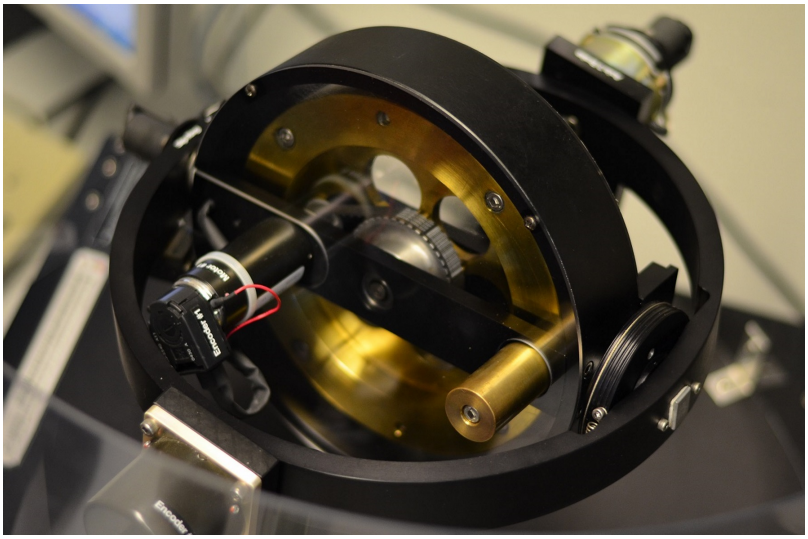
TUHH
Technische Universität Hamburg

ICS
Institute of Control Systems

Control Lab
ORC2

Robust $\mathcal{H}_\infty$ 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 *ORC2—Robust 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.

This document is accompanied by MATLAB files that you need to modify and execute in order to develop your own design.

`orc2_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 and uses the section numbers of this document for orientation.

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

orc2_GyroSimulation.slx This file contains the nonlinear simulation model.

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

orc2_simulation.m This file deals with evaluation in nonlinear simulation.

! You need to submit this file.

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 `https://www.tuhh.de/rzt/usc/matlab/index.html` or use the pool computers.

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: Review the concept of closed loop shaping in the *Optimal and Robust Control* lecture notes.

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

Task

Design a 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 `orc2_design.m`.

☐ I completed `orc2_simulation.m`.

☐ I attach 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.

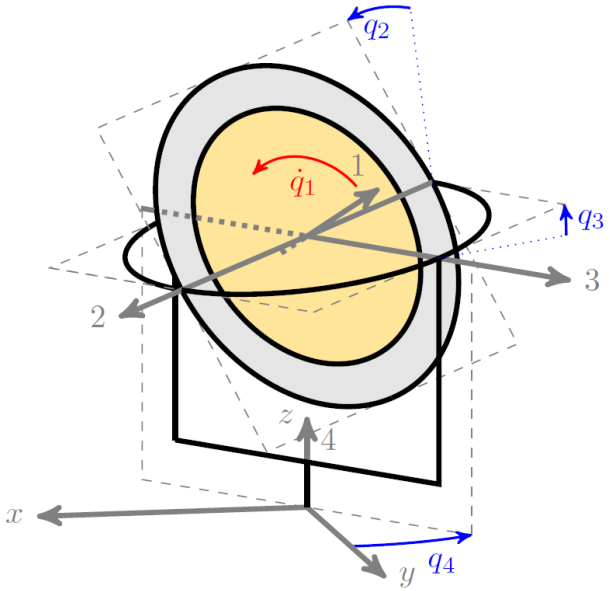


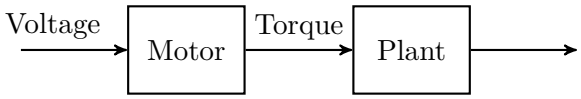
Figure 1: Kinematics of the control moment gyroscope

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

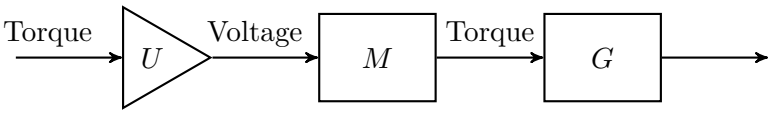
$$M(q)\,\ddot{q} + k(q,\dot{q}) = f(\dot{q}) + \begin{bmatrix} I \\ 0 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} \quad ,$$

with the generalized coordinates $q = [q_1\ q_2\ q_3\ q_4]^{\mathrm{T}}$, 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

1.1 Linearized Model

The system is nonlinear, but it can be approximated by a Jacobian linearization about an operating 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 linear time invariant (LTI) state space model

$$\begin{aligned} \dot{x} &= A\,x + B\,u\ , \\ y &= C\,x\ , \end{aligned} \tag{1}$$

with state vector $x = [\delta q_3\ \delta q_4\ \delta \dot{q}_2\ \delta \dot{q}_3\ \dot{\delta} q_4]^{\mathrm{T}}$, input $u = [\delta T_1\ \delta T_2]^{\mathrm{T}}$ and output $y = [\delta q_3\ \delta q_4]^{\mathrm{T}}$. We will drop the δ in the following to simplify notation and use the plant as

shown in Figure 3 with

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

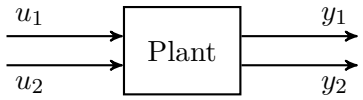


Figure 3: Inputs and outputs of the model

In order to justify our linear design, we have to make sure that we stay "close to" the operating point, which is arguably a rather blurry expression. In fact there are several methods how to quantify the region in which a model is valid but they are beyond the scope of this course. For now, we might assume that we can deal with about $\pm 5\,\text{rad/s}$ deviation in $\dot{q}_1$ and $\pm 10°$ deviation in q_2 and q_3. These are however values obtained from experience and guarantee nothing.

Prep. 1.1: Review the concept of linearizing nonlinear plants (e. g. Problem 1.10 in the *Control Systems Theory and Design* Lecture Notes) and make sure you understand that we are now considering the perturbations around an operating point, i. e. $\delta q_3 = q_3 - q_{3,0}$.

The effects of the control inputs on the controlled outputs depend strongly on the angle q_2 and thus the 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. This consideration is quite hypothetical, since applying a torque T_2 also changes q_2. Thus, in practice q_2 will almost always be nonzero and the inputs have a combined effect on the outputs. This is termed *cross coupling*. The effect of cross coupling depending on the angle q_2 is illustrated in Figure 4.

It can be observed that the directions of the cross couplings depend on the sign of q_2. For $q_2 = \pm 90°$ the effect of the inputs on the outputs completely swaps. However, we will consider only the operating point $\dot{q}_{1,0} = 45$, $q_{2,0} = 0$, $q_{3,0} = 0$ throughout the document.

Prep. 1.2: What do you expect in terms of cross couplings when we design our controller for the given (decoupled) operating point and apply it to the real plant?

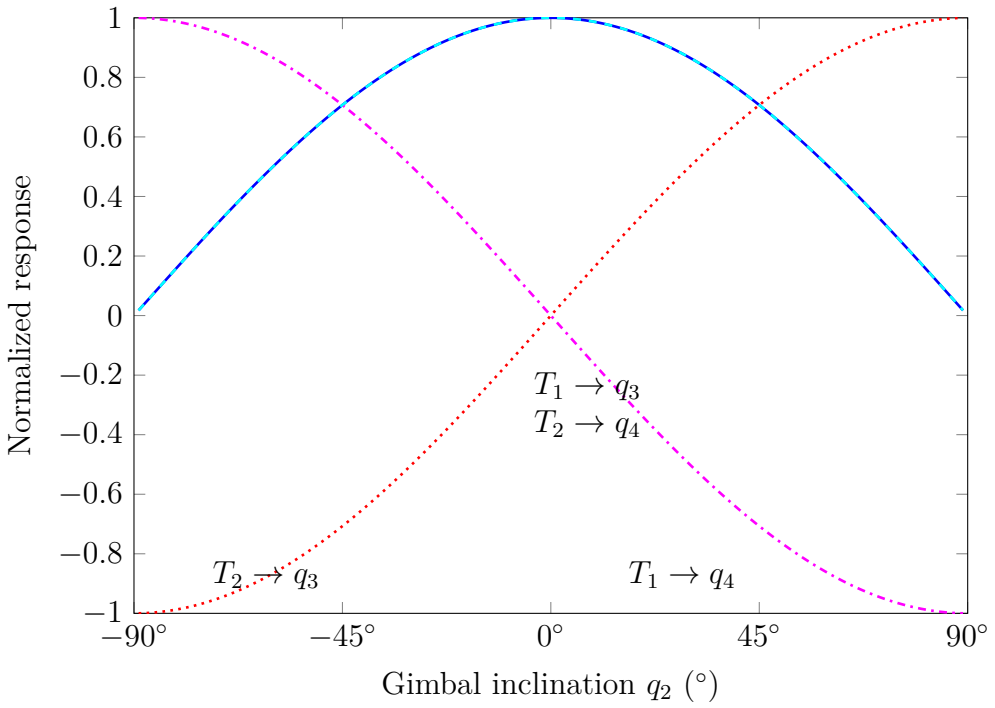


Figure 4: Normalized input response gain for different values of $q_2 \in [-90°\ ,\ 90°]$

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

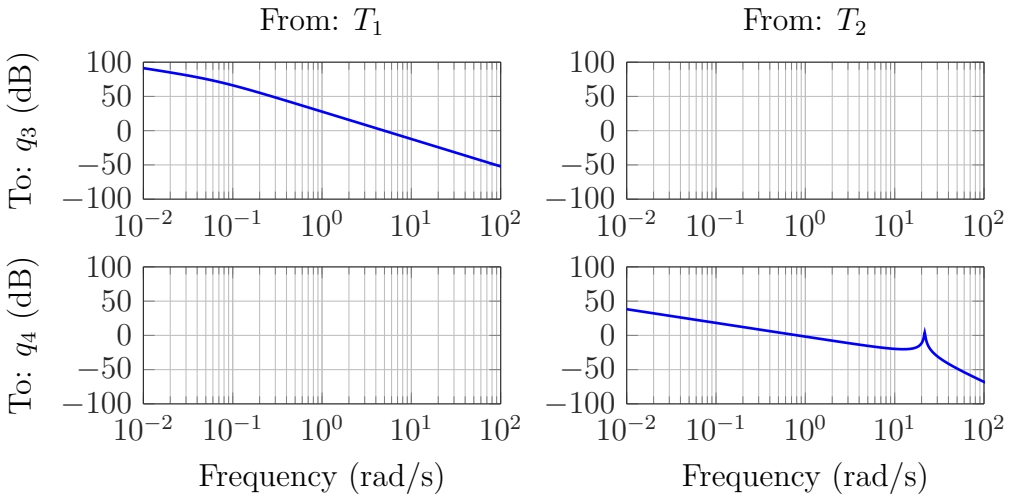
```
Gp = linearize_gyro(q1dot_0,q2_0,q3_0);
```

which contains a parameterization of the linearization at different operating points. You can use the commands

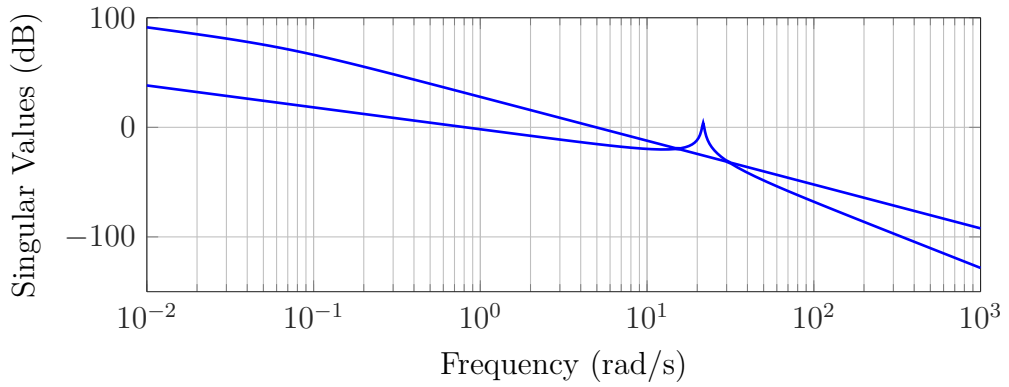
```
bodemag(Gp)
sigma(Gp)
```

to bring up a Bode magnitude plot and a Sigma plot of the plant as shown in Figure 5. Note that the selected design point does not cover any cross-couplings.

Prep. 1.3: Recall why we usually use Sigma plots instead of Bode plots for multivariable systems and why it is not that important in the special case considered here.



(a) Bode magnitude plot of the Plant



(b) Sigma plot of the Plant

Figure 5: Frequency response of the plant

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 1 corresponds both to the maximum expected change in the reference signal $D_{r,\max}$ and the maximum actuator capacity $D_{u,\max}$, i. e.

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

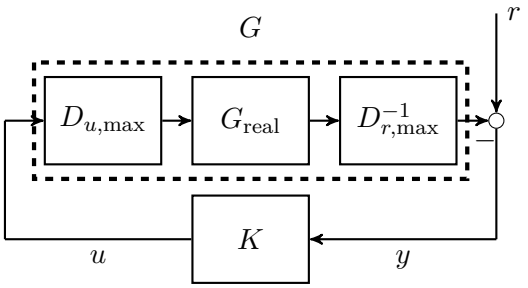
$$y = D_{r,\max}^{-1}\, y_{\text{real}} \tag{3}$$

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 reference step now corresponds to the maximum input that we expect and that the available control signal is now also one, see Figure 6. Thus, if a unit step results in a control signal less than one, 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:

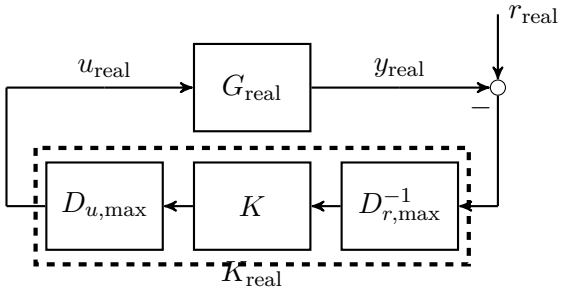
$$u = K\,y \tag{4}$$

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

Figure 6 illustrates the procedure.



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



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

Figure 6: 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

$$q_{3,\max} = 10\frac{\pi}{180}, \qquad q_{4,\max} = 45\,\frac{\pi}{180}$$

since we mostly control the rotation about the vertical axis described by q_4.

2 CONTROLLER DESIGN

The task will be to design an LTI output feedback controller, which stabilizes the plant and achieves performance specifications in terms of the $\mathcal{H}_\infty$-norm of a generalized plant.

2.1 S/KS closed-loop shaping filter design

Mixed sensitivity design methods are useful tools in order to express design specifications on the controller. The main ideas of these methods should be already familiar (see lecture notes *Optimal and Robust Control*). We decide to start with an S/KS weighting scheme, which is more or less a standard approach. 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 singular values of the transfer functions. These filters are part of the generalized plant.

While we could use the parameterization from the ORC lecture notes and build the filters manually as `tf` objects, a built-in command that we can use 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` species the frequency at which the gain is one and `feedthroughgain` specifies the high frequency gain. For W_S, the steady-state gain determines our inverse error constant, the bandwidth of the error dynamics and the feedthrough gain limits the peak of S. For W_K, the low-frequency gain determines available control effort, the bandwidth corresponds to the available actuator rates and the feedthrough gain limits authority at high frequencies. Note that these filters despite having 'clear interpretation' are tuning knobs, so you can (and should) iteratively adjust them for your design.

A good starting point for tuning is nevertheless provided by physical insight and this is again a reason why we needed to scale the plant in the previous section. For tracking applications, we usually require a small steady state error and hence the sensitivity function must be close to zero in the low frequency range. You can think of it as the maximum error measured in the size of our maximum change in the reference signal. Thus if we want a steady state error of less than 1 % of this value, we should set `dcgain1` to at least 100; if we desire 0.1 % we have to use 1000 and so on. The feedthrough gain of the sensitivity filter on the other hand determines our peak and since we know that $S = I$ at large frequencies, any choice larger than one does not make any sense. Usually we want to limit the peak of the sensitivity function to something like 6 dB (that means a factor of two) and consequently `feedthroughgain1` $= 0.5$ is usually a good initial value.

The bandwidth, on the other hand, determines how fast the error decays and thus gives us direct control over the speed of the response of the system. It helps to think in terms of a first order system, although of course the sensitivity function will be of higher order. So if we want, let's say a settling time of 4 s, a bandwidth of $1\,\frac{\text{rad}}{\text{s}}$ would be roughly what we are aiming at. The choice of W_K follows the same considerations. In the low frequency range it provides an upper bound on the available actuator capacity, which with proper scaling is one. Thus, choosing something like 0.9 for `dcgain2` appears to be reasonable (we cannot directly choose one with the parameterization that we use, thus we choose something close by). To enforce a roll-off, we should use something like 100 or 1000 in the high frequency range. A good starting point for the bandwidth is the actual physical actuator bandwidth, although noise considerations can also be used. Figure 7 shows a typical selection of weighting filters for the control moment gyroscope. These weights are used throughout this document in an example design.

Prep. 2.1: Review the concept of closed-loop shaping, e. g., in Chapter 17 and 18 of the *Optimal and Robust Control* lecture notes. Make sure you understand how the tuning works qualitatively.

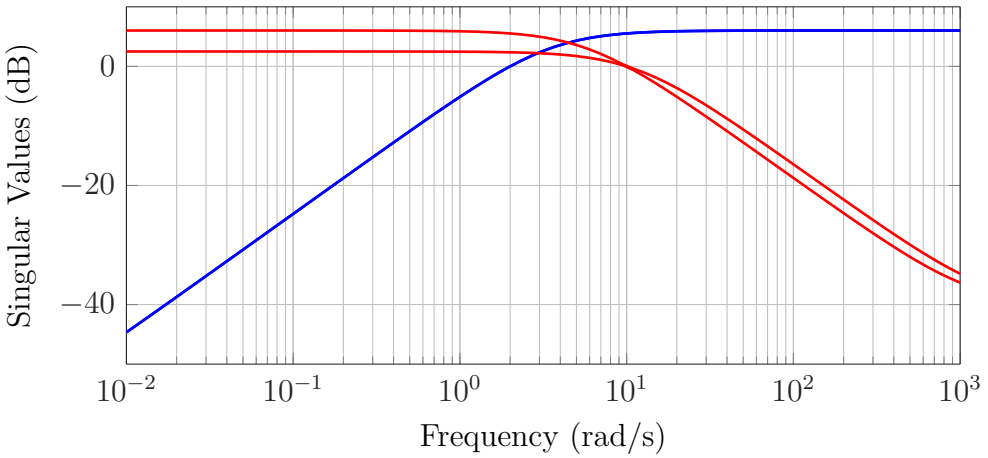


Figure 7: Choice of weighting filters. Plotted are the inverse weights W_S^{-1} (blue) and W_K^{-1} (red)

2.2 Generalized Plant formulation

As you remember from the lecture *Optimal and Robust Control*, a generalized plant is of the form

$$\begin{aligned}\dot{x} &= Ax + B_w w + B_u u \ , \\ z &= C_z x + D_{zw} w + D_{zu} u \ , \\ v &= C_v x + D_{vw} w \ ,\end{aligned}$$

with external input w, controller output u, performance output z and controller input v. For more details review Chapter 19 of lecture notes *Optimal and Robust Control*.

For our synthesis tools to work, we need to first assemble a generalized plant that includes the desired performance inputs and outputs. The generalized plant for the S/KS formulation is depicted in Figure 8.

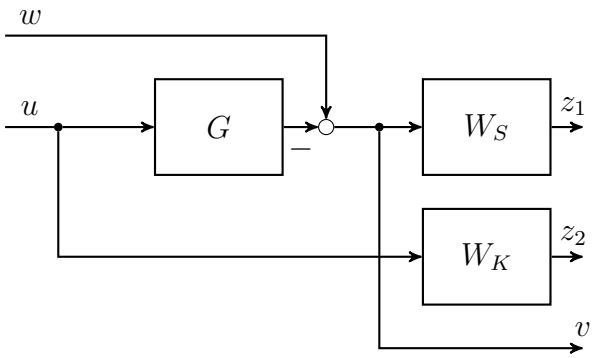


Figure 8: Open-loop generalized Plant P for the S/KS mixed sensitivity design

In order to assemble it, it is convenient to use MATLABS `sysic` command. 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
```

You can verify that you correctly interconnected the systems by using the built-in command `augw` (*aug*ment with *w*eights),

```
P = augw(G,W_S,W_K);
```

which does exactly the same thing but may order the states differently. Therefore you should use Sigma plots to verify your results instead of looking at the matrix entries.

2.3 Synthesis

Stability and $\mathcal{H}_\infty$ constraints can be expressed as linear matrix inequalities (LMIs). If there exists a Lyapunov matrix P, which satisfies

$$P^T = P > 0\,,$$
$$\begin{bmatrix} A^TP + PA & PB & C^T \\ B^TP & -\gamma I & D^T \\ C & D & -\gamma I \end{bmatrix} < 0\,, \quad \text{(Theorem 18.3 ORC lecture notes)}$$

the system is stable and $\|T_{zw}\|_\infty < \gamma$ for the overall transfer function T_{zw} from w to z and some performance index $\gamma > 0$.

The output feedback controller synthesis is based on solving such LMIs for the closed-loop system, which is shown in Figure 9. Because the inequality constraints will be nonlinear in the variables, transformations are applied in order to develop equivalent LMI constraints, which are affine in the variables. Since the controller variables were also affected by the transformations, the computed controller has to be transformed back in order to obtain the controller for the original system (see pp. 137 of the *Optimal and Robust Control* lecture notes for more details).

This quite lengthy procedure is luckily already implemented in MATLAB, where we

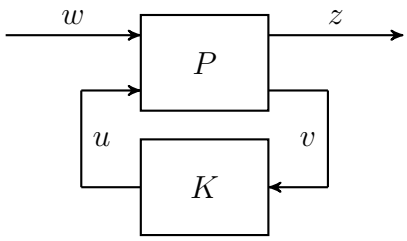


Figure 9: Closed-loop interconnection of generalized plant and controller

just need to provide the generalized plant `P` and the number of available measurements `nmeas` and control inputs `ncont`:

```
[K,CL,gam,INFO] = hinfsyn(ssbal(P),nmeas,ncont,'method','lmi');
```

The first thing to note here is that we do not use the generalized plant "as it is" but that we do apply a so called balancing transformation by invoking `ssbal`. This transformation scales the states of the system in such a way that the entries in the `B` and `C` matrices are "of the same size". Since a state transformation does not affect the input-output behavior of the plant, we don't need to reverse this transformation later on. Its only purpose is to avoid numerical issues related to representing the numbers with different orders of magnitude in finite precision. It is easy to find examples in which the synthesis code without this balancing does not produce any useful results at all, so balancing is one of the things that should always be performed to avoid unnecessary numerical issues.

A second thing to note is that optimality with respect to the $\mathcal{H}_\infty$ norm is not necessarily what we are looking for. The reasons here are a bit more complex, and again are largely related to numerical issues within the solution of the LMI problem. Further, theory suggests that a slight decrease in $\mathcal{H}_\infty$ performance can largely increase performance as measured in the $\mathcal{H}_2$-norm, which in general is also a good thing. So often a suboptimal, rather than a truly optimal controller is desired. (Note further that even the solution of the optimization problem is only approximately optimal because we use numeric methods to find it.) We can get a suboptimal controller with a specified loss-of-performance by again calling the `hinfsyn` command, but this time tell the optimization to stop when a predefined γ is reached. We know from the first synthesis that we can find a controller that achieves `gam`. Hence there also exists a controller that achieves at least 1.1 `gam`,

which is what we solve for in the second step. This condition means, that the performance degradation in the $\mathcal{H}_\infty$ sense is at most 10 %.

```
[K,CL,gam,INFO] =
            hinfsyn(ssbal(P),nmeas,ncont,'method','lmi','GMIN',1.1*gam);
```

Note that we need a first run to determine the `gam` that we use in this second synthesis. What might also be of interest for implementation, is the fastest controller pole:

```
fastestpole = max(abs(eig(K.a)))
```

For the example, we get $1.5118 \cdot 10^3 \frac{\text{rad}}{\text{s}}$, which is relatively fast. But since we have a sampling rate of 1100 Hz available on the experimental device, it should nevertheless cause no problems.

Prep. 2.2: Verify that the pole is not too fast for implementation.

Prep. 2.3: Why is it desirable to accept suboptimality in $\mathcal{H}_\infty$ performance? Think about possible benefits of better $\mathcal{H}_2$ performance especially for transfer function KS related to the control effort.

3 LINEAR EVALUATION

3.1 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);
loops.So    % S
loops.Si    % Si
loops.To    % T
loops.Ti    % Ti
loops.CSo   % KS
loops.PSi   % SG
loops.Stable % 1 if 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. Further note that the `loopsens` command expects a *negative feedback* controller. Depending on how you defined your generalized plant, you might need to use `loopsens(G,-K)` instead. This is again one of the many reasons why we should always check whether our interconnection is really stable.

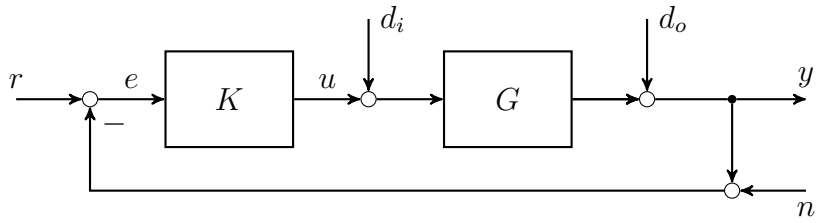


Figure 10: Closed-loop system with input and output disturbance, noise and reference signal.

Prep. 3.1: Consider the block diagram shown in Figure 10. Derive the six transfer functions stated above on your own and pay attention to the fact that transfer function matrices in general do not commute, i. e. $G\,K \neq K\,G$. Make sure you understand which signal path (e. g. input disturbance to error) each of the transfer functions represents. Chapter 16 of the *Optimal and robust Control* lecture notes will help you getting started.

We can identify quite a few relevant properties of our design by looking at the frequency response of what is sometimes called "the gang of four" (S, SG, KS, T_i). Note that these are exactly the four transfer functions that define internal stability of feedback interconnections, see *Control Systems Theory and Design* lecture notes pg. 86. The closed-loop frequency response for our example S/KS design is shown in Figure 11. For example, we can see that we will achieve tracking of step responses since the sensitivity function starts with a slope of +20 dB per decade. Further, KS tells us that the controller rolls off at high frequencies, i.e., noise does not cause large control signals. The transfer function, SG on the other hand, shows the same peak as our open loop plant, which indicates poor damping. This is due to the fact that the S/KS formulation encourages the controller to simply cancel lightly damped poles. Further, the low frequency gain of SG is large, which means that input disturbances are not suppressed but amplified! This is a consequence of the fact that the controller does not include integral action but

just makes use of the integral behavior of the plant to achieve tracking. We can verify this by taking a look at the frequency response of the controller shown in Figure 12.

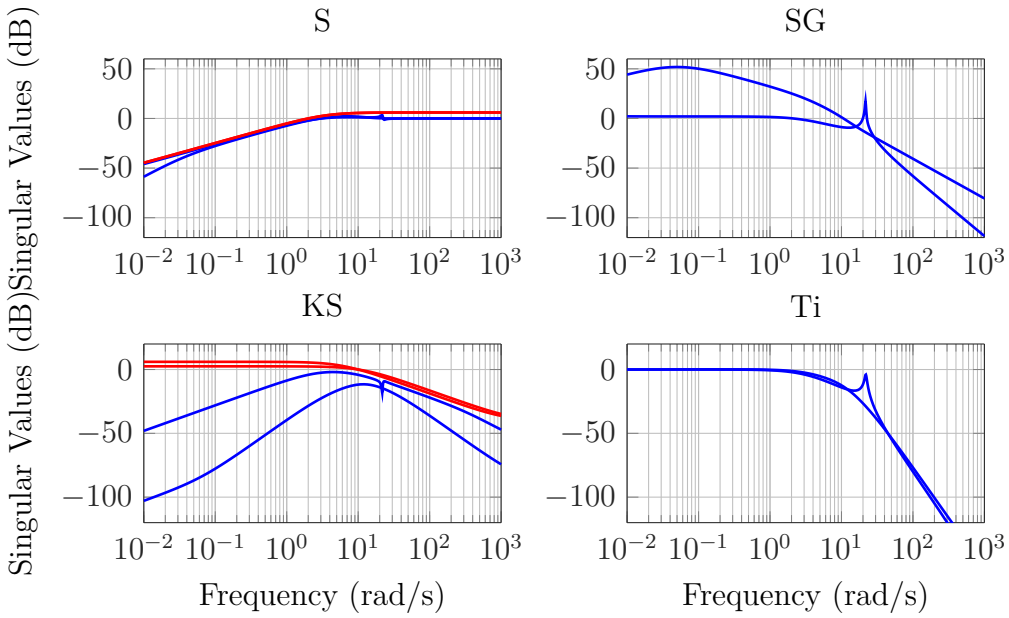


Figure 11: Closed-loop transfer functions (blue) with S/KS mixed sensitivity design and inverse weights (red

3.2 Time Domain Analysis

The pole cancellation problem with the S/KS design can be best realized when looking at the pole zero map of the open-loop plant and the controller shown in Figure 13. You can bring it up by using

```
pzplot(G,K,'r')
```

The pole-zero map shows us that the controller indeed simply cancels the lightly damped pole pair of the plant. Since this cancellation does not occur when looking at the transfer function $S\,G$, input disturbances are not adequately damped.

Prep. 3.2: Why is input disturbance rejection important?

Since we are mainly interested in tracking a signal in the time domain, let's take a look at the step responses of the complementary sensitivity T. It shows us how the

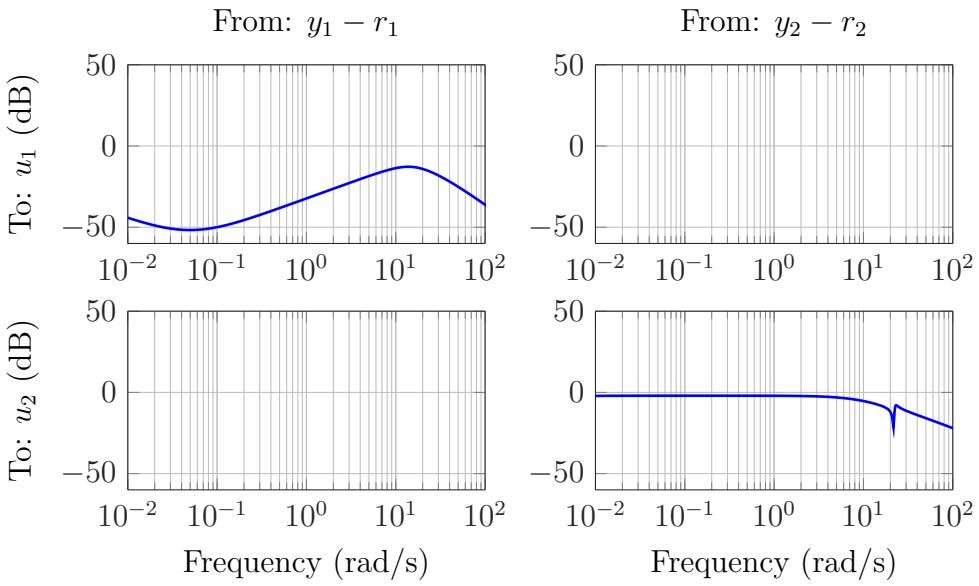


Figure 12: Example controller synthesized with S/KS mixed sensitivity design

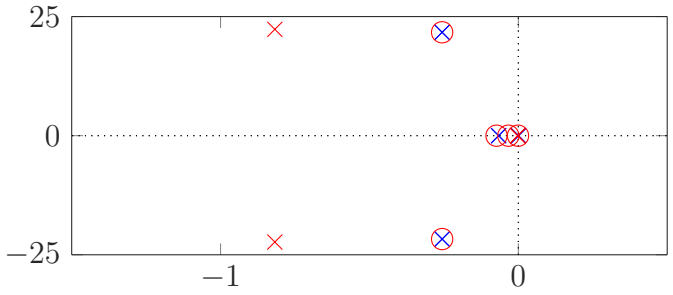


Figure 13: Typical pole-zero map of plant (blue cross) and a controller synthesized with S/KS mixed sensitivity design (red cross/red dots)

plant reacts to a change in the reference signal. If we are further interested in what control effort is required, we can plot the step response of $K\,S$. Since we scaled the plant properly, both responses should be of the same magnitude and we can plot them together in Figure 14.

```
step(loops.To,'b',loops.CSo,'g',10)
```

Even though there is some oscillation visible in the second response, the overall tracking can be seen to be sufficient. If, on the other hand, we consider input disturbances, the deficits of the present design that we expected from the frequency response analysis become very apparent:

```
step(loops.PSi,'b',10)
```

Figure 15 shows that disturbances are amplified and weakly damped.

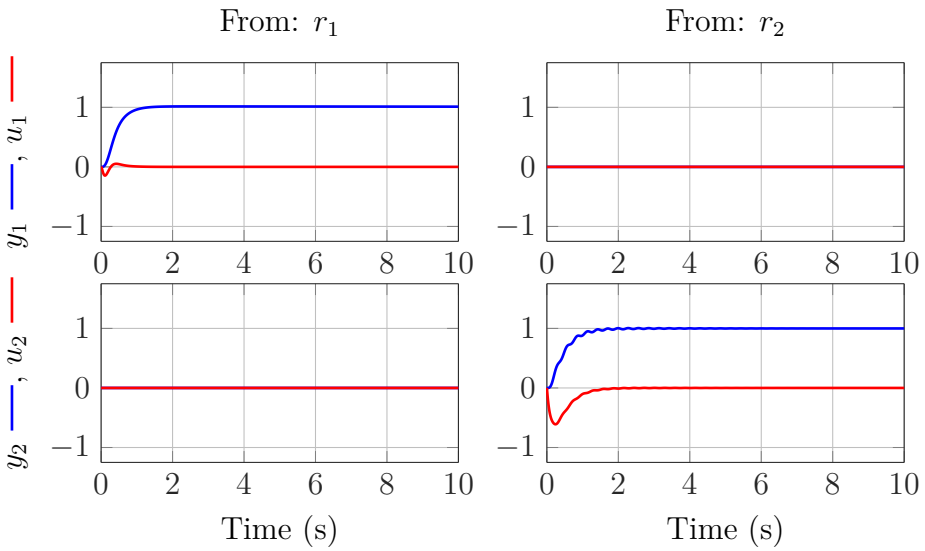


Figure 14: Step response with a controller synthesized with S/KS mixed sensitivity design

3.3 Robustness

There is a variety of linear robustness margins. For simplicity, we only consider the (rather conservative) multiloop disk margin. These are computed by using what is called structured singular values and which are beyond the scope of the *Optimal and Robust Control* class. The important thing however is that they correspond to simultaneous gain and phase variations at *all* plant inputs and outputs and are therefore a relatively safe measure for robustness of MIMO systems.

```
mm = loopmargin(G*K,'m');
minGM = db(mm.GainMargin(2))
minPM = min(mm.PhaseMargin(2))
```

For the example design, we get something like 9 dB gain margin and 50 degree phase margin, which is an excellent result.

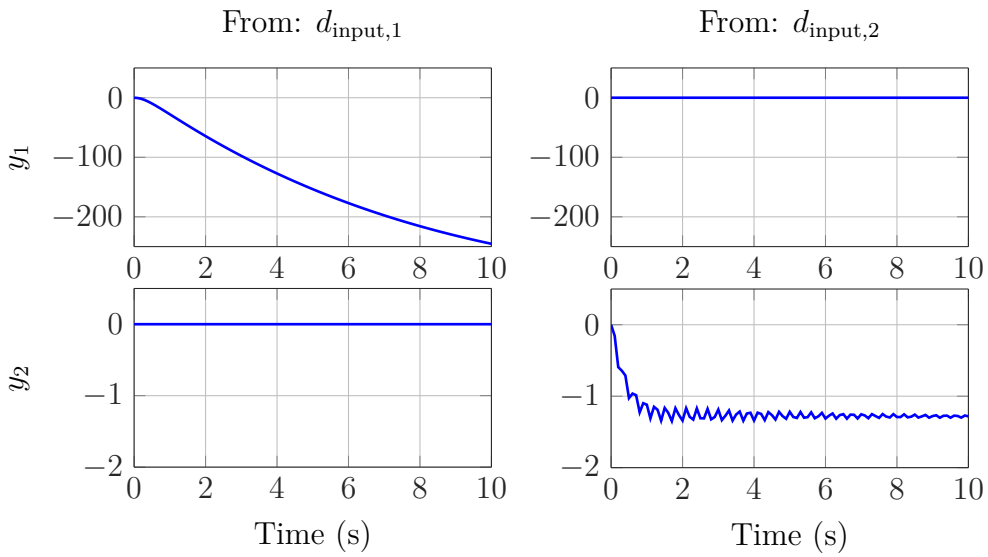


Figure 15: Response to a step disturbance at the plant input with a controller synthesized with S/KS mixed sensitivity design

How to Proceed

Take a look at the provided MATLAB file and complete it in a way that allows you to design your own controller. Repeat the design process a few times and iteratively adjust the weights to see what happens. Once you have a design that looks promising in the linear analysis, you can proceed to the following Section 4, which discusses a more advanced design approach.

To give you a little orientation, try to come up with a design that achieves

- a 5 % settling time of less than 2 seconds on both channels
- a steady state error of less than 1% on both channels
- that the control input magnitude never exceeds 1 at any time
- a controller with a fastest pole $< 2000\,\frac{\text{rad}}{\text{s}}$
- at least 3 dB gain and 25 degree phase multiloop disk margin

4 ADVANCED DESIGN CONCEPTS

There is a variety of different design approaches other than the S/KS formulation that we discussed in Section 3. We will consider a relatively simple extension to show you that design is inherently about trade-offs (usually robustness vs. performance) and that there is usually room for improvement at the expense of (design) complexity...

4.1 Four-block problem

In what is called a four-block design, we explicitly consider input disturbances and therefore avoid the cancellation problems known from the S/KS formulation. The only difference in this setup is the additional input disturbance weight W_d, which is set to identity for our example design. The generalized plant is depicted in Figure 16 and the only difference to the S/KS formulation is that an additional disturbance at the plant input is considered. There are thus not two but four transfer functions involved.

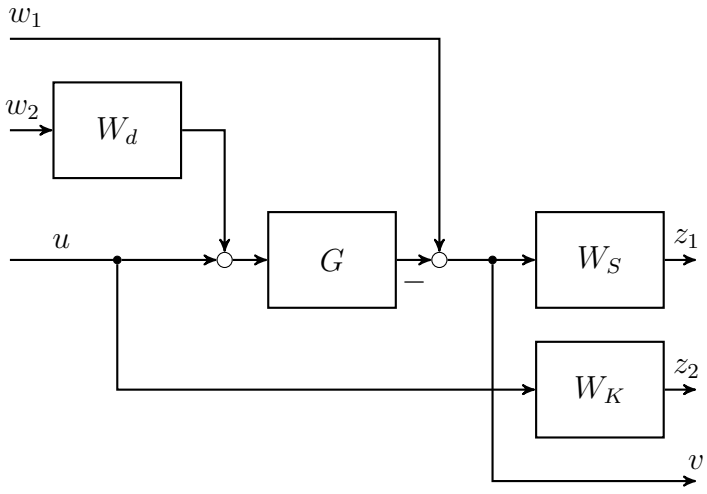


Figure 16: Open-loop generalized Plant P for the four-block mixed sensitivity design

Prep. 4.1: Figure out which four transfer functions are involved. Review exercise 17.2 of the *Optimal and Robust Control* lecture notes.

In MATLAB, we can set up the generalized plant with

```
systemnames = 'G Wd W_S W_K';
inputvar = ['[w1{2}; w2{2}; u{2}]'];
outputvar = ['[W_S; W_K; w1-G]'];
input_to_G = '[u+Wd]';
input_to_Wd = '[w2]';
input_to_W_S = ['[w1-G]'];
input_to_W_K = '[u]';
P = sysic;
```

and then run the same synthesis procedure as before.

The main difference in the closed-loop transfer functions shown in Figure 17 now is that the transfer function SG related to the input disturbance is drastically decreased in magnitude. In the low frequency range it tends to zero, thus step disturbances are completely regulated in steady state. To achieve this, the controller now actually includes integral action, i. e., it has a high gain at low frequencies. We can verify this by looking at Figure 18. We further see from the pole-zero map in Figure 19 that there are no pole cancellations anymore.

If we look at the responses to the reference step in Figure 20, the result looks however much worse than with the S/KS controller. The large overshoot is—loosely speaking—caused by the steep slope of the sensitivity function. If you take a close look at the sensitivity plot in Figure 17, you can see that the sensitivity now has a +40dB per decade slope, which is 20dB more than we desired. This is a consequence of the fact that we need integral action to reject disturbances and that in the sensitivity function, both integral behavior from the plant and the controller add up. The disturbance response in Figure 21 now looks much better than before but if we test for robustness, we note that it is decreased (to around 5 dB gain and 30 degree phase). Again, this is a consequence of the larger controller gains, or more precisely of the larger control bandwidth.

We see, that while we indeed are able to improve disturbance rejection, we pay a price for this in terms of trajectory following as well as robustness. Such trade-offs are inherent to feedback control design and are most of the time impossible to overcome. Experiment with the four-block design to confirm this on your own. In general, decreasing W_d brings the design closer to (with $W_d = 0$ being obviously the same as) the S/KS design. That is, decreasing W_d should improve trajectory tracking and robustness, while the disturbance rejection gets worse.

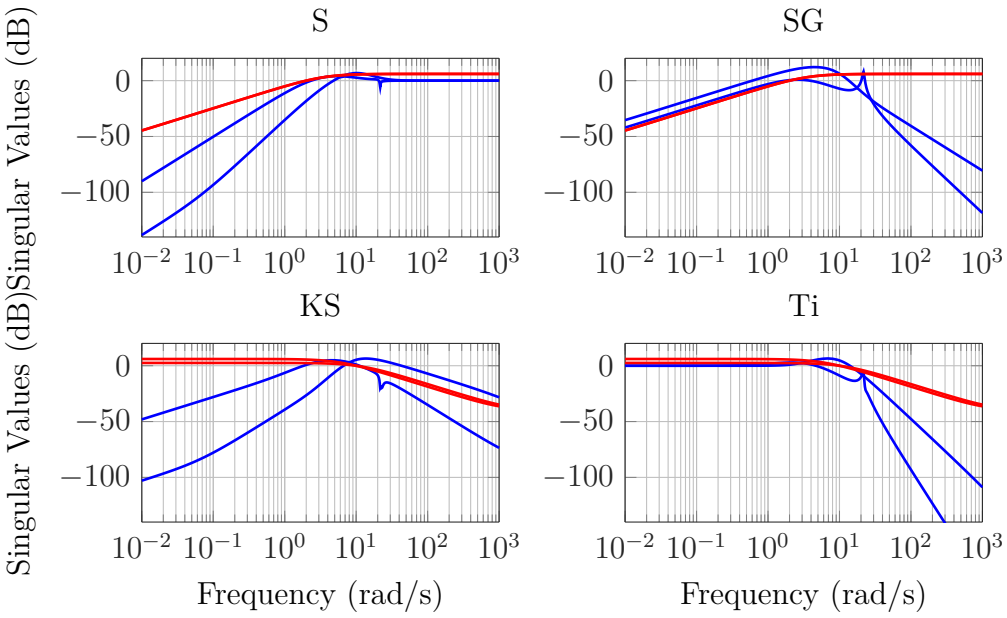


Figure 17: Closed-loop transfer functions (blue) with four-block mixed sensitivity design and inverse weights (red)

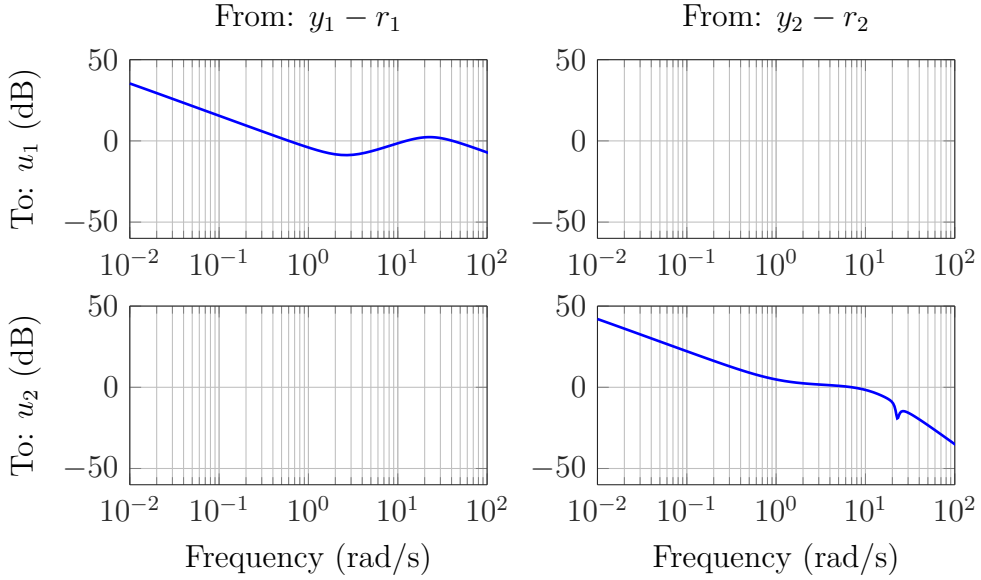


Figure 18: Example controller synthesized with four-block mixed sensitivity design

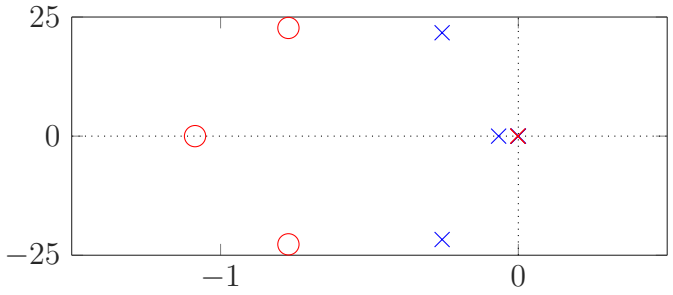


Figure 19: Pole-zero map of plant (blue cross) and a controller synthesized with four-block mixed sensitivity design (red cross/red circle).

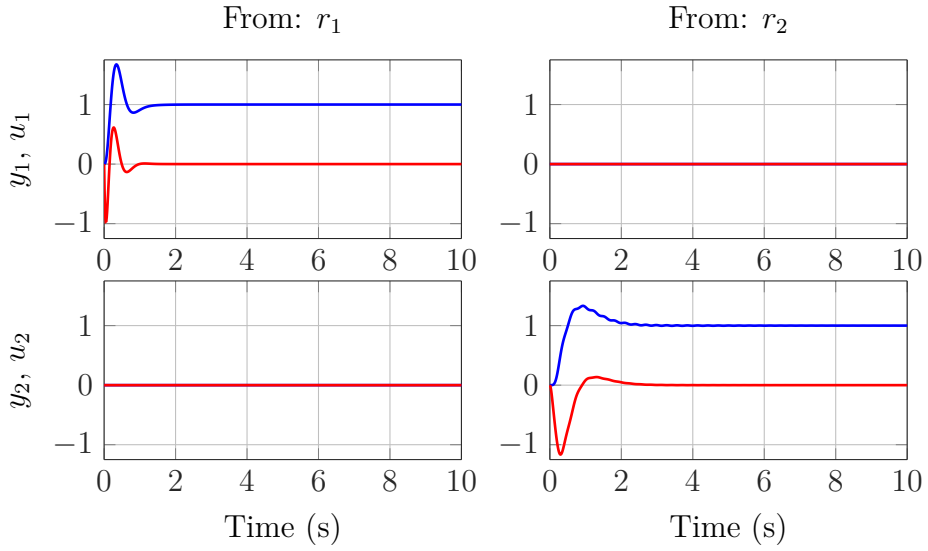


Figure 20: Step response with a controller synthesized with four-block mixed sensitivity design

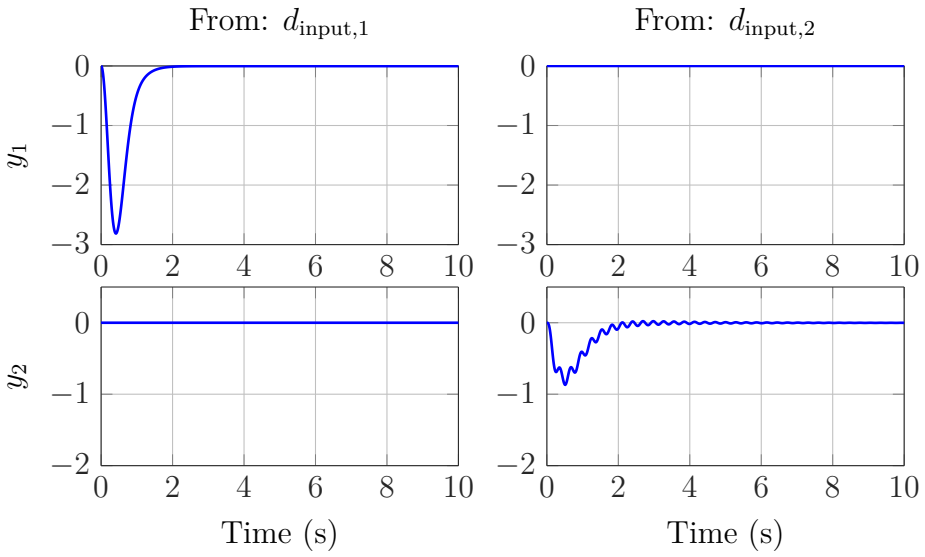


Figure 21: Response to a step disturbance with a controller synthesized with four-block mixed sensitivity design

4.2 Design with two-degrees-of-freedom

If you played with the tuning of the four-block problem, you might have realized that it is very hard to achieve satisfactory tracking performance and disturbance rejection simultaneously. The main reason is that disturbance rejection and tracking are coupled: Since the controller receives only the error signal, both unknown disturbances and known set-point changes enter the controller in the same way. We can however take advantage of the fact that they are not the same by processing them separately. This is referred to as two-degrees-of-freedom control and—without going into detail—is usually advantageous for tracking applications. We consider a very simple modification of the four-block problem that results in a two-degrees-of-freedom control law. The only necessary change in the generalized plant formulation is the additional input r and the fact that it is directly provided as a measured signal to the controller as shown in Figure 22.

The closed-loop transfer functions (Figure 23) now are derived using only the feedback path of the controller ($v_1 = \begin{bmatrix} y_1 & y_2 \end{bmatrix}^{\mathrm{T}}$) and not the two additional feedforward inputs ($v_1 = \begin{bmatrix} r_1 & r_2 \end{bmatrix}^{\mathrm{T}}$). The controller is shown in Figure 24. We can build the closed loop (which is now different from the complementary sensitivity T since it involves the two-degree-of-freedom controller) using `sysic`:

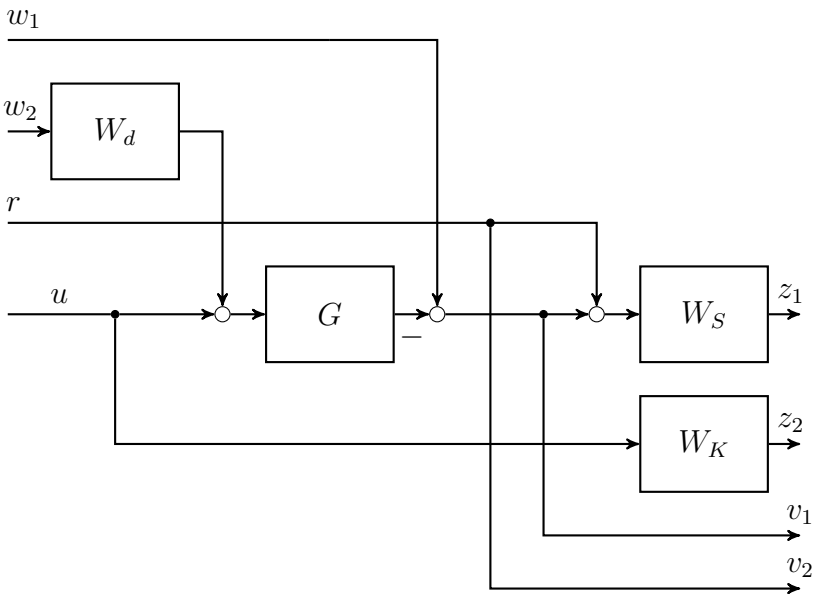


Figure 22: Open-loop generalized Plant P for the two-degrees-of-freedom design

```
systemnames = 'G K';
inputvar = ['[r{2}]'];
outputvar = '[G]';
input_to_G = '[K]';
input_to_K = '[-G;r]';
CL = sysic;
```

The controller now receives the measurement and the reference as two independent signals. The step response to changes in the reference signal now looks exactly as we want: no overshoot, and almost no oscillations (Figure 25). The response to the input disturbance (Figure 26) looks also nice and very similar to the one that we obtained from the four-block design. Also, it can be confirmed that robustness is pretty much the same as with the four-block design (around 5dB gain and 30 degrees phase for the example design).

The two-degrees-of-freedom design allows to combine the benefits of the S/KS design (good tracking performance) with the benefits of the four-block design (integral disturbance rejection). It thus relieves the trade-off between tracking performance and disturbance rejection that we encountered in the single-degree-of-freedom four-block design. The second inherent trade-off—disturbance rejection vs. robustness—is however

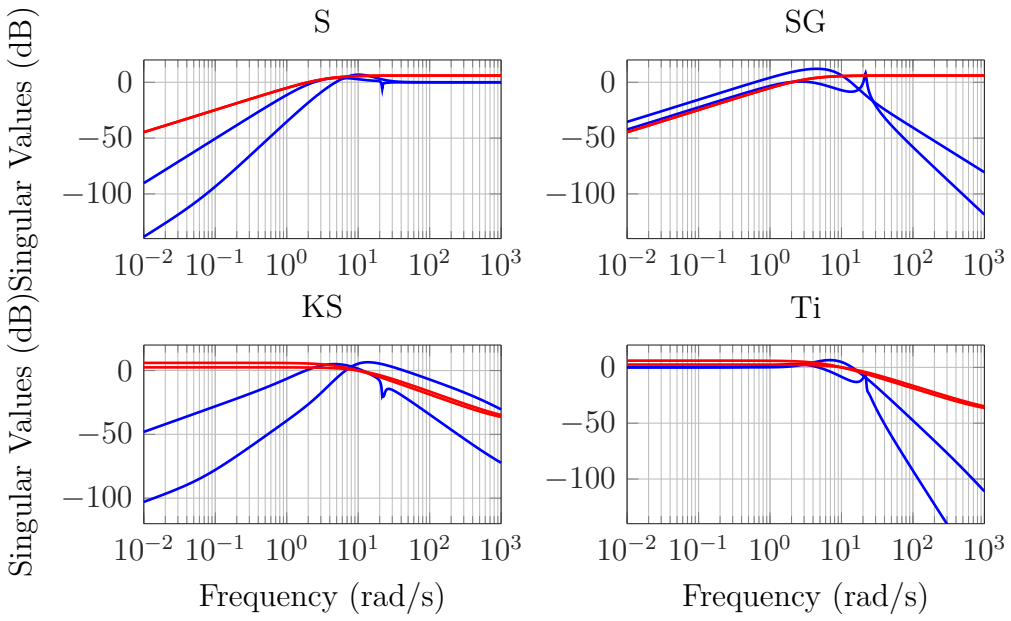


Figure 23: Typical closed-loop transfer functions (blue) with two-degrees-of-freedom design and inverse weights (red)

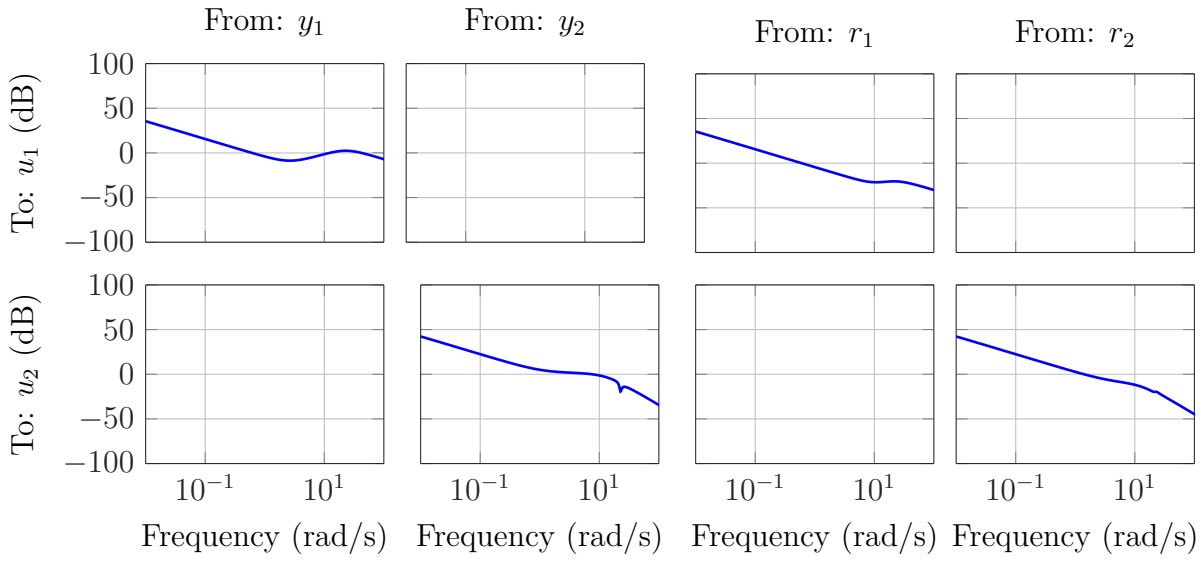


Figure 24: Typical controller synthesized with two-degrees-of-freedom design

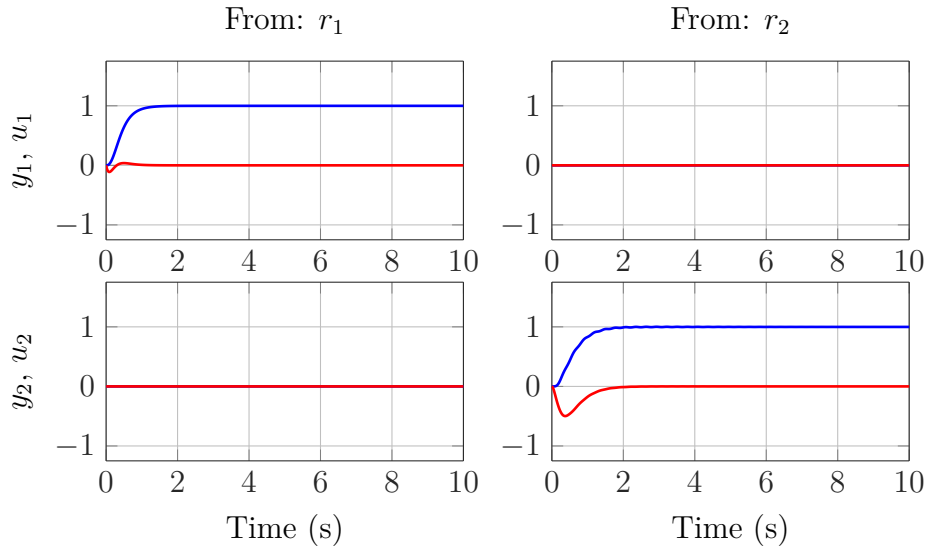


Figure 25: Typical step response with a controller synthesized with two-degrees-of-freedom design

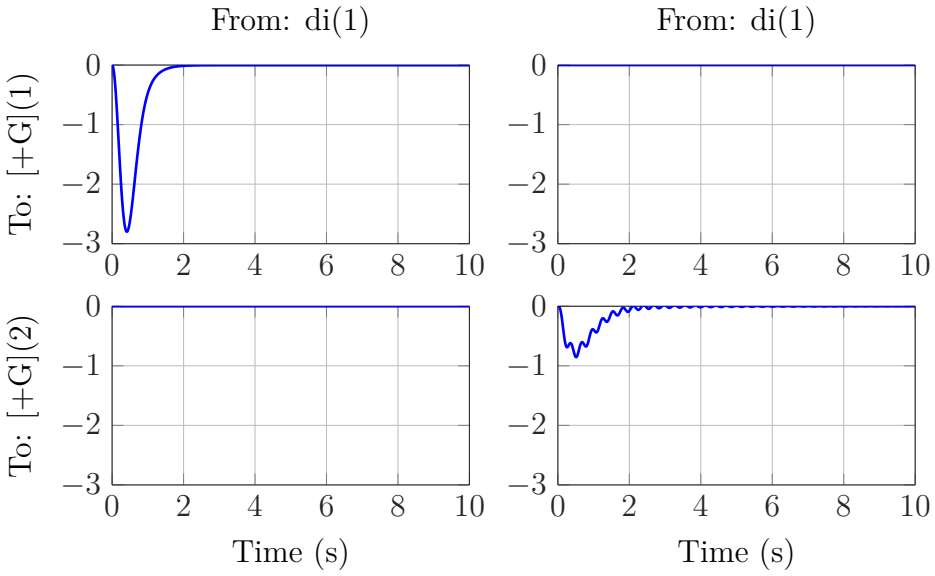


Figure 26: Typical response to a step disturbance with a controller synthesized with two-degrees-of-freedom design

still very visible (as confirmed by the lowered margins) and as designers we have to pay close attention which is more important to us in a certain situation.

Prep. 4.2: Draw conclusions about the three design approaches and compare them with each other. What are the benefits of each and which trade-offs do they address? Is there still a trade-off in the two-degrees-of-freedom control design?

How to Proceed

Again, repeat the design process a few times and iteratively adjust the weights to see what happens. Once you have a design that looks promising in the linear evaluation, you can proceed to the following Section 5 to verify your controller in nonlinear simulation.

To give you some orientation, try to come up with a design that achieves

- a 5% settling time of less than 2 seconds on both channels
- a steady state error of less than 1% on both channels
- that the control input magnitude never exceeds 1 at any time
- a controller with a fastest pole < 2000 rad/s
- at least 3 dB gain and 25 degree phase multiloop disk margin
- that disturbances are compensated after 2 seconds on both channels
- that disturbances lead to outputs of less than 3 in magnitude

5 NONLINEAR SIMULATION

The file `orc2_GyroSimulation.slx` provides a nonlinear simulation environment resembling the control moment gyroscope.

The structure of the model is depicted in Figure 27.

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 $u = [T_1 \;\; T_2]^{\mathrm{T}}$ as an input (in Nm). The outputs of the block are the physical available output $y = [q_3 \;\; q_4]^{\mathrm{T}}$ [rad] 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 `orc2_simulation.m`

The red blocks represent the controllers and a switching logic to select whether single-degree-of-freedom or two-degrees-of-freedom control is used. It also contains the reverse scalings that we used during the design.

! You have to select which controller is used in the file `orc2_simulation.m`

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 `orc2_simulation.m` contains code for plotting this data.

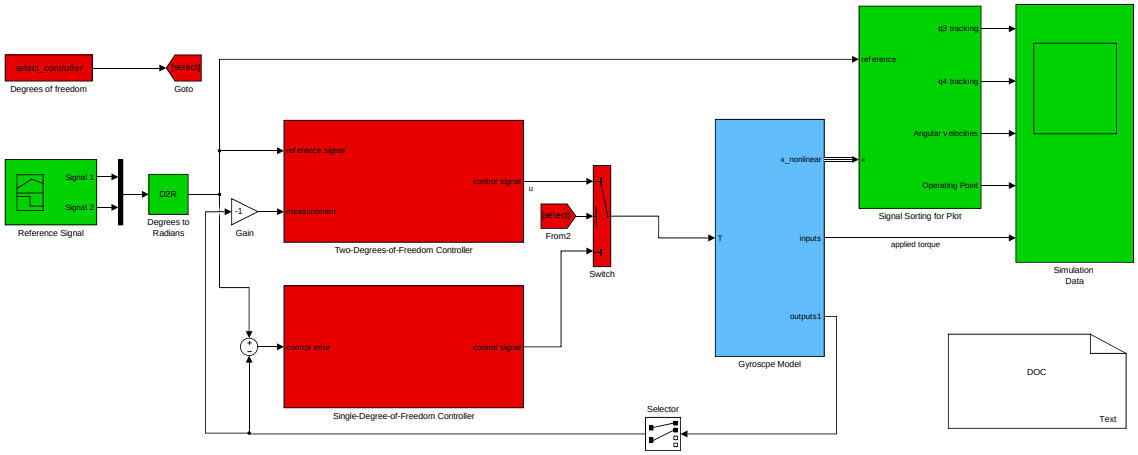


Figure 27: Simulink Simulation: Plant, Controller and interface-related blocks

Prep. 5.1: Familiarize yourself with the simulation. You should be able to explain the purpose of every single block.

If we apply our two-degrees-of-freedom controller example design to the nonlinear simulation, we get the results depicted in Figures 28 and 29. They look reasonably similar to the linear simulation results that we used during the design.

An important difference is however visible when we look at Figure 30, which shows the state variables that we used to define our linear operating point. Although we did assume that they are constant, they obviously are not. We are dealing with a nonlinear

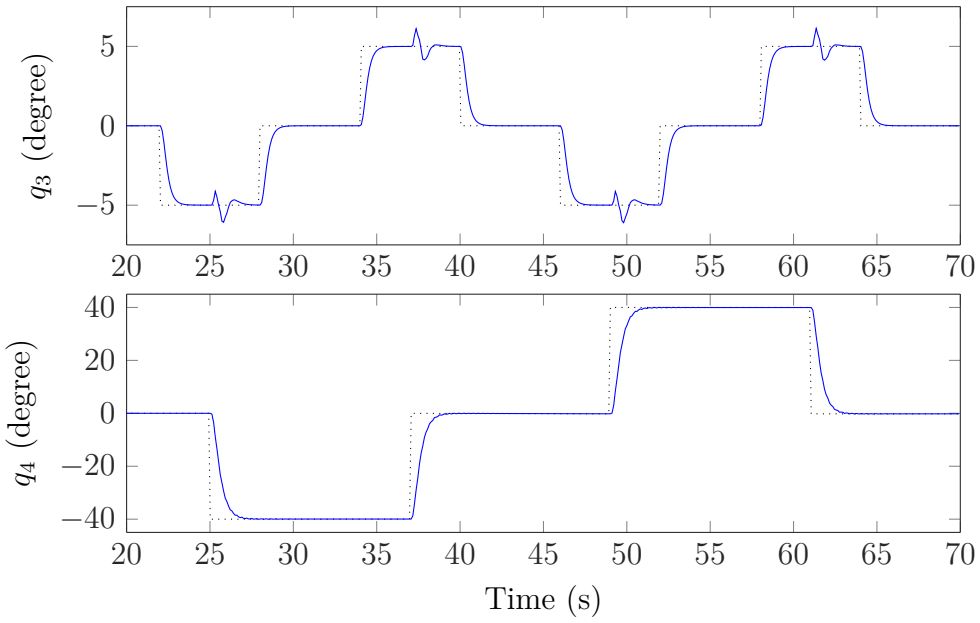


Figure 28: Typical trajectory for tracking in nonlinear simulation with two-degrees-of-freedom controller.

plant here, and it should not surprise us that these states are affected by what we are doing to the others. As discussed in Section 1.1, we have to make sure that these states stay "close to" the operating point. If you experience an unstable closed-loop in simulation, it is most likely due to being too far away from the operating point. An approach that promises relief from these problems is to adjust the controller to the current operating point online and therefore to make sure that the system is always "right on" the operating point, which of course is even better than just "close to". This approach is called *gain-scheduling* and is treated in detail in the lecture course *Advanced Topics in Control* and the corresponding *Control Lab* experiment *Gain-Scheduled Control of a Gyroscope*.

A second problem with our controller are cross-couplings that will show up in the nonlinear simulation although they were not present in the linear simulations used for design. Our controller does simply "not know" that there are cross-couplings since the model that we used for design did not contain them. It should thus not surprise us, that the nonlinear simulation reveals this weakness. In fact, this weakness can easily become the limiting factor in the design since large cross-couplings are likely to drive the system far from its operating point which then might cause instability. Again, gain-scheduling is

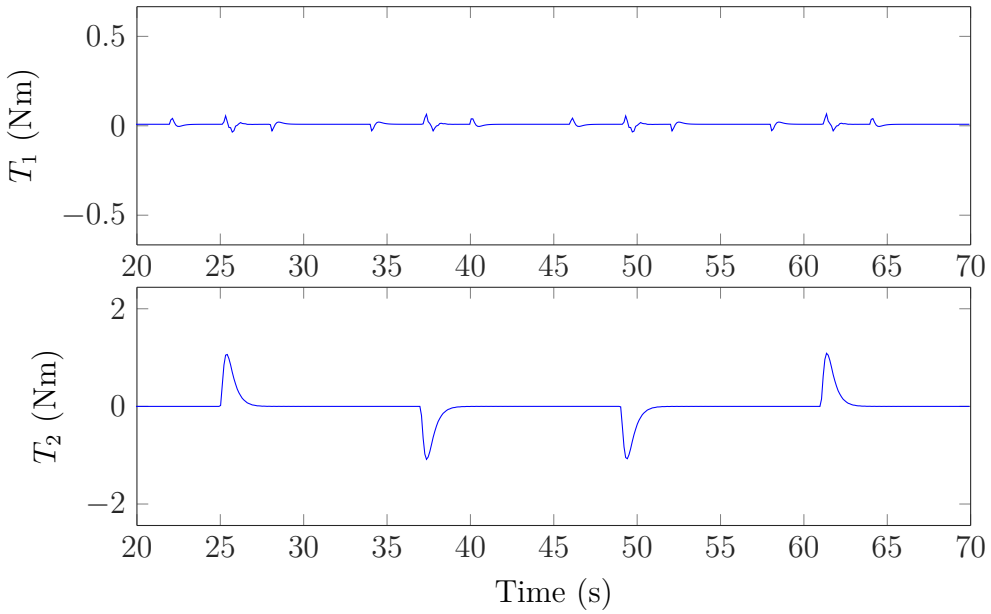


Figure 29: Typical input signal in nonlinear simulation for tracking with two-degrees-of-freedom controller.

an appropriate tool to counter this effect and you are welcome to experience this on your own in the *Control Lab* experiment *Gain-Scheduled Control of a Gyroscope*.

Prep. 5.2: Make sure that your controller works in nonlinear simulation and delivers consistent results with what you achieved during the design phase. Compare the single-degree-of-freedom controller with the two-degrees-of-freedom controller with respect to the cross-coupling problem. Pay close attention to the available control action since saturation might easily cause instability on this plant. Also try out your S/KS design, although you are likely to produce an unstable closed-loop. Think about why this happens even though the robustness margins were larger for the S/KS than for the four-block design.

6 EXPERIMENTAL SESSION

6.1 Advanced two-degrees-of-freedom design

As part of the design task *ORC2—Robust Control of a Gyroscope* of the *Control Lab*, different concepts for the design of an output feedback controller for the gyroscope were

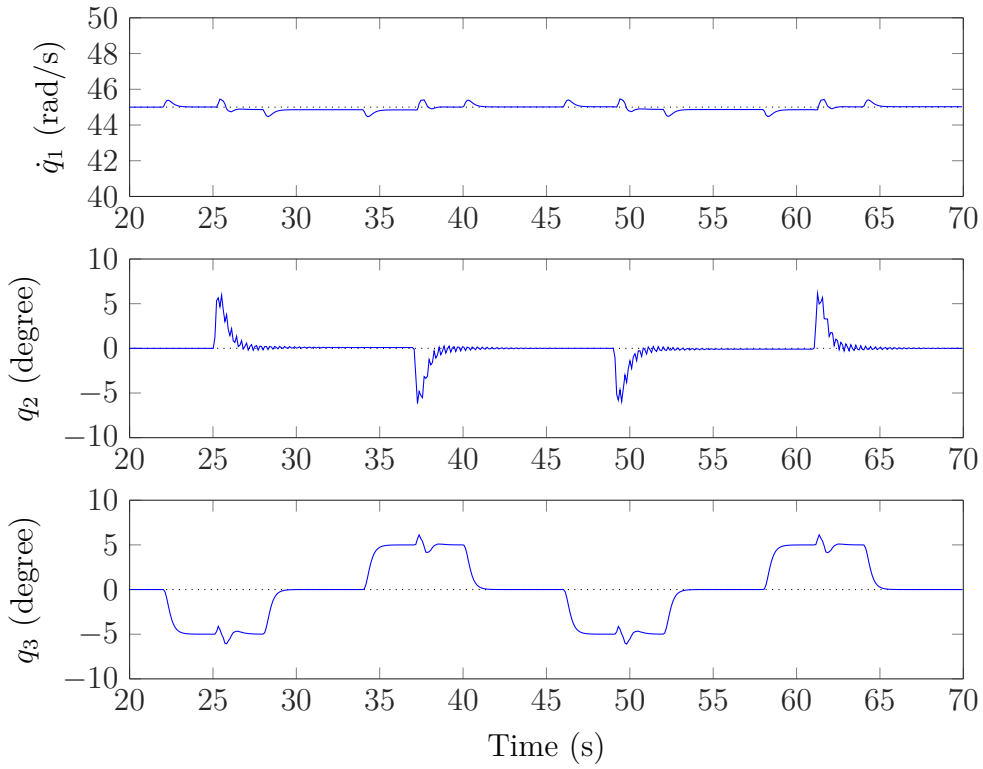


Figure 30: Typical trajectory of the operating point in nonlinear simulation with two-degrees-of-freedom controller.

examined. So far, only the angles q_3 and q_4 were considered as output variables in the feedback loop. However, all angles can be measured and the angular velocities can be determined by using first order differentiation filters. This raises the question of whether the controller performance can be significantly improved by taking additional output variables into account.

Note that for instance $\dot{q}_2$ is not visible for the controller so far, so it does not contribute to an improved damping though this would be desirable on the actual plant. Further note that the model inaccuracy due to neglecting the differentiation filters in the model is addressed in the controller design, since we assume measurement noise on all channels. This is an important advantage over directly using state feedback. From a theoretical point of view, not even the slightest disturbance in the states would be allowed for state feedback.

In the following, your task is to design an advanced two-degrees-of-freedom controller using the full state vector $x = [q_3 \; q_4 \; \dot{q}_2 \; \dot{q}_3 \; \dot{q}_4]^{\mathrm{T}}$ as output variable y for feedback. We

want to track the first two outputs, but simultaneously reduce disturbances on the three other outputs, i.e. improve damping. Tracking of more than two outputs is no feasible control objective because the actual plant only has two inputs. So, the system would be underactuated.

As we are considering five output variables now, the shaping filter W_S has to be of dimension five. As before, the first two channels should be weighted with integral behavior in order to achieve good reference tracking. All other channels are only provided to improve damping and should therefore be chosen as constants. Note that the reference only acts on the first two output variables. So the input to W_S should look something like

```
input_to_W_S = ['[w1(1:2)-G(1:2)+r ; w1(3:5)-G(3:5)]'];
```

How to Proceed

Design an advanced two-degrees-of-freedom controller for the gyroscope. Use the full state vector $x = [q_3\ q_4\ \dot{q}_2\ \dot{q}_3\ \dot{q}_4]^T$ as output variable y for feedback. Again, repeat the design process a few times and iteratively adjust the weights to come up with a promising design in linear evaluation. Compare the advanced two-degrees-of-freedom design to the two-degrees-of-freedom design of Section 4 to work out why using the velocities in the feedback loop is useful. Verify your controller in nonlinear simulation and try to come up with the design requirements given in Section 4. For nonlinear simulation, the file `orc2_GyroSimulation2.slx` will be provided during the lab. This file contains the advanced two-degrees-of-freedom design, otherwise it looks pretty much the same as the file `orc2_GyroSimulation.slx`.

6.2 Experiment

During the lab, the file `orc2_GyroExperiment.slx` will be provided to interface the control moment gyroscope. Its structure is depicted in Figure 31. 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 robust 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 the output variables in order to define an operating point.

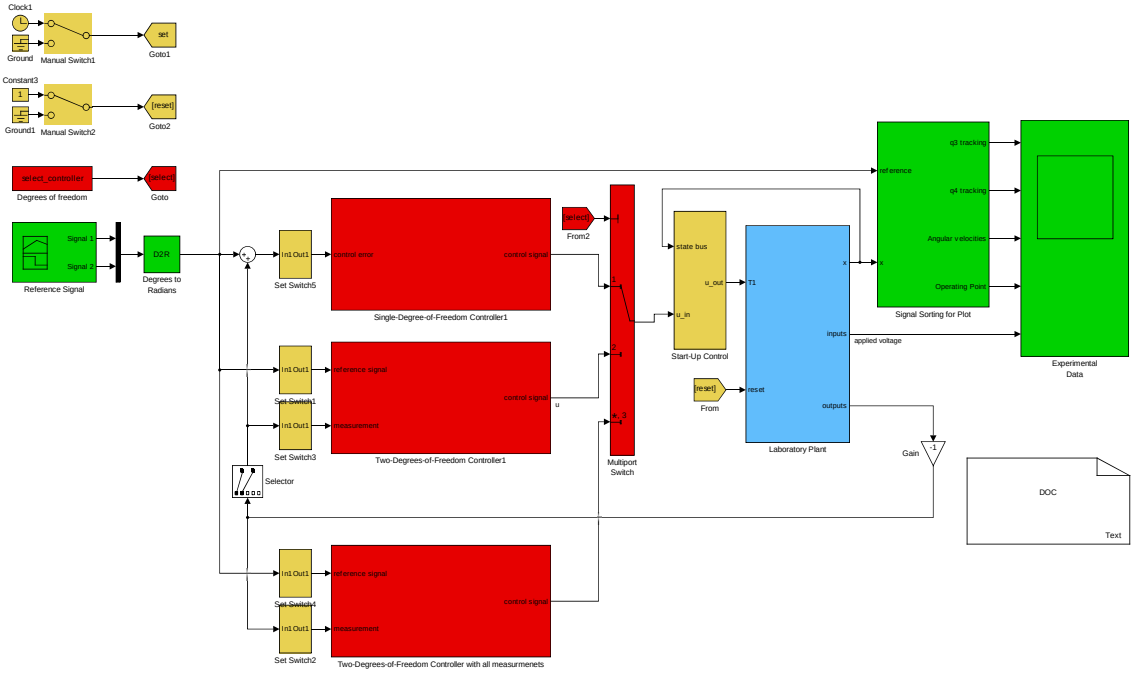


Figure 31: Simulink model: Plant, Controller, Start-Up Controller and interface-related blocks

Prep. 6.1: Make sure your design file produces all the files necessary to run the nonlinear simulation by simply executing it.

! Bring this file 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 5° steps in q_3 and 40° steps in q_4, you should be able to achieve

- stability throughout the runtime
- a 5% settling time of less than 2 seconds on both channels
- a steady state error of less than 1% on both channels
- cross coupling should not exceed 1° per 5° command (i. e. should be less than 20%)
- an overshoot of less than 10%
- 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. Also, compare your experimental results to the simulation results that you have obtained.

! 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.