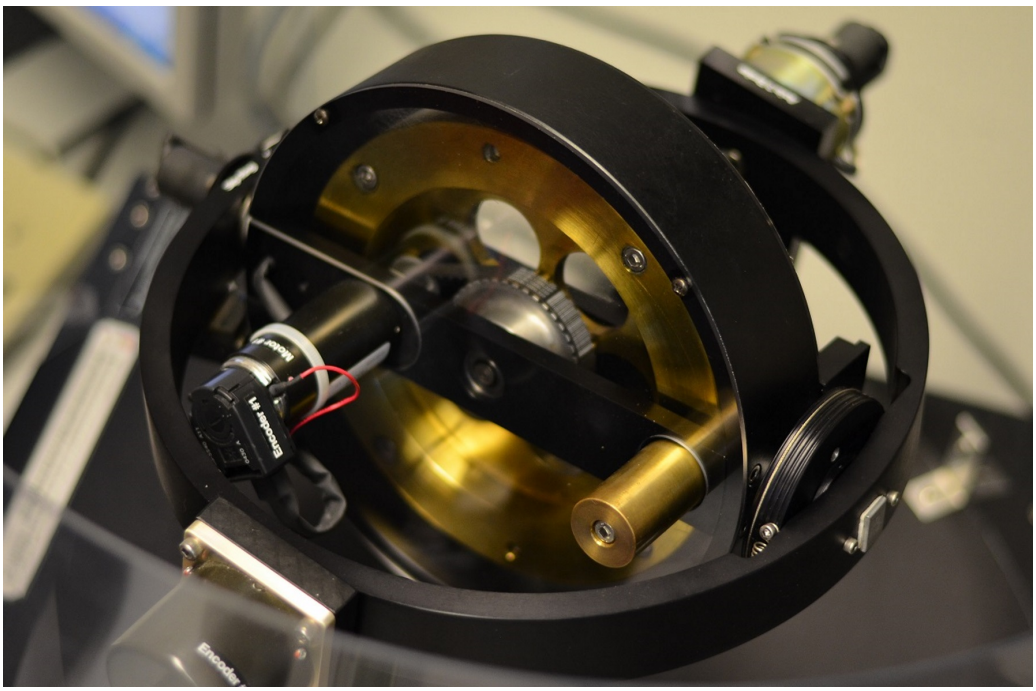
TUHH
Technische Universität Hamburg

ICS
Institute of Control Systems

Control Lab
ATC1

Gain-Scheduled Control of a Gyroscope



12th November, 2020

Room: Online | N-1.077

Winter Semester 2020/21

Contents

About this Document

This document is intended to help you with the design task *ATC1—Gain-Scheduled Control of a Gyroscope* of the *Control Lab* practical course. It is written mainly in the style of a tutorial and should provide you with all the necessary tools and Matlab commands to solve the task. There are several Matlab files that you need to modify and

execute in order to develop your own design. You are also encouraged to write your own code!

atc1_design.m This file deals with control design and linear evaluation. It is meant to provide a structure very similar to the tutorial given in this document.

! You need to complete the code on your own and submit the file.

atc1_GyroSimulation.slx This file contains the nonlinear simulation model.

! You may need to modify the block diagram to suite your own design.

atc1_simulation.m This file deals with evaluation in nonlinear simulation.

! You need to submit a published version of this file no later than one week before the scheduled date for the experiment.

There are also several other auxiliary files provided. The code is guaranteed to work with Matlab 2016b 64bit, other versions might not be supported. You can get the latest Matlab version from http://www.tuhh.de/rzt/usc/matlab/index.html or use the pool computers.

In addition to these files, we will provide you with the synthesis tools that you will need in order to design your gain scheduled controller.

In this document, you will encounter blocks that indicate Matlab code:

```
[MATLAB COMMANDS]=USEFUL(TOOLS)
```

These are meant to get you started. You can (and should) use the `help` command within Matlab to find out more about a particular command.

Another thing that you will encounter are preparation tasks:

Preparation: Think about the benefits of gain-scheduling when compared to a single robust controller.

These are meant to prepare you for the question session that will take place prior to conducting the experiment.

Task

Design an LPV controller for the control moment gyroscope and evaluate it in nonlinear simulation. Attach all necessary MATLAB and SIMULINK files that were provided to you

no later than one week before the experiment via email to the responsible Tutor and Supervisor. You will get an email when your preparation is not sufficient to pass the Lab and will get time to revise your design.

Checklist

- ☐ I read this whole document carefully.
- ☐ I did all preparation tasks and can explain them.
- ☐ I completed `atc1_design.m'.
- ☐ I completed `atc1_simulation.m'.
- ☐ I submitted all MATLAB files.

1 PLANT

A control moment gyroscope is a spinning rotor (flywheel) suspended in two motorized gimbal mountings and modeled as a four-degrees-of-freedom multibody system. The motorized mountings can tilt the flywheel's angular momentum, which causes a gyroscopic torque. A kinematic model is shown in Figure 1. Each body is linked to the previous body by a rotational joint perpendicular to the last joint axis.

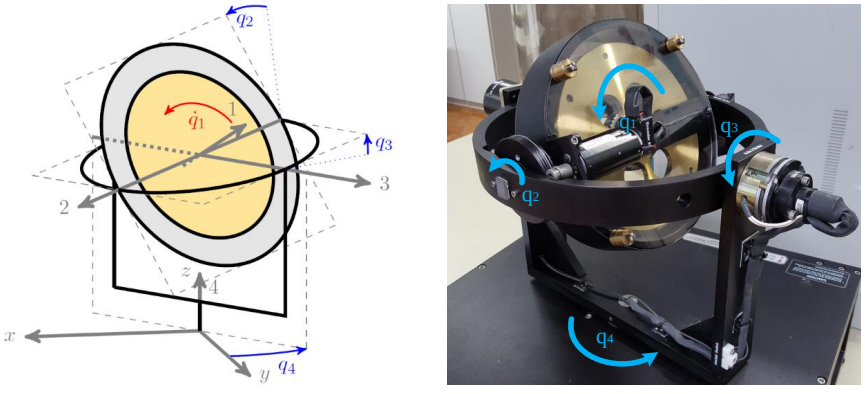


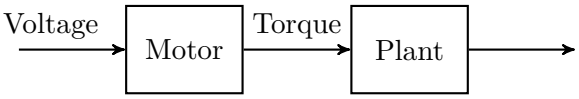
Figure 1: Kinematics of the control moment gyroscope

The nonlinear equation of motion which can be derived from mechanic principals, e. g., by using the Newton-Euler or Lagrange formalism, is

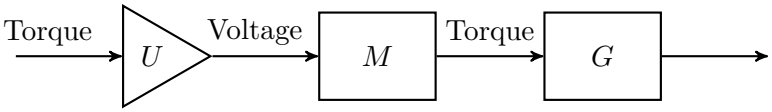
$$M(q)\,\ddot{q} + k(q,\dot{q}) = f(\dot{q}) + \begin{bmatrix} I \\ 0 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} ,$$

with the generalized coordinates $q = [q_1\ q_2\ q_3\ q_4]^T$, the generalized inertia M, the vector of generalized non-dissipative forces k and the vector of generalized dissipative forces f. The inputs T_1 and T_2 of the system represent torques, which are applied by electric motors at axis 1 and axis 2. The controlled outputs are the unactuated angles q_3 and q_4.

If we consider the actual physical plant, we notice that the torques are generated by motors, which themselves take voltages as inputs. In order to simplify the design, we neglect any dynamics of the motor and assume that it is just a constant gain. We can then add an inverse model of the motor in order to recover torques as control variables. Figure 2 illustrates the concept, where M denotes the motor model and U is the inverse model, i. e., $M\,U \approx I$.



(a) Actual physical plant with actuator



(b) Plant with actuator and inverse actuator model

Figure 2: Actuator model

Prep. 1.1: Is this a valid assumption? What exactly are we assuming by neglecting actuator dynamics?

1.1 Linearization and LPV Modeling

The system is nonlinear, but it can be approximated by a Jacobian linearization about an operation point given by a fixed flywheel rotation speed $\dot{q}_{1,0}$ and fixed angular positions

$q_{2,0}$ and $q_{3,0}$. This yields a collection of linear time invariant (LTI) state space model

$$\begin{aligned} \dot{x} &= A(\dot{q}_{1,0}, q_{2,0}, q_{3,0})\, x + B(\dot{q}_{1,0}, q_{2,0}, q_{3,0})\, u\ , \\ y &= C(\dot{q}_{1,0}, q_{2,0}, q_{3,0})\, x\ , \end{aligned} \tag{1}$$

with state vector $x = [q_3\ q_4\ \dot{q}_2\ \dot{q}_3\ \dot{q}_4]^T$, input $u = [T_1\ T_2]^T$ and output $y = [q_3\ q_4]^T$. We can collect the continuum of all admissible operating points into the scheduling vector $\rho = [\dot{q}_{1,0}\ q_{2,0}\ q_{3,0}]^T$. This is *not* an equivalent representation of the nonlinear system, because we are neglecting first and second order derivatives of $\dot{q}_{1,0}$, $q_{2,0}$, and $q_{3,0}$ that would result from a true "linearization about a time-varying operating point". What we have here is rather a continuous parameterization of a family of Jacobi linearizations, i. e., for a given value of ρ, the LPV model coincides with the LTI model obtained from linearization at that operating point. Since we use states of the system to define this operating point, we are generating a quasi-LPV model.

Prep. 1.2: Make sure you understand why this is not an equivalent representation of the nonlinear plant and hence why we have no a priori guarantees that the controller works on the nonlinear plant. Compare the approach to what is discussed in Expl. 1.1 of the *Advanced Topics in Control* lecture notes.

We will use the plant, scheduled by $\rho = [\dot{q}_1\ q_2\ q_3]^T$, as shown in Figure 3 with

$$y = \begin{bmatrix} q_3 \\ q_4 \end{bmatrix}\ , \qquad u = \begin{bmatrix} T_1 \\ T_2 \end{bmatrix}\ , \qquad \text{and further} \qquad r = \begin{bmatrix} q_{3,\text{ref}} \\ q_{4,\text{ref}} \end{bmatrix} \qquad \text{as a reference signal.}$$

for design purposes.

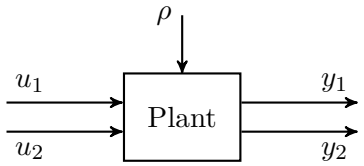
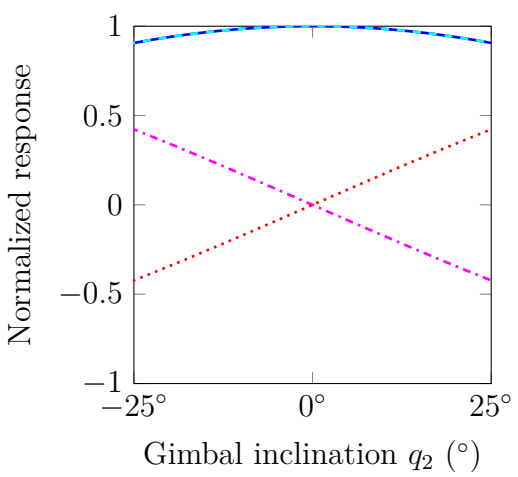
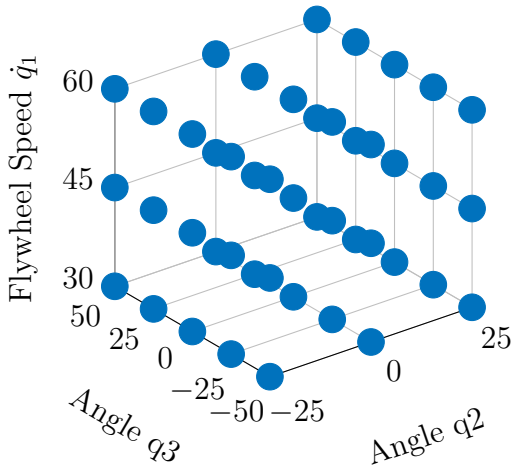


Figure 3: Inputs and outputs of the model

The effects of the control inputs on the controlled outputs depend strongly on the angle q_2 and thus the current operating point. For $q_2 = 0$ the system is decoupled, which means that the input T_1 has an exclusive effect on q_3 and T_2 has an exclusive effect on q_4.



(a) Normalized input response gain for different values of q_2 $T_1 \to q_3$ and $T_1 \to q_4$; $T_2 \to q_3$ and $T_2 \to q_4$



(b) Grid representation of the admissible parameter space

This consideration is quite hypothetical, since applying a torque T_2 also changes q_2. Thus, in practice $q_2 \neq 0$ and the inputs have a combined effect on the outputs. This is termed *cross coupling*. The effect of cross coupling dependent on the angle q_2 is illustrated in Figure 4(a), where it can be seen that not only the magnitude but also the directions of cross couplings depend on the sign of q_2.

Prep. 1.3: What do you expect in terms of cross-couplings when we design a controller for a single operating point and apply it to the real plant?

In order to use the synthesis techniques for gain-scheduled controllers that were discussed in class, we first need to define a grid representation of the parameter-space ($\mathcal{P}$) and the rate bounds ($\mathcal{V}$). In order to do this, we can use the following Matlab commands:

```
q1dot   = pgrid('q1dot',30:15:60,[-10 10]);
q2      = pgrid('q2',[-25*pi/180, 0, 25*pi/180], [-2 2]);
q3      = pgrid('q3',[-50*pi/180:25*pi/180:50*pi/180], [-2 2]);
dom     = rgrid(q1dot,q2,q3);
```

To load the linearized model into the Matlab workspace you can use the provided function

```
Gl = linearize_gyro(q1dot.griddata,q2.griddata,q3.griddata);
```

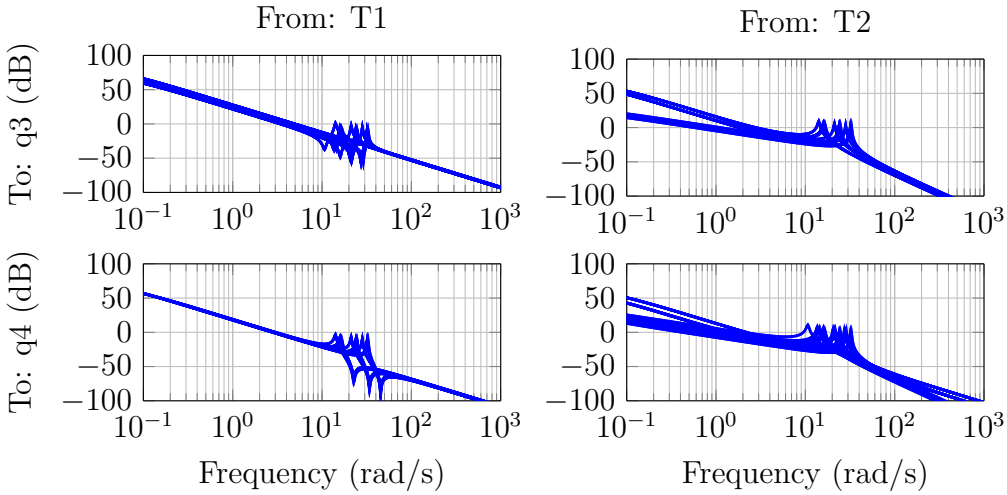
which contains a parameterization of the linearization at different operating points. We can then form a parameter-dependent state space object by calling

```
Gp = pss(Gl,dom)
```

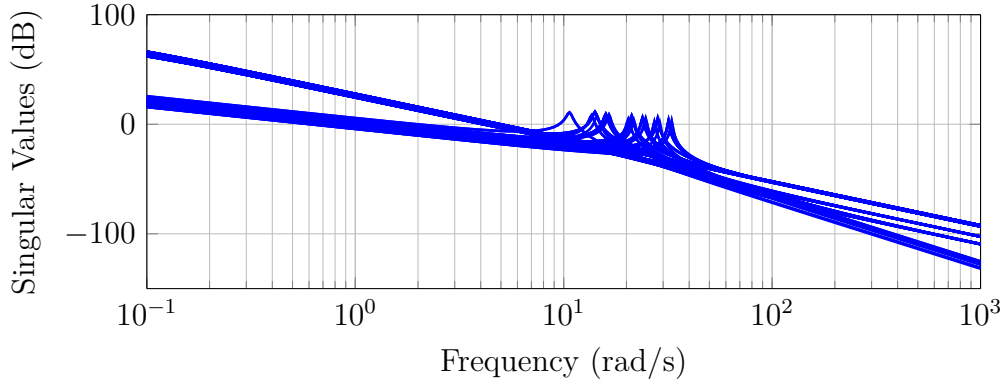
This object can be used very similar to the `ss` objects that you are familiar with. We can for example use

```
bodemag(Gp)
sigma(Gp)
```

to bring up a bode magnitude plot and a sigma plot of the plant as shown in Figure 4.



(c) Bode magnitude plot of the Plant



(d) Sigma plot of the Plant

Figure 4: Frequency response of the plant

Prep. 1.4: From Figure 4, are the parameter variations large? What appear to be the most dominant effects?

Note that we can access a variety of different properties of `pss` objects, i. e.,

```
properties(Gp)     % shows available properties
Gp.Data            % returns underlying array of ss objects
Gp.Parameter       % returns scheduling parameters of the model
```

Prep. 1.5: Which representation of the frequency response, Bode plot or sigma plot, is more useful for design purposes? Why?

1.2 Initial Scaling

Scalings are of dire importance when working with frequency domain synthesis techniques. The following (very simple) scaling just normalizes the input and output values of the plant, such that the magnitude one corresponds both to the maximum expected change in the reference signal $D_{r,\max}$ and the maximum actuator capacity $D_{u,\max}$, i. e.

$$u = D_{u,\max}^{-1}\, u_{\text{real}} \tag{2}$$

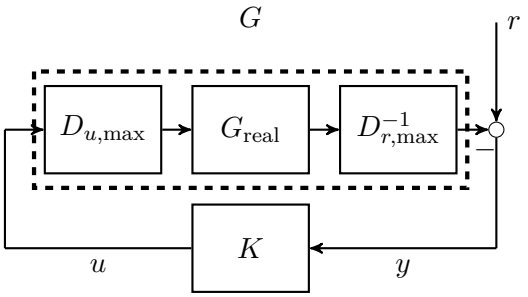
$$y = D_{r,\max}^{-1}\, y_{\text{real}} \tag{3}$$

Just think of a rescaled input vector $B_s = B\, u_{\max}$ and a rescaled output vector $C_s = \frac{1}{r_{max}} C$ for the SISO case and keep in mind that since we are dealing with MIMO systems, we need to use diagonal matrices $D_{u,\max}$ and $D_{r,\max}$ instead. The usefulness of this scaling becomes apparent when we realize that a unit step reference now corresponds to the maximum input that we expect and that the available control signal is now also 1. Thus, if a unit step results in a control signal less than 1, the actuators are likely not to saturate. In terms of frequency domain indicators, this means that we can look at the transfer function $K\,S$ and try to achieve $\|K\,S\|_\infty < 1$. Once a controller for the synthesis model is designed, the scalings have to be reversed when the controller is applied to the original model:

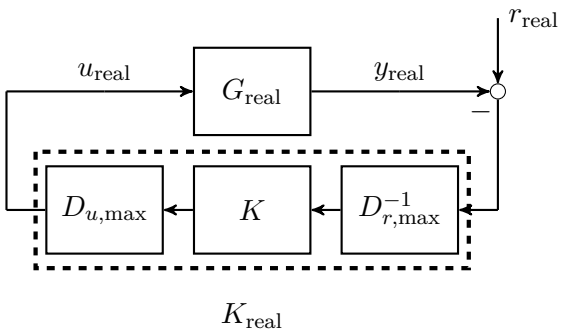
$$u = K\, y \tag{4}$$

$$u_{real} = \underbrace{D_{u,\max}\, K\, D_{r,\max}^{-1}}_{K_{\text{real}}}\, y_{\text{real}} \tag{5}$$

Figure 5 illustrates the procedure.



(a) Scaling the actual plant to obtain a synthesis model



(b) Scaling the controller for implementation on the actual plant

Figure 5: Scaling for controller design based on frequency responses

For the Gyroscope, reasonable assumptions on the largest allowed input signals are

$$T_{1,\max} = 0.666\,\text{Nm}, \qquad T_{2,\max} = 2.44\,\text{Nm}$$

due to the different gearing mechanisms of the two motors. The outputs can be scaled for instance with

$$q_{3,\max} = 45\frac{\pi}{180}, \qquad q_{4,\max} = 90\,\frac{\pi}{180}\ .$$

2 CONTROLLER DESIGN—STATE FEEDBACK

We will consider two different control design tasks in this lab: an augmented state-feedback controller and an output feedback controller, both self-scheduled within the

LPV framework. We will make use of the mixed sensitivity design framework and express design specifications in terms of a generalized plant. If you feel you need to review fundamentals about mixed sensitivity design, see chapter 17 and 18 of the *Optimal and Robust Control* lecture notes and the *Control Lab—ORC2* documentation.

2.1 S/KS Design

We decide to us an S/KS weighting scheme for state feedback design. The sensitivity S defines our nominal performance while shaping KS provides some basic robustness and limits control effort. The transfer functions S and KS are weighted by shaping filters W_S and W_K, which act as an upper bound on the induced $\mathcal{L}_2$-norm. These filters are part of the generalized plant.

An important question in LPV control is whether we want to use parameter-dependent or constant weighting filters. The former allow us to specify different performance for different operating points, while the latter seek to generate a closed-loop system that behaves similarly irrespective of the operating condition. The choice strongly depends on the specific application. For controllers with humans in the loop, homogeneous behavior is usually desired to facilitate handling. Maximum achievable performance at every operating condition, on the other hand, usually requires to explicitly take into account variations.

Prep. 2.1: Make sure you understand the different philosophy behind constant and parameter-varying weights in LPV control.

In order to specify weights, we could use the parameterization from the ORC lecture (possibly with `pgrid` objects for parameter-dependent weights) to define weighting filters manually. A built-in command of the Robust Control Toolbox that we can use instead is

```
W_1 = makeweight(dcgain1, bandwidth1, feedthroughgain1)
W_2 = makeweight(dcgain2, bandwidth2, feedthroughgain2);
W   = mdiag(W_1,W_2);
```

where `dcgain` specifies the low frequency gain, `bandwidth` specifies the frequency at which the gain is one and `feedthroughgain` specifies the high frequency gain.

We need a generalized plant of the form

$$\left[\begin{array}{c} \dot{x} \\ \hline z_1 \\ z_2 \end{array}\right] = \left[\begin{array}{c|cc} A(p) & B_w(p) & B_u(p) \\ \hline C_1(p) & 0 & 0 \\ C_2(p) & 0 & I \end{array}\right] \left[\begin{array}{c} x \\ \hline w \\ u \end{array}\right] \tag{6}$$

with external input w, controller output u and performance output z. Hereby, we assume that all states of the generalized plant are available for feedback, so we do not explicitly need to add an output for the controller.

For our synthesis tools to work, we need to first assemble the generalized plant that includes the desired performance inputs and outputs. The generalized plant for the S/KS formulation is depicted in Figure 6. In order to assemble it, use the `sysic` command.

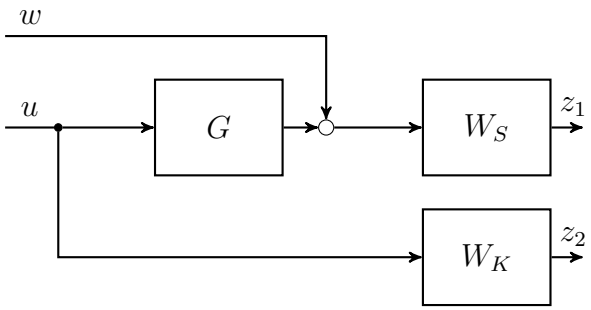


Figure 6: Open-loop generalized Plant P for the S/KS mixed sensitivity design

You should look up the help in Matlab, but to get you started, consider the following code

```
systemnames = 'sys1 sys2 sys3';
inputvar = '[w{2}; u{2}]';
outputvar = '[sys1; sys2; sys3+w]';
input_to_sys1 = '[u]';
input_to_sys2 = '[sys3+w]';
input_to_sys3 = '[u]';
P = sysic
```

2.2 Synthesis

Stability and the induced $\mathcal{L}_2$-norm constraints can be expressed as linear matrix inequalities (LMIs), see Theorem 3.1 of the *Advanced Topics in Control* lecture notes. We already defined the gridded parameter space and the rate bounds, so all that is left, is to define basis functions that represent the functional dependence of the candidate Lyapunov matrix P on the scheduling parameters, i. e.,

$$P(\rho) = \sum f_i(\rho)\, P_i \,. \tag{7}$$

Prep. 2.2: Why do we need to assign basis functions? What are the decision variables when solving the optimization problem?

We can do this in Matlab by using

```
Pbasis = [];
Pbasis(1) = basis(1,0);
```

The first argument of `basis` is the function and the second argument is the partial derivative. Thus, the example above defines a parameter-independent Lyapunov matrix $P = P_1$ and the partial derivative of this constant is zero. If we want to add parameter-dependent terms, the `pgrid` objects can be used:

```
Pbasis(2) = basis(q2,1);
```

This adds a second term that is linear in q_2 (with a partial derivative of one with respect to this one parameter). The Lyapunov matrix is thus $P = P_1 + P_2\, q_2$. Of course, we can add arbitrary complex expressions such as

```
Pbasis(3) = basis((q2-4)^2/3*sin(q1dot),'q2',...
    (sin(q1dot)*(2*q2 - 8))/3,'q1dot',(cos(q1dot)*(q2 - 4)^2)/3));
```

With this, the Lyapunov matrix becomes $P = P_1 + P_2\, q_2 + P_3\, \frac{(q_2-4)^2}{3}\, \sin(\dot{q}_1)$ Obviously, we need to pay close attention to defining the partial derivatives correctly. How to chose a "good" set of basis functions is to a large extend still not clear, see also exercise 3.2 in the *Advanced Topics in Control* lecture notes.

Having defined both the generalized plant and the basis functions, we can solve for a state feedback gain:

```
[F,gam,INFO] = lpvsfsyn(P,ncont,Pbasis);
```

Prep. 2.3: Experiment with different choices of basis functions. How do they affect synthesis complexity and achievable performance?

2.3 Augmented Controller

By employing state feedback synthesis, we assume that all states of the *generalized* plant are available for feedback. So in addition to all states of the plant, the controller also requires access to the weighting filters' states. Thus, if we seek to implement the controller, these filters have to be implemented as part of the controller.

This step is very similar to the integral augmentation for state feedback control that was introduced in *Control Systems Theory and Design*, see chapter 4 and 5 of the *CSTD* lecture notes. The similarity becomes even more obvious if W_S is selected as a $\frac{1}{s}$ and W_K is selected to be static. In this case, all we do is adding the integrated error. Figure 7 shows the resulting controller.

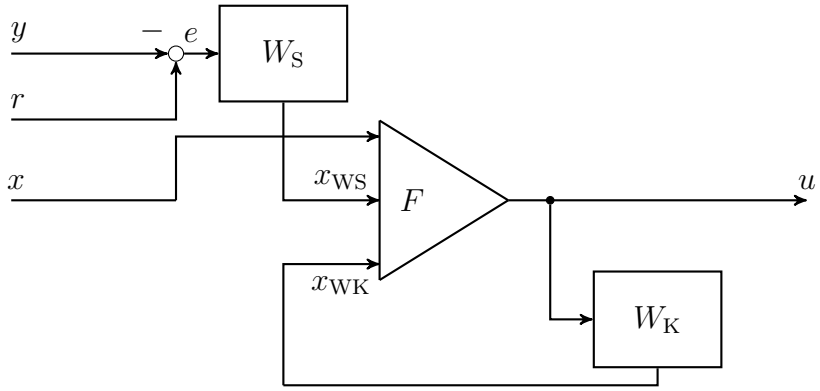


Figure 7: Augmented state feedback controller

Prep. 2.4: Make sure you understand the dynamic augmentation and why it is necessary to obtain a "good" controller".

Prep. 2.5: Can we simply implement the filters W_S and W_K? Does the controller depend on a specific state space realization of these filters?

Prep. 2.6: S/KS problems are known to lead to pole-zero cancellations for output feedback problems (see, e. g., Control Lab ORC2). Why is this not a problem in the present case?

2.4 Design Task

Design a gain-scheduled state feedback controller to achieve the following design specifications:

- a 5% settling time of less than 4 seconds on both channels at every grid point
- a steady state error of less than 1% on both channels at every grid point
- that the control input magnitude never exceeds 1 at any time
- a controller with a fastest pole < 2000 rad/s when evulated at a grid point
- that disturbances are compensated after 5 seconds on both channels at every grid point
- that disturbances lead to outputs of less than 3 in magnitude at every grid point

Note that we consider a larger operating range than in the *ORC2* lab and that hence the specifications appear to be reduced. For example, we asked for a settling time of 2 seconds in that experiment, but considered reference steps of 45°, where here, we ask for 90° in 4 seconds. Thus, in fact we have the same speed-of-response specification for the nonlinear plant.

Prep. 2.7: Why do we have to adjust the specifications for (quasi-) linear design?

Also design LTI controllers with the same methodology and compare them to their gain-scheduled counterparts. In order to use the same framework, you can simply extract a single operating point from the plant, i. e.

```
Glti = lpvsplit(G,'q1dot',45,'q2',0,'q3',0);
```

and use this pss object with the exact same tools as before.

You will most likely need to decrease your desired bandwidth considerably in order to achieve a similar level of performance since, naturally, the specifications are much easier to achieve for a single operating point than for the whole parameter space (compare also the *Control Lab—ORC2* documentation and exercise 3.2 of the *Advanced Topics in Control* lecture notes). Therefore, splitting the generalized plant is not advised. You should rather construct a new generalized plant `Plti`. Simply omit the basis function argument to use a parameter-independent Lyapunov matrix for synthesis:

```
[Flti,gam,INFO] = lpvsfsyn(Plti,ncont);
```

3 SYNTHESIS MACHINERY—A CLOSER LOOK

The state feedback problem is solved with the elimination approach introduced by Wu, see Theorem 3.1 of the *Advanced Topics in Control* lecture notes. The main reasons why we use this approach is that the synthesis complexity of the elimination approach is smaller compared to, let's say, a linearizing change of variables. With synthesis complexity actually still being a limiting factor in the control design, this reduced complexity can be very beneficial.

The elimination, however, requires the generalized plant to be in a specific form. When setting up a problem, the generalized plant will usually not look like the one given in the *Advanced Topics in Control* lecture notes. Nevertheless, as this chapter shows, we can usually transform the problem such that it suits the special form required for elimination.

Note that the tools that we use perform these steps by themselves, so this chapter is really just meant to provide you with some additional insight into how the synthesis machinery works and is not essential for performing the design task.

3.1 Lifting Assumptions on the Generalized Plant

Given the generalized plant

$$\left[\begin{array}{c} \dot{x} \\ \hline z_1 \\ z_2 \end{array}\right] = \left[\begin{array}{c|cc} A(p) & B_w(p) & B_u(p) \\ \hline C_1(p) & 0 & 0 \\ C_2(p) & 0 & I \end{array}\right] \left[\begin{array}{c} x \\ \hline w \\ u \end{array}\right] \tag{8}$$

we can obtain a state feedback controller by finding $P(\rho)$ such that $\forall(\rho) \in \mathcal{P}, \dot{\rho} \in \mathcal{V}$

$$P > 0 \tag{9a}$$

$$\begin{bmatrix} P\,(A - B_u\,C_2)^T + (A - B_u\,C_2)\,P - B_u\,B_u^T - \sum_{i=1}^{n_\rho} \frac{\partial P}{\partial p_i}\,\dot{\rho} & P\,C_1^T & \frac{1}{\gamma}\,B_w \\ C_1\,P & -I & 0 \\ \frac{1}{\gamma}\,B_w^T & 0 & -I \end{bmatrix} < 0 \tag{9b}$$

where the state feedback gain is given by the explicit formula

$$F = -B_u^T\,P^{-1} - C_2 \tag{9c}$$

Here, two assumptions were made on the generalized plant

$$\left[\begin{array}{c}\dot{x}\\\hline z\end{array}\right] = \left[\begin{array}{c|cc}A(p) & B_w(p) & B_u(p)\\\hline C(p) & D_{zw}(p) & D_{zu}(p)\end{array}\right] \left[\begin{array}{c}x\\\hline w\\u\end{array}\right] .$$

These are: $D_{zw} = 0$ and $D_{zu} = \left[\begin{smallmatrix}0\\I\end{smallmatrix}\right]$. While it is shown in the lecture notes that this can be easily achieved for the S/KS design, it is still rewarding from a practical point of view to take a closer look at how these restrictions can be lifted. For example, $D_{zw} \neq 0$ allows us to specify the shape of our sensitivity function with a bi-proper weight, which either gives more control over the peaks or allows to use a lower-order filter. We will see that assumption $A2$ in the lecture notes, i. e. D_{zu} has full column rank $\forall p$, is all it takes (in addition of course to controllability of the pair (A, B_u)).

Let's first take a look at how to bring D_{zu} into the required form and to that end still assume that we have no feedthrough from w to z. The generalized plant is thus

$$\left[\begin{array}{c}\dot{x}\\\hline z_1\\z_2\end{array}\right] = \left[\begin{array}{c|cc}A(p) & B_w(p) & B_u(p)\\\hline C_1(p) & 0 & D_{z1u}(p)\\C_2(p) & 0 & D_{z2u}(p)\end{array}\right] \left[\begin{array}{c}x\\\hline w\\u\end{array}\right] \tag{10}$$

With $\left[\begin{smallmatrix}D_{z1u}\\D_{z2u}\end{smallmatrix}\right]$ by assumption having full column rank, we can define

$$R_u = \left(\begin{bmatrix}D_{z1u}\\D_{z2u}\end{bmatrix}^T \begin{bmatrix}D_{z1u}\\D_{z2u}\end{bmatrix}\right)^{1/2} . \tag{11}$$

The exponent $1/2$ here denotes the matrix square root which can be defined to describe the decomposition of a matrix M into two factors S such that $M = S\,S^T$.

We can now define

$$Q_1 = \begin{bmatrix}D_{z1u}\\D_{z2u}\end{bmatrix} R_u^{-1} \tag{12}$$

since R_u is invertible. We further note that Q_1 is an orthogonal matrix since $Q_1\,Q_1^T = I$. Second, we can find an orthogonal basis Q_2 for the Nullspace of Q_1^T by performing a

(complete) QR-Decomposition of Q_1:

$$\begin{bmatrix}\tilde{Q}_2 & Q_2\end{bmatrix} \begin{bmatrix}\tilde{R} \\ 0\end{bmatrix} = Q_1 \tag{13}$$

With this, we can then apply an input/output transformation to the generalized plant

$$P_{\text{norm}} = \begin{bmatrix}Q_2^T \\ Q_1^T\end{bmatrix} P \begin{bmatrix}I & 0 \\ 0 & R_u^{-1}\end{bmatrix} \tag{14}$$

which in terms of the state space representation corresponds to

$$\left[\begin{array}{c}\dot{x} \\ \hline z_1 \\ z_2\end{array}\right] = \left[\begin{array}{c|cc} A & B_w & B_u\, R_u^{-1} \\ \hline Q_2^T \begin{bmatrix}C_1 \\ C_2\end{bmatrix} & 0 & Q_2^T\, D_{z1u}\, R_u^{-1} \\ Q_1^T \begin{bmatrix}C_1 \\ C_2\end{bmatrix} & 0 & Q_1^T\, D_{z2u}\, R_u^{-1}\end{array}\right] \left[\begin{array}{c}x \\ \hline w \\ u\end{array}\right] \tag{15}$$

From this representation, we can infer two things: First, the norm of the performance outputs remains unchanged since Q_1 and Q_2 are orthogonal, i, e.

$$\left\| \begin{bmatrix}z_1 \\ z_2\end{bmatrix} \right\| = \left\| \begin{bmatrix}z_1 \\ z_2\end{bmatrix} \begin{bmatrix}Q_2^T \\ Q_1^T\end{bmatrix} \right\| \tag{16}$$

which means that we do not change the induced norm from w to z by this transformation and hence still solve the same control design problem. Second, from the definition of R_u in Equation (12), it follows

$$\begin{bmatrix}Q_2^T\, D_{z1u}\, R_u^{-1} \\ Q_1^T\, D_{z2u}\, R_u^{-1}\end{bmatrix} = \begin{bmatrix}Q_2^T \\ Q_1^T\end{bmatrix} Q_1 = \begin{bmatrix}0 \\ I\end{bmatrix} \tag{17}$$

which is exactly the normalization that we require in the theorem. We thus see that it is indeed possible to guarantee the special structure of the D_{zu} matrix.

Prep. 3.1: Make sure you understand the normalization procedure. It might be a good idea to code a small example for yourself, to see that it actually works.

Next, we will present the synthesis problem for the case $D_{zw} \neq 0$, i. e. for the generalized

plant

$$\left[\begin{array}{c} \dot{x} \\ \hline z_1 \\ z_2 \end{array}\right] = \left[\begin{array}{c|cc} A & B_w & B_u \\ \hline C_1 & D_{z1w} & 0 \\ C_2 & D_{z2w} & I \end{array}\right] \left[\begin{array}{c} x \\ \hline w \\ u \end{array}\right] . \tag{18}$$

The elimination-based synthesis problem can then be stated as the task to find $P(\rho)$ such that $\forall(\rho) \in \mathcal{P}, \dot{\rho} \in \mathcal{V}$

$$\begin{aligned} P &> 0 \\ \bar{\sigma}\left(\begin{bmatrix} D_{z1w} \\ D_{z2w} \end{bmatrix}\right) &< \gamma \\ \begin{bmatrix} P\,(A - B_u\,C_2)^T + (A - B_u\,C_2)\,P - B_u\,B_u^T - \sum_{i=1}^{n_\rho} \frac{\partial P}{\partial p_i}\,v_i & P\,C_1^T & \frac{1}{\gamma}\,\hat{B}_w \\ C_1\,P & -I & \frac{1}{\gamma}\hat{D}_{z1w} \\ \frac{1}{\gamma}\,\hat{B}_w^T & \frac{1}{\gamma}\hat{D}_{z1w}^T & -I \end{bmatrix} &< 0 \end{aligned}$$

where we can note a few subtle changes compared to the original formulation (9). First, there is a new condition $\bar{\sigma}\left(\left[\begin{smallmatrix} D_{z1w} \\ D_{z2w} \end{smallmatrix}\right]\right) < \gamma$, resulting from the projection $V_\perp\,R\,V < 0$ (see the *Advanced Topics in Control* lecture notes). This condition is intuitively clear, since the largest singular value of $\left[\begin{smallmatrix} D_{z1w} \\ D_{z2w} \end{smallmatrix}\right]$ determines the largest singular value of the performance channel at infinite frequency and hence is a lower bound on the achievable performance. We can further note that a term involving D_{z1w} appears in the off-diagonal entries of the bounded real lemma condition and finally that B_w is replaced with $\hat{B}_w = B_w - B_u\,D_{z2w}$. Also, the formula for the feedback gain looks now a little more complicated:

$$F = -\left(B_2^T + \frac{1}{\gamma^2}\,D_{z2w}\,\hat{B}_2^T - D_{z2w}\,D_{z1w}^T\,\left(\frac{1}{\gamma^2}\,D_{z1w}\,D_{z1w}^T - I\right)^{-1}\,\left(C_1\,P + D_{z1w}\,\hat{B}_w\right)\right)P^{-1} - C_2$$

We are not proving this result. You are welcome to prove it yourself by simply following the proof given in the lecture notes while keeping track of a non-zero D_{zw}. The purpose of showing you this result, however, is to show you that we can also handle generalized plants with arbitrary D_{zw}-matrices since this is what is implemented in the tools that we use for control design.

Prep. 3.2: Verify that for the case $D_{zw} = 0$, this version of the theorem still yields the familiar results.

3.2 Suboptimal Synthesis

Just as with regular $\mathcal{H}_\infty$ control, it is usually desirable to use a suboptimal, rather than a truly optimal synthesis (compare the *Control Lab—ORC2 Manual*). Since LPV synthesis is even more prone to numerical issues than the $\mathcal{H}_\infty$ synthesis, this is a very important step. One possibility for such a suboptimal synthesis is given by the following three step procedure.

1. Semidefinite Program—optimal synthesis
   - $\min \gamma$ such that Bounded Real Lemma (BRL) condition holds
   - Relax γ by a factor of $\sqrt{1.1}$ and fix it
2. Semidefinite Program—upper bound on spectral radius
   - $\min \bar{\lambda}$ such that $P < \bar{\lambda} I$ and BRL condition holds
   - Relax γ again by a factor of $\sqrt{1.1}$ and fix it
   - Relax $\bar{\lambda}$ by a factor of 1.1 and fix it
3. Semidefinite Program—lower bound on spectral radius
   - $\max \underline{\lambda}$ such that $\underline{\lambda} I < P < \bar{\lambda} I$ and BRL condition holds

The current Beta Version of the `LPVTools` has no suboptimal state feedback synthesis implemented. The modified synthesis routine

```
[F,gam,INFO] = lpvsfsynth(P, ncont, Pbasis);
```

does implement the three step procedure described above.

Other common possibilities are the inclusion of something similar to pole region constraints or the use of penalty terms in the minimization object, for example the trace of the Lyapunov matrices (which again is an upper bound for the spectral radius…).

Prep. 3.3: Compare the results that you get from the optimal synthesis using `lpvsfsyn` and the suboptimal synthesis using `lpvsfsynth`. Depending on the problem, the difference might actually be small. It is however possible, that the optimal synthesis leads to very large entries in the state feedback matrix. Why is this a consequence of a nearly singular Lyapunov matrix P?

4 CONTROLLER DESIGN—OUTPUT FEEDBACK

Often, not all states of a system are available for feedback. Even if they can be estimated, as in the present case, it might not be desirable to use state feedback control. State feedback assumes *perfect* knowledge of all states, that is, the effects of measurement noise, uncertainty, etc. are inherently ignored. Still, as you can see for yourself in this lab, such state feedback controllers can perform very well in practice. The "correct" way of addressing feedback problems is nevertheless to consider output feedback with possibly imperfect measurements. In this chapter, we thus design a gain-scheduled output feedback controller.

Prep. 4.1: What is the main difference between state and output feedback? Think of observer-based state feedback and the effect that the observer has on the overall dynamic controller.

4.1 Four-Block Design with Velocity Feedback

In what is called a four-block design, we seek to minimize the weighted norms of the transfer functions S, KS, SG and T_i, see Exercise 17.2 in the *Optimal and Robust Control* lecture notes and the *Control Lab—ORC2* documentation. In the *Control Lab—ORC2* experiment, it becomes apparent that while disturbance rejection and robustness are generally good with this design approach, simultaneously achieving satisfactory tracking performance requires the use of what is called a two-degrees-of-freedom controller. Such a controller receives the available measurements and the reference as two independent signals (instead of a single error signal) and consequently decouples the tracking problem from other requirements such as disturbance rejection. We consider the same modification of the four-block problem that was used in the *Control Lab—ORC2*.

The generalized plant for an output feedback problem is of the form

$$\left[\begin{array}{c} \dot{x} \\ \hline z \\ v \end{array}\right] = \left[\begin{array}{c|cc} A(p) & B_w(p) & B_u(p) \\ \hline C_z(p) & D_{zw}(p) & D_{zu}(p) \\ C_v(p) & D_{vw}(p) & D_{vu}(p) \end{array}\right] \left[\begin{array}{c} x \\ \hline w \\ u \end{array}\right] \tag{20}$$

where v now denotes the information provided to the controller.

The generalized plant that we are using is depicted in Figure 8 and has three different performance inputs: r is the reference signal for y, d_i is an input disturbance acting on

the plant and d_o can be thought of as measurement noise or disturbances on all available feedback signals. The performance outputs z_1 and z_2 reflect the design objectives and the available outputs are v_1 and v_2. The signal v_1 is used for feedback while v_2 is actually the reference signal and hence used in a feedforward fashion.

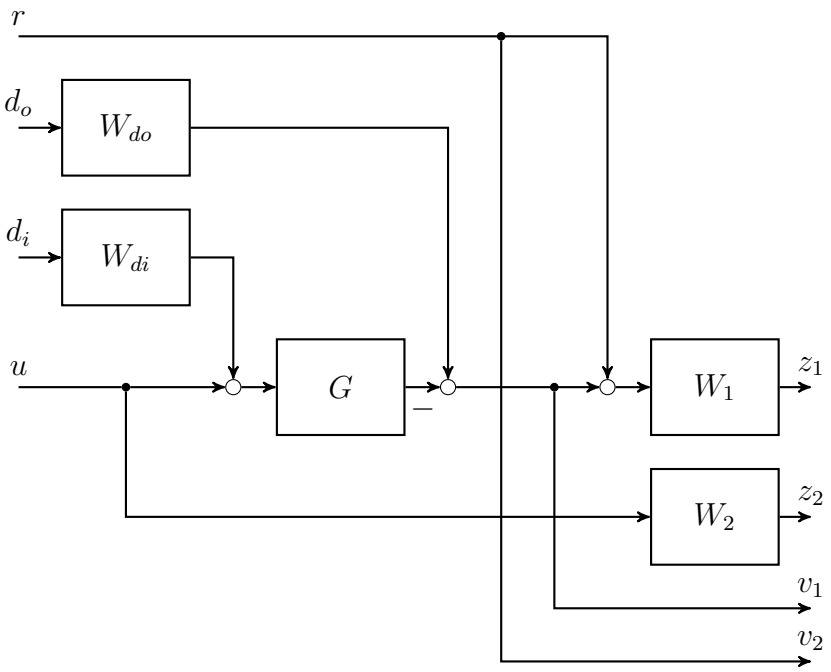


Figure 8: Open-loop generalized Plant P for the four block mixed sensitivity design with two degrees of freedom

Selecting the weights for this problem is not trivial and it is important to understand what we actually want to achieve. From the synthesis, we get a two-degrees-of-freedom controller, i. e. a controller with two independent inputs:

$$u = \begin{bmatrix} K_y & K_r \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} K_y & K_r \end{bmatrix} \begin{bmatrix} d_o - y \\ r \end{bmatrix} . \tag{21}$$

We can separate these two independent controllers and draw the resulting closed-loop system as in Figure 9. The transfer functions that appear in our design problem can

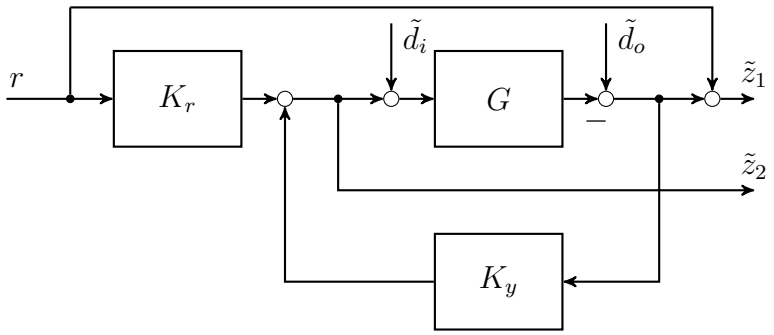


Figure 9: Closed-loop with two-degrees-of-freedom controller

thus seen to be

$$\begin{bmatrix} \tilde{z}_1 \\ \tilde{z}_2 \end{bmatrix} = \begin{bmatrix} S\,G & S & I - S\,G\,K_r \\ T_i & K_y\,S & S_i\,K_r \end{bmatrix} \begin{bmatrix} \tilde{d}_i \\ \tilde{d}_o \\ r \end{bmatrix} \tag{22}$$

The transfer functions from the first two inputs, i. e. d_i and d_o form the classical four block design problem. We can verify, that the transfer function from r to y is $S\,G\,K_r$ and hence the term $I - S\,G\,K_r$ in Equation (22) represents the tracking error $(r - y)$. What we are mainly interested in are disturbance rejection $(S\,G)$ and tracking $(I - S\,G\,K_r)$ for low frequencies and a roll-off in the controller ($K_y\,S$ and $S_i\,K_r$) for high frequencies. Thus, the standard choice of W_1 as a low pass and W_2 as a high pass make also sense with this setup. The trade-off between tracking, disturbance rejection and control effort is handled via the input weights W_{di} and W_{do}

Prep. 4.2: Understand the weighting structure and how to use it for tuning.

Prep. 4.3: Why is two-degree-of-freedom control helpful for tracking?

In addition to the measured angles given by y, we will also consider the angular velocities $\dot{q}_2$, $\dot{q}_3$ and $\dot{q}_4$ (i. e., $\dot{y}$) as available feedback signals. In practice, we still need to estimate these by implementing differentiation filters, which causes phase loss and amplifies measurement noise. We neglect the phase loss that is introduced by these filters in our model and rely on the robustness of the controller to deal with it. Note that contrary to the state feedback case, this decision is theoretically backed up, as we assume the measurements to be imperfect (with d_o acting as some disturbance). Experiments

show that with these extra measurements, the controller is able to increase damping much better than with just the angles. It is hard to rigorously justify this by theory, but in practice this "velocity feedback" is quite common. Make sure you understand that although we add additional outputs to the plant, our tracking objective is still just concerned with y, not $\dot{y}$. Thus, r is a vector with 2 entries while y is a vector with 5 entries. The error $y - r$ thus is not well defined unless we are more precise and write (in Matlab notation) $r - y(1:2)$. We can add the additional outputs to the plant using

```
Gv = pss(G.a,G.b,[G.c;zeros(3,2) eye(3)],[G.d;zeros(3,2)]);
```

4.2 Synthesis

The LMI conditions for output feedback are given by Theorem 3.2 of the *Advanced Topics in Control* lecture notes. Besides stabilizability of (A, B_u) and detectability of (A, C_v), we require that D_{zu} has full column rank and that D_{vw} has full row rank. If these conditions hold, techniques similar to those described in the previous section can be used to transform the generalized plant into

$$\left[\begin{array}{c} \dot{x} \\ \hline z_1 \\ z_2 \end{array}\right] = \left[\begin{array}{c|cc} A(p) & B_w(p) & B_u(p) \\ \hline C_z(p) & D_{zw}(p) & \left[\begin{smallmatrix} 0 \\ I \end{smallmatrix}\right] \\ C_v(p) & \left[\begin{smallmatrix} 0 & I \end{smallmatrix}\right] & 0 \end{array}\right] \left[\begin{array}{c} x \\ \hline w \\ u \end{array}\right] \tag{23}$$

These transformations, as well as balancing and scaling to reduce sensitivity to numerical issues are automatically performed within the LPVTools' `lpvsyn` command. Further, a suboptimal synthesis is invoked, basically using the first and third step of the procedure described in the previous section. The default suboptimality factor is 1.2, i. e., 20%.

Again we need to define basis functions, this time for the two candidate Lyapunov matrices X and Y. The controller is then obtained by the command

```
[K,gam,INFO] = lpvsyn(P,nmeas,ncont,Xbasis,Ybasis);
```

Depending on the dynamic order of the system, the number of basis functions for the Lyapunov matrices and especially the number of grid points, this synthesis may take very long (45 grid points with affine Lyapunov matrix take around 20 minutes for the gyroscope). Tuning the controller can thus become a really time-consuming task, and as

we all know: time is money. Thus, it is usually a better idea to use either parameter-independent Lyapunov matrices while trying to find weights or to simply perform a number of LTI synthesis steps with a common weighting structure. The main conceptual difference is the following: The first approach gives an upper bound during tuning, since the use of a common constant Lyapunov matrix is conservative. Our responses will usually look better when using the actual parameter-dependent synthesis once a satisfactory tuning was found. The second approach, on the other hand, gives a lower bound. We cannot exceed the performance of an LTI controller at a single grid point, even if we used infinitely many basis functions for the Lyapunov matrices. Our results will thus usually look worse with the LPV controller and we can add basis functions in order to try to make the deterioration small.

4.3 Controller Post-Processing

If a parameter-dependent Lyapunov matrix X is used, the controller also depends on scheduling rates. Often, this dependence can simply be ignored, although again, any theoretical guarantees are lost in doing so. For the current case, we would require $\ddot{q}_1$, $\dot{q}_2$ and $\dot{q}_3$ to be available online. It's your decision whether you want to implement the rate dependence with differentiation filters or ignore it. In order to neglect it, we can interpolate the controller for zero rates:

```
K0 = lpvelimiv(lpvinterp(K,'q1dotDot',0,'q2Dot',0,'q3Dot',0));
```

4.4 Design Task

Design a gain-scheduled output feedback controller to achieve the following design specifications:

- a 5% settling time of less than 4 seconds on both channels at every grid point
- a steady state error of less than 1% on both channels at every grid point
- that the control input magnitude never exceeds 1 at any time
- a controller with a fastest pole < 2000 rad/s when evulated at a grid point
- at least 2dB gain and 10 degree phase multiloop disk margin at every grid point

- that disturbances are compensated after 10 seconds on both channels at every grid point
- that disturbances lead to outputs of less than 5 in magnitude at every grid point

Also design LTI controllers with the same methodology and compare them to their gain-scheduled counterparts. Again, `lpvsplit` and `lpvsyn` (without basis functions) can be used.

5 LINEAR EVALUATION OVER THE GRID

We can only assess whether our controller performs well in nonlinear simulation and on the actual experiment. Nevertheless, linear analysis for individual grid points can be very helpful during the design. If the linear responses are not satisfactory, for sure, the nonlinear ones won't be either.

5.1 Frequency Response Analysis

A first step in evaluating our controller can be a frequency response analysis. We can calculate all the six different transfer functions of interest using `loopsens`. The structure `loops` then contains all closed-loop transfer functions which are accessible as

```
loops = loopsens(G,K);
lpvgetfield(loops,'So')      % S
lpvgetfield(loops,'Si')      % Si
lpvgetfield(loops,'To')      % T
lpvgetfield(loops,'Ti')      % Ti
lpvgetfield(loops,'CSo')     % KS
lpvgetfield(loops,'PSi')     % SG
lpvgetfield(loops,'Stable') % verifies that the closed loop is stable
```

Matlab uses a different convention than we do and labels the plant with P instead of our usual G, and the controller with C instead of K. Sticking to our notation, the transfer functions that we obtain are in the order of appearance $S = (I + G\,K)^{-1}$, $S_i = (I + K\,G)^{-1}$, $T = I - S$, $T_i = I - S_i$ as well as $K\,S$ and $G\,S_i$, which are the same as $S_i\,K$ and $S\,G$, respectively.

Since we are considering nonsquare plants for the output feedback case, make sure that you look at the correct input/output pairs. For example, when looking at step responses, we are interested in just y and not $\dot{y}$, so only the first two channels are of interest.

5.2 Time Domain Analysis

You can perform step response simulations on the linear models at all grid points simultaneously with the `step` command.

5.3 Robustness

Linear robustness margins cannot guarantee anything for LPV systems. Nevertheless, the same argument as for the step responses remains true: the parameter varying nature will just make things worse and hence linear margins at a grid point are a first indicator whether a design can succeed. Just as in *Control Lab—ORC2*, we only consider the multiloop disk margin and are further just interested in a worst case analysis, i.e. the lowest margins across the grid. Note that this is a VERY conservative measure, so we can't expect too much. Still, try to aim for something like 3dB gain and 15 degree phase margin.

```
[~,~,~,~,~,~,MMIO]= loopmargin(Gv,Ky);

GM = lpvgetfield(MMIO,'GainMargin');
PM = lpvgetfield(MMIO,'PhaseMargin');
lpvmin(GM(2))
minGM = db(lpvmin(GM(2)))
minPM = min(lpvmin(PM(2)))
```

Note that the analysis for the state feedback controller would require to break the loops for every *state* in addition to the outputs. You can try this on your own if you like.

6 NONLINEAR SIMULATION AND IMPLEMENTATION

6.1 Simulation Environment

The file `atc1_GyroSimulation.slx` provides a nonlinear simulation environment resembling the control moment gyroscope. The structure of the model is depicted in Figure 10.

The blue block represents the control moment gyroscope and contains the equations of motion implemented as a mex-file/C-code. It takes the control signal $u = [T_1\ T_2]^T$ as an input (in Nm). The outputs of the block are the physical available output $y = [q_3\ q_4]^T$ [rad], the estimated velocities $\dot{y} = [\dot{q}_2\ \dot{q}_3\ \dot{q}_4]^T$ [rad/s] used for feedback, the saturated (i. e. actually applied) inputs [Nm] and a signal bus that contains all states of the simulation model.

! You have to set the correct initial conditions in the file `atc1_simulation.m`

The red block represents the controllers. It also contains the reverse scalings that we used during the design.

The green blocks are associated with the user interface. On the left side, you can see the signal generator for the reference trajectory. On the right, the state bus is sorted into different signals that can be viewed in the scope. The scope data is further written to the workspace as a structure `simdata`. The file `atc1_simulation.m` contains code for plotting these data.

Prep. 6.1: Familiarize yourself with the simulation. You should be able to explain the purpose of every single block.

Prep. 6.2: Make sure that your controllers, both state feedback and output feedback, work in nonlinear simulation and deliver consistent results with what you achieved during the design phase. Pay close attention to the available control action since saturation might easily cause instability on this plant.

Prep. 6.3: Compare the two controllers. Which controller performs better and why?

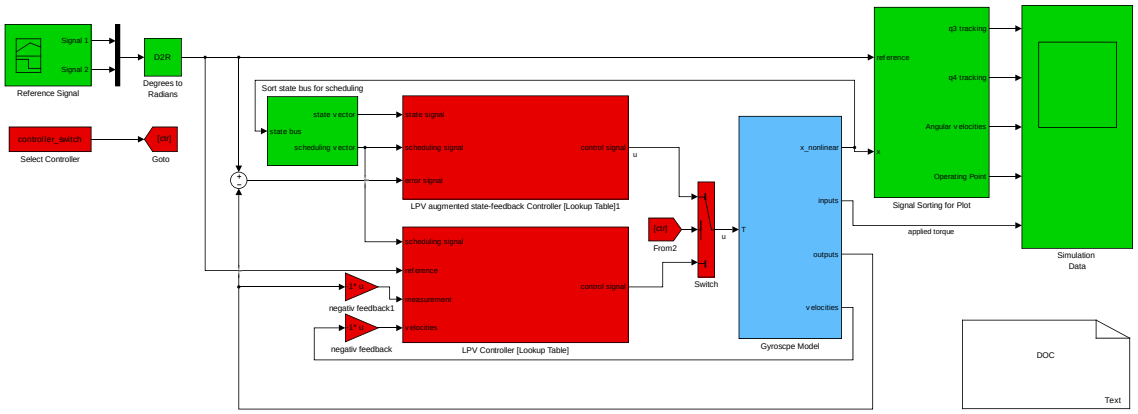


Figure 10: Simulink Simulation: Plant, Controller and interface-related blocks

Prep. 6.4: Also try out an LTI controller. In order to use the provided structure, you can simply implement it as an LPV controller with just a single grid point. Vary the size of the reference trajectory and find out the admissible operating range. Compare the results of the LPV and LTI controllers that you obtain especially with respect to the cross-coupling problem.

6.2 Lookup Table Implementation

The default method of implementing gridded LPV controllers is in terms of lookup tables with element-wise linear interpolation. Simulink provides a block called `n-D Lookup Table` for this purpose.

We need to specify the number of dimensions, the table data and breakpoints. An example implementation for the state feedback gain dependent on the three parameters $\dot{q}_1$, q_2, and q_3 looks as follows:

```
Table data:   reshape(F.Data,[size(F,1)*size(F,2), size(F.Domain)])
Breakpoints 1:  1:size(F,1)*size(F,2)
Breakpoints 2:  F.Parameter.q1dot.GridData
Breakpoints 3:  F.Parameter.q2.GridData
Breakpoints 4:  F.Parameter.q3.GridData
```

As inputs, a vector containing the linear indices `1:size(F,1)*size(F,2)` and the three parameters are required. The output is a vector containing all entries of the matrix F. It can be transformed into the correct dimensions using the `reshape` block.

A dynamic controller is implemented analogously: The three state space matrices A, B, and C can all be stored as lookup tables in the form described above and then the dynamic system is formed by including an integrator of compatible dimension.

Prep. 6.5: Verify the implementation in the Simulink model. You can use the `display` block or `scopes` to actually see the varying entries in the matrices during simulation.

Alternatively, a complete nonlinear implementation is possible that calculates the controller directly from the Lyapunov matrices in every sampling instant, see Exercise 3.2 of the *Advanced Topics in Control* lecture notes. In this case, memory requirement is usually lower, since only the coefficients of the Lyapunov matrices need to be stored. Computational complexity is on the other hand larger, since several matrix multiplications and inversions need to be performed. While this is in general more accurate and computationally indeed tractable, the implementation requires extreme care to include all scalings and transformations performed during synthesis. As these quantities are not accessible when using commercial tools such as the LPVTools, we will not address this possibility further.

7 EXPERIMENT

During the lab, the file `atc1_GyroExperiment.slx` will be provided to interface the control moment gyroscope. Its structure is depicted in Figure 11. As you can see, it is very similar to the nonlinear simulation.

The blue block represents the control moment gyroscope and contains interfaces to the Digital/Analog and Analog/Digital converters of the physical plant. The inputs and outputs of the block are the same as in simulation except that differentiation filters are used to estimate the states.

The red blocks are identical to that used in the simulation.

The green blocks are identical to that used in the simulation except that the output structure is now named `expdata`. Code for plotting this data will also be provided.

The yellow blocks are associated with a dedicated "start-up" controller and the switching logic necessary to switch to the LPV controller. The start-up controller is

used to bring the plant to a certain operating point. It consists of a simple PI controller acting on $\dot{q}_1$ and q_2. The first yellow switch on the upper left is used to activate/deactivate the start-up controller. The second can be used to reset q_3 and q_4 in order to define an operating point.

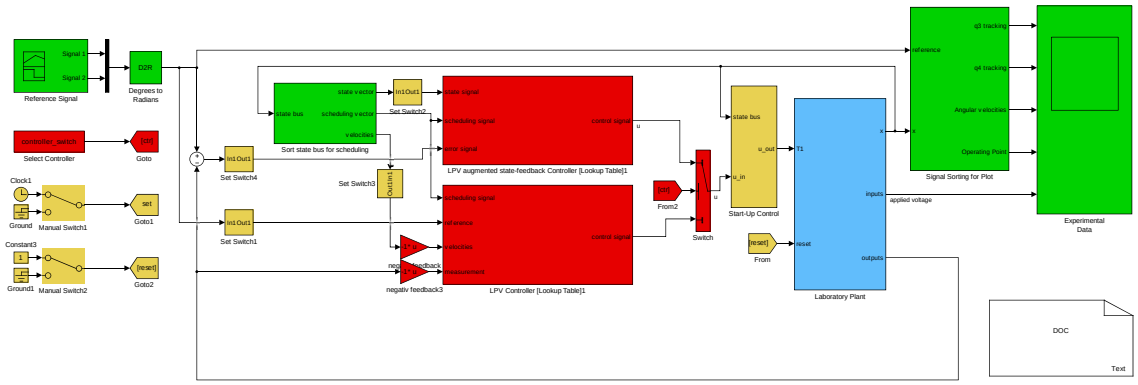


Figure 11: Simulink model: Plant, Controller, Start-Up Controller and interface-related blocks

Prep. 7.1: Make sure your design file produces all the files necessary to run the nonlinear simulation by simply executing it.

! Bring this files with you to the experiment as you may need to retune your controller.

Goal

Implement your controller in the Simulink real-time interface and try it out on the actual plant. On the predefined trajectory that captures 45° steps in q_3 and 90° steps in q_4, you should be able to achieve

- stability throughout the runtime
- a 5% settling time of less than 4 seconds on both channels

- a steady state error of less than 1% on both channels
- cross coupling should not exceed 2.5° per 45° command (i. e. should be less than 6%)
- that the control input magnitude never exceeds 10 V at any time
- that the control input is smooth and does not result in mechanical wear (you'll hear what we mean if you fail to achieve this).

If necessary, retune your controller and try again.

! Be careful and obey all safety requirements when working on the experimental device. An unstable controller can cause fast and unpredictable motion of the gimbals which can cause severe injuries.