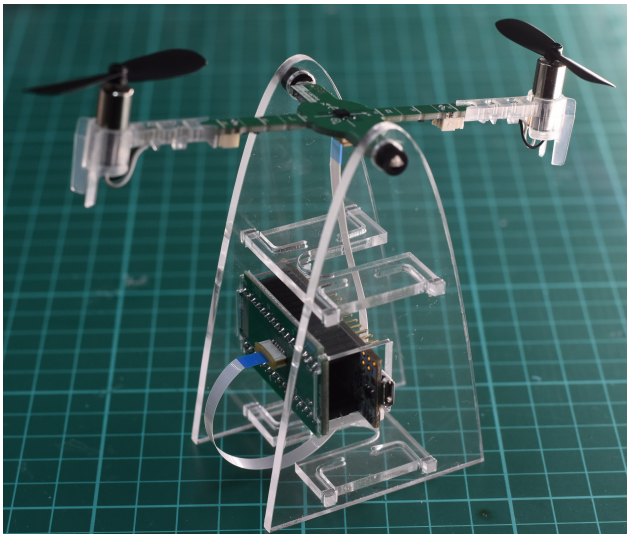
Control Systems I: Seesaw project description

Seesaw - Designing a controller

-



Julian Zilly
Prof. Dr. Emilio Frazzoli
Institute for Dynamic Systems and Control
Swiss Federal Institute of Technology Zurich (ETH)

2017

Institute for Dynamic Systems and Control
IDSC
Institut für Dynamische Systeme und Regelungstechnik

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Project description

This project is meant to showcase your understanding of Control Systems I and your ability to put this knowledge into action. In the following you will be asked to work on three tasks.

- To document your work, you will have to submit a report no longer than 3 pages length. including figures, using the template that is provided to you. You will write your report within the *report.tex* file.
- Please change the names titled "Student name" and the student number(s) XX-XXX-XXX to your own.
- Please also replace the email address proxy with your own and your teammates name. You may work either alone or in groups of 2.
- Please note that "most" supported environment to do this project in is MAC and LINUX. We will offer a Windows solution as well however.

Please send the completed project reports to `jzilly@ethz.ch` with the subject line **[Seesaw-Project]** (with the brackets). Submissions without the report template or correct subject line will not be considered. The grading bonus will be given on a pass/fail basis.

The purpose of writing this report is both to showcase the work that you have done with the Seesaw as well as to introduce you to writing proper reports in Latex (download `https://www.texstudio.org/`) which you yourself can still look up years later.

Setup

- **Seesaw code:** In the *Seesaw* Github folder (`https://github.com/idsc-frazzoli/Seesaw`), we uploaded a "SeesawProject" folder within which most files necessary for the project can be found. Update your existing Seesaw code by redownloading the Seesaw code or by typing "git stash", "git pull" in your terminal while you are in the Seesaw folder. For the following three tasks, you are provided a template file for each task. These are entitled `tControl_task1.c`, `tControl_task2.c`, and `tControl_task3.c`, respectively. Please copy and overwrite the current `tControl.c` file with the respective task file. These files provide you with an interface and compute the reference signal. For your work to be comparable, do not change the reference signal. In the `tControl.c` file within which you have substituted the task code, the controller is also used to calculate the input u based on either the error e or the measurements *dAnglePos* and the reference r. By default, a PID-controller is selected. You may venture and use other types of controllers for task 1 however.

- **Seereceive code:** In the *seereceive* Github folder (`https://github.com/idsc-frazzoli/seereceive`), we uploaded code to be able to read out data from the Seesaw. This will be necessary to show the work you have done in the project report.

  If you are using Windows, first click on the gray bar on the left and select the branch "Windows". Download the code from the repository by clicking the green button on the top right and downloading the .zip-file. Unzip the file.

  Create a port.properties file in this folder. You can do this using a text editor. Please

make sure that the ending of the file is **not** .txt but .properties. On Windows it should not be indicated as a .txt file even if the name is "port.properties".

In the port.properties file, we need to communicate on which port the Seesaw is connected to our computer. To find out on which port you are connected the following operating specific steps should be taken.

MAC: Open a terminal. Type "cd /dev". Type "ls". These are the ports to your computer. Plug in the Seesaw and note which port appears that was not there before. Open the port.properties file and write in this port. This should be something similar to: "port=/dev/tty.usbmodem1423" (without exclamation marks)

LINUX: Open a terminal. Type "cd /dev". Type "ls". These are the ports to your computer. Plug in the Seesaw and note which port appears that was not there before. Open the port.properties file and write in this port. This should be something similar to: "port=/dev/ttyACM0" (without exclamation marks)

WINDOWS: Go to "Devices". Locate the Seesaw and find on which port it is connected. Open the port.properties file and write in this port. This should be something similar to: "port=COM3" (without exclamation marks)

Running *seereceive*:

Note you may have to install the JAVA development JDK (`http://www.oracle.com/technetwork/java/javase/downloads/jdk9-downloads-3848520.html`) to run the seereceive-file. Open a terminal, navigate to your *seereceive* folder. Then type "`java -jar seesaw.jar`". If you are using Windows, type "`java -jar seesaw_faze.jar`" instead. If you do not have this file, please go back and download the *seereceive* folder from the "Windows" branch.

```
=====
ETH
t = 334.59000000000003[s]
r = 0.0[ticks]
y = 1532.0[ticks]
u = 0.0[PWM]
numReceived = 3889
=====
```

should appear. After 3000 steps counted in "numReceived" the data is saved in your home folder in a folder called "seesawState". Keep this folder. If it already exists a folder called "seesawState (copy)" will be created (and so forth).

To correctly interface with your Matlab plotting scripts, change the names of the .m file in your seesawState folder the following way.

- **Task 1:** Rename the `seesawState.m` file to `seesawState_task1.m` .
- **Task 2:** Rename the `seesawState.m` file to `seesawState_task2.m` .
- **Task 3 - no ARW:** Rename the `seesawState.m` file to `seesawState_task3_no_ARW.m` .
- **Task 3 - ARW:** Rename the `seesawState.m` file to `seesawState_task3_ARW.m` .

Keep all these files in your original "seesawState" folder. This folder should be automatically added to your Matlab path in the `seesaw_data_readout.m` file.

- **Making plots and computing performance metrics:** There is a `seesaw_data_readout.m` file in the

`SeesawProject/Matlab_plotting` folder within the Seesaw Github repository. This file can be used to plot your results in Matlab. Within this script you should find the places highlighted with "CHANGE HERE" and change the existing folder names with the ones on your computer. If you do not move the template, you may not have to change anything here. This way the plots of the script will be directly saved into your project reports "figures" folder.

- **Running your Latex files:** You should also have access to the project report template in Latex which is in the `SeesawProject/Seesaw_report_template` folder. To be able to successfully edit and compile the .tex file you need to install a Latex compiler as mentioned in the beginning of this document. With this at hand, you should be able to write the report. Plots, if named correctly and if in the correct "figure"-folder, should automatically be integrated into your report template. You may, of course, manually copy the .pdf-files of the plots into your "figures"-folder of the report template.

Seesaw model

The Seesaw model we already published in the exercises is the following

$$Y(s) = \beta \frac{\omega_0^2}{s^2 + 2\delta\omega_0 s + \omega_0^2} U(s)$$

$$\omega_0 = 2.01 \qquad \delta = 0.93 \qquad \beta = 407950$$

It may help you in designing adequate controllers for your Seesaw in the following tasks. Please note that the values may differ for your individual Seesaw. The general behavior should however be quite similar.

Task 1: Trajectory following

For this task we provide you with a reference signal $r = 6000 \cdot \sin(t/period)$. Please see the template `tControl_task1.c` provided in the Seesaw project folder in the Seesaw Github repository and replace the current `tControl.c` file with it. Rename the substituted file `tControl_task1.c` to the previous name `tControl.c`.

Tune your controller (of your choosing, but as a default we consider a PID controller) to fulfill the following task.

a) Given the reference signal, plot the reference and measurements of and inputs to your Seesaw in the provided figure slots in your report template. Briefly describe the behavior you see. State what your error value is in the provided slot of the report template.

   To successfully pass the baseline your error value should be below

$$\text{Error}_{\text{Task 1}} < 60$$

   In our own experiments we reached a value of 31.2.

b) What algorithmic and/or hardware changes might improve the result further?

Task 2: Building a controller up to specifications

a) The new template script to complete is `tControl_task2.c` and can be found in the

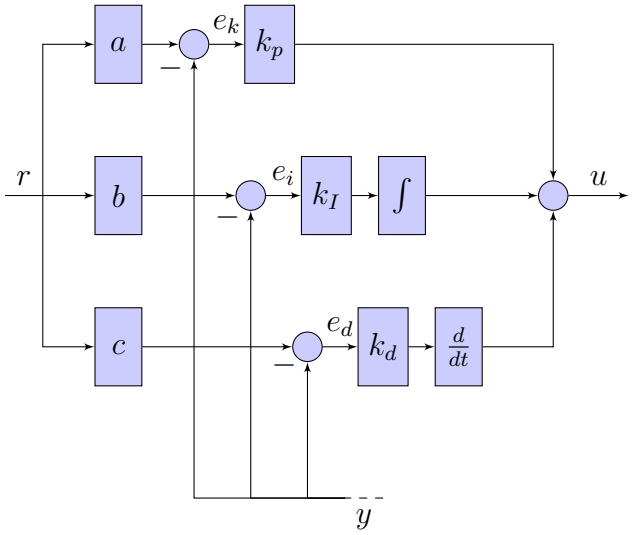


Figure 1: Illustration of multiplication factors a, , b, c used to scale the reference signal.

SeesawProject-folder. Please again substitute `tControl.c` with this code. In this task, we would like you to design a controller that tracks **step responses** well. You may again use the PID template code from the previous task and exercises of the course. Describe the controller behavior you see. What may be an issue with large jumps in reference values?

b) You may have noticed that tuning a controller for a non-negligible step response can be difficult. You should now design a PID controller which has setpoint weights a, b, c multiplied to the reference before computing the error as depicted in figure 1. Augment the `tControlPID.c` file to allow for this rescaling. Alternatively, you could create new files `tControlPIDabc.c` and `tControlPIDabc.h` to write your code and also substitute the controller in your `tControl.c` file.

Please implement this in your code. Run your code, record and plot your results with the provided template. Again describe the behavior of your Seesaw.

You should reach values better than

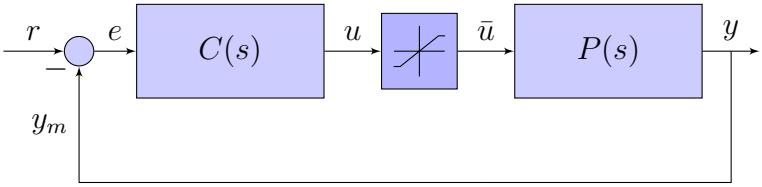
$$\begin{aligned} \text{Maximal overshoot } M_p &< 60\% \\ \text{Rise time } T_{90} &< 1.6\,s \\ \text{Error}_{\text{Task 2}} &< 40 \end{aligned}$$

In our experiments we obtained a maximal overshoot $M_{p-max} = 21.1\%$, and a rise time $T_{90} = 1.09\,s$ and an error of $\text{Error}_{\text{Task 2}} = 27.1$.

c) Why is it useful to have a weighting factor for the reference signal? Without r-term for derivative - qualitatively explain why this may help.

Task 3: PID with anti-reset windup

You might have already noticed that when you hold down the Seesaw on one side for a while, it will reacts strongly into the other directions once released if an I-part is present in the controller. This is due to prolonged integration of errors in the I-part of the PI/PID controller. A method to address this issue is an anti-reset windup (ARW) scheme. The new template script to complete is `tControl_task3.c` and can be found in the *SeesawProject*-folder. Please again substitute `tControl.c` with this code.



a) Implement an anti-reset windup (ARW) scheme for a PID-controller onto your Seesaw. To do this augment your `tControlPID.c` file. Describe how an ARW scheme works. *Hint: Reusing the setpoint weight code from exercise 2 may be helpful.*

b) Show the behavior of your system to a prolonged error accumulation **with** and **without** your ARW-scheme in the figure specified in your report template. Pick a sufficiently large gain for the integrator to demonstrate the effect with and without ARW. You may have to stop the Seesaw **without** ARW from going back and forth to violently with your hands from time to time. Describe the behavior you see.

c) Demonstrate the behavior of the Seesaw with ARW to your TA in the last exercise class. Alternatively we will also arrange times in beginning/mid January to demonstrate your work. Another method will be to send your report together with a short video of the Seesaw behavior with and without ARW. We would like to observe the ARW-effect qualitatively both in real life as well as in your plots. The reference and input behavior in `tControl_task3.c` should provide a realistic scenario to test the effect. You may also push the Seesaw down manually to realize a similar "integration of errors" effect.