

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

1	Controller Design and Evaluation	1
1.1	Plant Identification	1
1.1.1	Principle	1
1.1.2	Identifying the Plant Attenuator+RF	1
1.2	Measurement Filter	4
1.3	Controller Design	9
1.3.1	Choosing a Controller Type	9
1.3.2	Designing a Discrete Time PID Controller	9
1.3.3	Analyzing the Stability and Disturbance Rejection	9
1.3.4	Offline Evaluation with Measured Data	9
1.4	Software Design	10
1.5	Online Evaluation of the Control System	11
	Appendix	15
A	Lab Test and Measurement Equipment	15
	Acknowledgments	19

List of Figures

1.1	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_{cavity}	2
1.2	Validation of the estimated process models for the plant; the legend also shows the model fits in percent	3
1.3	Impulse responses of a moving average filter ($N = 100$), a FIR lowpass ($N = 50$, $f_c = 0.1$ Hz) and a IIR Butterworth lowpass ($N = 50$, $f_c = 0.1$ Hz)	5
1.4	Effects of the three different lowpass filters in Figure 1.3 on noisy data	6
1.5	Pole-Zero maps for two FIR filters with a common cutoff frequency $f_c = 10$ mHz but different filter orders N ; \circ denotes zeros, \times denotes poles, (n) is a n -times pole	7
1.6	Magnitude response of two FIR filters with a common cutoff frequency $f_c = 10$ mHz but different filter orders N	7
1.7	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)	11

1.8	Spectrogram of the cavity power in Figure 1.7	11
1.9	Comparison between the control system on and off	12
1.10	Relative standard deviation $STD\%(t, T)$	13
1.11	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 $T = T_0$	13
1.12	Power spectrum of the plots in Figure 1.7 computed with Welch's method; shaded areas show the uncertainty according to ??	14

List of Tables

1.1	Process models of the plant as estimated by the Matlab System Identification Toolboxes process model estimator	3
1.2	Quantitative assessment of the controllers performance	12
A.3	Agilent 34411A specifications	15
A.4	Agilent 34411A some SCPI commands	15
A.5	Keysight 34470A specifications	15
A.6	Keysight 34470A some SCPI commands	15
A.7	Keysight 34972A specifications	16
A.8	Keysight 34972A some SCPI commands	16
A.9	Tektronix MSO64 specifications	16
A.10	Tektronix MSO64 some SCPI commands	16
A.11	Rohde and Schwarz SMC100A specifications	16
A.12	Rohde and Schwarz SMC100A some SCPI commands	17
A.13	HP E4419B specifications	17
A.14	HP E4419B some SCPI commands	17
A.15	Agilent E5071C specifications	17
A.16	Holzworth HA7062C specifications	17

1. 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 ??, 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.

1.1 Plant Identification

1.1.1 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, 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.

1.1.2 Identifying the Plant Attenuator+RF

The input PRBS signal is generated with the Python script in ??. 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 ??, this equals a span in attenuation of 6.892 dB to

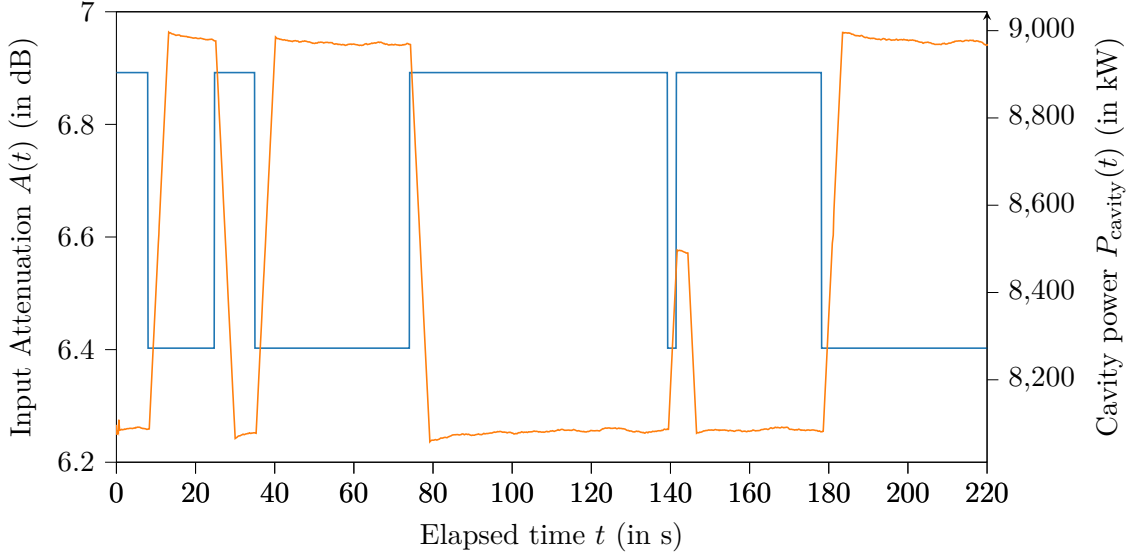


Figure 1.1: Section of the input sequence (blue) and the system response (orange); Note the inverse relation: A higher attenuation A causes a lower cavity power P_{cavity}

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

Listing 1.1: 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 1.1.

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 ≈ 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, continuous time transfer functions with different numbers of zeros and poles are estimated. 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 1.2). 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 1.1 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.

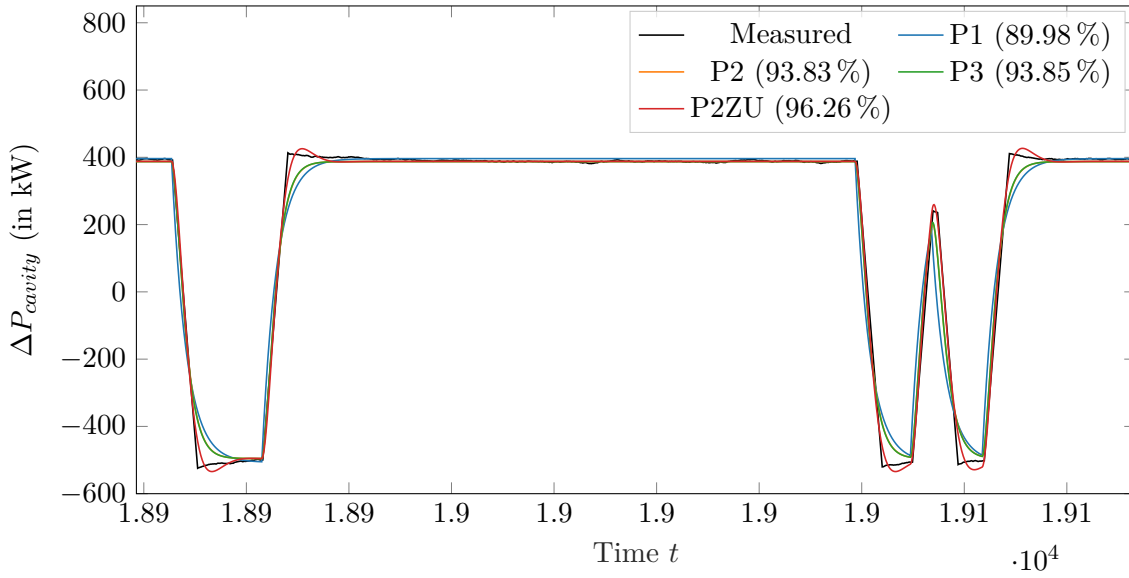


Figure 1.2: Validation of the estimated process models for the plant; the legend also shows the model fits in percent

Table 1.1: 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		$(s + 1 \times 10^6)(s + 0.7376)(s + 0.669)$	-8.9019×10^8	93.85 %
P2ZU	$(s - 2846)$	$(s^2 + 0.8014s + 0.3195)$	0.20296	96.26 %

Figure 1.2 and Table 1.1 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

$$P(s) = \frac{0.6352s - 1808}{3.13s^2 + 2.508s + 1}. \quad (1.1)$$

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

$$P[z] = \frac{-10.91z^{-1} - 10.41z^{-2}}{1 - 1.84z^{-1} + 0.8519z^{-2}}. \quad (1.2)$$

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

1.2 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:

$$\text{MA}_{x,\text{non-causal},L}[n] = \frac{1}{L} \sum_{i=n-\frac{L-1}{2}}^{n+\frac{L-1}{2}} x[i] \quad (1.3)$$

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

$$\text{MA}_{x,\text{causal},L}[n] = \frac{1}{L} \sum_{i=n-(L-1)}^n x[i] \quad (1.4)$$

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 $h[n]$:

$$h[n] = \frac{1}{N} \underbrace{[1, 1, \dots, 1]}_N \quad (1.5)$$

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

$$\frac{\tau_g(f)}{T_s} = \frac{d\phi(f)}{df} \quad (1.6)$$

which is given normalized to the sampling time T_s [19, 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 [19, p. 165]:

$$\frac{\tau_g(f)}{T_s} = \frac{d\phi(f)}{df} = \frac{d\phi}{df} = \frac{N}{2} \quad (1.7)$$

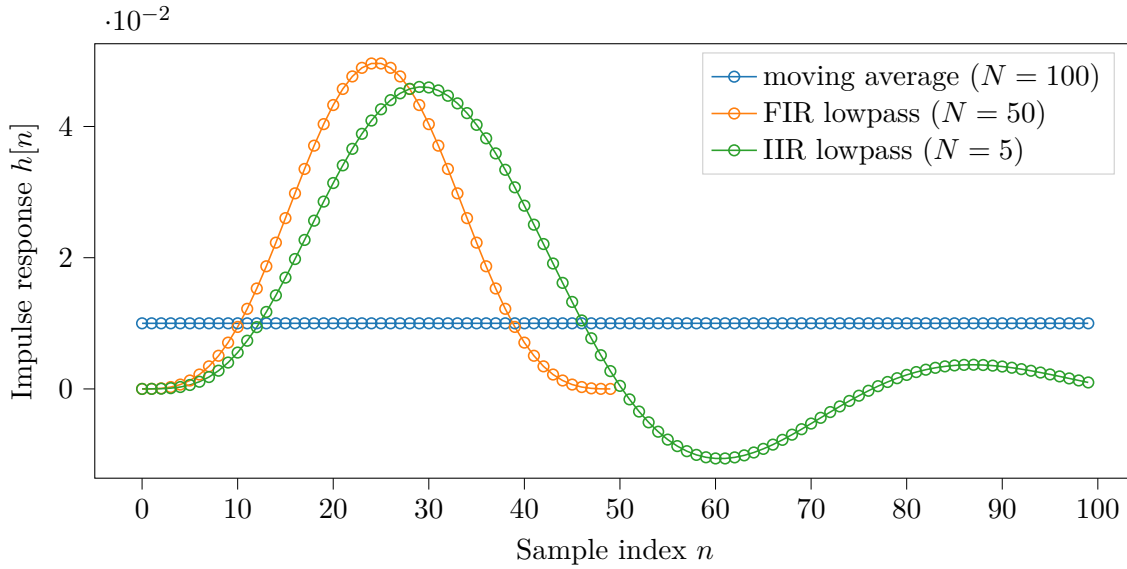


Figure 1.3: Impulse responses of a moving average filter ($N = 100$), a FIR lowpass ($N = 50$, $f_c = 0.1$ Hz) and a IIR Butterworth lowpass ($N = 50$, $f_c = 0.1$ Hz)

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 $h_{\text{FIR}}[n]$. [20, 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. [20, 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 1.7 and it is in general frequency depend.

Figure 1.3 shows the impulse responses of the moving average, the FIR lowpass and the IIR lowpass (truncated to $N = 100$).

In Figure 1.4, 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 delay², 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

²In 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.

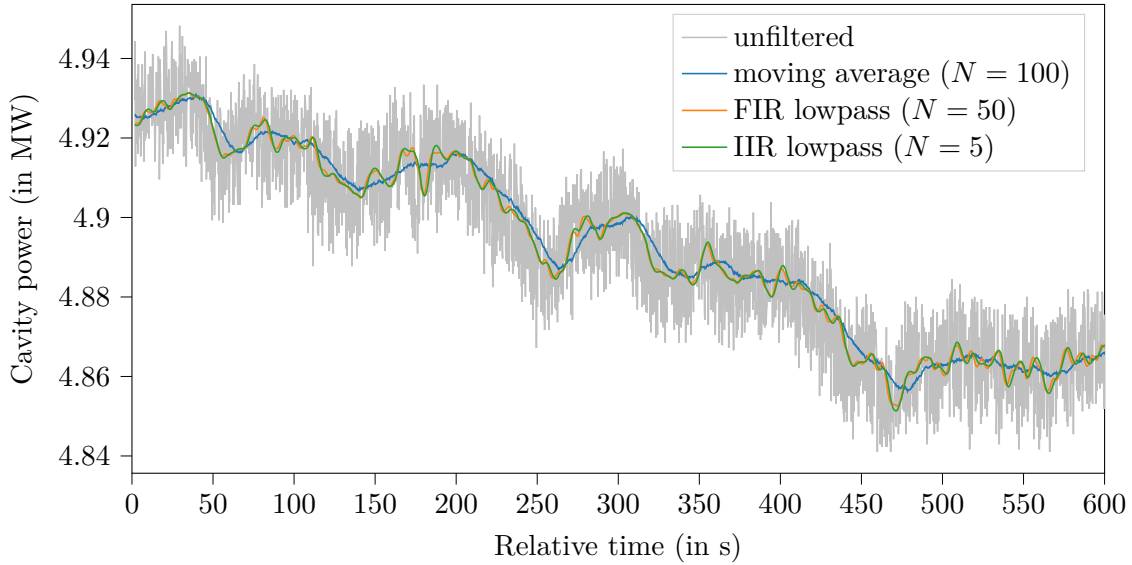


Figure 1.4: Effects of the three different lowpass filters in Figure 1.3 on noisy data

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

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 $f_c = 10$ MHz and order $N = 10$ has the transfer function

$$H[z] = \frac{1}{b_{10}z^{10} + b_9z^9 + b_8z^8 + b_7z^7 + b_6z^6 + b_5z^5 + b_4z^4 + b_3z^3 + b_2z^2 + b_1z + b_0} \quad (1.8)$$

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

$$\vec{b} = [0.0876, 0.0896, 0.0911, 0.0922, 0.0929, 0.0931, 0.0929, 0.0922, 0.0911, 0.0896, 0.0876] \quad (1.9)$$

Instead of stating the transfer function, a plot of the poles and zeros is a more intuitive representation. In Figure 1.5, the poles and zeros of the FIR filter with $N = 10$ are compared to one with $N = 40$. In addition, Figure 1.6 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.

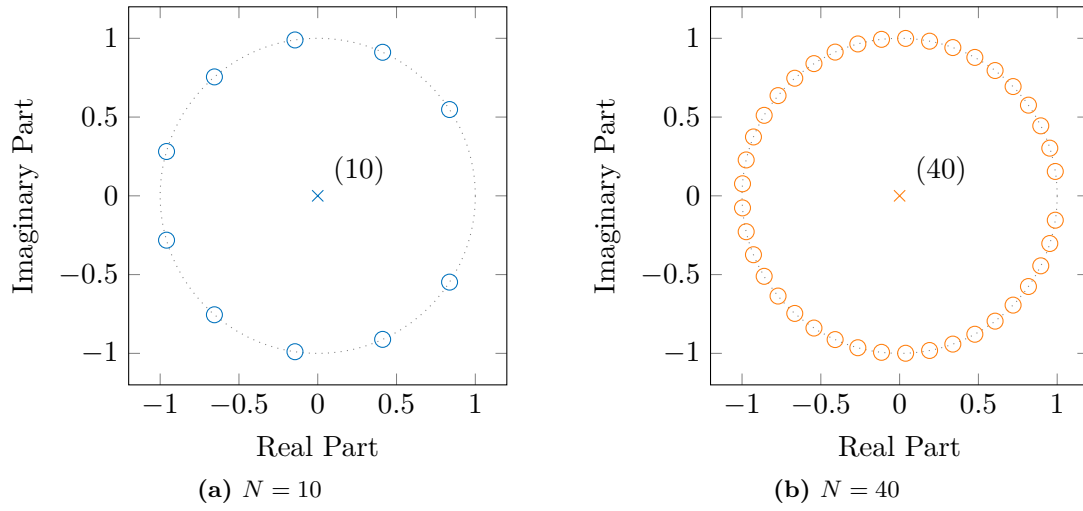


Figure 1.5: Pole-Zero maps for two FIR filters with a common cutoff frequency $f_c = 10$ mHz but different filter orders N ; \circ denotes zeros, \times denotes poles, (n) is a n -times pole

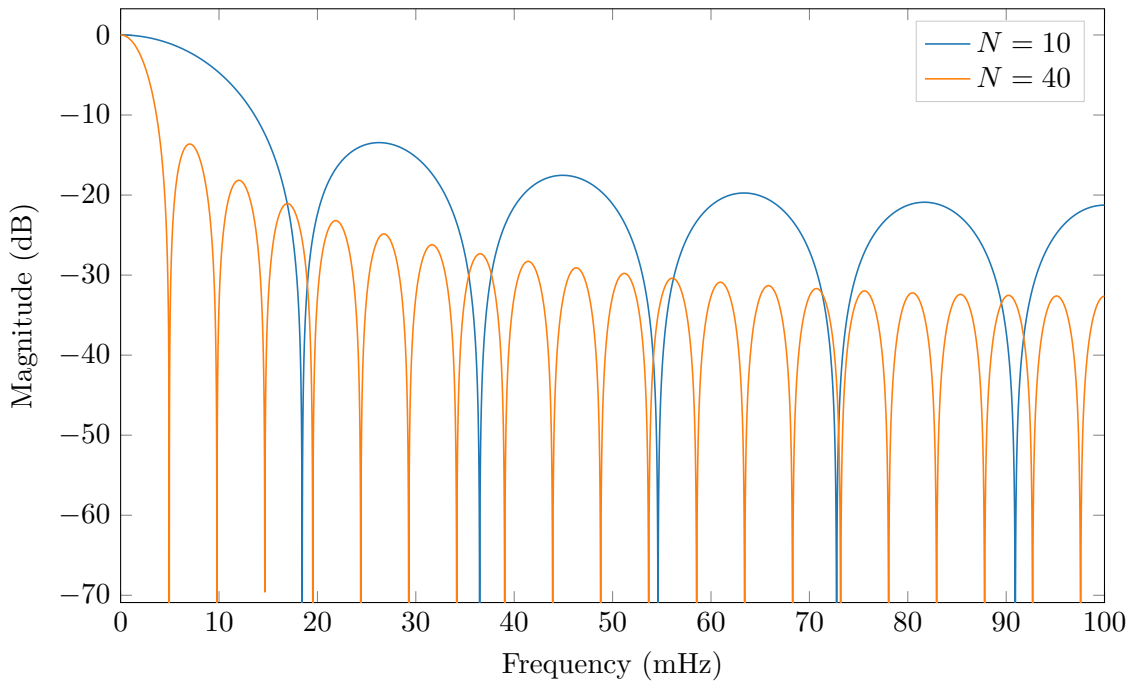


Figure 1.6: Magnitude response of two FIR filters with a common cutoff frequency $f_c = 10$ mHz but different filter orders N

Real-time implementation of a FIR Filter in Python

Similar to Equation 1.4, a FIR filter designed with the SciPy function `signal.firwin()` can be used in a causal manner to filter real-time data.

Applying the filter on pre-recorded data, like in Figure 1.4 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 1.2 by looping through pre-recorded data point-by-point.

Listing 1.2: 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])
```

1.3 Controller Design

Based on the estimated model of the plant $P(s)$ (or $P(z)$) in ?? and the type of

1.3.1 Choosing a Controller Type

1.3.2 Designing a Discrete Time PID Controller

1.3.3 Analyzing the Stability and Disturbance Rejection

1.3.4 Offline Evaluation with Measured Data

1.4 Software Design

As integrating a new subsystem into EPICS takes some time and effort and the control system is designated a temporary solution, it is more viable to operate it as an independent system.

Before choosing a programming language, software frameworks, etc. the key requirements for the software are discussed:

- Communication with EPICS to get values and with the RF attenuator
- Efficient and lightweight to achieve clock cycles times of a most 0.1 s
- Easy implementation of a (time discrete) PID controller
- GUI to show input, output and error signals
- Possibility to log signals to file for documentation

With these in mind first programming languages are regarded. As there are EPICS libraries for both C++ and Python, these two languages are examined in more detail.

While C++ as a compile language promises speed, all other requirements are possible but would take much greater effort in C++ compared to Python. For that reason, in the following a small test program is written to evaluate the fastest clock cycle possible with a simple Python program.

shows that retrieving one value of an EPICS channel and setting a new attenuator voltage takes only about 20 ms, thus using C++ is not necessary and Python can be utilized instead.

To create a PID controller in software instead of a continuous time system, only discrete time implementations are possible. Choosing a high clock cycle frequency however approximates the continuous time system. To get the error signals for the integral and derivative part, the integral is replaced with a cumulative recursive sum as

$$e_i[n] = e_i[n - 1] + e[n] \cdot dt, \quad (1.10)$$

while the derivative is replaced by a difference

$$e_d[n] = \frac{e_i[n - 1] - e[n]}{dt}. \quad (1.11)$$

For the GUI a common framework should be used. In Python Tk and wxwidgets are common ways to build a GUI. Another viable option is PyQt, which, as the name suggests, is a Python port of the Qt framework. One major advantage of using PyQt is the possibility to import .ui files describing the GUI directly from the Qt RAD designer called “Qt designer”, removing the need to create the GUI programmatically. Furthermore using PyQt enables the usage of *pyqtgraph*, a highspeed plotting library only compatible with Qt. This ensures plotting live data does not bottleneck performance, which is often the problem with naive *matplotlib* based solutions.

To log all relevant data from RAM to non-volatile memory (hard drive or network share), a simple approach with a line by line CSV writer is used.

1.5 Online Evaluation of the Control System

To evaluate the performance of the control system, 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 1.7. Note, that in this test about 7 h into the experiment there was an unintentional shutdown of FLUTE and the corresponding block of data is removed before further processing.

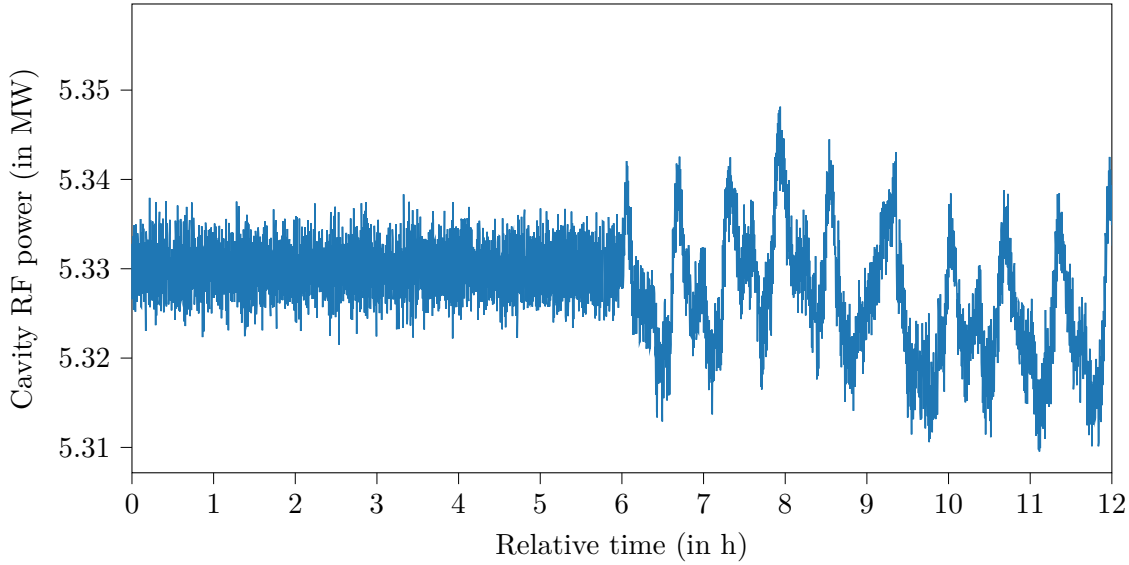


Figure 1.7: Cavity power over about 15 h (about three hours of downtime removed for clarity around the 7 h mark); control system switched off at 6 h (recording started 2021/05/01 20:00)

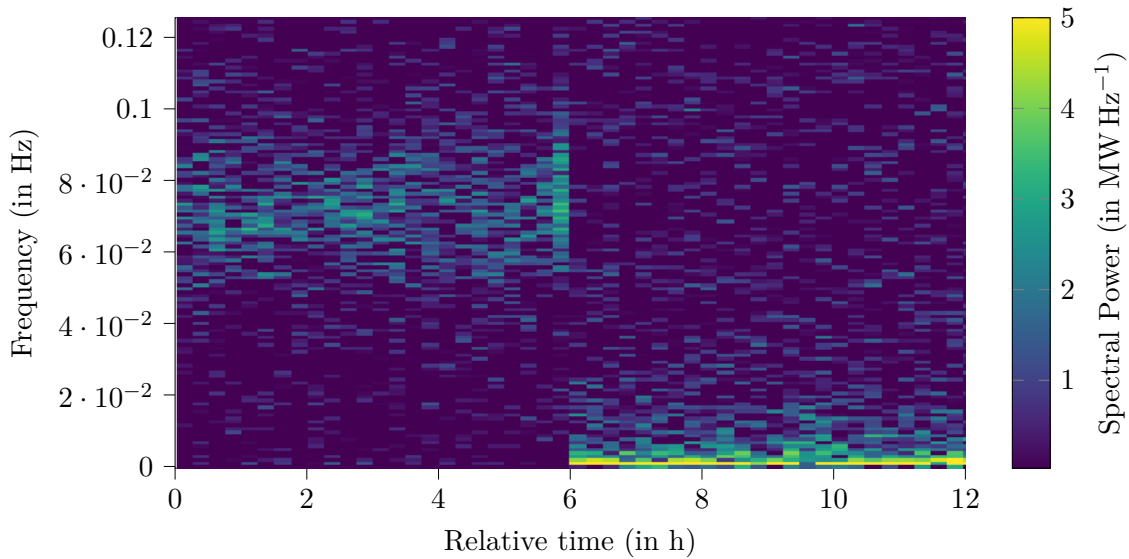


Figure 1.8: Spectrogram of the cavity power in Figure 1.7

The time plot and also the spectrogram in Figure 1.8 shows the cavity RF power approximately reaches stationarity in $[0, 2]$ hour respectively $[6, 12]$ hour. This allows the spectrum for these two blocks to be estimated using a periodogram method. In Figure 1.8 the spectra for each case are shown.

With the metrics from ??, the success of the control system is assessed in Table 1.2.

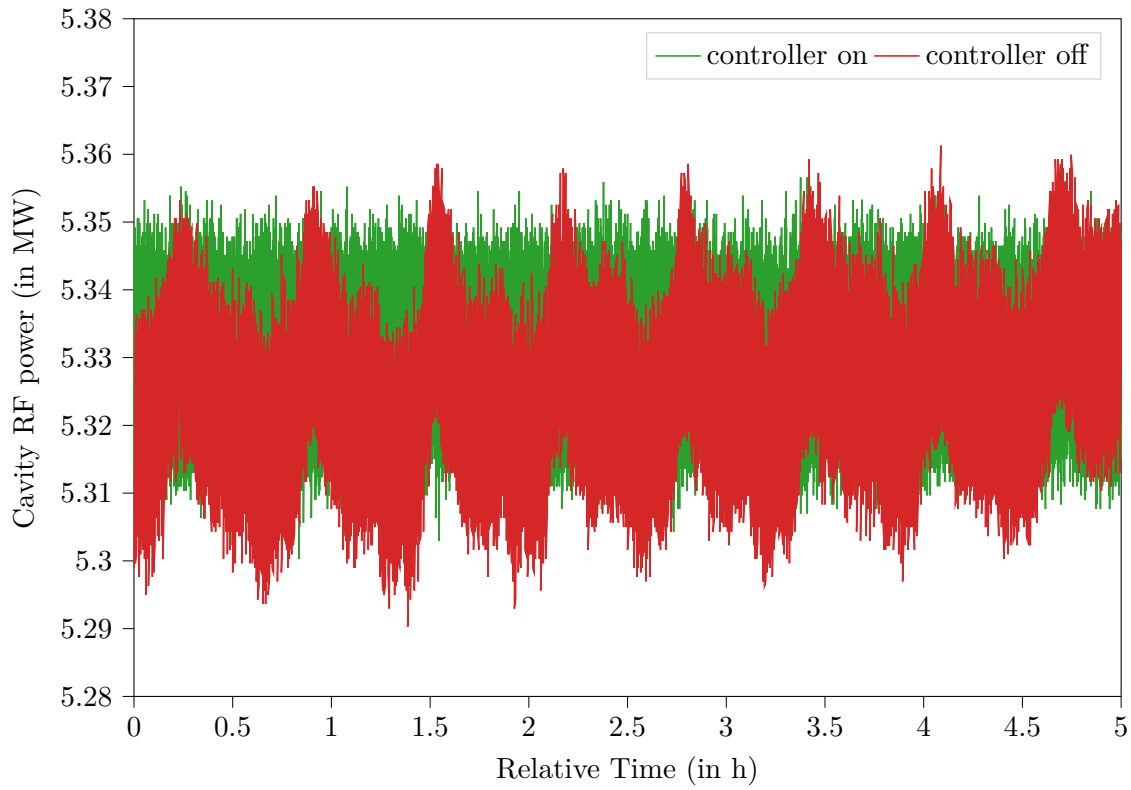


Figure 1.9: Comparison between the control system on and off

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

Table 1.2: Quantitative assessment of the controllers performance

Metric	Controller off	Controller on	Controller off/Controller on
% <i>STD</i>	0.001 155 9	$5.992\,25 \times 10^{-5}$	9.2891
<i>MSE</i>	37.639	0.101 33	371.44
<i>MPN</i>	487 309.29	14 386.25	33.873

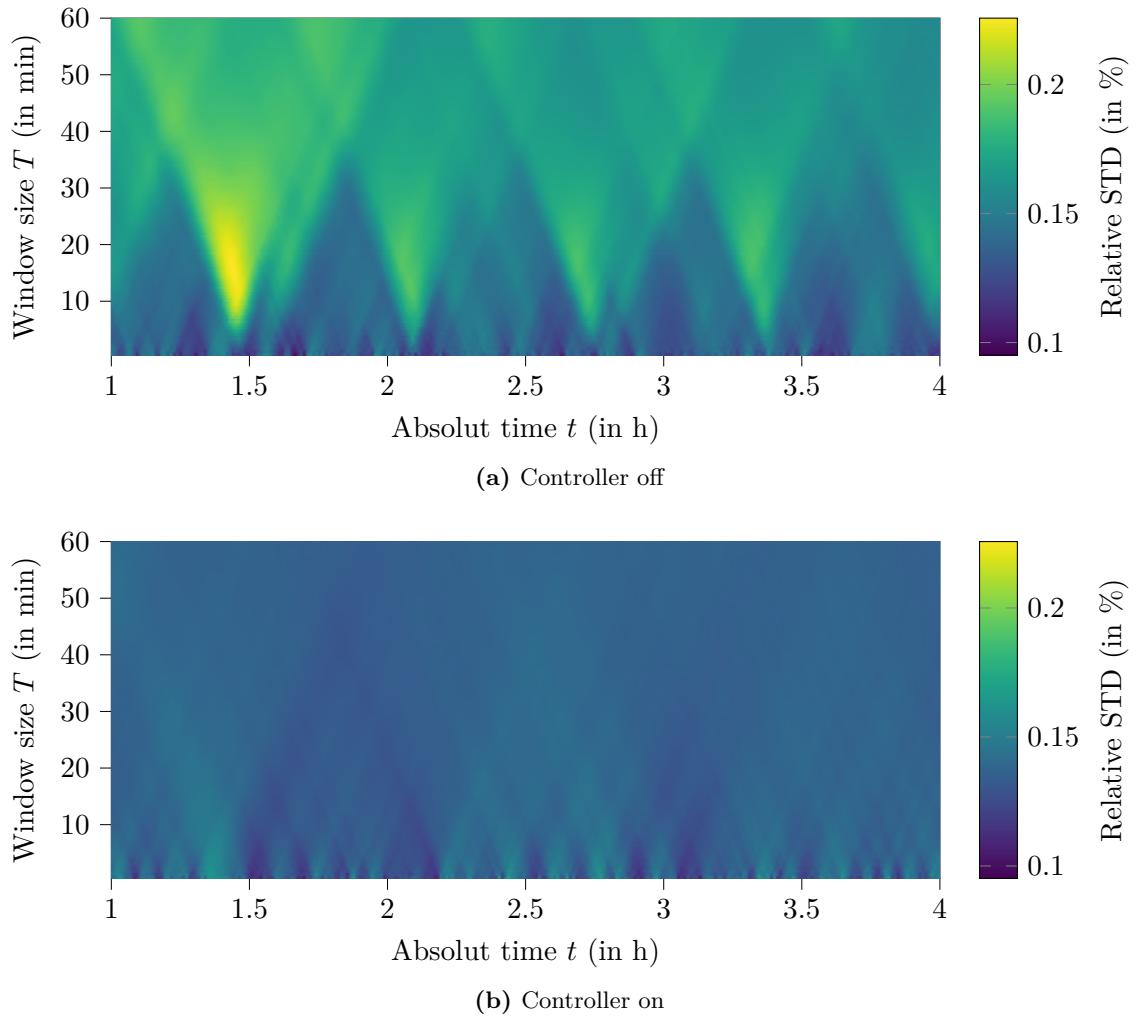


Figure 1.10: Relative standard deviation $STD\%(t, T)$

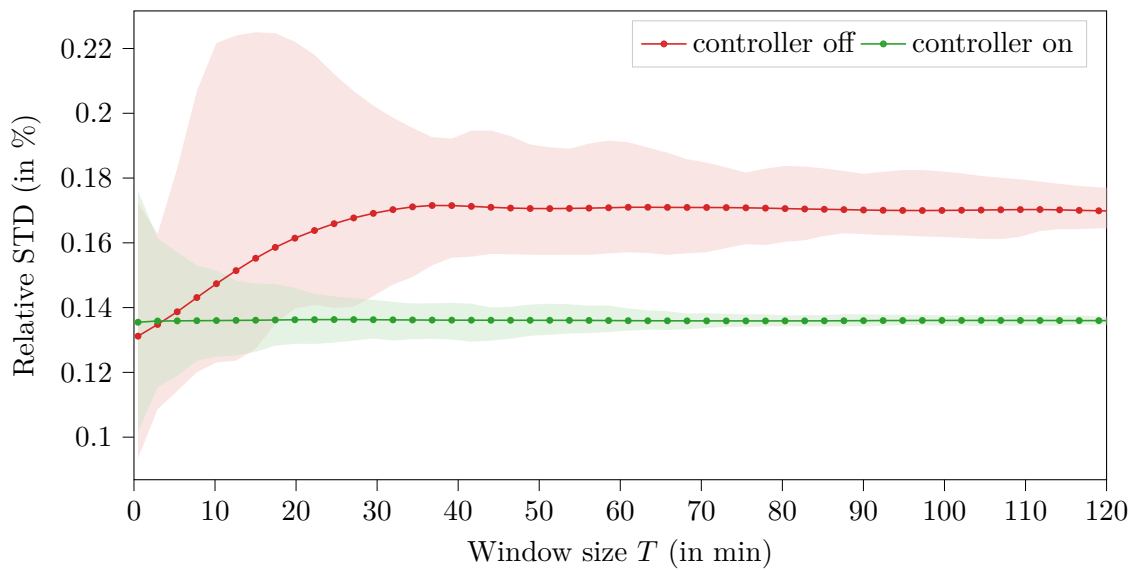


Figure 1.11: Relative standard deviation $STD\%(T)$, shaded areas show $\min\{STD\%(t, T_0)\}$ and $\max\{STD\%(t, T_0)\}$, solid lines show $\text{mean}\{STD\%(t, T_0)\}$ for a fixed window size $T = T_0$

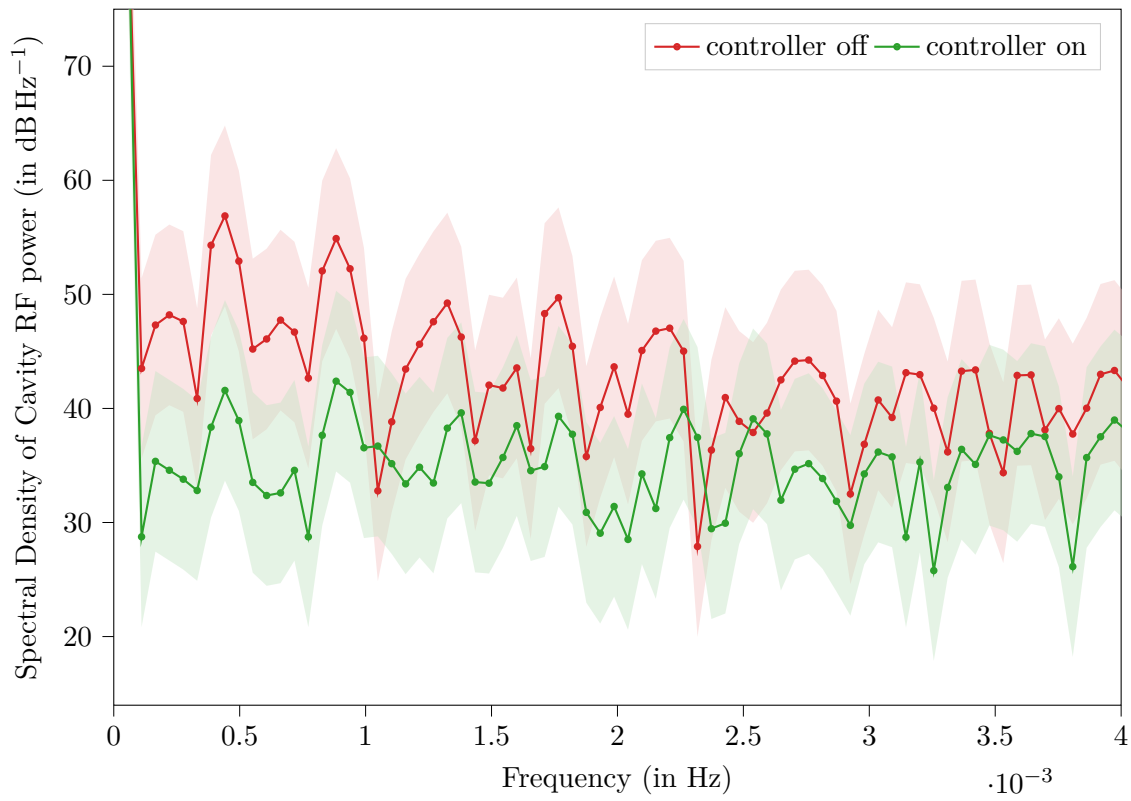


Figure 1.12: Power spectrum of the plots in Figure 1.7 computed with Welch's method; shaded areas show the uncertainty according to ??

Appendix

A Lab Test and Measurement Equipment

A.1 Benchtop multimeters

A.1.1 Agilent 34411A

Table A.3: 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 A.4: Agilent 34411A some SCPI commands

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

A.1.2 Keysight 34470A

Table A.5: 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 A.6: Keysight 34470A some SCPI commands

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

A.2 Data Acquisition/Switch Unit

A.2.1 Keysight 34972A

Table A.7: Keysight 34972A specifications

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

Table A.8: 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)	

A.3 Oscilloscopes

A.3.1 Tektronix MSO64

Table A.9: 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 A.10: Tektronix MSO64 some SCPI commands

Description	Example command	Example return
Read mean of measurement 1 (current acq.)	MEASUREMENT:MEAS1:RESULTS:CURR:MEAN?	3.0685821787408

A.4 RF signal generator

A.4.1 Rohde and Schwarz SMC100A

Table A.11: 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 A.12: 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	

A.5 RF power meter

A.5.1 HP E4419B

Table A.13: 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 A.14: HP E4419B some SCPI commands

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

A.6 Vector Network Analyzer

A.6.1 Agilent E5071C

Table A.15: Agilent E5071C specifications

Specification	Value
Frequency range	9 kHz to 8.5 GHz

A.7 Phase noise analyzer

A.7.1 Holzworth HA7062C

Table A.16: Holzworth HA7062C specifications

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

Acknowledgments

A

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 π Attenuator”, vol. 35, no. 5, pp. 280–284, 1992.
- [18] The LXI Consortium. (May 29, 2021). LXI instrument communication, [Online]. Available: <https://www.lxi-standard.org/about/vxi-11-and-lxi.aspx>.

- [19] Kammeyer, *Digitale Signalverarbeitung Filterung und Spektralanalyse*. StuttgartLeipzig-Wiesbaden: Teubner, 2002, ISBN: 9783519461227.
- [20] A. Oppenheim, *Discrete-time signal processing*. Upper Saddle River, NJ: Pearson, 2010, ISBN: 9780131988422.