**Universität Stuttgart**
**Institut für Systemtheorie und Regelungstechnik**
Prof. Dr.–Ing. Frank Allgöwer

# $H_\infty$-control

## 1 Introduction

Typically the concept of $H_\infty$ controller design is fairly easy to grasp. However, as controller synthesis is done numerically, a major problem for people new to the subject is *how to write the Matlab code*. I will here try to give a short overview of some useful Matlab functions. Hopefully this will help you when trying to design your first $H_\infty$-controller.

There are *many* $H_\infty$ related functions available in Matlab and its toolboxes. The important toolboxes are, in addition to the Control System Toolbox, the mu-Analysis and Synthesis Toolbox (mu-tools), the Robust Control Toolbox (RCT) and the LMI Control Toolbox. LMI and mu-tools are both included in RCT v.3.0.1 which comes with Matlab 7, in earlier versions they are separate.

I have also prepared an m-file where I have tried to use as many of the functions discussed here as possible. The m-file is included in the appendix and can also be downloaded from the robust control webpage.

A mixed $S/KS$ synthesis problem will be used to illustrate the use of a handful of useful functions. Let's take a look at the this problem first.

## 2 Shaping closed loop transfer functions

The mixed S/KS problem can be illustrated with the block diagram shown in Figure 1. The closed loop transfer function $T = F_l(P, K)$ from $w$ to $z$ can be found by visual inspection as

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} W_s S \\ W_{ks} K S \end{bmatrix} r. \tag{1}$$

The generalized plant $P(s)$ (see Figure 2) is

$$\left[\begin{array}{c} z_1 \\ z_2 \\ \hline e \end{array}\right] = \left[\begin{array}{c|c} W_s & -W_s G \\ 0 & W_{ks} \\ \hline I & -G \end{array}\right] \left[\begin{array}{c} r \\ \hline u \end{array}\right]. \tag{2}$$

If we have the state space realizations

$$G \stackrel{s}{=} \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right], \quad W_s \stackrel{s}{=} \left[\begin{array}{c|c} A_s & B_s \\ \hline C_s & D_s \end{array}\right], \quad W_{ks} \stackrel{s}{=} \left[\begin{array}{c|c} A_{ks} & B_{ks} \\ \hline C_{ks} & D_{ks} \end{array}\right],$$

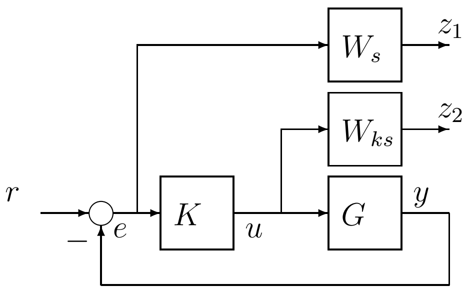


Figure 1: Mixed S/KS problem.

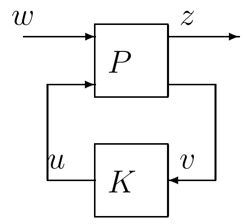


Figure 2: General control problem.

it can be shown that a possible state space realization for $P(s)$ is given by

$$P \stackrel{s}{=} \left[\begin{array}{ccc|cc} A_s & 0 & -B_sC & B_s & -B_sD \\ 0 & A_{ks} & 0 & 0 & B_{ks} \\ 0 & 0 & A & 0 & B \\ \hline C_s & 0 & -D_sC & D_s & -D_sD \\ 0 & C_{ks} & 0 & 0 & D_{ks} \\ 0 & 0 & -C & I & -D \end{array}\right]. \tag{3}$$

(I leave this as an exercise for you.)

The weights $W_s$ and $W_{ks}$ are your tuning parameters, and it typically requires some iterations to obtain weights which will yield a good controller. That being said, a good starting point is to choose

$$W_s = \frac{s/M + \omega_0}{s + \omega_0 A}; \quad W_{ks} = const. \tag{4}$$

where $A < 1$ is the maximim allowed steady state offset, $w_0$ is the desired bandwidth and $M$ is the sensitivity peak (typically $A = 0.01$ and $M = 2$). For the controller synthesis, the inverse of $W_s$ is an upper bound on the desired sensitivity loop shape, and $W_{ks}^{-1}$ will effectively limit the controller output $u$.

In some cases, you would also like to shape the complementary sensitivity function $T = GK(I + GK)^{-1}$ (done by adding an extra output $z_3 = W_t y$ in Figure 1). A starting point is to choose

$$W_t = \frac{s + \omega_0/M}{As + \omega_0}, \tag{5}$$

which is symmetric to $W_s$ around the line $\omega = \omega_0$. The two weighting functions are shown in Figure for the parameter values $A = 0.01(= -40dB)$, $M = 2(= 6dB)$ and $\omega_0 = 1$ rad/sec.

# 3 Obtaining the subsystems

There are several ways to obtain the dynamical systems $G$, $W_s$ and $W_{ks}$ in Matlab. Methods you probably already have heard about are `ss`, `tf` and `zpk` in Control System Toolbox. Mu-tools offer a variety of similar possibilities like `pck`, `nd2sys` and `zp2sys`. Other methods are `mksys` and `tree`. You should be aware however, that mu-tools uses a different

Figure 3: Inverse of weighting functions $W_s$ and $W_t$

.

representation than the Control System Toolbox, called a system matrix. Thus you cannot just pass a system generated with e.g. `Gcst = ss(A,B,C,D)` in Control System Toolbox to a function found in mu-tools (with RCT v.3.0.1 this is no longer so, most functions have been rewritten to accept both system representations). Which one to choose is a matter of convenience, you can transfer back and forth between the different representations quite easily. One possibility is to write `[A,B,C,D]=ssdata(Gcst); Gmu=pck(A,B,C,D)`. The opposite way would be `[A,B,C,D]=unpck(Gmu); Gcst = ss(A,B,C,D)`. Take a look at the documentation to see other options.

# 4 Obtaining the generalized plant $P$

Also in creating $P$ you have many options. I list five:

1. Write down the transfer function matrix in (2) directly. I prefer to use mu-tools for this option. If you afterwards convert to state-space, you should use e.g. `minreal` to obtain a minimal realization. Useful commands: `sbs` (side-by-side), `abv` (above), `mmult` (multiply), `minv` (inverse).

2. Write down the state space matrices A,B,C,D in (3) and use `P = pck(A,B,C,D)`.

3. Use `sysic` (system interconnect), an m-file in mu-tools where you specify your subsystems and the interconnection between them.

4. Use `sconnect`, a function in LMI-tools where subsystems, inputs and outputs are passed as parameters, and `sconncet` returns the connected system.

5. Use `iconnect` in RCT v3.0.1, functionally similar to `sysic`.

Of these methods I personally prefer `sysic` and `iconnect` because they are flexible and easy to use also for more complex systems where method 1 and 2 are no longer feasible.

Generally it is a good idea to use a balanced realization to avoid numerical problems. A balanced realization can be obtained e.g. with `balreal` in Control System Toolbox.

# 5 Synthesizing controller

The $H_\infty$ S/KS synthesis problem is to find a controller $K$ which stabilizes $G$ and minimizes the $H_\infty$ cost function

$$\|F_l(P,K)\|_\infty = \left\| \begin{array}{c} W_s S \\ W_{ks} K S \end{array} \right\|_\infty .$$

I guess by now you are not surprised to hear that there are several methods available to synthesize $H_\infty$ controllers. Typically you would use `hinfsyn`, `hinfric` or `hinflmi` which all have P in the System (mu-tools) representation as an input. In RCT v3.0.1, there is the function `mixsyn` with $G$, $W_s$, $W_{ks}$ (and $W_t$, a weight for the complementary sensitivity function) as inputs, that is, you do not need the generalized plant $P$ at all. The main difference between the methods is whether they use Riccati equations and gamma-iteration or linear matrix equalities to solve the optimization problem. The LMI approach does not require all of the technical assumptions needed when using Riccati equation based solvers.

There are a variety of other commands like `ncfsyn` and `loopsyn` (for $H_\infty$ loop shaping of the open loop transfer function $L = GK$), `hinfmix` and `msfsyn` (multi-objective). Check out the manual.

# 6 Analysing the results

After the controller has been synthesized, it is time to analyse the results. This can be done using Control System Toolbox commands like `lsim`, `step` (step response), `bode` (bode plot), `sigma` (singular value plot) and `freqresp` (frequency response) on typical transfer matrices like $S$, $KS$, $T$, $K$ and $GK$. Similar functions in mu-tools are `trsp` (time response), `frsp` (frequency response), `vsvd` (singular values) and `vplot`.

# 7 Conclusions

As you have seen, there are many options. To avoid going from one representation to another and back again, I prefer to use functions found in mu-tools and RCT as much as possible. If you know that there exists a function in the Control System Toolbox, chances are high you will find the same function in mu-tools, only with a sligthly different name. If you know what you want to do but cannot remember the command, the functions by category part of the matlab manual is a good reference.

Hopefully this short introduction to Matlab and $H_\infty$ will make it a little easier for you to synthesize your first $H_\infty$ controller, good luck!