**Department of Physics and Astronomy**
**Heidelberg University**

Bachelor Thesis in Physics
submitted by

**Ronja Hinterding**

born in Münster (Germany)

**2025**

# Towards a Multidimensional Calibration of Neuromorphic Hardware Using a Parameter Transformation Model

This Bachelor Thesis has been carried out by Ronja Hinterding at the
Kirchhoff-Institut für Physik in Heidelberg
under the supervision of
Prof. Dr. Johannes Schemmel

**Abstract**

Analog neuromorphic hardware is subject to fixed-pattern noise stemming from the manufacturing process and resulting in different analog behaviour in identically designed components. Calibration counteracts this mismatch by finding a set of hardware parameters that yield a desired behaviour. BrainScaleS-2 is a mixed-signal neuromorphic hardware platform emulating spiking neural networks. The current calibration framework for BrainScaleS-2 only supports single operation point calibrations, meaning that for each different calibration target, a new calibration needs to be run, which is time-consuming. Thus, the goal of this thesis is to start developing a parameter transformation model which supplies hardware parameter settings for arbitrary model parameters. As a proof of concept the transformation model is constructed and evaluated on two parameters of the leaky integrate-and-fire neuron. As these two parameters exhibit dependencies on each other's hardware parameter, a joint transformation is developed. Even though the calibration using the transformation shows some systematic deviations, its accuracy is comparable to the fixed-point calibration leading to the conclusion that the results indicate potential for a transformation model encompassing all parameters.

**Zusammenfassung**

Analoge neuromorphe Hardware weist zeitlich konstante Variationen auf, welche durch den Herstellungsprozess verursacht werden und zu unterschiedlichem Verhalten zwischen identisch entworfenen Komponenten führt. Kalibration wirkt diesen Abweichungen entgegen, indem sie einen Satz an Hardwareparametern findet, die ein gewünschtes Verhalten der Hardware bewirken. BrainScaleS-2 ist eine neuromorphe Hardwareplattform, die gepulste neuronale Netze emuliert. Die aktuelle Kalibration für BrainScaleS-2 unterstützt ausschließlich Kalibrationen auf einen einzelnen Operationspunkt, das bedeutet, dass für jedes unterschiedliche Kalibrationsziel eine neue Kalibration ausgeführt werden muss, was zeitaufwendig ist. Deshalb ist das Ziel dieser Arbeit, ein Transformationsmodell zu entwickeln, welches die Hardware-Einstellungen für beliebige Modellparameter liefert. Um die Umsetzbarkeit des Konzepts nachzuweisen, wird das Transformationsmodell für zwei exemplarische Parameter des Leaky Integrate and Fire Neurons konstruiert und evaluiert. Da diese Parameter Abhängigkeiten vom jeweils anderen Hardwareparameter aufweisen, wird eine gemeinsame Transformation entwickelt. Obwohl die Kalibration mit der Transformation systematische Abweichungen aufweist, ist ihre Genauigkeit vergleichbar zur aktuellen Kalibration. Die Ergebnisse deuten darauf hin, dass ein Transformationsmodell aller Parameter möglich ist.

# Contents

# 1 Introduction

With recent advancement and rising relevance of artificial neural networks (ANN) in various fields, high computational costs and energy consumption have become a prominent concern [7]. An alternative to ANNs is the so-called third generation of neural networks, the spiking neural networks (SNN) [13], which are expected to improve the computational performance and efficiency of neural networks, especially on specifically designed hardware [5].

The mixed-signal neuromorphic platform BrainScaleS-2 (BSS-2) [14] which emulates spiking neural networks is used in this thesis. The analog circuits of the system suffer from fixed-pattern noise, i.e. temporally constant systematic deviation of parameters from the values targeted during chip design [19]. These deviations are an unavoidable result of the manufacturing process and cause differing analog behaviour (e.g. differing time constants) between identically designed instances when choosing the same hardware settings. Calibration is capable of reducing this fixed-pattern noise. The concept of calibration is to find the hardware settings that yield a targeted analog behaviour. Therefore, calibration can on the one hand equalize the behaviour across identically designed hardware components or, if intended, can achieve targeted behaviour that differs across these components.

The current calibration [18] is a single-operation-point calibration, which means that for each different calibration target a new calibration needs to be run. For most parameters the calibration routines perform a (noisy) binary search on the digital settings which is time-consuming due to the large number of measurement iterations. One can use the programmable plasticity unit (PPU) [12] to speed up the calibration, however, this is still a single-operation point approach.

In contrast, the calibration routines for the preceding BrainScaleS-1 system employ a lookup-based approach [15], where the model parameters (meaning parameters describing the analog behaviour) are measured as a function of the hardware settings once. Then, the obtained data is used to create a translation from model parameters to hardware parameters, which can then be used for calibration without having to measure again. A prerequisite for the feasibility of this lookup-based approach is the long term stability of the fixed-pattern noise, since the characterization of the hardware should be done once and will then be reused. The long term stability was shown in [18] for BSS-2.

Experiments that would benefit from the lookup-based approach compared to the fixed-point calibration are experiments where different calibration targets need to be tested beforehand or experiments which try to learn the model parameters which entails frequent changes of the calibration target during training.

Hence, the goal of this thesis is to develop a lookup-based parameter transformation model for BSS-2, which provides the hardware parametrization for arbitrary calibration targets.

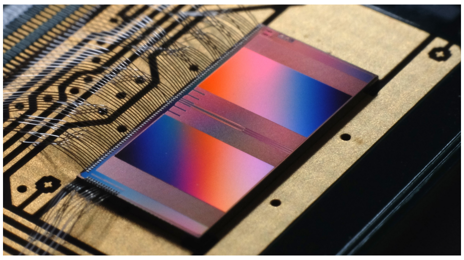
In this thesis, the transformation model will be developed for two exemplary parameters and in the end evaluated by calibrating a chip using the transformation. The circuits of the chosen parameters are designed such that one hardware parameters controls one model parameter. The transformation model will be constructed by measuring model parameters as a function of their digital hardware setting and then fitting a function to the obtained data. The resulting function forms the transformation model. Due to the fixed-pattern noise, the resulting functions for identically designed instances will be of the same form but differently parametrized. Since some model parameters exhibit dependencies not only on their respective hardware parameters, but also on the hardware parameters of other model parameters, thought needs to be put into how to implement these multidimensional transformations.

# 2 Methods

## 2.1 Hardware Setup

In this thesis, the HICANN-X v3, the current version of the BSS-2 [14] system, is used.

Figure 2.1 shows an image and a schematic arrangement of the chip. This section will describe the relevant parts of the chip for this thesis.



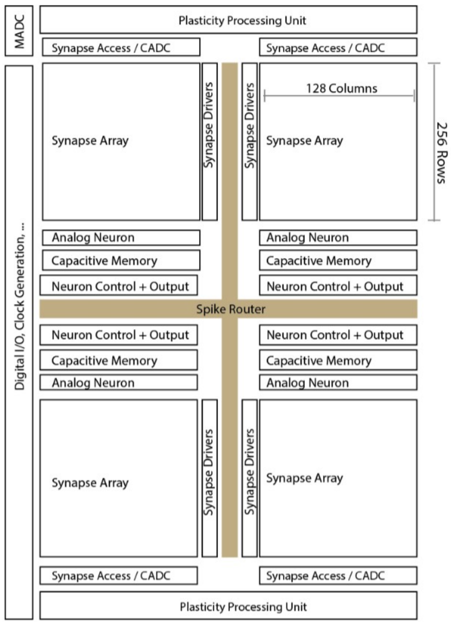



Figure 2.1: Left: Image of the HICANN-X chip. Right: Schematic floorplan of the chip. One can see the division into four quadrants with 128 analog neuron circuits and a parameter storage system, the capacitive memory (CapMem), each. Also shown are the columnar analog-to-digital converters (CADC) and the single membrane analog-to-digital converter (MADC). Taken from [14].

The chip is divided into four quadrants with 128 neuron circuits each. Each neuron receives input from 256 synapses, resulting in a total number of 131 072 synapses. There are two different analog-to-digital converters dedicated to digitizing the potentials in the neuron circuits: the columnar analog-to-digital converters (CADC) and the membrane analog-to-digital converter (MADC).

There are 256 CADC channels per quadrant, since there is a causal and an acausal channel for each neuron. This also allows the CADC to record the potential of all neurons in parallel. In contrast, there is only one MADC with two channels per chip, which implies that it can not read all neurons in parallel like the CADC. The advantage of the MADC over the CADC is a higher sampling frequency of approximately 30 MHz compared to a sampling frequency of around 1 MHz of the CADC.

Next to each neuron block of 128 neurons, the parameter storage system, the capacitive memory (CapMem) [11], is located. It stores the analog parameters of the neuron circuits as well as quadrant-global parameters. It will be described in section 2.3.

The chip is controlled in real-time by sending instruction from a host computer to a field programmable gate array (FPGA).

## 2.2 The Neuron Circuit

The neuron circuits of the HICANN-X chip emulate the adaptive exponential integrate-and-fire (AdEx) model [6]. In this thesis, only the leaky integrate-and-fire (LIF) [1] part of this model is used. The model describes the temporal evolution of the membrane potential, as well as defining a threshold for spiking. The following equation describes the subthreshold dynamics of the membrane potential $V$ according to the LIF model:

$$\tau_{\mathrm{mem}}\dot{V} = -[V(t) - V_{\mathrm{leak}}] + \frac{I(t)}{g_{\mathrm{leak}}} \tag{2.1}$$

with the leak conductance $g_{\mathrm{leak}}$ that pulls the membrane potential to the leak potential $V_{\mathrm{leak}}$, the membrane time constant $\tau_{\mathrm{mem}}$ which is given by $\tau_{\mathrm{mem}} = C_{\mathrm{mem}}/g_{\mathrm{leak}}$ where $C_{\mathrm{mem}}$ is the membrane capacitance, and the current $I$ which is the sum of an external current and excitatory and inhibitory synaptic currents. When the membrane potential reaches the threshold potential $V_{\mathrm{thresh}}$, an output spike occurs and the membrane potential is pulled to the reset potential $V_{\mathrm{reset}}$ for the refractory period $\tau_{\mathrm{ref}}$.

The subthreshold solution for an initial condition of $V(t_0) = V_{\mathrm{leak}} + \Delta V$ is:

$$V(t) = \Delta V \exp\left(-\frac{t - t_0}{\tau_{\mathrm{mem}}}\right) + V_{\mathrm{leak}} \tag{2.2}$$

for $t > t_0$, meaning that the membrane potential decays exponentially back to the leak potential.

Figure 2.2 shows a schematic diagram of how the LIF neuron is realized in the circuitry of the HICANN-X.

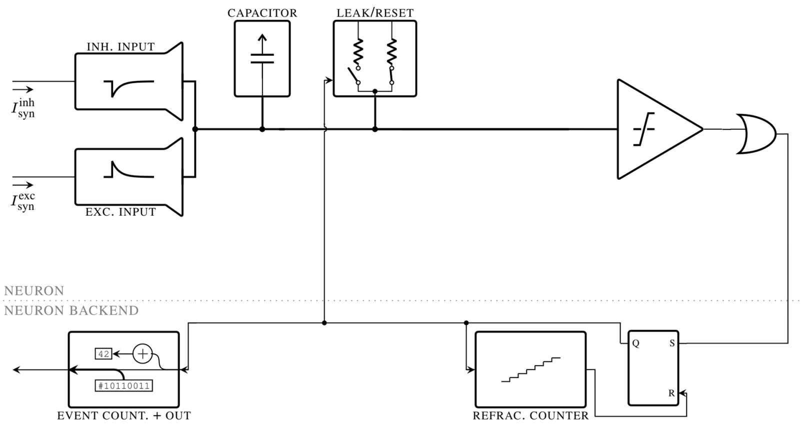



Figure 2.2: Schematic of the LIF part of the neuron circuit. The conductivity $g_{\mathrm{leak}}$ that is responsible for the leak term, is realized by a operational transconductance amplifier (OTA) which is controlled by the leak bias current $I_{\mathrm{bias\_leak}}$. The reset and leak mechanism is realized using a single OTA, when the threshold is reached the OTA is reconfigured from the leak state to the reset state. The schematic also shows the threshold comparator as well as the inhibitory and excitatory synaptic inputs which can be disabled. Adapted from [4].

## 2.3 Storage System for Analog Parameters: Capacitive Memory

There are 24 analog parameters (8 voltages and 16 currents) for each neuron that can be adjusted in order to achieve a targeted behaviour of the neuron. These parameters are stored by a capacitive memory (CapMem) [11] which consists of an array of cells where each column belongs to one neuron circuit and each row belongs to one analog parameter (cf. fig. 2.3). Each cell can store a 10 bit value, which is converted to a voltage or a current. The CapMem consists of a voltage generator creating a linearly increasing voltage ramp. Simultaneous to the voltage ramp, a 10-bit quadrant-global counter is incremented, whose value is compared to the value of each cell. When the counter value matches the digital value of a cell, the storage capacitor of the cell is updated to the current value of the voltage ramp [11]. The digital cell values will be referred to as CapMem values from now on. As fig. 2.1 shows, there is one CapMem per quadrant.

A problem occurs with the current implementation of the CapMem. When a large number of cells of one CapMem, i.e. of one quadrant, is set to the same value, the stored voltages and currents differ from the value they would have if only one cell was active.

The current calibration counteracts this problem by adding a noise of $\pm 5\,\mathrm{LSB}$ [18] to the CapMem values. In [9] we evaluated the extent of this CapMem crosstalk as it might pose a problem for the parameter transformation model. The problem is that we do not want to model these complex dependencies as they lead to a large number of data points. However, the conclusion of [9] was that the parameter shift due to CapMem crosstalk in the range of

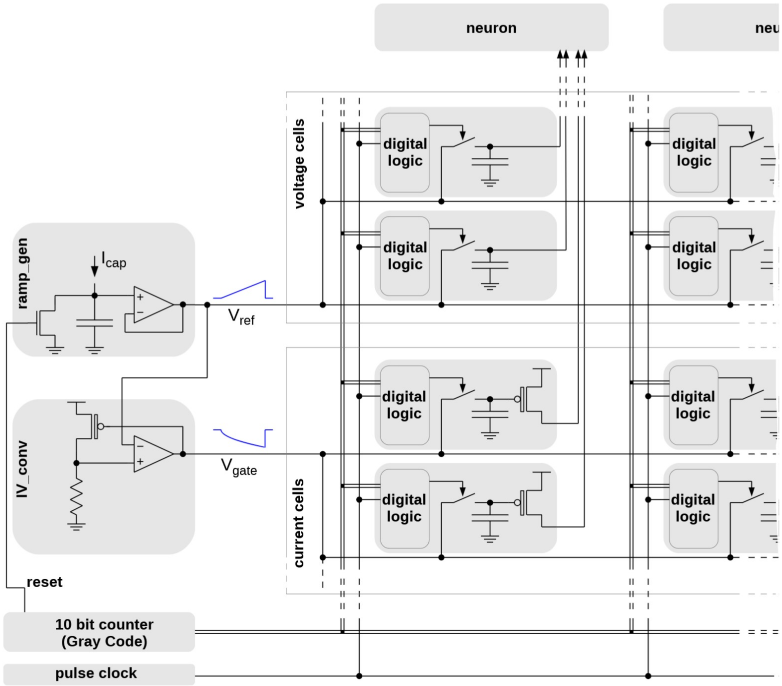



Figure 2.3: The analog parameter storage system: the CapMem. One can see the arrangement of the cell array, where one row belongs to one neuron and one column belongs to one analog parameter (current or voltage). There is one voltage ramp per quadrant. When the digital value of the cell matches the digital value of the 10 bit counter, the cell is charged to the value of the ramp. Taken from [10].

interest is small enough to be neglected in the transformation.

## 2.4 Current Calibration Framework: Calix

The current calibration framework for BSS-2 is the so-called calix library [8]. It finds the suitable hardware parameters to achieve a given target model parameter. The structure of the software will shortly be described here, since parts of it were used in this thesis and since the calibration results using calix will be compared with the results using the lookup-based parameter transformation model.

The framework consists on the one hand of calibrations, which provide the necessary methods for each model parameter to be measured and for the respective hardware parameter to be configured. On the other hand, there are algorithms which define how the optimal hardware setting is found.

The base of the software is formed by a calibration class with the following methods: run(), prelude(), postlude(), measure_results() and configure_parameters(). The run() method is called for executing a calibration of a certain parameter. It takes the target model parameters for all neurons as an argument, as well as an algorithm and a connection to the chip that is calibrated. The run() method first calls prelude() to configure the chip for the calibration and to perform other preparations. After preparing the measurement, the model parameter is measured and then the hardware parameter is changed according to the algorithm by using configure_parameters(). The process of measuring and configuring hardware parameters according to the algorithm is repeated until the optimal hardware setting is found. Lastly, the postlude() method is called to apply necessary configurations of the chip after calibration.

A noisy binary search is used as an algorithm for $\tau_{\mathrm{mem}}$ and $V_{\mathrm{leak}}$. The binary search can be used for parameters for which the model parameter is a monotonous function of the hardware parameter. As mentioned, some noise is added to the starting values of this binary search because of the CapMem crosstalk problem which occurs when a large number of cells is set to the same value.

For each parameter, there is a calibration class that is derived from the base calibration class, where prelude(), postlude(), measure_results() and configure_parameters() are implemented.

In particular, the implemented measure_results() methods are used in this thesis for measuring the model parameters. Furthermore, we will use the result of a default calix calibration to preconfigure the chip before measurements. This calibration is run and saved nightly on the setups.

### 2.4.1 Leak Potential Calibration

The measure_results() method for the leak potential calibration takes one CADC read for digitizing the resting potential. The corresponding hardware parameter, that is adjusted according to the provided algorithm, is the CapMem value for the leak potential $V_{\mathrm{leak}}^{\mathrm{CapMem}}$.

### 2.4.2 Membrane Time Constant Calibration

The membrane time constant is given by $\tau_{\mathrm{mem}} = C_{\mathrm{mem}}/g_{\mathrm{leak}}$. The calibration of the membrane time constant in calix keeps the membrane capacitance constant and tunes the leak conductivity to achieve the desired $\tau_{\mathrm{mem}}$. The leak conductivity is controlled by the OTA's bias current, which is generated and stored by a CapMem cell. This CapMem value of the leak bias current $I_{\mathrm{bias\_leak}}^{\mathrm{CapMem}}$ is the corresponding hardware parameter of the membrane time constant when keeping the capacitance fixed.

For measuring the membrane time constant, there are three different calibrations that use different methods for measuring. The method that uses the CADC for measuring was not used in this thesis. The other two use the MADC for recording a trace, i.e. measuring the membrane potential as a function of the time. They then use a fit to determine the time constant.

One of the methods measures the response to a step current, and an exponential fit to the decaying membrane potential (cf. eq. (2.2)) determines the membrane time constant. This measurement method does not generate reliable results for membrane time constants under 3 µs because the offset current is too weak to stimulate the membrane significantly. There is a leak division or multiplication mode, where either multiplication or division or neither can be enabled and which scales the leak transconductance by a factor of nine [3], which allows for wider ranges of $\tau_{\mathrm{mem}}$. With leak multiplication enabled, $\tau_{\mathrm{mem}}$ is mostly below 3 µs, which is why the offset method does not support leak multiplication.

The other method does support these short time constants. It fits an exponential to the trace after a reset. To ensure a good amplitude for the fit, the leak potential and the reset potential are changed, which means that it can not be used for measurements of the dependency of the membrane time constant on the leak potential.

# 3 Results

This section presents the key steps carried out for the construction of the transformation, as well as an evaluation of the calibration using the transformation. The model parameters are first measured as a function of the hardware setting. Then, a function is fitted to the obtained data. This function forms the transformation model for which an interface is implemented and which can then be used for a look-up based calibration.

## 3.1 Characterization of the Analog-to-Digital Converters

With the current implementation of calix, the unit of the calibration targets for all voltages is the readout value of the respective analog-to-digital converters (ADC). It is of interest to know the voltages in SI units, because the ADC read values are units which are not interpretable since they depend on the ADC configuration. This section describes how a conversion from columnar analog-to-digital converters (CADC) read values to volts is found, which is used for the evaluations.

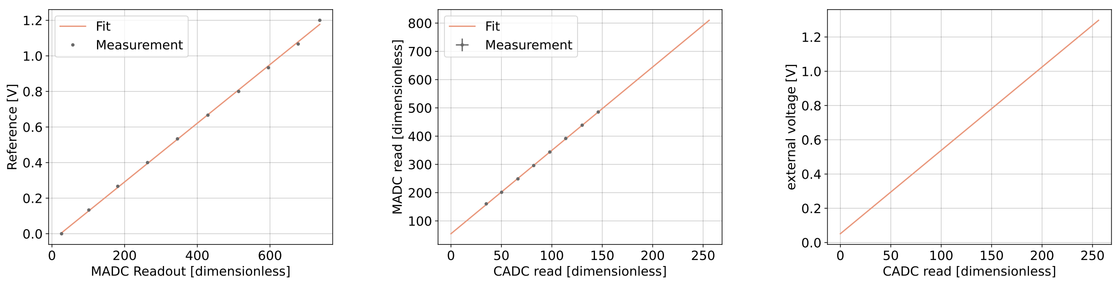



Figure 3.1: (a) Linear translation of the MADC read value to volts using this script. (b) Linear translation from CADC to MADC read value by setting the CapMem value of $V_{\text{leak}}$ and then measuring$V_{\text{leak}}$ ten times with the CADC and ten times with the MADC on one neuron. The error bars show the standard deviation over the ten measurements. (c) Composition of the first two linear functions to get a conversion from CADC read values to volts. Shown here: translation for chip W61F0 using a CADC calibration performed by calix.

There already exists a script that translates the MADC read values to volts by connecting the MADC to an external analog-to-digital converter (DAC) which is sufficiently accurate.

This conversion is linear. By finding a translation from MADC read values to CADC read values, a translation from CADC read values to volts can be inferred.

This conversion is achieved by setting the CapMem value of the leak potential of one neuron to a certain value and then measuring $V_{\text{leak}}$ once with the MADC and once with the CADC. Repeating this process for different CapMem values yields a set of points of MADC read values with corresponding CADC read values. The translation from CADC to MADC is then determined by a linear fit. Combining the two linear translations yields a conversion from CADC read values to volts.

Figure 3.1 shows the three translations for the CADC calibration that is used for all measurements in this thesis. When a voltage in volts is shown in this thesis, this translation is used. The assumption of a linear translation is not true across the entire parameter range [18]. Furthermore, there might also be inaccuracies that are caused by only measuring the translation from CADC to MADC on one neuron and thus neglecting differences between neurons caused by the readout. However, these translations are sufficient to provide a conversion for better interpretability.

## 3.2 Measurements

This section describes the procedures for measuring the model parameter as a function of its respective hardware parameter and as a function of other hardware parameters it depends on. The measurements are carried out for the leak potential and the membrane time constant, since the membrane time constant is an example for a current based parameter and the leak potential for a voltage based parameter.

Furthermore, these model parameters show dependencies on the respective hardware parameter of the other parameter: the membrane time constant not only depends on the CapMem value of the leak bias current that controls the leak conductance, but also exhibits a dependency on the CapMem value of the leak potential. Likewise, the leak potential mainly depends on its CapMem value, but is also dependent on the leak bias current.

For all following measurements, the synaptic input is disabled so that the characteristics of $\tau_{\text{mem}}$ and $V_{\text{leak}}$ can be measured in isolation. By applying a default (nightly) calibration, we assure that the CADCs are calibrated.

### 3.2.1 General Structure

The objective of the measurements is to sweep through an array of hardware settings and measure a model parameter for each setting. For this, a base class for executing the sweep as well as the measurement is created. It has a method called `sweep_one_neuron()` that first prepares the measurement, and then performs a loop over the array of hardware setting for one neuron. In each iteration, the hardware setting is applied, and then the model parameter is measured for that neuron. After the loop, the data is serialized and saved for later evaluation.

For each hardware parameter, a derived class must define how to configure the hardware settings. For CapMem values, not only the value of the cell of the neuron that is measured is set, but all neurons in order to avoid CapMem crosstalk. All other neurons are set to a value that differs by five from the value of the neuron that is swept because the parameter shift due to CapMem crosstalk occurs not only when a large number of cells is set to the same value but also for the neighbouring two values (cf. fig. 3.2). When configuring a CapMem cell to a new value, the analog value is not reached instantly. Thus, a wait of 20 ms after configuring is used, before measuring the model parameter.

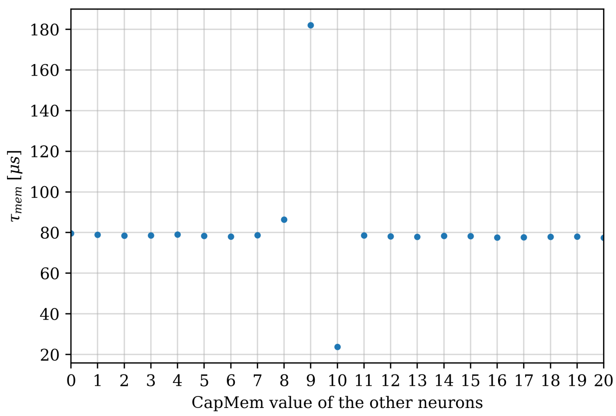



Figure 3.2: Measured membrane time constant of one arbitrarily chosen neuron as a function of the CapMem value of the leak bias current of all other neurons in the quadrant. The standard deviation over ten measurement of $\tau_{\mathrm{mem}}$ is around 0.2 µs. The CapMem value of the chosen neuron is kept constant at 10 while the value of all other 127 neurons in the quadrant was varied from 0 to 1022. The membrane time constant of the chosen neuron is affected if the CapMem value of other neurons in the quadrant have a value one or two below the value or the same value of the measured neuron's cell.

In addition to CapMem values, there are other hardware parameters that affect the model parameters. For example, the leak division or multiplication mode, where either multiplication or division or neither can be enabled and which scales the leak transconductance by a factor of nine [3]. The scaling of the conductance leads to a scaling of the membrane time constant, allowing for a wider range of the time constant. Since the membrane time constant is one of the parameters for which the transformation is developed in this thesis, the setting of this scaling factor is also swept.

The `sweep_one_neuron()` function takes a "recorder class" as an argument. This recorder class implements a method to measure a specific model parameter. These measurement methods are taken from calix and are executed multiple times in order to get an estimate for the variance in the measurement and to reduce measurement-to-measurement noise. For all measurements in this section, the number of measurements is ten.

For now, the 512 neuron circuits are swept in succession, possibilities for parallelization are described in the discussion.

Figure 3.3 and Figure 3.4 show that the leak potential and the membrane time constant exhibit significant dependencies on each other's hardware parameter, leading to the conclusion that one dimensional, independent transformations for the two parameters would not represent the hardware's behaviour accurately enough to get a good calibration. Therefore, both model parameters are measured as a function of the $V_{\text{leak}}^{\text{CapMem}}$ as well as the $I_{\text{bias\_leak}}^{\text{CapMem}}$. Since the leak potential is expected to depend linearly on $V_{\text{leak}}^{\text{CapMem}}$, the $V_{\text{leak}}^{\text{CapMem}}$ is swept for 20 equidistant values from 0 to 1000. For $I_{\text{bias\_leak}}^{\text{CapMem}}$ a logarithmically spaced list of 18 setting between 0 and 1022 was chosen due to the hyperbolic dependency of the membrane time constant on the leak bias current [3]. For all measurements in this section, a grid of all the combinations of these lists of settings is swept.

### 3.2.2 Leak Potential

The recorder for the leak potential simply uses the measure_results() function of the leak potential calibration class from calix to measure the leak potential using the CADC.

Figure 3.3 shows the measured $V_{\text{leak}}$ as a function of the $V_{\text{leak}}^{\text{CapMem}}$ for different setting of $I_{\text{bias\_leak}}^{\text{CapMem}}$ for three exemplary neurons once with leak division enabled and once with leak division and multiplication disabled. One can see, that for some neurons the dependency on the $I_{\text{bias\_leak}}^{\text{CapMem}}$ is stronger than for others and that the direction of the dependency differs from neuron to neuron. Further, if one were to neglect this dependency, it could result in a difference of approximately 100 mV.

### 3.2.3 Membrane Time Constant

For the measurements of the membrane time constant as a function of $V_{\text{leak}}^{\text{CapMem}}$ and $I_{\text{bias\_leak}}^{\text{CapMem}}$, the measurement method which applies an offset current to the neuron and then fits an exponential to the decaying membrane potential is used (cf. section 2.4.2). Slight changes were made to this method: the maximum value of the fit-result is increased from 100 µs to 1000 µs. Furthermore, the sampling time, i.e. the duration that the membrane potential is recorded, is changed (cf. section 3.2.3) and the method is changed such that only one neuron is measured at a time.

The offset method does not support the measurement of time constants below approximately 3 µs and consequently does not support leak multiplication. The method from calix that measures the time constant after reset would be able to measure the smaller membrane time constants, but can not be used for this two-dimensional measurement because the

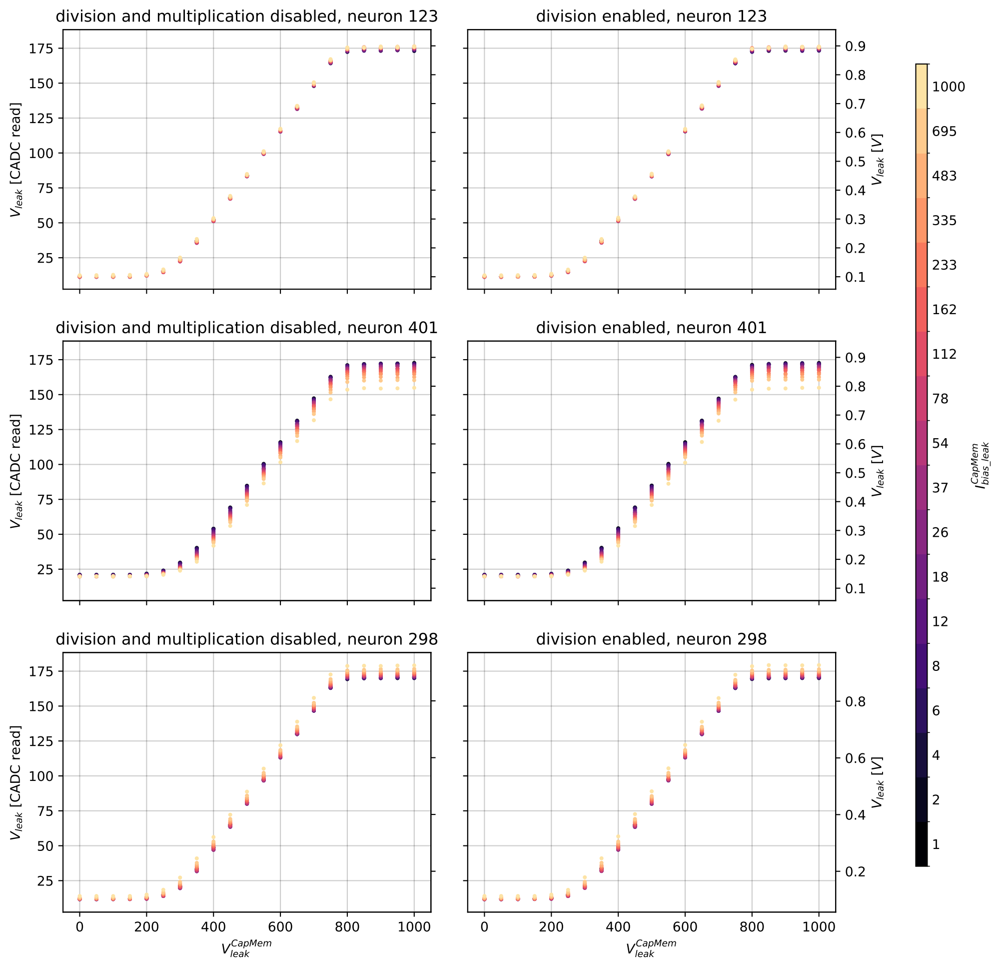



Figure 3.3: $V_{\text{leak}}$ as a function of $V_{\text{leak}}^{\text{CapMem}}$ for different settings of $I_{\text{bias\_leak}}^{\text{CapMem}}$. These three neurons are chosen because they show that the dependency of $V_{\text{leak}}$ on $I_{\text{bias\_leak}}^{\text{CapMem}}$ varies between the neurons. For some neurons the spread between different $I_{\text{bias\_leak}}^{\text{CapMem}}$ of the measured $V_{\text{leak}}$ for a given $V_{\text{leak}}^{\text{CapMem}}$ is wider than for others. Additionally, the direction of the dependency varies from neuron to neuron. The leak division setting has no visible impact on the behaviour of the leak potential. The measurements of neuron 401 and 298 show, that the dependency on the $I_{\text{bias\_leak}}^{\text{CapMem}}$ can not be neglected as it can cause a change of $V_{\text{leak}}$ of up to 100 mV. The variance of the CADC read of the leak potential from 10 measurements are so small that the error bars are not visible. The measurements are conducted on chip W61F0.

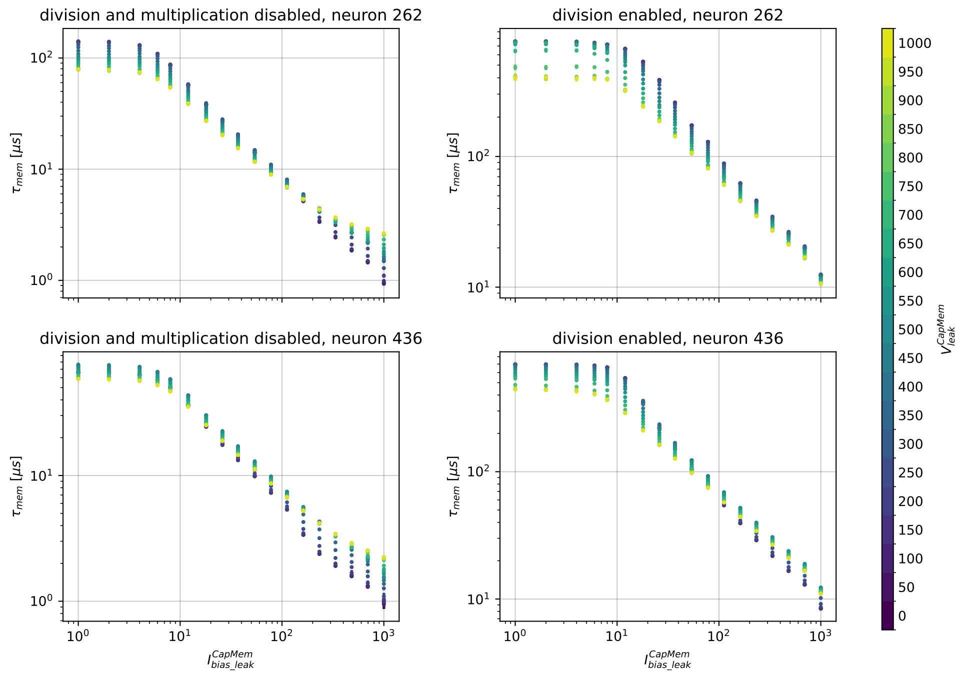


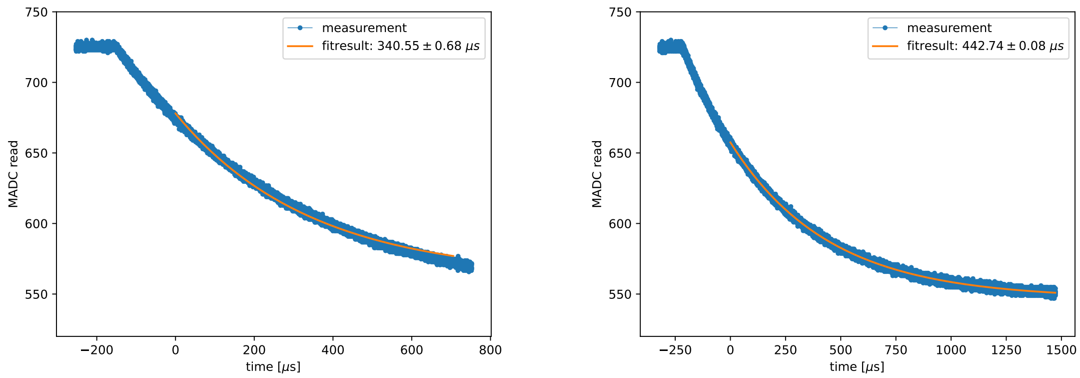

Figure 3.4: Membrane time constant as a function of the $I_{\text{bias\_leak}}^{\text{CapMem}}$ for different $V_{\text{leak}}^{\text{CapMem}}$ for two exemplary neurons and with division enabled or disabled. The time constant is measured using a method from calix which records the decaying membrane potential after a step current was applied to the neuron and then fits an exponential to the recorded data (cf. section 2.4.2). The plot clearly shows a dependency of the membrane time constant on the $V_{\text{leak}}^{\text{CapMem}}$. The differently scaled y-axes show the effect of the leak division mode. The measurements are taken on chip W61F0.

method alters the $V_{\text{leak}}^{\text{CapMem}}$ during measurement. Therefore, the measurements are only carried out with leak division enabled and with leak division/ multiplication disabled. Other measurement methods for the membrane time constant would need to be developed so that the measurements can be carried out with multiplication enabled.

Figure 3.4 shows the membrane time constant as a function of the $I_{\text{bias\_leak}}^{\text{CapMem}}$ for different settings of $V_{\text{leak}}^{\text{CapMem}}$. The plot clearly shows that $\tau_{\text{mem}}$ does not only depend on $I_{\text{bias\_leak}}^{\text{CapMem}}$, but also on $V_{\text{leak}}^{\text{CapMem}}$, showcasing the necessity of a two-dimensional transformation.

### Problems of the measurement method

A problem occurs for long time constants. At first, every time constant was measured using a sampling time of 1000 µs. However, for long time constants (approximately above 200 µs) the fit result differed significantly when varying the sampling time. Figure 3.5 shows a worst case of how the result changed when varying the sampling time. In this case the fit result varied by over 100 µs. Therefore, for $I_{\text{bias\_leak}}^{\text{CapMem}}$ lower than 54 and with division enabled, the sampling time was increased to 2200 µs.

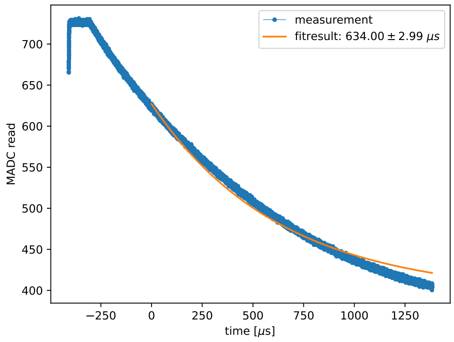



(a) Sampling time of 1000 µs

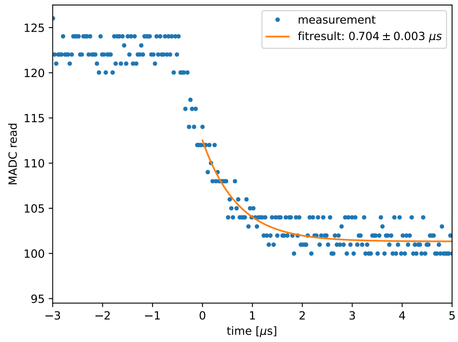
(b) Sampling time of 2200 µs

Figure 3.5: Exponential fit to the decaying membrane potential that the measurement method from calix uses to measure $\tau_{\text{mem}}$ for different sampling times. The difference in the measured $\tau_{\text{mem}}$ is more than 100 µs. Therefore, $\tau_{\text{mem}}$ is measured with a sampling time of 2200 µs for higher $\tau_{\text{mem}}$. Measured on chip W61F0, neuron 436 with division enabled and $I_{\text{bias\_leak}}^{\text{CapMem}}$=6, $V_{\text{leak}}^{\text{CapMem}}$=750.

Figure 3.6a shows that for even larger $\tau_{\text{mem}}$ one should use an even higher sampling time, but this is not possible with the mode of the MADC used by calix, where the number of samples is limited. One could for example reduce the sampling frequency in order to be able to record for longer durations.




(a) Long $\tau_{\text{mem}}$: The sampling time is 2200 µs. The fit could be improved with a longer sampling time, which would require changing the sampling frequency of the MADC. Measured on chip W61F0, neuron 436 with leak division enabled and $I_{\text{bias\_leak}}^{\text{CapMem}}$=1 and $V_{\text{leak}}^{\text{CapMem}}$=500.




(b) Short $\tau_{\text{mem}}$: The offset current can not stimulate the membrane enough for a reliable and precise fit result. Measured on chip W61F0 neuron 222, with leak division and multiplication disabled and $V_{\text{leak}}^{\text{CapMem}}$=100 and $I_{\text{bias\_leak}}^{\text{CapMem}}$=1000.

Figure 3.6: Exponential fit to determine $\tau_{\text{mem}}$ for a large and a small membrane time constant using the offset method from calix. The method does not produce reliable results for very long and very short $\tau_{\text{mem}}$.

At the lower end of the $\tau_{\text{mem}}$ range, meaning approximately below 3 µs, the measurement method also shows some issues. Due to the high leak conductance, the offset current is not able to stimulate the membrane potential significantly (cf. fig. 3.6b).

### 3.2.4 Time for Measuring

Table 3.1 gives an overview over how long each part of the measurement with the current implementation takes. For each neuron, a grid of $20 \cdot 18 \cdot 2 = 720$ hardware setting points is swept (20 $V_{\text{leak}}^{\text{CapMem}}$ settings, 18 $I_{\text{bias\_leak}}^{\text{CapMem}}$ settings, division enabled or disabled). For each hardware setting point, the settings are firstly configured which takes around 90 ms for setting $I_{\text{bias\_leak}}^{\text{CapMem}}$, $V_{\text{leak}}^{\text{CapMem}}$ and the leak division setting. Then, the model parameter is measured ten times, which takes around 400 ms for $\tau_{\text{mem}}$ and around 160 ms for $V_{\text{leak}}$. Before starting a measurement, a `prepare_meas()` function is called, which takes around 250 ms for $\tau_{\text{mem}}$ and is not needed for $V_{\text{leak}}$. In total, the sweep of one neuron takes around 6 min for $\tau_{\text{mem}}$ and around 3 min for $V_{\text{leak}}$, which results in a time of 51 h for $\tau_{\text{mem}}$ and 26 h for $V_{\text{leak}}$ for all 512 neurons.

There are multiple possibilities to optimize the measurement in order to make it faster. The measurements for $V_{\text{leak}}$ could for example be parallelized by measuring all four quadrants in parallel, or even more neurons in parallel by not choosing the same CapMem values for the sweeps. For $\tau_{\text{mem}}$, since it is measured with the MADC, only two neurons can be

| | time for $\tau_{\mathrm{mem}}$ | time for $V_{\mathrm{leak}}$ |
|---|---|---|
| prepare meas | 250 ms | - |
| measure result 10x | 410 ms | 160 ms |
| setting CapMem 2x | 90 ms | 90 ms |
| one neuron (720 points) | ~6 min | ~3 min |
| 512 neurons | 51 h | 26 h |

Table 3.1: Time for measurements of the membrane time constants by recording the membrane potential using the MADC and the leak potential using the CADC.

measured in parallel. However, one could change the CapMem values for multiple neurons simultaneously and then measure, which would minimize the overall time needed for setting the CapMem. Again, one has to take the CapMem crosstalk problematic into account, i.e. not setting the CapMem cells to the same values.

## 3.3 Fitting of the Transformation Model

The recorded data will now be used to create the transformation by fitting a function to it. Since the transformation model will be two-dimensional, one could consider fitting a function $f : \mathbb{R}^2 \to \mathbb{R}^2$ where the domain is a point in the $\tau_{\mathrm{mem}}$-$V_{\mathrm{leak}}$-plane and the range a point in the $V_{\mathrm{leak}}^{\mathrm{CapMem}}$-$I_{\mathrm{bias\_leak}}^{\mathrm{CapMem}}$-plane. However, this would require knowledge of the form of the dependency of the model parameters on each other's hardware parameter. Figure 3.7 shows the model parameters as a function of each other's respective hardware parameter. The plots do not indicate an obvious function to fit to this dependency. Yet, one can see, that the dependency of the model parameters on their respective hardware parameter is stronger than on each other's hardware parameter.

This leads to the conclusion that, since it is challenging to find a function $f : \mathbb{R}^2 \to \mathbb{R}^2$ to fit to the data, but we know the form of the dependency of the model parameters on their respective hardware parameter, we instead choose to fit a set of curves $f_c : \mathbb{R} \to \mathbb{R}$. This means that for each $I_{\mathrm{bias\_leak}}^{\mathrm{CapMem}}$ a function is fitted to the data $V_{\mathrm{leak}}$ as a function of the $V_{\mathrm{leak}}^{\mathrm{CapMem}}$. Likewise, for each $V_{\mathrm{leak}}^{\mathrm{CapMem}}$ a function is fitted to the data $\tau_{\mathrm{mem}}$ as a function of $I_{\mathrm{bias\_leak}}^{\mathrm{CapMem}}$.

An idea for a function $f : \mathbb{R}^2 \to \mathbb{R}^2$ could be a two-dimensional polynomial, but this approach is not pursued in this thesis.

### 3.3.1 Leak Potential

As mentioned we expect $V_{\mathrm{leak}}$ to exhibit a linear dependency on $V_{\mathrm{leak}}^{\mathrm{CapMem}}$, but only in a certain range. Therefore, before fitting a linear function to the data, the linear part of the data needs to be selected. This can not be done by hand for all neurons and all $I_{\mathrm{bias\_leak}}^{\mathrm{CapMem}}$.

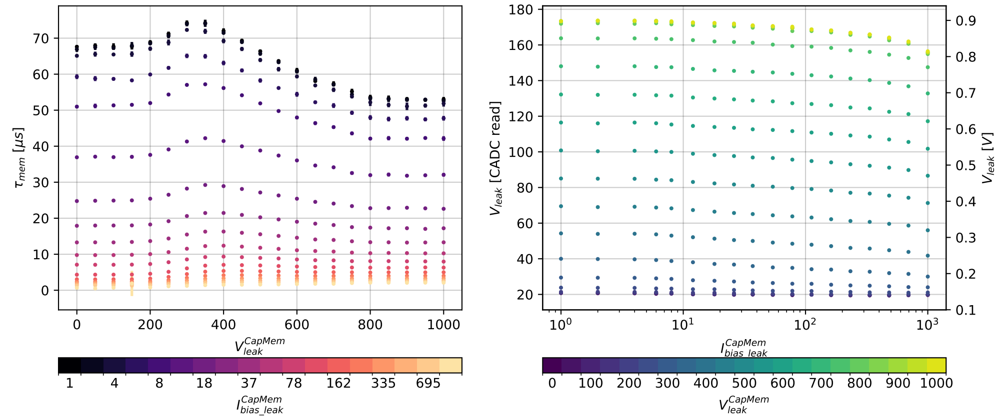



Figure 3.7: Left: Membrane time constant as a function of $V_{\text{leak}}^{\text{CapMem}}$ for different settings of $I_{\text{bias\_leak}}^{\text{CapMem}}$. Right: leak potential as a function of $I_{\text{bias\_leak}}^{\text{CapMem}}$ for different settings of $V_{\text{leak}}^{\text{CapMem}}$. Neuron 401 on chip W61F0, leak division disabled.

Thus, this selection needs to be automated. The chosen approach is to calculate a numeric second derivative from the data points and then cutting at the minimum and maximum of the second derivative such that the points of the minimum and maximum are excluded.

After selecting the range, a linear function $f(x) = ax + c$ is fitted to the data in the direction $V_{\text{leak}}$ as a function of $V_{\text{leak}}^{\text{CapMem}}$, using a weighted least squares fit with weights $w_i = \frac{1}{\sigma_i^2}$ where $\sigma_i$ is the standard deviation of the ten CADC reads of $V_{\text{leak}}$. The fit returns the fit parameter for the offset $a$ and the slope $b$.

For the lookup-based transformation model the user needs a function that converts $V_{\text{leak}}$ into a $V_{\text{leak}}^{\text{CapMem}}$ value. Thus, the function $f$ is inverted resulting in a linear function $f^{-1}(x) = a'x + b'$ with $a' = \frac{1}{a}$ and $b' = -\frac{b}{a}$. The function $f$ is evaluated at the outermost hardware parameters of the points that were used for the fit in order to get the model parameter range.

Figure 3.8 shows the fitted functions $f^{-1}$ for one neuron for the different $I_{\text{bias\_leak}}^{\text{CapMem}}$ settings and the residuals of the fit. Figure 3.9 shows the residuals of the function $f^{-1}$ (measured $V_{\text{leak}}^{\text{CapMem}}$ minus predicted $V_{\text{leak}}^{\text{CapMem}}$) as a function of the predicted $V_{\text{leak}}^{\text{CapMem}}$ for all 512 neurons on one chip and for different $I_{\text{bias\_leak}}^{\text{CapMem}}$ settings.

Since the residuals were not spread uniformly around zero, a polynomial fit was tried out to achieve better fit results. For polynomials of arbitrary degree, the approach of fitting in the direction "model parameter as a function of the hardware parameters" and then simply inverting it into the form specified by the transformation interface (cf. section 3.4) does not work analytically anymore. Therefore, the fit is directly performed in the direction "hardware parameter as a function of the model parameter" using scipy.odr [17] for an orthogonal distance regression due to the measurement errors being on the x-axis. A fourth-degree

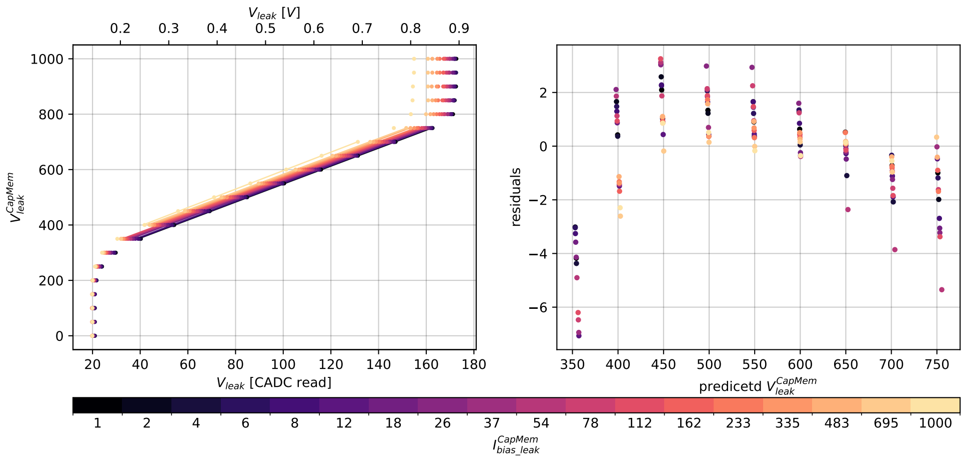



Figure 3.8: Linear fits for $V_{\text{leak}}^{\text{CapMem}}$ as a function of $V_{\text{leak}}$ for different $I_{\text{bias\_leak}}^{\text{CapMem}}$ for neuron 401 on chip W61F0. Residuals show that the dependency is not fully linear.

polynomial produced the best results among all polynomials up to degree four.

Figure 3.9 shows the residuals with a polynomial of degree four. The residuals are on average smaller and spread more evenly around zero. Therefore, the result from the polynomial fit will be used for later evaluation.

### 3.3.2 Membrane Time Constant

For the membrane time constant the function eq. (3.1) is fitted to the data $\tau_{\text{mem}}$ as a function of $I_{\text{bias\_leak}}^{\text{CapMem}}$ and inverted analytically afterwards. This function is chosen because the equations describing the leak OTA state that its transconductance is linearly dependent on the leak bias current for small bias currents and is proportional to the square root of the leak bias current for higher bias currents [3]. An interpolation between these two cases and the fact that $\tau_{\text{mem}}$ is inversely proportional to $g_{\text{leak}}$ for constant $C_{\text{mem}}$, yields eq. (3.1).

$$f(x) = b \cdot (x - c)^a \tag{3.1}$$

with $a < 0$, $b > 0$ and $x - c > 0 \; \forall x$.

The second derivative of the function is positive for all valid $x$. This property can be used to select a part of the data for the fitting. The numeric second derivative of $\tau_{\text{mem}}$ as a function of $I_{\text{bias\_leak}}^{\text{CapMem}}$ is calculated in the same way as for the leak potential using np.gradient. Then, the data is cut at the first negative second derivative going from high to low $I_{\text{bias\_leak}}^{\text{CapMem}}$ values, while ensuring that at least three data points are left for the fit.

Then the function eq. (3.1) is fitted to the selected part of the data using scipy.curvefit() [16] with the standard deviation of the membrane time constant measurements as y-errors.

For the user of the parameter transformational model, the function is inverted again, to

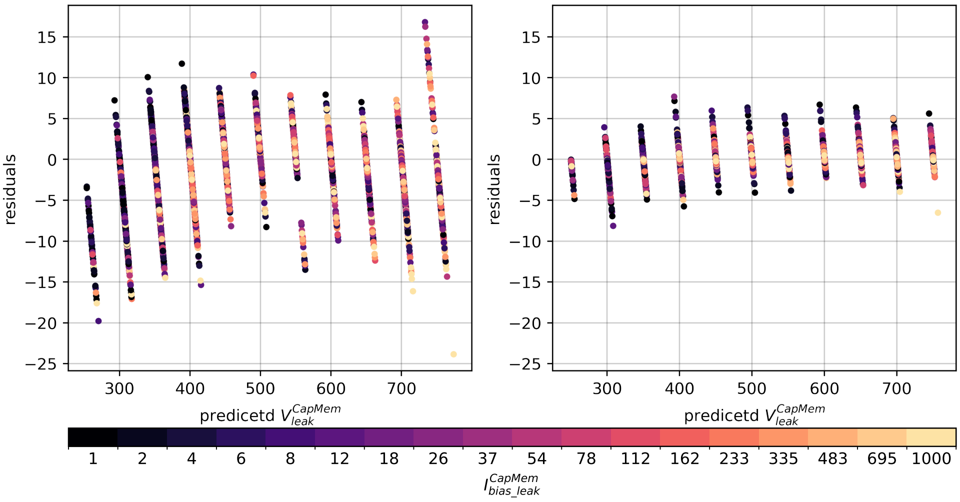



Figure 3.9: Residuals of the linear fit (left) and polynomial (right) for all neurons and for different settings of $I^{\mathrm{CapMem}}_{\mathrm{bias\_leak}}$. The residuals of the linear fit show systematic deviation from zero. Thus, a polynomial fit was tested. The right plot shows the residuals with a polynomial of fourth degree. The residuals are smaller and spread more evenly around zero. The polynomial seems to describe the hardware's behaviour better than the linear model. The residuals do not depend on the $I^{\mathrm{CapMem}}_{\mathrm{bias\_leak}}$ setting. Measured on chip W61F0.

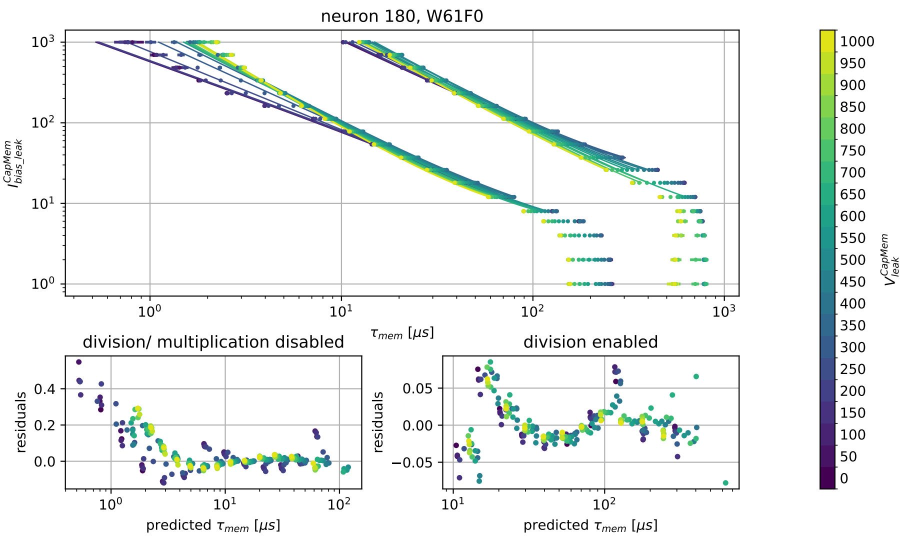



Figure 3.10: Fit results of the function eq. (3.2) to the data $I^{\mathrm{CapMem}}_{\mathrm{bias\_leak}}$ as a function of $V_{\mathrm{leak}}$ for division enabled and division disabled for different $V^{\mathrm{CapMem}}_{\mathrm{leak}}$ settings for one neuron. The second row shows the relative residuals of the function (3.1). Measured on chip W61F0.

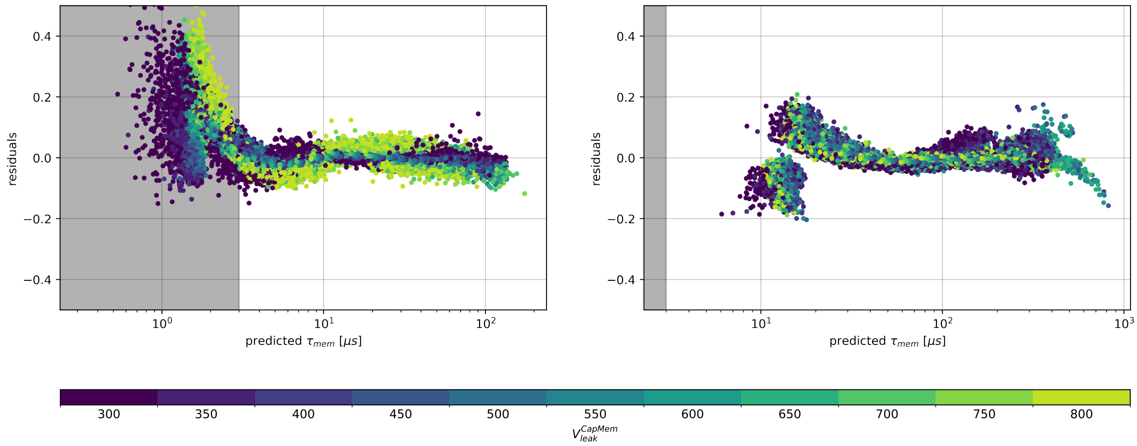



Figure 3.11: Relative residuals of the fit of the function (3.2) to the data $I_{\text{bias\_leak}}^{\text{CapMem}}$ as a function of $\tau_{\text{mem}}$ for all 512 neurons on chip W61F0 once with division enabled (right) and disabled (left). The $V_{\text{leak}}^{\text{CapMem}}$ values were reduced to the values from 300 to 800 since this is the range of the $V_{\text{leak}}$ fits (cf. fig. 3.9). The grey block marks the time constants below 3 µs since the used measurement method does not support these small time constants, which might explain the increase of the residuals towards these small $\tau_{\text{mem}}$ for the left plot. The colours indicate that the residuals depend on the $V_{\text{leak}}^{\text{CapMem}}$. Overall, the residuals are not spread uniformly, which means that a different function would fit the data better. Excluding the range below 3 µs, the residuals stay below 20 %.

get the hardware parameter as a function of the model parameter. The inverted function is:

$$f^{-1}(x) = d \cdot x^e + g \tag{3.2}$$

with $g = c$, $d = b^{-1/a}$ and $e = 1/a$.

Figure 3.10 shows the resulting fits for one exemplary neurons for leak division enabled as well as for leak multiplication and division disabled and for different values of the CapMem value of the leak potential. It also shows the residuals of the fit with division mode and without division mode.

Figure 3.11 shows the relative residuals for all neurons and the different settings of $V_{\text{leak}}^{\text{CapMem}}$, but only from 300 to 800 because this is the maximum range that the fits for $V_{\text{leak}}$ yield (cf. fig. 3.9). The fits converge for all neurons, but the residuals show systematic deviations, meaning that for better results one would have to find a different function to fit to the data.

## 3.4 Transformation Interface

The results from fitting are (for each neuron) two sets of curves: $I_{\text{bias\_leak}}^{\text{CapMem}}$ as a function of $\tau_{\text{mem}}$ for different $V_{\text{leak}}^{\text{CapMem}}$ and $V_{\text{leak}}^{\text{CapMem}}$ as a function of $V_{\text{leak}}$ for different $I_{\text{bias\_leak}}^{\text{CapMem}}$. Each curve additionally specifies a model parameter range, i.e. a range for which the transformation is valid.

The task of the transformation interface is to return a pair of CapMem values ($V_{\text{leak}}^{\text{CapMem}}$, $I_{\text{bias\_leak}}^{\text{CapMem}}$) for a given pair of model parameters ($V_{\text{leak}}$, $\tau_{\text{mem}}$) using the two sets of curves.

Since we did not use a two-dimensional fit, the idea is to interpolate between the curves. Figure 3.12 shows the two sets of curves in three dimensions, and the surfaces represent the interpolation between the curves. The interpolations now provide both model parameters as a function of the pair of hardware parameters. In order to get a pair of hardware parameters from the pair of model parameters, one can imagine drawing a contour line at the target model parameters for each of the model parameters. This results in two lines in the plane of the two hardware parameters. An intersection of the contour lines forms the transformation result.

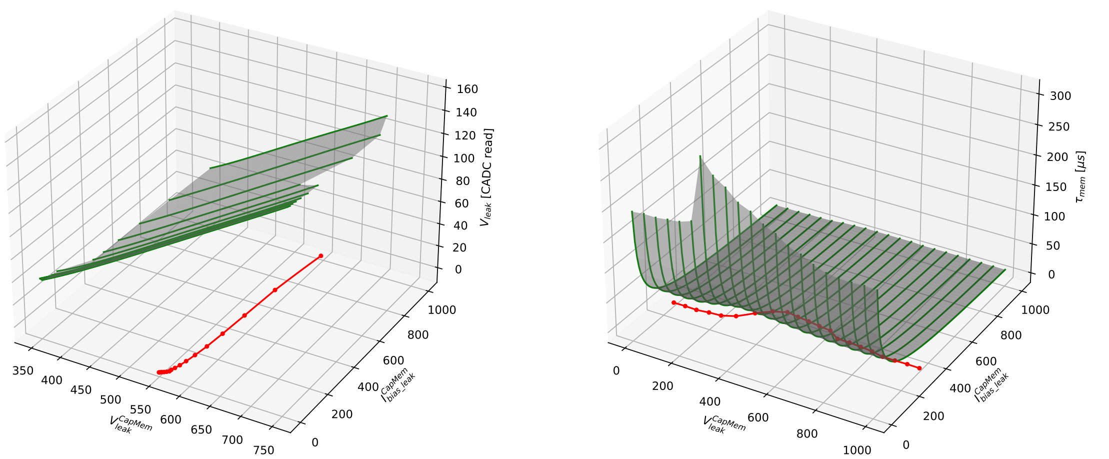



Figure 3.12: Visualization of the two sets of curves (green) resulting from the fits and the interpolation (grey surface) between the set of curves. A pair of hardware settings corresponding to a pair of model parameters is retrieved by finding the intersection of the two contour lines (red) at the respective model parameter. Red lines show the contour lines for $\tau_{\text{mem}}$=25 µs and $V_{\text{leak}}$=100 CADC read.

In practice, the transformation result is computed in the following way. Given a pair ($V_{\text{leak}}$, $\tau_{\text{mem}}$), for each curve $V_{\text{leak}}$ as a function of $V_{\text{leak}}^{\text{CapMem}}$, the hardware parameter $V_{\text{leak}}^{\text{CapMem}}$ is calculated for the given $V_{\text{leak}}$. Since each of these curves is assigned to a $I_{\text{bias\_leak}}^{\text{CapMem}}$, the result is a set of point in the $I_{\text{bias\_leak}}^{\text{CapMem}}$-$V_{\text{leak}}^{\text{CapMem}}$-plane. Likewise, the $I_{\text{bias\_leak}}^{\text{CapMem}}$ is computed for each curve $\tau_{\text{mem}}$ as a function of $I_{\text{bias\_leak}}^{\text{CapMem}}$ at the given $\tau_{\text{mem}}$, also resulting in a set of point in the $I_{\text{bias\_leak}}^{\text{CapMem}}$-$V_{\text{leak}}^{\text{CapMem}}$-plane. If the given model parameter is not in the model parameter range of a curve, this point is not added to the set of points. Connecting the points of each set of points is equivalent to the interpolation.

Now, the last step is to find an intersection of these lines to get the hardware parameter pair. The lines are a set of connected line segments defined by the points. The approach to find the intersection is to test for each segment of line 1 whether it intersects with any segment of line 2. There is a well-defined way of finding out whether an intersection of two non-parallel line segments in two dimensions exists and if so, how to calculate the intersection [2]. Given the coordinates of the endpoints of segment 1 $(x_1, y_1)$, $(x_2, y_2)$ and of segment 2 $(x_3, y_3)$, $(x_4, y_4)$, the following equation needs to be solved:

$$\begin{pmatrix} x_2 - x_1 & -(x_4 - x_3) \\ y_2 - y_1 & -(y_4 - y_3) \end{pmatrix} \begin{pmatrix} s \\ t \end{pmatrix} = \begin{pmatrix} x_2 - x_1 \\ y_3 - y_1 \end{pmatrix} \tag{3.3}$$

where $s$ and $t$ parametrize the lines. If the solutions $s_0$ and $t_0$ are $0 \leq s_0, t_0 \leq 1$, an intersection of the two lines exists:

$$\begin{pmatrix} x^* \\ y^* \end{pmatrix} = \begin{pmatrix} x_1 + t_0(x_2 - x_1) \\ y_1 + t_0(y_2 - y_1) \end{pmatrix} \tag{3.4}$$

We assume that there is only one intersection because the two contour lines are nearly orthogonal to each other (cf. fig. 3.12). Therefore, the linear search algorithm stops after finding an intersection. The intersection point forms the transformation result after it is rounded to an integer, since the digital values are integers.

## 3.5 Evaluation of the Calibration using the Transformation Model

The interface provides a transformation from an arbitrary pair of model parameters ($V_{\text{leak}}$, $\tau_{\text{mem}}$) to a pair of hardware settings ($V_{\text{leak}}^{\text{CapMem}}$, $I_{\text{bias\_leak}}^{\text{CapMem}}$) for all 512 neurons. The aim of this section is to evaluate the calibration using the parameter transformation model for these two parameters. This means, calculating the hardware settings for a given pair of target model parameters for all neurons, then applying them to the chip and measuring the effect of the hardware settings on the model parameters for all neurons.

As described, the curves that define the transformation also specify a model parameter range. If a targeted model parameter is outside this range, or the contour lines do not intersect, the transformation does not find a set of hardware parameters, meaning the transformation does not exist for the given pair of $\tau_{\text{mem}}$ and $V_{\text{leak}}$. Figure 3.13 shows the number of neurons for which the transformation model can find a set of hardware parameters as a function of the model parameters. The evaluation is then carried out for all model parameter pairs in fig. 3.13 for which at least 502 neurons have a transformation, which is 98 % of all neurons.

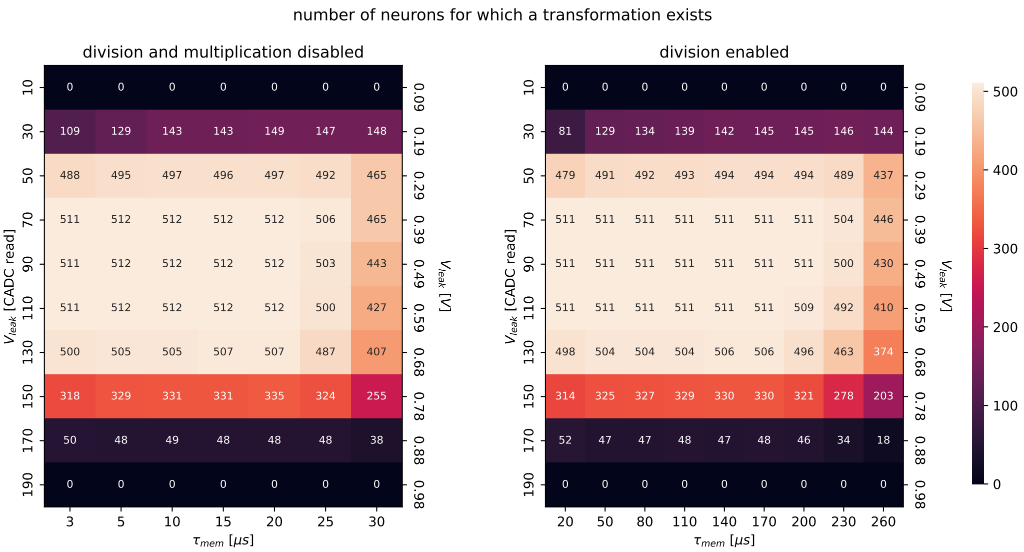



Figure 3.13: Number of neurons for which a transformation result exists as a function of the target $\tau_{\text{mem}}$ and $V_{\text{leak}}$. The ranges for $\tau_{\text{mem}}$ with division enabled and disabled overlap. Transformations for chip W61F0.

For each of these model parameter pairs, the hardware parameters are computed using the transformation and applied to the chip. Then, the two model parameters are measured again. The neurons for which no transformation exists are excluded in the evaluation. The membrane time constant is measured with a sampling time of 2000 µs, which should be long enough since the largest membrane time constant is around 230 µs. The results are shown in Figure 3.14 with division disabled and Figure 3.15 with division enabled.

The plots show the distributions as well as the deviations of the mean from the target of the measured model parameters over all neurons with a transformation (at least 502 neurons) on chip W61F0 after calibration using the transformation for different target model parameter pairs. The target membrane time constant increases from left to right, and the target leak potential increases from top to bottom. The middle of the cross in each plot marks the point of the target model parameters. The range of the $\tau_{\text{mem}}$-axis is scaled with the target $\tau_{\text{mem}}$, while the range of the $V_{\text{leak}}$-axis is kept constant, because the characteristics of the potential should be independent of its absolute value. The mean and standard deviations of the distributions are noted in each plot.

The first obvious observation is that most of the distributions are not centred, i.e. they exhibit systematic deviations from the target, which is demonstrated by the deviations of the mean from the target.

The membrane time constant after calibration with division disabled is larger than the target for a target of 3 µs. Then, for a target of 5 µs, the measured $\tau_{\text{mem}}$ is smaller than the target. From a target of 10 µs to a target of 25 µs, the mean relative deviation from the target is positive and increases. But the deviation from the target does not exceed 4 %. With leak division enabled, $\tau_{\text{mem}}$ exhibits a similar behaviour. The relative deviation from the target is at its maximum for a target of $\tau_{\text{mem}}$ of 20 µs, then turns negative for a target of 50 µs and increases with increasing target-$\tau_{\text{mem}}$. The maximum deviation is 6.5 %. But the target of 20 µs can be excluded since it is covered by the settings with division disabled. Then, the deviation of $\tau_{\text{mem}}$ overall stays below 4 %.

The behaviour of the deviation from the target aligns with the behaviour of the residuals (cf. fig. 3.11). For the smallest $\tau_{\text{mem}}$ (3 µs or 20 µs), most of the residuals are positive, meaning that the measured $\tau_{\text{mem}}$ are larger than the model predicts, which means that the model predicts a $I_{\text{bias\_leak}}^{\text{CapMem}}$ value that is smaller than the measured value which would cause a larger time constant. This aligns with a positive deviation from the target. From the 3 µs to 5 µs for division disabled and from 20 µs to 50 µs with division enabled, the residuals fall below zero, which also aligns with the negative deviation from the target. From there, the increase of the residuals when going to larger $\tau_{\text{mem}}$ again aligns with the increase of the deviation from the target.

The mean of $V_{\text{leak}}$ is systematically below the target. However, the deviations of the mean of the distributions from the target are not significant when comparing them to the standard deviations of the distributions that are shown in section 3.5.1. The deviations are not larger than 2.3 mV for the both cases of division enabled and disabled.

### 3.5.1 Comparison to fixed-point Calibration

Figure 3.16 shows the distribution of $\tau_{\text{mem}}$ and $V_{\text{leak}}$ after calibrating to a typical operation point using the fixed-point calibration. Since the current calibration framework does not support two-dimensional transformation, it is important to mention that the membrane time constant was calibrated before the leak potential, which is probably the cause of the deviation of the mean of the membrane time constant from the target. This calibration result will now be used as a reference to compare the calibration result based on the transformation to.

The standard deviation for the calix calibration of $\tau_{\text{mem}}$ is around 1.5 %. For the transformation model, the standard deviations have a similar dimension, ranging from 0.6 % to 2.2 % for the different targets.

The standard deviation of $V_{\text{leak}}$ is 2.0 mV when using calix and ranges from 2.0 mV to 7 mV for different targets when using the transformation.

In general, the accuracy of the calibration is limited by the resolution of the CapMem. For $V_{\text{leak}}$, the order of magnitude of the resolution is $10^{-3}$ V per CapMem value (cf. fig. 3.8). The standard deviation of the calibration using the transformation is of the same order of

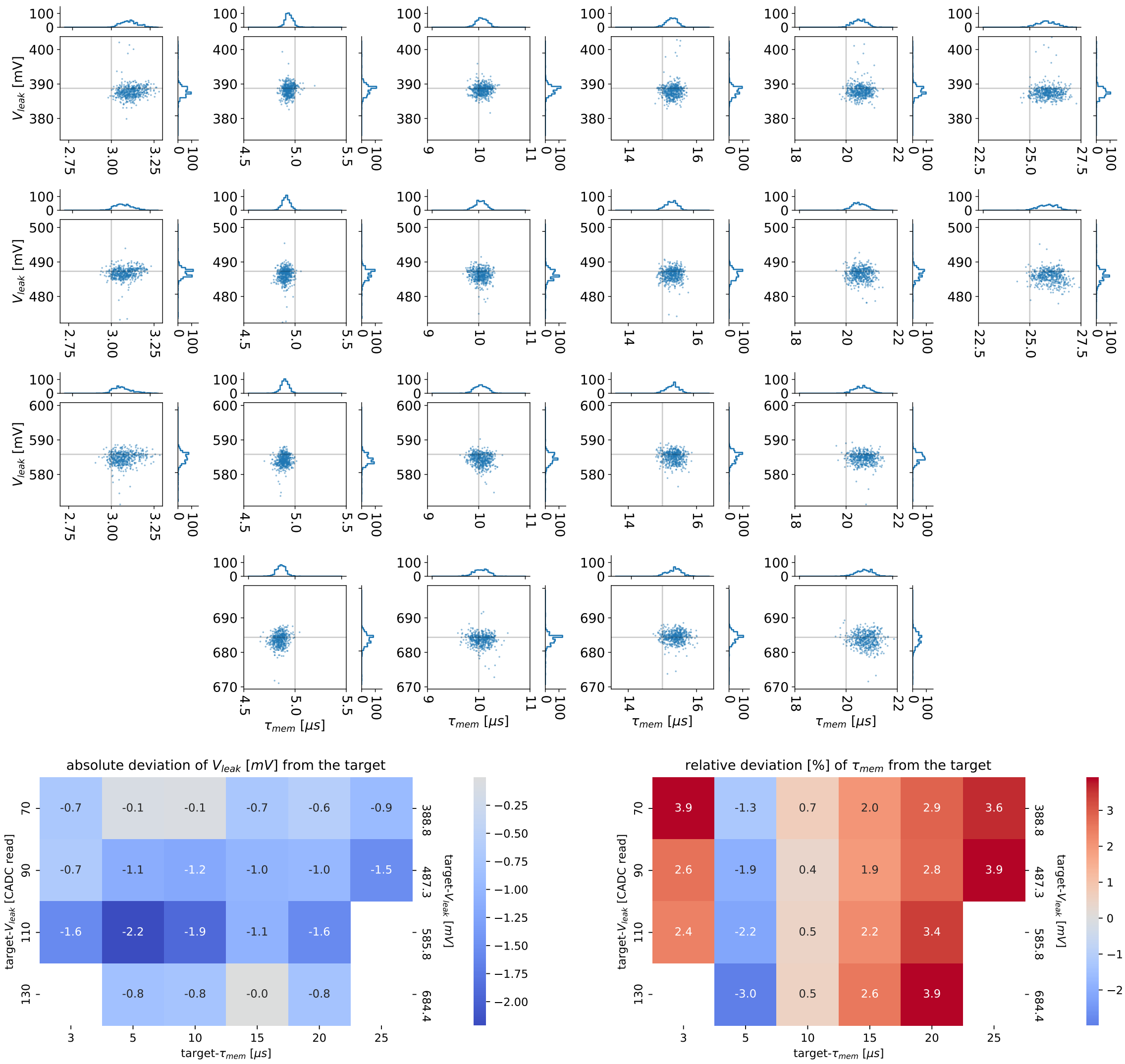



Figure 3.14: Distributions and mean deviations from the target of $\tau_{\text{mem}}$ and $V_{\text{leak}}$ over all neurons after calibration using the transformation for different calibration targets and with leak division disabled. The standard deviation of $\tau_{\text{mem}}$ does not exceed 4 %, but the distributions show systematic deviation from the calibration target, which can be attributed to the fit, since the trend of the deviations align with the residuals fig. 3.11. The mean of $V_{\text{leak}}$ is systematically below the target, but the deviations are not significant.

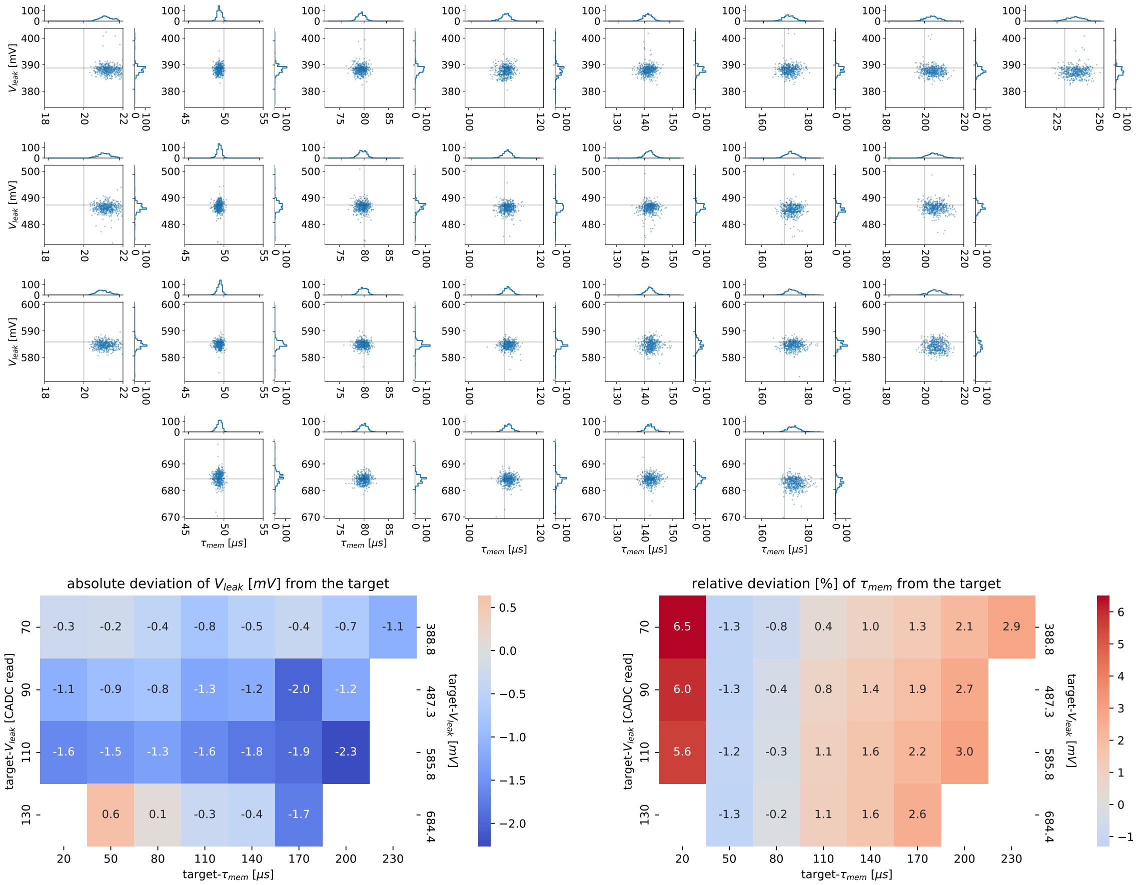



Figure 3.15: Distributions and mean deviations from the target of $\tau_{\text{mem}}$ and $V_{\text{leak}}$ over all neurons after calibration using the transformation for different calibration targets and with leak division enabled. The maximum standard deviation of $\tau_{\text{mem}}$ is 6.5 %. The deviation of the mean $\tau_{\text{mem}}$ from the target shows similar behaviour as in Figure 3.14 and can also be attributed to the fit. The mean of $V_{\text{leak}}$ is systematically below the target, but the mean deviations from the target are not significant.

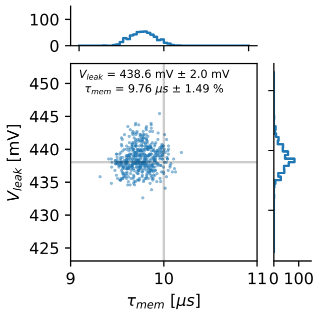
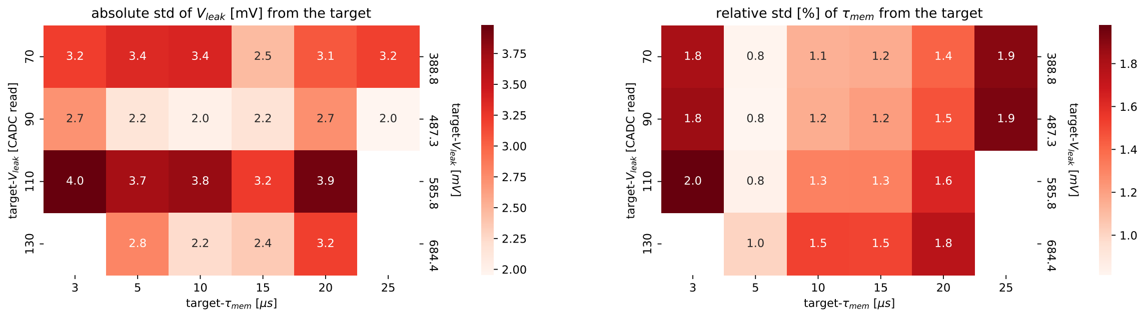
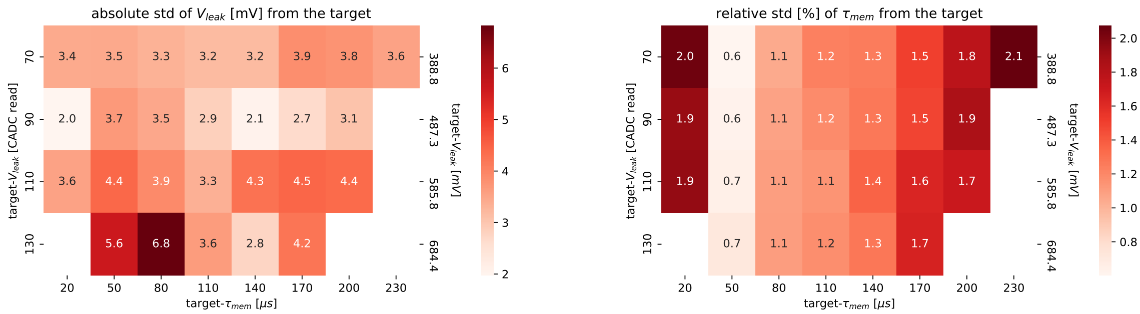



Figure 3.16: Distribution of $V_{\text{leak}}$ and $\tau_{\text{mem}}$ after calibrating to the default operation point of the nightly calibration using calix. This calibration serves as a reference.




(a) Leak division/ multiplication disabled




(b) Leak division enabled

Figure 3.17: Standard deviation of $V_{\text{leak}}$ and $\tau_{\text{mem}}$ over all neurons after calibration using the transformation for different calibration targets. The maximum standard deviation of $\tau_{\text{mem}}$ is 2.1 % and the 6.8 mV for $V_{\text{leak}}$.

magnitude as the resolution.

For $\tau_{\mathrm{mem}}$, the resolution depends on the $I_{\mathrm{bias\_leak}}^{\mathrm{CapMem}}$ value, since the dependency of $\tau_{\mathrm{mem}}$ on $I_{\mathrm{bias\_leak}}^{\mathrm{CapMem}}$ can not be approximated as linear. Calculating the slope of a fit from $I_{\mathrm{bias\_leak}}^{\mathrm{CapMem}}$ to $\tau_{\mathrm{mem}}$ shows that the order of magnitude of the resolution ranges from 1 % per CapMem value for small $I_{\mathrm{bias\_leak}}^{\mathrm{CapMem}}$ to $10^{-1}$ % per CapMem value for larger $I_{\mathrm{bias\_leak}}^{\mathrm{CapMem}}$. When excluding 3 µs for division disabled and 20 µs for division enabled, the standard deviations increase from around 0.7 % to around 2 %, which aligns with the trend of the resolution and matches the order of magnitude of the resolution.

To conclude, the standard deviation of $\tau_{\mathrm{mem}}$ after calibration using the transformation model is of similar size as the standard deviations of the parameters when using the fixed-point calibration, while the standard deviation of $V_{\mathrm{leak}}$ is slightly larger on average than the standard deviation of the reference calix calibration. Further, the standard deviation is of the same order of magnitude as the upper limit of the accuracy imposed by the resolution of the CapMem.

The time for calibration for $V_{\mathrm{leak}}$ and $\tau_{\mathrm{mem}}$ using the transformation is approximately 4 s, whereas the calix calibration takes approximately 70 s for the two parameter. The code for calculating the transformation for all 512 neurons is not optimized, which means that even shorter calibration times could be achieved.

# 4 Discussion

In this thesis, a transformation model for calibration of BSS-2 for the two model parameters, membrane time constant $\tau_{\text{mem}}$ and leak potential $V_{\text{leak}}$, of the LIF neuron was developed and evaluated, since this approach allows faster calibration than the current fixed-point calibration.

The first step was to measure the model parameters as a function of the hardware parameters. The measurements showed that the two model parameters exhibit dependencies not only on their respective hardware parameter but also on the hardware parameter of each other. Thus, $\tau_{\text{mem}}$ and $V_{\text{leak}}$ were both measured as a function of the hardware parameter that controls $\tau_{\text{mem}}$ and the hardware parameter that controls $V_{\text{leak}}$. Additionally, the measurements were carried out once with leak division enabled and once with leak division and multiplication disabled, which scales the leak conductance, allowing for a wider range of $\tau_{\text{mem}}$. The measurements were not conducted with multiplication enabled, since no suitable measurement method existed.

Some issues occurred when conducting the measurement for $\tau_{\text{mem}}$. They mainly concerned the measurement method, which did not provide accurate result for very long and very short $\tau_{\text{mem}}$. The measurements for all neurons on one chip took several days with the current implementation because all neurons are measured sequentially. However, this can be improved by measuring several neurons in parallel, which requires an approach on how to avoid configuring all neurons to the same hardware setting since this would lead to a shift of the analog parameters due to the CapMem crosstalk problematic. Besides, the measurement only has to be done once for generating the transformation and in turn provides a very fast calibration.

The second step, was to fit a model to the obtained data. Since it is challenging to find a function $f : \mathbb{R}^2 \to \mathbb{R}^2$ to fit to the data, and we had some knowledge about the form of the dependency of the model parameters on their respective hardware parameter, two sets of parametrizations of transformation functions $f_c : \mathbb{R} \to \mathbb{R}$ were fitted, i.e. each of the two hardware parameter was fitted as a function of its respective model parameter for different settings of the other hardware parameter. Before fitting, a part of the data was selected for the fit, this was automated with the help of a numeric second derivative of the measured data.

For the leak potential, we expected a linear functional dependency of $V_{\text{leak}}$ on its CapMem value. However, the residuals of the fit were not randomly scattered around zero. Thus, a polynomial fit was chosen whose residuals were spread more evenly around zero and were smaller. For $\tau_{\mathrm{mem}}$ the assumption was, that the data follows the functional dependency of (3.2). For parts of the parameter range the function fitted the data well, but the residuals were also not spread randomly around zero, leading to the conclusion that a different function could achieve better results.

With the fitted result, it was now possible to create the transformation. A transformation object for one neuron contains two sets of curves, where each curve supplies the transformation of a model parameter to its respective hardware parameter but for different hardware settings of the other hardware parameter. A curve is defined by its fit parameters and a model parameter range, and is assigned to a hardware setting of the other parameter. The transformation object can be serialized and saved in a database. For a given set of model parameters, the hardware parameter for each curve is computed, resulting in two sets of points in the hardware settings plane. The intersection of the lines, resulting from connecting the points of each line, forms the calibration result.

The last step was to use the transformation for calibration and evaluate the results. The transformation was computed for a grid of target model parameters using the transformation interface and then applied to the chip and the model parameters were measured again. There was a systematic deviation of the membrane time constant from the target which is caused by the fit and could probably be minimized by a different fit function. The maximum deviation of the mean over all neurons from the target was 3.9 %. The mean of $V_{\mathrm{leak}}$ after calibration showed no significant deviation from the target and did not exceed 2.5 mV.

In comparison to the calibration results using the existing fixed-point calibration for a typical operation point, the standard deviation over all neurons was found to be of similar size for $\tau_{\mathrm{mem}}$ and slightly larger for $V_{\mathrm{leak}}$. The maximum standard deviation of $V_{\mathrm{leak}}$ was 6.8 mV and for $\tau_{\mathrm{mem}}$ 2.1 %. The time for calibration of $V_{\mathrm{leak}}$ and $\tau_{\mathrm{mem}}$ using the transformation was 4 s with an unoptimized code for the calculation of the transformation, whereas the calibration using the current fixed-point calibration, takes approximately 70 s for the two parameters. This clearly shows the speed advantage of the calibration using the transformation.

To conclude, the transformation model was successfully implemented for one current-based and one voltage-based parameter that exhibited dependencies on each other. The model would be improved and would provide more accurate results if a different fit function for the membrane time constant was used. When comparing the calibration using the transformation to a calibration using the current calibration framework, the calibration using the transformation is slightly less accurate. However, there are two main advantages of the transformation over the current fixed-point calibration: the first one is the speed of the calibration using the transformation and the second is that it takes interdependencies between parameters into account.

# 5 Outlook

This thesis represents only the initial steps toward a lookup-based transformation model for calibration. Therefore, potential next steps are outlined here.

Further steps concerning the transformation of $\tau_{\mathrm{mem}}$ and $V_{\mathrm{leak}}$ could be improving the measurement method. For the membrane time constant, mainly short and long time constants could not be measured reliably. Thus, developing a measurement method for very small time constants which does not change the hardware setting of $V_{\mathrm{leak}}$ would enable the development of a transformation with leak multiplication enabled. For very long time constants, one could consider implementing a measurement method that allows for longer measurements.

Concerning the process of measuring the model parameter as a function of the hardware setting, it would be of benefit to minimize the measurement time. This especially becomes important when the number of parameters for which the transformation model is created increases, since we do not want the measurements for all parameters to take weeks. The measurement time can be minimized by measuring multiple neurons in parallel. A first step would be to measure all quadrants in parallel, as there is only CapMem crosstalk within each quadrant. Further parallelization would require that the CapMem values of all neurons would be at least two values apart from each other to avoid crosstalk.

The transformation interface of the two-dimensional transformation could be used for a pair of parameters that shows similar dependencies. Furthermore, the interface could be extended for the membrane time constants such that the transformation automatically decides whether leak division or multiplication should be enabled. This could be realized by a section-wise transformation.

As the transformation model in this thesis was only constructed and evaluated for one chip, it would be of interest to test this approach for more chips, even tough we do not expect major differences.

Lastly, the next obvious step is to expand the transformation to more parameters of the neuron circuit. For this, the developed software structure for measuring the model parameters as a function of hardware parameter can be used. It would just require implementations of the respective measurement method and the setting of the hardware parameter, as well as finding out which parameters exhibit interdependencies. An important parameter one could look at next is the synaptic input, since it might have an impact on the transformation of $\tau_{\mathrm{mem}}$ and $V_{\mathrm{leak}}$.

Ultimately, if a transformation model for all parameters would be developed, numerous experiments could benefit from shorter calibration times.

# Acronyms

**ADC** analog-to-digital converters

**AdEx** adaptive exponential integrate-and-fire

**ANN** artificial neural networks

**BSS-2** BrainScaleS-2

**CADC** columnar analog-to-digital converters

**CapMem** capacitive memory

**DAC** analog-to-digital converter

**FPGA** field programmable gate array

**HICANN** High Input Count Analog Neural Network

**LIF** leaky integrate-and-fire

**MADC** membrane analog-to-digital converter

**OTA** operational transconductance amplifier

**PPU** programmable plasticity unit

**SI** International System of Units

**SNN** spiking neural networks

# 6 References


[1] Laurence F. Abbott. "Lapicque's introduction of the integrate-and-fire model neuron (1907)". In: *Brain Research Bulletin* 50.5 (1999), pp. 303–304. ISSN: 0361-9230. DOI: https://doi.org/10.1016/S0361-9230(99)00161-6.

[2] Franklin Antonio. "Graphics Gems III". In: Morgan Kaufmann, 1992. Chap. IV.6 - Fatser Line Segment Intersection. ISBN: 978-0-12-409673-8. DOI: https://doi.org/10.1016/B978-0-08-050755-2.50045-2.

[3] Sebastian Billaudelle. "From transistors to learning systems. circuits and algorithms for brain-inspired computing". PhD thesis. Ruprecht-Karls-Universität Heidelberg, 2022.

[4] Sebastian Billaudelle, Johannes Weis, Philipp Dauer, and Johannes Schemmel. "An accurate and flexible analog emulation of AdEx neuron dynamics in silicon". In: *29th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. 2022, pp. 1–4. DOI: 10.1109/ICECS202256217.2022.9971058.

[5] Maxence Bouvier, Alexandre Valentian, Thomas Mesquida, François Rummens, Marina Reyboz, Elisa Vianello, and Edith Beigné. "Spiking Neural Networks Hardware Implementations and Challenges: a Survey". In: *CoRR* abs/2005.01467 (2020).

[6] Romain Brette and Wulfram Gerstner. "Adaptive Exponential Integrate-and-Fire Model as an Effective Description of Neuronal Activity". In: *Journal of Neurophysiology* 94.5 (2005), pp. 3637–3642. DOI: 10.1152/jn.00686.2005.

[7] Alex de Vries. "The growing energy footprint of artificial intelligence". In: *Joule* 7.10 (2023), pp. 2191–2194. ISSN: 2542-4351. DOI: https://doi.org/10.1016/j.joule.2023.09.004.

[8] Electronic Visions(s), Heidelberg University. *calix*. Calibration routines for HICANN-X.

[9] Ronja Hinterding. "Evaluation of the Calix Calibration and the CapMem Crosstalk". Internship Report. Universität Heidelberg, 2024.

[10] Matthias Hock. "Modern Semiconductor Technologies for Neuromorphic Hardware". PhD thesis. Ruprecht-Karls-Universität Heidelberg, 2014.

[11] Matthias Hock, Andreas Hartel, Johannes Schemmel, and Karlheinz Meier. "An Analog Dynamic Memory Array for Neuromorphic Hardware". In: *2013 European Conference on Circuit Theory and Design (ECCTD)* (2013).

[12] Aron Leibfried. "On-chip calibration and closed-loop experiments on analog neuromorphic hardware". Master thesis. Ruprecht-Karls-Universität Heidelberg, 2021.

[13] Wolfgang Maass. "Networks of spiking neurons: The third generation of neural network models". In: *Neural Networks* 10.9 (1997), pp. 1659–1671. ISSN: 0893-6080. DOI: https://doi.org/10.1016/S0893-6080(97)00011-7.

[14] Christian Pehle, Sebastian Billaudelle, Benjamin Cramer, Jakob Kaiser, Korbinian Schreiber, Yannik Stradmann, Johannes Weis, Aron Leibfried, Eric Müller, and Johannes Schemmel. "The BrainScaleS-2 Accelerated Neuromorphic System with Hybrid Plasticity". In: *Front. Neurosci.* 16 (2022). ISSN: 1662-453X. DOI: 10.3389/fnins.2022.795876.

[15] Hartmut Schmidt. "Large-Scale Experiments on Wafer-Scale Neuromorphic Hardware". PhD thesis. Ruprecht-Karls-Universität Heidelberg, 2024. DOI: 10.11588/heidok.00034446.

[16] scipy. *SciPy API references: curvefit.*

[17] scipy. *SciPy API references: Orthogonal distance regression.*

[18] Johannes Weis. "Inference with Artificial Neural Networks on Neuromorphic Hardware". Master's thesis. Universität Heidelberg, Sept. 2020.

[19] Timo Wunderlich, Akos F. Kungl, Eric Müller, Andreas Hartel, Yannik Stradmann, Syed Ahmed Aamir, Andreas Grübl, Arthur Heimbrecht, Korbinian Schreiber, David Stöckel, Christian Pehle, Sebastian Billaudelle, Gerd Kiene, Christian Mauch, Johannes Schemmel, Karlheinz Meier, and Mihai A. Petrovici. "Demonstrating Advantages of Neuromorphic Computation: A Pilot Study". In: *Frontiers in Neuroscience* 13 (2019), p. 260. ISSN: 1662-453X. DOI: 10.3389/fnins.2019.00260.

# Danksagung

Ich möchte mich bedanken bei

- Phillip und Jakob für ihre ständige Bereitschaft alle mein Fragen zu beantworten, mir bei jeglichen Problemen zu helfen und mich immer zu motivieren. Außerdem für das Korrekturlesen meiner Arbeit.
- bei Eric für zahlreiche hilfreiche Anregungen und ebenfalls das Korrekturlesen meiner Arbeit.
- bei Amani and Simon for being the best office colleagues you could wish for.
- bei der gesamten Electronic Visions Gruppe – insbesondere für die unterhaltsamen Darts-Partien nach dem Essen, die für eine willkommene Abwechslung gesorgt haben.
- bei meinen Freunden und meiner Rugbymannschaft, die immer für mich da waren. Vor allem bei Ejona, die ebenfalls immer für mich da war und mir in dieser Zeit sehr geholfen hat.
- bei meinen Eltern sowie Carli und Lotte, ohne deren Unterstützung diese Arbeit nicht möglich gewesen wäre.

# Erklärung

Ich versichere, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

The work carried out in this Bachelor thesis used systems, which received funding from the European Union's Horizon 2020 Framework Programme for Research and Innovation under the Specific Grant Agreements Nos. 720270, 785907 and 945539 (Human Brain Project, HBP) and Horizon Europe grant agreement No. 101147319 (EBRAINS 2.0).