Robust Control [SC42145]

Practical Assignment: Control Design for a Floating Wind Turbine

Delft Center for Systems and Control
Delft University of Technology

November 20, 2023

Introduction

This document gives a description of the requirements associated with the practical assignment for the *Robust Control* course [SC42145]. In order to complete the assignment you need to submit 2 reports, one on Part 1 of the assignment and one on Part 2 of the assignment. The deadlines are given on Brightspace as well as in this document at the respective parts of the assignment. Groups of a maximum of two students are allowed for completing this assignment. You should register your group on Brightspace. Each of the reports has to be submitted together with the necessary MATLAB files (please create a RAR/ZIP archive) via Brightspace as a group submission before the deadline given on Brightspace.

Outline

In this assignment, you will investigate the challenges and limitations in designing a control system for a floating wind turbine, see Figure 1 for normal operation. In this assignment, you will be asked to design several controllers

and evaluate them in the frequency and time domain. On Brightspace you will find a MATLAB file which contains the state-space model of a floating wind turbine. Within this MATLAB file, you will also find 10 minutes of wind data, this is needed for the exercises in Section 2.1. A full description of the operational parameters of the system is given in Table 1.

Rated Power	1.8 $[MW]$
Rated Torque	10^4 $[Nm]$
Rated Speed	180 $[rad/s]$

Table 1: Operational parameters for the FWT

Figure 1: An example of a Floating Wind Turbine (FWT)

Model Description

The linear state-space model has been derived using a first principles modeling procedure and it corresponds to a specific linearisation point. The parameter with respect to which the linearisation is performed is the wind

speed ($V_{lin} = 14$ $[m/s]$). We refer to [3] for a more detailed model description. The model has three inputs (namely the blade pitch angle β $[rad]$, the generator torque τ_e $[Nm]$ and the wind speed V $[m/s]$), of which only the first two are control inputs. The third input will be used as a disturbance input around the linearization point. The model also has two output channels which correspond to a specific choice of measured quantities in this floating wind turbine, namely the generator speed ω_r $[rad/s]$ and the fore-aft tower top displacement z $[m]$. In the provided MATLAB model the internal states employed to capture the dynamics of the system are $x_1 = \omega_r$, $x_2 = \dot{z}_1$, $x_3 = z_1$, $x_4 = \dot{z}_2$, $x_5 = z_2$, where $z_{1,2}$ are a decomposition of the fore-aft tower displacement, corresponding to the tower bending and platform tilting modes respectively. Figure 2 displays a schematic view of fore-aft movement for a floating wind turbine. The inputs of the MATLAB model are ordered as $u_1 = \beta$, $u_2 = \tau_e$, $u_3 = d = V$.

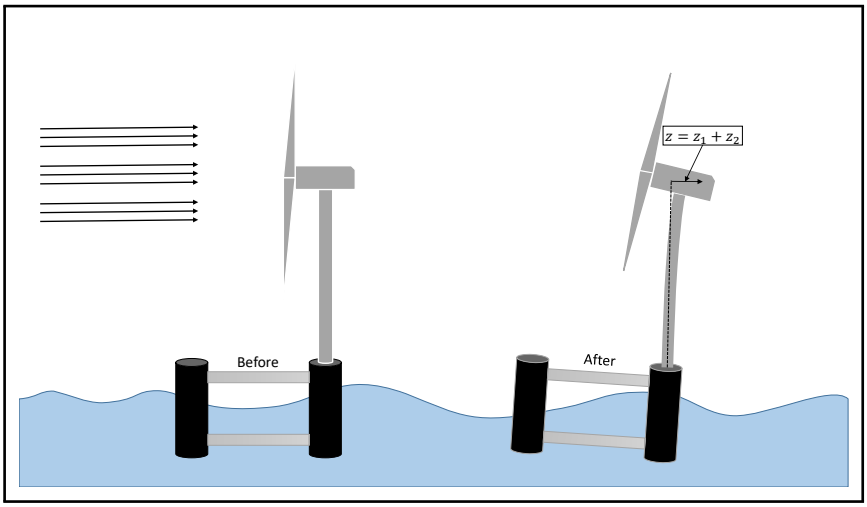

Figure 2: Schematic depiction of fore-aft movement.

The regulation problem when the rated power of 1.8 MW is produced deals with keeping the output power $P_o = \tau_e \cdot \omega_r$ constant by pitching the blades. We are also interested in scenarios where the power demand changes and we use the blade pitching control input to drive the generator to a new operating point. More explicitly [4], we can pitch the turbine blades in order to control the rotor speed and thus regulate power without changing generator torque. In this assignment, you will design three different controllers all required to achieve the same performance goals.

Part 1 SISO Analysis and Control Design

Due: 15-Dec-2023

For the first part of this assignment, you will design a SISO controller for the floating wind turbine. As the wind turbine is in operation the power demand is increased and a new rotational velocity target (ω_r) for the generator is set. By only designing a SISO controller for the blade pitch input we are guaranteed the generator torque remains constant. The goal of the SISO controller is to pitch the blades to increase the rotational velocity of the turbine and acquire the desired generator speed. To this end, your task is to design a reference tracking controller that achieves the highest possible bandwidth whilst remaining within the design requirements:

1. Small settling-time.
2. Overshoot $< 1\%$.
3. No steady state error.

Your report should cover the design process you went through and explain how you arrived at your controller.*Important: When making the SISO transfer function from blade pitch β to generator speed ω_r, multiply this transfer function with -1. This will make control design significantly easier. If done correctly, the bode plot of the transfer function should start at* 360° *phase.* It is easy to get lost in endlessly tuning a controller. To ensure enough time for the rest of the assignment, be sure to not spend more than 1 week on the SISO design. The following steps will help you on your way to your control design.

1. Take a look at the open-loop Bode plot and the pole-zero map of the SISO system. Are there any features of the system that might hinder achieving a high bandwidth, and if so why do they make it difficult?
2. For the SISO controller we have formulated 3 design requirements. These time domain requirements can be translated into frequency domain requirements. For example, there exists a relation between overshoot on a step response and the phase margin of the loop gain. For each of the time-domain requirements, explain what parameters or metrics you can look at in the frequency domain to help you reach these design requirements before doing time-domain simulations.

3. Once you have achieved a satisfactory design perform time-domain simulations of your closed-loop system (please see a list of useful Matlab commands at the end of this document, you can also use Simulink if you see fit) and test the performance of your SISO controller by having it track a reference signal. For the reference input, you can use a step input. Plot and evaluate the simulation results and highlight your achieved performance specifications (bandwidth, phase margin, settling time etc.). The Matlab command `minreal()` can be useful to increase simulation performance.

4. Consider that we use your SISO controller for rejecting output disturbance. Without changing the controller, perform the same time-domain simulations as in the previous question. For this simulation apply the disturbance step input on the third input channel of the state space. Would you change your controller for the disturbance rejection scenario?

Multi variable Mixed-Sensitivity

Now we consider both channels as control inputs: (β and τ_e), and both outputs of the system: ω_r and z. Even though you can now use τ_e as a second controllable input, we want to keep the power production $P_o = \tau_e \cdot \omega_r$ unchanged. Therefore, this control input is not allowed to have a static (steady-state) contribution (i.e. a unit step change in the wind disturbance channel should still result in $\lim_{t\to\infty} \tau_e(t) \to 0$). Remember that the linearized model operates at the settings given in Table 1.

You are now asked to design a Mixed-Sensitivity centralized controller [1, pages 94-96]. You should **perform and explain** the following exercises:

1. Compute the RGA for $\omega = 0$ and $\omega = 0.3 \times 2 \times \pi$. What can you conclude?

2. Compute the MIMO poles and zeros. What can you conclude?

3. Design an appropriate $W_{p_{11}}$ (see Table 2), for a cut-off frequency of 0.3 Hz, to get an attenuation of low-frequency disturbances of 10^{-4}, and an $\mathcal{H}_\infty$ norm of the sensitivity function of 3.

4. Draw a block diagram with the model of the floating wind turbine and controller and include the performance weights W_p and W_u (see Table 2) in this block diagram. Define all signals that are relevant for the mixed sensitivity problem (use negative feedback). **Define the performance signals z_1 and z_2 as the signals coming out of W_p and W_u respectively**

W_p	$\begin{bmatrix} W_{p11} & 0 \\ 0 & 0.2 \end{bmatrix}$
W_u	$\begin{bmatrix} 0.01 & 0 \\ 0 & \frac{5 \cdot 10^{-3} s^2 + 7 \cdot 10^{-4} s + 5 \cdot 10^{-5}}{s^2 + 14 \cdot 10^{-4} s + 10^{-6}} \end{bmatrix}$

Table 2: Performance weights

5. Derive a generalized plant of the previously designed block diagram and analyse it, i.e. assign the variables and give the mathematical description. You can use Matlab, to check if the number of states is as expected. When using Matlab make sure to use the command `minreal()` to get a minimal realization.

6. Give an interpretation of the given performance (both W_p and W_u) weights. Try to relate this to the control design targets explained in the introductory text of this part.

7. Using the generalized plant you found, design a mixed-sensitivity generalized controller (hint: use `hinfsyn` in MATLAB). Is the closed-loop system internally stable while using this controller (look at the generalized Nyquist plot)? How many states does the controller have? and the generalized plant?

8. Perform the same time-domain (reference tracking and disturbance rejection) simulations as you did with your self-designed SISO controller and compare their performance. Comment on the differences in time response for the two different controllers.

Fixed Structure Control Design

In the previous section, we have designed an optimal $\mathcal{H}_\infty$ controller. This controller has good performance but in the real world might be difficult to implement. In this subsection, we will cover a control design technique based on the $\mathcal{H}_\infty$ framework but where we can design our own controller structure. This technique is called fixed structure control design, see also: (`https://nl.mathworks.com/help/robust/ref/dynamicsystem.hinfstruct.html`).

To see how this control design technique can help with designing viable, yet optimal, controllers consider the following steps: First, we start with what we could consider a bit overkill, but we are going to redesign the reference tracking SISO controller from the first part using this method. This will also help familiarize ourselves with the MATLAB toolbox. As this is a $\mathcal{H}_\infty$ control design method we will need to define a weight on our sensitivity function for the controller optimization. For the SISO controller, you can take the performance weight as:

$$W_{p_{siso}} = 0.95 \cdot \frac{s + 0.04\pi}{0.016\pi + s}. \tag{1}$$

To help you along do the following exercises:

1. Draw the SISO block diagram with $W_{p_{siso}}$ as performance weight.

2. Set up the plant to synthesize a fixed structure controller. An example of how this can be done in MATLAB for a simple PID controller is given at the back of this document.

3. Make a choice for the type of tunable controller, feel free to add (or remove) elements to the example code, and synthesize your controller. It is no problem if the peak gain given as output by the `hinfstruct` command is larger than 1.

4. Simulate the reference tracking performance of your optimal controller and compare it to your manual PID controller. Comment on the differences between the two.

Now we are going to design structured controllers for the $\mathcal{H}_\infty$ MIMO problem. For this part, you can use the weights W_p and W_u as previously defined. The following steps will help you design your own MIMO optimal controller

1. First define a new MIMO generalized plant that incorporates the to-be-designed fixed MIMO controller.

2. Choose what kind of parametric MIMO controller you want to design.

3. Synthesize your new controllers with the given weights. As with the SISO design, it might be difficult to achieve $||\mathcal{H}_\infty|| < 1$ with your design. Testing the closed-loop system in the time domain can provide insight into its performance compared to using the full $\mathcal{H}_\infty$ controller.

4. Once you have settled on a design compare the sensitivity and complementary sensitivity functions for the full $\mathcal{H}_\infty$ system and your fixed structure controllers and comment on any differences.

5. Perform the reference tracking and disturbance rejection tasks as in the previous section and compare with the $\mathcal{H}_\infty$ results.

6. Based on your findings would you prefer to use a $\mathcal{H}_\infty$ controller or use the fixed-structure controller? Explain the advantages and disadvantages of the two methods.

Part 2 Robust Analysis and Controller Design

Due: 19-Jan-2024

In this second part of the assignment, we start from the given (nominal) model and derive a perturbed (uncertain) model, followed by an analysis of the robustness associated with the previous controller design. The perturbations are characterised by Δ which belongs to a certain set, i.e. $\Delta \in \underline{\Delta}$. Including this uncertainty in the model results in a (infinite) set of models describing the behaviour of the floating wind turbine. Stability and performance analysis should hence be done for a set of models and not for one single (nominal) mode, like you have done in the first computer session. We call stability and performance analysis of a set of models analysis of the Robust Stability (RS) and analysis of the Robust Performance (RP) respectively.

In this third part you will use the Robust Control Toolbox in MATLAB to perform a robustness analysis on an uncertain model of the floating wind turbine when using the Mixed-Sensitivity controller you designed in the first computer session.

2.1 Analysis of Robust Stability (RS) and Robust Performance (RP)

In this part of the assignment we are first going to add input and output multiplicative uncertainty to the nominal model describing the dynamical behaviour between the inputs β and τ_e and the outputs ω_r and z. You will use the numerical uncertainty weights given in Table 3. We define the weight on the input uncertainty as $W_i = \texttt{diag}(W_{i_1}, W_{i_2})$ and the weight on the output uncertainty as $W_o = \texttt{diag}(W_{o_1}, W_{o_2})$. Their corresponding perturbation blocks are $\Delta_i = \texttt{diag}(\delta_{i_1}, \delta_{i_2})$ and $\Delta_o = \texttt{diag}(\delta_{o_1}, \delta_{o_2})$ respectively. Furthermore, we have that $\delta_{i_1}, \delta_{i_2}, \delta_{o_1}, \delta_{o_2}$ are all complex scalars and their absolute value is bounded by 1. Define the uncertainty block $\Delta = \texttt{diag}(\Delta_i, \Delta_o)$. You should **perform and explain** the following steps:

1. Draw a block diagram with the model of the floating wind turbine and controller and include the performance weights W_p and W_u (Table 2) and uncertainty weights W_i and W_o (Table 3) in this block diagram. Define all signals and use negative feedback.

2. Derive a mathematical expression for the new generalised plant with performance and uncertainty weights as given in Table 2 and Table 3 respectively [1, section 8.3].

3. Plot the frequency responses of the uncertainty weights and interpret them accordingly. Use [1, section 7.2, 7.4, 8.2] to see how these weights might have been derived and explain your findings. In addition, plot the singular values of the uncertain transfer matrix describing the dynamic behaviour between the inputs β and τ_e and the outputs ω_r and z. You can use `ultidyn` in Matlab to create the perturbation blocks.

4. Use the uncertainty and performance weights with numerical values as given in Table 3 and 2 respectively and program the generalized plant in MATLAB (hint: you can augment the generalized plant code from the nominal design).

5. Use the mixed-sensitivity controller found in the first computer session and the formulated generalized plant found in this computer session and check for nominal stability (NS) using the generalized Nyquist criterion, nominal performance (NP), robust stability (RS) and robust performance (RP) using the Structured Singular Value (SSV). See [1, sections 8.5-8.10] for the necessary background information. For the NP, RS and RP tests, you should make corresponding statements about the obtained values for μ.

6. How does the value of μ for NP relate to the $\mathcal{H}_\infty$ norm of the objective function in the mixed-sensitivity design? Write down your results and explain them.

2.2 Robust Controller Design

In this final section of the assignment you are asked to use the same settings as in the previous one and to synthesize a controller that achieves robust stability (RS) and robust performance (RP). For this use the D-K iteration procedure as shown in [1, section 8.12]. The MATLAB script from the corresponding lecture slides [2] is also useful for implementing this procedure. You can use `musyn` in MATLAB. We expect the following items in your report

$W_{i_1} = \dfrac{\frac{1}{16\pi}s + 0.3}{\frac{1}{64\pi}s + 1}$	$W_{i_2} = \dfrac{\frac{1}{16\pi}s + 0.3}{\frac{1}{64\pi}s + 1}$
$W_{o_1} = \dfrac{0.05s + 0.2}{0.01s + 1}$	$W_{o_2} = \dfrac{0.05s + 0.2}{0.01s + 1}$

Table 3: Uncertainty weights

1. Give and explain the symbolic expression for the D matrices used in the DK-synthesis for this particular problem.

2. Design a robust controller using DK-iterations. Once synthesized, check for NS, NP, RS and RP using the structured singular values (SSV).

3. Perform the same time-domain simulation scenarios again with the synthesized controller and compare the results with the previously designed MIMO controllers. Explain the differences you see in the results using relevant frequency-domain plots.

lsim	simulate time response of dynamic system to arbitrary inputs
minreal	minimal realization or pole-zero cancellation in a linear system
lft	generalized feedback interconnection of two models
series	series interconnection for two models
sysic	create interconnection structure for certain and uncertain matrices/systems
append	group models by appending their inputs and outputs
norm	norm of linear model
ultidyn	create uncertain LTI object
hinfsyn	compute H_∞-optimal controller for LTI plant
mussv	calculate upper and lower bounds for the structured singular value of a block
fitmagfrd	fit frequency response magnitude data
mussvextract	extract *muinfo* structure returned by *mussv*
frd	create frequency response data model
fnorm	pointwise peak gain of FRD model

Table 4: Recommended MATLAB functions

Fixed Structure Code

```
1
2  %% Fixed Structure SISO controller
3  Kp = realp('Kp',1);
4  Ki = realp('Ki',1);
5  Kd = realp('Kd',1);
6  Tf = realp('Tf',1);
7
8  Wp_simple = 0.95*(s+0.02*2*pi)/(0.0001*2*pi+s);
9  C_struct = Kp+Ki/s+(Kd*s)/(Tf*s+1);
10
11 % SISO hinfstruct
12
13 Wp_siso = Wp_simple;
14 Wp_siso.u = 'y_act';
15 Wp_siso.y = 'z1';
16
17 G_siso = -1*Gsiso;
18 G_siso.u = 'u';
19 G_siso.y = 'y_plant';
20
21 C_siso = C_struct;
22 C_siso.u = 'e';
23 C_siso.y = 'u';
24
25 Sum1 = sumblk('e = w_ref-y_act');
26 Sum2 = sumblk('y_act = y_plant+y_dist');
27 Siso_Con = connect(G_siso,Wp_siso,C_siso,Sum1,Sum2,{
      'w_ref','y_dist'},{'y_plant','z1'});
28
29 opt = hinfstructOptions('Display','final','
      RandomStart',5);
30 N_siso = hinfstruct(Siso_Con,opt);
31
32 % Extract controller gains:
33 Kp_opt = N_siso.Blocks.Kp.Value;
```

```
Ki_opt = N_siso.Blocks.Ki.Value;
Kd_opt = N_siso.Blocks.Kd.Value;
Tf_opt = N_siso.Blocks.Tf.Value;

Kfb_opt = Kp_opt+Ki_opt/s+(Kd_opt*s)/(Tf_opt*s+1);
```

References

[1] S. Skogestad and I. Postlethwaite - *Multivariable Feedback Control: Analysis and Design*, John Wiley and Sons, Chichester, England, 2nd Edition, 2005.

[2] J.W. van Wingerden - *Lecture Slides on Robust and Multivariable Control [SC4015]*, Technische Universiteit Delft, 2015.

[3] G.J. van der Veen, I.A. Couchman and R.O. Bowyer - *Control of Floating Wind Turbines*, American Control Conference, Montreal, Canada, 2012.

[4] J.M. Jonkman - *Influence of Control on the Pitch Damping of a Floating Wind Turbine*, ASME Wind Energy Symposium, Reno, Nevada, U.S.A., 2008.