B,C,D]=ssdata(Gss);
%Cost function
Q=3*C'*C; R=0.5;
%Optimal control u=-Lx, regulator problem
[L,S,E]=lqr(A,B,Q,R);
damp(A-B*L);
%Closed loop
Gclss=ss(A-B*L,zeros(2,1),C,0);
%Simulation
x0=[-1,1]';
T=0:0.01:10;
U=zeros(size(T));
lsim(Gclss,U,T,x0)
%Optimal control u=-Lx+kr, servo problem
k=1/(C*inv(-A+B*L)*B);
G2clss=ss(A-B*L,k*B,C,0);
figure
step(G2clss)
```

Problem 2:

Consider the multivariable plant

$$G(s)=\frac{1}{(0.2s+1)(s+1)}\begin{bmatrix}1 & 1\\ 1+2s & 2\end{bmatrix}$$

a. Use RGA analysis to evaluate how bad the interconnections between the channels are. Calculate RGA both at zero frequency and the gain crossover frequency. Choose the preferred *pairing* and design *decentralized* PID controllers to control the system. Implement the controller in Matlab/Simulink and plot the responses of the outputs when **a.** a unit step enters the reference of channel 1, **b.** a unit step enters the reference of channel 2 and **c.** unit steps enter at both channels simultaneously. You may use Matlab in implementing the controller and simulating the closed loop, or you can use Simulink if you wish.

b. Design a decoupling controller using the singular value decomposition at zero frequency and choosing the weight matrices W_1 and W_2 accordingly (see Chapter 6 in the lecture slides). Use PID-controllers in the decoupled system. Simulate as in part a.

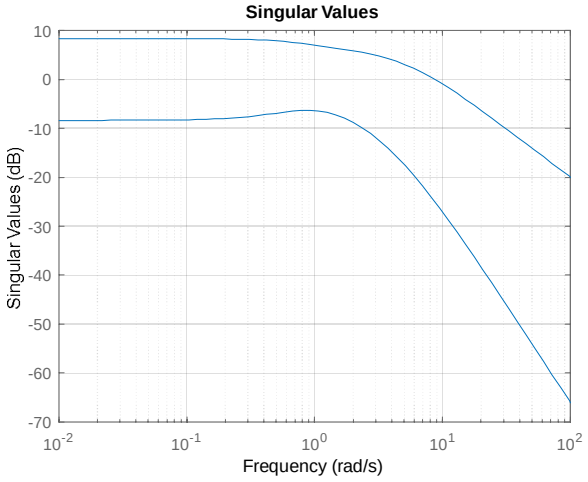
Hints: Gain crossover of a MIMO system is calculated based on the largest singular value. In tuning the PID controllers you may use the tuning functions in Matlab, see e.g. *pidtune*.

Solution:

The Matlab code is at the end of the solution. (Note that many of the commands have been set as comments, %. The idea is that for different experiments only a part of the commands is made active by removing the %.)

a. The RGA at zero frequency is

$RGAG0 = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$. By the command *sigma* we get the singular values of $G(s)$.



The gain crossover (larger singular value) is approximately 8.8 rad/s. The RGA as a function of angular frequency is

$$RGA = G(j\omega).*G^{-1}(j\omega)^T = \frac{1}{(1+j0.2\omega)(1+j\omega)}\begin{bmatrix} 1 & 1 \\ 1+j2\omega & 2 \end{bmatrix}$$
$$.*\frac{(1+j0.2\omega)(1+j\omega)}{1-j2\omega}\begin{bmatrix} 2 & -(1+j2\omega) \\ -1 & 1 \end{bmatrix} = \frac{1}{1-j2\omega}\begin{bmatrix} 2 & -(1+j2\omega) \\ -(1+j2\omega) & 2 \end{bmatrix}$$

Clearly, at the zero angular frequency we get the same result as above. But for ω = 8.8 the result becomes (after some algebra)

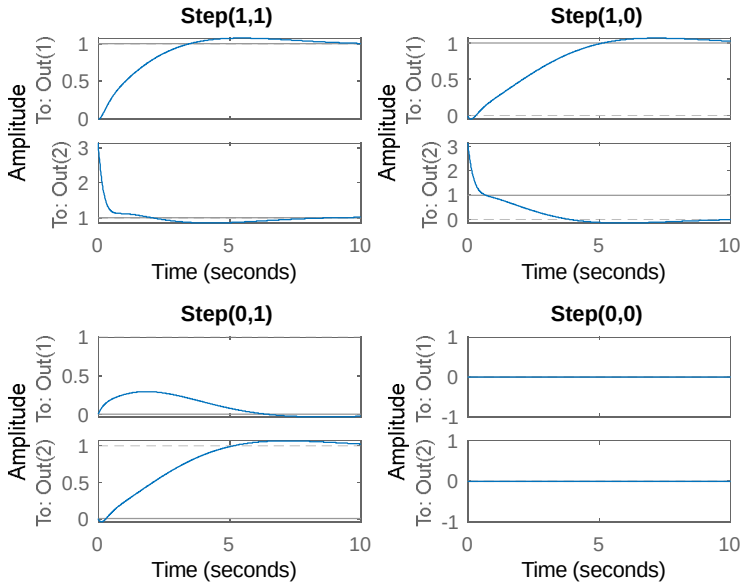
$$RGA \approx \begin{bmatrix} 0.0064 + j0.1133 & 0.9936 - j0.1133 \\ 0.9936 - j0.1133 & 0.0064 + j0.1133 \end{bmatrix}$$

(As for the algebra, for example

$$\frac{2}{1 - j2\omega} = \frac{2(1 + j2\omega)}{(1 - j2\omega)(1 + j2\omega)} = \frac{2 + j4\omega}{1 + 4\omega^2} = \frac{2}{1 + 4\omega^2} + j\frac{4\omega}{1 + 4\omega^2}$$)

Now it is interesting to note that at the zero angular frequency the preferable pairing would seem to be $u_1 \leftrightarrow y_1, u_2 \leftrightarrow y_2$, although neither one is really good. But at the gain crossover frequency the pairing $u_1 \leftrightarrow y_2, u_2 \leftrightarrow y_1$ should be preferred.

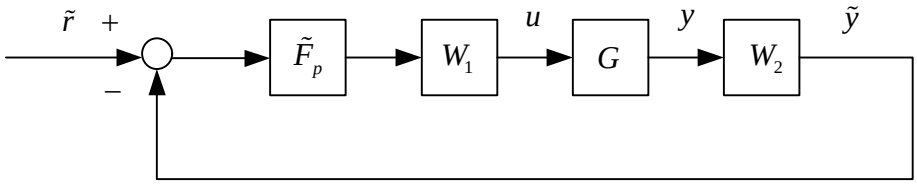
Simulation: PID ($u_1 \leftrightarrow y_1, u_2 \leftrightarrow y_2$) tuned by Matlab's *pidtune*. The solution is unstable. The tuning values for *Fy1* were *Kp* = 1.95, *Ki* = 2.65, *Kd* = 0.314, and for *Fy2* *Kp* = 0.973, *Ki* = 1.32, *Kd* = 0.157. But when they are changed to *Fy1* *Kp* = 1, *Ki* = 1, *Kd* = 0.314, *Fy2 Kp* = 0.5, *Ki* = 0.5, *Kd* = 0.157 the following result is obtained

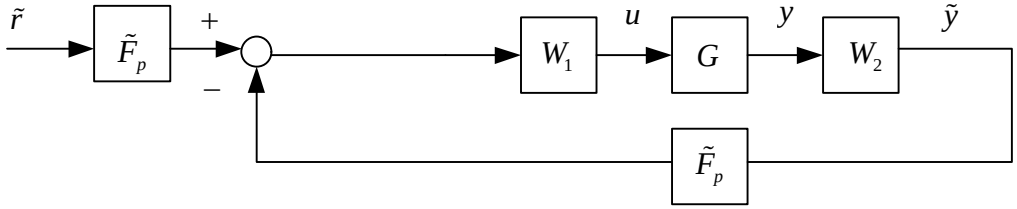


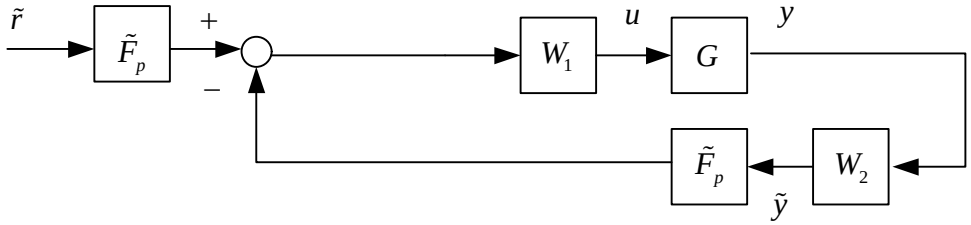
PID ($u_1 \leftrightarrow y_2, u_2 \leftrightarrow y_1$) TUNED BY Matlab's *pidtune*. The result is unstable. In this case it turned out to be difficult to re-tune the controller such that a good response would be obtained.

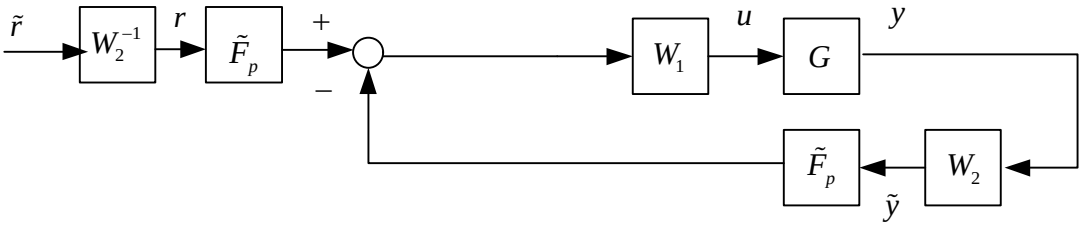
Conclusion: *pidtune* is for SISO tuning only. There is no guarantee that it would be good for MIMO systems. The RGA analysis showed that at zero frequency the coupling $u_1 \leftrightarrow y_1, u_2 \leftrightarrow y_2$ should be preferred. For step responses that seems to be so. At the gain crossover frequency the pairing $u_1 \leftrightarrow y_2, u_2 \leftrightarrow y_1$ seemed to be preferable, but that could not be observed by simulations with step reference inputs.

b. There are many ways to solve this. The solution below is based on the below figure, from which the program code becomes understandable. See also Chapter 6 in the lecture slides.

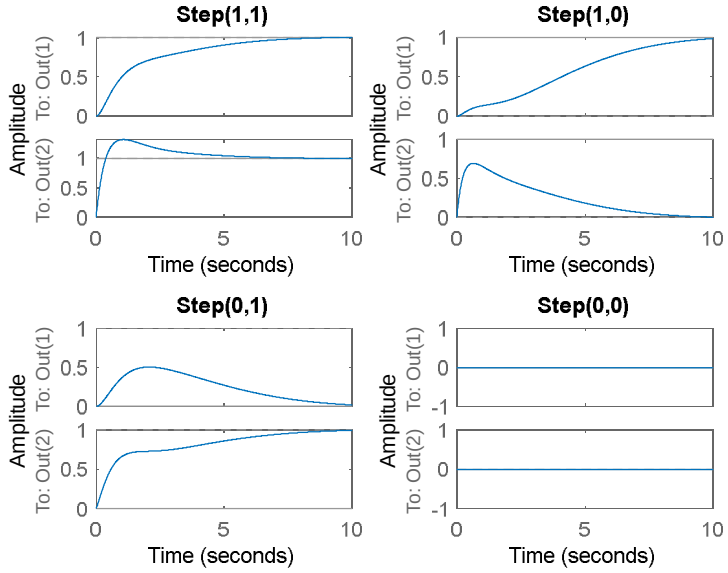








The last part of the figure can be explained by noting that since $y = W_2^{-1}\tilde{y}$ then also $r = W_2^{-1}\tilde{r}$. The reference signal has to be scaled to get the static gain from *r* to *y* to be 1. The following results are obtained by the below program code. The result is acceptable but a bit slow. Again, the PID tuning could be improved.



Program code:

```
% Model-based control systems
%
%
s=tf('s');
G=1/((0.2*s+1)*(s+1))*[1 1;1+2*s 2];
G0=[1 1;1 2];
RGAG0=G0.*(inv(G0))';
sigma(G);
%
% Let us try a pairing u1-y1, u2-y2 first.  PID
control.
%[Fy1,Info1]=pidtune(G(1,1),'pid');
%[Fy2,Info2]=pidtune(G(2,2),'pid');

%1 DOF controllers
%Fy=[Fy1 0;0 Fy2];
%Fr=Fy;

% Let us then try a pairing u1-y2, u2-y1.  PID
control.
%[Fy1,Info1]=pidtune(G(1,2),'pid');
%[Fy2,Info2]=pidtune(G(2,1),'pid');
```

```
%1 DOF controllers
%Fy=[0 Fy1;Fy2 0];
%Fr=Fy;

%SVD and construction of pre- and post
compensators
[U,S,V]=svd(G0);
W1=V; W2=U';
%Diagonalazied plant; design of controller
Gworm=minreal(W2*G*W1);
[Fp1worm,Info1]=pidtune(Gworm(1,1),'pid');
[Fp2worm,Info2]=pidtune(Gworm(2,2),'pid');
%1 DOF controller in "worm" domain
Fpworm=[Fp1worm 0;0 Fp2worm];
% 2 DOF
Fy=Fpworm*W2;
Fr=Fpworm*inv(W2);
G=G*W1; % Modified plant

%Sensitivity functions
L=minreal(G*Fy);
S=minreal(inv(eye(2)+L));
Tcomp=minreal(eye(2)-S);
Gc=S*G*Fr;
Gc2=Gc;

figure(1)
sigma(L,S,Tcomp)
title('Functions L, S, T')
%
%
%Simulation
T=0:0.01:10;
figure(2)
Uinp=ones(length(T),2);
subplot(221)
lsim(Gc2,Uinp,T)
```

```
title('Step(1,1)')
%
Uinp=[ones(length(T),1) zeros(length(T),1)];
subplot(222)
lsim(Gc2,Uinp,T)
title('Step(1,0)')
%
Uinp=[zeros(length(T),1) ones(length(T),1)];
subplot(223)
lsim(Gc2,Uinp,T)
title('Step(0,1)')
%
Uinp=[zeros(length(T),1) zeros(length(T),1)];
subplot(224)
lsim(Gc2,Uinp,T)
title('Step(0,0)')
```