DEPARTMENT OF PHYSICS
Institute for Beam Physics and Technology

# Stability Improvements at FLUTE
## (Verbesserung der Stabilität von FLUTE)

Master thesis
of

Marvin-Dennis Noll

at the Institute for Beam Physics and Technology

| | |
|---|---|
| Reviewer: | Prof. Dr.-Ing. John Jelonnek (IHM) |
| Second Reviewer: | Prof. Dr. Anke-Susanne Müller (IBPT) |
| Advisor: | Dr. Nigel Smale (IBPT) |

15.11.2020 – 01.07.2021

# Erklärung zur Selbstständigkeit

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde und dass ich die Satzung des KIT zur Sicherung guter wissenschaftlicher Praxis in der gültigen Fassung vom 24.05.2018 beachtet habe.

Karlsruhe, den 01.07.2021, _____

Marvin-Dennis Noll

Als Prüfungsexemplar genehmigt von

Karlsruhe, den 01.07.2021, _____

Prof. Dr.-Ing. John Jelonnek (IHM)

# Contents

# List of Figures

# List of Tables

## Abstract

*F*erninfrarot *L*inac- *u*nd *T*est-*E*xperiment (FLUTE), a compact linear accelerator, is currently designed and under commission at the Karlsruhe Institute of Technology (KIT). Its main purposes are to serve as a technology platform for accelerator research, the generation of strong and ultra short THz pulses and in the future as an injection device for **c**ompact **St**orage ring for **A**ccelerator **R**esearch and **T**echnology (cSTART).

At the current commissioning state, the klystron which powers the electron gun/RF cavity and in later stages the linear accelerator is fed by a pulse forming network, which is driven by a high voltage source connected to mains power. For high and a stable output power of the cavity resonator, several parameters have to be tuned to the correct values and kept inside of sometimes small tolerance bands.
In the past, the coolant temperature of the cavities water cooling system and the dependency of the pulse forming network output of the mains voltage phase were predominant sources of instability. After dealing with these issues, the cavity output power stability was improved significantly but further improvements to the stability were still desired.

In this work instead of passively optimizing the stability of system parameters, an active approach is evaluated. By controlling the amplitude of the RF input signal of klystron, which is easily possible since it is low power, the effects of noise and/or drifts are mitigated. Here it is evaluated if a simple of the shelf voltage controllable attenuator is a feasible choice to control the RF input signal, which input data should be used and which algorithm and/or control system is suitable to determine the needed attenuator setting to stabilize RF output (of the cavity).
Furthermore since the next stage in the system depends on a stable electron bunch charge rather than cavity power, it is determined whether the charge measurements of a Faraday cup can be used to directly control electron bunch charge.

## Kurzfassung

–TODO–

# 1. Introduction

# 2. Theoretical Framework

## 2.1 Linear Accelerators

In a linear particle accelerator (LINAC), charged particles such as electrons are accelerated to increase their total energy over their energy at rest.

Compared to heavier particles, such as protons ($m_p = 938.27\,\mathrm{MeV/c^2}$), electrons are fairly light ($m_p = 0.511\,\mathrm{MeV/c^2}$). Therefore they need to be brought to speeds comparable to the speed of light to achieve an useful energy increase. For this reason relativistic mechanics are needed to describe their movements.[1] [1]

With the speed of light $c = 2.998 \times 10^8\,\mathrm{m\,s^{-1}}$ and the particle velocity $v$, it is common to define [2]:

$$\text{(normalized velocity)} \qquad \beta = \frac{v}{c} \tag{2.1}$$

$$\text{(relativistic mass factor)} \qquad \gamma = \frac{1}{\sqrt{1-\beta^2}} \tag{2.2}$$

The total energy of a particle is

$$U = \underbrace{(\gamma - 1)\, mc^2}_{\text{kinetic engery } W} + \underbrace{mc^2}_{\text{rest energy}} = \gamma mc^2. \tag{2.3}$$

With the kinetic electron energy out of the FLUTE electron gun of $W = 7\,\mathrm{MeV}$, rearranging Equation 2.1 and using in $W = (\gamma - 1)\, mc^2$ yield

$$\gamma = \frac{W}{m_e\, c^2} + 1 = 14.699 \tag{2.4}$$

$$\beta = \sqrt{1 - {}^1\!/\!_{\gamma^2}} = 0.997\,68. \tag{2.5}$$

## 2.2 Beam Diagnostic Devices for Linear Accelerators

All (linear) particle accelerators need diagnostic devices to monitor the beam sizes, shapes, positions, length or charge.

### Beam Position Monitor (BPM)

### Faraday Cup

## 2.3 Signal Analysis

### 2.3.1 Auto correlation and Cross correlation

The *cross covariance* between two stochastic processes $x[n]$ and $y[n]$ is a measure of the similarity between $x[n]$ at index $n_1$ and $y[n]$ at index $n_2$ and is defined as

$$r_{xy}[n_1, n_2] = \mathrm{E}\left\{(x[n_1] - \mu_x[n_1])(y[n_2] - \mu_y[n_2])^*\right\}. \tag{2.6}$$

[1] As relativistic mechanics are a super set of classical mechanics, the equations also apply for slower particles.

For the special case of $y[n] := x[n]$, $r_{xx}[n_1, n_2]$ is called *auto covariance* and is a measure of self similarity of $x[n]$ [3, p. 172].

The processes $x[n]$ and $y[n]$ are called *wide sense stationary* (WSS) if the following two properties hold [3, p. 167]. First, their means $\mu_\xi[n]$ are constant, i.e. they do not depend on the sample index:

$$\mu_x[n] = \mu_x \tag{2.7}$$

$$\mu_y[n] = \mu_y \tag{2.8}$$

Also the auto covariance does not depend on the absolute sample indices $n_1$ and $n_2$, but merely on the difference between them:

$$r_{xy}[n_1, n_2] = r_{xy}[m], \qquad \text{with: } m := n_2 - n_1 \tag{2.9}$$

If both process in Equation 2.6 are WSS, Equation 2.6 simplifies to

$$r_{xy}[m] = \mathrm{E}\left\{(x[n] - \mu_x)(y[n-m] - \mu_y)^*\right\}. \tag{2.10}$$

For the auto covariance both means are identical and can be moved outside the expectation operator:

$$r_{xx}[m] = \mathrm{E}\left\{(x[n])(y[n-m])^*\right\} - \mu_x^2. \tag{2.11}$$

When analyzing signals, the stochastic processes are often unknown and only one realization $x[n]$ is known. But if the process generating $x[n]$ is *(weakly) ergodic*, then one realization is enough to determine the auto covariance of the process [4, p. 252]. Then the auto covariance can be estimated with

$$\hat{r}[m] = \frac{1}{N} \sum_{n=m+1}^{N} x[n]\, x^*[n-m] \qquad m \in [0,\, N-1] \tag{2.12}$$

### 2.3.2 Estimating the Spectrum of a Stochastic Process

For a deterministic, time-discrete signal $x[n] \in \mathcal{L}_1$, the discrete Fourier transform (DFT) exists[5] and is defined as

$$X[k] = \sum_{n=0}^{N-1} x[n]\, \mathrm{e}^{-j\frac{2\pi}{N}k\,n} \qquad k, n \in [0,\, N-1], \tag{2.13}$$

using $k = \frac{N}{2\pi}\,\omega = N\,f$ as the independent, discrete frequency variable. From the complex sequence $X[k]$, often only the magnitude (or energy) is of greater interest while the phase information are neglected. Therefore, $S_{xx}$ is defined as

$$S_{xx} = |X[k]|^2 \tag{2.14}$$

and called the *energy spectral density (ESD)*.

If $x[n]$ is the realization of a stochastic process, then it is of random nature rather than deterministic. Because realizations of physical processes do not posses finite energy, they are not in the $\mathcal{L}_1$ set and their DFT is not defined [6, p. 5].

In this case instead of an energy spectral density, the spectrum of the average power of the process, called the *power spectral density (PSD)*, can be used instead. To compute the PSD, either there are two possibilities:

$$\Phi_{xx}[k] = \sum_{m=-\infty}^{\infty} r[m]\, \mathrm{e}^{-j\frac{2\pi}{N}k\,m} \tag{2.15}$$

$$\Phi_{xx}[k] = \lim_{N\to\infty} \mathrm{E}\left\{\frac{1}{N}\left|\sum_{n=0}^{N-1} x[n]\, \mathrm{e}^{-j\frac{2\pi}{N}k\,n}\right|^2\right\} \tag{2.16}$$

When assuming $r[m]$ decays "fast enough", i.e.

$$\lim_{N\to\infty} \frac{1}{N} \sum_{m=-N}^{N} |m|\ |r[m]| = 0 \tag{2.17}$$

then Equation 2.15 and Equation 2.16 are equal[6, p. 7].

For measured data however neither equations can be used directly. For Equation 2.15 the auto covariance sequence $r[m]$ is unknown. But it could be estimated with Equation 2.12. In case of Equation 2.16 it is not possible to evaluate the limit, because only finite length data can be sampled and also the expectation can not be computed since in general there is only one realization available. Both operations can be neglected when doing an estimation.

With these practical changes in place, Equation 2.15 and Equation 2.16 become

$$\hat{\Phi}_{c,\,xx}[k] = \sum_{m=-(N-1)}^{N-1} \hat{r}[m]\, \mathrm{e}^{-j\frac{2\pi}{N}k\,m} \qquad \text{(Correlogram)} \tag{2.18}$$

$$\hat{\Phi}_{p,\,xx}[k] = \frac{1}{N}\left|\sum_{n=0}^{N-1} x[n]\, \mathrm{e}^{-j\frac{2\pi}{N}k\,n}\right|^2 \qquad \text{(Periodogram).} \tag{2.19}$$

Both methods yield equal results, if $r[m]$ is estimated with the biased estimator $\hat{r}[m]$ in Equation 2.12 in contrast to the unbiased estimator (compare [6, p. 24])

$$\hat{r}_{\text{unbiased}}[m] = \frac{1}{N-m} \sum_{n=m+1}^{N} x[n]\, x^*[n-m] \qquad m \in [0,\, N-1]. \tag{2.20}$$

[7] shows one key weakness of the unmodified periodogram method in Equation 2.19: The variance does not decrease significantly with more samples $N$. Instead the variance of the periodogram for each frequency approaches the square of the actual PSD:

$$\lim_{N\to\infty} \mathrm{Var}\left\{\hat{\Phi}_{p,\,xx}[k]\right\} = \Phi_{xx}^2[k] \tag{2.21}$$

Furthermore the periodogram/correlogram suffer from the smearing and leakage effects because the limited length of the data samples always causes an implicit windowing, thus reducing frequency resolution.

There are several popular methods that improve on the periodogram/correlogram concepts:

**Blackman-Tukey:** Because of the poor accuracy of $\hat{r}[m]$ for $k \approx N$ in the definition of $\hat{\Phi}_{c,\,xx}[k]$ and the bigger the $N$, the more small errors in $\hat{r}[m]$ sum up, truncating/windowing of $\hat{r}[m]$ with $w[k]$ (length $M$) can be beneficial for the accuracy of the estimation.

$$\hat{\Phi}_{BT,\,xx}[k] = \sum_{m=-(M-1)}^{M-1} w[k]\hat{r}[m]\, \mathrm{e}^{-j\frac{2\pi}{N}k\,m} \tag{2.22}$$

The choice of the window $w[k]$ trades frequency resolution for variance and smearing for leakage reduction [6, p. 41].

**Barlett:** The Barlett method reduces the variance of the periodogram by splitting the $N$ data samples in $Q = {}^{N}/_{M}$ blocks and averaging together the sub-periodograms:

$$\hat{\Phi}_{q,\,xx}[k] = \frac{1}{M}\left|\sum_{n=0}^{M-1} x_q[n]\, \mathrm{e}^{-j\frac{2\pi}{M}k\,n}\right|^2 \tag{2.23}$$

$$\hat{\Phi}_{B,\,xx}[k] = \frac{1}{Q}\sum_{q=1}^{Q} \hat{\Phi}_{q,\,xx}[k] \tag{2.24}$$

The variance of the estimation scales with $Q$ [7, p. 6]:

$$\mathrm{Var}\left\{\hat{\Phi}_{B,xx}[k]\right\} = \frac{1}{Q}\Phi_{xx}^2[k] \tag{2.25}$$

**Welch:** The Welch method combines splitting the data into $Q$ segments with windowing each segment and allowing the segments to overlap. With $P = {}^1\!/_M \sum_{n=0}^{M-1} |w[n]|^2$ being the "power'' of the window, the Welch method is computed as

$$\hat{\Phi}_{s,xx}[k] = \frac{1}{MP}\left|\sum_{n=0}^{M-1} x_s[n]\,\mathrm{e}^{-j\frac{2\pi}{M}k\,n}\right|^2 \tag{2.26}$$

$$\hat{\Phi}_{W,xx}[k] = \frac{1}{Q}\sum_{s=1}^{Q}\hat{\Phi}_{s,xx}[k]. \tag{2.27}$$

Compared to the Barlett method, the overlapping of up to 50 % (see [8]) allows increasing $Q$, thus reducing the variance.

$$\mathrm{Var}\left\{\hat{\Phi}_{W,xx}[k]\right\} = \frac{1}{Q}\Phi_{xx}^2[k] \tag{2.28}$$

In case on a non-stationary signal $x[n]$, one possibility to analyze and display the spectral content is the use of the short time Fourier transform (STFT) and the spectrogram, which is a two dimensional power spectral density function mapping frequency and time to a third coordinate like height, intensity or color.

To calculate the spectrogram, the signal is split into segments with the sliding window $w[n-m]$ for which duration the signal is assumed to be stationary. For each segment at time index $m$, the periodogram is calculated according to

$$\hat{\Phi}_{xx}[k,m] = \frac{1}{N}\left|\sum_{n=0}^{N-1} w[n-m]x[n]\,\mathrm{e}^{-j\frac{2\pi}{N}k\,n}\right|^2. \tag{2.29}$$

## 2.4 Feedback Control Systems

Feedback control systems are used to control a dynamic system (plant) in such a way that its output $y(t)$ follows a certain input $x(t)$ and disturbances on the output $d(t)$ are rejected. The general structure of a closed loop control system is shown in Figure 2.1. To achieve the tracking of the input and the stabilization of the output, a controller $G(s)$ uses the error $e(t)$ to control the plant $P(s)$ accordingly. The error is defined as

$$e(t) = x(t) - r(t) = x(t) - [y(t) * h(t)] \tag{2.30}$$

with $h(t)$ being the inverse Laplace transform of the filters transfer function $H(s)$.

Feedback control systems, or closed-loop systems are to be differentiated from open-loop systems, in which there is no return path, so they cannot compensate for *un*known disturbances. If $d(t)$ is known $\forall t$, then an open loop system would be possible and any errors could simply be compensated.

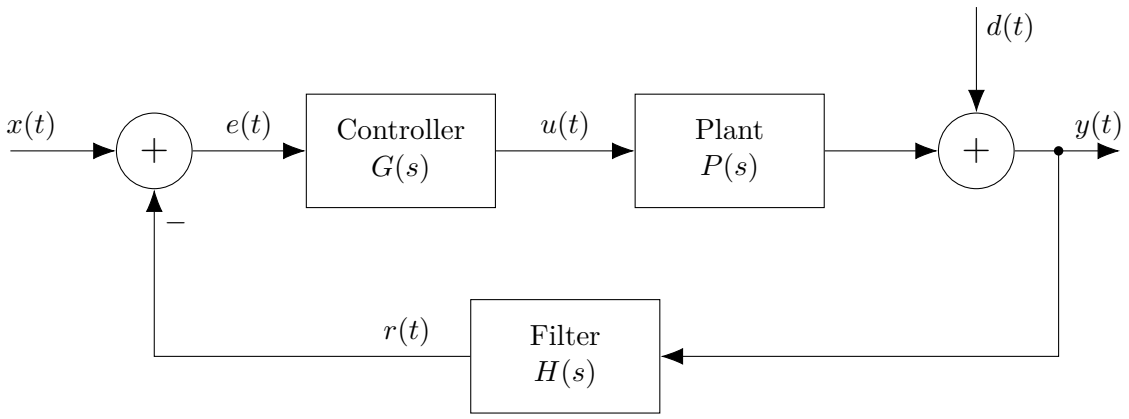



**Figure 2.1:** General structure of a time continuous feedback control system

### 2.4.1 Disturbance Rejection and Input Tracking

Disturbance rejection and input tracking are two important characteristics to evaluate a stable controller. To calculate them, the block diagram in Figure 2.1 and the Laplace transform of the inputs/outputs is used.[2]

To calculate how the output $y(t)$ depend on the input $x(t)$, the input tracking transfer function can be used.[9, p. 88] It is calculated as the transfer function $F_T = {}^{Y(s)}/_{X(s)}$ by setting $d(t) = 0$:

$$Y(s) = G(s)P(s)E(s) \quad \text{and} \quad E(s) = X(s) - H(s)Y(s) \tag{2.31}$$

$$\Leftrightarrow Y(s)\left[1 + G(s)P(s)H(s)\right] = G(s)P(s)X(s) \tag{2.32}$$

$$\Leftrightarrow F_T := \frac{Y(s)}{X(s)} = \frac{G(s)P(s)}{1 + G(s)P(s)H(s)}. \tag{2.33}$$

On the other hand, the transfer function $F_{DR} = {}^{Y(s)}/_{D(s)}$ can be used to describe the systems response to a disturbance.[9, p. 88] It is defined by letting $x(t) = 0$ and calculating

$$Y(s) = G(s)P(s)E(s) + D(s) \quad \text{and} \quad E(s) = -H(s)Y(s) \tag{2.34}$$

$$\Leftrightarrow Y(s)\left[1 + G(s)P(s)H(s)\right] = D(s) \tag{2.35}$$

$$\Leftrightarrow F_{DR} := \frac{Y(s)}{D(s)} = \frac{1}{1 + G(s)P(s)H(s)}. \tag{2.36}$$

### 2.4.2 Stability

The application of a controller to a system is only useful if the resulting system has a stable behavior. One possible definition of stability is the bounded input bounded output (BIBO) criterion[9, p. 82]:

**Definition 1.** *(BIBO stability) A LTI system is said to be BIBO stable if for some $M, N \in \mathbb{R}^+$, the response to a bounded input $|u(t)| \leq M$ results in a bonded output $|y(t)| \leq N$.*

For a given control system, one way to analyze its stability is to plot the locus $z = F_o(s = j2\pi f)$ of the open loop frequency response

$$F_o(s) = G(s)P(s)H(s) \tag{2.37}$$

from $f = 0$ to $f = \infty$ and using the Nyquist stability criterion. For the special case of a stable open loop $F_o(s)$ [3] the Nyquist stability criterion can be stated as[9, p. 111]

[2] The Laplace transform of a function in time $f(t)$ is written as $F(s) = \mathcal{L}\{f(t)\}$.

[3] The stability of $F_o(s)$ can often easily be determined from the block diagram.

**Definition 2.** *(Nyquist stability criterion) If the open loop $F_o(s)$ is stable, then the closed loop is stable if $z = F_o(s = j2\pi f)$ does not go through or encircles $z = -1$.*

# 3. Current and Needed Stability of the Electron Gun

In this chapter the current stability of the RF system powering the electron gun is analyzed and the needed stability is defined.

## 3.1 Metrics to quantify stability

"Stability'' can have different meanings depending on the context. In case of signal processing, a signal is usually said to be *stable* if it has only little variation around its mean or some target value, i.e. the mean has to be constant and the variance stays below some threshold. Stability is not to be confused with stationarity, which requires the mean and the variance and the autocorrelation stay constant over time [10]. To express stability as a single numerical value, there are several possibilities, some are described in the following.

### Relative Standard Deviation

This measures the stability as the standard deviation but related to the mean value to make it comparable to other quantities with different scaling or units.

The relative, or percentual, standard deviation of the stationary stoachastic process $X$ is defined using the mean $\mu_X$ and the standard deviation $\sigma_X$ as

$$\%\mathrm{STD}_\mathrm{X} := \frac{\sigma_X}{\mu_X}. \tag{3.1}$$

Is the process $X$ non stationary, $\%\mathrm{STD}_\mathrm{X}$ depends on the absolute time $t$ and the window size $T$ for which the process is assumed to be stationary:

$$\%\mathrm{STD}_\mathrm{X} = \%\mathrm{STD}_\mathrm{X}(t, T) \tag{3.2}$$

In that case, for a fixed window size $T = T_0$, a mean percentual standard deviation can be computed with (assuming discrete time steps $t_n$, $n \in [0, N-1]$)

$$\%\mathrm{STD}_\mathrm{X}(T = T_0) = \frac{1}{N} \sum_{n=0}^{N-1} \%\mathrm{STD}_\mathrm{X}(t_n, T_0) \tag{3.3}$$

### Mean Squared Error

The mean squared error sums up the squared errors $(x[n] - x_t[n])^2$ of $x[n]$ from a set value $x_t[n]$. To remove the effect of the length of the data sequence, the sum is devided by the length of the sequence $N$:

$$MSE_x := \frac{1}{N} \sum_{n=0}^{N-1} (x[n] - x_t[n])^2 \tag{3.4}$$

#### Relative Power of Most Prominent Noise

This novel approach compares the power of the most prominent noise power source $P_{noise,\,max}$ of the signal $x$ with the total power $P_x$:

$$MPN_x := \frac{P_{noise,\,max}}{P_x} \tag{3.5}$$

## 3.2 Determine the Needed Stability for FLUTE Operation

## 3.3 Analyzing the Current Stability

# 4. Interfacing FLUTE

This chapter covers methods on interfacing the FLUTE accelerator, that is how to read diagnostic measurements into the control system from FLUTE and how to influence the electron acceleration appropriately to achieve stabilization.
In this chapter *input* and *output* refer to the view from the control system.

## 4.1 Inputs

FLUTE uses the *Experimental Physics and Industrial Control System (EPICS)*[11] for control of various parts of the accelerator, to send real time data to be archived and to build user interfaces via *Control System Studio (CSS)*[12], a JAVA based integrated development environment (IDE).

EPICS offers client/server and publish/subscribe paradigms to access data in so called process variables (PV) through channels. Modules are usually written in the C programming language. To ease the access to EPICS channels in programs written with the Python language, the package *PyEpics*[13] can be used. Since all data of interest as input for the control system can be extracted through an EPICS channel, the next section deals with using PyEpics to obtain the data.

### 4.1.1 Accessing EPICS channels in Python with PyEpics

Before usage PyEpics needs to be installed, e.g. with `pip3 install pyepics` from the *PyPi* repository. If the computer running the Python code can reach the EPICS CA repeater on the machine network, the connection is established automatically in the background. To get a channel value asynchronously, i.e. at an arbitrary time, the function `caget(pvname)` can be used with the name of the desired process value, see Listing 4.1.

**Listing 4.1:** Using `caget()` to get the value of an EPICS process value

```
1 from epics import caget
2 print(f"Cavity RF power: {caget('F:RF:LLRF:01:GunCav1:Power:Out')}")
```

Another way is to setup a channel object and create a subscription with an user defined callback function that is executed each time the process variable changes. This implements synchronous access to the PV and can be compared to an interrupt rather than polling the variable as in **??**.
For a non trivial example see Listing 4.2. In this program, the time differences between new values and their statistics are printed to the console.

**Listing 4.2:** Using a user defined callback function to access an EPICS process value

```
1 from epics import ca
2 import time
3 import numpy as np
4
```

```
dts=np.array([])
lastCalled=time.time()

def call(pvname, value, **kwargs):
  global lastCalled, dts
  now=time.time()
  dt=now-lastCalled
  lastCalled=now
  dts=np.append(dts, dt)

chid=ca.create_channel("F:RF:LLRF:01:GunCav1:Power:Out")
_, _, eventID=ca.create_subscription(chid, callback=call, use_time=True)

while(True):
  time.sleep(2)
  print(f"N:{len(dts)},mean:{np.mean(dts)},min:{np.min(dts)},max:{np.max(dts)},std:{np.std(dts)}")
```

### 4.1.2 Properties of the Available Process Variables

In this section some process variables that may be used as inputs for the control system are analyzed. These are:

- `F:RF:LLRF:01:GunCav1:Power:Out Value`: The RF power measured immediately before the cavity
- `F:AX:DAQDT:01:1:Wave:05:Sample Value`: The charge measured with a Faraday cup (RadiaBeam Technologies FARC-04 [14]) and amplified with a charge sensitive amplifier (PCB 421A25 [15])
- `F:INJ-1:Gun:01:Temperature:Body Value`: The body temperature of the cavity

In Figure 4.1 all three are plotted for a duration of 15 min without any interference to the system and the system being in steady state operation.

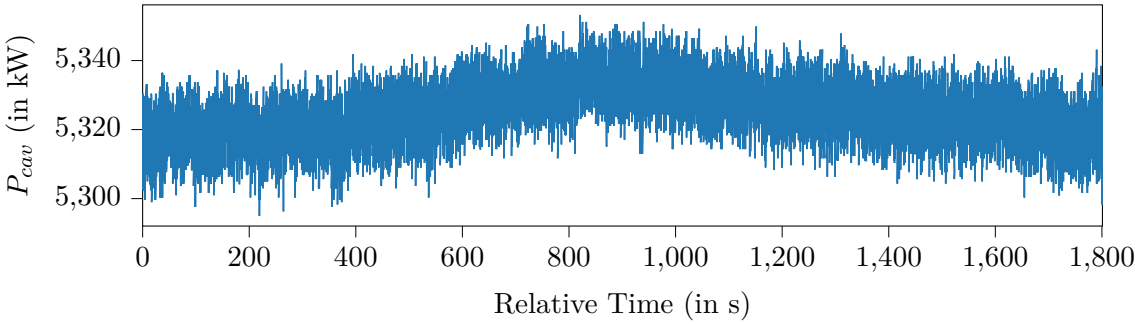
In addition the sample times of the process variables is examined if the method with a custom callback is used (see Listing 4.2) or the data is extracted from the archive. The differences in the sample times are calculated according to

$$\Delta = t_{n+1} - t_n. \tag{4.1}$$

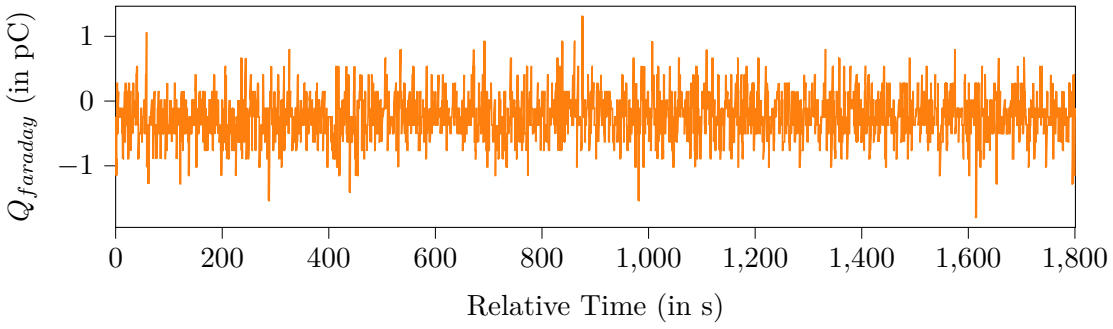
Then a histogram with the relative frequency on the y axis is used as an estimator for the probability density function of the sample time intervals (see Figure 4.2). The histogram shows that the time series resulting from recording process variables out of the EPICS system are highly unevenly spaced. Thus the data needs to be converted to posses evenly spaced sample times to use common signal processing methods like the DFT or digital LTI filters. For offline analysis of prerecorded data, it can easily be resampled to a fixed time grid. But for online operation of a filter or a whole control system it is not possible to use an arbitrary resample method because they are often non causal or introduce significant group delay when made causal. Instead calling `caget()` with a (software-) timer can be used to get evenly spaced samples online.

**Table 4.1:** Comparing quantization steps

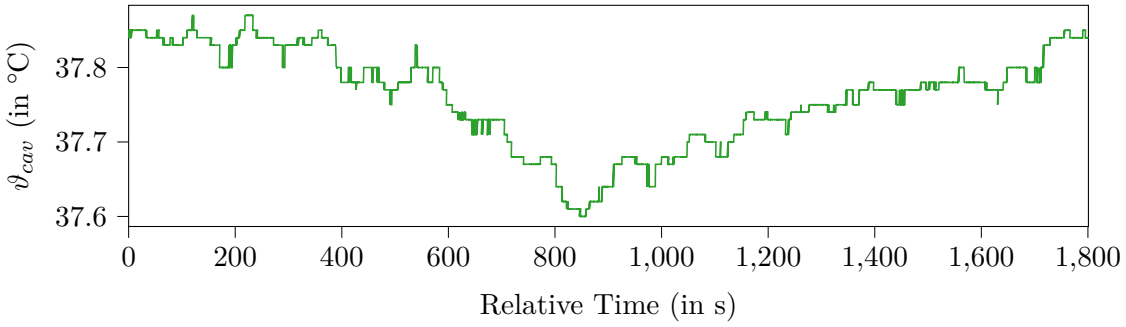
| PV | $N_{unique}$ | $q_{avg}$ | $q_{norm}$ |
|---|---|---|---|
| F:RF:LLRF:01:GunCav1:Power:Out Value | 84 | 0.6935 | 0.011 904 |
| F:AX:DAQDT:01:1:Wave:05:Sample Value | 22 | 0.015 | 0.045 45 |
| F:INJ-1:Gun:01:Temperature:Body Value | 18 | 0.1294 | 0.055 55 |




**(a)** Cavity RF power `F:RF:LLRF:01:GunCav1:Power:Out Value`




**(b)** Faraday cup charge `F:AX:DAQDT:01:1:Wave:05:Sample Value`




**(c)** Cavity body temperature `F:INJ-1:Gun:01:Temperature:Body Value`

**Figure 4.1:** Comparing the quantization noise of three process variables

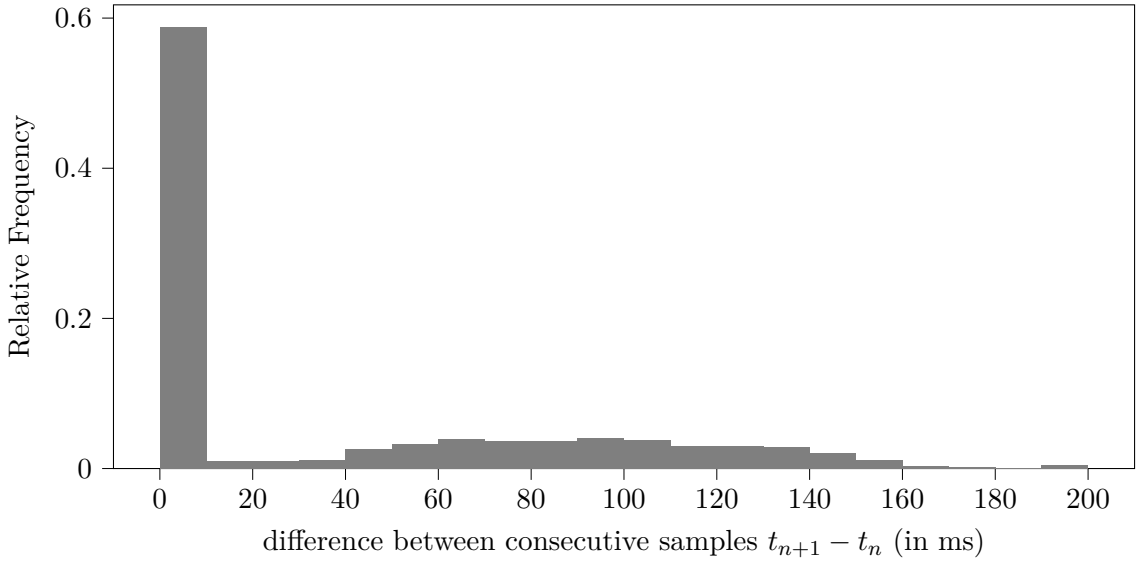



**Figure 4.2:** Histogram of the sample time intervals $\Delta$ of the plots in Figure 4.1

### 4.1.3 Filtering the RF power Signals

In case of the klystron output RF power or the power at the cavity, filtering of the signal is needed to remove zero outliers. These outliers occur if a high voltage arc occurs inside the cavity, it is detected and the LLRF control system shuts of the RF power for the current pulse (called a "breakdown''). These outliners are not representative of the average RF power inside the cavity over multiple pulses and thus would greatly impair the controller performance. For that reason, before any further filtering to remove noise etc, a breakdown removal filter is used (Listing 4.3). In principle the new power value is checked to be inside a band which size is determined by the mean deviation of the $N_{filt}$ previous values and a scaling $m$. The percentile differences are used here as they are robust against outliers (i.e. other breakdowns) in the $N_{filt}$ previous values opposed to a normal standard deviation. The scaling with $(2 * 1.2815)^{-1}$ is used to make the mean deviation comparable to a standard deviation.

**Listing 4.3:** Breakdown removal filtering

```
1 if(abs(P[i]−np.median(P[i−3∗Nfilt:i−Nfilt])) <
2 m∗(np.percentile(P[i−3∗Nfilt:i−Nfilt],90)−np.percentile(P[i−3∗Nfilt:i−Nfilt],10)/(2∗1.2815))):
3     P_filt =np.append(P_filt,P[i])
4 else:
5     breakdown_locations_predicted=np.append(breakdown_locations_predicted,i)
6     P_filt =np.append(P_filt,np.median(P[i−3∗Nfilt:i−Nfilt]))
```

## 4.2 Output: Controllable RF Attenuator

The output signal computed by the control systems has to have a way of influencing the RF power send to the cavity. This could be done over an EPICS channel (e.g. with the PyEpics function `caput()` to set the value of a process variable via a channel access). However to make it possible to move the control system from a general purpose personal computer to a dedicated digital signal processor, FPGA or similar in the future, using a physical device in the signal path is preferred.

### 4.2.1 Defining Requirements

The controllable attenuator should allow to vary the attenuation in a span big enough to counteract typical instabilities on the FLUTE RF power. The set attenuation should be stable. This is especially needed in cases where the control system is not enabled. Then the attenuator should not add noise or drift to stay "transparent'' for other systems. To allow for other attenuator or amplifiers in the signal path to compensate the attenuation around its operating point, the nominal attenuation should be as low as possible. The attenuation resolution should be low enough to allow for fine control and not to add noticeable quantization noise. Also the setup time for a new value should be small enough to not be visible to the control system.

Other factors like the control/supply voltages and the operating temperature range are limited by the available hardware or governed by the mounting location.

All requirements are summarized in Table 4.2.

**Table 4.2:** Requirements for the controllable RF attenuator

| Requirement | Value/Range |
|---|---|
| attenuation adjustment | ±0.2 dB |
| attenuation stability | ±0.001 dB |
| nominal attenuation at operating point | < 10 dB |
| attenuation resolution | 0.001 dB |
| setup time | 10 ms |
| operating temperature range | (25.0 ± 0.1) °C |
| supply voltage | 3 V to 12 V |
| control voltage | 3 V to 12 V |

### 4.2.2 Evaluation of the ZX73-2500-S+ Controllable RF Attenuator

The *ZX73-2500-S+* is a voltage controllable RF attenuator with coaxial SMA connectors by the company Mini-Circuits. As there is no alternative model from Mini-Circuits and devices from other manufactures are offered with similar specifications, only the ZX73-2500-S+ is evaluated in detail in this section.

The ZX73-2500-S+ attenuator consists of a brass housing/shielding containing the Mini-Circuits RVA-2500+, a variable SMD attenuator in the DV874 case form factor. The RF input and output are connected with female SMA screw connectors. The power supply and control voltage are connected with solder pins. In order to use shielded cables and a reliable connection, for all measurements SMA connectors are soldered to the supply and control pins. According to equivalent circuit in the data sheet[16] it can be assumed it is based on the common quad-$\pi$ pin diode design[17].

The attenuation versus frequency measurements from the manufactures data sheet are redone to both get a first impression of the device and verify it is generally operational and also as a sanity check for the used laboratory test equipment. Because the signal used by the FLUTE RF system is a 3 GHz single harmonic, the frequency measurement range is augmented over the maximum of 2.5 GHz in the data sheet to 4 GHz. Between the measurement of each network analyzer trace, the control voltage $V_{control}$ of the attenuator is set to 0 V, 2 V, 4 V, 6 V or 12 V. The result is shown in Figure 4.5[1]. When comparing the measured plots to the plots in the data sheet, there are obvious discrepancies. For $V_{control} = 0\,\mathrm{V}$ the attenuator is very susceptible to noise on the control input, which could explain the differences for this curve. In the case of 2 V and 4 V, the almost constant offset scales with a similar logarithmic fashion as the attenuation does, which suggests device tolerances causing the deviations.

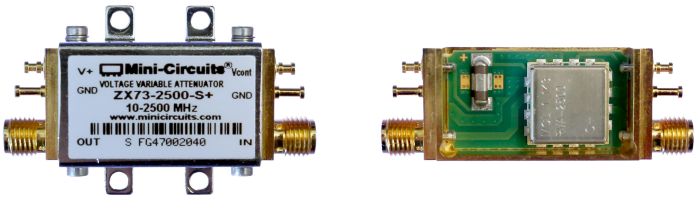
From this quick examination it is not possible to predict how the attenuator behaves for small changes in $V_{control}$ and how changes in the environment, such as the body temperature or the supply voltage, cause unwanted variations in the attenuation.

For this reason, with different measurement setups, the ZX73-2500-S+ is examined in greater detail in the next sections.

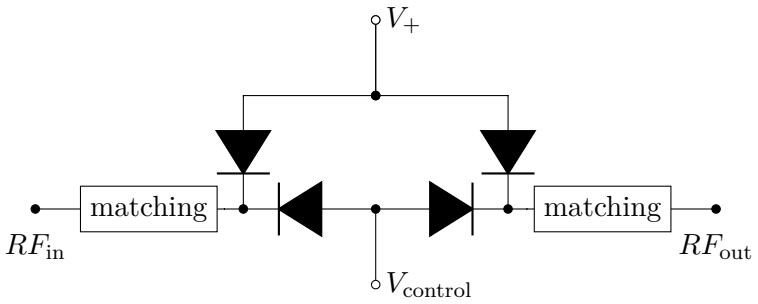
To compare desired and spurious influences on the attenuation, the following model is used. The attenuation of the ZX73-2500-S+ $A$ depends on the control voltage $V_{control}$ but also on the supply voltage $V+$, the case temperature $\vartheta_{case}$ and the RF frequency $f$:

$$A = A(\underbrace{V_{control},\, V_+,\, \vartheta_{case},\, f}_{\vec{x}}) = A\left(\vec{x}\right) \tag{4.2}$$

[1] A through calibration of the network analyzer reduces the influence of the cables and connectors on the measurement. Change in attenuation due to play in the connectors and slight bend changes in the cables exceed the trace noise (0.004 dBrms) and cause an uncertainty of about 0.5 dB




**(a)** left: ZX73-2500-S+ with cover; right: cover removed showing RVA-2500+ and a buffer capacitor




**(b)** equivalent circuit from the manufacturer (redrawn from [16])

**Figure 4.3:** caption

Other influences are not modeled as they are difficult to control, such as the manufacturing tolerances between different devices, or assumed to be negligible, such as component degradation or humidity.

In the next sections, the influence of each component of the parameter vector $\vec{x}$ on $A$ is measured. As a coarse approximation, all influences are assumed to have linear effect. Then the total derivative of $A$, $\Delta A$ can be written as

$$\underbrace{A(\vec{x} - \vec{x_o}) - A(\vec{x_o})}_{\Delta A} = \sum_{j=0}^{3} \frac{\Delta A}{\Delta x_j} \Delta x_j \tag{4.3}$$

where the $\Delta A / \Delta x_j$ approximate the partial derivatives $\mathrm{d}A / \mathrm{d}x_j$.

With Equation 4.3, the maximum error on the attenuation, that is the variation around the operating point with a fixed $V_{control}$, can be approximated with

$$\Delta A_{\max} = \sum_{j=0}^{3} \left| \frac{\Delta A}{\Delta x_j} \cdot \Delta x_j \right| = \sum_{j=0}^{3} \left| \frac{\Delta A}{\Delta x_j} \right| \cdot |\Delta x_j| \,. \tag{4.4}$$

If not specified otherwise, the operating point

$$A(\vec{x_o}) =: A_o = A(V_{control},\, V_+,\, \vartheta_{case},\, f) \tag{4.5}$$

$$A_o = A\,(10\,\mathrm{V},\, 3\,\mathrm{V},\, 20\,^\circ\mathrm{C},\, 3\,\mathrm{GHz}) \tag{4.6}$$

is used.

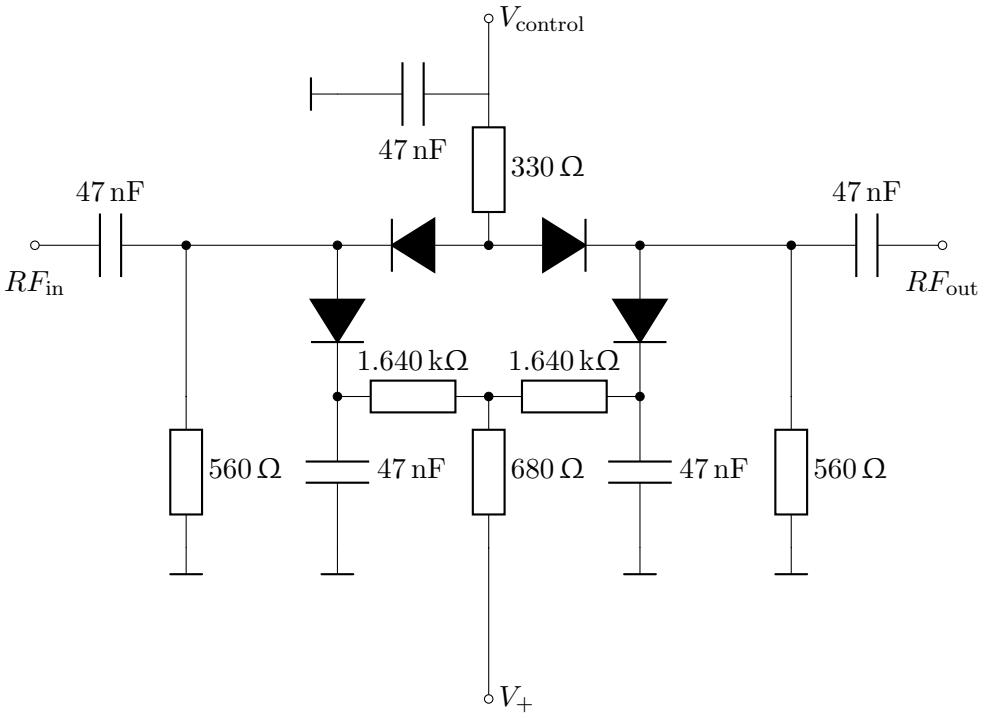



**Figure 4.4:** Controllable attenuator design around the HP HSMP-3814 RF PIN diodes (redrawn and simplified from [17])

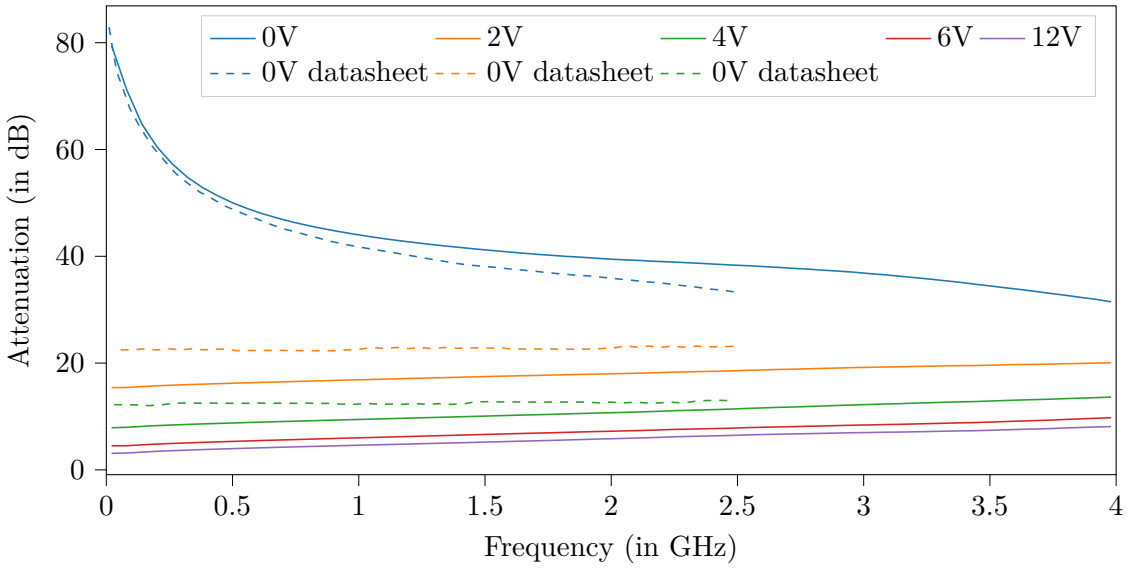



**Figure 4.5:** Attenuation vs. frequency over DC control voltage; measured with network analyzer (see B.6.1, parameters: $\#AVG$: 16, $IF\text{-}BW$: 10 kHz); plotted in dashed lines are the measurements from the data sheet (see [16, p. 2])

#### 4.2.2.1 Common Measurement Setup

For the following measurements a common setup is used to ease the recording of data and the sequential control for parameter studies. The needed tasks are:

- Supply the attenuator with the supply voltage $V_+$
- Supply the attenuator with the adjustable control voltage $V_{control}$
- Supply the RF input power with variable frequency $f$
- Measure the RF output power
- Record all data as time series to a computers storage device

To achieve this, the setup in Figure 4.6 is used. The supply and control voltage are generated by a Keysight 34972A with a 34907A module (see subsubsection B.2.1). To get a more accurate measurement of the actual voltages, two digital multimeters(Models 34470A and 34411A, see subsubsection B.1.2) are connected directly at the $V_{control}$ and $V_+$ pins. The RF signal is generated by a Rhode und Schwarz SMC 100A signal generator. With the HP E4419B and its two inputs and a power splitter, it is possible to directly get the attenuation of the ZX73-2500-S+. The body temperature of the device is monitored with a PT100 temperature sensor connected to the 34972A.

Each of the lab devices used is compatible with VXI11, which is a widely adopted standard that is used to send ASCII SCPI commands over an ethernet network (called LXI) or GPIB [18]. This enables remote and programatically control of the devices over the network. With the library *python-vxi11*, it is now possible to write a custom script, that sets the measurement devices to known initial conditions, drives the inputs of the attenuator and records the generated data. Timing the output of new set-values and recording of data is done with the *Advanced Python Scheduler*.[19]
The whole setup of the hardware in Figure 4.6 and the Python software, a measurement frequency of about 0.5 Hz to 1 Hz can be achieved, which is enough because most measurements taken are of a static nature and in the case of the temperature influence measurement, the thermal time constant is in the order of a few seconds. The limiting device is the HP E4419B which takes the longest to perform one measurement. Without it, measurement frequencies of over 4 Hz are possible.

All measurements are performed in the “RF lab’’ in building 348 (KARA hall on KIT Campus North) which is air conditioned to about $(20 \pm 2)\,°\mathrm{C}$.

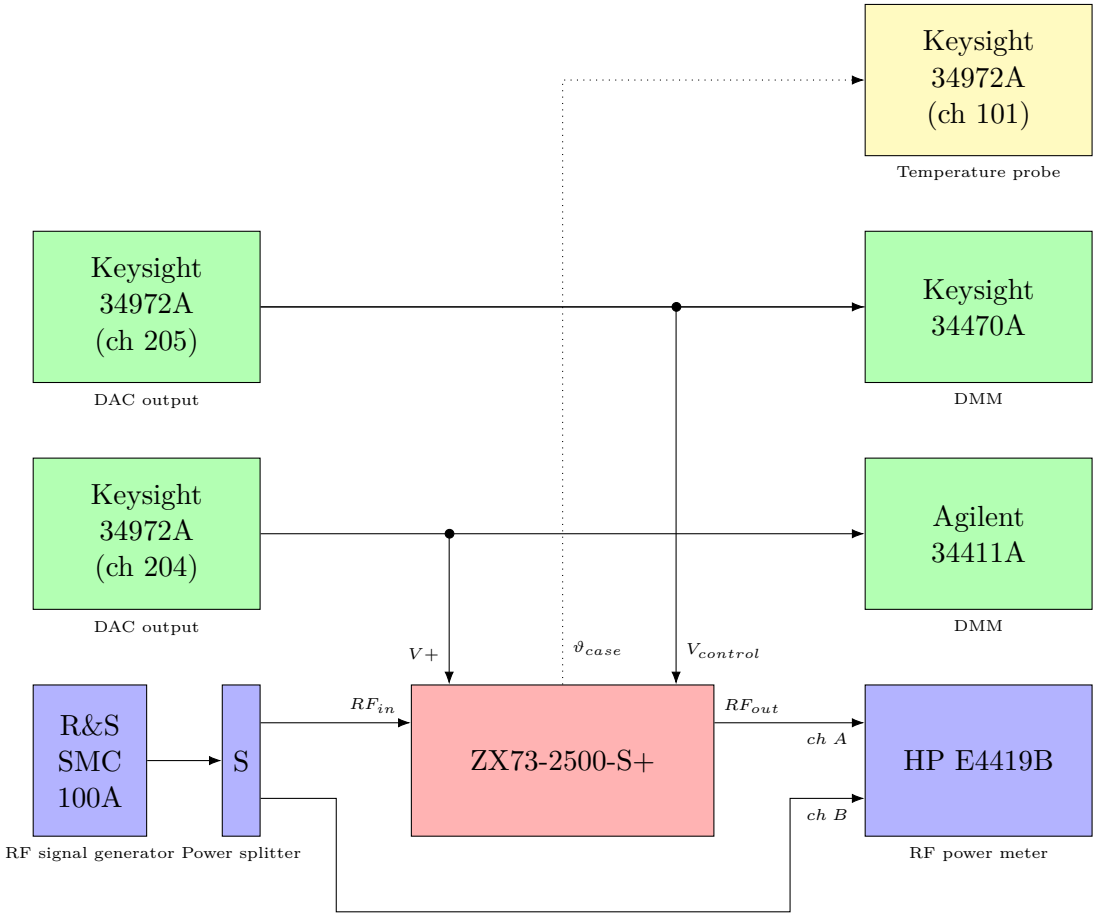



**Figure 4.6:** Measurement setup: DUT(red), RF generator/power splitter/meter(blue), DC sources/meters(green), temperature probe(yellow)

#### 4.2.2.2 Relation between Control Voltage and Attenuation

In this section the relation between the control voltage and the attenuation is examined. All other parameters are kept constant:

$$A(\vec{x}) := A(V_{control}) \tag{4.7}$$

The measured spectra in Figure 4.5 already suggest that there is a non linear relationship between the control voltage $V_{control}$ and the attenuation $A(V_{control})$ of the attenuator:

$$A(V_{control}) \neq const. \cdot V_{control} + A_0. \tag{4.8}$$

This also means the sensitivity

$$S(V_{control}) := \frac{\Delta A(V_{control})}{\Delta V_{control}} \tag{4.9}$$

is not a constant. In other words, for a desired relative change in attenuation, the needed adjustment in $V_{control}$ depends on the chosen operating point $A_o = A(V_{control,o})$.

With all other variables being at the operating point in Equation 4.5, $A(V_{control})$ is measured by stepping $V_{control}$ in 0.1 V increments up and down between 0 V and 12 V several times with each valued held constant for 30 s. $A(V_{control})$ is then calculated as the mean of all measurements $A_j(V_{control})$ with the same $V_{control}$ with

$$A(V_{control}) = \frac{1}{N} \sum_{j=0}^{N-1} A_j(V_{control}). \tag{4.10}$$

With the averaging done in Equation 4.10 and $N = 120$, the resulting mean standard deviation is $\sigma_A = 0.005\,74\,\mathrm{dB}$. Figure 4.7 shows the resulting plot.

The plot shows the attenuation can be varied over more than 30 dB and the magnitude of the sensitivity being large for small control voltages (1 V to 3 V). Since the required change in attenuation of less than 1 dB is much smaller, it is only necessary to vary $V_{control}$ around the operating point.

The optimal operating point for $V_{control}$ is selected by considering the following three aspects. First, the absolute attenuation at the operating point should be as low as possible to not worsen the SNR of the signal path. Second the control voltage has to fit in the possible output voltage range of the voltage source and still allow for adjustment towards both lower and higher voltages. In case of the Keysight 34972A/34907A the maximum possible output voltage is 12 V. Third, the sensitivity should be as low as feasible to make the attenuation less susceptible to noise on the $V_{control}$ input.

For these reasons, $V_{control,o} = 10\,\mathrm{V}$ (as already used in Equation 4.5) is chosen as the operating point, which allows $V_{control}$ to be varied in 8 V to 12 V. At the operating point $\vec{x_o}$ (with $V_{control,o} = 10\,\mathrm{V}$), the measured attenuation is

$$A(V_{control}) = 6.4859\,\mathrm{dB} \tag{4.11}$$

Figure 4.8 shows the attenuation and sensitivity around the operating point.

The sensitivity at the operating point is determined with discrete forward differentiation as

$$S(V_{control}) = S(10\,\mathrm{V}) = \frac{\Delta A}{\Delta V_{control}} = -0.151\,\mathrm{dB\,V^{-1}}. \tag{4.12}$$

For further error calculations, the maximum ${}^{\Delta A(V_{control})}/_{\Delta V_{control}}$ is needed. Figure 4.8 shows the maximum magnitude of ${}^{\Delta A(V_{control})}/_{\Delta V_{control}}$ to be at the lower edge of the $V_{control}$ range of $S(8\,\mathrm{V})$:

$$\left[\frac{\Delta A(V_{control})}{\Delta V_{control}}\right]_{\max} = -0.242\,\mathrm{dB\,V^{-1}} \tag{4.13}$$

The minimum possible step size of the attenuation can be calculated using $[{}^{\Delta A(V_{control})}/_{\Delta V_{control}}]_{\max}$ and the DAC output resolution of the Keysight 34907A ($^{24\,\mathrm{V}}/_{2^{16}-1} = 366.22\,\mu\mathrm{V}$):

$$\delta A(V_control) = 0.242\,\mathrm{dB\,V^{-1}} \cdot 366.22\,\mu\mathrm{V} = 0.000\,088\,625\,24\,\mathrm{dB} \tag{4.14}$$

To asses the stability of $V_{control}$, delivered from the Keysight 34972A (see subsubsection B.2.1) DAC, its stability over the course of one day is measured. For that the voltage is taken once every 2 seconds with a Keysight 34470A multimeter (see subsubsection B.1.2). The result is shown in Figure 4.9.

This measurement shows the stability of $V_{control}$ to be

$$\sigma_{V_{control}} = 0.173\,\mathrm{mV} \tag{4.15}$$

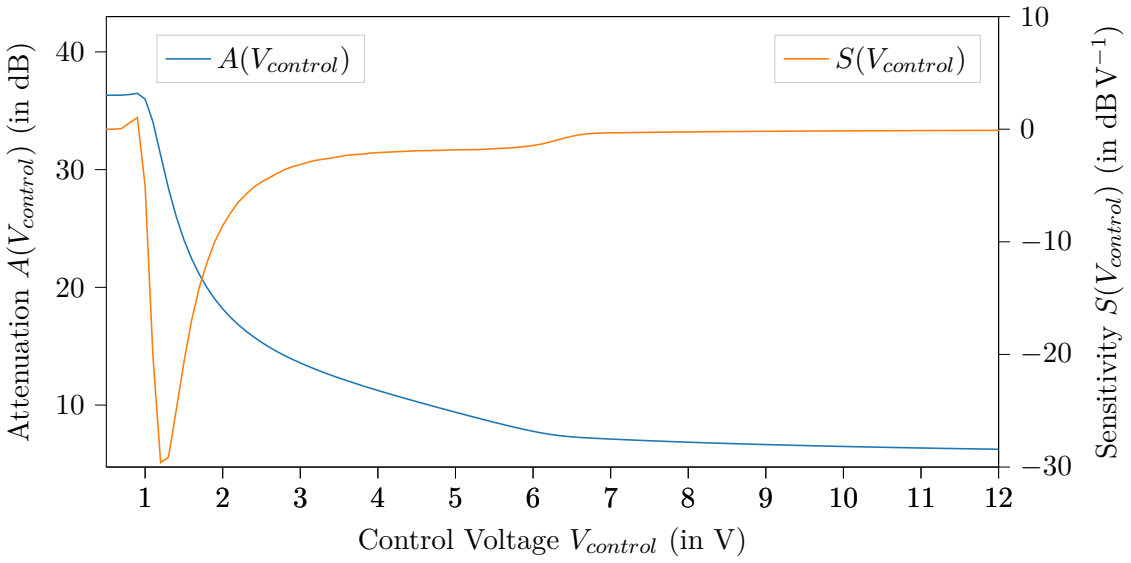



**Figure 4.7:** Measured attenuation $A(V_{control}, f = 3\,\text{GHz})$ of the ZX73-2500-S+ as a function of the control voltage input $V_{control}$ along with the sensitivity $S(V_{control}, f = 3\,\text{GHz})$

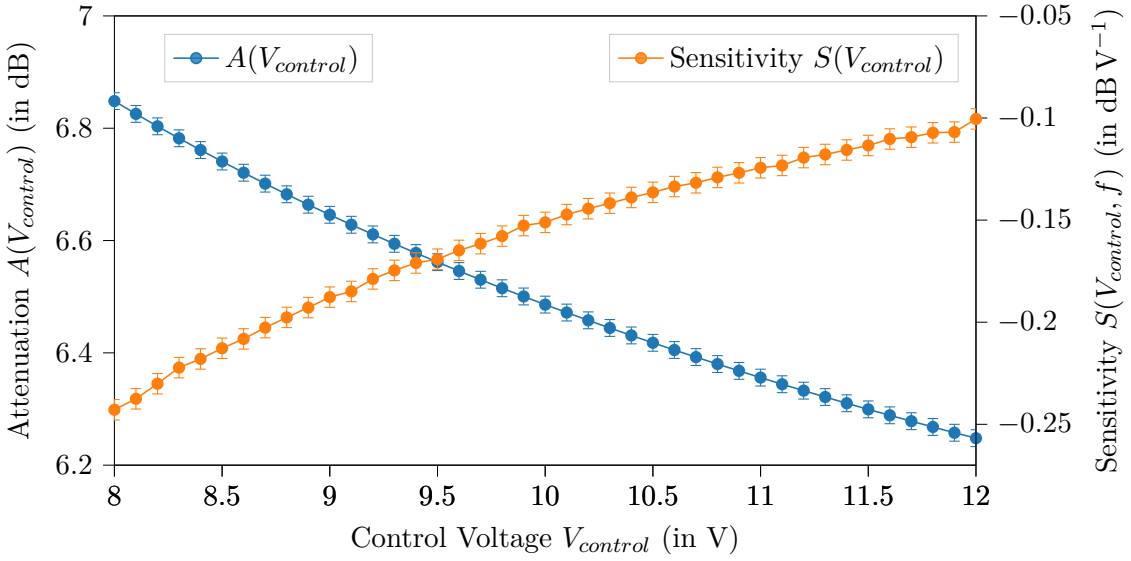



**Figure 4.8:** Zoomed in version of Figure 4.7 shows the attenuation and sensitivity around the operating point $V_{control,o} = 10\,\text{V}$

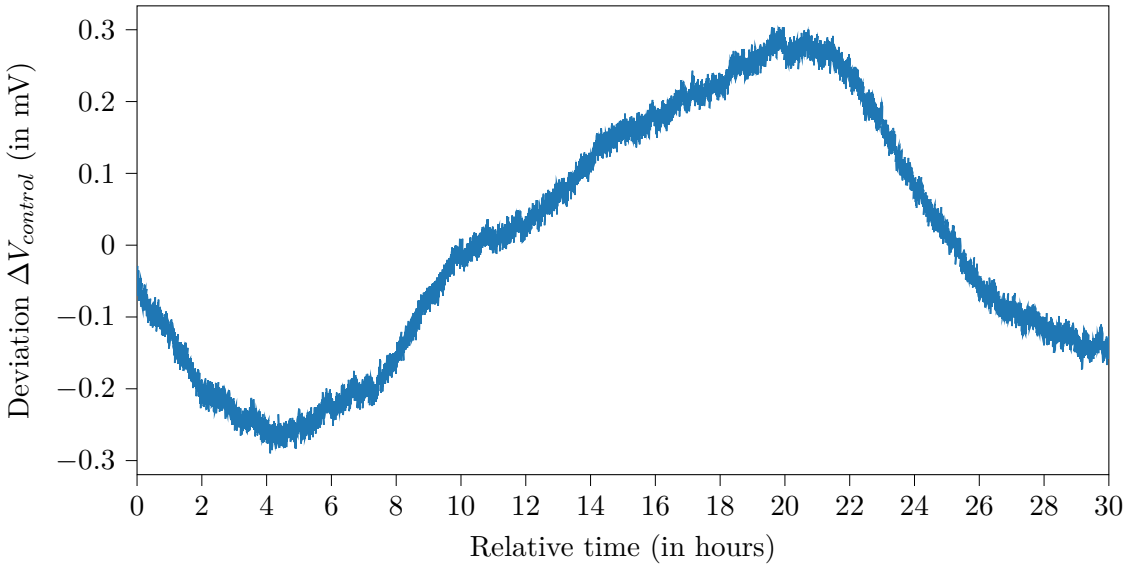



**Figure 4.9:** Long term stability of $V_{control}$ as delivered by the Keysight 34972A DAC (ch. 205); measured with Keysight 34470A;
room temperature during measurement: $\mu_\vartheta = 19.12\,°\text{C}$, $\sigma_\vartheta = 0.28\,°\text{C}$

#### 4.2.2.3 Influence of Supply Voltage Noise on Attenuation

To get the required stability for the power supply voltage, the effect of the supply voltage $V_+$ on the attenuation ${}^{\Delta A}/_{\Delta V_+}(V_+)$ has to be examined first. To do that $V+$ is varied $\pm 0.2\,\mathrm{V}$ around the nominal supply voltage at the operating point $V_{+_o} = 3\,\mathrm{V}$, all other parameters are kept constant and the attenuation is measured. To make the measurement more robust against fluctuations of the room temperature and drift of the devices, the procedure of stepping through the voltages is repeated in a similar fashion as for the influence of $V_{control}$ and the means for each set $V_+$ are computed. The result is shown in Figure 4.10.

The plot shows $A(V_+)$ to be of linear nature over the measured range. Therefore using a linear regression of the measured data points, the influence on the attenuation can be estimated to

$$\frac{\Delta A(V_+)}{\Delta V_+}(V_+) \approx \frac{\Delta A(V_+)}{V_+} = \left[\frac{\Delta A(V_+)}{V_+}\right]_{\max} = 0.003\,559\,2\,\mathrm{dB\,V^{-1}}. \tag{4.16}$$

Next the stability of the supply voltage is measured. for that the stability over the course of one day is measured. The voltage is taken once every 2 seconds with a Keysight 34470A multimeter (see subsubsection B.1.2). The result is shown in Figure 4.11.

Comparing Figure 4.11 with Figure 4.9 suggest a relation between the deviations in $V_{control}$ and $V_+$. Since they are both generated by the same Keysight 34907A module, this is plausible. The slightly changing room temperature is assume to be the common cause. By calculating the correlation coefficients between $\Delta V_+$ and $\Delta V_{control}$ and also between $\Delta V_+$ and $\Delta\vartheta_{ambient}$, this can be verified:

$$\mathrm{Corr}(\Delta V_+, \Delta V_{control}) = 0.992\,04 \tag{4.17}$$

$$\mathrm{Corr}(\Delta V_+, \Delta\vartheta_{ambient}) = -0.922\,42 \tag{4.18}$$

$$\tag{4.19}$$

The long term measurement yields a standard deviation of

$$\sigma_{V,+,longterm} = 0.154\,\mathrm{mV}, \tag{4.20}$$

which is also used as the worst case stability

$$\sigma_{V_+} = 0.154\,\mathrm{mV} \tag{4.21}$$

There is also a constant offset of $\mu_{V,+,longterm} = -1.35\,\mathrm{mV}$, bu it is disregarded since it can easily be compensated for it by slightly increasing the supply voltage.

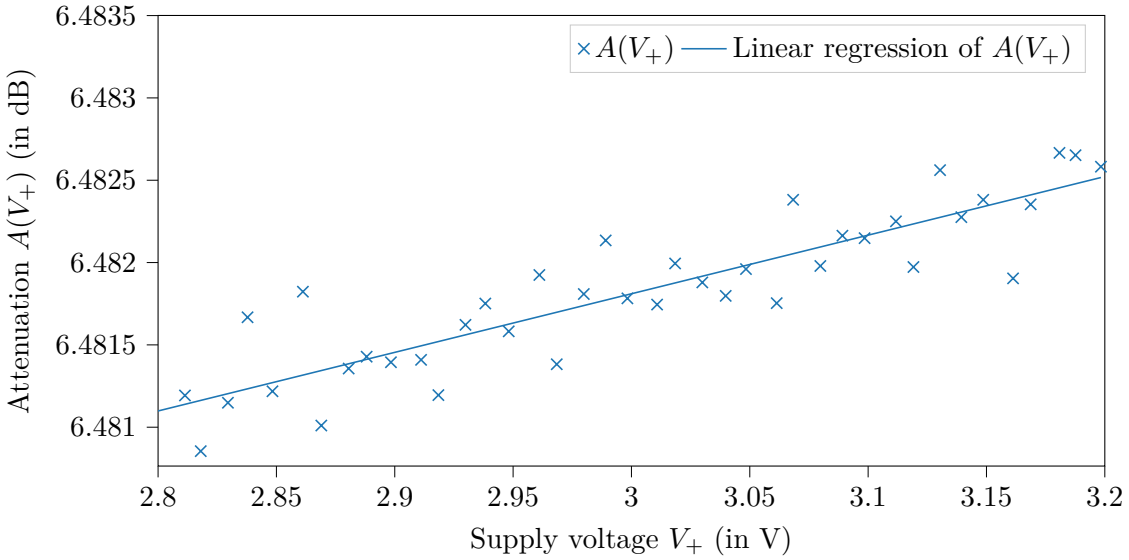



**Figure 4.10:** Influence of the supply voltage $V_+$ on the attenuation

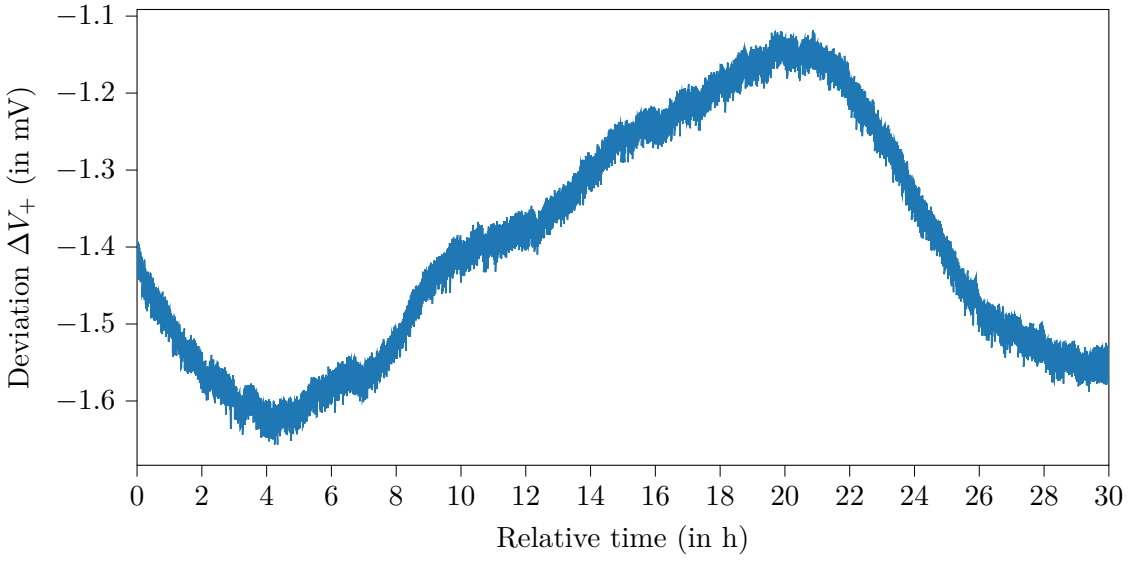



**Figure 4.11:** Long term stability of $V_+$ as delivered by the Keysight 34972A DAC (ch. 204); measured with Keysight 34470A;
room temperature during measurement: $\mu_\vartheta = 19.12\,°\mathrm{C}$, $\sigma_\vartheta = 0.28\,°\mathrm{C}$

#### 4.2.2.4 Influence of RF Frequency on Attenuation

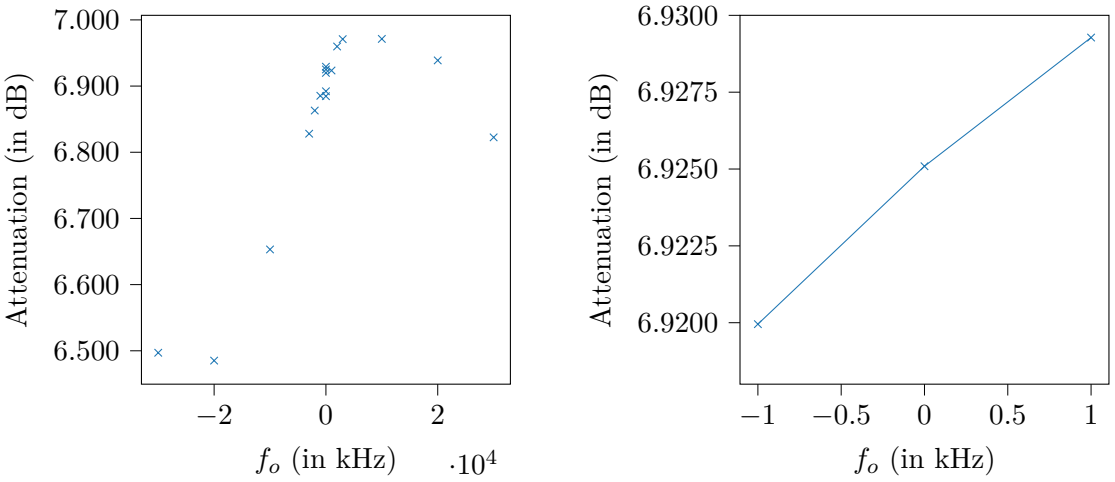
In this section the influence of a varying RF frequency on attenuation $\Delta A/\Delta f(f)$ is examined. For that the set frequency of the R&S SMC100 signal generator is varied while the attenuation is measured. The result is shown in Figure 4.12.

Since the stability of the FLUTE oscillator is measured using a phase noise analyzer to be

$$\sigma_f = 0\,\mathrm{Hz}, \tag{4.22}$$

it is sufficient to regard Figure 4.12 (b). In the range of $\pm 1\,\mathrm{kHz}$ around the operating point frequency of $3\,\mathrm{GHz}$, $\Delta A/\Delta f(f)$ is almost constant. From the data points, the value can be calculated to

$$\frac{\Delta A(f)}{\Delta f}(f) \approx \frac{\Delta A(f)}{\Delta f} = \left[\frac{\Delta A(f)}{\Delta f}\right]_{\max} = 0.000\,004\,661\,9\,\mathrm{dB\,Hz^{-1}}. \tag{4.23}$$




**(a)** $f = f_o \pm 30\,\text{kHz}$ **(b)** $f = f_o \pm 1\,\text{kHz}$

**Figure 4.12:** Influence of an offset frequency $f - 3\,\text{GHz}$ on the attenuation

#### 4.2.2.5 Influence of Case Temperature Variations on Attenuation

To get insight into the importance of a stable case temperature, its influence on the attenuation ${}^{\Delta A}/_{\Delta\vartheta_{case}}(\vartheta_{case})$ and the temperature stability both in the "RF lab'' and the final mounting position insight the FLUTE LLRF cabinet are measured.

To measure ${}^{\Delta A}/_{\Delta\vartheta_{case}}(\vartheta_{case})$, the following experimental setup is used.
The bottom of the attenuator is fixed to a rectangular iron profile with zip ties. Then the iron profile is heated with the tip of a soldering iron (set to 150 °C) for 1 min and then allowed to cool for 20 min. This cycle is repeated three times and the device temperature and the attenuation are measured once every 2 s. Due to heat flowing from the soldering iron to the iron profile and through device itself, a strong hysteresis between the heating and cooling cycle can be observed in Figure 4.13.

To get ${}^{\Delta A}/_{\Delta\vartheta_{case}}(\vartheta_{case})$ from the plot in Figure 4.13 an approximately linear relationship $A(\vartheta_{case})$ is assumed. To get the slope, the $\vartheta_{case}$ component of the measured data points $(\vartheta_{case}|A(\vartheta_{case}))$ are rounded to one decimal place. Then data points with the same $A(\vartheta_{case})$ are averaged together and a linear regression is applied to the result. See Figure 4.14 for the resulting plot. The linear regressor yields

$$\frac{\Delta A}{\Delta\vartheta_{case}}(\vartheta_{case}) \approx \frac{\Delta A}{\Delta\vartheta_{case}} = \left[\frac{\Delta A}{\Delta\vartheta_{case}}\right]_{\max} = 0.004\,324\,49\,\text{dB}\,^\circ\text{C}^{-1}. \tag{4.24}$$

Next, the temperature stability in both the "RF lab'' and in the FLUTE LLRF cabinet is measured. For that in each environment the temperature is recorded over night with a thermo element connected to the Keysight 34907A inside the Keysight 34972A every two seconds.

From the data in Figure 4.15, the corresponding stabilities are calculated to be

$$\sigma_{\vartheta_{ambient,\text{RF lab}}} = 0.1067\,^\circ\text{C} \tag{4.25}$$

$$\sigma_{\vartheta_{ambient,\text{LLRF cabinet}}} = 0.0288\,^\circ\text{C}. \tag{4.26}$$

This assumes the case of the attenuator and the ambient around it being in thermal equilibrium. With the thick brass case and its only weakly mechanically mounting to the metal support, the attenuator posses a significant thermal time constant $\tau_{th} = R_{th}C_{th}$.

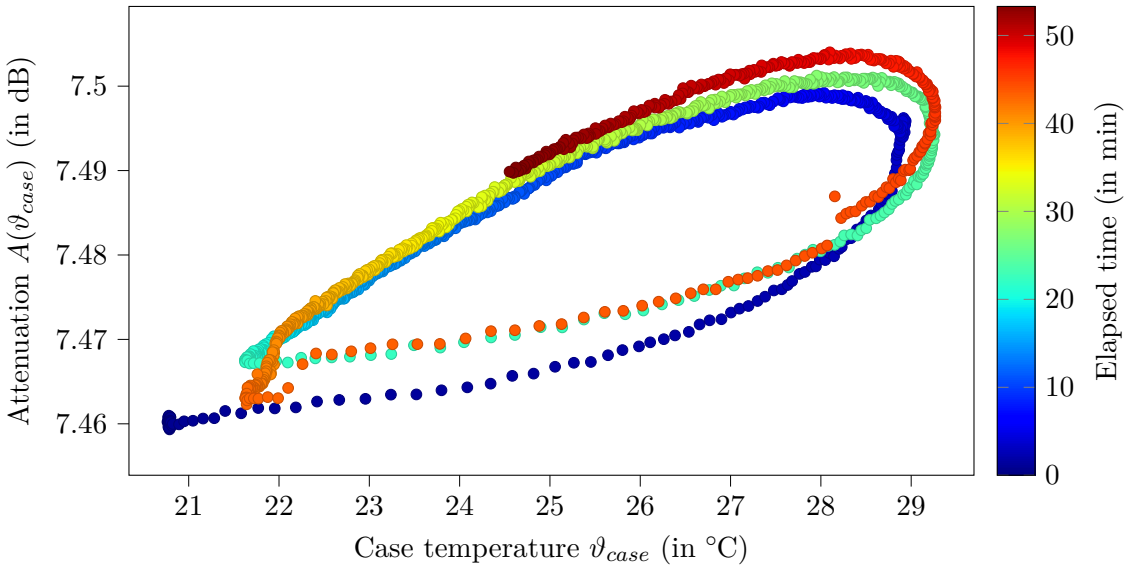



**Figure 4.13:** Raw measurement result of the influence of the case temperature $\vartheta_{case}$ on the attenuation; color coded is the relative elapsed time of the measurement

Therefore, thermally the attenuator is a lowpass filter and cannot follow fast changes in the ambient temperature. This means only slow changes of the ambient temperature have an influence on $A$, so short changes, like opening a door, have only a minor effect.

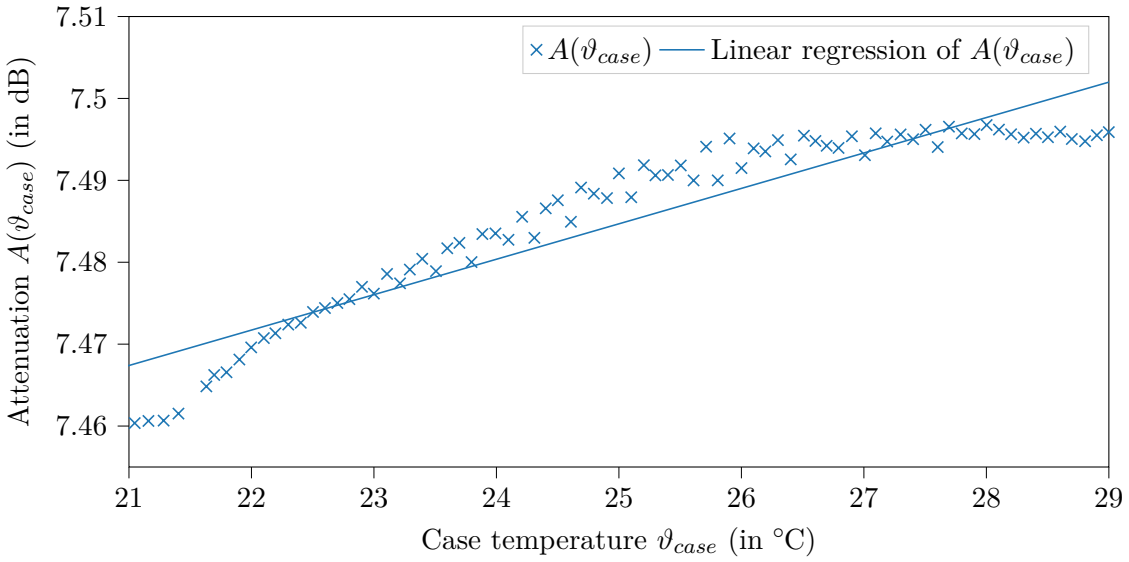



**Figure 4.14:** Influence of the case temperature $\vartheta_{case}$ on the attenuation

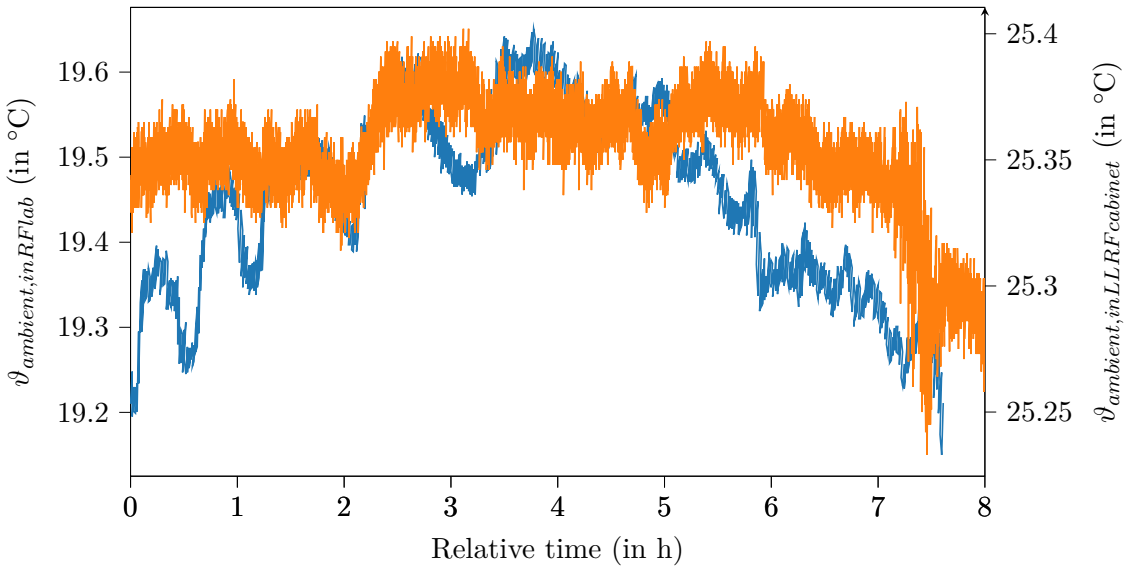



**Figure 4.15:** Comparison between the temperature stability in the "RF lab" and inside the FLUTE LLRF cabinet

#### 4.2.2.6 $V_{control}$ Frequency response

Using a non inverting adder with a *TS912IN* operational amplifier, a sine wave with constant offset is made, which is then used to drive the control voltage $V_{control}$. The result for different sine wave frequencies is shown in Figure 4.16.

This result is to be interpreted purely qualitatively, as neither measurement device and software is designed, nor is the circuit used optimal. But the traces in Figure 4.16 verify that the ZX73-2500-S+ attenuator is able to follow changes of $V_{control}$ at least up to a few ten kHz. If the attenuator should be set to a new value, this is equivalent to applying a step at the attenuators $V_{control}$ input, which according to the Fourier transform of a rect($t$) pulse, a sinc($f$) contains high frequencies.

This is at several orders of magnitude bigger than the control system can input, compute and output new values to the attenuator.

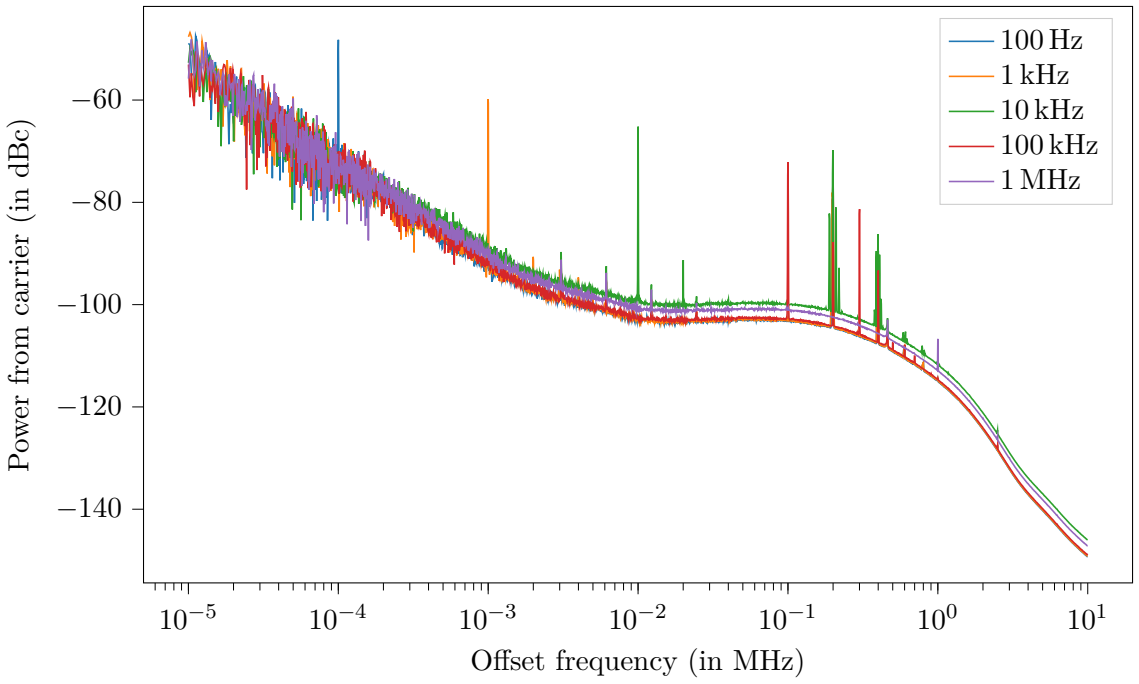



**Figure 4.16:** Spectrum (measured with Holzworth HA7062A (subsubsection B.7.1)) showing the effect of modulating $V_{control}$ with different frequencies (Modulation amplitude: 1 V)

#### 4.2.2.7 Combined Maximum Error of the Attenuation and Conclusion

Using Equation 4.4 and setting $\Delta x_j = \sigma_{x_j}$ and ${}^{\Delta A}/_{\Delta x_j} = \left[{}^{\Delta A}/_{\Delta x_j}\right]_{\max}$, the upper bound on the deviation of the attenuation, that is the error $\Delta A$ in the worst case, can now be calculated:

$$\Delta A_{\max} = \sum_{j=0}^{3} \left| \frac{\Delta A}{\Delta x_j} \cdot \Delta x_j \right| \tag{4.27}$$

$$= \left| \frac{\Delta A}{\Delta V_{control}} \cdot \Delta V_{control} \right| + \left| \frac{\Delta A}{\Delta V_{+}} \cdot \Delta V_{+} \right| + \left| \frac{\Delta A}{\Delta f} \cdot \Delta f \right| + \left| \frac{\Delta A}{\Delta \vartheta_{case}} \cdot \Delta \vartheta_{case} \right| \tag{4.28}$$

$$= \underbrace{0.242\,\mathrm{dB\,V^{-1}} \cdot 0.173\,\mathrm{mV}}_{41.9\,\mu\mathrm{dB}} \tag{4.29}$$

$$+ \underbrace{0.003\,559\,2\,\mathrm{dB\,V^{-1}} \cdot 0.154\,\mathrm{mV}}_{548\,\mathrm{ndB}} \tag{4.30}$$

$$+ \underbrace{0.000\,004\,66\,\mathrm{dB\,Hz^{-1}} \cdot 0\,\mathrm{Hz}}_{0\,\mathrm{dB}} \tag{4.31}$$

$$+ \underbrace{0.004\,325\,\mathrm{dB\,°C^{-1}} \cdot 0.11\,\mathrm{°C}}_{475.75\,\mu\mathrm{dB}} \tag{4.32}$$

$$= 518\,\mu\mathrm{dB} = 0.000\,518\,\mathrm{dB} \tag{4.33}$$

This shows even in the worst case, the ZX73-2500-S+ attenuator is stable within the requirement.

The other requirements from Table 4.2 are also fulfilled, see Table 4.3.

**Table 4.3:** Controllable RF attenuator evaluation test results

| Requirement | Value/Range set | Value/Range actual | Verdict |
|---|---|---|---|
| attenuation adjustment | $\geq \pm 0.2\,\mathrm{dB}$ | $\pm 0.3\,\mathrm{dB}$ | pass |
| attenuation stability | $\leq \pm 0.001\,\mathrm{dB}$ | $0.000\,518\,\mathrm{dB}$ | pass |
| nominal attenuation at operating point | $\leq 10\,\mathrm{dB}$ | $6.6\,\mathrm{dB}$ | pass |
| attenuation resolution | $\leq 0.001\,\mathrm{dB}$ | $0.000\,089\,\mathrm{dB}$ | pass |
| setup time | $\leq 10\,\mathrm{ms}$ | $1\,\mathrm{ms}$ | pass |
| operating temperature range | $(25.0 \pm 0.1)\,^{\circ}\mathrm{C}$ | $(25.0 \pm 0.1)\,^{\circ}\mathrm{C}$ | pass |
| supply voltage | $3\,\mathrm{V}$ to $12\,\mathrm{V}$ | $3\,\mathrm{V}$ | pass |
| control voltage | $3\,\mathrm{V}$ to $12\,\mathrm{V}$ | $8\,\mathrm{V}$ to $12\,\mathrm{V}$ | pass |

### 4.2.3 Test of the Attenuator with FLUTE

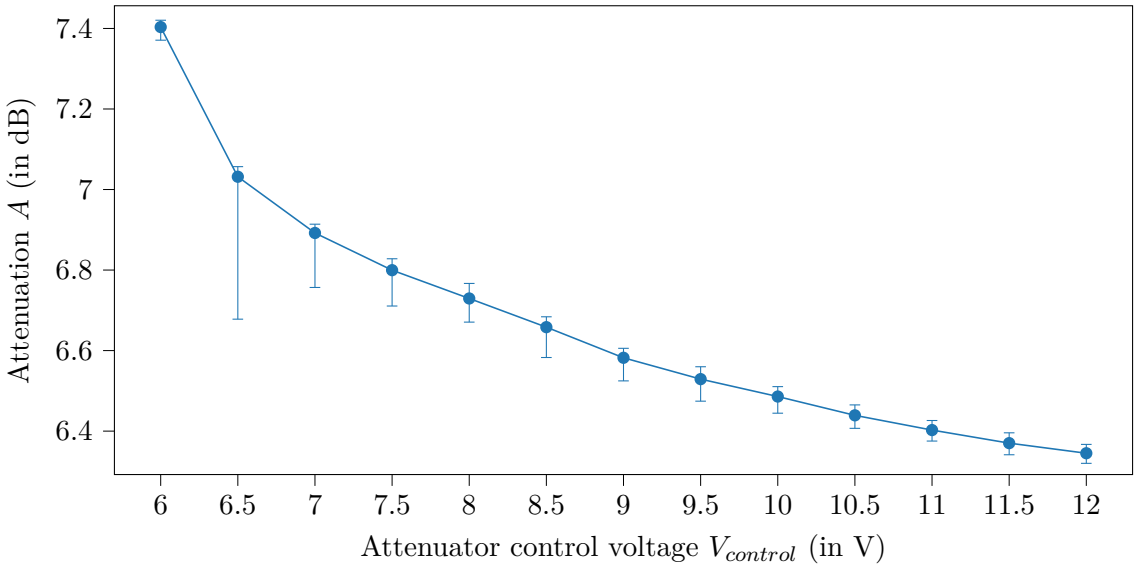

7.4
7.2
7
6.8
6.6
6.4
Attenuation $A$ (in dB)
6
6.5
7
7.5
8
8.5
9
9.5
10
10.5
11
11.5
12
Attenuator control voltage $V_{control}$ (in V)

**Figure 4.17**

# 5. Controller Design and Evaluation

In this chapter a control system is designed and evaluated to stabilize FLUTEs RF system. Referring back to the block diagram of a generic control system in Figure 2.1, there are three blocks to determine. First the plant transfer function, which describes the system that is to be controlled. Second, based on the plant, an appropriate controller type is chosen and its parameters are calculated. Third a measurement filter is used to improve the quality of the feedback signal path. As the choice of the measurement filter influences the controller design, its design is treated before the controller.

## 5.1 Plant Identification

### 5.1.1 Principle

Before choosing an appropriate controller, some insight of the system response has to be obtained. Therefore in this section the plants transfer function is estimated. In the context of this chapter "plant'' refers to everything from the attenuation set at the controllable attenuator to the system output, e.g. the cavity RF power.

In the time domain, a LTI system is described by its impulse response $h(t)$, that is the reaction of the plant to an impulse at the input. Using this definition directly, the plants impulse response $p(t)$ could be measured by applying a short peak in the attenuation setting on the attenuator. The effect on the output is not easy to measure and a single measurement of this kind is very susceptible to noise. Therefore it is more common to measure the step response[20], which is the output of a system, when a step function is applied to its input. As the step function is the time integral of the impulse function, the step response can be converted to the impulse response by differentiation in time.

Instead of measuring a single step response, often several step responses are measured and their average is computed to reduce the variance of the estimation. When measuring a step response the minimum needed measuring time depends on the systems time constants, but they are often not known beforehand.

That is why when there is no prior knowledge of the system, the identification is sometimes done with a (pseudo) random binary sequence to excite the system with step functions of different lengths. The PRBS is chosen in a way that that some of the steps will probably last longer than a few dominant time constants of the system.

To get the transfer function $P(s) = \mathcal{L}\{p(t)\}$ of the plant from its step response(s), several methods are common, including correlation based and frequency response based algorithms.

### 5.1.2 Identifying the Plant Attenuator+RF

The input PRBS signal is generated with the Python script in Listing 5.1. Based on the value of a pseudo random number generator, the sequence toggles the attenuator between $V_{\text{control}} = 7\,\text{V}$ and $V_{\text{control}} = 7\,\text{V}$. Using Figure 4.7, this equals a span in attenuation of

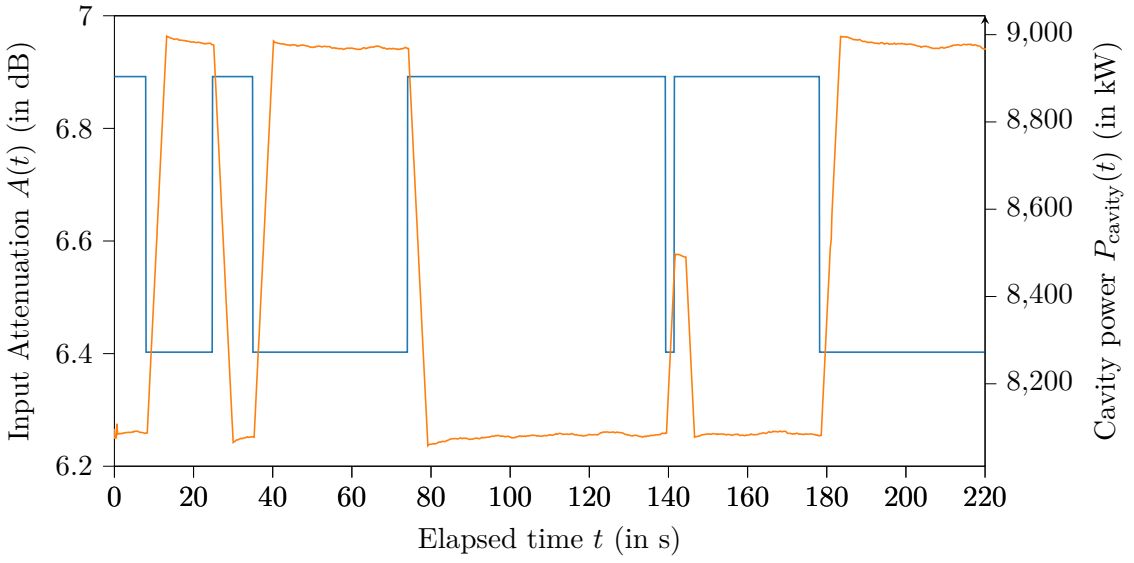



**Figure 5.1:** Section of the input sequence (blue) and the system response (orange); Note the inverse relation: A higher attenuation $A$ causes a lower cavity power $P_{\text{cavity}}$

6.892 dB to 6.4026 dB. With the parameter `toggleP`, the average length of one constant voltage level can be controlled.

**Listing 5.1:** Function to get a pseudo random binary sequence

```
1 def randomBinarySequence(N,toggleP):
2     u=[False]*N
3     for i in range(1,len(u)):
4         if(np.random.binomial(1,toggleP,1)[0]):
5             u[i]=not u[i-1]
6         else:
7             u[i]=u[i-1]
8     return list(map(lambda x: 7 if x==False else 11,u))
```

In a test run over six hours (after all FLUTE subsystems had stabilized), the attenuator was driven with such a PRBS. The result is shown in Figure 5.1.

The time signals $A(t)$ and $P_{\text{cavity}}(t)$ are then split into a *estimation* data set (about 80 % of the samples) and a *validation* data set (the remaining $\approx$ 20 %). This is done so the bulk of the available information is used to estimate the model, but there is data left that the model hasn't seen before. This smaller portion is used to validate the models performance, hence it is called the validation data set.

The two data sets are then loaded into the MATLAB *System Identification Toolbox*. With the toolboxes pre-processing tools, first the means of both the input and output are removed, which is required by the estimators used. Then, using the "process model" estimator, linear, continuous time transfer functions with different numbers of zeros and poles are estimated (See screenshot of the GUI in Figure A.1). After that, to check the accuracy of the estimations, the System Identification Toolbox is used to simulate the output of the estimated systems (see Figure 5.2). For that the aforementioned validation data set is used: The measured output is compared with the outputs predicted by the models.

Using the Matlab function `zpk()` the models are converted into the zero-pole-gain representation. In Table 5.1 the estimated models are listed with their zeros, poles, gains and the model fit, as it is computed by the System Identification Toolbox. `P1`, `P2` and `P3` are models with one, two and three poles and a gain as free parameters. `P2ZU` consists of a complex pole pair, a zero and a gain.

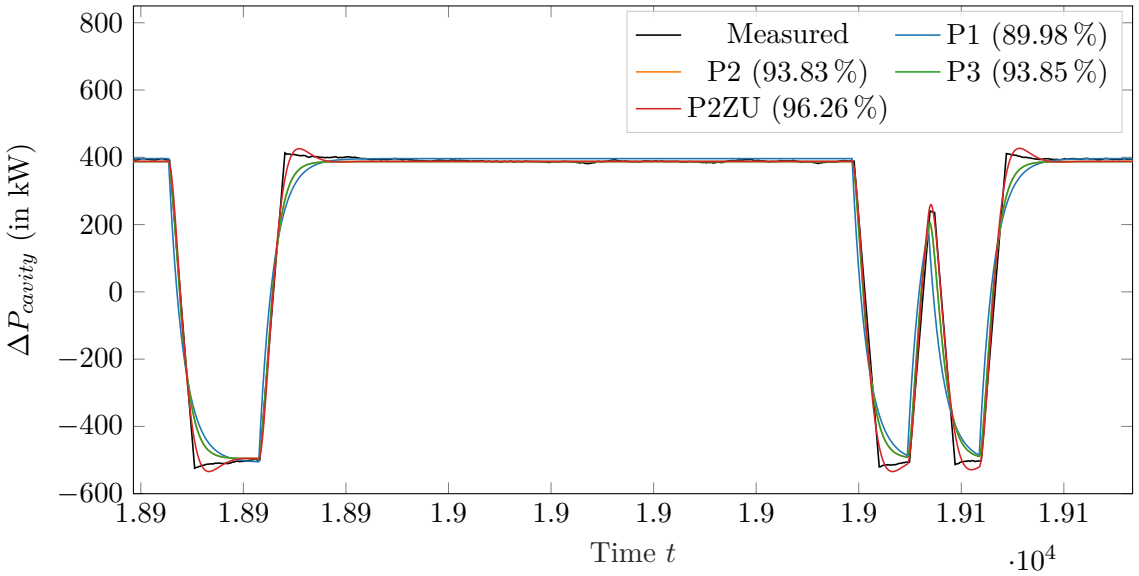



**Figure 5.2:** Validation of the estimated process models for the plant; the legend also shows the model fits in percent

**Table 5.1:** Process models of the plant as estimated by the Matlab System Identification Toolboxes process model estimator

| Model name | Zeros | Poles | Gain | Model fit |
|---|---|---|---|---|
| P1 | | $(s + 0.3505)$ | $-646.94$ | 89.98 % |
| P2 | | $(s + 0.6875)(s + 0.7039)$ | $-873.18$ | 93.83 % |
| P3 | | $(s + 1 \times 10^6)(s + 0.7376)(s + 0.669)$ | $-8.9019 \times 10^8$ | 93.85 % |
| P2ZU | $(s - 2846)$ | $(s^2 + 0.8014s + 0.3195)$ | 0.20296 | 96.26 % |

Figure 5.2 and Table 5.1 show the `P2ZU` model to have the best fit. Therefore it is accepted as the plants transfer function. Using the Matlab function `tf()`, its time continuous transfer function can be stated as

$$P(s) = \frac{0.6352s - 1808}{3.13s^2 + 2.508s + 1}. \tag{5.1}$$

Using the Matlab function `c2d()`[1] and the sample time $T_s = 0.2\,\mathrm{s}$, $G(s)$ can be converted to the time discrete transfer function

$$P[z] = \frac{-10.91z^{-1} - 10.41z^{-2}}{1 - 1.84z^{-1} + 0.8519z^{-2}}. \tag{5.2}$$

The high model fit percentage of 96.26 % justifies the choice of a linear model with few parameters and further estimation attempts using other (non-) linear models are not needed. Considering the accuracy of the model, it is important to note this estimation is only valid at the time the measurements are taken. With FLUTE being a large experimental setup being still under commission, it is always possible that small changes to certain (sub-) systems can lead to minor or major influences to others. For the time being, the estimation is redone a day later and for both measurements days, different parts of the about six hour measurements each are used as the estimation and validation data sets. Doing so shows no significant change in the estimated coefficients and the resulting model fits, indicating the estimated models are at least plausible.

[1] Without specifying a different method, `c2d()` discretizes the continuous-time model zero-order hold on the input.

To account for errors in the estimated model (and possible other errors), when designing the controller, sufficient gain and phase margins are set to ensure stable operation.

## 5.2 Measurement Filter

Like all measurements of physical quantities, the measuring of the system output of the control system is subjected to noise. In addition to these disturbances of thermal or electrical origins, also high frequency variations of the system output has detrimental effects on the control systems performance. For example the magnitudes of the bunch-by-bunch changes of the measured cavity power are often in the same order as the long-term drifts. Trying to correct for them instead of the long term drifts often leads to overcompensating and can even make the system unstable.

To remove the high frequency components a low pass filter is used as the measurement block $H(s)$.

In pre-tests the incoming signal was simply filtered with a moving average filter. Commonly, the moving average is defined as the mean of a signal $x$ inside a window of length $L$, centered around the current time or sample index $n$, that is shifted along the signal. This smooths out small variations thus the moving average acts as a low pass filter. This *non-causal* version of the moving average can only be used with already measured data as to compute the moving average at $n$, future values at $n + i$ are needed:

$$\mathrm{MA}_{x,\text{non-causal},L}[n] = \frac{1}{L} \sum_{i=n-\frac{L-1}{2}}^{n+\frac{L-1}{2}} x[i] \tag{5.3}$$

When filtering real-time data, future values are not available and a shifted, *causal* version, of the moving average

$$\mathrm{MA}_{x,\text{causal},L}[n] = \frac{1}{L} \sum_{i=n-(L-1)}^{n} x[i] \tag{5.4}$$

is used.

In case of the cavity RF power, experiments show a window length of about $L = 100$ or more is necessary to sufficiently smooth the measured power signal. When comparing the original signal with the filtered one, it is apparent that in addition to the desired smoothing effect, the filtered signal also is delayed in time, with the delay being dependent on the window size. To quantify the delay, the alternative definition of the moving average as a digital FIR filter is used. One possibility to describe a FIR filter is by giving its impulse response, i.e. the output signal when the input of the filter is an impulse with unity height. In case of the moving average filter, the coefficient sequence of the corresponding FIR filter has the length $N = L$ and is equal to the the impulse response $h[n]$:

$$h[n] = \frac{1}{N}[\underbrace{1,\ 1,\ ...,\ 1}_{N}] \tag{5.5}$$

The delay introduced by a digital filter can be quantified with the filters group delay

$$\frac{\tau_g(f)}{T_s} = \frac{\mathrm{d}\phi(f)}{\mathrm{d}f} \tag{5.6}$$

which is given normalized to the sampling time $T_s$ [21, p. 70]. In case of a FIR filter with linear phase (with a symmetrical impulse response), the group delay is always constant over all frequencies and is only dependent on the filter length $N$[21, p. 165]:

$$\frac{\tau_g(f)}{T_s} = \frac{\mathrm{d}\phi(f)}{\mathrm{d}f} = \frac{\mathrm{d}\phi}{\mathrm{d}f} = \frac{N}{2} \tag{5.7}$$

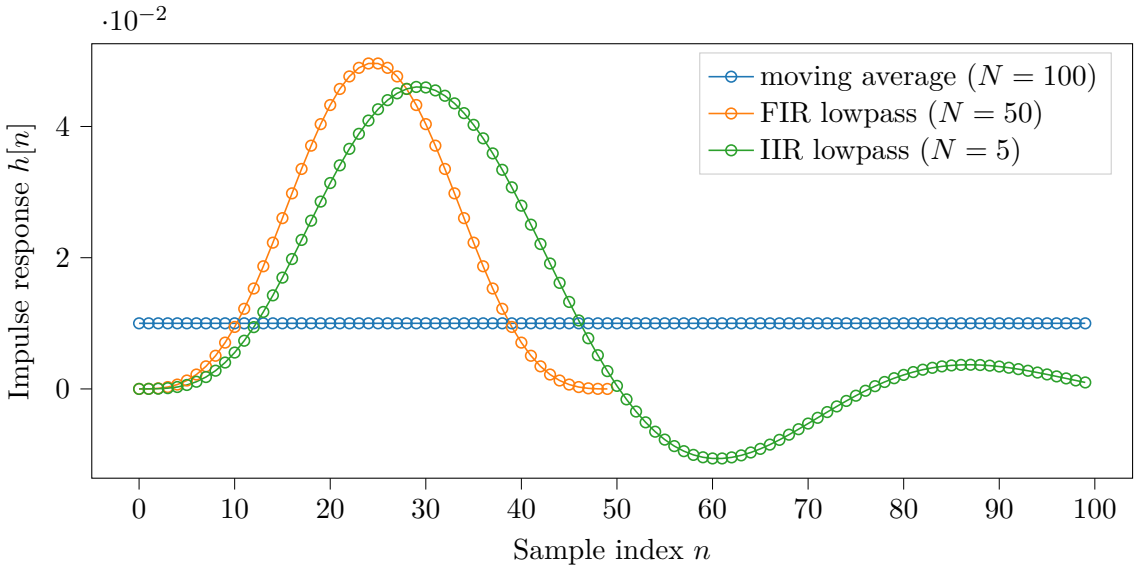



**Figure 5.3:** Impulse responses of a moving average filter ($N = 100$), a FIR lowpass ($N = 50$, $f_c = 0.1\,\text{Hz}$) and a IIR Butterworth lowpass ($N = 50$, $f_c = 0.1\,\text{Hz}$)

With a sampling time of $T_s = 1/5\,\text{Hz} = 200\,\text{ms}$ and $N = 100$ the group delay is 10 s. In case of a steady operation this is acceptable, as the disturbances to compensate happen on a timescale in the order of several minutes. But in case of ongoing transients in case of user changes to the control system parameters, or short error bursts on the measured signal, this long delay causes problems and therefore should be reduced.

Therefore a more sophisticated digital filter is designed to replace the simple moving average.

On the one hand, a FIR filter is designed with the Kaiser window method. This method starts with the desired frequency response, which is usually given piece-wise. In case of the low pass filter it is a step function at a cutoff frequency $f_c$. Then the IDFT is used to compute the corresponding impulse response $h_{\text{IIR}}[n]$, which is in general infinitely long. Windowing with e.g. a Kaiser window and then truncating the impulse response then yields the impulse response of the desired FIR filter $h_{\text{FIR}}[n]$.[22, p. 533]. With SciPy using `b=signal.firwin(N, fc,fs)`, the coefficients of a FIR with this method can be calculated.

On the other hand, an IIR filter is designed with the impulse invariance method and an analogue Butterworth filter. This method could be interpreted as sampling the infinitely long analogue impulse response.[22, p. 497] With SciPy using `b,a=signal.butter(N,fc,'lowpass',fs,)`, the coefficients of an IIR with this method can be calculated. For an IIR filter, the group delay cannot be calculated with Equation 5.7 and it is in general frequency depend.

Figure 5.3 shows the impulse responses of the moving average, the FIR lowpass and the IIR lowpass (truncated to $N = 100$).

In Figure 5.4, the three filter types described above are compared by filtering a ten minute long segment of pre-recorded data. The filtering is done with the SciPy function `signal.lfilter()` which does causal filtering and does not compensate group delay[2], so the results are the same as they would be for real-time data.

The plot shows the FIR lowpass filter requiring ten times the number of coefficients to achieve about the same result as the IIR lowpass filter. Also the moving average filter

[2]In contrast to `signal.filtfilt()`, which applies the filter both forward and backward achieving zero phase/group delay, but this cannot be done for incoming real-time data.

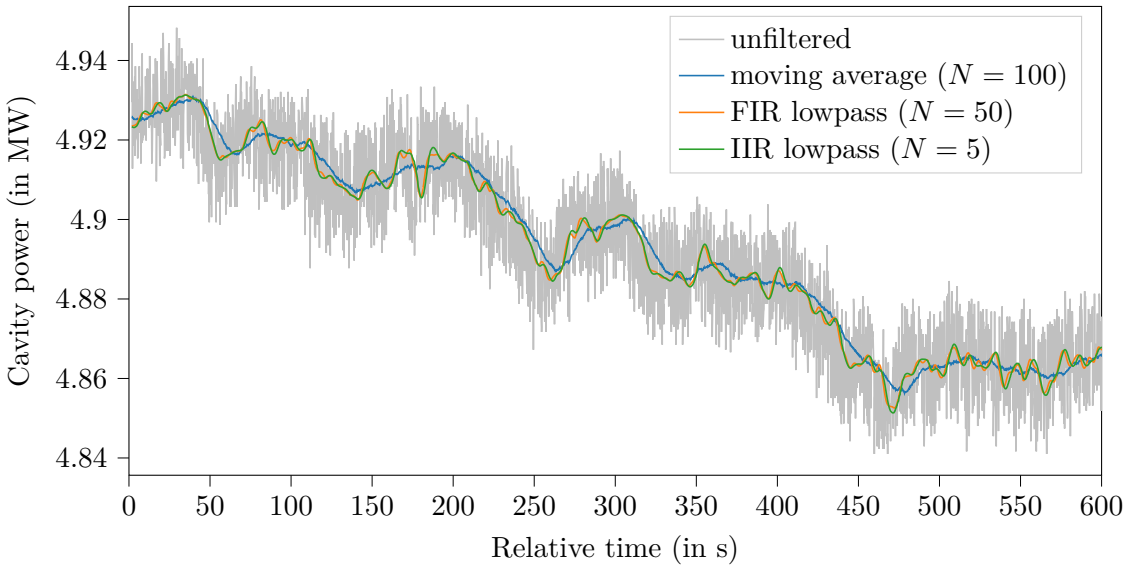



**Figure 5.4:** Effects of the three different lowpass filters in Figure 5.3 on noisy data

has double the number of coefficients as the FIR lowpass filter, but there is still high frequency noise in the output (caused by the sinc(·) shape of its frequency response $H[f] = \mathrm{DFT}\{h[n]\}$).

Compared to the FIR lowpass, the moving average offers no benefit besides its easy implementation. When comparing the FIR with the IIR approach, the IIR has the advantage of needing less coefficients, thus occupying less memory, which is not really an advantage when the control system is implemented on a personal computer, which typically has enough free memory to hold millions of floating point numbers. Also the IIR filter has a non-constant group delay and is not guaranteed to be stable like all FIR filters are.

For these reasons, in the following a FIR lowpass filter is used.

One example filter generated with `signal.firwin()` with a cutoff frequency $f_c = 10\,\mathrm{mHz}$ and order $N = 10$ has the transfer function

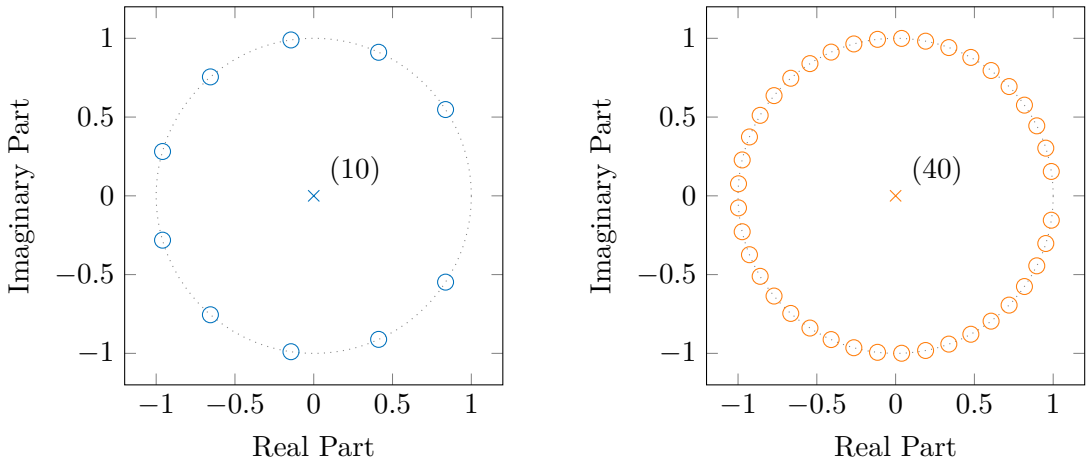
$$H[z] = \frac{1}{b_{10}z^{10} + b_9z^9 + b_8z^8 + b_7z^7 + b_6z^6 + b_5z^5 + b_4z^4 + b_3z^3 + b_2z^2 + b_1z + b_0} \tag{5.8}$$

with the coefficient vector $\vec{b} = [b_{10}, ..., b_0]$, with

$$\vec{b} = [0.0876, 0.0896, 0.0911, 0.0922, 0.0929, 0.0931, 0.0929, 0.0922, 0.0911, 0.0896, 0.0876]. \tag{5.9}$$

Instead of stating the transfer function, a plot of the poles and zeros is a more intuitive representation. In Figure 5.5, the poles and zeros of the FIR filter with $N = 10$ are compared to one with $N = 40$. In addition, Figure 5.6 shows the magnitude responses of such filters.

For the controller design in the next section and the later implemented real-time system, the cutoff frequency and the filter order are kept variable to leave room for improvement.




**(a)** $N = 10$ **(b)** $N = 40$

**Figure 5.5:** Pole-Zero maps for two FIR filters with a common cutoff frequency $f_c = 10\,\text{mHz}$ but different filter orders $N$;
$\circ$ denotes zeros, $\times$ denotes poles, $(k)$ is a $k$-times pole

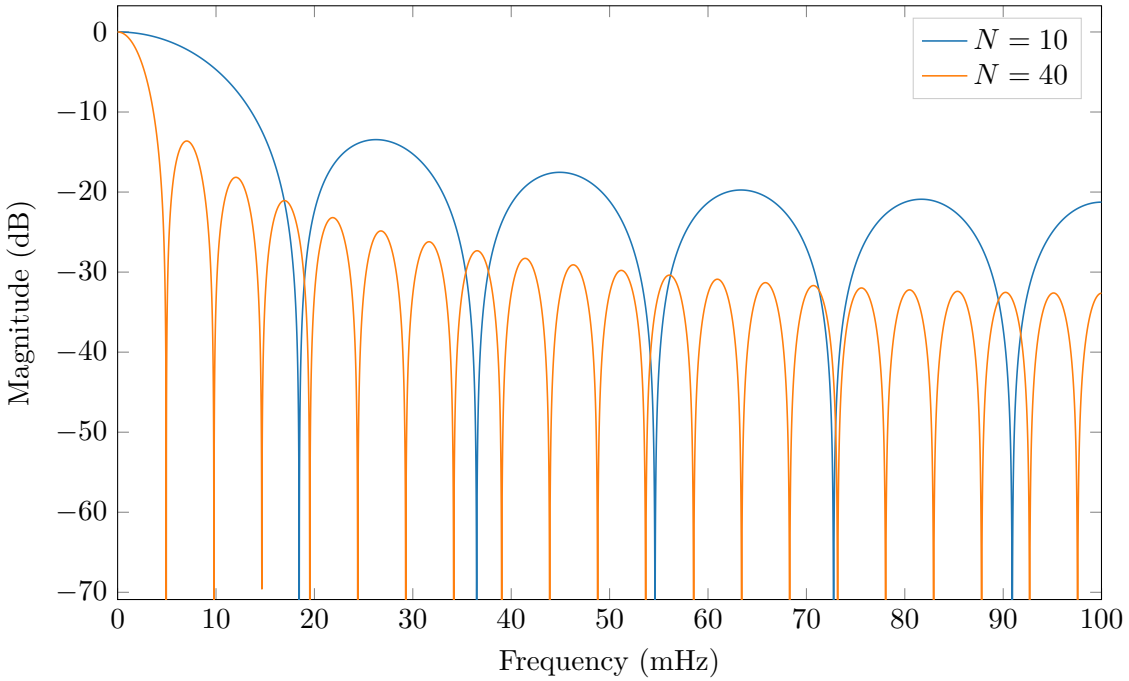



**Figure 5.6:** Magnitude response of two FIR filters with a common cutoff frequency $f_c = 10\,\text{mHz}$ but different filter orders $N$

### Real-time implementation of a FIR Filter in Python

Similar to Equation 5.4, a FIR filter designed with the SciPy function `sigal.firwin()` can be used in a causal manner to filter real-time data.

Applying the filter on pre-recorded data, like in Figure 5.4 can be done with `signal.lfilter()`. To use `signal.lfilter()` on sample-wise incoming real-time data, the "initial in'' input and "final out'' output of `signal.lfilter()` can be used to keep the filters state. This is demonstrated in Listing 5.2 by looping through pre-recorded data point-by-point.

**Listing 5.2:** Demonstration of the `zi` and `zf` variables when using `signal.lfilter()`

```
1 x=df2["F:RF:LLRF:01:GunCav1:Power:Out Value"].to_numpy()
2 y=np.array([])
3 zf=signal. lfilter_zi (b, 1)
4 for i in range(len(x)):
5     y0,zf=signal. lfilter (b, 1, [x[i ]], zi=zf)
6     y=np.append(y,y0[0])
```

## 5.3 Controller Design

Based on the estimated model of the plant $P(s)$ (or $P[z]$) in section 5.1 and the type of measurement filter designed in section 5.2, in this section an appropriate controller to stabilize the plant is designed and its performance is evaluated.

Tuning the controller and testing its capabilities is performed both *offline* with simulations using the building blocks $P[z]$, $H[z]$ and the yet to be defined $G[z]$ and *online* using a software implementation of the control system with the real hardware, i.e. FLUTE and the controllable attenuator.

### 5.3.1 Choosing a Controller Type

The plant has been identified using a linear model in section 5.1 resulting in the LTI system $P(s)$ or $(P[z])$. The matching controller does not necessarily have to be a LTI system, but choosing a LTI system simplifies design and analysis and is a justifiable choice here, as there a no good reasons against it, yet.

The class of LTI controllers is dominated by the PID controller and its variants. PID stands for "proportional'', "integral'' and "derivative'', which are LTI systems themselves, performing scaling, integration or differentiation. Depending on the application, variants, such as the PI controller, or the PD controller are used as well.

According to [23, p. 111] PID control is applicable for plants with order two or less. Compared to other types of controllers, using a PID controller has many advantages. It is is quick to design and does not depend on an accurate plant model. Also after the designing step, the few free parameters can easily be presented to an operator and online fine tuning is possible.[3]

In the next section such a PID controller is designed. As it is to be implemented in Software later, the design is done in discrete time.

### 5.3.2 Designing a Discrete Time PID Controller

The output signal $u(t)$ of a generic time continuous PID controller (see Figure 5.7) consists of the weighted sum of three error signals.

First, a The first is simply the controller input error signal $e(t)$ scaled by the gain $k_p$. This is the proportional part of the PID controller. The integral part is calculated by computing

[3] In contrast a compensating controller is based on an accurate model of the plant and changes on the fly are difficult.[24]

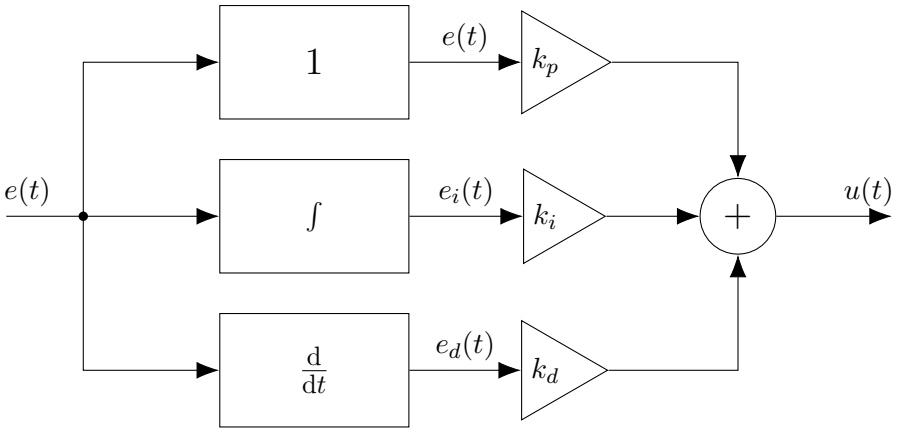



**Figure 5.7:** Block diagram of a generic PID controller

the running time integral

$$e_i(t) = \int_0^t e(\tau)\mathrm{d}\tau. \tag{5.10}$$

This is than scaled by the constant $k_i$. To get the derivative part, the derivative

$$e_d(t) = \frac{\mathrm{d}}{\mathrm{d}t}e(t) \tag{5.11}$$

is calculated and weighted with $k_d$. All three are then summed to get $u(t)$ in the so called parallel form[25, p. 5]:

$$u(t) = k_p e(t) + k_i \int_0^t e(\tau)\mathrm{d}\tau + k_d \frac{\mathrm{d}}{\mathrm{d}t}e(t) \tag{5.12}$$

Often instead of the parallel form, the PID controller is stated in *standard form*. Instead of using the gains $k_{p,\,i,\,d}$, the parameters proportional gain $K$, integral time $T_i$ and derivative time $T_d$ are used.[23, p. 76] With the conversions

$$K = k_i \tag{5.13}$$

$$T_i = \frac{k_p}{k_i} = \frac{K}{k_i} \tag{5.14}$$

$$T_d = \frac{k_d}{k_p} = \frac{k_d}{K} \tag{5.15}$$

the PID controller in standard form is

$$u(t) = K\left[e(t) + \frac{1}{T_i}\int_0^t e(\tau)\mathrm{d}\tau + T_d\frac{\mathrm{d}}{\mathrm{d}t}e(t)\right]. \tag{5.16}$$

The transfer function $G(s)$ is given by the Laplace of $u(t)$ using the computation rules[26]

$$\mathcal{L}\left\{\int_0^t y(\tau)\mathrm{d}\tau\right\} = \frac{1}{s}Y(s) \tag{5.17}$$

$$\mathcal{L}\left\{\frac{\mathrm{d}}{\mathrm{d}t}y(t)\right\} = sY(s) \tag{5.18}$$

as

$$G(s) = K\left[1 + \frac{1}{sT_i} + sT_d\right]. \tag{5.19}$$

To get the discrete transfer function, either the Laplace transform $G(s) = {}^{U(s)}/_{E(s)}$, is discretized or the $\mathcal{Z}$ transform of $u[n]$ is calculated. First, to get $u[n]$, the derivative in Equation 5.11 is approximated by

$$e_d[n] = \frac{e[n] - e[n-1]}{T_s}. \tag{5.20}$$

This assumes $e(t)$ is sampled with a sampling rate of $f_s = {}^1/_{T_s}$ to get $e[n]$. In a similar fashion, the integral in Equation 5.10 is approximated by

$$e_i[n] = e_i[n-1] + e[n]T_s. \tag{5.21}$$

Using the shift rule of the $\mathcal{Z}$ transform[26]

$$y[n-k] \circ\!\!-\!\!\bullet z^{-k}Y[z], \tag{5.22}$$

the discrete transfer function of the PID controller is

$$G[z] = K\left[1 + \frac{T_s}{T_i}\frac{z}{z-1} + \frac{T_d}{T_s}\frac{z-1}{z}\right]. \tag{5.23}$$

In the time domain, this becomes

$$u[n] = k_p e[n] + k_i\left(\underbrace{e_i[n-1] + T_s e[n]}_{e_i[n]}\right) + k_d\left(\underbrace{\frac{1}{T_s}e[n] - e[n-1]}_{e_d[n]}\right) \tag{5.24}$$

#### 5.3.2.1 Controller Tuning

Next the three free parameters $k_{p,i,d}$ or $K$, $T_{i,d}$ are to be chosen in such a way that the controller has optimal performance. This process is called *tuning*. Tuning of the parameters can be performed online or offline.

Online tuning is done by using the physical[4] plant. The most popular member of this class is the Ziegler-Nichols tuning[27]. The method relies on experiments and tabulated values. The mostly used variant uses one measured step response.

Using this method at FLUTE produced mixed results. While being a very simple and fast process, the resulting controller is often unstable. This is partially due to errors when extracting the tabulated values from the noisy step response, but also the method intrinsically leads to poor stability margins.[23, p. 142] Nonetheless the Ziegler-Nichols method yields a usable starting point, if the strategy is to fine tune the controller manually by intuition and experience of the user.

As the Ziegler-Nichols method does not yield an acceptable parameter set and the method combined with fine tuning by hand takes a considerable amount of time, next tuning the parameters offline using only the plants transfer function and the measurement filter is done. The offline tuning can be done analytically or using different numerical optimization strategies. In [28] analytical methods, such as the internal model control design or the pole placement design are discussed. Both require the systems transfer function in a closed form.
Tuning by numerical optimization chooses the parameters by a simulation or measured data. Goal of the optimization is to minimize a cost function $J$ like

$$J(\theta) = \sum_{n=0}^{\infty} e_\theta[n]^2 \tag{5.25}$$

with the parameter vector $\theta = [k_p,\, k_i,\, k_d]$ or $\theta = [K,\, T_i,\, T_d]$:

$$\theta_{\text{opt}} = \underset{\theta}{\operatorname{argmin}} J(\theta) \tag{5.26}$$

A convenient way to do such an optimization by simulation using the transfer function of the estimated plant is the Matlab *PID Tuner*.

**Tuning the Controller with the Matlab PID Tuner**

The Matlab PID Tuner accepts different kinds of system models used by the Matlab/Simulink ecosystem. It is possible to directly use estimated models, models defined via their transfer function or the zero-pole-gain representation, Simulink models or combinations of those[5].

The manual of the PID Tuner[29] describes the expected input (see Figure 5.8). The feedback path has to have unity gain, i.e. there is no measurement filter allowed. Therefore the measurement filter is moved before the junction where the output $y[n]$ would normally be measured. Then the measurement filter $H[z]$ is combined with the plant $P[z]$ to form $Sys[z]$. For that reason $y[n]$ becomes an internal signal of $Sys[z]$. However the new system output $y'[n] = h[z] * y[n]$ is not too different from the old, since $H[z]$ is designed to remove high frequency noise but retain the rest of $y[n]$.

The manufactures documentation [30] does not disclose any internals of the PID Tuner nor state which optimization technique is used, but three tuning objectives are stated:

[4] "Physical" in the sense that one could touch it. (But should one?)
[5] Series connection can be established by multiplying the models transfer functions.

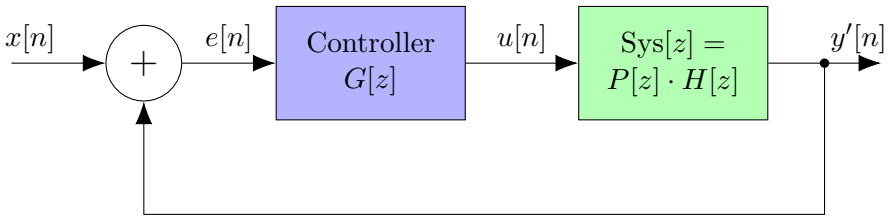



**Figure 5.8:** Required system architecture for the Matlab PID Tuner; PID Tuner input is the system block $Sys[z]$ (green), generated output is the controller $G[z]$ (blue)

- (Stability) The closed loop should be stable (that is BIBO stable as defined in 1)
- (Performance) The closed loop system tracks the input well and rejects disturbances as rapidly as possible (see subsection 2.4.1)
- (Robustness) A gain and phase margin accounts for errors in the system model (see section 5.1)

Using Listing 5.3, the estimated plant $P[z]$ is loaded, a measurement filter $H[z]$ is generated and the combination $Sys[z] = P[z]H[z]$ is fed into the PID Tuner.

**Listing 5.3:** Matlab script to generate an input system for PID Tuner

```
1  %design lowpass measurement filter H
2  N=10;
3  lpFilt1 = designfilt('lowpassfir', 'FilterOrder', N, 'CutoffFrequency', ...
4                       0.01, 'SampleRate', 0.2, 'DesignMethod', ...
5                       'window', 'Window', 'kaiser');
6
7  %convert filter to a dynamic system
8  [z1,p1,k1]=zpk(lpFilt1);
9  H1=zpk(z1,p1,k1,0.2);
10
11 %load estimated transfer function and convert it to discrete form
12 load('P2ZU.mat')
13 P=c2d(idtf(P2ZU),0.2);
14
15 Sys1=tf1*H1; %Combine plant P and measurement filter H
16 pidTuner(Sys1); %Launch pidTuner
```

Using the PID Tuner GUI (see Figure A.2), the controller is designed by changing the design parameters `Response Time` (RT) and `Transient Behavior` (TB). Changing the response time (in seconds) influences how fast the controller acts on changes. With the transient behavior setting, the allowed over- and under-shoots, that is the deviation above and below the final value for $t \to \infty$ can be set qualitatively.

With this tool four different controllers are designed, two for each measurement filters order of $N = 10$ and $N = 40$. The transient behavior with 0.2 for all controllers is chosen as a compromise between speed and overshoot behavior. The optimized PID parameters (for a discrete time PID controller in parallel form) are listed in Table 5.2.

**Table 5.2:** Parameters of a discrete time PID controller in parallel form calculated with the Matlab PID Tuner; $N$ is the order of the used measurement filter

| Name | $N$ | RT (in s) | TB | $k_p$ | $k_i$ | $k_d$ |
|---|---|---|---|---|---|---|
| $G_1[z]$ | 10 | 5 | 0.2 | $-0.000\,577$ | $-0.000\,197$ | $-0.000\,421$ |
| $G_2[z]$ | 10 | 10 | 0.2 | $-0.000\,155$ | $-0.000\,111$ | $-5.43 \times 10^{-5}$ |
| $G_3[z]$ | 40 | 5 | 0.2 | $-0.000\,559$ | $-2.27 \times 10^{-5}$ | $-0.000\,396$ |
| $G_4[z]$ | 40 | 10 | 0.2 | $-0.000\,35$ | $-9.68 \times 10^{-5}$ | $-0.000\,316$ |

### 5.3.3 Analyzing the Input Tracking and Disturbance Rejection

The PID Tuner defaults to show the input tracking step response (as in Figure A.2). The input tracking is calculated based on Equation 2.31. To calculate it, the Matlab function `step()` is used which generates the step response plot of a dynamic system, in this case $F_T$. As the whole system consists only of linear building blocks, using an arbitrary step height of 1 is possible. The response is then normalized in such a way that the final value is also 1.

In Figure 5.9 and Figure 5.10 the input tracking step responses for all four controllers are plotted. Note the different $t$ scales. As expected, for longer measurement filters, i.e. filters of higher orders, the controller has to act less aggressive. This causes longer settling times. Especially Figure 5.10 shows that setting a shorter response time does not automatically cause the controller to set the output faster to its final value. While for the shorter response time, the set value is reached quicker, there is also a strong oscillation, so choosing the longer response time (see $G_4[z]$) can be beneficial.

Because for FLUTE the set-point only changes occasionally, basically a fixed set-point controller is needed. Therefore the disturbance rejection is more important than the input tracking when analyzing the controllers performance. The step responses of the disturbance rejection $F_{DR}$ are calculated in a similar way as the input tracking, using `??` and `step()`. The $y$ axis are normalized so that the initial value is 1 and the final value for $t \to \infty$ is 0.

In Figure 5.11 and Figure 5.12 the results for all controllers in Table 5.2 are plotted.

This shows that even for an optimistically low measurement filter order of $N = 10$, the settling time is in the order of slightly under a minute. This is sufficient for the application but has more of a practical drawback: when fine-tuning the PID parameters manually on the machine, it is necessary to wait a few minutes after setting new parameters before assessing the change.

Especially when looking a the plots of $G_3[z]$, it is obvious that there can be stability issues when using a too aggressive controller. For that reason in the next section the stability of the four controllers is analyzed.

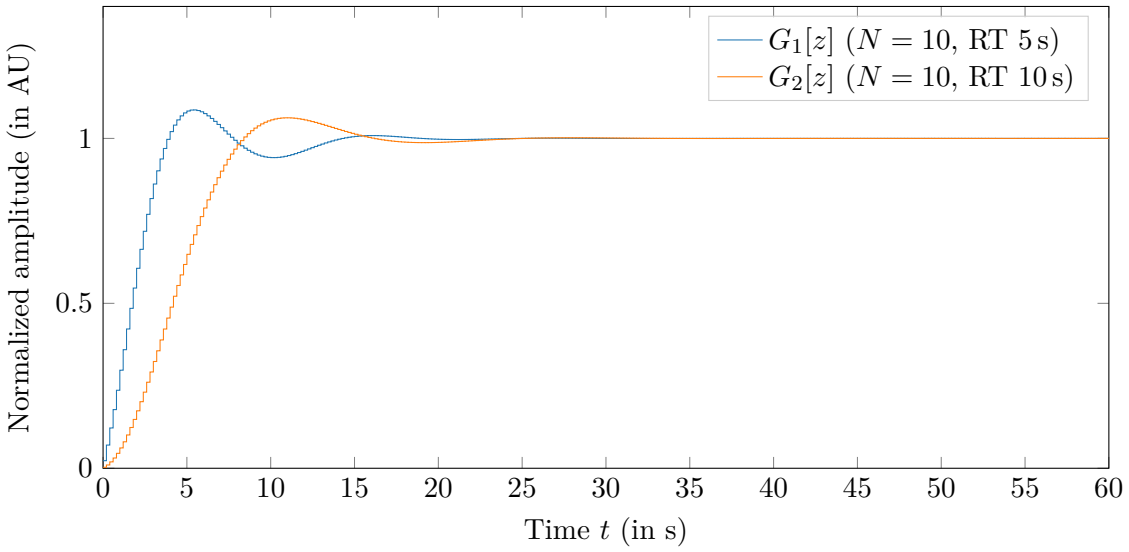



**Figure 5.9:** Step response of the input tracking $F_T$ for controller $G_1[z]$ and $G_2[z]$, designed with the plant $P[z]$ and a measurement filter of order $N = 10$, response time goal to 5 s and 10 s

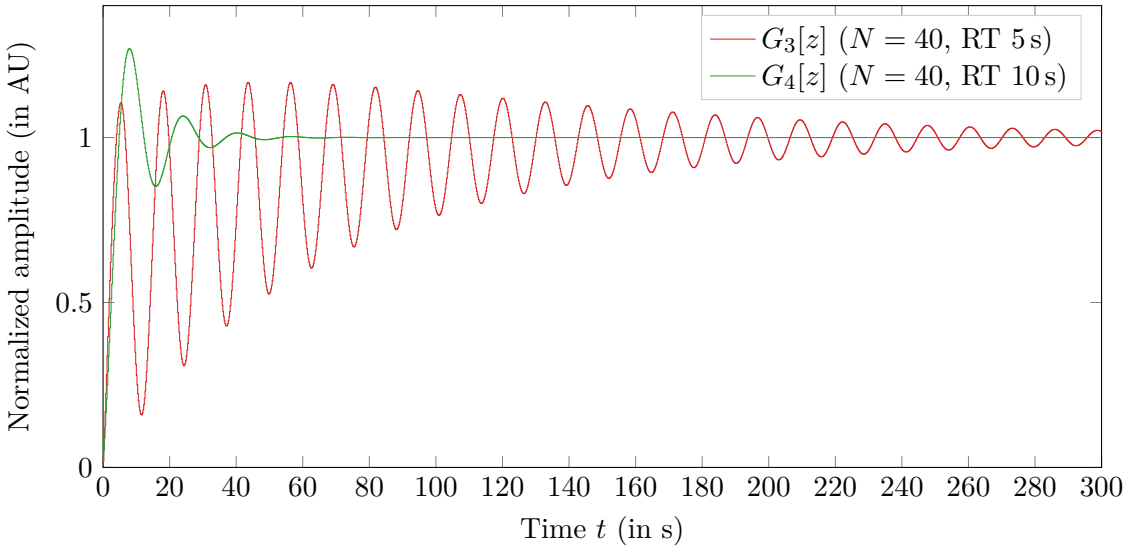



**Figure 5.10:** Step response of the input tracking $F_T$ for controller $G_3[z]$ and $G_4[z]$, designed with the plant $P[z]$ and a measurement filter of order $N = 40$, response time goal to 5 s and 10 s

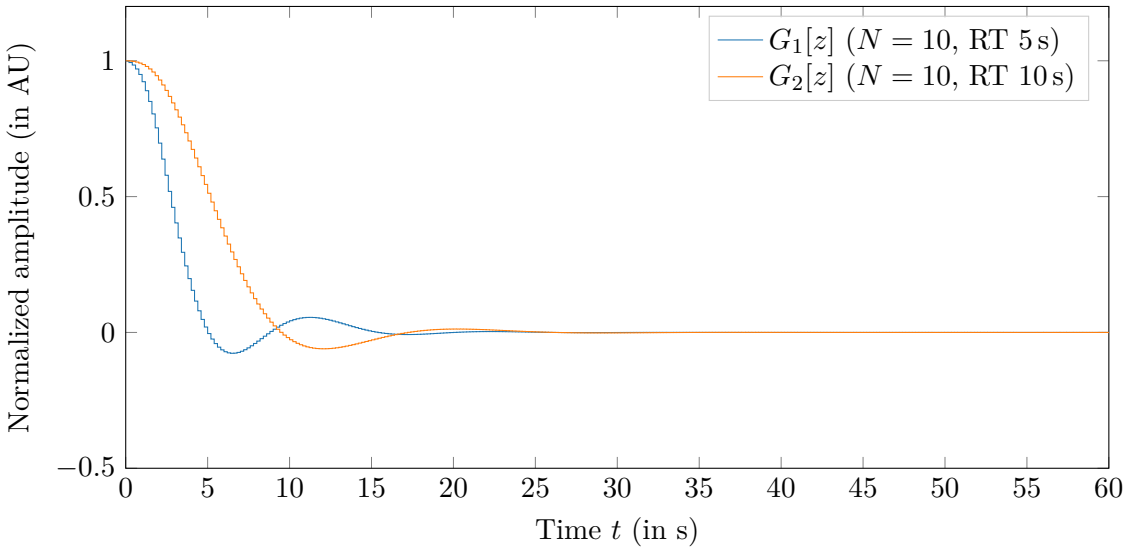



**Figure 5.11:** Step response of the disturbance rejection $F_{DR}$ for controller $G_1[z]$ and $G_2[z]$, designed with the plant $P[z]$ and a measurement filter of order $N = 10$, response time goal to 5 s and 10 s

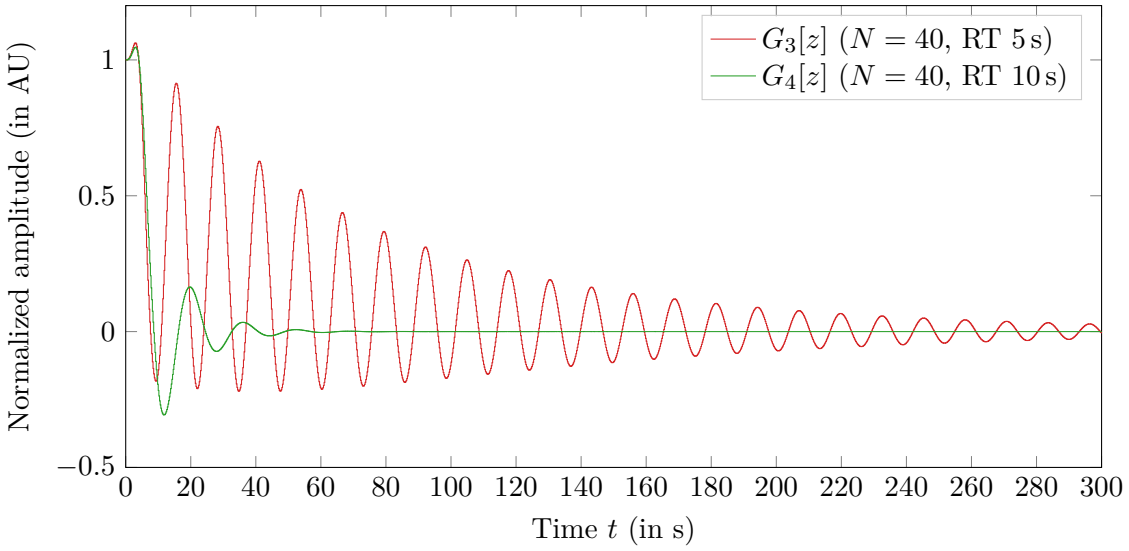



**Figure 5.12:** Step response of the disturbance rejection $F_{DR}$ for controller $G_3[z]$ and $G_4[z]$, designed with the plant $P[z]$ and a measurement filter of order $N = 40$, response time goal to 5 s and 10 s

### 5.3.4 Analyzing the Stability

In this section the stability of the controllers $G_i[z]$ from the last section is evaluated.

For that the Nyquist criterion according to 2 is used. With the Matlab function `nyquist()`, the locus $w = F_o[w = j2\pi f]$ of the open loops

$$F_o[z] = G[z]P[z]H[z] \tag{5.27}$$

are calculated. For all controllers designed, they are plotted in Figure 5.13.

This shows the gain margin for $G_3[z]$ (in combination with the $N = 40$ filter), that is the distance to the critical point $w = (-1, 0j)$ is already very small. So in addition to its oscillatory behavior and the longer settling time, there is also the risk that the system can become unstable if there are even small system parameter changes.

In Figure 5.14 another issue is highlighted. If a controller is designed with a certain measurement filter order $N = N_0$ in mind but is used together with a filter of order $N = N_1 > N_0$, it is very likely the closed loop system becomes unstable. In the figure the controller $G_1[z]$ is used together with the $N = 40$ filter. Both the disturbance rejection and the Nyquist plot shows the closed loop to be unstable.

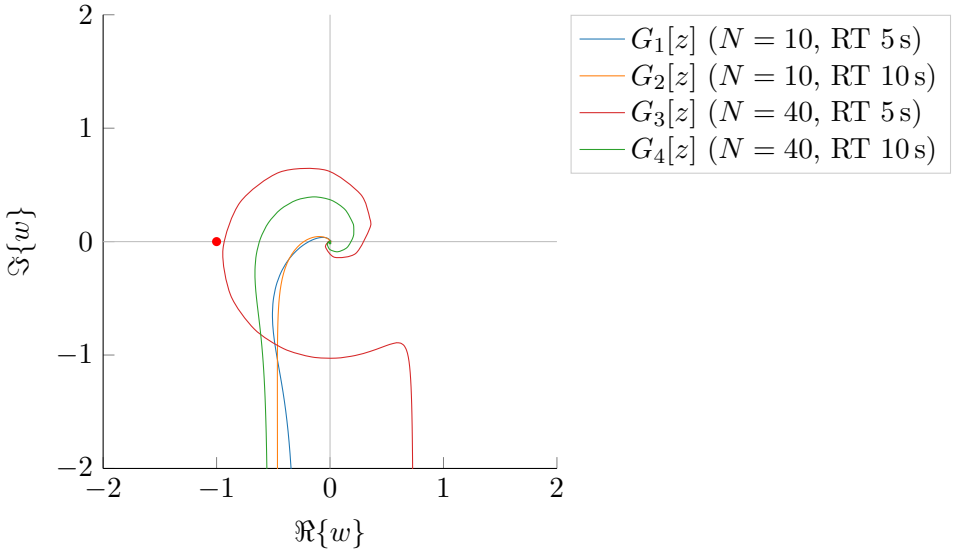
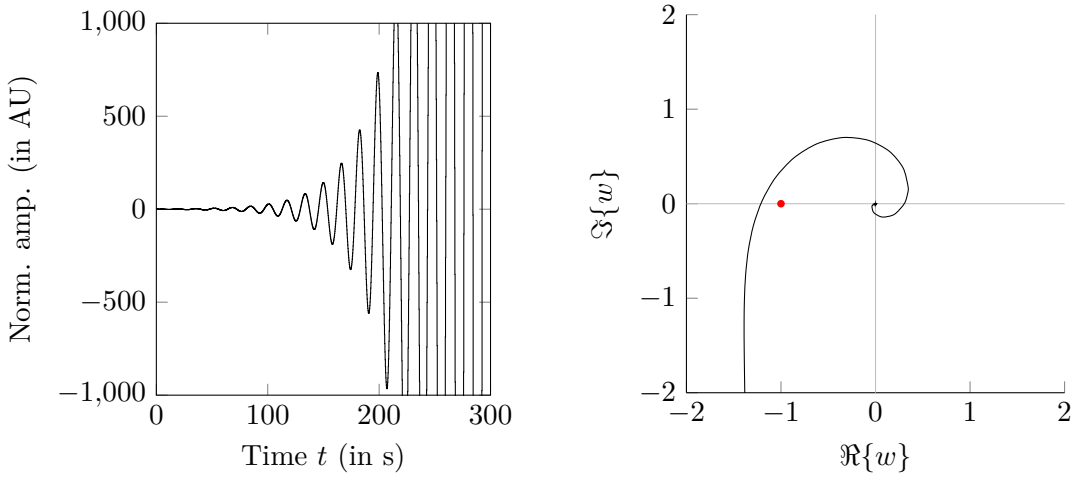



**Figure 5.13**




**(a)** a **(b)** b

**Figure 5.14:** caption

### 5.3.5 Offline Evaluation with Measured Data

Now the performance of the controller $G_1[z]$ together with the filter order $N = 10$ should be evaluated with measured data before testing it with FLUTE. This simulation is called offline evaluation.

Offline evaluation of the control system is potentially less accurate due to an incomplete or erroneous model. But on the upside it requires no access to FLUTE and no potentially downtime of the machine. Also with modern computers the simulation time advances much faster than real-time. Another big advantage is repeatability. With a pre-recorded data-set as the disturbance input, simulations are consistent and do not depend on a changing environment or system parameters.

The offline evaluation is done with Simulink, a block diagram development and simulation environment based on Matlab. The Simulink model (see Figure A.3) uses the disturbance

$$d[n] = \Delta P_{\text{cavity}}[n] = P_{\text{cavity}}[n] - \mu_{P_{\text{cavity}}} \tag{5.28}$$

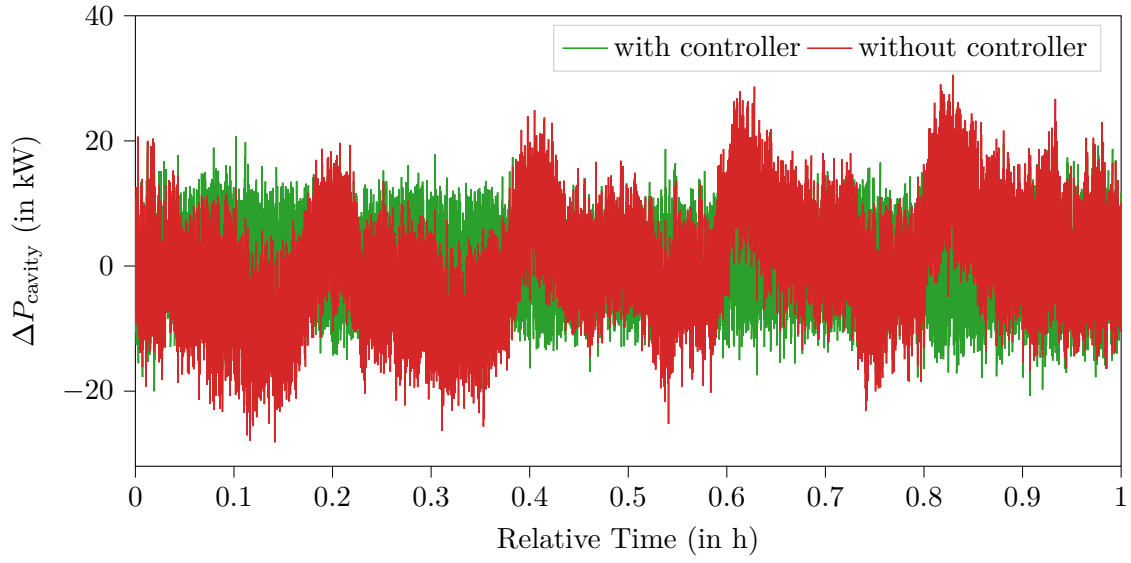



**Figure 5.15:** Output of the Simulink model in Figure A.3; step size $T = 0.2\,\text{s}$, end time 3600 s. Simulation time on an Intel i7-3770: 3.5 s

with $\mu_{P_{\text{cavity}}}$ being the time average of $P_{\text{cavity}}[n]$ over one hour. $P_{\text{cavity}}[n]$ is re-sampled to $T_s = 0.2\,\text{s}$. With this calculation of $d[n]$, it is assumed that the disturbance is the deviation of the cavity power from a (theoretical) set-point of $\mu_{P_{\text{cavity}}}$.

With the simulation the effect of adding $d[n]$ to a system without any feedback and a system with the controller $G_1[z]$ and the measurement filter is compared. The result is shown in Figure 5.15

## 5.4 Implementation of the Control System in Software

In this section the findings from the beginning of this chapter are used to implement a time discrete control system, that is a time discrete PID controller that is used to drive the controllable attenuator based on filtered measurements from the system.

The system should be implemented as software that runs on most personal computers and requires few external dependencies to make it as portable as possible. The other core requirements can be summarized to be:

- Communicate with EPICS (over an Ethernet connection) to read in data such as $P_{\text{cavity}}[n]$
- Provide means to filter the incoming data with a measurement filter $H[z]$ with variable order $N$ and cutoff frequency $f_c$
- Calculate the control error $e[n]$ and based on that the controller output $u'[n]$. Then convert the controller output, an attenuation, to the matching control voltage $V_{\text{control}}$
- Communicate with the Keysight 34972A over VXI-11 (over an Ethernet Network) to set the control voltage $V_{\text{control}}[n]$ of the attenuator
- The control routine of the program needs to be light-weight enough so a scheduler can call it faster than five times a second to achieve a sampling time of 0.2 s
- Show the input, the output and the error signals to the user on a GUI
- Provide graphical input elements to let the user modify the measurement filter and the controller parameters ($k_p$, $k_i$, $k_d$)
- Log the input, output and parameters to disk for later reference

Since many other choices depend on it, first the programming language has be picked. As Python was used in earlier chapters and the communicating abilities to both EPICS and the Keysight 34972A were already proven, it is the obvious choice.

From there on the GUI framework is the next decision to be made. For Python popular choices are *Tkinter*, a build in implementation of the Tk/tcl GUI toolkit, *wxPython*, a Python binding to the cross-platform GUI library wxWidgets, or *PyQt*[31], a set of Python bindings for the Qt GUI framework[32]. While Tkinter has the advantage of being build into the Python language, wxPython offers more functionality and is more widely adopted. PyQt profits from the large Qt ecosystem, so is it for example possible to develop the GUI separately from the code using *Qt Designer*.

For (live-) plotting of data in the GUI, the standard library in Python is *matplotlib*[33]. It can be used with all three plotting libraries. However the biggest drawback is the possible update speed of the plots. With more than about a hundred points on the screen, the update frequency drops to well below 1 Hz. A much faster alternative is *pyqtgraph*[34], a scientific plotting library written in Python and using the Qt GraphicsView. It is only compatible only with PyQt (or the PySide alternatives).

Therefore PyQt and pyqtgraph are used to build handle the GUI and plot the live data. Also Qt Designer is used to model the GUI graphically.

The control algorithm is implemented by directly using the equations derived in subsection 5.3.2. The time domain representation of the PID controller is (restated from Equation 5.24)

$$u[n] = k_p e[n] + k_i \left( \underbrace{e_i[n-1] + T_s e[n]}_{e_i[n]} \right) + k_d \left( \underbrace{\frac{1}{T_s} e[n] - e[n-1]}_{e_d[n]} \right) \tag{5.29}$$

When translating Equation 5.29 to code, the recursion ($e_i[n]$ depending on $e_i[n-1]$) is solved by introducing a variable that keeps track of $e_i[n]$ over time. The last value of $e[n]$, $e[n-1]$ is kept in memory for the derivative part. An example in pseudo code is listed in Listing 5.4.

**Listing 5.4:** PID controller implemented in pseudo code

```
1 while(true) {
2     e       := x_set - x_actual
3     e_i     := e_i_old + e*T_s
4     e_d     := (e-e_old)/T_s
5     e_i_old := e_i
6     e_old   := e
7     wait_sec(0.2)
8 }
```

To get the timing right and to allow the GUI to be responsive while the control algorithm runs in the background, a scheduler that activates a callback function every 0.2 s should be used. Since in section 4.2 the Advanced Python Scheduler (AP scheduler) is used successfully, it should also be used for the control system. Unfortunately the AP scheduler is not compatible with PyQt and using it interferes with the internal Qt timing systems. For that reason instead a timer is constructed with the `QTimer` class.[35]

Logging the data to disk is done by using the same `QTimer`, as is used for the control algorithm, to write one line of CSV data to disk each time the timer is triggered.

With these building blocks ready, the GUI is build with Qt Designer and based on the `.ui` file generated from Qt Designer the GUI Application is implemented. For a screenshot of the program running see Figure A.4.

## 5.5 Evaluation of the Control System (Online)

After evaluating the controller offline, it is now time to evaluate the performance of the control system on the machine. To do so, FLUTE is operated with and without the control system switched on for 6 h each. Before the test, FLUTE is allowed to run a few hours for all components to reach operating temperatures. The result of the test is shown in Figure 5.16. Note, that in this test about 7 h into the experiment there was an unexpected shutdown of FLUTE and the corresponding block of data is removed before further processing.

The time plot Figure 5.16 and also the spectogram in Figure 5.17 show the cavity RF power approximately reaches stationarity in the [0, 2] hour interval respectively in the [6, 12] hour interval. This allows the spectrum for these two blocks to be estimated using a periodogram method. In Figure 5.19 the spectra for each case are shown. The corresponding time data is plotted in Figure 5.18

With the periodogram data, the control system success can be measured with the most prominent noise metric.

Next the cases controller on and controller off should be compared with the relative standard deviation. Measuring and displaying the relative standard deviation $STD\%$ is also supported on the LLRF CSS panel at FLUTE, so while the experiments run, the values are checked from time to time. It can be observed that different values are shown over time although the plotted time signal looks stationary by eye. This suggests the relative standard deviation does not only depend on the window size $T$ over which is is calculated, but also heavily on the absolute position in time $t$. This effect is more visible for small window sizes. The issue is illustrated in Figure 5.20, where $STD\% = STD\%(t, T)$ is plotted as a function of the time $t$ and the window size $T$.

For that reason, for small window sizes, time averages of $STD\%$ should be used instead of single values. For the measured $STD\%(t, T)$ for the controller off and the controller on, $STD\%(T)$ is plotted in Figure 5.21. This plot shows what could already be guessed from Figure 5.20: with the $STD\%$ metric, the system seems *less* stable with the controller on, if the $STD\%$ is calculated over small window sizes e.g. $T = 1\,\text{min}$. For very long windows, the dependency on the window position becomes very small. So for the comparison with the other metrics, $STD\%(T = 4\,\text{h})$ is used.

With the metrics from section 3.1, the success of the control system is assessed in Table 5.3.

This shows for example the mean squared error is improved by a factor of about 371 by using the control system.

**Table 5.3:** Quantitative assessment of the controllers performance

| Metric | Controller off | Controller on | Controller off/Controller on |
|---|---|---|---|
| $\%STD(4\,\text{h})$ | 0.001 155 9 | $5.992\,25 \times 10^{-5}$ | 9.2891 |
| $MSE$ | 37.639 | 0.101 33 | 371.44 |
| $MPN$ | 487 309.29 | 14 386.25 | 33.873 |

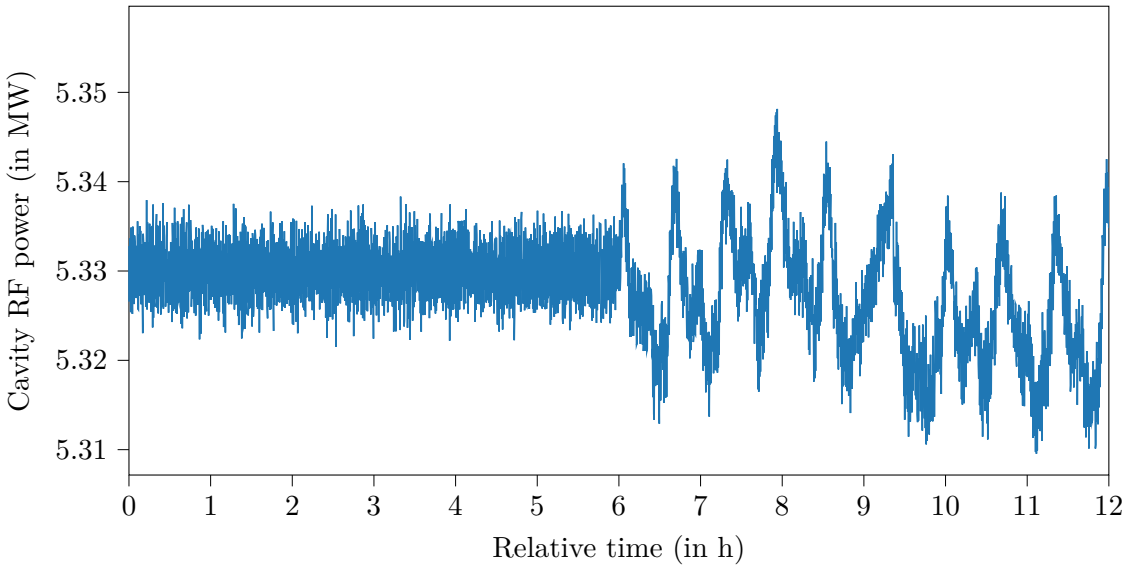



**Figure 5.16:** Cavity power over about 15 h (about three hours of downtime removed for clarity around the 7 h mark); control system switched off at 6 h (recording started 2021/05/01 20:00)

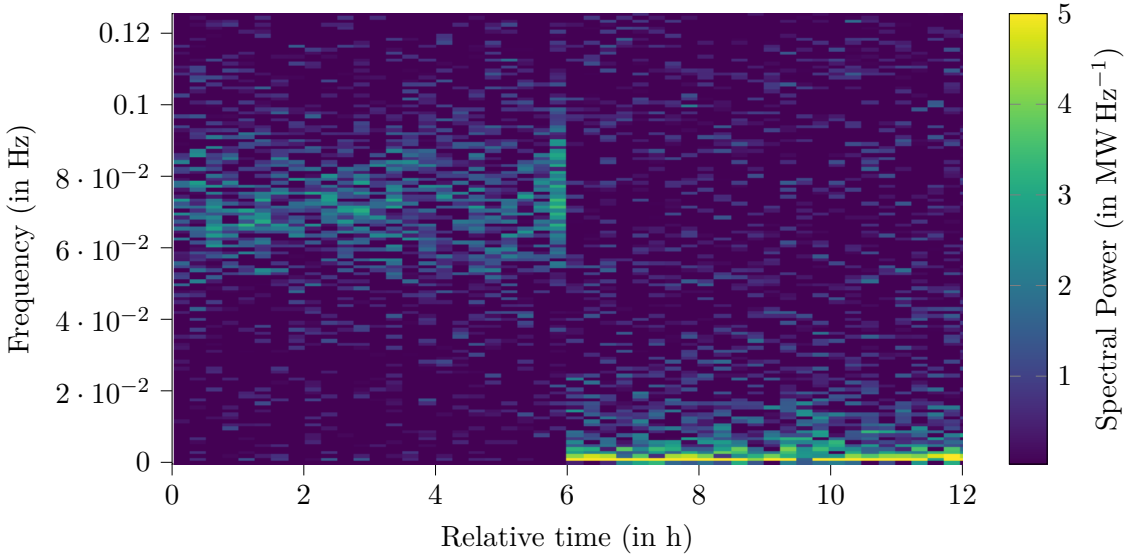



**Figure 5.17:** Spectrogram of the cavity power in Figure 5.16

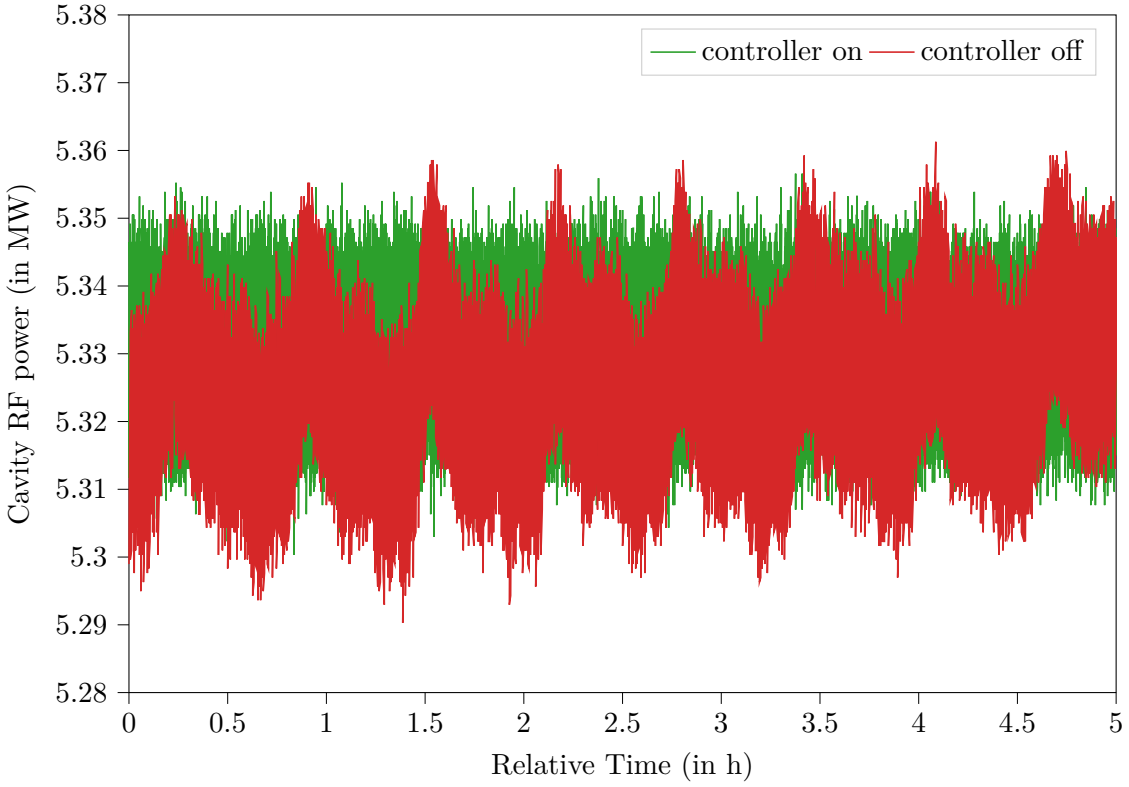



**Figure 5.18:** Time plots comparing between the control system on and off

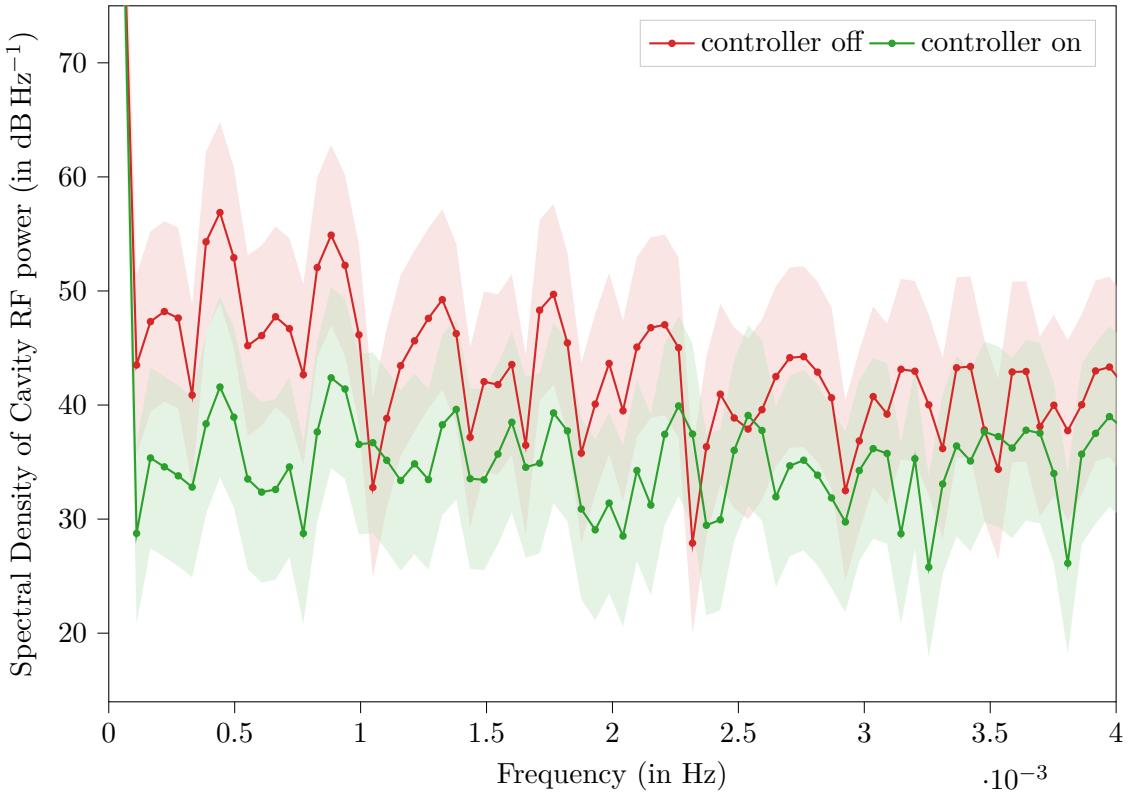
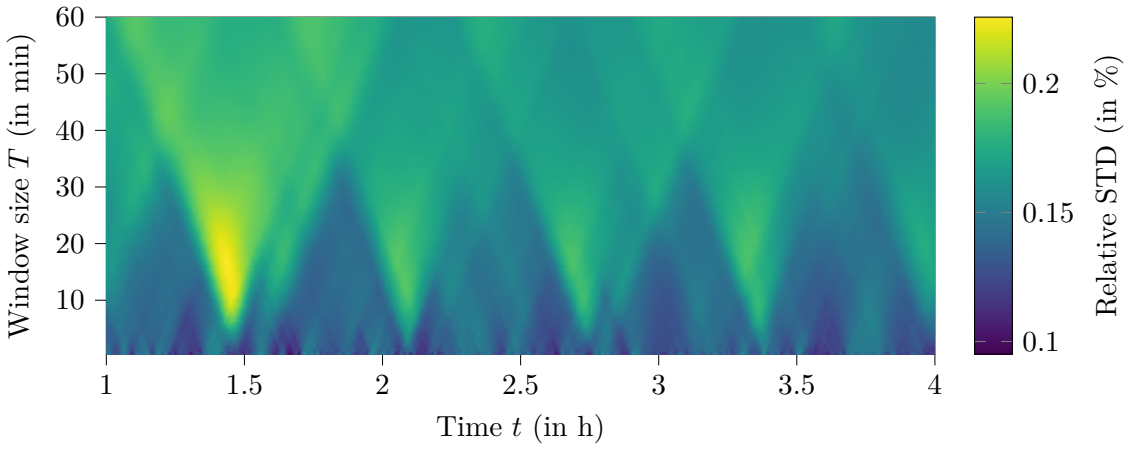
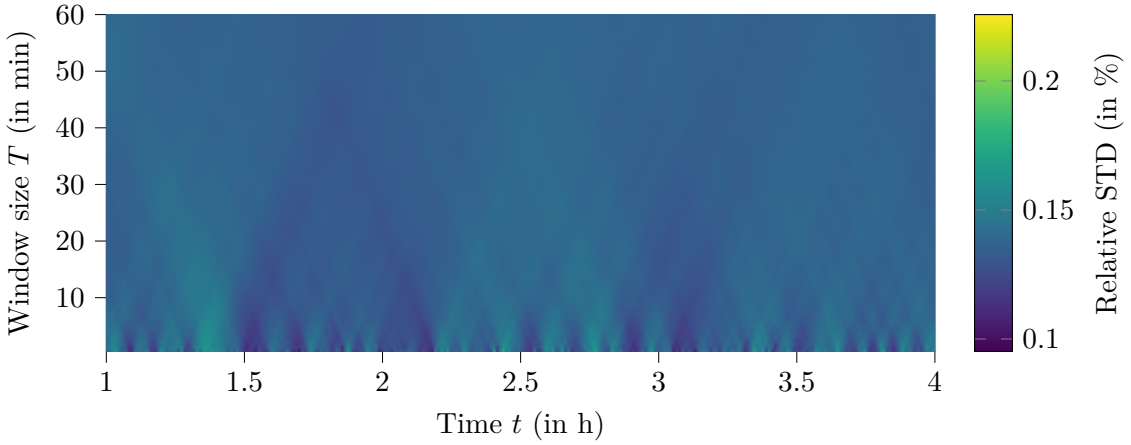



**Figure 5.19:** Power spectrum of the plots in Figure 5.16 computed with Welch's method; shaded areas show the uncertainty according to Equation 2.28




**(a)** Controller off




**(b)** Controller on

**Figure 5.20:** Relative standard deviation $STD\%(t, T)$

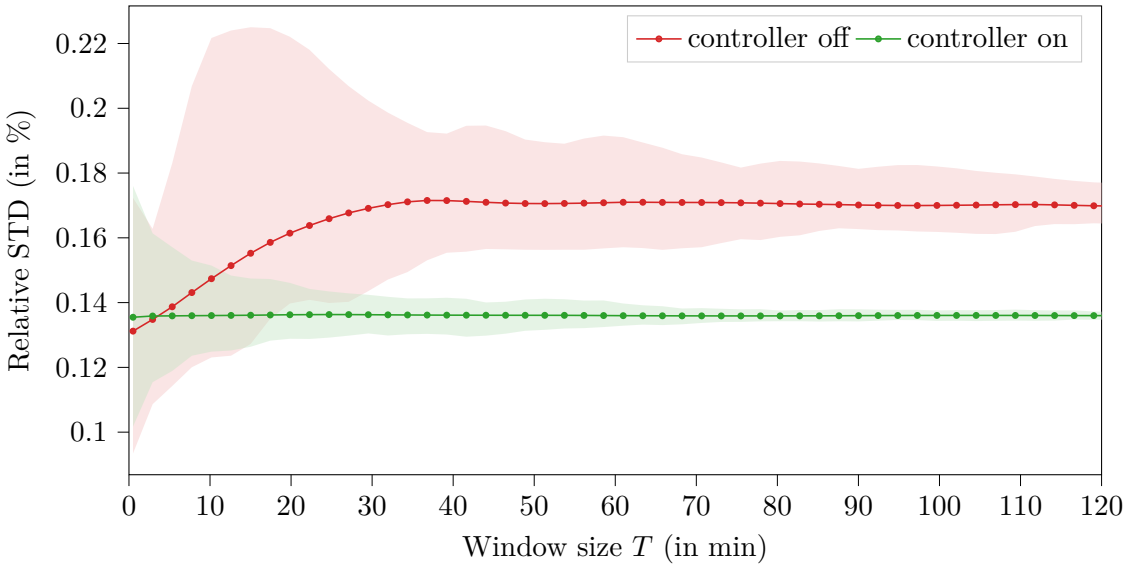



**Figure 5.21:** Relative standard deviation $STD\%(T)$, shaded areas show $\min\{STD\%(t, T_0)\}$ and $\max\{STD\%(t, T_0)\}$, solid lines show $\text{mean}\{STD\%(t, T_0)\}$ for a fixed window size $T = T_0$

# 6. Summary and Outlook

# Appendix

## A Complementary Material Controller Design

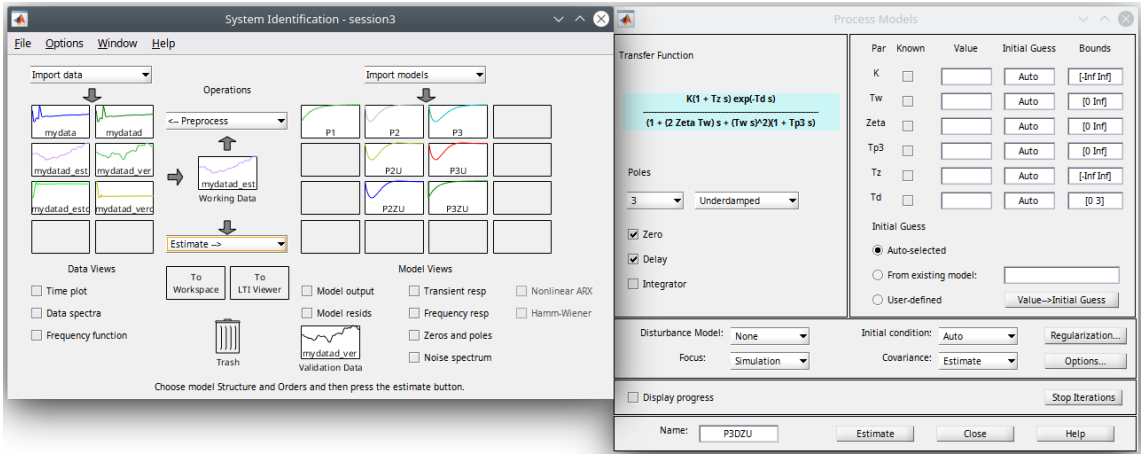



**Figure A.1:** Screenshot of the Matlab System Identification Toolbox; to the right the process models estimator window

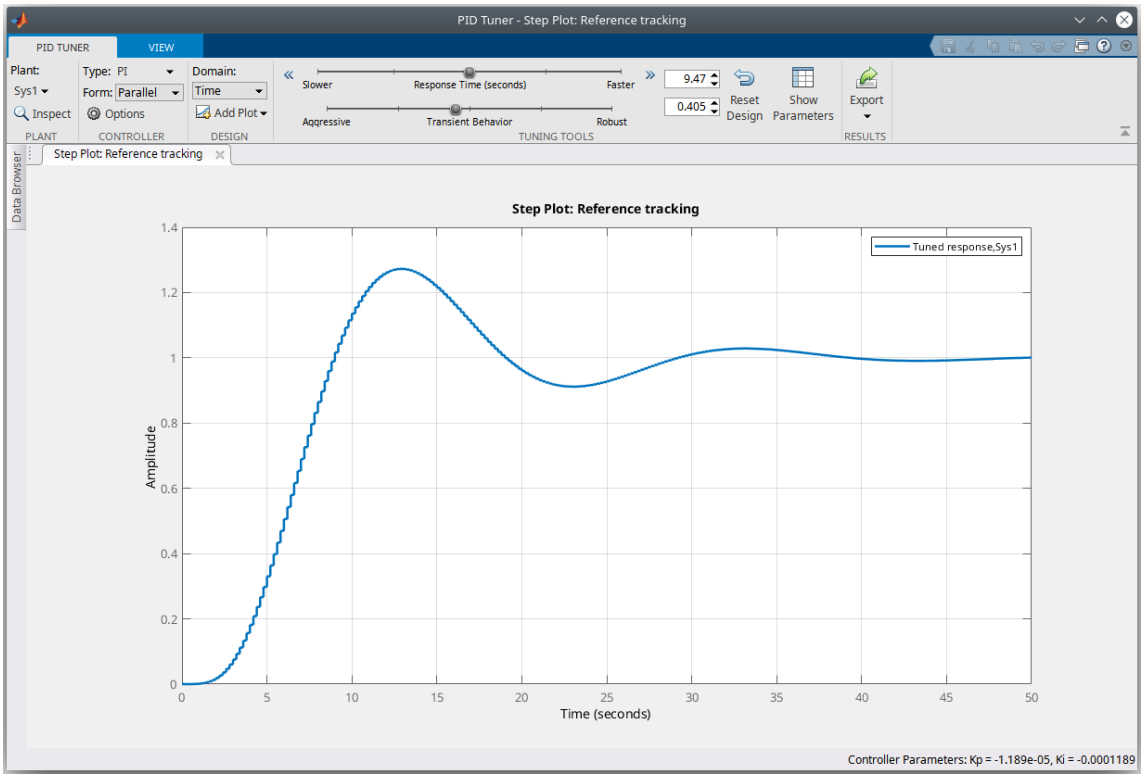



**Figure A.2:** Screenshot of the Matlab PID Tuner from the Control Systems Toolbox

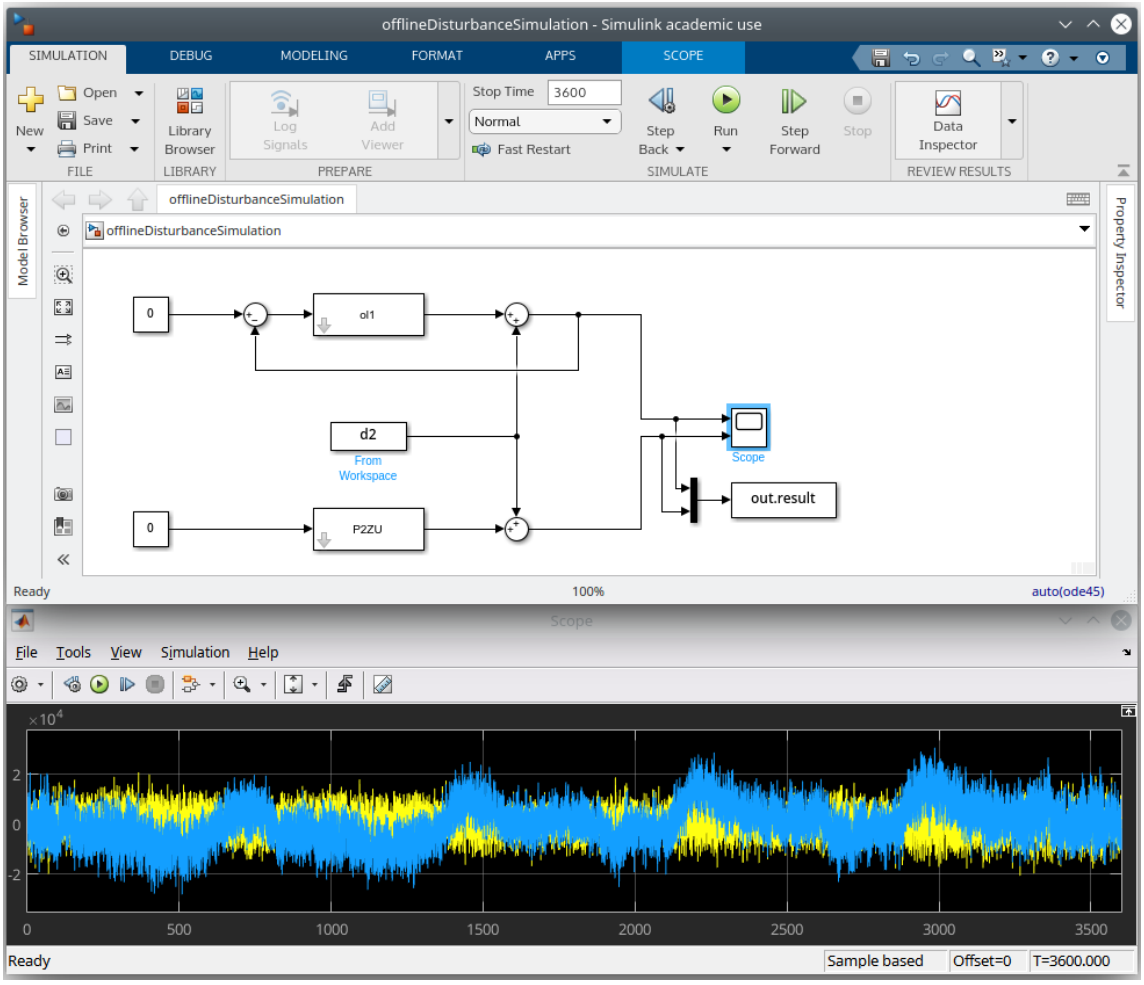



**Figure A.3:** Simulink model to evaluate the designed controller together with the measurement filter (`ol1`) compared to the uncontrolled system (in `P2ZU`) using measured disturbance data (in the vector `d2`); below a view of the scope data

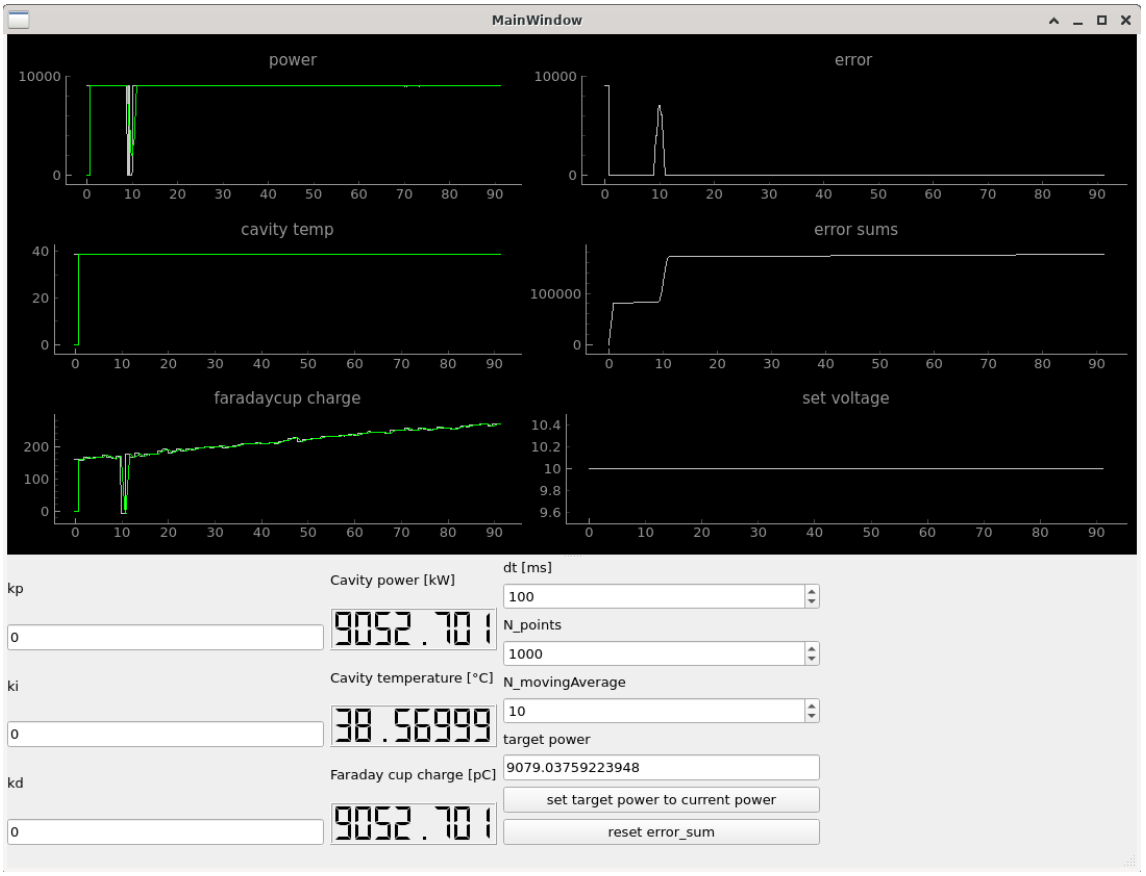



**Figure A.4:** Screenshot of the control systems GUI application

# B Lab Test and Measurement Equipment

## B.1 Benchtop multimeters

### B.1.1 Agilent 34411A

**Table B.1:** Agilent 34411A specifications

| Specification | Value |
|---|---|
| | DC volt |
| Digits | 6 1/2 |
| Measurement method | cont integrating multi-slope IV A/D converter |
| Accuracy (10 V range, 24 hours) | 0.0015 %+0.0004 % (% of reading + % of range) |
| Bandwidth | 15 kHz (typ.) |

**Table B.2:** Agilent 34411A some SCPI commands

| Description | Example command | Example return |
|---|---|---|
| Read current measurement | `READ?` | `+2.84829881E+00` (2.848 V) |

### B.1.2 Keysight 34470A

**Table B.3:** Keysight 34470A specifications

| Specification | Value |
|---|---|
| | DC volt |
| Digits | 7 1/2 |
| Measurement method | cont integrating multi-slope IV A/D converter |
| Accuracy (10 V range, 24 hours) | 0.0008 %+0.0002 % (% of reading + % of range) |
| Bandwidth (10 V range) | 15 kHz (typ.) |

**Table B.4:** Keysight 34470A some SCPI commands

| Description | Example command | Example return |
|---|---|---|
| Read current measurement | `READ?` | `+9.99710196E+00` (9.997 V) |

## B.2 Data Acquisition/Switch Unit

### B.2.1 Keysight 34972A

**Table B.5:** Keysight 34972A specifications

| Specification | Value |
|---|---|
| 34907A (Multifunction module) | |
| DAC range | ±12 V |
| DAC resolution | 16 bit ($^{24\,V}/_{2^{16}}$ = 366.21 µV per bit) |
| DAC maximum current | 10 mA |
| 34901A (20 channel multiplexer) | |

**Table B.6:** Keysight 34972A some SCPI commands

| Description | Example command | Example return |
|---|---|---|
| Read current measurement | `READ?` | `+2.00200000E+01` (20.02 °C) |
| Set DAC voltage of ch 204 to 3.1 V | `SOUR:VOLT 3.1,(@204)` | |

## B.3 Oscilloscopes

### B.3.1 Tektronix MSO64

**Table B.7:** Tektronix MSO64 specifications

| Specification | Value |
|---|---|
| Bandwidth | 6 GHz |
| Sample rate | 25 GS/s |
| ADC resolution | 12 bit |
| DC gain accuracy (@ 50 Ω, >2 mV/div) | ±2 % |

**Table B.8:** Tektronix MSO64 some SCPI commands

| Description | Example command | Example return |
|---|---|---|
| Read mean of measurement 1 (current acq.) | `MEASUrement:MEAS1:RESUlts:CURR:MEAN?` | `3.0685821787408` |

## B.4 RF signal generator

### B.4.1 Rohde and Schwarz SMC100A

**Table B.9:** Rohde and Schwarz SMC100A specifications

| Specification | Value |
|---|---|
| Frequency range | 9 kHz to 3.2 GHz |
| Maximum power level | 17 dBm |
| SSB phase noise (@ 1 GHz, $f_o = 20$ kHz, $BW = 1$ Hz) | −111 dBc |
| Level error | <0.9 dB |

**Table B.10:** Rohde and Schwarz SMC100A some SCPI commands

| Description | Example command | Example return |
|---|---|---|
| Set RF power level to 10.5 dBm | `SOUR:POW 10.5` | |
| Set RF frequency to 3.1 GHz | `SOUR:FREQ:FIX 3.1e9` | |
| Enable the RF output | `OUTP on` | |

## B.5 RF power meter

### B.5.1 HP E4419B

**Table B.11:** HP E4419B specifications

| Specification | Value |
|---|---|
| Digits | 4 |
| Accuracy (abs. without power sensor) | ±0.02 dB |
| **Power probe: E4412A** | |
| Frequency range | 10 MHz to 18 GHz |
| Power range | −70 dBm to 20 dBm |

**Table B.12:** HP E4419B some SCPI commands

| Description | Example command | Example return |
|---|---|---|
| Measure power on input 1 | `MEAS1?` | `+2.89435802E+000` (2.894 dBm) |

## B.6 Vector Network Analyzer

### B.6.1 Agilent E5071C

**Table B.13:** Agilent E5071C specifications

| Specification | Value |
|---|---|
| Frequency range | 9 kHz to 8.5 GHz |

## B.7 Phase noise analyzer

### B.7.1 Holzworth HA7062C

**Table B.14:** Holzworth HA7062C specifications

| Specification | Value |
|---|---|
| DUT input frequency | 10 MHz to 6 GHz |
| Measurement bandwidth | 0.1 Hz to 40 MHz offsets |

# Acknowledgments

First i would like to thank Prof. Anke-Susanne Müller and the whole FLUTE team for providing me with the opportunity to write my thesis at their research facility. FLUTE proved to be a one-of-a-kind machine and i really appreciate the work that has been put into it.

Many thanks go also to Prof. Dr.-Ing. John Jelonnek from the institute for pulsed power and microwave technology for taking the responsibility of being my first reviewer. Also many thanks to his PhD students Benjamin Ell and Alexander Marek for being very patient while listening to and optimizing my presentations.

I'm especially in debt to my excellent advisor Dr. Nigel John Smale, for always having the time to go over both fundamentals and nifty details of electronics and physics alike. Without his support i wouldn't even be able to switch on FLUTE by now and much less a sucessful thesis would've been possible.

Thanks to Igor Križnar for constructive discussions and programming the GUI panel to control the charge sensitive amplifier.

Thanks Olena Manzhura for her support and for prove-reading the thesis multiple times.

And lastly but maybe most importantly, a big thanks goes out to my family, that is my mom Sonja, Stefan and my grand-parents Ernst and Gretel, for supporting me during my whole course of studies.

# Bibliography


[1] F. Hinterberger, *Physik der Teilchenbeschleuniger und Ionenoptik mit durchgerechneten Beispielen und 99 Übungsaufgaben mit vollständigen Lösungen*. Berlin, Heidelberg, New York: Springer, 1997, ISBN: 9783540612384.

[2] T. Wangler, *RF linear accelerators*. Weinheim: Wiley-VCH, 2008, ISBN: 9783527623426.

[3] K. I. Park, *Fundamentals of Probability and Stochastic Processes with Applications to Communications*. Springer International Publishing, Dec. 4, 2017, 288 pp., ISBN: 3319680749.

[4] F. Puente León, *Messtechnik*. Springer-Verlag GmbH, 2019, ISBN: 3662597667.

[5] A. Lapidoth, *A Foundation in Digital Communication*. Cambridge University Press, Mar. 26, 2019, 920 pp., ISBN: 1107177324.

[6] P. Stoica, *Introduction to spectral analysis*. Upper Saddle River, N.J: Prentice Hall, 1997, ISBN: 0132584190.

[7] D. Rowell. (2008). 2.161 Signal Processing: Continuous and Discrete, Massachusetts Institute of Technology: MIT OpenCourseWare, [Online]. Available: `https://ocw.mit.edu/courses/mechanical-engineering/2-161-signal-processing-continuous-and-discrete-fall-2008/#`.

[8] P. Welch, "The use of fast Fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms", *IEEE Transactions on Audio and Electroacoustics*, vol. 15, no. 2, pp. 70–73, Jun. 1967. DOI: `10.1109/tau.1967.1161901`.

[9] O. Föllinger, *Regelungstechnik*. Vde Verlag GmbH, Jun. 2016, ISBN: 3800742012.

[10] W. F. Guthrie, *NIST/SEMATECH e-Handbook of Statistical Methods (NIST Handbook 151)*, en, 2020. DOI: `10.18434/M32189`.

[11] L. R. Dalesio, M. R. Kraimer, and A. J. Kozubal, "EPICS Architecture", *ICALEPCS*, vol. 91, 1991.

[12] Control System Studio. (2021). Control System Studio, [Online]. Available: `https://controlsystemstudio.org/about` (visited on 05/26/2021).

[13] M. Newville and A. Gratton. (2019). PyEpics: Python Epics Channel Access, [Online]. Available: `https://cars9.uchicago.edu/software/python/pyepics3/`.

[14] RadiaBeam, *Faraday Cups*. [Online]. Available: `http://www.radiabeam.com/upload/catalog/pdf/14272334342015-03-24_faraday-cups.pdf`.

[15] P. Synotech, *PCB 421A25 Charge Amplifier*.

[16] Mini-Circuits, *ZX73-2500+ Voltage Variable Attenuator*.

[17] R. W. Waugh, "A Low-Cost Surface Mount PIN Diode $\pi$ Attenuator", vol. 35, no. 5, pp. 280–284, 1992.

[18] The LXI Consortium. (May 29, 2021). LXI instrument communication, [Online]. Available: `https://www.lxistandard.org/about/vxi-11-and-lxi.aspx`.

[19] A. Grönholm. (2021). Advanced Python Scheduler, [Online]. Available: `https://apscheduler.readthedocs.io/`.

[20] L. Wang, *From plant data to process control : ideas for process identification and PID design*. London New York: Taylor & Francis, 2000, ISBN: 0748407014.

[21] Kammeyer, *Digitale Signalverarbeitung Filterung und Spektralanalyse*. StuttgartLeipzig-Wiesbaden: Teubner, 2002, ISBN: 9783519461227.

[22] A. Oppenheim, *Discrete-time signal processing*. Upper Saddle River, NJ: Pearson, 2010, ISBN: 9780131988422.

[23] Åström and Hägglund, *PID controllers*. Research Triangle Park, N.C: International Society for Measurement and Control, 1995, ISBN: 1556175167.

[24] S. Zacher and M. Reuter, *Regelungstechnik für Ingenieure*. Vieweg+Teubner Verlag, Dec. 7, 2010, 514 pp., ISBN: 9783834898371.

[25] S. J. Dodds, *Feedback Control*. Springer-Verlag GmbH, Jul. 1, 2015, ISBN: 1447166744.

[26] F. León, *Signale und Systeme*. BerlinBoston, Mass: De Gruyter Oldenbourg, 2015, ISBN: 9783110403855.

[27] J. G. Ziegler and N. B. Nichols, "Optimum Settings for Automatic Controllers'', *Transactions of the ASME*, vol. 64, pp. 759–768, 1942.

[28] I. D. Díaz-Rodríguez, S. Han, and S. P. Bhattacharyya, *Analytical Design of PID Controllers*. Springer International Publishing, May 29, 2019, 316 pp., ISBN: 3030182274.

[29] The Mathworks. (Jun. 16, 2021). Open PID Tuner for PID tuning - MATLAB pidTuner, [Online]. Available: `https://de.mathworks.com/help/control/ref/pidtuner.html` (visited on 06/16/2021).

[30] The Mathworks. (Jun. 16, 2021). PID Tuning Algorithm - MATLAB & Simulink, [Online]. Available: `https://de.mathworks.com/help/control/getstart/pid-tuning-algorithm.html` (visited on 06/16/2021).

[31] R. Computing. (2021). PyQt, [Online]. Available: `https://riverbankcomputing.com/software/pyqt/` (visited on 06/17/2021).

[32] The Qt Company. (2021). Qt Framework, [Online]. Available: `https://doc.qt.io/` (visited on 06/15/2021).

[33] J. D. Hunter, "Matplotlib: A 2D graphics environment'', *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007. DOI: `10.1109/MCSE.2007.55`.

[34] L. Campagnola. (2011). PyQtGraph - Scientific Graphics and GUI Library for Python, [Online]. Available: `https://www.pyqtgraph.org/` (visited on 06/14/2021).

[35] The Qt Company. (2021). QTimer Class, [Online]. Available: `https://doc.qt.io/qt-5/qtimer.html` (visited on 06/12/2021).