Zentralinstitut für Engineering, Elektronik und
Analytik (ZEA) · Systeme der Elektronik (ZEA-2)

Characterization of Distortions in Charge Stability Diagrams and Their Simulation in Modeled Data

Sarah Fleitmann

Jül-4444

Mitglied der Helmholtz-Gemeinschaft

JÜLICH
Forschungszentrum

Zentralinstitut für Engineering, Elektronik und Analytik (ZEA) · Systeme der Elektronik (ZEA-2)

Characterization of Distortions in Charge Stability Diagrams and Their Simulation in Modeled Data

Sarah Fleitmann

Berichte des Forschungszentrums Jülich
Jül-4444 · ISSN 0944-2952
Zentralinstitut für Engineering,
Elektronik und Analytik (ZEA)
Systeme der Elektronik (ZEA-2)
Jül-4444

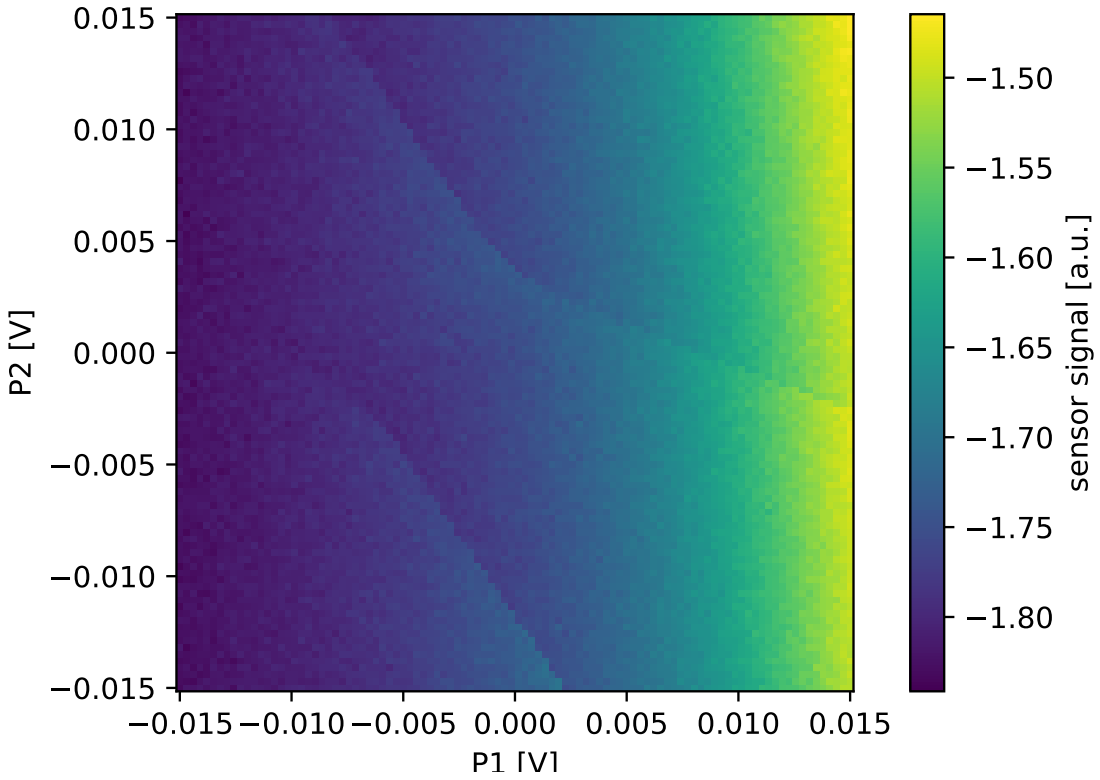
DE-A96 (Master FH Aachen, 2022)

Vollständig frei verfügbar über das Publikations-
portal des Forschungszentrums Jülich (JuSER)
unter www.fz-juelich.de/zb/openaccess

Forschungszentrum Jülich GmbH · 52425 Jülich
Zentralbibliothek, Verlag
Tel.: 02461 61-5220 · Fax: 02461 61-6103
zb-publikation@fz-juelich.de
www.fz-juelich.de/zb

ZEA-2

Electronic Systems



Abstract

Charge stability diagrams provide information about the electron occupation of double quantum dots. They are used during the tuning process of double quantum dots, necessary to enable their operation as quantum bits.
Simulated charge stability diagrams are required for testing and developing automated tuning algorithms. They are well suited for that because they can be generated fast and the ground truth occupation is known for a simulated data point. In contrast, the measuring of experimental datasets takes a long time and the ground truth is unknown. This thesis deals with the simulation of distortions in simulated charge stability diagrams.
For the simulation of the undisturbed occupation data, the capacitive model [1] and the Hubbard model [2] are presented. However, both models are not suited for the simulation of the honeycomb structures visible in available experimental charge stability diagrams. Another approach, currently developed by Fabian Hader from the ZEA-2, is used to overcome this problem.
To simulate realistic charge stability diagrams, the sensor response including distortions has to be added to the clean occupation data. Five types of distortions are identified: cross-coupling between sensor and double dot plunger gates, white noise, pink noise, random telegraph noise, and dot jumps. For a realistic simulation of these, procedures to determine parameter ranges from the experimental charge stability diagrams are developed and applied. Then, the generated simulated dataset is evaluated visually and by different metrics. To improve the quality of the simulated dataset, the initial parameters are adjusted, and the simulation model itself is refined. Finally, the optimized simulated dataset is evaluated with the same metrics, and the results are discussed.

Kurzfassung

Ladungsstabilitätsdiagramme liefern Informationen über die Elektronenbesetzung von Doppelquantenpunkten. Sie werden während des Kalibrierungsprozesses von Doppelquantenpunkten verwendet, der notwendig ist, um sie als Quantenbits zu nutzen.
Simulierte Ladungsstabilitätsdiagramme werden zum Testen und Entwickeln automatisierter Kalibrierungsalgorithmen benötigt. Sie sind gut dafür geeignet, da sie schnell erzeugt werden können und die Elektronenbesetzung für einen simulierten Datensatz bekannt ist. Im Gegensatz dazu dauert die Messung von experimentellen Datensätzen lange und die Elektronenbesetzung ist unbekannt. Diese Arbeit beschäftigt sich mit der Simulation von Störungen in simulierten Ladungsstabilitätsdiagrammen.
Für die Simulation der unverfälschten Besetzungsdaten werden das kapazitive Modell [1] und das Hubbard-Modell [2] vorgestellt. Beide Modelle eignen sich jedoch nicht für die Simulation der wabenförmigen Strukturen, die in den verfügbaren experimentellen Ladungsstabilitätsdiagrammen sichtbar sind. Um dieses Problem zu überwinden, wird ein anderer Ansatz verwendet, der derzeit von Fabian Hader vom ZEA-2 entwickelt wird.
Für die Simulation von realistische Ladungsstabilitätsdiagramme muss die Sensorantwort einschließlich Störungen zu den sauberen Besetzungsdaten hinzugefügt werden. Es werden fünf Arten von Störungen unterschieden: Kreuzkupplung zwischen dem Sensor und den Plunger Gates des Doppelquantenpunktes, weißes Rauschen, pinkes Rauschen, zufälliges Telegrafenrauschen und Dot Jumps. Um diese realistisch zu simulieren, werden Verfahren zur Bestimmung von Parameterbereichen aus den experimentellen Ladungsstabilitätsdiagrammen entwickelt und angewandt. Schließlich wird der generierte simulierte Datensatz visuell und anhand verschiedener Metriken bewertet. Um die Qualität des simulierten Datensatzes weiter zu verbessern, werden dann die Anfangsparameter angepasst und das Simulationsmodell verfeinert. Schließlich wird der optimierte simulierte Datensatz mit denselben Metriken bewertet und die Ergebnisse werden diskutiert.

Contents

List of Symbols and Abbreviations

CNN convolutional neural network

CSD charge stability diagram

DC direct current

DQD double quantum dot

FFNN feed-forward neural network

PCA principle component analysis

PCB printed circuit board

PSD power spectral density

qubit quantum bit

RF radio frequency

RTN random telegraph noise

SD sensing dot

ZEA Central Institute of Engineering, Electronics, and Analytics

2DEG two-dimensional electron gas

1 Introduction

This chapter provides insight into the topic of this thesis. First, the institute where the work has been carried out is presented. Then, a general description of quantum computing follows, and finally, the motivation for this work is given.

1.1 Central Institute of Engineering, Electronics, and Analytics

In cooperation with other institutes of the Forschungszentrum Jülich GmbH as well as universities and other scientific institutions all over the world, the Central Institute of Engineering, Electronics, and Analytics (ZEA) develops and implements devices, experiments, processes, analytical procedures, measuring, and control equipment, detector systems, computer-assisted tools, and imaging techniques required for research that are not available on the market [3].
The institute is divided into three subinstitutes: Engineering and Technology (ZEA-1), Electronic Systems (ZEA-2), and Analytics (ZEA-3). This work is carried out in the ZEA-2.
During the development of complex system solutions in the ZEA-2, highly integrated System-on-Chip (SoC) solutions as well as fast data acquisition and processing play a big role [4]. The applications areas for these technologies comprise detector systems, measurement systems as well as nano- and microelectronic systems.
In the research field of quantum computing, the ZEA-2 develops scalable control and read-out electronics to enable the scalability of quantum computers. In this context, another main goal is the automated tuning of semiconductor double quantum dots (DQDs) for the realization of quantum bits (qubits). In this field, the ZEA-2 works in close cooperation with the JARA[1] Institute for Quantum Information, which is a shared initiative of

[1] Jülich Aachen Research Alliance

the RWTH[2] Aachen and the Forschungszentrum Jülich GmbH [5].
To understand the importance of this research field a short introduction to quantum computing is given in the following section.

1.2 Quantum Computing

Quantum computing describes the idea of using quantum effects to solve dedicated problems faster than on any classical computer. The history of quantum computing started in 1981 with the idea of Richard Feynman to draft a computer that enables the simulation of physics, especially quantum physics [6]. He stated that it is impossible to simulate quantum physics efficiently on a classical computer, thus motivating the need for quantum computers.
The first algorithm for quantum computers enables prime factorization in a polynomial runtime and was invented by Peter Shor in 1994 [7]. This algorithm endangers the safety of modern encryption methods like RSA[3] [8] if it can be executed on large quantum computers.
In 1997, the first algorithm was experimentally executed on a quantum computer in collaboration with IBM [9]. Many years later in 2019, a team lead by Google published a paper stating that its Sycamore processor is the first quantum computer achieving quantum supremacy. It was able to execute a task in 200 seconds which would take about 10,000 years for a classical super computer [10].
Today, there are many different possible application areas for which quantum computers are expected to be beneficial. These include optimization problems such as in artificial intelligence and machine learning, weather forecasting, traffic and supply chain optimization, as well as research use cases for material science or drug development [11]. Moreover, quantum computers will have an impact on cryptography as the widely used RSA-algorithm has to be replaced by other encryption methods, which do not rely on prime factorization. One possibility would be to use quantum cryptography methods instead.

1.3 Motivation

The smallest logical unit of a quantum computer is a qubit. The functionality of qubits is further described in Section 2.1. To enable the operation of semiconductor DQDs as

[2] Rheinisch Westfälisch Technische Hochschule
[3] invented by Rivest, Shamir, and Adleman

qubits, they have to be calibrated by a complex tuning process. Therefore, in a fundamental step, charge stability diagrams (CSDs) are recorded and analyzed. This work aims to contribute to the generation of simulated realistic CSDs, especially concerning the distortions present in experimental CSDs.
Simulating CSDs is an important task. In addition to the fact that the ZEA-2 cannot experimentally measure CSDs on its own, developing algorithms aiming at an automated set up of qubits (see Section 2.2) requires a large number of labeled datasets. Without a simulation, the acquisition and labeling are very time-consuming. The labeling of the datasets is also unreliable because even for a human expert it is not always visible where a transition line is proceeding and the ground truth is not known. Moreover, available data often does not reflect the real circumstances because only usable measurements are published. For creating an automated tuning algorithm also unusable datasets have to be detected to initiate appropriate measures for bringing the DQD back into a usable state.

The theoretical fundamentals for this thesis are laid out in Chapter 2. This includes a description of qubits in general (Section 2.1) and their implementation as DQDs (Section 2.2). For the simulation of undisturbed CSDs, a geometrical model derived from observations of physical basis models is used. These basis models are described in Section 2.4. Moreover, the identified noise types and artifacts which can appear in a CSD are explained in Section 2.5.
The methods used to determine the parameters for the simulation of CSDs are described in Chapter 3. This is a technically challenging task because individual noise types have to be extracted from experimental data with many different interfering distortions.
Finally, the simulated data are compared with actual measurements visually and by metrics to assess the quality of the simulated noise (Chapter 4). As CSDs are high dimensional datasets, the comparison of the distribution of simulated and actual measurements is not directly possible. Instead, the datasets have to be transformed into lower dimensional spaces or only specific features of the datasets can be compared. After the comparison of the first simulated dataset with experimental data, the parameters for the simulation are optimized and the final simulated dataset is evaluated with the same evaluation methods as before.
In addition to the evaluation, a discussion of the meaning of the evaluation results for the aim of this thesis follows (Chapter 5). In the end, a conclusion and an outlook are given.

2 Theoretical Fundamentals

This chapter provides the theoretical foundation for this thesis. This includes an explanation of the functionality of qubits and their implementation in DQDs, recent related work, different physical basis models for the simulation of undisturbed CSDs, and a description of the different distortions which often occur in a CSD.

2.1 Quantum Bits

The quantum equivalent to a bit is called a qubit. Contrary to a classical bit, a qubit stores a superposition of the two basis states "0" and "1". If the state of a qubit is measured, the superposition gets lost, and one of the basis states is measured with a certain probability. The state of a qubit can be represented in the ket notation $|\cdot\rangle$. This notation denotes that the states are orthogonal vectors. The state of a qubit can be given by

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

with $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$. In this case $|\alpha|^2$ and $|\beta|^2$ give the probability of measuring "0" or "1", respectively [12, chpt. 2.1].
Due to superposition, it is possible to represent 2^n states simultaneously with n qubits [13], whereas a classical computer would only be able to represent one of these states at a time.
Another quantum physical principle used in a quantum computer is the entanglement of qubits. If two qubits are entangled, their states can only be described for the whole system and not independently for each qubit. This means that when measuring one qubit, the result will always be correlated with the outcome of the measurement of the other qubit.
Taking superposition and entanglement together, a quantum computer can represent all combinations of zeros and ones in one step, while a classical computer would have to cycle through every bit combination step by step. However, when reading out the qubits,

the superposition is lost, meaning that one of the possibilities is chosen. Thus, for creating fast algorithms the problem cannot be treated as a black box, instead one has to enlarge the probability for the correct solution by exploiting destruction inference to minimize the probability of wrong possibilities [14].
There are many different possibilities to implement a qubit. One of them is the use of DQDs in a semiconductor heterostructure where the information of the qubit is encoded in the spin states of the electrons. The spin is a quantum mechanical property that can be imagined as a spin around its axis having an “up” or “down” state. An electron can also be in a superposition of these two states, which makes it feasible to use for the implementation of a qubit.

2.2 Tuning of Double Quantum Dots

In the setup used by the Quantum Technology Group of the RWTH, the quantum dots reside in a two-dimensional electron gas (2DEG) that is formed by a GaAs/AlGaAs[1] heterostructure. The gate layout used for their operation is shown in Figure 2.1. The voltages applied to the gate electrodes deplete the area underneath the gates. Thereby, the barrier gates, named with B, are intended to control the tunnel barriers between adjacent quantum dots as well as between quantum dots and the electron reservoir. Moreover, the plunger gates, named with P, are designed to affect the electrochemical potential of the related quantum dots. The tunnel barriers influence the probability for an electron to jump from one location in the heterostructure to another. The electrochemical potential determines the number of electrons that fit into the quantum dot.

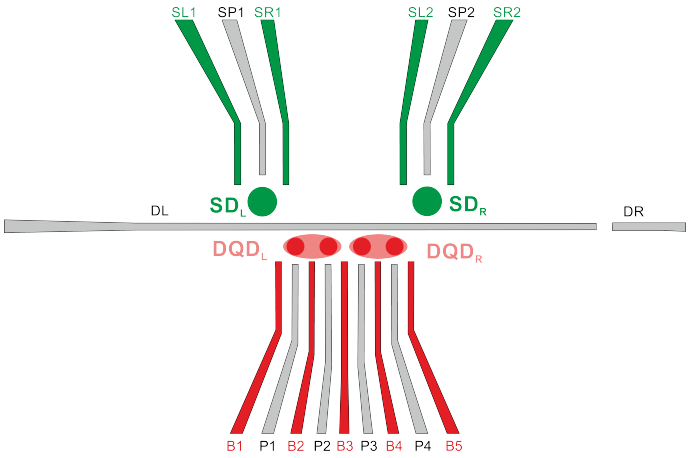


Figure 2.1: Gate layout for two DQDs. The green dots mark areas where sensing dots (SDs) are formed, and the red dots mark the areas for quantum dots.

[1] gallium-arsenide/aluminium-gallium-arsenide

The experimental setup is shown in Figure 2.2. It is similar to those described by Botzem [12] and Cerfontaine [15]. The DQD has to be cooled down in a dilution refrigerator because the qubit is stable only at a temperature of a few milli-Kelvin. Otherwise thermal noise would lead to a short decoherence time because the electrons would move too much to catch them in the quantum dots. To thermalize the signal again, different cooling stages are used. Furthermore, the setup is split into direct current (DC) control and readout as well as radio frequency (RF) reflectometry control and readout. The simpler DC measurements are used for steps in the tuning of the quantum dots which make a slower control in the timescale of milliseconds to seconds acceptable. Moreover, they are used for the first tuning steps which are only possible with DC measurements. RF measurements, on the other hand, are used to operate the qubit in timescales of nanoseconds but also for other steps of the tuning procedure.

The voltages for the DC measurements are generated by a digital-to-analogue converter (DecaDAC). There are several low-pass filters installed in the breakout box, the RC-filter box, and the sample printed circuit board (PCB), which should reduce the noise coming from the control electronics and the environment. The gate structure shown in Figure 2.1 is placed on the sample PCB.

Two carrier signals are generated by the RF source for the readout of the SDs. Less filters are used for the RF readout than for the DC readout, resulting in more noise in the measured signal.

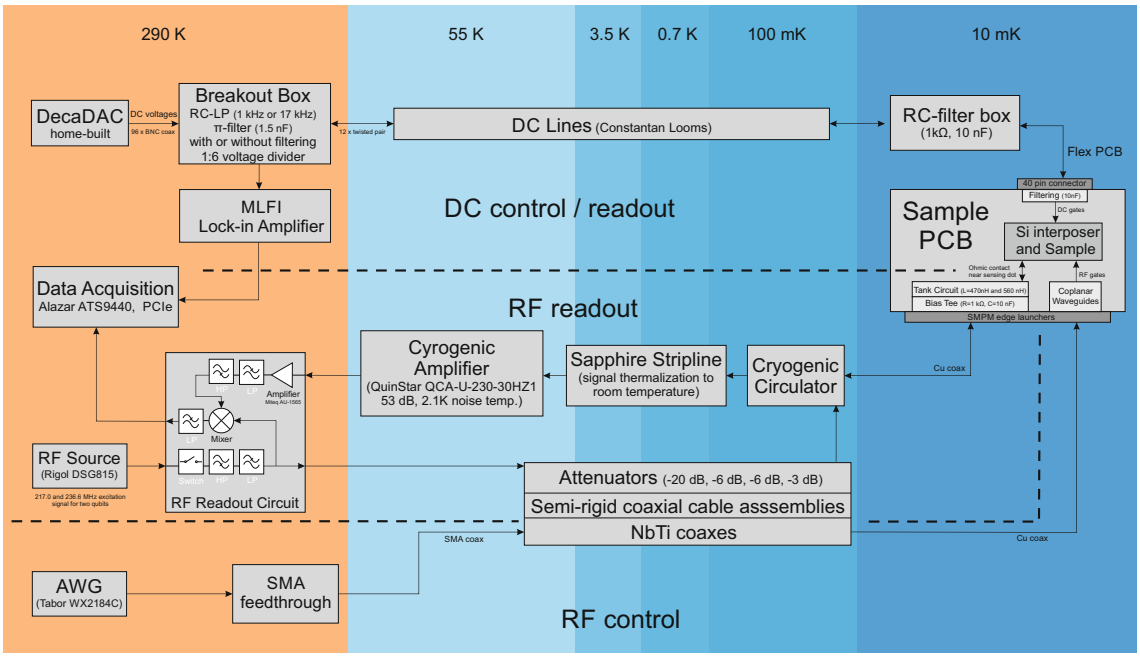
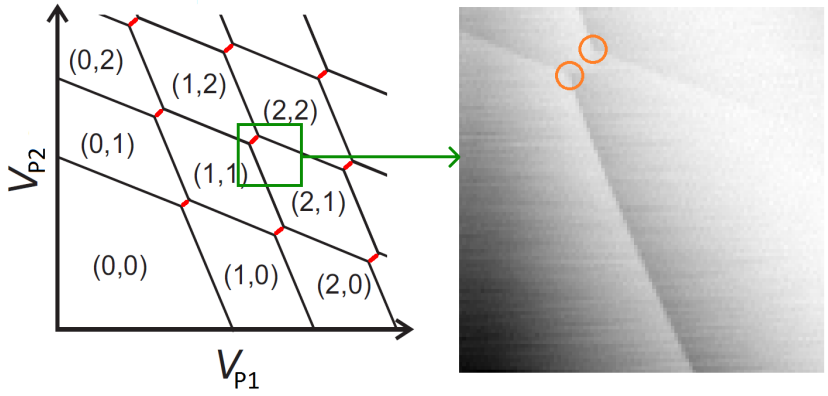


Figure 2.2: Experimental setup with different cooling stages, adapted from [16]

For operating DQDs as qubits, a tuning process [17] has to be executed. It is divided into a coarse- and a fine-tuning part. As a first step of the coarse-tuning, the SD is set to a

state that it is optimally sensitive to the charges in the DQD. After that, the gate voltages for the DQDs are adjusted to build two charge islands for every DQD. To overcome the problem of capacitive coupling between different gate electrodes, virtual gates can be configured next or later. Each virtual gate should affect only one parameter of the system by using a linear combination of multiple physical gates compensating for the capacitive coupling. The final step of the coarse-tuning tunes the occupation of the charge islands. In the fine-tuning stage, among other things, the inter-dot tunnel coupling and the coupling to the leads are adjusted. Furthermore, some specific configuration points are defined that are used to operate, readout, and reset the qubit. This is required to be able to execute high-fidelity quantum gates[2] on the qubit [18].

CSDs help to determine the electron occupation inside the quantum dots. In the described setup, they are derived by measuring the conductance through a nearby SD in dependency on the voltages applied to the two plunger gates of the DQD. While the gate on the horizontal axis of a CSD is swept continuously and the sensor response averaged over a pixel distance is saved as the pixel value, the vertical axis gate is changed stepwise after the measuring of a line has finished. The typical structure of a CSD looks like a honeycomb pattern. The left side of Figure 2.3 shows this pattern where (N_l, N_r) indicates the number of electrons in the left respective right quantum dot. On the right side, a measured CSD is visible. The lines in the schematic CSD as well as the edges in the measured one indicate a transition of an electron between the reservoir and the DQD or between the dots, also called lead and inter-dot transition, respectively. However, in the measured CSDs, the transition lines are often diffuse, disturbed by noise and artifacts, or sometimes not even visible.



(a) (b)

Figure 2.3: Example of a CSD. (a) Schematic CSD (adapted from [19, Ch. 8]) and (b) one experimentally measured part of it. The black lines in (a) indicate lead transitions, and the red lines indicate inter-dot transitions of an electron. The orange circles in (b) indicate the triple points.

[2] quantum gates are the basic quantum circuit operations used for any computation

With the help of a CSD, other properties of the DQD can also be determined. These are the inter-dot tunnel coupling as well as the tunnel coupling to the reservoir. These properties define how easily an electron can transition between the dots and between the reservoir and a dot. In the CSD, a high inter-dot tunnel coupling leads to curved triple points (marked by orange circles in Figure 2.3b) and blurred inter-dot transitions.
All experimental CSDs used in this thesis are kindly provided by the Quantum Technology Group of the RWTH.

2.3 Related Work

There are many recent approaches to the automated tuning of DQDs. In coarse-tuning, machine learning techniques play a dominant role in determining the charge state of the quantum dots. Many authors use convolutional neural networks (CNNs), which are artificial neural networks designed for image classification and pattern recognition [20, 21, 22, 23, 24], but also feed-forward neural networks (FFNNs) [25] and various binary classifiers [26] are applied.
In addition to the supervised machine learning techniques, classical signal processing is used. In [27], the authors remove the background of the CSDs and use a modified Hough transform [28] or EDLines [29] to detect curved lines.
Some of the mentioned methods already include simulated data in their training procedure [20, 21, 23, 24, 25]. However, they often do not include the simulation of distortions in the CSDs or only simulate preprocessed CSDs. Daruvola et al. [20] compared the results for synthetic and experimental training data, showing an advantage for experimental or a combination of synthetic and experimental data. As a reason for this, the authors stated that the noise in the experimental data was not understood well enough to accurately simulate it for synthetic training data.
According to Ziegler et al. [24], the performance of the CNN could be improved by including simulated data in the training procedure. Their simulated CSDs are an extension of the Qflow lite dataset [30], which does not include any distortions. However, they only provide their simulated datasets and not a framework for simulating CSDs, which would be beneficial, especially for examining the effect of different noise strengths on the tuning procedure. For the creation of simulated distorted CSDs, first, the undistorted occupation data for the CSD have to be created. Both Ziegler and Darulova used the QFlow lite dataset for this but Darulova also used a capacitive model. To create data with similar honeycomb patterns as visible in the experimental data, a model with adaptable parameters, like the capacitive model, should be used.

2.4 Physical Models for Charge Stability Diagrams

This section describes two different physical models used for the simulation of charge stability diagrams. The capacitive model (Section 2.4.1) does not include any quantum effects, whereas the Hubbard model (Section 2.4.2) includes tunnel coupling effects between the dots. At the end of the section, the models are compared.

2.4.1 Capacitive Model

In the capacitive model described by van der Wiel et al. [1], quantum effects are neglected. Each of the two dots in a double dot system is capacitively coupled to

- a plunger gate with voltage V_{p_i} and capacitance C_{p_i},
- the leads, that is the source and drain with voltage V_{l_i} and capacitance C_{l_i},
- the other dot with capacitance C_m, and
- to the plunger gate of the other quantum dot with capacitance C_{ij}, which was neglected in [1] but added in [31].

These parameters are also schematically visualized in Figure 2.4.

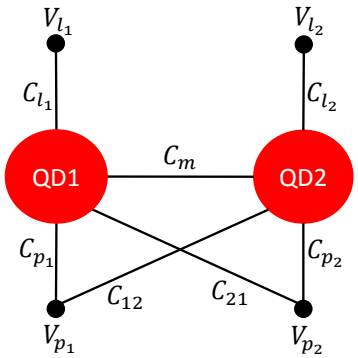


Figure 2.4: Schematic visualization of the different capacitances and voltages incorporated in the model. Figure adapted from [31].

The charge Q_j at each conductor, i.e., gate or dot, in the system can be derived from the sum of the charges on all capacitors connected to this conductor

$$Q_j = \sum_{k=0}^{N} q_{ij} = \sum_{k=0}^{N} C_{jk}(V_j - V_k), \tag{2.1}$$

where V_j is the electrostatic potential of conductor j and the summation runs over every dot and gate the conductor j is coupled to. This can also be written in a more compact form with the help of the capacitance matrix $\mathbf{C}$:

$$\vec{Q} = \mathbf{C}\vec{V}. \tag{2.2}$$

The diagonal elements of $\mathbf{C}$ consist of the total capacitances of the respective dot or gate, while the off-diagonal element C_{ij} is the negative capacitance of conductor i and conductor j. This system of equations can be divided into subsystems

$$\vec{Q} = \begin{pmatrix} \vec{Q}_c \\ \vec{Q}_v \end{pmatrix} = \begin{pmatrix} \mathbf{C}_{cc} & \mathbf{C}_{cv} \\ \mathbf{C}_{vc} & \mathbf{C}_{vv} \end{pmatrix} \begin{pmatrix} \vec{V}_c \\ \vec{V}_v \end{pmatrix} = \mathbf{C}\vec{V}, \tag{2.3}$$

where $\vec{Q}_c$ and $\vec{V}_c$ are the charges and voltages on the dots and $\vec{Q}_v$ and $\vec{V}_v$ are the charges and voltages on the voltage sources. $\vec{Q}_c$ is given by $-|e|(\vec{N} - \vec{N}_0)$ with $\vec{N}_0$ denoting the electron numbers in the dots when no voltage is applied, $|e|$ denoting the charge of one electron, and $\vec{N}$ denoting the current number of electrons in the dots. As the voltages on the voltage sources are known, the capacitance matrix has to be inverted only for the voltages at the dots, leading to

$$\vec{V}_c = \mathbf{C}_{cc}^{-1}(\vec{Q}_c - \mathbf{C}_{cv}\vec{V}_v). \tag{2.4}$$

The total electrostatic energy of the system is defined by

$$E(N_1, N_2) = \frac{1}{2}\vec{V}^{\mathsf{T}}\vec{Q} = \frac{1}{2}\vec{V}^{\mathsf{T}}\mathbf{C}\vec{V}. \tag{2.5}$$

When calculating the electrostatic energy of the dots, Equation (2.4) can be plugged into Equation (2.5) leading to

$$E_{dot}(N_1, N_2) = \frac{1}{2}(\vec{Q}_c - \mathbf{C}_{cv}\vec{V}_v)^{\mathsf{T}}\mathbf{C}_{cc}^{-1\,\mathsf{T}}(\vec{Q}_c - \mathbf{C}_{cv}\vec{V}_v). \tag{2.6}$$

From this, the electrochemical potential of the dots, defined as the energy to add the last electron to the dot, can be derived by

$$\begin{aligned} \mu_{dot1}(N_1, N_2) &= E_{dot}(N_1, N_2) - E_{dot}(N_1 - 1, N_2) \\ \mu_{dot2}(N_1, N_2) &= E_{dot}(N_1, N_2) - E_{dot}(N_1, N_2 - 1) \end{aligned} \tag{2.7}$$

Finally, the CSD is defined by using the electrochemical potentials to determine the equilibrium electron numbers N_1 and N_2 as a function of the gate voltages V_{p_1} and V_{p_2}. These are the highest values of N_1 and N_2 for which both $\mu_1(N_1, N_2)$ and $\mu_2(N_1, N_2)$ are less

than or equal to zero. This leads to a system of inequalities in the form

$$\mathbf{A}\vec{N} - \vec{B} \leq 0, \tag{2.8}$$

as the quadratic terms of $\vec{Q}_c$ and thus $\vec{N}$ in Equation (2.6) cancel out because of the difference between the two electrostatic energies in Equation (2.7) for the electrochemical potential.
For simulations with this model, the implementation from the bachelor thesis of Hangleiter [31] is used. First, Hangleiter solves the system of inequalities in Equation (2.8) for equality and then determines the correct physical states. The variables $\mathbf{A}$ and $\vec{B}$ are calculated for every gate voltage applied by exploiting

$$\vec{\mu}(\vec{N} = \vec{0}) = \vec{B} \text{ and } \vec{\mu}(N_i = 1, N_j = 0, j \neq i) + \vec{B} = \mathbf{A}_{\cdot i}, \tag{2.9}$$

where $\mathbf{A}_{\cdot i}$ is the i-th column of $\mathbf{A}$. Then, the linear system can be solved with the help of the least squares method. Afterwards, the correct physical configuration is found with the help of the algorithm presented in Figure 2.5.

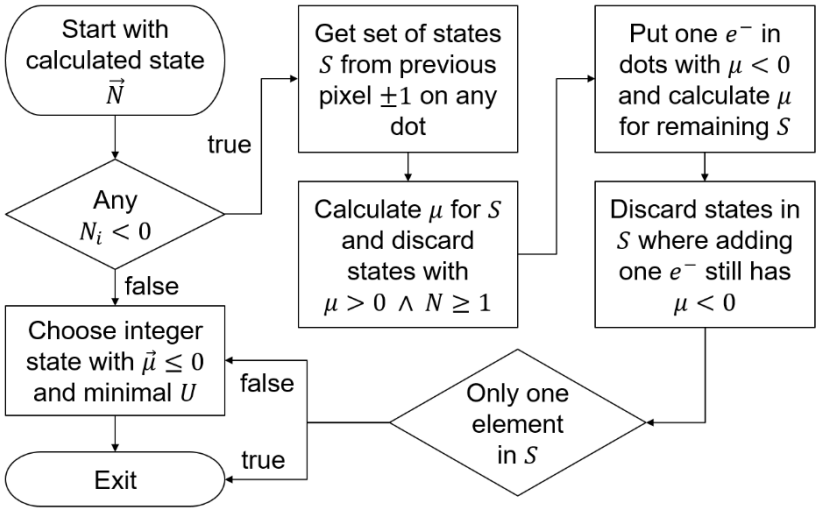


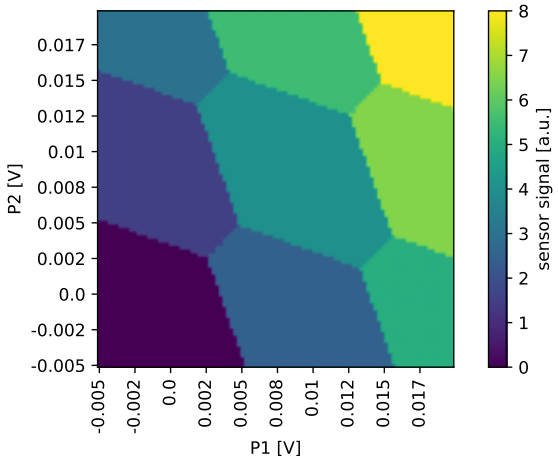
Figure 2.5: Flowchart of the algorithm used to determine the correct physical number of electrons [31]

The required parameters for the simulation of a CSD using the described capacitive model are listed in the following. For every parameter, the effect is shown in example images (see Figures 2.7 to 2.11). The default image which can be used for the comparison is shown in Figure 2.6.

- `number_dots` is the number of dots in the system. For a DQD, this equals two.

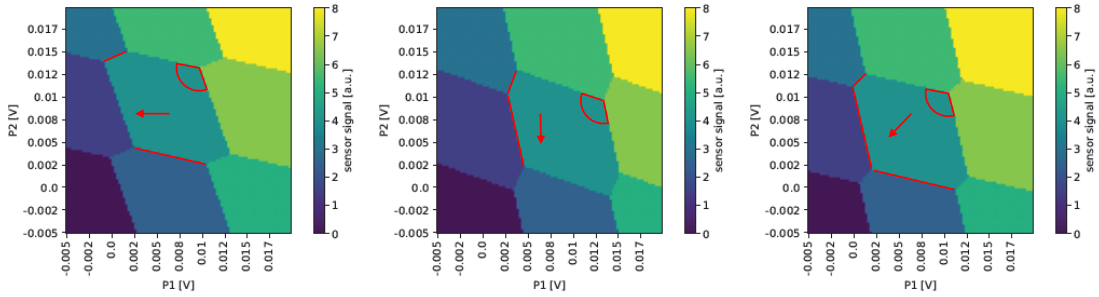
- `electron_number_no_voltage` = $\begin{pmatrix} n_{0_1} \\ n_{0_2} \end{pmatrix}$
  represents the number of electrons present in the dots without external voltage applied. For higher values, more negative voltages are needed to deplete the quantum dots. Thus, the honeycomb structure of the CSD is shifted to the left, respectively, lower side. This parameter should be set to 0 for both dots to be consistent with the Hubbard model, which is presented in Section 2.4.2.
- `lead_capacitances` = $\begin{pmatrix} C_{l_1} \\ C_{l_2} \end{pmatrix}$
  contains the capacitances for the coupling to the leads. Larger values result in a smaller angle, greater than 90°, between the lead transitions and a smaller length of the inter-dot transition, because the slopes of the lead transitions are changed. Moreover, the honeycomb structure is shifted to the left, respectively, lower side of the image. The impacts of changing this parameter are visible in Figure 2.7.
- `gate_capacitances` = $\begin{pmatrix} C_{p_1} \\ C_{p_2} \end{pmatrix}$
  contains the plunger gate capacitances. For larger values, the honeycombs are smaller on the corresponding axis of the diagram. The impacts of changing this parameter are visible in Figure 2.8.
- `cross_capacitances` = $\begin{pmatrix} C_{12} & C_{21} \end{pmatrix}$
  is an (n-1)x2 matrix containing the capacitances between the dots and the next-neighbor plunger gates. It is reasonable to set $C_{12} = C_{21}$ because the gate layout is symmetric. In that case, the honeycombs are more compressed diagonally, along the angle bisector between both axes for higher values (see Figure 2.9).
- `cross_dot_capacitances` = $\begin{pmatrix} C_m \end{pmatrix}$
  is an (n-1)x1 matrix containing the cross-dot capacitance between the dots. It affects the angle between the lead transitions which increases with larger values. Moreover, the length of the inter-dot transition increases (see Figure 2.10).
- `lead_voltages` = $\begin{pmatrix} V_{l_1} \\ V_{l_2} \end{pmatrix}$
  contains the lead voltages. The lead voltages are the voltages applied at the source and drain, i.e., the electron reservoir. For higher values, the honeycomb structure shifts to the left, respectively, lower side of the diagram. This effect can be seen in Figure 2.7.
- `coeff` = $\begin{pmatrix} ax_1 & ay_1 & b_1 \\ ax_2 & ay_2 & b_2 \end{pmatrix} \overset{\text{(no virtual gates)}}{=} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$
  contains the coefficients for setting the voltages for the virtual gates according to $Vg = \vec{ax} * Vx + \vec{ay} * Vy + \vec{b}$.
- `lims` = $\begin{pmatrix} \min Vx & \max Vx \\ \min Vy & \max Vy \end{pmatrix}$
  contains the voltage ranges for the plunger gates.

- `nPoints` = $\begin{pmatrix} nx & ny \end{pmatrix}$
  contains the number of measuring points in the horizontal and vertical direction.



`lead_capacitances`: $2 \cdot 10^{-17}, 2 \cdot 10^{-17}$,
`gate_capacitances`: $1.5 \cdot 10^{-17}, 1.5 \cdot 10^{-17}$,
`cross_capacitances`: $1 \cdot 10^{-18}$,
`cross_dot_capacitances`: $1.5 \cdot 10^{-17}$,
`lead_voltages`: $0.001, 0.001$

Figure 2.6: Example of a CSD simulated with the capacitive model. The parameters used here are taken as default for Figures 2.7 to 2.11. They are chosen to visualize the effect but are not assumed to be realistic.

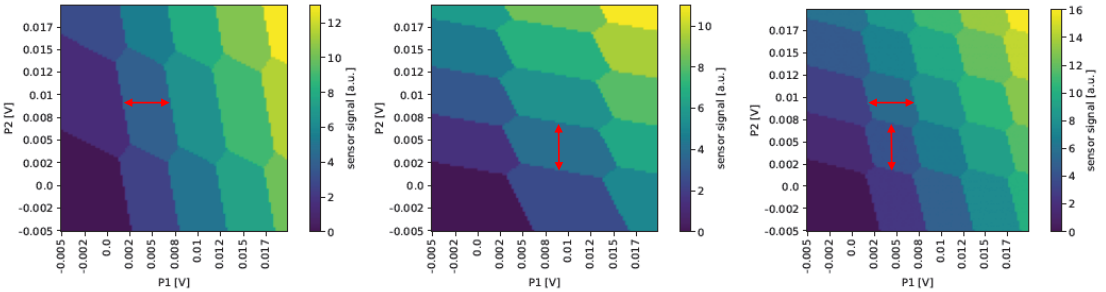


(a) `lead_capacitances`: $6 \cdot 10^{-17}, 2 \cdot 10^{-17}$

(b) `lead_capacitances`: $2 \cdot 10^{-17}, 6 \cdot 10^{-17}$

(c) `lead_capacitances`: $6 \cdot 10^{-17}, 6 \cdot 10^{-17}$

Figure 2.7: Effects of the change of `lead_capacitances`. Larger values result in a smaller angle between the lead transitions, and smaller length of the inter-dot transition, as well as a shallower, respectively steeper, slope of the lead transitions in P2/P1-direction, and a shift of the honeycomb to the left, respectively, lower side of the diagram a change of C_{l_1}/C_{l_2}.

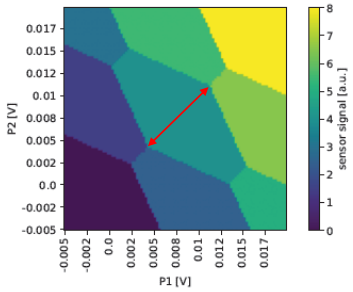


(a) `gate_capacitances`: $3 \cdot 10^{-17}, 1.5 \cdot 10^{-17}$

(b) `gate_capacitances`: $1.5 \cdot 10^{-17}, 3 \cdot 10^{-17}$

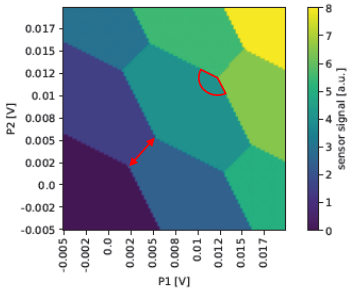
(c) `gate_capacitances`: $3 \cdot 10^{-17}, 3 \cdot 10^{-17}$

Figure 2.8: Effects of the change of `gate_capacitiances`. For larger values, the lead transitions are closer together on the corresponding axis of the diagram.



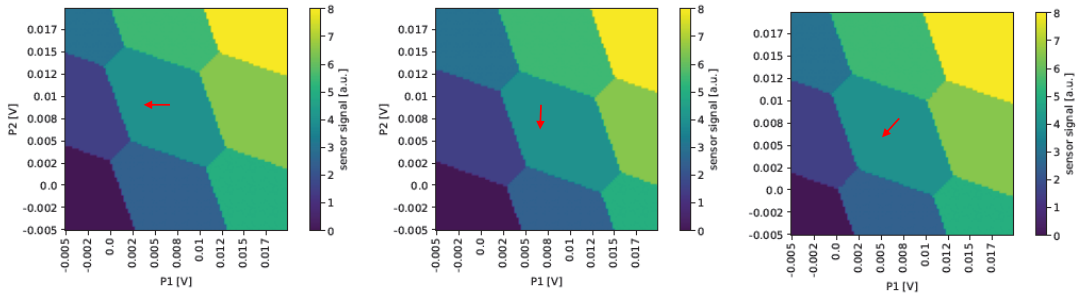
`cross_capacitances`: $3 \cdot 10^{-18}$

Figure 2.9: Effects of the change of `cross_capacitances`. The honeycombs are more compressed diagonally, along the angle bisector between horizontal and vertical axis for higher values.



`cross_dot_capacitances`: $3 \cdot 10^{-17}$

Figure 2.10: Effects of the change of `cross_dot_capacitances`. With increased values, the angle between the lead transitions increases and the inter-dot transition becomes longer.



(a) `lead_voltages`: 0.003, 0.001

(b) `lead_voltages`: 0.001, 0.003

(c) `lead_voltages`: 0.003, 0.003

Figure 2.11: Effects of the change of `lead_voltages`. For higher values, the honeycomb structure shifts to the left respective lower side of the diagram.

2.4.2 Hubbard Model

In the absence of quantum effects, the Hubbard model is equivalent to the capacitive model [2]. However, it can be extended to include quantum effects such as tunnel coupling between the two quantum dots. Yang et al. [2] limited their Hubbard model to a DQD with each dot capable of holding between zero and two electrons.
To find a mapping between the capacitive and the Hubbard model, Equation (2.6) should be reformulated using the charging energies E_{C_i} and electrostatic coupling energies E_{C_m} instead of the capacitance matrix. The matrix containing these energies is defined by

$$\mathbf{E}_C = e^2 \mathbf{C}_{cc}^{-1}, \tag{2.10}$$

where the diagonal of $\mathbf{E}_C$ contains the charging energy and the off-diagonal entries contain the electrostatic coupling energy. The charging energy is the energy required to add one electron to the dot, while the electrostatic coupling energy describes the change in energy on one dot when an electron is added to the other dot. Using the energy matrix $\mathbf{E}_C$ instead of $\mathbf{C}_{cc}$ leads to

$$E_{dot}(N_1, N_2) = \frac{1}{2} e^2 (\vec{Q}_c - \mathbf{C}_{cv}\vec{V}_v)^\mathsf{T} \mathbf{E}_C{}^\mathsf{T} (\vec{Q}_c - \mathbf{C}_{cv}\vec{V}_v). \tag{2.11}$$

For the Hubbard model without quantum effects, the electrostatic energy can be described by

$$E'(N_1, N_2) = \sum_{i=1,2} \left[\mu_i N_i + \frac{U_i}{2} N_i (N_i - 1) \right] + U_{12} N_1 N_2, \tag{2.12}$$

where U_i is the Coulomb interaction[3] at dot i and U_{12} is the Coulomb interaction between the dots. The electrochemical potential can be calculated by

$$\begin{aligned} \mu_1 &= |e|(\alpha_1 V_{P_1} + (1 - \alpha_1) V_{P_2}) + \gamma_1, \\ \mu_2 &= |e|((1 - \alpha_2) V_{P_1} + \alpha_2 V_{P_2}) + \gamma_2. \end{aligned} \tag{2.13}$$

Comparing Equation (2.12) with Equation (2.11) leads to the following mapping [2]:

$$\begin{aligned} U_i &= E_{C_i}, \\ U_{12} &= E_{C_m}, \\ \gamma_i &= -U_i/2, \\ \alpha_1 &= \frac{(U_2 U_{12}) U_1}{U_1 U_2 U_{12}^2}, \\ \alpha_2 &= \frac{(U_1 U_{12}) U_2}{U_1 U_2 U_{12}^2}. \end{aligned} \tag{2.14}$$

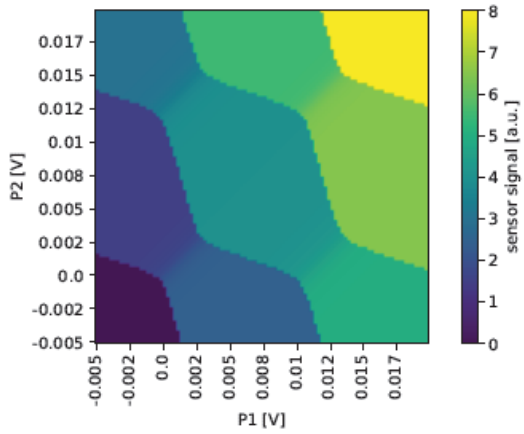
[3] electrostatic interaction between electric charges

When quantum effects should be visible in the simulated CSD, this model has to be extended. The extended Hubbard model can be described by

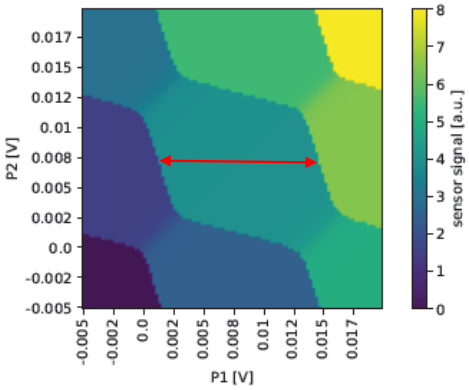
$$H = H_\mu + H_U + H_t + H_J \tag{2.15}$$

with H_μ and H_U describing the terms including the electrochemical potential and the Coulomb interaction which are already included in the Hubbard model without quantum effects, H_t representing the terms belonging to tunneling effects, and H_J including spin-exchange, pair-hopping, and occupation-modulated hopping. Spin-exchange means that two electrons swap their spins, while the overall spin state in the system does not change. Pair-hopping means that two electrons tunnel from one dot into the other at the same time, and the occupation-modulated hopping term describes situations where the occupation of the dots influences the electron tunneling. However, these second-order terms have only a negligible influence on the electrostatic energy of the system for low occupation numbers. To simulate a CSD, the eigenenergy has to be computed first. This is done by computing an eigenvalue decomposition of the Hamiltonian matrix belonging to H. The electron number in the two dots is then calculated by computing the minimum of the eigenenergy. For the simulation of CSDs with the Hubbard model, the implementation available in QuDiPy [32] is used. It is capable of simulating n-dot systems. However, each dot can hold only two electrons. The term H_J is neglected in this implementation. Parameters for the Hubbard model implementation are listed in the following.

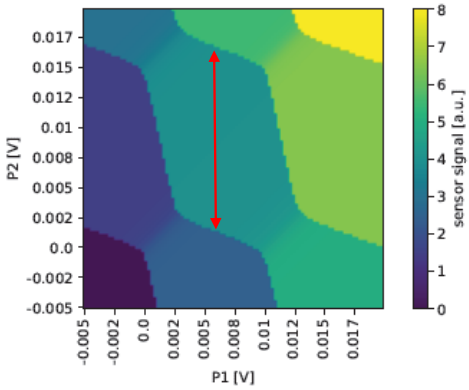
- `n_sites` is the number of dots, being two for a DQD.
- `n_electrons` is the maximal number of electrons in the dot system, with two times the number of dots being the maximal possible value. As many transition lines are favored, this value is set to four.
- `U_11`, `U_22`, and `U_12` are the Coulomb interactions at the two dots and between the dots. `U_11` and `U_22` stretch the honeycombs horizontally and vertically, respectively (see Figure 2.12b and Figure 2.12c), whereas `U_12` stretches the honeycomb diagonally and thus influences the angle at the triple points and the length of the inter-dot transition (see Figure 2.12d). By stretching the honeycombs the slopes of the lead transitions are also affected.
- `t_12` is the tunnel coupling between the dots. With increasing values the inter-dot transitions get more blurred and the triple points more curved (see Figure 2.12e).
- `inital_v` $= \begin{pmatrix} \texttt{v\_g1\_min} & \texttt{v\_g2\_min} \end{pmatrix}$ is the initial voltage applied at the double dot plunger gates
- `v_g1_max`, and `v_g2_max` determine the maximal voltage applied at the double dot plunger gates.
- `num` gives the resolution in the number of pixels along each axis leading to a CSD with $\texttt{num}^2$ pixels.



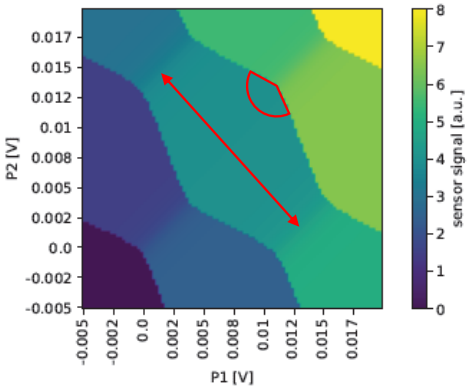
(a) $\texttt{U\_11} = 8.36 \cdot 10^{-3}$, $\texttt{U\_22} = 9.38 \cdot 10^{-3}$, $\texttt{U\_12} = 2.60 \cdot 10^{-3}$, $\texttt{t\_12} = 0.30 \cdot 10^{-3}$



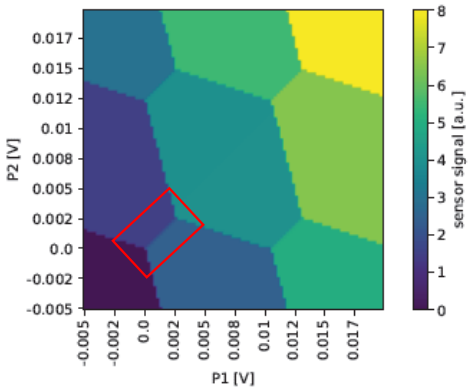
(b) $\texttt{U\_11} = 10.36 \cdot 10^{-3}$
Horizontally stretched for larger values



(c) $\texttt{U\_22} = 11.38 \cdot 10^{-3}$
Vertically stretched for larger values



(d) $\texttt{U\_12} = 3.60 \cdot 10^{-3}$
Diagonally stretched for higher values, influencing the angles



(e) $\texttt{t\_12} = 0.01 \cdot 10^{-3}$
Sharper inter-dot transition for lower values

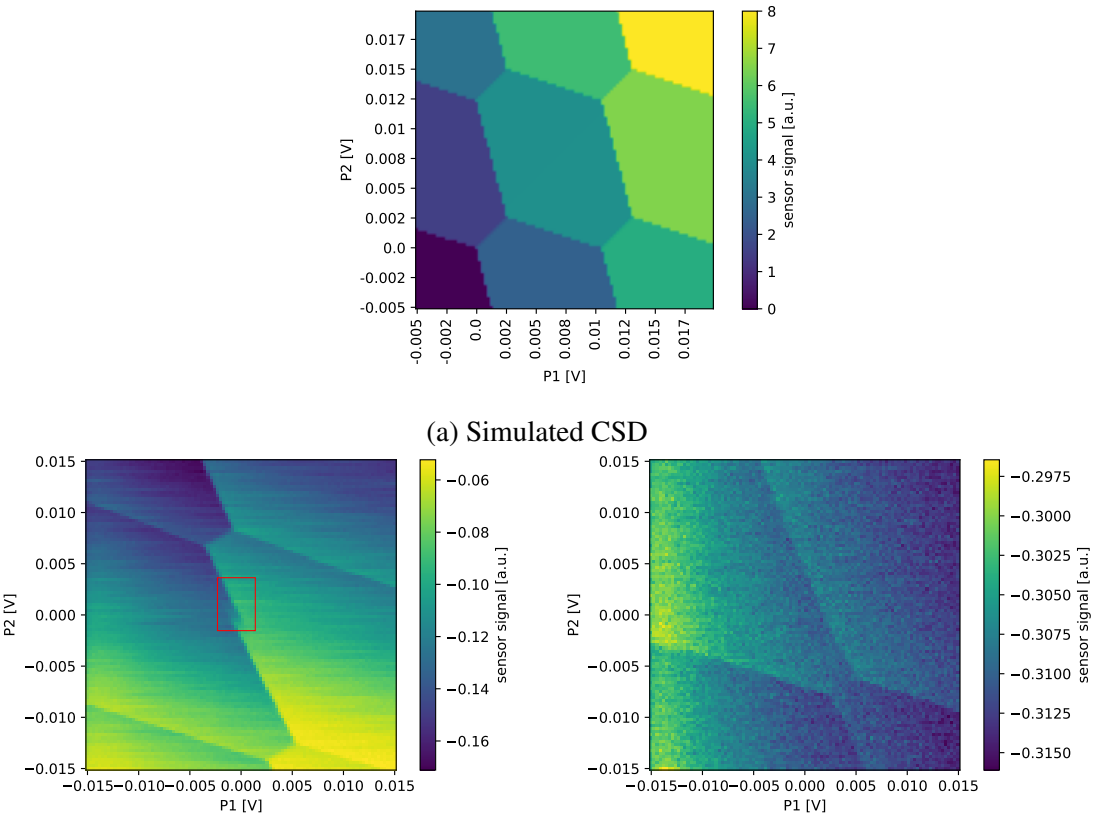
Figure 2.12: Examples of CSDs simulated with the Hubbard model for different parameters. The parameters are chosen to visualize their effect but are not assumed to be realistic. In (b, c), the slope of the respective lead transition is also influenced.

2.4.3 Comparison of Models

Both the capacitive and the Hubbard model have advantages and drawbacks. The capacitive model has more parameters, which leads to more possible shapes of the honeycomb pattern. However, it is also harder to determine useful parameters because more than one parameter influences the same property of the diagram. The Hubbard model has less parameters and the authors of this model describe a way to determine these parameters from experimental CSDs [2]. This makes it easier to create authentic simulated diagrams.
While the capacitive model does not contain quantum effects, it can, unlike the Hubbard model, simulate more than two electrons per dot. However, this does not impose a strict limitation on the Hubbard model as also the available experimental CSDs usually only contain one inter-dot transition. Moreover, if applications focus on triple points and inter-dot transitions, simulations from the Hubbard model should be included, because quantum effects change their appearance drastically. For applications that mainly analyze lead transitions, e.g., for classification if a single or double dot is present, the capacitive model might suffice. However, it might lead to worse classification results for these applications if the transitions are more curved.
Generally, the Hubbard model is more beneficial for the simulation of the ground truth CSD data as it includes more relevant phenomena of the real world.

2.5 Distortions in Charge Stability Diagrams

The CSDs examples presented in Section 2.4 are still very different from the CSDs generated by experimental measurements (see Figure 2.13). This is because there are many different types of distortions in experimental CSDs.
Instead of homogeneous honeycombs, value drifts are visible inside them. This phenomenon results from a cross-coupling between the SD and the plunger gates of the DQD. Moreover, different types of noise are present in the experimental CSDs and at dot jumps the transition lines are shifted (see red box in Figure 2.13).
In this section, the different noise types and artifacts that appear in CSDs are analyzed, including their physical sources and simulations as well as the parameters required for their simulation.



(a) Simulated CSD

(b) Experimental CSDs

Figure 2.13: Comparison between (a) a CSD simulated with the Hubbard model and (b) experimental ones. The red box in the first experimental CSD marks a dot jump further described in Section 2.5.5.

2.5.1 Cross-Coupling between Sensor and Double Dot Plunger Gates

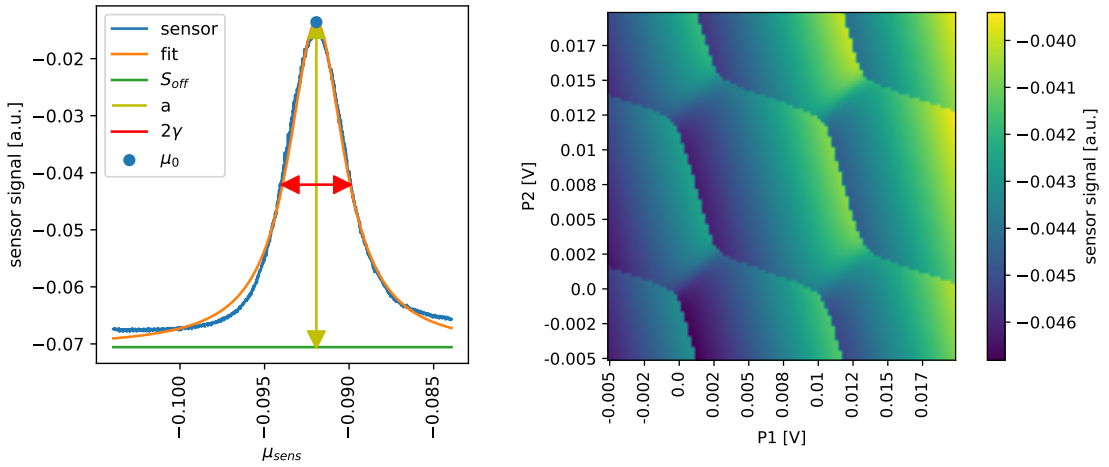
Cross-coupling effects between the SD and the plunger gates of the DQD influence the visibility of the charge transitions in the CSDs and lead to value shifts inside the honeycombs. Physical models like the Hubbard model only simulate the occupation states for the different gate voltages. They incorporate a very basic sensor response which only consists of a linear combination of the number of electrons in the dots.
The following model gives a more realistic version of the SD response:

$$\begin{aligned} \mu_{sens} &= \sum_{i=1}^{2} \left[\alpha_{dot,i} \cdot n_{elec,i} + \alpha_{gate,i} \cdot v_{gate,i} \right] + \mu_{off} \\ S &= S_{off} + a \cdot \frac{\gamma^2}{\gamma^2 + (\mu_{sens} - \mu_0)^2}, \end{aligned} \tag{2.16}$$

where $i = 1$ corresponds to the left dot, whose plunger voltage is usually displayed on the horizontal axis, and $i = 2$ to the right dot, whose plunger voltage is usually displayed on

the vertical axis of the CSD. In this model, μ_{sens} represents the electrochemical potential of the SD, which is influenced by the number of electrons n_{elec} in the two dots and the voltages applied to the plunger gates v_{gate}. The parameters $\alpha_{dot,i}$ for the SD potential influence the sharpness of the edges, while the parameters $\alpha_{gate,i}$ influence the drifts within the honeycombs. The effect of the applied voltages and the number of electrons is contrary to each other. The number of electrons has a negative effect on the potential (α_{dot} negative), whereas the applied voltage has a positive effect (α_{gate} positive). Furthermore, an offset μ_{off} is added to the potential, which determines where the CSD measurement starts on the curve.

The SD response S is approximated by a simplified Lorentzian [33], see Figure 2.14a. γ determines its width and μ_0 is the potential at the peak of the curve. As the sensor response is transformed by linear filters in the experimental setup, the Lorentzian is scaled by a factor a and an offset S_{off} is added. In Figure 2.14b, an example of a simulated CSD that includes the cross-coupling effects is shown. Instead of the electrochemical potential, the sensor scans are measured in dependency on the voltage applied at the sensor plunger gate (see Figure 2.14a). As the applied voltage is proportional to the electrochemical potential at the sensor, both terms are used interchangeably in the following.



(a) Measured sensor response (blue) with Lorentzian fit (orange)

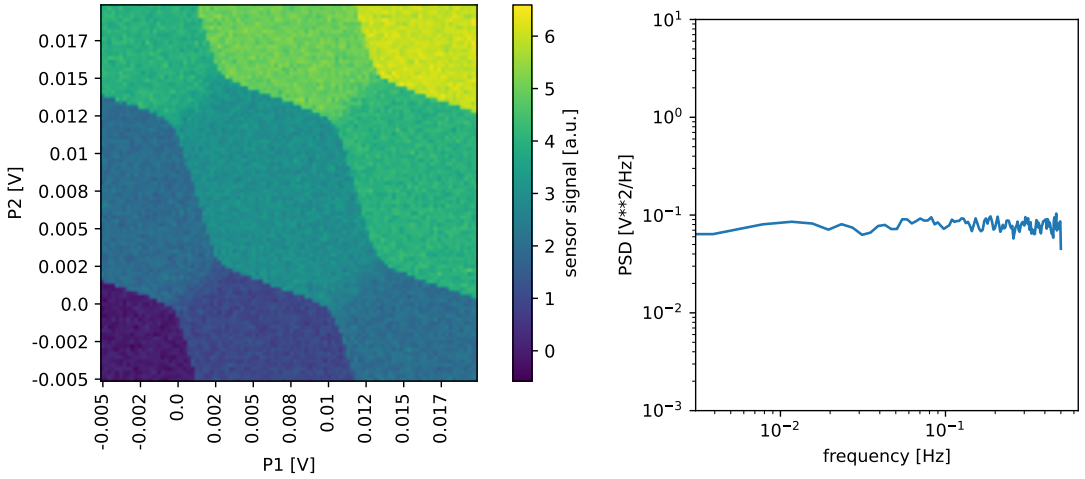
(b) Simulated CSD including sensor response

Figure 2.14: Example of the sensor response simulated at the left side of the peak leading to low values in the lower left region of the CSD and high values in the upper right region.

2.5.2 White Noise

White noise consists of thermal and shot noise. Thermal noise originates from the thermal agitation of charge carriers, usually electrons, in an electrical conductor [34]. Thus, it is

stronger for higher temperatures.
Unlike thermal noise, shot noise depends on the discrete charges in the current flow and does not have a relation to the temperature at which the system is operating [35].
Both types of noise are approximately white, which means that their strength is not frequency dependent leading to a constant power spectral density (PSD) (see Figure 2.15). While thermal noise has a nearly Gaussian amplitude distribution, shot noise can be modeled by a Poisson process. However, as the Poisson distribution can be approximated by a normal distribution, all white noise can be generated by adding Gaussian distributed random variables to the clean image. The only parameter needed is the standard deviation of the Gaussian distribution σ_{white}.



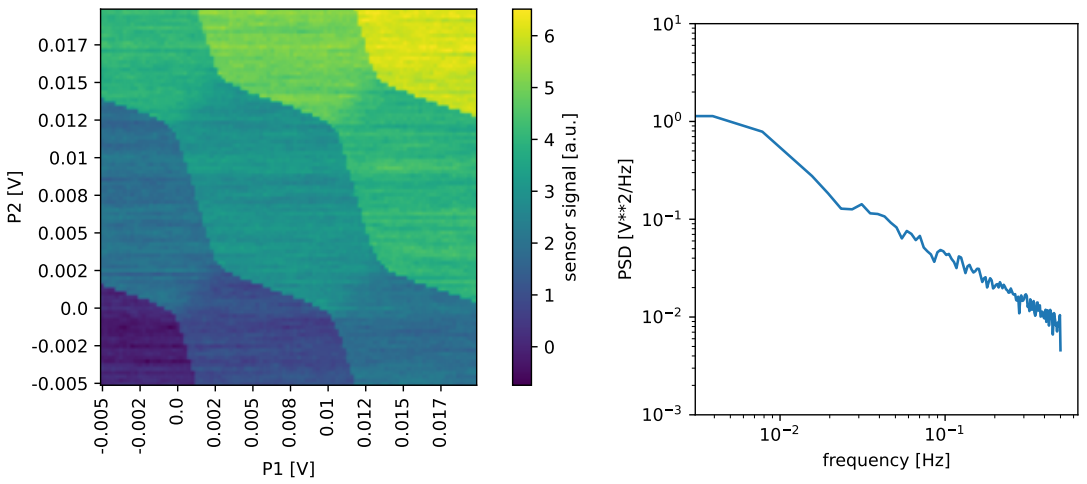
(a) Example

(b) PSD

Figure 2.15: Example for the white noise simulation. a) Simulated CSD with added white noise and b) PSD of the generated noise.

2.5.3 Pink Noise

Pink noise, also known as $1/f$ or flicker noise, is observed in most electronic devices, and originates from the internal heterogeneity of electronic components, such as oxide traps or lattice dislocations [36]. In contrast to white noise, the PSD is inversely proportional to the frequency (see Figure 2.16b). This leads to stripes inside the linewise measured CSDs that are also visible in the simulated CSDs with added pink noise (see Figure 2.16a).



(a) Example

(b) PSD

Figure 2.16: Example of the pink noise simulation. a) Simulated CSD with added pink noise and b) PSD of the generated noise.

The simulation of pink noise is described in [37] and implemented in the python module `colorednoise` [38]. The algorithm allows for randomness in both the phase and the amplitude of the noise time series. It consists of the following steps:

1. For each Fourier frequency ω_i of the data, draw two Gaussian distributed random numbers and multiply them by $\sqrt{\frac{1}{2}S(\omega_i)}$ with $S(\omega_i)$ being the power law spectrum of the noise. For pink noise, it equals $\frac{1}{\omega}$.

2. Use the generated random variables as the real and imaginary part of the Fourier transform of the desired noise time series. This leads to a Fourier transform $f(\omega) \sim \mathcal{N}(0, \frac{1}{2}S(\omega)) + i\mathcal{N}(0, \frac{1}{2}S(\omega))$.

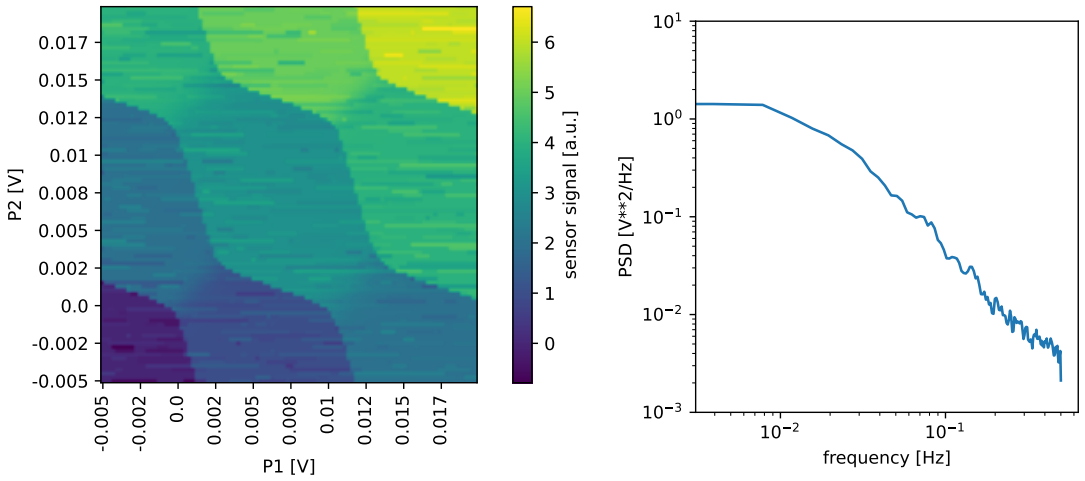
3. To obtain a real-valued time series, choose the Fourier transform of the negative frequencies as their complex conjugation.

4. Obtain the time series by applying a backward Fourier transform from the frequency to the time domain.

2.5.4 Random Telegraph Noise

Random telegraph noise (RTN) or burst noise randomly switches between two or multiple discrete levels [36]. It can be explained by a time-dependent random capture/emission

process of charge carriers caused by oxide traps [39]. Its PSD is proportional to $1/f^2$ (see Figure 2.17b), which means that it only plays a role for low frequencies. In the simulated image, it is visible as stripes with a specific beginning and ending (see Figure 2.17a).



(a) Example

(b) PSD

Figure 2.17: Example for the RTN simulation. (a) Simulated CSD with added RTN and (b) PSD of the generated noise.

It can be simulated using a geometric distribution for the occurrence of bursts and a normal distribution for the height of jumps [24]. The parameters for this noise type are the expectation of the length of the jumps, the standard deviation of the height of the jumps, and the expectation of the height of the jumps.

2.5.5 Dot Jumps

Dot jumps have a similar origin as RTN. However, while the RTN originates from the effects of random charge traps on the sensor, dot jumps describe a charge-trapping effect on the quantum dots themselves, which are pandered by fabrication-related imperfections. Dot jumps lead to structural offsets in the CSDs [24]. An analysis of the available experimental data shows, that the dot jumps most likely happen at specific voltage configurations and are thus deterministic. That means they can be simulated by shifting a block of columns horizontally or a block of lines vertically. In the first case, the first plunger causes the occurrence of the dot jumps. Thus, the charge trap also affects the electrochemical potential of the first dot, leading to a shift in the horizontal direction. In the other case, the second plunger causes the occurrence of dot jumps. Figure 2.18 shows a simulation of a dot jump in the horizontal direction.

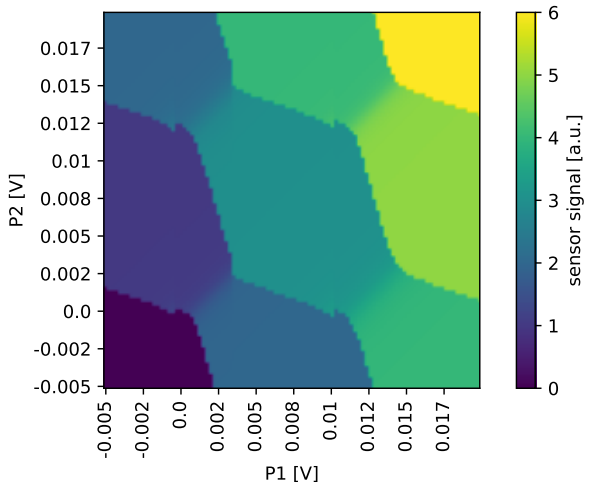


Figure 2.18: Example of a simulated CSD with dot jumps

The occurrence of jumps can be simulated by a geometric distribution, while the strength of the jumps is determined by a Poisson distribution [24]. The parameters are the expectation for the length of the jumps and the expectation for the height of the jumps.

3 Determination of Parameters for the Simulation of Charge Stability Diagrams

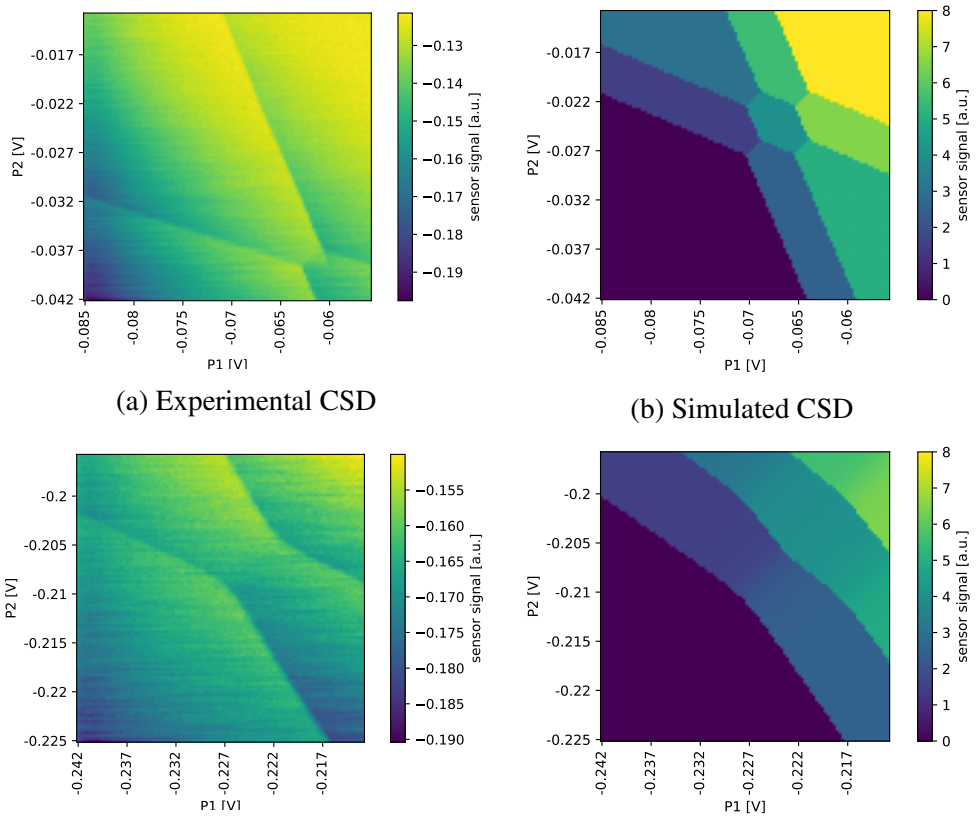
Depending on the experimental setup and various external influences, which can change for each cool-down cycle, the parameters' values for the identified distortions vary. To simulate realistic distortions for the CSD data, parameter ranges have to be extracted from the experimental images.
In addition to the parameters of the distortions, the parameters for the undisturbed CSDs have to be determined.

3.1 Hubbard Model

In [2], a strategy for determining the Coulomb interactions required for simulations with the Hubbard model is given. The inter-dot Coulomb interaction `U_12` is determined by the length of the inter-dot transition which equals $\sqrt{2} \cdot$ `U_12`. The authors assume equal onsite Coulomb interactions `U_11` and `U_22` and suggest deriving them from the slope of the (1,1)-(2,1)-transition. This slope equals $\frac{\alpha}{\alpha-1}$, where α refers to α_1 and α_2 from Equation (2.14). Then, U = `U_11` = `U_22` can be obtained from

$$\alpha = \frac{(U - \texttt{U\_12}) \cdot U}{U^2 - \texttt{U\_12}^2} = \frac{(U - \texttt{U\_12}) \cdot U}{(U - \texttt{U\_12}) \cdot (U + \texttt{U\_12})} = \frac{U}{U + \texttt{U\_12}}. \tag{3.1}$$

Using this approach, the length of the inter-dot transitions and slopes of the lead transitions are mimicked well (see Figure 3.1). For the shown images, the determination of the tunnel coupling `t_12` takes place manually by choosing a value that results in a similar blurring and rounding as in the experimental CSDs. However, the size of the honeycombs does not fit the size of the honeycombs in the experimental data.



(a) Experimental CSD

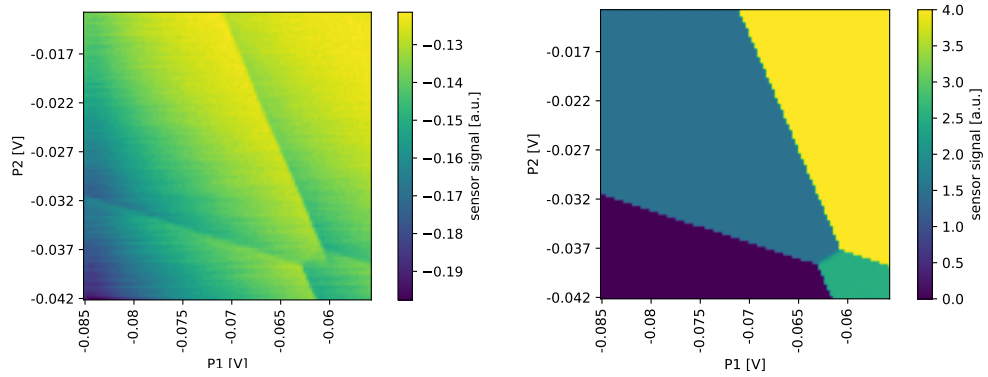
(b) Simulated CSD

(c) Experimental CSD

(d) Simulated CSD

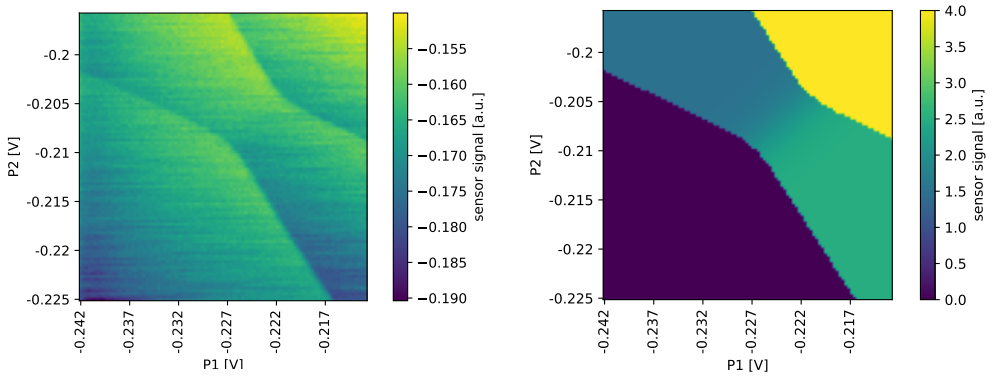
Figure 3.1: Two examples of noisy experimental CSDs with a simulated version without noise, whose parameters are determined as suggested in [2]

To overcome this problem, another model based on observations from the Hubbard model is used to simulate the honeycomb structure of the CSDs. This model is subject to current research by Fabian Hader, working at ZEA-2, with publications being expected in the future. The model uses the slopes of the lead transitions, the width and length of the inter-dot transition, and the lengths of the lead transitions in the horizontal and vertical direction, which can be extracted directly from the experimental CSDs (see Figure 3.3a). This leads to simulated CSDs that are very similar to the experimental ones (see Figure 3.2).



(a) Experimental CSD

(b) Simulated CSD



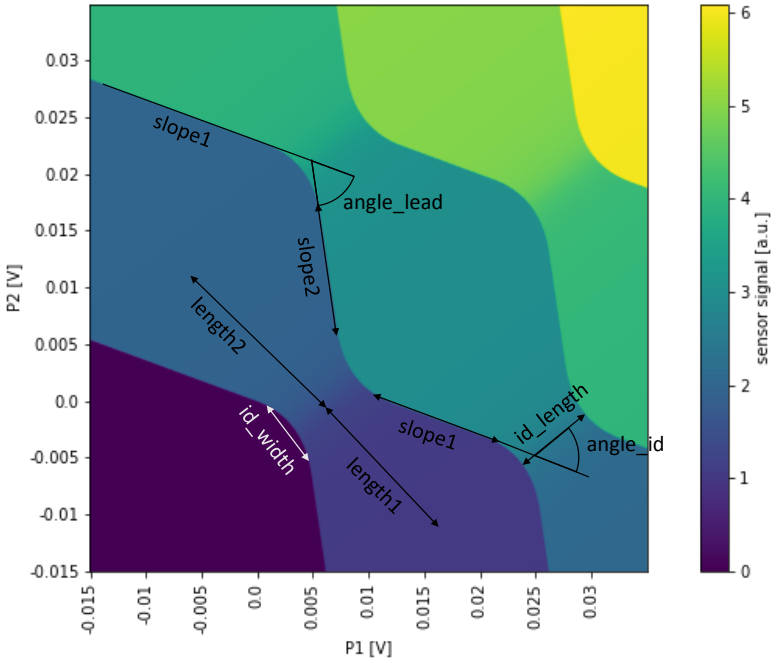
(c) Experimental CSD

(d) Simulated CSD

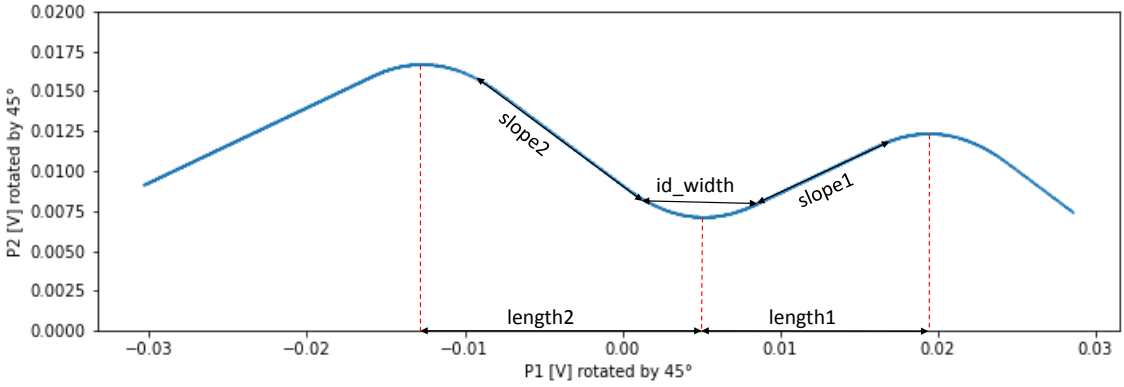
Figure 3.2: Two examples of noisy experimental CSDs with a noiseless version simulated with Hader's model

For the simulation of a large set of CSDs, parameter ranges have to be extracted from the experimental data. The different parameters which should be determined are visualized in Figure 3.3. As `slope2` is sometimes near infinity in the original CSDs the wavefronts are rotated by 45° for the simulation (see Figure 3.3b). The required parameters for the simulation are listed in the following.

- `slope1` is the slope of the lines that indicate a transition of an electron between the reservoir and the right dot. In the rotated space, this is the slope between a minimum and the following maximum of a wavefront.
- `slope2` is the slope of the lines that indicate a transition of an electron between the reservoir and the left dot. In the rotated space, this is the slope between a maximum and the next minimum.
- `length1` and `length2` are the lengths of the lead transitions given by the distance on the horizontal axis between a minimum and a maximum respective maximum and minimum in the rotated space.
- `angle_lead` names the angle between the lead transitions. It is used in the simulation to ensure that randomly chosen slopes have a reasonable relation to each other.
- `angle_id` denotes the angle between the lead transitions of the second quantum dot and the inter-dot transition. It is used in the simulation to determine the slope of the inter-dot transition.
- `id_length` is the length between two triple points, i.e., the length of the inter-dot transition.
- `id_width` specifies the length of the curvature of a transition line, which mainly depends on the tunnel coupling between the dots, i.e., the width of the inter-dot transition.



(a) Original



(b) Rotated

Figure 3.3: Visualization of the different parameters determined from the experimental CSDs. (a) Shows the parameters in the original image, and (b) shows the second wavefront of this image rotated by 45°.

To determine the parameter ranges, the points marked with arrows in Figure 3.3 are extracted from 60 experimental CSD with visible edges. However, CSDs that were measured with virtual gates cannot yet be created with the simulation model of Fabian Hader. This means that they are not used for the parameter determination and the evaluation of the simulated CSDs.

To determine the parameters from the experimental CSDs, the diagrams are displayed with the help of `OpenCV` and a callback function is added to the window containing the CSD. It is executed after a mouse click inside the image window and asks the user for the type of the point clicked on. After all the points are marked, the parameter ranges shown

in Table 3.1 are calculated. Only `length1` is not directly extractable from the CSDs used for the parameter determination because, in most of the diagrams, the voltages space is not large enough to show more than one inter-dot transition. To overcome this problem, the determined range for `length2` is assumed to be a reasonable approximation for `length1` due to the symmetry of the used gate layout.
Moreover, the relation between `id_width` and `id_length` is important for the simulation because a more curved triple point also leads to a longer inter-dot transition. The relation between `id_width` and `id_length` lies between 0.86922 and 9.05538.

Table 3.1: Parameters for the simulation of the honeycomb pattern extracted from the experimental CSDs. They are determined from the image rotated by 45°.

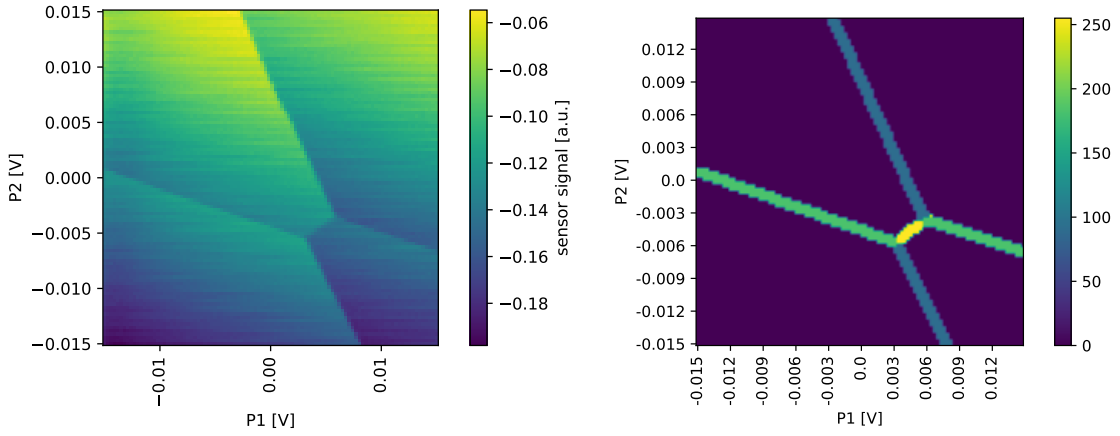
parameter	minimum	maximum
`slope1`	0.21429	0.54688
`slope2`	-0.44	-0.07692
`length1`	0.01543	0.01671
`length2`	0.01543	0.01671
`angle_lead`	25.07313°	96.19746°
`angle_id`	33.31974°	80.0387°
`id_length`	0.00261	0.00987
`id_width`	0.00043	0.00814

3.2 Sensor Response

The parameters of the Lorentzian, describing the sensor peak, can be derived from so-called sensor scans. They show the sensor response as a function of the voltage applied at the plunger gate of the corresponding SD, SP1, or SP2. As the plunger gate has a direct influence on the electrochemical potential of the SD μ_{sens}, the parameters S_{off}, a, γ and μ_0 can principally be derived by fitting the Lorentzian (see Equation (2.16)) to the experimental sensor scan. Then, α_{dot} and α_{gate} (see Section 2.5.1) can be determined from a measured CSD using the fitted Lorentzian. As the shape of the sensor peak depends on the voltage configuration of the DQD and SD barrier gates, a corresponding sensor scan with matching voltage configurations is required for each CSD. Moreover, to avoid different external influences on the measurements, the sensor scan should be done shortly before the CSD measurement.
After the parameters of the Lorentzian are determined, the inverse function of the fitted sensor curve can be used to calculate the electrochemical potential of the SD for the experimental CSD. However, as the Lorentzian is not a monotone function, a true inverse

does not exist. Instead, only the left side of the curve is used. This is possible because of the symmetry of the Lorentzian used. The calculated electrochemical potential at the sensor is then used to calculate α_{dot} and α_{gate}. For this, the absolute value of the gradient of the electrochemical potential can be used because the sign of both parameters is known. In this case, the gradients in the horizontal and vertical direction are used to calculate $\alpha_{dot,1}/\alpha_{gate,1}$ and $\alpha_{dot,2}/\alpha_{gate,2}$, respectively. The median of the absolute gradient values in regions without edges approximates α_{gate}. Using the median instead of the mean ensures higher robustness to outliers which are caused by noise. After that, $\sum_{i=1}^{2} \alpha_{gate,i} \cdot v_{gate,i}$ is subtracted from the electrochemical potential at the sensor, to compensate for the effect of α_{gate} for a better approximation of α_{dot}. α_{dot} is then calculated as the negative value of the median of the highest 5% of gradient values inside the regions marked as edges.

To be able to determine the regions with edges, masks are generated manually, as shown in Figure 3.4. As $\alpha_{dot,1}$ determines the edge strength in the horizontal direction and $\alpha_{dot,2}$ in the vertical direction, the edges in both directions are marked with different labels to avoid using the wrong edge for $\alpha_{dot,1}$ or $\alpha_{dot,2}$.



(a) CSD

(b) Mask

Figure 3.4: Example of a mask used to mark edges to determine the parameters of the sensor response

The problem with this approach is that the value ranges of the CSDs do not match the value ranges of the corresponding sensor scans in most cases. This is because the CSDs are transformed linearly during the measurement. As these postprocessing steps are not reversible, the fitted sensor responses are also transformed to match the value range of the CSD. This is done by calculating the inverted Lorentzian of the CSD in non-edge regions and fitting the offset S_{off} and the scaling factor a by minimizing the sum of the squared standard deviation of its gradients in the horizontal and vertical direction. The standard

deviations of the gradients should be low because the value of α_{gate} should be constant throughout the entire CSD. Without any normalization of the inverted Lorentzian, the minimization would choose infinitely high scaling factors. This is the case because the corresponding inverted values are very close to each other, leading to gradient values near zero with a low standard deviation. To avoid this, the inverted Lorentzian is applied to the CSD, the result is scaled to values between zero and one, and the gradient of the normalized inverted Lorentzian is used for the computation of the error function. To minimize the optimization function, `SciPy`'s minimization function is used [40]. Per default, it uses a BFGS[1] [41] method for the minimization.
The parameters for the sensor response are determined in 16 CSDs and their corresponding sensor scans. Table 3.2 lists the determined ranges.

Table 3.2: Parameter ranges of the sensor response determined from 16 experimental CSDs and their corresponding sensor scans

parameter	minimum	maximum
S_{off}	-2.06246	-0.08386
μ_0	-0.12168	-0.03824
a	0.02251	1.9362
γ	0.0009636	0.0029509
$\alpha_{gate,1}$	0.00622	0.15057
$\alpha_{gate,2}$	0.00675	0.19451
$\alpha_{dot,1}$	-0.0005532	-0.0000318
$\alpha_{dot,2}$	-0.0004404	-0.0000151

For the simulation with these parameters, the ratio between α_{gate} and γ is also important because for very high ratios, the peak of the sensor curve only occupies a very small region in the CSD (see Figure 3.5). In this case, the peak of the sensor curve is visible inside a CSD as a diagonal stripe with high values, which can be interrupted by the lead transitions. α_{gate} should be less than $100 \cdot \gamma$ for the simulated CSDs as this is the magnitude that can be observed in the experimental CSDs.

[1] developed by Broyden, Fletcher, Goldfarb, and Shanno

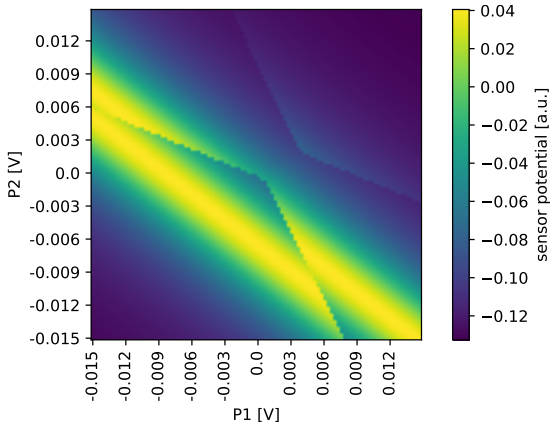


Figure 3.5: Simulated CSD with a ratio between α_{gate} and γ of more than 150 with $\alpha_{gate,1} = 0.15057$, $\alpha_{gate,2} = 0.19451$, and $\gamma = 0.0009636$

3.3 Pink and White Noise

The strengths of both pink and white noise can be determined with the help of the PSD of the noise in the experimental CSDs. The PSD is computed with the help of Welch's method [42] implemented in `SciPy` [43]. It functions by dividing the data into overlapping segments, computing a modified periodogram for each segment, and averaging these. The modified periodogram uses a special window to minimize artifacts in the Fourier transform of the auto-correlation function of the signal. To compute the PSD, the two-dimensional image is flattened, meaning that the rows of the image are concatenated to a one-dimensional array. This brings the pixels in the order they were measured.
The power spectral densities for white and pink noise are given by

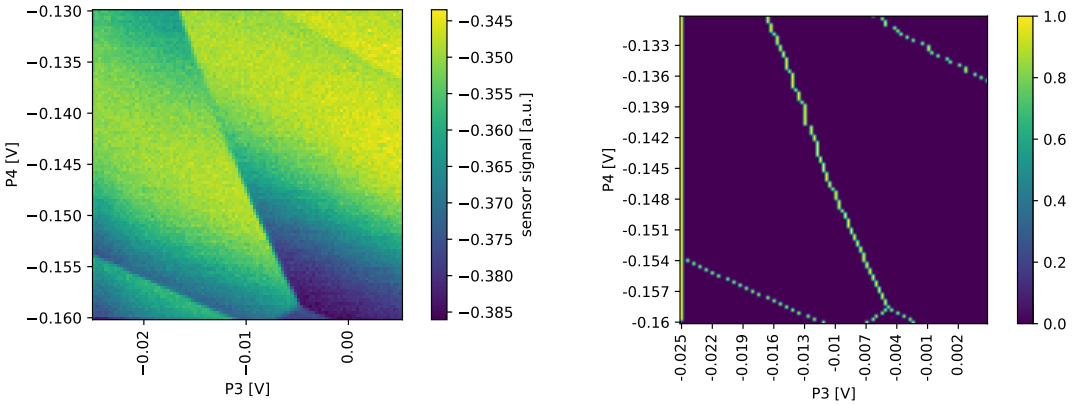
$$\begin{aligned} PSD_{white} &= c_{white} \cdot \sigma_{white}, \\ PSD_{pink} &= \frac{c_{pink}}{f} \cdot \sigma_{pink}, \end{aligned} \tag{3.2}$$

where $c_{white} \approx 2$ and $c_{pink} \approx 0.1$. These constants are determined by generating white, respectively, pink noise with strengths between 0.00001 and 0.1 and computing its estimated PSD with Welch's method. Then, the formulas given in Equation (3.2) are fitted to the estimated PSDs of the generated noise by minimizing the sum of squared errors with the help of `SciPy`'s minimization function [40]. The median of the minimization results is taken as an approximation for c_{white} and c_{pink}.
The values for σ_{white} and σ_{pink} can be determined similarly. If both types of noise are

present, the following equation holds:

$$PSD_{white+pink} = c_{white} \cdot \sigma_{white} + \frac{c_{pink}}{f} \cdot \sigma_{pink}. \tag{3.3}$$

However, this cannot be directly applied to the experimental CSDs because the honeycomb pattern of the CSD influences the shape of the PSD. That means the noise inside the experimental CSDs has to be approximated first. For images without sharp edges, this can be done by applying a Gaussian filter and a spline interpolation to the image and subtracting the smoothed image from the original one [16]. However, for CSDs with distinct edges, this leads to an overestimation of the noise at edges. A solution to this is to remove the edges from the signal. As edge detection in CSDs is still an ongoing task, the edges have to be marked manually. This is done with a mask as shown in Figure 3.6. In addition to the edges, every first pixel of a row is marked as an edge because the concatenation of the rows leads to edges at the concatenation points.



(a) CSD

(b) Edge mask

Figure 3.6: Example of a mask used to mark edges to determine the strengths of white and pink noise

To remove the edges, the gradient of the CSD is approximated b backward differences. Then, the edges can be removed by adding the backward difference of the edge pixel to its value. As edges are often wider than only one pixel, the same is done for the neighboring pixels of the marked edge. Figure 3.7 shows an example of the edge removal in a line of a CSD. For the edges, which are created at the beginning of each line by flattening the image, the neighboring pixels are not affected by the edge removal.

The lowest frequencies in the PSD might contain other types of noise like RTN and also the signal itself might affect these frequencies. Therefore, the lowest 15 frequencies are excluded for the determination of σ_{white} and σ_{pink}. Per default Welch's method computes the estimated PSD for frequencies between 0Hz and 0.5HZ in steps of $\frac{1}{256}$. Leaving out the 15 lowest frequencies leads to a frequency range of 0.05859375Hz to 0.5Hz considered for the estimation of the noise strength.

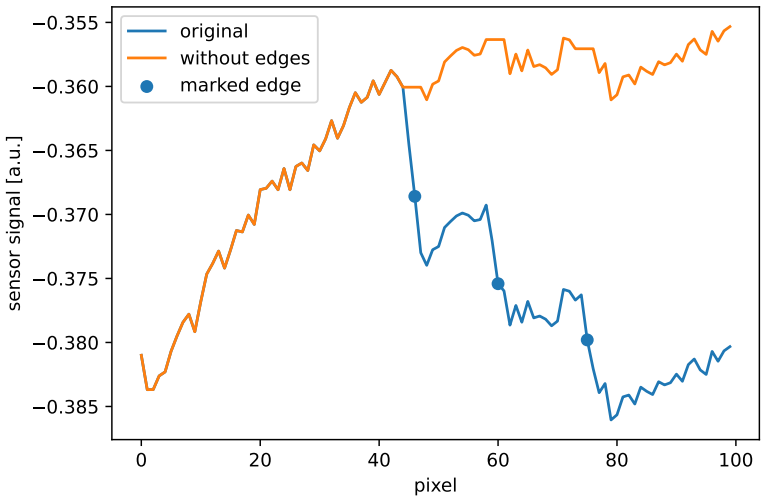


Figure 3.7: Example of a line of a CSD where the edges are removed

To test the accuracy of this estimation method, the noise strengths are estimated for a simulated image with different combinations of white and pink noise. The three different cases in Table 3.3 are considered. For the simulation of the noiseless data, a representative set of parameters is chosen for the structure of the honeycombs and the sensor response. The resulting image is shown in Figure 3.8.

Table 3.3: Cases analyzed for the accuracy of the estimation of pink and white noise in CSDs with clear edges

evaluation case	σ_{white}	σ_{pink}
$\sigma_{white} = \sigma_{pink}$	0.001	0.001
$\sigma_{white} < \sigma_{pink}$	0.0001	0.001
$\sigma_{white} > \sigma_{pink}$	0.001	0.0001

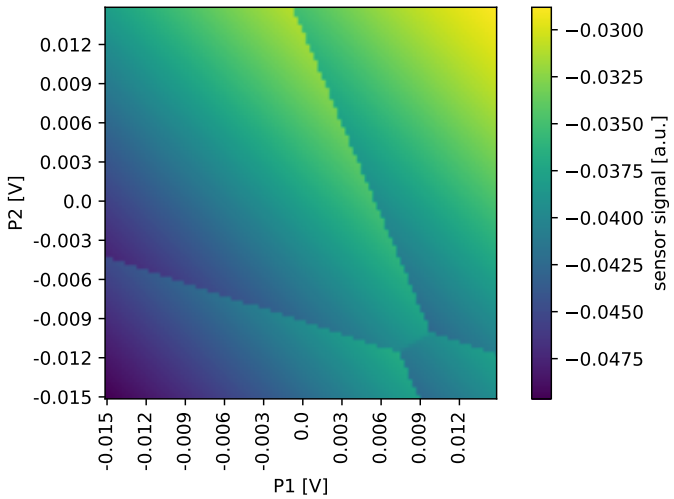
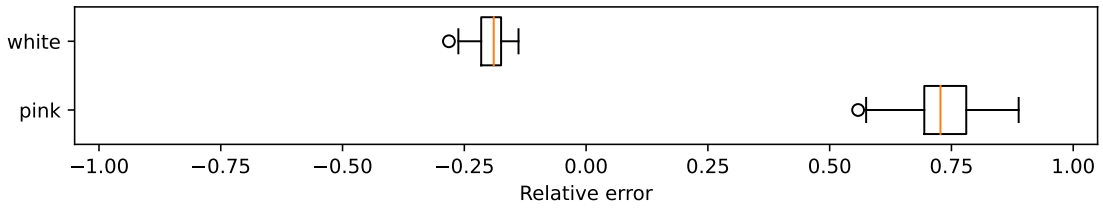


Figure 3.8: Simulated noiseless test image with strong edges

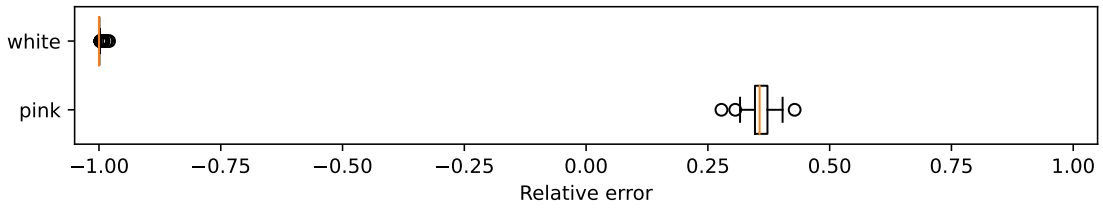
The resulting relative noise strength errors derived from 100 simulations each are shown in Figure 3.9. The relative error is calculated as

$$\frac{\sigma_e - \sigma_{gt}}{\sigma_{gt}},$$

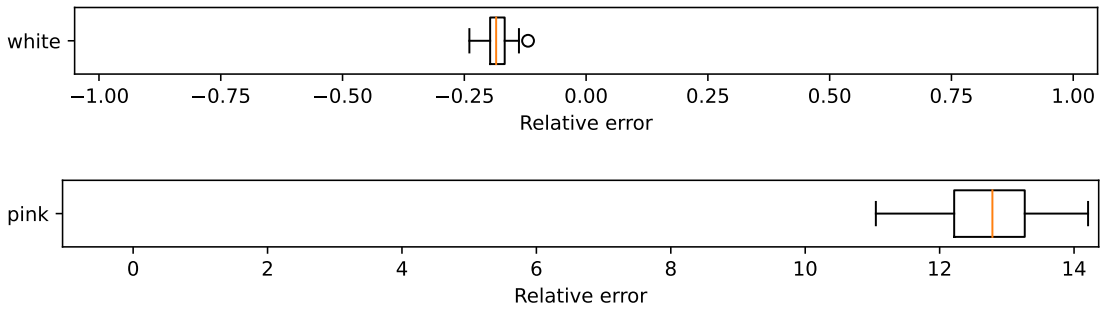
where σ_e is the estimated noise strength and σ_{gt} is the ground truth. For a similar amount of white and pink noise, the estimation seems to work well. However, white noise is slightly underestimated, while pink noise is overestimated. This behavior is even stronger in the other cases. For lower white noise, the method often estimates no white noise, while for lower pink noise, the estimated pink noise strength is more than ten times higher than the ground truth. This is mainly because the noise cannot be approximated perfectly, which leads to higher PSD values, especially for low frequencies.
In conclusion, the estimated noise strength does not perfectly match the actual noise strength but the estimate is usable to get a first idea of the amount of noise typically present in the CSDs. For the simulation, it has to be kept in mind that pink noise is often overestimated and white noise is underestimated for this noise estimation method.



(a) $\sigma_{white} = \sigma_{pink}$



(b) $\sigma_{white} < \sigma_{pink}$



(c) $\sigma_{white} > \sigma_{pink}$

Figure 3.9: Boxplots of relative errors between the estimation and real noise strength for different ratios of white and pink noise in a simulated image with strong edges.

In addition to the CSDs with distinct edges, CSDs without edges or with only weak edges are analyzed. For these CSDs, a two-dimensional Gaussian kernel and bivariate spline interpolation are executed to approximate the noiseless signal. The noise is estimated by subtracting the noiseless approximation from the original image. To test the accuracy of this method, a simulated CSD with an α_{dot} ten times lower than the one in Figure 3.8 is used. The noiseless image is shown in Figure 3.10. The added noise has the same strength as in the investigations for strong edges.

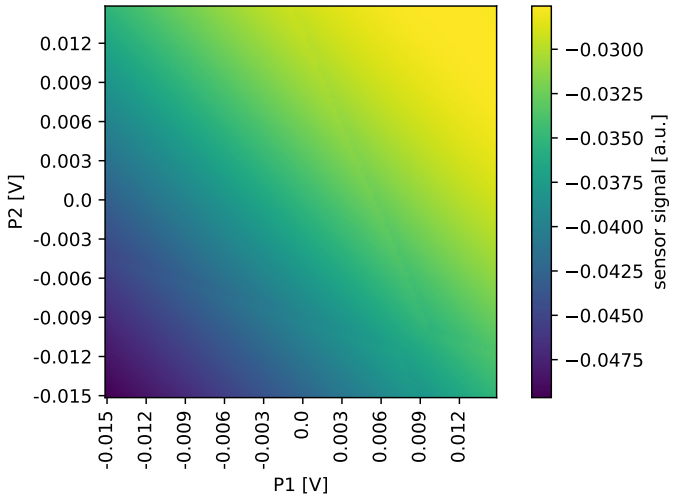
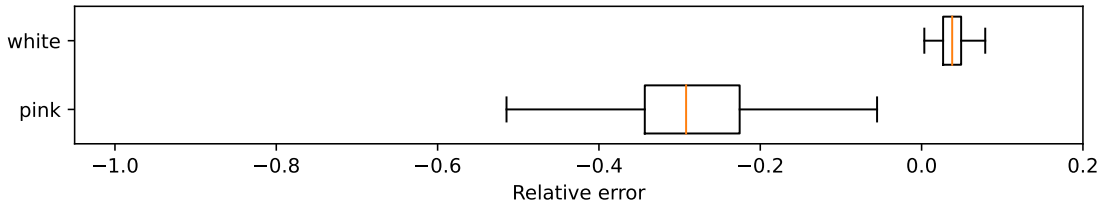
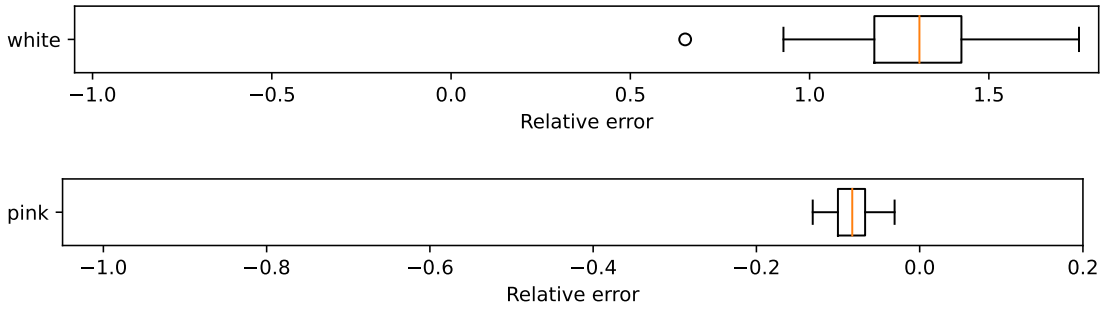


Figure 3.10: Simulated noiseless test image with low edge strength

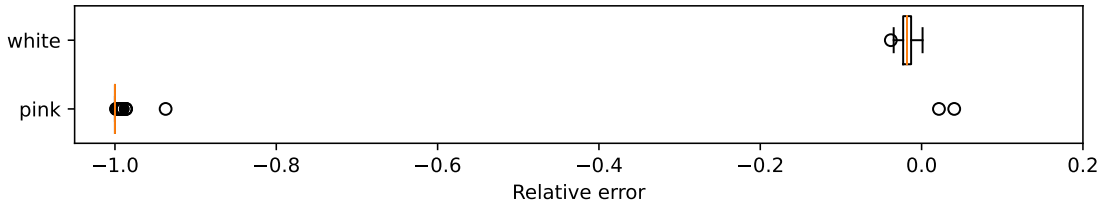
The resulting relative noise strength errors derived from 100 simulations each are shown in Figure 3.11. Contrary to the method used for strong edges, this method tends to underestimate pink noise in all cases and overestimate white noise in most cases (see Figure 3.11). However, for stronger pink noise, also white noise is underestimated slightly. The dominant type of noise is estimated quite well.
In conclusion, this type of estimation has a higher accuracy than the one used for CSDs with edges as the relative errors deviate less. Thus, it is suitable to get an insight into typical noise strengths in CSDs.
The strengths of white and pink noise are determined in 20 CSDs with visible edges and 20 CSDs without clearly visible edges. This results in the ranges shown in Table 3.4. For the results in this table, an outlier with a value of 0.0222391, determined in an image without edges, was removed. In this case, the high value comes from some frequencies with an overproportional high power in comparison to the other frequencies which is probably caused by an environmental influence.



(a) $\sigma_{white} = \sigma_{pink}$



(b) $\sigma_{white} < \sigma_{pink}$



(c) $\sigma_{white} > \sigma_{pink}$

Figure 3.11: Boxplots of relative errors between estimation and real noise strength for different ratios of white and pink noise in a simulated image with weak edges. For all three analyzed cases the errors for white and pink noise estimation are shown separately.

Table 3.4: Estimated ranges for the strengths of white and pink noise. For pink noise determined in images without edges, an outlier with a value of 0.0222391 is removed as the characteristic of noise in this image does not fit the other images.

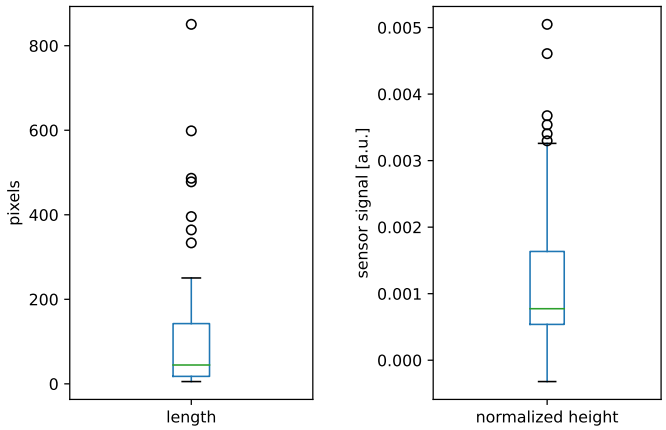
CSD type	parameter	minimum	maximum
with edges	σ_{white}	0.0	0.0007722
	σ_{pink}	0.0009544	0.0101090
without edges	σ_{white}	0.0000338	0.0053942
	σ_{pink}	0.0	0.0049787
total	σ_{white}	0.0	0.0053942
	σ_{pink}	0.0	0.010109

For the dataset without edges, the relation between white and pink noise is balanced, whereas pin noise dominates for the dataset with edges. This is most likely an effect of the edge removal algorithm as the pre-study concerning the accuracy of the noise estimation shows an overestimation of pink noise in all evaluation cases. For data without edges, the pre-study showed that pink noise is underestimated in more cases than white noise. This suggests that even though the strength of white noise is slightly higher in the set of CSDs without edges similar ranges should be used for the simulation of white and pink noise. Moreover, a noise strength of zero is unrealistic because both noise types are present in all electronic devices.

3.4 Random Telegraph Noise

The automatic detection of RTN in CSDs is still an unsolved problem. Therefore, the detection of bursts for the parameter determination of the RTN is done manually. However, the bursts in CSDs often cannot be distinguished from the pink noise. In contrast, bursts in sensor scans can be detected much easier. Because of this, sensor scans are used for the determination of the lengths and heights of the bursts. This is possible because the CSDs are measured with the same sensor used for the sensor scan which leads to similar characteristics of the noise. Of the 1293 available sensor scans, 118 are chosen randomly. In these scans, 21 bursts are manually detected and used for the parameter determination. RTN can be described as a time-dependent stochastic process. To translate the sensor scan burst length into the CSDs domain, the measurement time for one data point in both scans has to be compared. As this time is only known for 768 out of 1230 CSDs, the median of the available values is used for the simulated CSDs. This is a time of 0.00128s for each pixel which is used for 465 CSDs. Moreover, as CSDs are linearly transformed during the measurement (see Section 3.2), the heights of the bursts in the sensor scans have to be transformed similarly. To get a height which is independent of the transformation, the absolute height of each burst is divided by the scaling factor a of the Lorentzian fitted to the sensor scan. This height is called the normalized height of the bursts in the following. The lengths and heights of the bursts are extracted with the help of a `canvas.mpl_connect` function which is added to the `Matplotlib` figure displaying the sensor scan. If a point in the diagram is selected by a left click, this function saves the coordinates of the clicked point. For every burst, three points have to be selected to calculate the length of a burst in the pixel space and the height of the burst as the difference in the sensor response. After that, the length is translated into the time-domain by multiplying it by the time used for measuring each pixel. Finally, the length is translated back into the CSD pixel domain by division by the median of the measuring time per pixel in the CSDs. The normalized

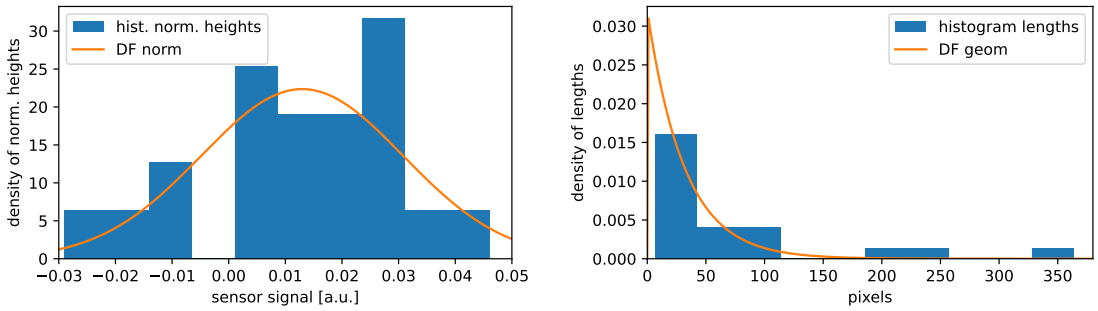
height is applied to the simulated CSD by multiplying it with the scaling factor a used for the simulated sensor response.
In this way, the parameter ranges shown in Figure 3.12 are extracted from the experimental CSDs.



(a) Lengths of the bursts (b) Heights of the bursts

Figure 3.12: Boxplots of the translated lengths and heights of the bursts found in the sensor scans

The approximated density function of the heights and lengths is calculated with the help of the `Matplotlib` `hist` function. It fits the density function of a normal distribution (see Figure 3.13a), respectively, geometric distribution (see Figure 3.13b). This indicates that the assumption in Section 2.5.4 is correct.



(a) Height (b) Length

Figure 3.13: Comparison of the approximated and theoretical probability density of the (a) normally distributed heights of the bursts and (b) geometrically distributed lengths of the bursts

The parameters of this type of noise are determined with the help of the mean of the lengths, as well as the mean and empirical standard deviation of the heights, shown in Table 3.5.

Table 3.5: Parameters for the RTN determined from experimental CSDs

parameter	determined value
expected length	74.56704
expected height	0.01297
standard deviation of height	0.01784

3.5 Dot Jumps

As for the RTN, there is no method for detecting the dot jumps automatically yet. In the available data, there are six CSDs with shifted edges that could belong to a dot jump. However, no jump back is visible inside these CSDs. Hence, only the strength of the jumps can be extracted but not the length. Moreover, in the horizontal direction, only one jump could be observed. This indicates that for the used sample, it is more likely that the charge traps affect the second plunger gate.
The determined jump strengths are between 4 and 8 pixels with a mean of 6 pixels. As the jumps back are not visible inside the CSDs, an expectation of 100 pixels for the length of the jumps is assumed for the simulation. This is the size of most CSDs. This leads to the following parameters which can be used for the simulation of CSDs.

Table 3.6: Parameters for the dot jumps determined from the experimental CSDs

parameter	determined value
expected length	100
expected strength	6

4 Generation and Evaluation of Simulated Data

In this chapter, the evaluation of the simulated data is described and the results are presented. In the beginning, the generation of a simulated dataset of CSDs is explained. The dataset should be evaluated concerning the fidelity and diversity of the generated samples. This is done by comparing it with an experimental dataset. However, not all experimental CSDs are usable for that, as explained in Section 4.2.
To evaluate the authenticity of the simulated data, different approaches are presented in Section 4.3 and Section 4.4.
As possibly that not all parameters of the simulated CSDs are set correctly, the evaluation is split into two steps. This is the case because the different types of distortions might influence each other which leads to a bias in the parameters determined from the experimental CSDs.
For each step, a separate set of experimental data is used for the comparison. First, the simulated dataset is evaluated and some parameters are adjusted. Then, the second set of experimental data is used for the final evaluation of the data simulated with the adjusted parameters.

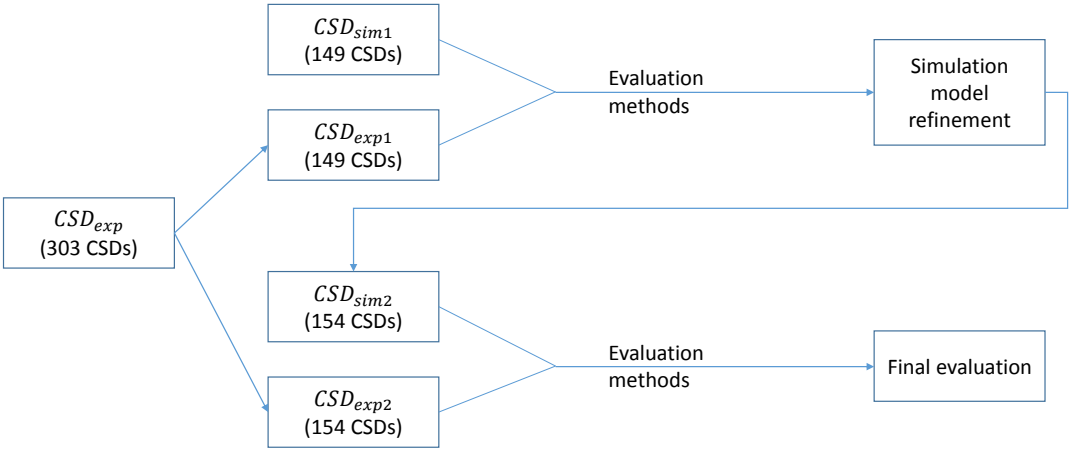


Figure 4.1: Flow chart of the evaluation procedure

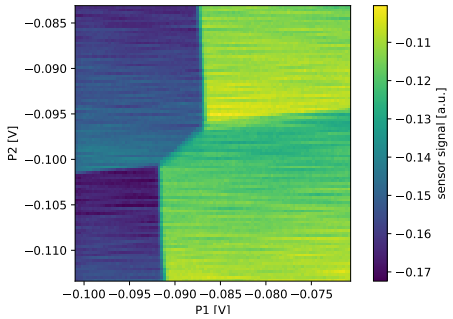
In Figure 4.1, the procedure of the evaluation is outlined. The available experimental data CSD_{exp} is divided into two datasets CSD_{exp1} and CSD_{exp2}, also referred to as the optimization and evaluation dataset. For the optimization of the simulation parameters and the refinement of the simulation model, the first simulated dataset CSD_{sim1} is compared with CSD_{exp1}. After that, a new optimized simulated dataset CSD_{sim2} is generated and compared with CSD_{exp2} for the final evaluation of the simulation.

4.1 Generation of Simulated Charge Stability Diagrams

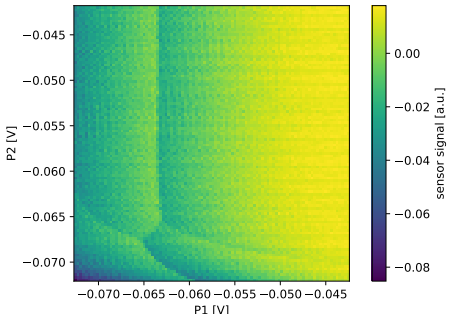
The generation of simulated CSDs consists of the simulation of the clean occupation data and the addition of the distortions described in Section 2.5. For the generation of the simulated CSDs, uniform random samples of the determined parameter ranges of Chapter 3 are used. However, it also has to be checked if the relation between the sampled parameters is consistent with the experimental data. For the clean occupation data, this includes the calculation of the angle between the simulated lead transitions and comparison to the `angle_lead` values determined from the experimental CSDs. Furthermore, the `id_width` is related to the `id_length` because more curved triple points increase the distance between them at the inter-dot transition. The relation between the `id_length` and `id_width` lies between 0.86922 and 9.05538 for the experimental data. If the sampled parameters do not fulfill this, a new sample is drawn.
Moreover, `length1` and `length2` are sampled from the minimum to two times the maximum both given in Table 3.1. This is done because the lengths of the lead transitions are bigger than the determined lengths due to the restricted measurement area of the CSD.
After clean occupation data have been simulated, distortions are added. Their parameters are also sampled randomly and verified for consistency with the experimental data. For sensor response parameters, the relation between α_{gate} and γ should be below 100. Moreover, for the strength of always present white and pink noise, a minimum of $1 \cdot 10^{-6}$ is assumed instead of the determined minimum of zero. The maximum for both noise types is set to 0.006. RTN is only applied for $\frac{1}{6}$ of the CSDs as this is approximately the percentage of experimental CSDs which contain RTN. Also, dot jumps in the vertical and horizontal direction are only applied for $\frac{1}{10}$, respectively, $\frac{1}{60}$ of the CSDs. The starting point for the sensor response μ_{off} is sampled from the interval $[\mu_0 - 2\gamma, \mu_0 - \gamma]$. This region is taken because it has a steep slope which is one of the factors included in the selection of the sensor peak in the experiment.

4.2 Selection of Usable Experimental Charge Stability Diagrams

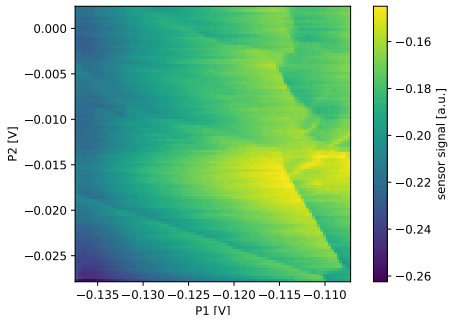
In the experimental CSDs, there are still some effects present, which cannot be simulated yet. These are CSDs that were measured with the help of virtual gates, leading to nearly horizontal respective vertical edges with high contrast and very low cross-coupling between the sensor and the virtual plunger gates (see Figure 4.2a). Moreover, physical effects like movements of the DQD and parasitic dots lead to curved lead transitions and irregular line patterns (see Figure 4.2b and Figure 4.2c). Additionally, unknown effects lead to CSDs that do not fit the current simulation model (see Figure 4.2d and Figure 4.2e). Thus, the corresponding experimental CSDs are removed before the comparison of both datasets.



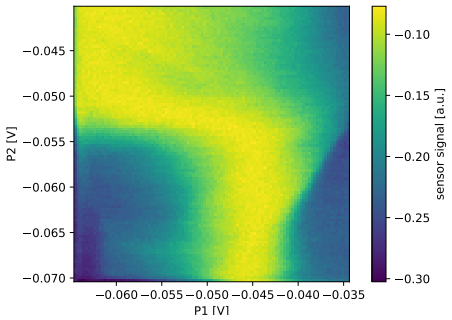
(a) Virtual gates



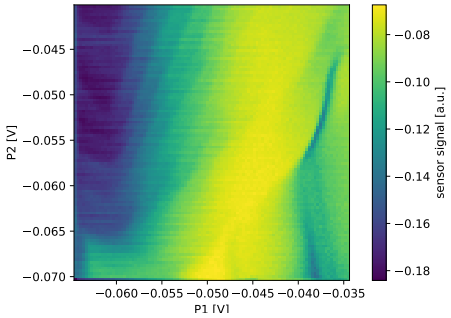
(b) Curved lead transitions



(c) Parasitic dot



(d) Irregular sensor response



(e) J-shaped lines

Figure 4.2: Experimental CSDs with effects that are not yet simulatable

Then, the remaining 303 CSDs are split into two datasets. As both the model refinement and the final evaluation are equally important, a similar amount of CSDs is used for both datasets. To represent run-specific effects in both datasets, the data in every run is equally distributed to the optimization CSD_{exp1} and evaluation dataset CSD_{exp2}. For an uneven number of samples in one run, the remaining sample is added to CSD_{exp2}. This leads to 149 CSDs in CSD_{exp1} and 154 in CSD_{exp2}.
In every experimental data point, there is one CSD measured with the left and one measured with the right SD. Due to the higher signal-to-noise ratio, only the CSD is used that is measured with the SD closer to the DQD to be tuned.

4.3 Metrics for Generative Models

The problem of comparing the distributions of generated and real data also appears in the context of machine learning when evaluating generative models. However, many of the used evaluation metrics, e.g., Inception Score [44] and Fréchet Inception Distance [45], are not applicable here because a classification network has to be trained to apply them or they rely on neural networks that are pretrained on natural images. Moreover, it is beneficial to compute the metric sample-wise because this allows for analyzing the flaws of the simulation model. Therefore, the α-Precision and β-Recall metrics are used to measure the fidelity and diversity of the generated datasets [46]. Both metrics are bound between zero and one with the aim to maximize them. In [46], a third metric is introduced that indicates if a generative model tends to only copy the training data. However, this cannot happen for the simulations described in this thesis as the experimental data is only used to determine parameter ranges. The chance that the same CSDs are generated by sampling parameters from these ranges is very low. Hence, the third metric is not crucial for the evaluation. To compute α-Precision and β-Recall, the data are first embedded into a lower dimensional hypersphere. Therefore, a neural network is trained on the first set of experimental data. It is trained to minimize the loss

$$L = R^2 + \frac{1}{\nu n} \sum_{i=1}^{n} \max\{0, \|\phi(x_i) - c\|^2 - R^2\}, \tag{4.1}$$

where R is the radius of the hypersphere, c is the center, ν is a balancing factor, and n is the number of data points. ϕ is the neural network that transforms the CSDs x_i into the lower dimensional hypersphere. This loss function is also used to train neural networks for anomaly detection [47]. A framework for training neural networks accordingly is available on Github [48]. The architecture used for the embedding of CSDs is an adaptation of the `MNIST_LeNet` implemented in [48]. The architecture is visualized in Figure 4.3. It

is selected because it was also applied in [47] for image data.

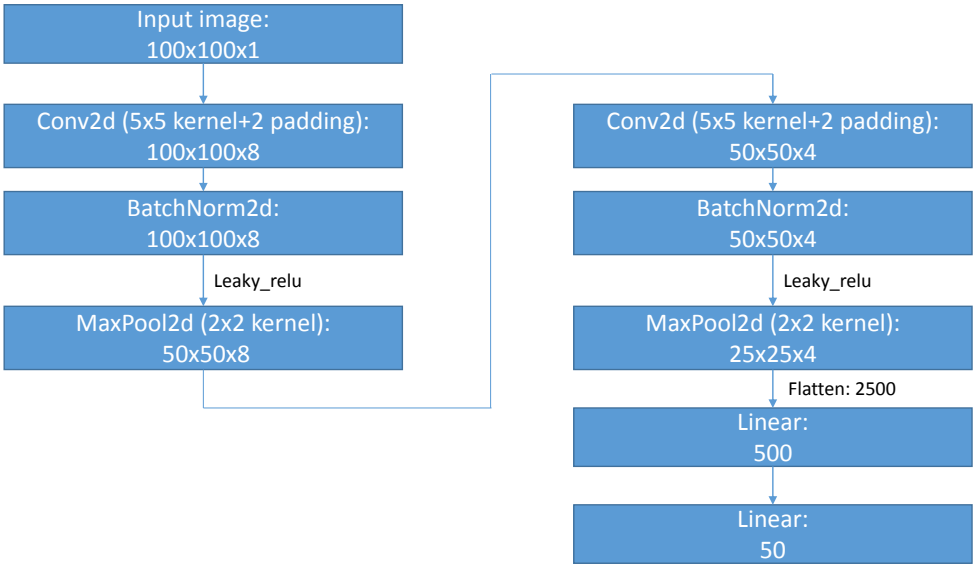


Figure 4.3: Architecture of the neural network used to transform the CSDs into a lower dimensional hypersphere

Both α-Precision and β-Recall are calculated as the mean of sample-wise classifiers. The precision classifier verifies if the generated input sample exists in the α-support of the real data, while the recall classifier checks if the real input sample exists in the β-support of the generated data. The α-support refers to the support of the distribution of the α most typical samples. For the calculation of α-Precision and β-Recall, the α- and β-supports have to be estimated. For the real data, the estimated α-support is

$$\hat{S}_r^\alpha = B(c, \hat{r}_\alpha), \tag{4.2}$$

where $\hat{r}_\alpha$ is the α-quantile of $\|\phi(x_{r,i}) - c\|$ for the real data $x_{r,i}$ and $B(c,r)$ is a Euclidean ball with center c and radius r. Thus, the precision classifier is

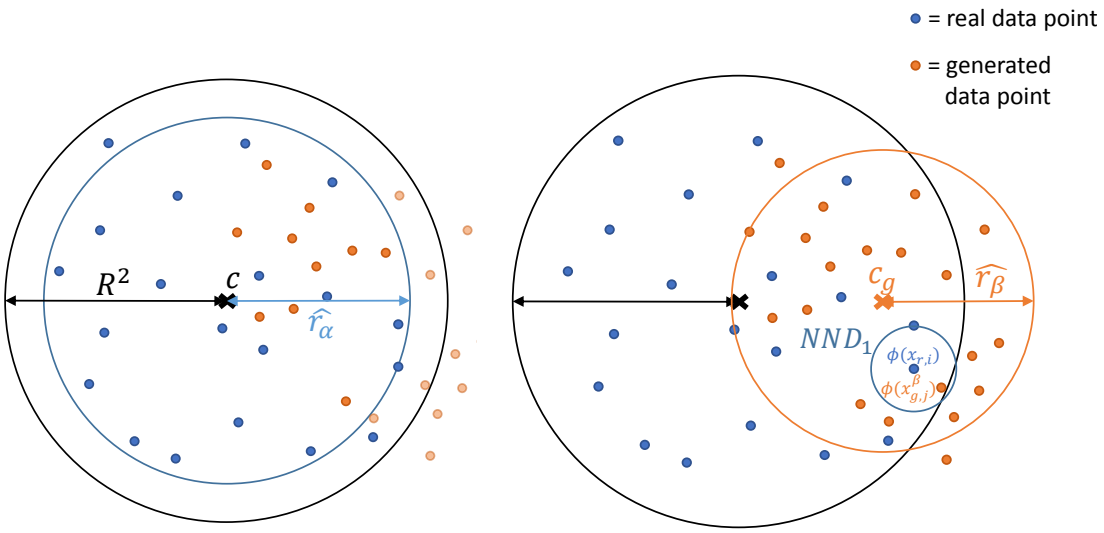
$$f_P(\phi(x_{g,i})) = \mathbf{1}\{\phi(x_{g,i}) \in B(c, \hat{r}_\alpha)\}. \tag{4.3}$$

In Figure 4.4a, the α-support of the real data are visualized in blue. For this example, half of the simulated data, displayed in orange, are inside the α-support region, while the other half, which is shown in a paler color, is outside the support region. That means the α-Precision would have a value of 0.5 for this example. In general, α-Precision is the fraction of synthetic samples that resemble the most typical fraction of real samples [46]. For the recall classifier, the Euclidean ball $B(c_g, \hat{r}_\beta)$ with $c_g = \frac{1}{m}\sum_{i=1}^{m}\phi(x_{g,i})$ and $\hat{r}_\beta = Q_\beta(\|\phi(x_g, i) - c_g\|)$, with Q_β indicating the β-quantile, has to be calculated. However, as ϕ is trained on the real data, it does not necessarily transform the support of the generated data into a hypersphere. Therefore $B(c_g, \hat{r}_\beta)$ cannot directly be used as an estimate for the

β-support of the generated data. Instead, the recall classifier is constructed as

$$f_R(\phi(x_{r,i})) = \mathbf{1}\{\phi(x^{\beta}_{g,j}) \in B(\phi(x_{r,i}), NND_k(\phi(x_{r,i})))\}, \tag{4.4}$$

where $\phi(x^{\beta}_{g,j})$ is the sample in $B(c_g, \hat{r}_\beta)$ nearest to $\phi(x_{r,i})$ and $NND_k(\phi(x_{r,i})))$ is the distance between $\phi(x_{r,i})$ and its k-nearest neighbor in the real dataset. In Figure 4.4b, $B(c_g, \hat{r}_\beta)$ is visualized and an example of a 1-nearest-neighbor region including a generated data point is shown. In contrast to the α-Precision, the β-Recall indicates the fraction of real samples covered by the most typical fraction of synthetic samples [46].



(a) α-Precision (b) β-Recall

Figure 4.4: Visualization of the items needed to calculate the α-Precision and β-Recall. (a) shows the hypersphere itself and the α-support for $\alpha = 0.9$. (b) shows the β-support for $\beta = 0.9$ and the 1-nearest-neighbor region for a real data point $\phi(x_{r,i})$, containing a simulated data point $\phi(x_{g,j})$ from the β-support.

4.3.1 Preprocessing

As the CSDs have different value ranges, the neural network ϕ might try to transform the data only considering the values while the structure of the images is not taken into account. To prevent this, the images are all transformed beforehand to mean zero and variance one.

Moreover, the applied voltages are not included in the training procedure and the data have to be resized to the same shape. As most CSDs have a size of 100x100 pixels, this size is used.

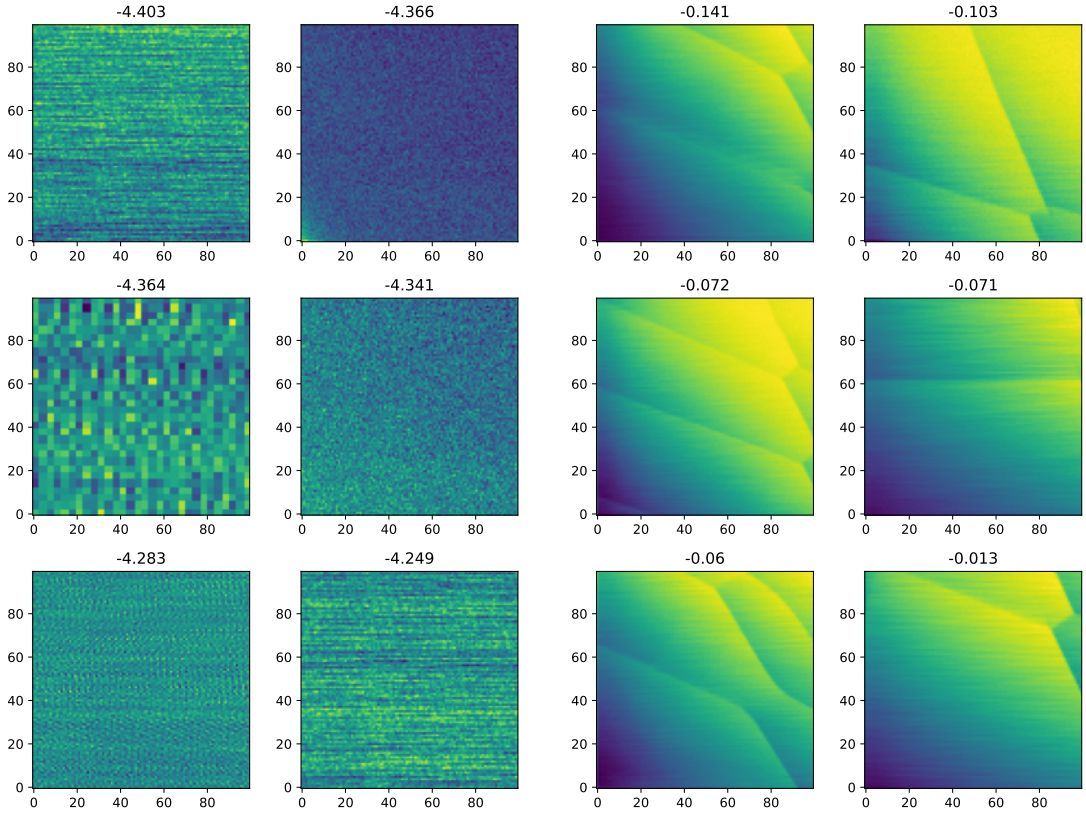
4.3.2 Hyperparameter Selection

Hyperparameters that have to be selected to compute the α-Precision and β-Recall metrics are:

- The balancing factor ν in the loss function used for the training of ϕ. It balances the compactness of the hypersphere, measured by R^2, with the distance of the outer samples to the border of the hypersphere. Here, ν is set to 0.01 as proposed in [46].
- The optimizer. Here, the Adam optimizer is used, which is well known for its effectiveness in solving minimization problems, especially in the deep learning context.
- The learning rate for the training. The default value of 0.001 set in the deep SVDD project [48] is used. This value is small enough to prevent the network from missing the minimum and thus underfitting. On the other hand, the risk for overfitting is lowered by adjusting the number of epochs.
- The batch size for the training. As there are not many CSDs available, this is set to the size of the whole dataset. Otherwise, the network would not be able to capture all the different structures in one batch and thus tend to underfit. In general, a smaller batch size leads to less precision in the approximation of the gradient. This causes the network to need more epochs to get to the minimum or to not reach it at all.
- The parameter for the weight decay regularizer used in [48]. The default value of $1 \cdot 10^{-6}$ is sufficiently small to prevent underfitting.
- The number of epochs for the the training of ϕ. This value was set to 200 because then the loss on the evaluation set is the lowest still similar to the training loss. This is important because otherwise, the hypersphere of the evaluation data would be significantly larger than the hypersphere of the optimization data, which is a sign of overfitting. Moreover, similar images have a low or high distance to the center of the hypersphere for the optimization and evaluation set (see Figures 4.5 and 4.6). In terms of outlier detection, the high scores given in Figures 4.5 and 4.6 indicate that the CSD is more likely to be an outlier.
- The parameter α of the α-Precision. It is set to 95% assuming that 5% of the experimental dataset are outliers. This is most likely an overestimation of the percentage of outliers because most CSD structures which could be regarded as outliers have already been removed from the dataset. However, as a lower α could only lead to a lower value for the α-Precision, the results should give a good insight into how similar experimental and simulated data are.
- The parameter k for the calculation of the k-nearest-neighbor distances in the β-Recall metric. This is set to 2, as for an outlier in the real dataset the 1-nearest-

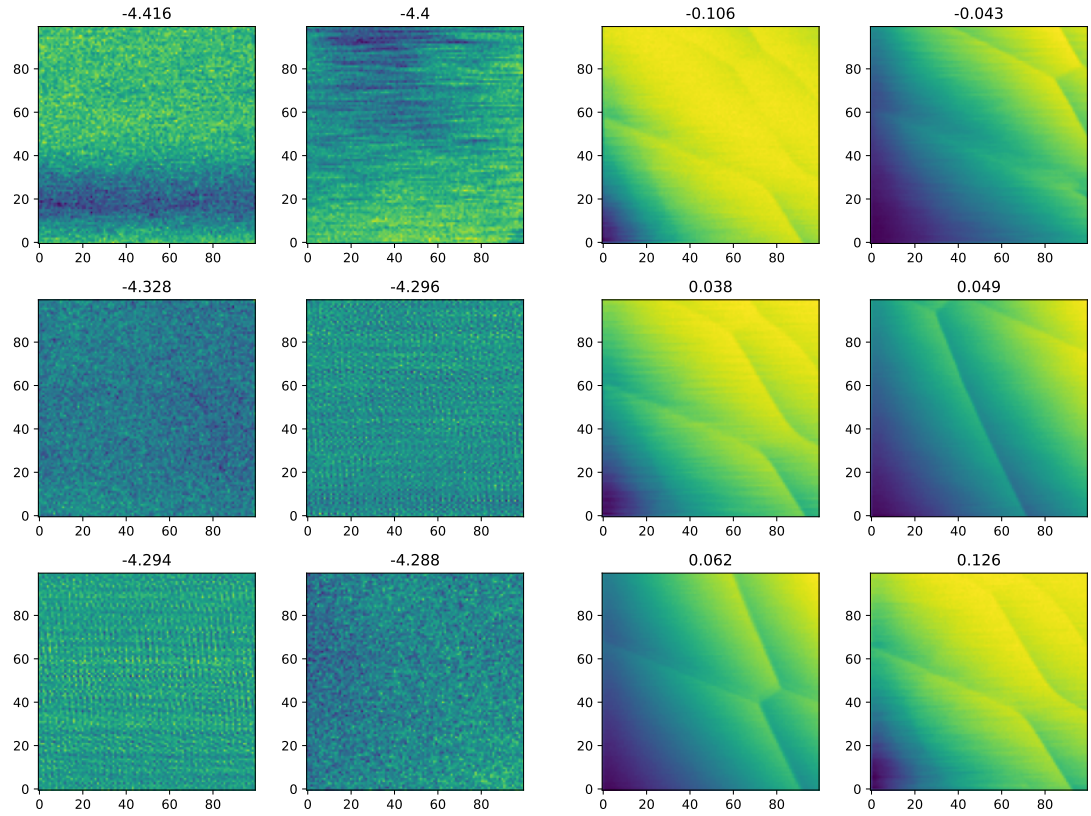
neighbor region could be completely outside the sphere of typical real datasets. Hence, the β-Recall metric would punish the simulation for not including outliers. Taking the value of 2 minimizes the risk that the β-Recall is only low because some outliers of the real dataset are not captured.

- The parameter β of the β-Recall. It is set to 100% because the simulated dataset should not contain any outliers.



(a) CSDs in the center of the hypersphere (b) CSDs at border of the hypersphere

Figure 4.5: Examples of CSDs for different hypersphere regions. (a) CSDs with a small score and (b) CSDs with a high score for CSD_{exp1}. The titles of each diagram contain the score calculated by $\|\phi(x_i) - c\|^2 - R^2$ with $R^2 = 2.184$. The transformation ϕ is calculated excluding any gate voltages. Colorbars are excluded for better clarity.



(a) CSDs in the center of the hypersphere (b) CSDs at the border of the hypersphere

Figure 4.6: Examples of CSDs for different hypersphere regions. (a) CSDs with a small score and (b) CSDs with a high score for CSD_{exp2}. The titles of each diagram contain the score calculated by $\|\phi(x_i) - c\|^2 - R^2$ with $R^2 = 2.184$. The transformation ϕ is calculated excluding any gate voltages. Colorbars are excluded for better clarity.

4.4 Other Evaluation Methods

In addition to the metrics explained in Section 4.3, some other evaluation methods are used to compare the experimental and simulated CSDs. This includes the comparison of the power spectral density, the distribution of the overall estimated noise strength, and the value distribution, as well as the visual comparison of the CSDs.

4.4.1 Power Spectral Density

The PSD provides information about the distribution of the power of the signal onto different frequencies. As explained in Section 3.3, this also provides information about the

types of noise present in the CSDs.
To be able to check if the PSD between the experimental and the simulated data match, the means of the PSDs are calculated and compared. However, this does not capture the variation of the PSD values per frequency, so the minimum and maximum for every frequency are calculated and included in the comparison.

4.4.2 Distribution of Estimated Noise Strength

While the PSD gives information about the composition of the noise, the distribution of the overall noise strength for different images is of interest to compare measured and simulated data.
To calculate the noise strength, the estimator of Chen et al. [49] is used. The author of [16] concluded that this noise estimator is best suited in terms of a low dispersion of the estimated noise strengths. To calculate the noise estimation, the implementation by Zongsheng Yue [50] is applied.
The noise estimator decomposes the CSD into patches and performs an eigenvalue decomposition of them. The lowest eigenvalues belong to noise and thus, can be used to determine the noise level in the image. This idea is also used for the principle component analysis (PCA). To determine the threshold that assigns the eigenvalues to a set belonging to the signal or one belonging to the noise, the mean and median of the lowest eigenvalues are calculated. As long as the mean differs from the median, the highest eigenvalue is removed. If the mean and median are sufficiently similar, the square root of the mean is used as an approximation for the standard deviation of the noise in the image.

4.4.3 Value Distribution

A further aspect of the comparison is the distribution of the values inside the CSDs. This is done visually with the help of a histogram and numerically utilizing the Kullback-Leibler [51] and Jensen-Shannon [52] divergence. Both are used to measure the statistical similarity of two distributions. However, the Kullback-Leibler divergence is not symmetric, which has to be kept in mind for the comparison of results, and sensitive to smaller differences between the distributions. Moreover, the Jensen-Shannon divergence is bound between 0 and $ln(2)$, whereas the Kullback-Leibler divergence does not have an upper bound. The implementation of these metrics is taken from [53].

4.4.4 Visual Inspection

For the visual inspection, the shape of the honeycomb pattern in the simulated data have to be compared with the experimental data. As the main task for this thesis was the simulation of the noise, it also has to be checked against the noise in the experimental data. Moreover, the sharpness of the edges and the sensor response have to be taken into account.

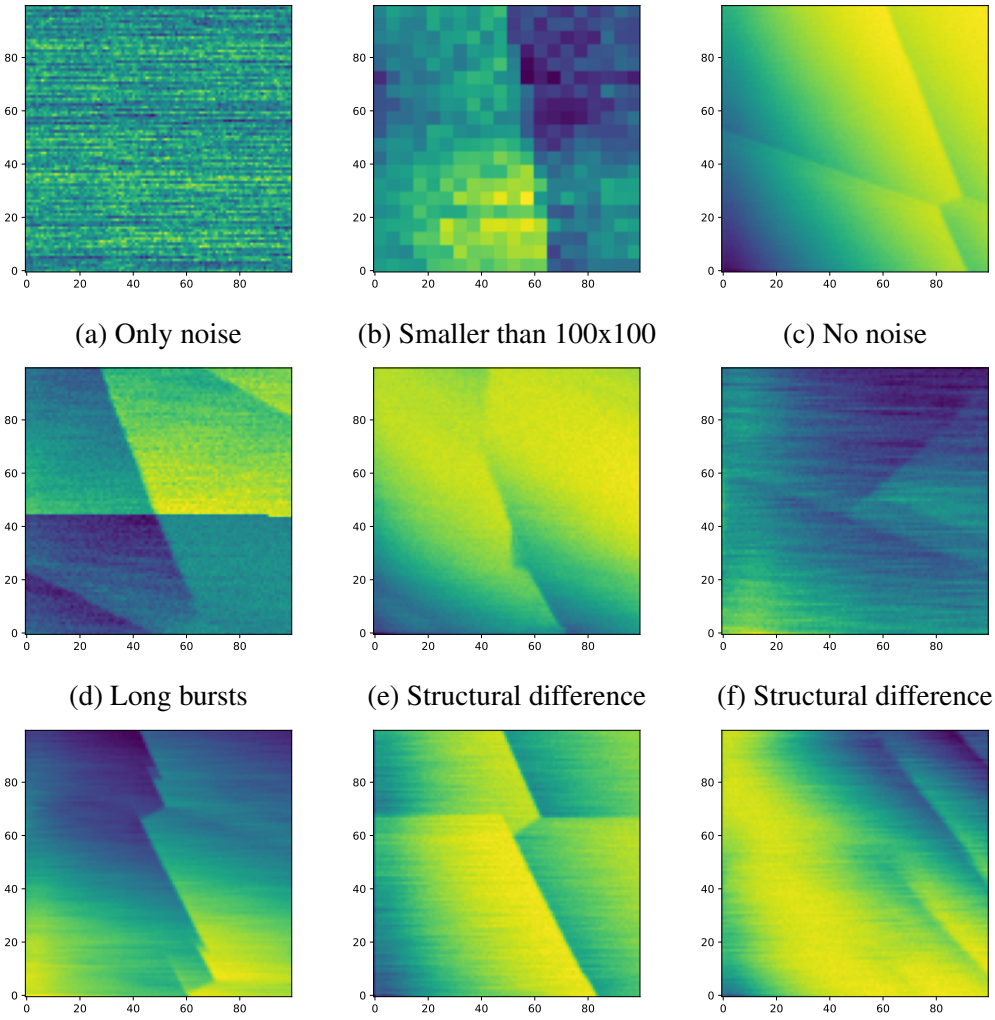
4.5 First comparison and model refinement

For the first assessment, 149 simulated CSDs are generated in the way described in Section 4.1 and compared with the 149 experimental CSDs of CSD_{exp1}. The number of simulated CSDs should match the number of experimental CSDs in the dataset to ensure that a different diversity is not only caused by a different amount of data in the simulated dataset. The results for the different evaluation methods are given in the following sections, including their discussion and adjustments for the optimized simulated dataset CSD_{sim2}.

4.5.1 α-Precision and β-Recall

The value for the α-Precision with $\alpha = 0.95$ is about 0.986 while the value for the β-Recall with $\beta = 1$ is only about 0.128. This means that the fidelity of the simulated data are excellent while their diversity is still low. As the β-Recall is calculated with the help of the 2-nearest-neighbor distance, it does not only compare the support of the simulated and real distribution but also includes a comparison of the distribution of simulated and real data. If the probability distribution of the experimental data is localized in the middle of the hypersphere, while the simulated data is uniformly distributed, there will be many small nearest-neighbor regions in the middle of the hypersphere that do not include a simulated CSD. On the other hand, if the distributions are similar, it is more likely that there will also be a simulated data point in the nearest-neighbor regions of the experimental data points. A similar distribution itself is not as important as a similar support of the distribution because the distribution depends highly on the measurements which were executed by the experimenter. However, the decisions of the experimenter are not necessarily representative for the tuning procedure. In order to check, independently of the distribution, whether the support of simulated and experimental data match, a new

simulated dataset with 1000 samples is created. In this case, the β-Recall is much higher and has a value of about 0.611 which means that the support of simulated and real data is more similar than indicated by the first calculated value.
To better understand experimental effects are not yet covered, the experimental CSDs without a simulated CSD in the k-nearest-neighbor region are analyzed. These include some CSDs that contain only noise without a visible sensor response (see Figure 4.7a). Moreover, CSDs with less than 100x100 pixels (see Figure 4.7b) and images with a clear honeycomb structure and nearly no noise (see Figure 4.7c) are included. Furthermore, there also are CSDs with very long bursts (see Figure 4.7d) which could not be identified by only looking at the sensor scans for the determination of the RTN parameters. The rest of the CSDs have either some changes in the honeycomb structure or a single dot instead of a DQD is visible, which could for example come from a too high tunnel coupling between the two dots (see Figures 4.7e to 4.7i). For some further CSDs, it is not directly explainable why they are not represented in the simulated dataset. Thus, further investigations are necessary because of the complexity of the data.
To optimize the simulation, the identified second type of RTN (see Figure 4.7d) is added, whose parameters are determined directly in the CSDs. As there is no jump back visible, a length of 10000 pixels is assumed, which corresponds to the size of a CSD. For the normalized height of the second RTN, a mean of 0.14208 and a standard deviation of 0.44248 are determined from 4 out of 149 CSDs. To get a similar rate, this type of noise is included in 3% of the simulated CSDs. As the scaling factor of the sensing response is not available in this case the height is instead normalized by the difference between the minimum and maximum of the CSD.
To also simulate CSDs with almost no noise, the lower bound for white and pink noise is adjusted downwards to $1 \cdot 10^{-10}$. For the simulation of images which seem to contain only noise, the starting point on the sensor curve should be further away from the peak. Thus, for 10% of the CSDs, it is chosen from the interval $[\mu_0 - 4 \cdot \gamma, \mu_0 - 3 \cdot \gamma]$. This region has a shallow slope in comparison to the region $[\mu_0 - 2 \cdot \gamma, \mu_0 - 1 \cdot \gamma]$ which is usually chosen (see Figure 4.8). The region between the orange and green one, it not used for starting values of the CSD because the experimental CSDs either show a region without visible sensor response or a region near the peak, which often also shows the peak itself in the image.
The creation of more images with single dots and highly coupled dots, which are also missing in the simulated dataset, is not further investigated because the main aspect of this thesis is the simulation of distortions.



(a) Only noise (b) Smaller than 100x100 (c) No noise

(d) Long bursts (e) Structural difference (f) Structural difference

(g) Structural difference (h) Structural difference (i) Single dot

Figure 4.7: Examples of experimental CSDs which are not represented in the simulated CSDs. The transformation ϕ is calculated excluding any gate voltages. Colorbars are excluded for better clarity.

4.5.2 Power Spectral Density

The PSD mean for the experimental and simulated CSDs differs in its level while the shape is similar. Moreover, the variation of the PSD values is much higher for experimentalthan for simulated CSDs (see Figure 4.9). The PSDs in Figure 4.9 are calculated with the help of Welch's method [42]. One parameter of the function is the number of samples per segment (`nperseg`), which is set to 256 for CSDs containing more than 256 samples (see Figure 4.9a). Additionally, the diagram in Figure 4.9c was calculated with 100 samples per segment for all CSDs. The simulated CSDs all have a size of 100x100 pixels, so all simulated CSDs have been used to calculate the diagrams in Figure 4.9b and Figure 4.9d.

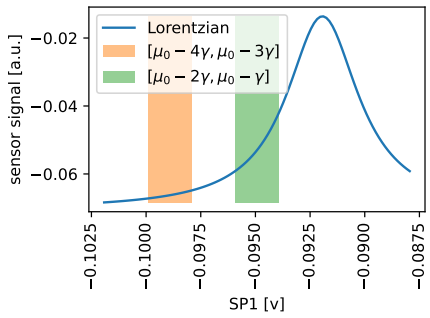
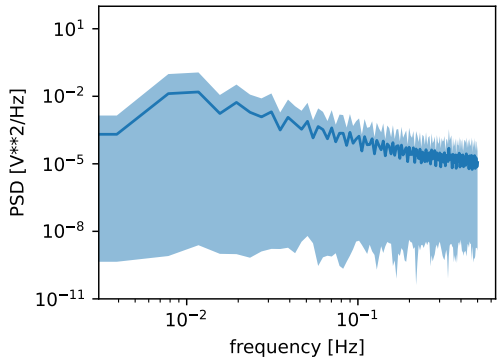
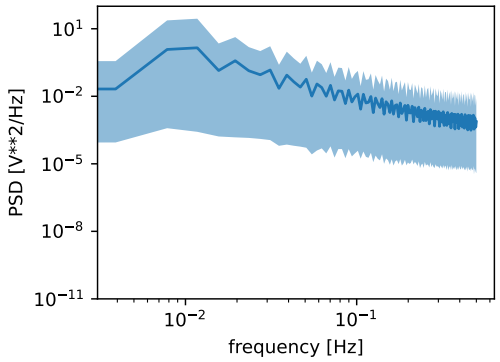


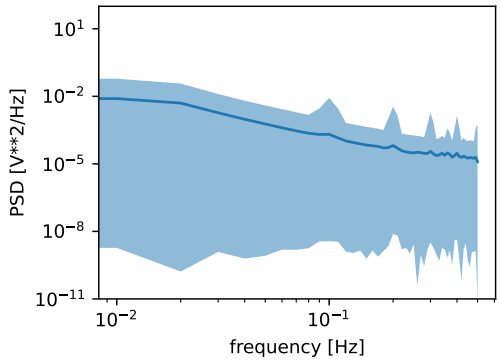
Figure 4.8: Lorentzian where the region $[\mu_0 - 4 \cdot \gamma, \mu_0 - 3 \cdot \gamma]$ (orange) and $[\mu_0 - 2 \cdot \gamma, \mu_0 - 1 \cdot \gamma]$ (green) is marked. In the orange region, the sensor is less sensitive than in the green one, leading to a lower value range of the CSD as well as more visible noise and less visible signal.



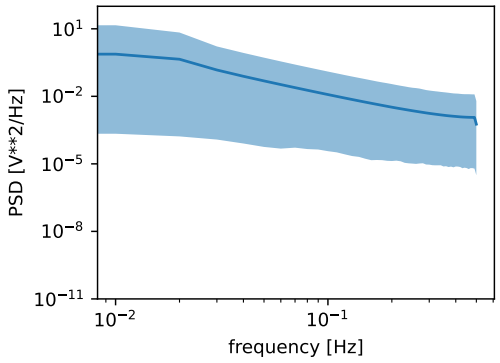
(a) Experimental CSDs, `nperseg` = 256



(b) Simulated CSDs, `nperseg` = 256



(c) Experimental CSDs, `nperseg` = 100



(d) Simulated CSDs, `nperseg` = 100

Figure 4.9: Mean and variation of the PSDs for (a, c) the experimental CSDs and (b, d) simulated CSDs. In (a,b) 256 samples per segment and in (c,d) 100 samples per segment are used for the computation.

The reason for the different levels of the mean PSD for the experimental and the simulated CSDs is the different value range. For the simulated CSDs the range is up to 1.8 while it is below 0.2 for the experimental CSDs. That means the scaling factor of the sensor

response determined in Section 3.2 is too high. To get lower scaling factors, the scaling factor is included in the error function used to determine the sensor response parameters. It is multiplied by the weight 0.001 to ensure that the main goal of minimizing the standard deviation of the gradient in non-edge regions is still met. Using this approach, the parameter ranges in Table 4.1 are determined.

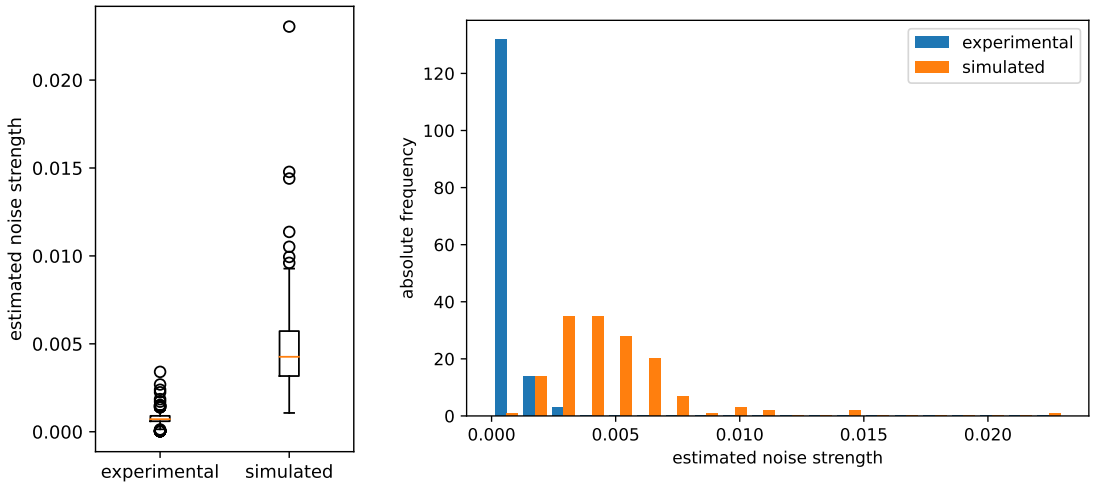
Table 4.1: Parameter ranges of the sensor response determined from 16 experimental CSDs and their corresponding sensor scans

parameter	minimum	maximum
S_{off}	-0.42275	-0.0838
μ_0	-0.12168	-0.03824
a	0.02245	0.19204
γ	0.0009636	0.0029509
$\alpha_{gate,1}$	0.02805	0.15093
$\alpha_{gate,2}$	0.03788	0.19491
$\alpha_{dot,1}$	-0.0007994	-0.0000961
$\alpha_{dot,2}$	-0.0005214	-0.0000630

However, this does not lead to a larger variation of the PSDs of the simulated CSDs. In Section 4.5.1, it was already mentioned that very noisy CSDs without visible sensor responses are not represented in the simulated dataset. These are also the cause for the low minimum of the PSD. The inclusion of CSDs starting at a sensor region with a low slope should also lead to a larger variation of the PSDs.

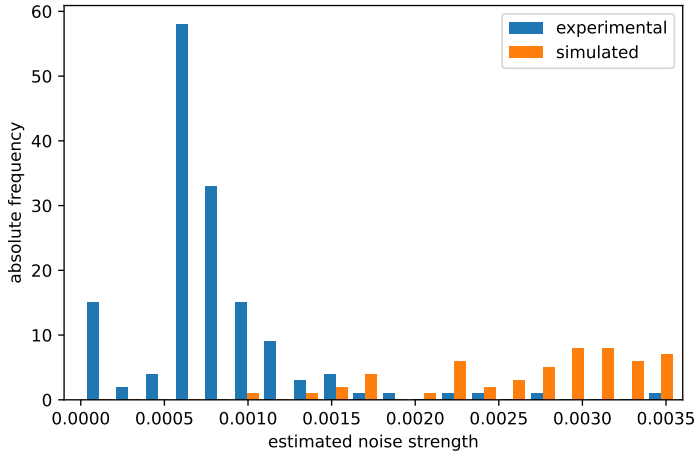
4.5.3 Distribution of Estimated Noise Strength

The distributions of the estimated noise strength for experimental and simulated data differ strongly (see Figure 4.10). For the simulated data higher noise strengths are estimated. That means that the chosen range for σ_{white} and σ_{pink} is probably too high.



(a) Boxplot

(b) Histogram

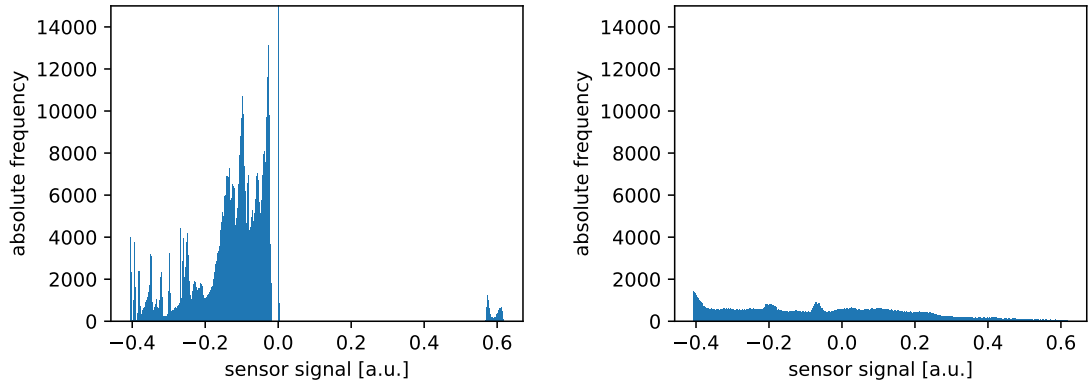


(c) More detailed histogram

Figure 4.10: (a) boxplot and (b,c) histogram of the estimated noise strengths in the experimental and simulated data. In (c), a smaller region with low values is shown to enable a more detailed look at the distribution of the noise strength in the experimental data.

4.5.4 Value Distribution

Looking at the values in the different experimental CSDs shows that there are two CSDs with very high values in comparison to the other CSDs. As this happens only for these two CSDs the reason for the high values seems to be a different transformation of the original values. The comparison of the histograms in Figure 4.11 shows that the values of the simulated datatsets have a much larger range. That is because the scaling factor for the sensor response is too high as mentioned in Section 4.5.2. The penalization of high scaling factors in the error function should solve this.



(a) Experimental CSDs

(b) Simulated CSDs

Figure 4.11: Histograms of the values in the (a) experimental CSDs in comparison to (b) the values in the simulated ones

The visual inspection of the distribution of the different values also reveals that values around zero appear very often in the experimental CSDs. This is mostly observed in CSDs which are very noisy and do not contain any structure. A first assumption in Section 4.5.1 has been that these CSDs come from the parts far away from the sensor peaks and thus do not show any structure because of the shallow slope. However, in this case, the values of these CSDs would be much smaller. One possible reason for the values around zero could be a transformation of the image values after the measurement. However, the real reason for these values is insufficiently known and thus cannot be simulated yet.

The values for the Kullback-Leibler and Jensen-Shannon divergence are 2.427 and 0.213. As there is no upper bound for the Kullback-Leibler divergence, its value can only be interpreted by comparing it with the value on CSD_{sim2}. For the Jensen-Shannon divergence, the value is in the lower half of its value range. However, it is far away from zero.

4.5.5 Visual Inspection

Looking at the structure of the honeycombs, the inter-dot transitions in the simulated data seem to be longer than the inter-dot transitions in the experimental CSDs (see Figure 4.13). The `id_lengths` determined in Section 3.1, also show that by far most of the lengths have lower values (see Figure 4.12).

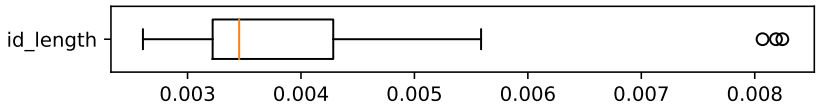
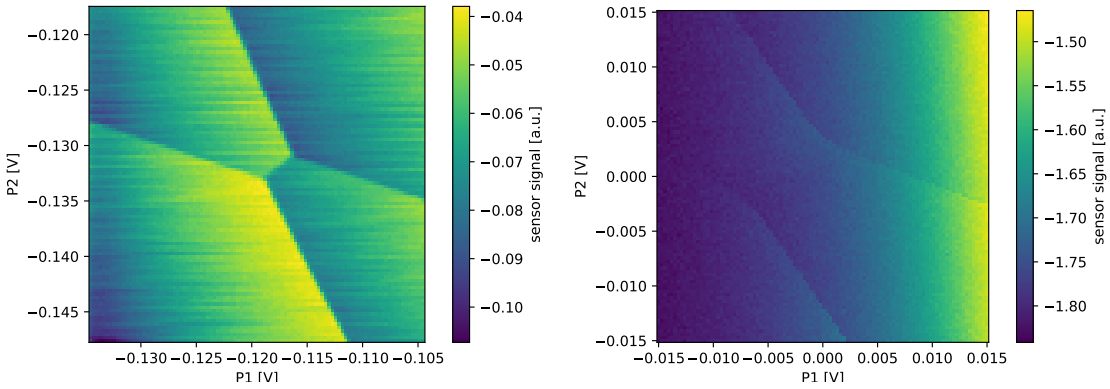


Figure 4.12: Boxplot of the `id_length` values determined in Section 3.1. Most determined `id_lengths` are in the lower region of the whole range.

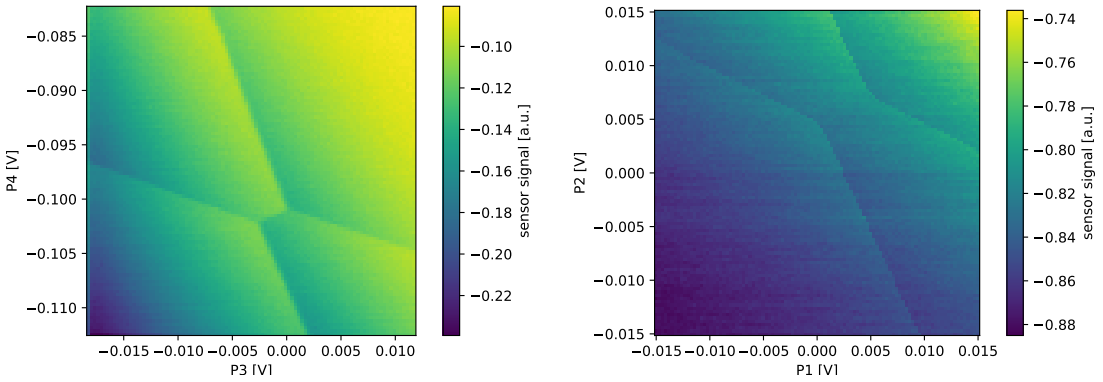
To avoid often large `id_lengths`, this parameter is not chosen from a uniform distribution but a normal distribution with an expectation of 0.004 and a standard deviation of 0.0015. These values are calculated from the previously determined parameters. Values below 0.00261 or above 0.00987 are discarded.



(a) Experimental CSD (b) Simulated CSD

Figure 4.13: Typical examples for the sizes of the inter-dot length for (a) an experimental CSD in comparison to (b) a simulated one

Moreover, most of the experimental CSDs have inter-dot transitions which are as sharp as the lead transitions (see Figure 4.13a). That means that in most CSDs the tunnel coupling is very low, thus, the minimum for `id_width` should be zero.
When comparing the lead transitions of the simulated and experimental CSDs, the simulated ones are sharper (see Figure 4.14). The reason for a slightly blurred edge in the experimental CSD is that, tunneling events often affect two neighbored pixel values during the measuring procedure. To reproduce this, a Gaussian blur is applied to the CSDs after adding the dot jumps and sensor response and before adding any noise. The parameter `sigma` of the Gaussian filter is set to 0.75 because only the neighboring pixels are influenced.

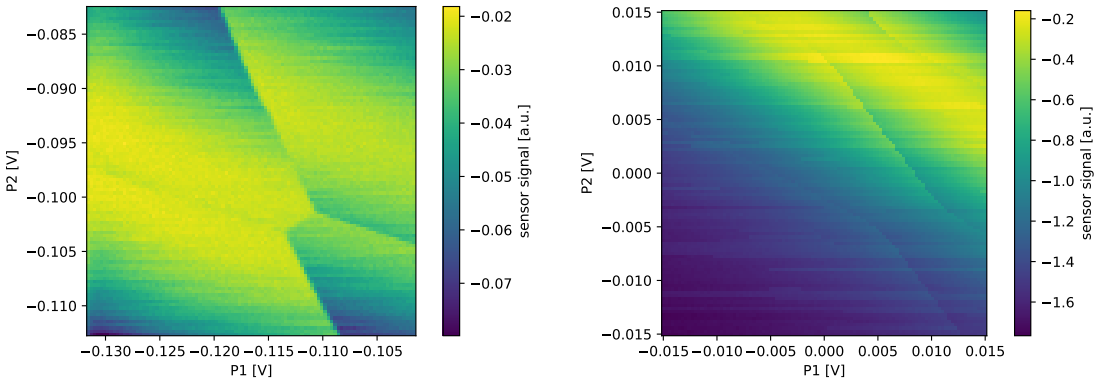


(a) Experimental CSD (b) Simulated CSD

Figure 4.14: Typical examples for the sharpness of the lead transitions for (a) an experimental CSD in comparison to (b) a simulated one

Furthermore, many simulated CSDs seem to contain low noise. However, this might result from their higher value range. Thus, the noise is less visible. In Section 4.5.3, it was already shown that the absolute strength of the noise is higher than in the real data. The compensation for this is the adaptation of the scaling factor for the sensor response and a lower range for the strength of white and pink noise.

In many simulated CSDs with RTN, the bursts are more clearly visible than in the experimental CSDs (see Figure 4.15). Moreover, the stripes in the experimental CSDs, most probably originating from pink noise and RTN, are not visible in the regions near the sensor peak. This is the case because pink noise and RTN seem to influence the sensor response not directly but via the electrochemical potential at the SD. Thus, they also have to be applied to the electrochemical potential of the SD in the simulation. The parameters for the strength of both noise types have to be adjusted by dividing them through the slope of the sensor response which is approximated by $\frac{a}{\gamma}$.



(a) Experimental CSD (b) Simulated CSD

Figure 4.15: Typical examples of RTN for (a) an experimental CSD in comparison to (b) a simulated one. In addition, pink noise is visible in the experimental CSD.

4.5.6 Summary of the Changes to the Simulation

Table 4.2 summarizes the adapted parameters for the optimized simulated dataset CSD_{sim2}. In addition to the changes of the parameters, new parameters for the Gaussian blur as well

Table 4.2: Summary of the adapted parameters for CSD_{sim2}

	parameter	new value(s)
senor response	S_{off}	[-0.42275, -0.0838]
	a	[0.02245, 0.19204]
	$\alpha_{gate,1}$	[0.02805, 0.15093]
	$\alpha_{gate,2}$	[0.03788, 0.19491]
	$\alpha_{dot,1}$	[-0.0007994, -0.0000961]
	$\alpha_{dot,2}$	[-0.0005214, -0.0000630]
white noise	σ_{white}	$[1 \cdot 10^{-10}, 0.003]$
pink noise	σ_{pink}	$[1 \cdot 10^{-10}, 0.005]$
Gaussian blur	σ_{blur}	0.75
RTN after sensor	expected length	10000
	expected height	0.14208
	standard deviation of height	0.44248

as the second type of RTN, called RTN after sensor, are included. Furthermore, the sampling of the `id_length` is changed to sampling from a normal distribution and the original RTN and pink noise are applied to the SD potential. The maximal strength of white noise is lower than for pink noise because it is applied after the sensor. As pink noise is applied before the sensor, it is only present in the given strength at the steepest slopes of the sensor response. In 10% of the time, the offset μ_{off} is chosen from the interval $[\mu_0 - 4\gamma, \mu_0 - 3\gamma]$ to simulate CSDs with a smaller slope of the sensor response.

4.6 Evaluation of the Optimized Simulated Dataset

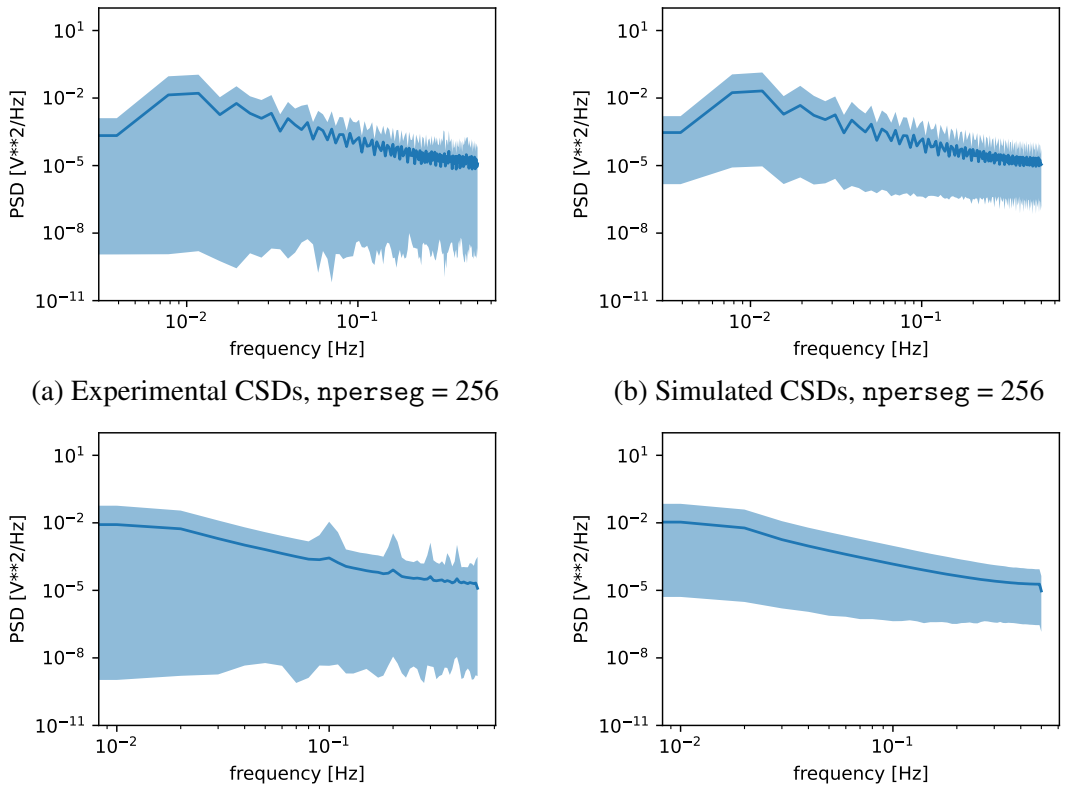
The optimized simulated dataset CSD_{sim2} is evaluated using the evaluation dataset CSD_{exp2} consisting of 154 experimental CSDs that were not used in the model refinement stage to avoid positive biasing.
In the following, the same evaluation methods are used as in Section 4.5.

4.6.1 α-Precision and β-Recall

The transformation of the CSDs is carried out with the help of the same network ϕ as used during the model refinement stage. The training on the new experimental dataset CSD_{exp2} may generate different transformations. Thus, the results would not be comparable with the results of the first simulated dataset CSD_{sim1} anymore.
α-Precision and β-Recall are first calculated on the optimization dataset CSD_{exp1} and the optimized simulated dataset CSD_{sim2} to check if a quality gain was achieved. However, these results generally have a lower significance because the optimization set has already been used to identify the problems of CSD_{sim1}. For reliable results, the α-Precision and β-Recall also have to be calculated for CSD_{exp2}.
The comparison of CSD_{exp1} with CSD_{sim2} leads to an α-Precision of about 0.928 and a β-Recall of about 0.308. For CSD_{exp2}, the values are similar.
In the following, the α-Precision and β-Recall of CSD_{exp1} and CSD_{sim1} are compared with the results for CSD_{exp2} and CSD_{sim2}. The α-Precision calculated on CSD_{exp2} and CSD_{sim2} is about 0.942 which is about 0.04 less than the value reached in the model refinement stage. However, the value for the β-Recall, indicating the diversity of the simulated dataset, increased by about 0.2 to a value of 0.331. This is a relative increase of about 160%. Although this is a large increase only about one-third of the experimental dataset is covered by the simulated CSDs. On the other hand, the fidelity of the simulated dataset decreased but is still at a very high level. This is important if machine learning algorithms should be trained with simulated CSDs.
However, for a large simulated dataset with 1000 samples, the β-Recall value does not increase in comparison to the result in the refinement stage. In this case, a value of about 0.557 is calculated for the comparison with CSD_{exp1} and a value of about 0.61 for CSD_{exp2}. This means that the model refinement did not improve the support of the simulated data. However, the distribution of CSD_{sim2} is more similar to the distribution of the experimental data than the distribution of CSD_{sim1}.
Usually, simulated CSDs leading to a lower α-Precision are very noisy images, indicating that their noise strength might be too high.
The simulated CSDs leading to a low β-Recall are still similar to those presented in Figure 4.7. Merely, the images with long bursts are now all represented in the simulated dataset. CSDs with structural differences or a size smaller than 100x100 pixels are still not included. Moreover, images with only noise or very low noise are not present in CSD_{sim2} according to the recall classifier.

4.6.2 Power Spectral Density

With the adjusted scaling factor for the sensor response, the PSD mean is similar for the experimental and simulated CSDs (see Figure 4.16). However, the variation of the PSD is still much smaller for the simulated CSDs. The reason for low PSD values in the experimental data are images with a value range of lower than 0.001. The lowest value range for the simulated CSDs is about 0.008.



(a) Experimental CSDs, `nperseg` = 256

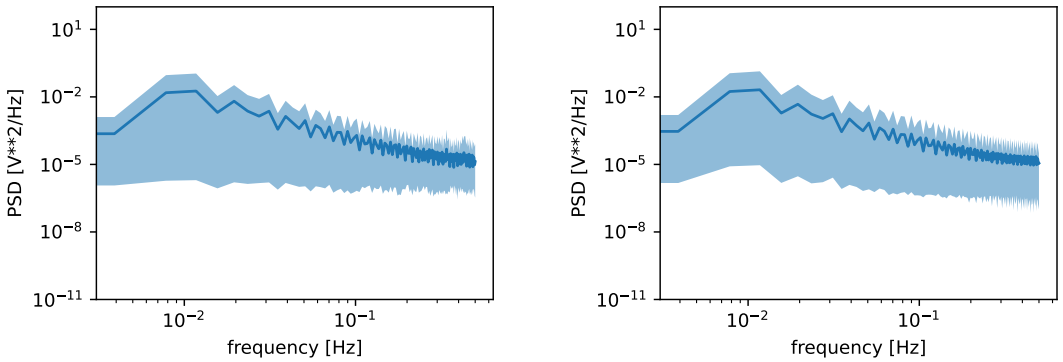
(b) Simulated CSDs, `nperseg` = 256

(c) Experimental CSDs, `nperseg` = 100

(d) Simulated CSDs, `nperseg` = 100

Figure 4.16: Mean and variation of the PSDs for the (a, c) experimental CSDs and (b, d) simulated CSDs. In (a,b) 256 samples per segment and in (c,d) 100 samples per segment are used for the computation.

In case of neglecting CSDs with value ranges smaller than 0.001 or with a size different from 100x100 pixel, the variation of the PSDs is more similar (see Figure 4.17). In total, 24 of the 154 CSDs in the evaluation dataset CSD_{exp2} are excluded for the diagram in Figure 4.17. In this case, the only difference left is that the lowest values for the experimental CSDs seem to originate from some CSDs containing only white noise, as they are constant over all frequencies. For the simulated data, there is either still some structure from the sensor response or pink noise present in the CSDs with the lowest PSD values.

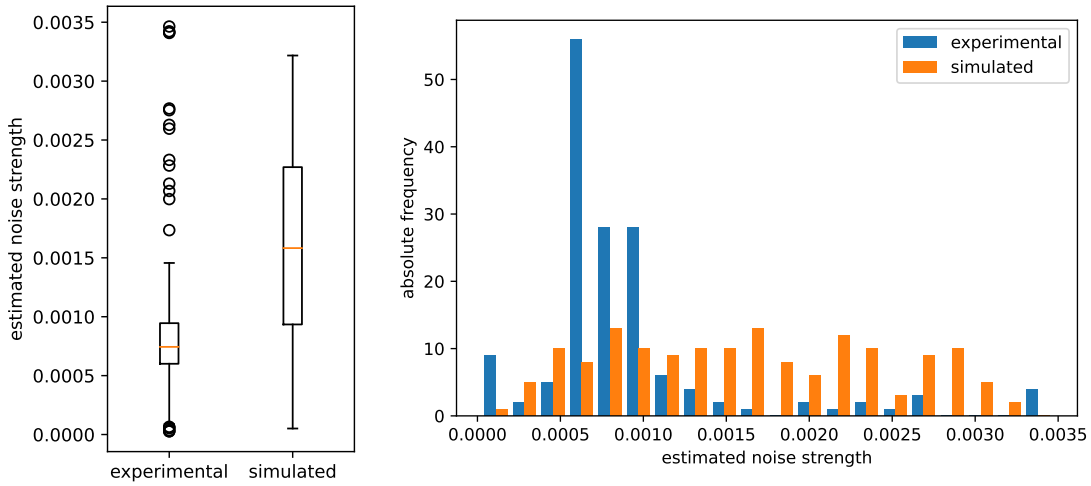


(a) Experimental CSDs (b) Simulated CSDs

Figure 4.17: Mean and variation of the PSDs for the (a) experimental CSDs and (b) simulated CSDs. For the computation, 256 samples per segment are used. For the experimental data, those CSDs are omitted that have a value range smaller than 0.001 or a size different from 100x100 pixels.

4.6.3 Distribution of Estimated Noise Strength

The adjustment of the upper bound for the intensity of white and pink noise leads to a more similar distribution of the estimated noise strength in the experimental and simulated dataset (see Figure 4.18).



(a) Boxplot (b) Histogram

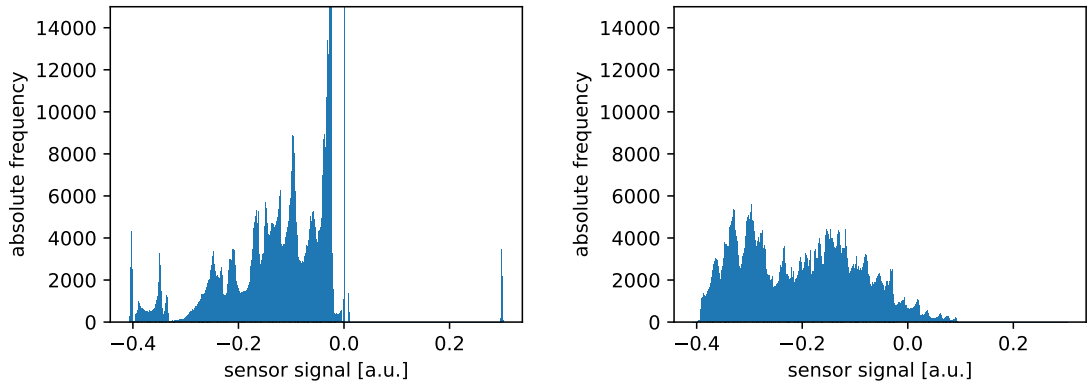
Figure 4.18: (a) boxplot and (b) histogram of the estimated noise strengths in the experimental and simulated data

One remaining difference is that the simulated dataset shows a non-skewed nearly uniform distribution, whereas the experimental dataset has a right-skewed distribution. This

implies that a normally distributed noise strength could be more realistic. However, a more detailed analysis of the experimental CSDs with noise estimations between 0.0006 and 0.001 is required to find out if the peak in the histogram only arises from CSDs that are measured in short succession, thus, showing similar noise characteristics.

4.6.4 Value Distribution

The adaption of the simulation parameters reduces the difference between the value range of the experimental and simulated CSDs (see Figure 4.19). Also, the similarity of the distribution of values is improved, as the values for the Kullback-Leibler and Jensen-Shannon divergence are only 0.66, respectively, 0.14. But values around 0 are still more common for the experimental images. Mostly, these values derive from the CSDs that lead to the small PSD values as described in Section 4.6.2.



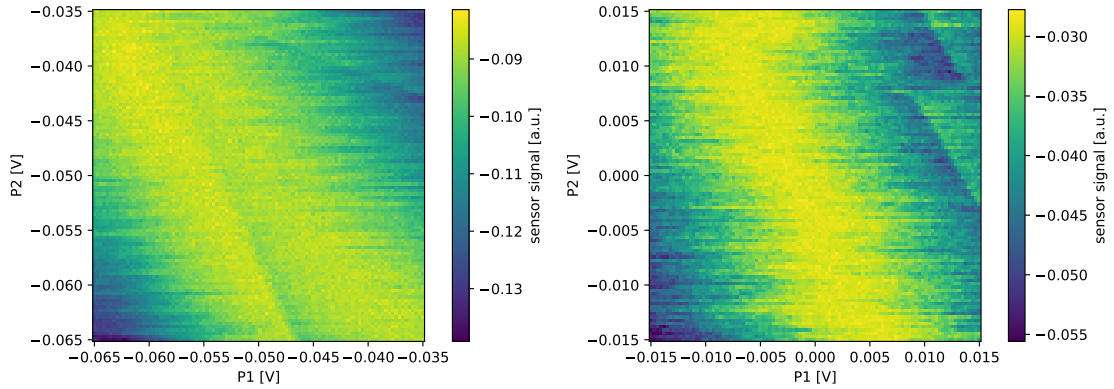
(a) Experimental CSDs

(b) Simulated CSDs

Figure 4.19: Histograms of the values in (a) the experimental CSDs and (b) the values in the simulated ones

4.6.5 Visual Inspection

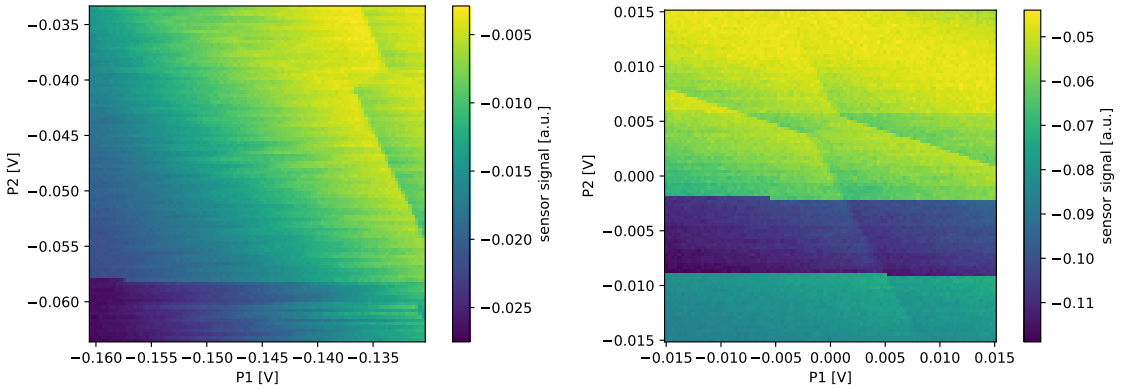
In the simulated dataset, some CSDs contain very high pink noise in comparison to the experimental CSDs (see Figure 4.20).



(a) Experimental CSD (b) Simulated CSD

Figure 4.20: Typical examples of (a) an experimental and (b) a simulated CSD with strong pink noise.

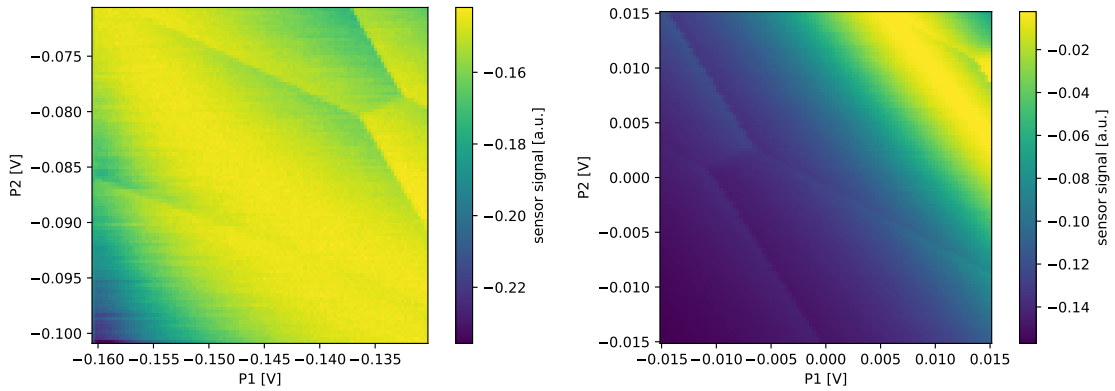
The same also applies to the RTN after the sensor. While a maximum of one jump is visible in experimental CSDs, there are more jumps in the simulated CSDs (see Figure 4.21).



(a) Experimental CSD (b) Simulated CSD

Figure 4.21: Typical example of (a) an experimental and (b) a simulated CSD with RTN after the sensor

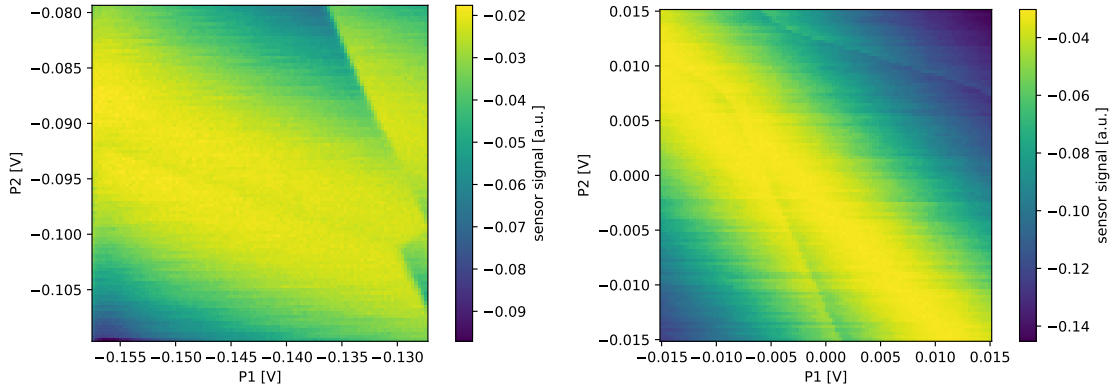
The comparison of the sensor response shows that the sensor peak in the experimental CSDs often seems to be wider or α_{gate} is larger in the simulated CSDs (see Figure 4.22).



(a) Experimental CSD (b) Simulated CSD

Figure 4.22: Typical example of a sensor peak in the (a) experimental and (b) simulated data

Notably, RTN and pink noise applied before the sensor look more realistic (see Figure 4.23). Furthermore, the sharpness of the edges is more realistic, but the experimental dataset also contains images with an even lower edge sharpness. Additionally, the length of the inter-dot transitions is now closer to the length visible in the experimental CSDs.



(a) Experimental CSD (b) Simulated CSD

Figure 4.23: Typical examples for RTN and pink noise before the sensor for (a) an experimental CSD in comparison to (b) a simulated one

5 Discussion

The following chapter summarizes and interprets the presented results concerning the aim of the thesis to simulate realistic distortions in CSDs. This also includes a discussion of the strengths and weaknesses of the evaluation methods.

5.1 Strengths and Weaknesses of the Evaluation Methods

Different evaluation methods with different strengths and weaknesses are used to evaluate if CSDs with realistic distortions are generated. The α-Precision and β-Recall metrics have the advantage, that they determine an interpretable number which is easily comparable. Moreover, this also enables further analysis of the reasons for low precision and recall as the metrics work sample-wise. On the other hand, there is a risk that the underlying neural network ϕ learned unsuitable transformations for the comparison. To lower this risk, the evaluation set CSD_{exp2} is used to compare if similar CSDs of CSD_{exp1} and CSD_{exp2} are transformed similarly (see Figures 4.5 and 4.6). Regarding the β-Recall metric, the choice of k for the k-nearest-neighbor region highly influences the resulting value. This is a problem if the density of the experimental CSDs varies inside the hypersphere while the density of the simulated data is homogeneous. In this case, there might be no simulated CSD in the k-nearest-neighbor-region but the overall support of both distributions might still be similar. Thus, low recall values indicate that the distribution of simulated and experimental data differs but not necessarily that only a small percentage of the experimental distribution is covered by the simulated data. To get a better insight into this problem, larger simulated datasets with 1000 samples have been created, and the β-Recall metric was recalculated. Another disadvantage of this evaluation method is the standardization of the image values before the transformation which makes it impossible to check if the value distribution of the experimental and simulated dataset matches.
To overcome this problem, this aspect was analyzed separately. The Kullback-Leibler and

Jensen-Shannon divergence are commonly used for the comparison of distributions. They enable an easy comparison if the value distribution of the CSD_{sim1} or the adapted CSD_{sim2} is more similar to the one of the experimental datasets. However, the interpretation of this value is difficult, especially for the Kullback-Leibler divergence as there is no upper bound for it. For this reason, the histograms of all values in the datasets are analyzed as well.

Not only is the value distribution important for the comparison, but also the frequency distribution which can be compared with the help of the PSD. The advantage of this evaluation method is that, up to a certain degree, the composition of the noise present in the CSDs can be analyzed. This aspect is especially important for the aim of this thesis as the focus lies on the simulation of realistic noise. A drawback of this approach is that as the PSD was already used for the parameter determination, it might not give more information than already used. Furthermore, it does not provide information about the distribution of the total noise strength in the images. This aspect is covered by the calculation of the estimated overall noise strength and the comparison of its distribution. The used noise estimator estimates the overall noise which looks like white noise. This includes also parts of the pink noise but not of the RTN and dot jumps. However, as these types of noise are not that common, it still provides a good insight into the distribution of the noise strength in the CSDs.

In the end, the combination of the different evaluation methods enables a differentiated assessment of the quality of the simulated dataset. To ensure that no difference between the simulated and experimental data is missed, the dataset is additionally analyzed visually.

5.2 Summary of the Evaluation Results

A comparison of the first simulated dataset CSD_{sim1}, the optimized simulated dataset CSD_{sim2}, and the experimental datasets $CSD_{exp1,2}$ shows, that there are still some differences between the simulated CSDs and the experimental CSDs. However, by optimizing some of the parameters and adapting the simulation model, the results of the final simulation are improved. The quality was measured using the α-Precision and β-Recall with $\alpha = 0.95$ and $\beta = 1$, which indicate the fidelity and diversity of the simulated dataset. The α-Precision has a high value above 0.94 for both CSD_{sim1} and CSD_{sim2}, while the diversity is not as high with a β-Recall value of below $\frac{1}{3}$ for both datasets. This results from the fact that datasets with only noise or with very low noise are not represented in both simulated datasets. This could be the case because the simulated noise differs from the experimental noise or because the simulated datasets still show too much of the sensor response. For the experimental data with low noise, it might be possible that the hon-

eycomb structures differ from the simulated honeycomb structures. This could be the case because different characteristics of the experimental structures are not yet included in the simulation. But another reason could also be that the determined parameters are not precise enough. However, the β-Recall is still increased by 0.2 (160%) for CSD_{sim2}. This means that especially the distribution of the simulated and experimental data is more similar for the optimized simulation model. For the large simulated datasets with 1000 samples, the value for the first and the optimized simulated dataset does not differ much. So, the support of the simulated data could not be increased. The other evaluation methods have shown a stronger effect when applying the adapted simulation model.
While the PSD and the value distribution indicate, that for the CSDs in CSD_{sim1} the value range has been too large, the range fits the experimental image ranges for CSD_{sim2}. This improvement is achieved by the punishment of large scaling factors during the fitting of the sensor response parameters. The comparison between the mean PSD of the simulated and experimental data also shows a similar frequency distribution. This indicates, that the noise model is working fine.
However, the lowest experimental PSD values are lower than the simulated ones (see Figure 4.16) due to the very low value ranges in the experimental CSDs. These value ranges are from similar CSDs like the ones which also lead to a small β-Recall for CSD_{sim2}. CSDs with such a small value range result from a badly tuned SD or a postprocessing of the image values. In both cases, it is questionable if the inclusion of these types of CSDs into the simulation model is useful as the SD should be tuned correctly before the measurement of a CSD. Moreover, the postprocessing steps have to be known to include them in the simulation. If the images with a very low range and a size different than 100x100 pixels are not considered, the variation of the CSDs is more similar (see Figure 4.17). However, the lowest PSD values probably originate from pure white noise which also indicates that the sensor is tuned to a non-sensitive region. Nevertheless, the inclusion of these datasets does not require an adaption of the simulation model but only a change of the offset μ_{off} for the sensor response. To create these CSDs, the initial position of the sensor should be further away from the peak of the sensor response.
Additionally, the distribution of the noise strength in the simulated and experimental data is significantly improved by choosing the noise strength for white and pink noise from a smaller range.
However, there are still some differences between the simulated and the experimental data opening the possibility to differentiate the simulated CSDs from the experimental ones. This includes the structure of the honeycombs which is often distorted in ways which are not included in the simulation yet (see Figures 4.7e to 4.7h) because not all of them are understood by now. Moreover, the sharpness of the edges seems to vary less in the simulated dataset CSD_{sim2}. Additionally, on average, the noise is stronger in the simulated CSDs (see Section 4.6.3). Finally, for the RTN after sensor, the length parameter seems

to be too low because in some images several jumps are visible (see Figure 4.21) whereas never more than one is visible in the experimental data.
Another issue is the width of the sensor response peak which seems to be too low in some simulated CSDs (see Figure 4.22). This indicates that either the peak is too narrow or α_{gate} is too large in relation to the width of the sensor response. The relation between α_{gate} and γ is already limited for the simulation, but could be further reduced for a higher percentage of simulated CSDs.

5.3 Interpretation of the Evaluation Results

The aim of this thesis is the simulation of realistic distortions in DQD CSDs to enable the development and testing of algorithms for the automated tuning of DQDs including machine learning algorithms.
Different evaluation methods have been used to evaluate if this aim is reached. In conclusion, they show that the simulated CSDs are a subgroup of the experimental CSDs. That means that they can be used to test algorithms and also train machine learning algorithms. However, at the moment, also experimental data have to be included in this process because there are even more possible distortions than described in Section 2.5. To enable the testing and training with only simulated CSDs, the simulation model first has to be extended.
In this context, it also has to be checked if the training including simulated CSDs leads to a lower generalization error for a machine learning model or if training with only experimental data works better. If the training with simulated data leads to an overfitting to special characteristics of the simulated data, the application on experimental data will not be successful. On one hand, the previous work of Darulova et al. [20] showed that training with only experimental data still worked better than using only simulated data. A combination of synthetic and experimental data showed a similar performance as the usage of only experimental data. Even, the addition of common noise types did not improve the performance much. On the other hand, the work of Ziegler et al. [24] showed a better performance for the classification of the occupation regime when noise is included in the simulated data.
Even if the training with simulated data does not directly improve the performance, it can help to understand better which aspects of the experimental data are important for the algorithm. This knowledge can then be used to improve the simulation, which could again lead to an improvement of the tuning algorithm.
Especially for deep neural networks, a better performance can be expected with the in-

clusion of simulated data, because the available experimental dataset does not contain enough samples for a proper training. Another problem with the experimental data is that the different honeycomb structures are not as diverse as in the simulated data because the same voltage configuration is measured several times leading to only minor changes between the datasets.
The generated simulated data represent typical CSDs which are measured during the tuning procedure. However, there are still some problems that could not be solved yet. This includes the simulation of virtual gates, which lead to nearly horizontal and vertical lead transitions and also remove the capacitive coupling effects between the swept gates and the sensor. Moreover, parameters for the generation of single dot line patterns should be determined to also include those single dots that are caused by a barrier too weak to separate the two dots of the DQD. Furthermore, other structural distortions of the honeycomb pattern resulting from moving dots or parasitic dots (see Figure 4.2) should be further analyzed for implementation in the simulation. However, depending on the type of algorithm to be tested, these types of data are not required. In general, the simulated dataset should be adapted to the application as not every type of distortion and honeycomb shape will be present for every step of the tuning algorithm.
However, CSD data are not sufficiently understood for their realistic simulation. Therefore, also a perpetual adaptation of the simulation model is important. That means, new knowledge gained during the algorithm development for the DQD tuning should be directly included in the simulation model to improve the simulation.
One reason for a slow knowledge gain concerning the understanding of CSDs is the amount of data available because each research group records its data without making them publicly available. This also makes the development of simulated CSDs harder. As a contribution to overcome this problem, the software developed in this thesis is planned to be published. Moreover, also some first methods for parameter determination and evaluation of the generated CSDs have been presented. This could help other research groups with the simulation of their measured CSDs.

6 Conclusion

This thesis deals with the characterization and simulation of distortions in CSDs with the aim of using them for the development and testing of algorithms for the automated tuning of DQDs. First, the used simulation model generates an undistorted CSD containing the occupation per dot. Then, distortions are added to the generated clean data. This has the advantage that a ground truth for each data point is directly available, and enables the generation of large labeled datasets.

Moreover, the suggested model has the advantage that the strength of each type of noise can be controlled. This is beneficial for the comparison of the strengths and weaknesses of different algorithms used for the tuning.

For the simulation of the undistorted CSDs, the capacitive model [1] and the Hubbard model [2] were compared. It was concluded that non of them enables the creation of the characteristics visible in the experimental CSDs recorded with the experimental setup described in Section 2.2. The capacitive model does not include the inter-dot tunnel coupling and the Hubbard model does not have enough parameters to enable the fitting of all honeycomb characteristics. This manifested itself in the fact that the honeycombs are too small if the other characteristics match. Thus, a new simulation method developed by Fabian Hader from the ZEA-2 was used.

For the simulation of distortions, the most commonly occurring distortions were identified. These are the cross-coupling between the sensor and double dot plunger gates, white noise, pink noise, RTN, and dot jumps. For each type of distortion parameter ranges for the simulation were determined.

A first simulated dataset CSD_{sim1} has been generated and evaluated using the α-Precision and β-Recall which are originally used for the evaluation of generative models. In addition, the PSD, the distribution of estimated noise levels, and the value distribution were compared between simulated data and experimental data. For a qualitative evaluation of the simulated data, a visual comparison regarding the honeycomb structure, edge sharpness, visibility of the sensor response, and characteristics of the distortions was carried out. These investigations have shown that CSD_{sim1} differs from the experimental data especially in terms of the value range, the visibility of pink noise and random telegraph noise, the strength of the noise, and the edge sharpness.

For the generation of the optimized simulated dataset CSD_{sim2}, the range for the sampling of the sensor dot scaling factor was lowered by punishing high scaling factors during the determination of the sensor response parameters. Moreover, the simulation of pink noise and RTN was applied before the sensor response so that the height of the jumps is dependent on the slope of the sensor response. As the estimated noise strengths were higher in the simulated data, the strength of white and pink noise was reduced. In addition, a smoothing of the lead transitions was applied because the edges in the first simulated dataset were too sharp.
CSD_{sim2} has been evaluated with the same methods as CSD_{sim1}. It was found that the similarity of the distribution, the PSD and the value range of the generated samples and the experimental data increased. Moreover, the visually analyzed characteristics were more similar.
In summary, the simulated data already emulate the real data quite well, making a step forward to a greater publicly available knowledge base for CSDs.

7 Outlook

In the future, other effects which are not yet covered should be included in the simulation. These comprise the simulation of virtual gates, parasitic dots, and moving dots, which lead to curved lead transitions. Moreover, the reason for the irregular sensor response (see Figure 4.2d) and J-shaped lines (see Figure 4.2e) should be analyzed and included in the simulation process. Another important aspect is the inclusion of the structural differences (see Figures 4.7e to 4.7h) and of single dots resulting from an insufficient barrier between the dots.

Until now, the voltage range of the experimental CSDs has not been taken into account for the determination of the parameters. However, at least for the structure of the honeycombs the swept range has an effect as the inter-dot tunnel coupling increases for larger voltages if no compensation via the other gate voltages is performed in between. Additionally, a correlation between the applied voltages and the resulting noise should be examined.

The simulation of the electron occupation tuning requires the generation of honeycomb structures with a varying parameter for the inter-dot tunnel coupling. After measuring one small CSD, the plunger gate voltages are adjusted to get to the (1, 1) electron region. Then, a new CSD is measured with the new voltages. This existing relationship between the small CSDs should later on also be represented in the simulated data.

Experimental data from quantum dots implemented in GaAs/AlGaAs heterostructures were used for the determination of the simulation parameters. However, the Quantum Technology Group of the RWTH is going to focus on Si/SiGe technology in the future. Moreover, there are also other promising technologies used for DQDs. Therefore, the simulation model, respectively, the parameters of the current model should be adapted to provide simulated CSDs for this realization of DQDs.

Finally, the impact of using simulated data for the development of automated tuning algorithms including machine learning algorithms should be examined. This could also lead to a symbiosis of continuously improving simulation and tuning algorithm, as the tuning algorithm will show the weaknesses of the simulation and the simulated data at the same time help to find improvements in the tuning algorithm.

Bibliography

[1] W. G. van der Wiel et al. “Electron transport through double quantum dots”. en. In: *Reviews of Modern Physics* 75.1 (Dec. 2002), pp. 1–22. ISSN: 0034-6861, 1539-0756. DOI: 10.1103/RevModPhys.75.1. URL: https://link.aps.org/doi/10.1103/RevModPhys.75.1 (visited on 09/22/2022).

[2] S. Yang, X. Wang, and S. Das Sarma. “Generic Hubbard model description of semiconductor quantum-dot spin qubits”. en. In: *Physical Review B* 83.16 (Apr. 2011), p. 161301. ISSN: 1098-0121, 1550-235X. DOI: 10.1103/PhysRevB.83.161301. URL: https://link.aps.org/doi/10.1103/PhysRevB.83.161301 (visited on 09/22/2022).

[3] *Central Institute of Engineering, Electronics and Analytics (ZEA)*. URL: https://www.fz-juelich.de/de/zea (visited on 09/22/2022).

[4] *About Us*. URL: https://www.fz-juelich.de/en/zea/zea-2/about-us (visited on 09/22/2022).

[5] *Institut für Quanteninformation*. URL: https://www.quantuminfo.physik.rwth-aachen.de/cms/~dqvn/Quantuminfo/ (visited on 09/22/2022).

[6] R. P. Feynman. “Simulating physics with computers”. en. In: *International Journal of Theoretical Physics* 21.6-7 (June 1982), pp. 467–488. ISSN: 0020-7748, 1572-9575. DOI: 10.1007/BF02650179. URL: http://link.springer.com/10.1007/BF02650179 (visited on 05/15/2022).

[7] P. W. Shor. “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer”. In: *SIAM Journal on Computing* 26.5 (Oct. 1997), pp. 1484–1509. ISSN: 0097-5397, 1095-7111. DOI: 10.1137/S0097539795293172. URL: http://arxiv.org/abs/quant-ph/9508027 (visited on 05/19/2022).

[8] R. L. Rivest, A. Shamir, and L. Adleman. “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems”. In: *Commun. ACM* 21.2 (Feb. 1978), pp. 120–126. ISSN: 0001-0782. DOI: 10.1145/359340.359342. URL: https://doi.org/10.1145/359340.359342.

[9] I. L. Chuang, N. Gershenfeld, and M. Kubinec. “Experimental Implementation of Fast Quantum Searching”. en. In: *Physical Review Letters* 80.15 (Apr. 1998), pp. 3408–3411. ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.80.3408. URL: https://link.aps.org/doi/10.1103/PhysRevLett.80.3408 (visited on 05/20/2022).

[10] F. Arute et al. “Quantum supremacy using a programmable superconducting processor”. en. In: *Nature* 574.7779 (Oct. 2019), pp. 505–510. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/s41586-019-1666-5. URL: http://www.nature.com/articles/s41586-019-1666-5 (visited on 05/20/2022).

[11] C. Dilmegani. *Top 20 Quantum Computing Use Cases in 2022*. URL: https://research.aimultiple.com/quantum-computing-applications/ (visited on 05/25/2022).

[12] T. Botzem. “Coherence and high fidelity control of two-electron spin qubits in GaAs quantum dots”. PhD thesis. 2017. DOI: 10.18154/RWTH-2017-04410. URL: https://publications.rwth-aachen.de/record/689507.

[13] C. O’Connell. *Quantum computing for the qubit curious*. URL: https://cosmosmagazine.com/physics/quantum-computing-for-the-qubit-curious. (Visited on 05/27/2022).

[14] S. Aaronson. “The Limits of Quantum Computers”. In: *Scientific American* 298.3 (Mar. 2008), pp. 62–69. ISSN: 0036-8733. DOI: 10.1038/scientificamerican0308-62. URL: https://www.scientificamerican.com/article/the-limits-of-quantum-computers (visited on 05/16/2022).

[15] P. Cerfontaine. “High-fidelity single- and two-qubit gates for two-electron spin qubits”. en. PhD thesis. RWTH Aachen University, 2019. DOI: 10.18154/RWTH-2019-09348. URL: http://publications.rwth-aachen.de/record/768380 (visited on 05/18/2022).

[16] F. Hader. “Noise Analysis and Estimation in Sensor Dot Fine Scans for Automated Tuning of Gate Defined Quantum Dots”. MA thesis. Fachhochschule Aachen, 2021.

[17] T. Botzem et al. “Tuning Methods for Semiconductor Spin Qubits”. In: *Phys. Rev. Applied* 10 (5 Nov. 2018), p. 054026. DOI: 10.1103/PhysRevApplied.10.054026. URL: https://link.aps.org/doi/10.1103/PhysRevApplied.10.054026.

[18] J. P. Zwolak and J. M. Taylor. *Colloquium: Advances in automation of quantum dot devices control*. Dec. 2021. URL: http://arxiv.org/abs/2112.09362 (visited on 05/16/2022).

[19] R. Hanson et al. “Spins in few-electron quantum dots”. In: *Reviews of Modern Physics* 79.4 (Oct. 2007), pp. 1217–1265. ISSN: 1539-0756. DOI: 10.1103/revmodphys.79.1217. URL: http://dx.doi.org/10.1103/RevModPhys.79.1217.

[20] J. Darulova, M. Troyer, and M. C. Cassidy. “Evaluation of synthetic and experimental training data in supervised machine learning applied to charge state detection of quantum dots”. In: *arXiv:2005.08131 [cond-mat, physics:quant-ph]* (May 2020). URL: http://arxiv.org/abs/2005.08131 (visited on 05/28/2022).

[21] S. S. Kalantre et al. “Machine learning techniques for state recognition and auto-tuning in quantum dots”. en. In: *npj Quantum Information* 5.1 (Dec. 2019), p. 6. ISSN: 2056-6387. DOI: 10.1038/s41534-018-0118-7. URL: http://www.nature.com/articles/s41534-018-0118-7 (visited on 05/28/2022).

[22] R. Durrer et al. “Automated Tuning of Double Quantum Dots into Specific Charge States Using Neural Networks”. en. In: *Physical Review Applied* 13.5 (May 2020), p. 054019. ISSN: 2331-7019. DOI: 10.1103/PhysRevApplied.13.054019. URL: https://link.aps.org/doi/10.1103/PhysRevApplied.13.054019 (visited on 05/28/2022).

[23] J. P. Zwolak et al. “Autotuning of Double-Dot Devices *In Situ* with Machine Learning”. en. In: *Physical Review Applied* 13.3 (Mar. 2020), p. 034075. ISSN: 2331-7019. DOI: 10.1103/PhysRevApplied.13.034075. URL: https://link.aps.org/doi/10.1103/PhysRevApplied.13.034075 (visited on 05/28/2022).

[24] J. Ziegler et al. “Toward Robust Autotuning of Noisy Quantum Dot Devices”. In: *arXiv:2108.00043 [quant-ph]* (July 2021). URL: http://arxiv.org/abs/2108.00043 (visited on 05/13/2022).

[25] S. Czischek et al. “Miniaturizing neural networks for charge state autotuning in quantum dots”. In: *arXiv:2101.03181 [cond-mat, physics:quant-ph]* (Jan. 2021). URL: http://arxiv.org/abs/2101.03181 (visited on 05/28/2022).

[26] J. Darulová et al. “Autonomous tuning and charge state detection of gate defined quantum dots”. In: *Physical Review Applied* 13.5 (May 2020), p. 054005. ISSN: 2331-7019. DOI: 10.1103/PhysRevApplied.13.054005. URL: http://arxiv.org/abs/1911.10709 (visited on 05/28/2022).

[27] M. Lapointe-Major et al. “Algorithm for automated tuning of a quantum dot into the single-electron regime”. en. In: *Physical Review B* 102.8 (Aug. 2020), p. 085301. ISSN: 2469-9950, 2469-9969. DOI: 10.1103/PhysRevB.102.085301. URL: https://link.aps.org/doi/10.1103/PhysRevB.102.085301 (visited on 05/28/2022).

[28] L. A. F. Fernandes and M. M. Oliveira. “Real-time line detection through an improved Hough transform voting scheme”. In: *Pattern Recognition* 41.1 (2008), pp. 299–314. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2007.04.003. URL: https://www.sciencedirect.com/science/article/pii/S0031320307001823.

[29] C. Akinlar and C. Topal. “EDLines: A real-time line segment detector with a false detection control”. In: *Pattern Recognition Letters* 32.13 (2011), pp. 1633–1642. ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2011.06.001. URL: https://www.sciencedirect.com/science/article/pii/S0167865511001772.

[30] J. P. Zwolak et al. “QFlow lite dataset: A machine-learning approach to the charge states in quantum dot experiments”. In: *PLOS ONE* 13.10 (Oct. 2018), e0205844. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0205844. URL: http://arxiv.org/abs/1809.10018 (visited on 02/21/2022).

[31] T. Hangleiter. “Simulation and Automated Analysis of Quantum Dot Charge Diagrams”. English. BA thesis. Feb. 2017.

[32] Coherent Spintronics Group of University of Waterloo. *Quantum Dots in Python (QuDiPy*. URL: https://github.com/mainCSG/QuDiPy/tree/4a8e1751e55d37c4ec596f1581b3f9bde54d759f (visited on 05/28/2022).

[33] L. P. Kouwenhoven et al. “Electron Transport in Quantum Dots”. In: *Mesoscopic Electron Transport*. Ed. by L. L. Sohn, L. P. Kouwenhoven, and G. Schön. Dordrecht: Springer Netherlands, 1997, pp. 105–214. ISBN: 978-90-481-4906-3 978-94-015-8839-3. DOI: 10.1007/978-94-015-8839-3_4. URL: http://link.springer.com/10.1007/978-94-015-8839-3_4 (visited on 07/12/2022).

[34] *RF Thermal Noise: Johnson-Nyquist Noise*. URL: https://www.electronics-notes.com/articles/basic_concepts/electronic-rf-noise/thermal-johnson-nyquist-basics.php (visited on 05/13/2022).

[35] *What is Shot Noise*. URL: https://www.electronics-notes.com/articles/basic_concepts/electronic-rf-noise/shot-noise-what-is.php (visited on 05/13/2022).

[36] S. Yuan et al. “The Characterization of Electronic Noise in the Charge Transport through Single-Molecule Junctions”. en. In: *Small Methods* 5.3 (Mar. 2021), p. 2001064. ISSN: 2366-9608, 2366-9608. DOI: 10.1002/smtd.202001064. URL: https://onlinelibrary.wiley.com/doi/10.1002/smtd.202001064 (visited on 05/15/2022).

[37] J. Timmer and M. Koenig. “On generating power law noise.” In: *Astronomy and Astrophysics* 300 (Aug. 1995), p. 707. ISSN: 0004-6361. URL: https://ui.adsabs.harvard.edu/abs/1995A&A...300..707T (visited on 06/08/2022).

[38] F. Patzelt. *colorednoise.py*. URL: https://github.com/felixpatzelt/colorednoise (visited on 06/07/2022).

[39] X. Chen et al. “Modeling Random Telegraph Noise as a Randomness Source and its Application in True Random Number Generation”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 35.9 (Sept. 2016), pp. 1435–1448. ISSN: 0278-0070, 1937-4151. DOI: 10.1109/TCAD.2015.2511074. URL: http://ieeexplore.ieee.org/document/7362179/ (visited on 05/13/2022).

[40] P. Virtanen et al. *scipy.optimize.minimize*. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html#scipy.optimize.minimize (visited on 09/22/2022).

[41] C. G. Broyden. “The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations”. In: *IMA Journal of Applied Mathematics* 6.1 (Mar. 1970), pp. 76–90. ISSN: 0272-4960. DOI: 10.1093/imamat/6.1.76. eprint: https://academic.oup.com/imamat/article-pdf/6/1/76/2233756/6-1-76.pdf. URL: https://doi.org/10.1093/imamat/6.1.76.

[42] P. Welch. “The use of fast Fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms”. In: *IEEE Transactions on Audio and Electroacoustics* 15.2 (1967), pp. 70–73. DOI: 10.1109/TAU.1967.1161901.

[43] P. Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[44] T. Salimans et al. *Improved Techniques for Training GANs*. June 2016. URL: http://arxiv.org/abs/1606.03498 (visited on 08/12/2022).

[45] M. Heusel et al. “GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS’17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6629–6640. ISBN: 9781510860964.

[46] A. M. Alaa et al. “How Faithful is your Synthetic Data? Sample-level Metrics for Evaluating and Auditing Generative Models”. In: *arXiv:2102.08921 [cs, stat]* (Feb. 2021). URL: http://arxiv.org/abs/2102.08921 (visited on 02/07/2022).

[47] L. Ruff et al. “Deep One-Class Classification”. In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by J. Dy and A. Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 4393–4402. URL: https://proceedings.mlr.press/v80/ruff18a.html.

[48] L. Ruff. *PyTorch Implementation of Deep SVDD*. URL: https://github.com/lukasruff/Deep-SVDD-PyTorch (visited on 08/12/2022).

[49] G. Chen, F. Zhu, and P. A. Heng. “An Efficient Statistical Method for Image Noise Level Estimation”. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 477–485. DOI: 10.1109/ICCV.2015.62.

[50] Z. Yue. *Noise Level Estimation for Signal Image*. URL: https://github.com/zsyOAOA/noise_est_ICCV2015 (visited on 09/02/2022).

[51] S. Kullback and R. A. Leibler. “On Information and Sufficiency”. In: *The Annals of Mathematical Statistics* 22.1 (1951), pp. 79–86. DOI: 10.1214/aoms/1177729694. URL: https://doi.org/10.1214/aoms/1177729694.

[52] J. Lin. “Divergence measures based on the Shannon entropy”. In: *IEEE Transactions on Information Theory* 37.1 (1991), pp. 145–151. DOI: 10.1109/18.61115.

[53] T. R. dos Reis. *Noise Level Estimation for Signal Image*. URL: https://medium.com/datalab-log/measuring-the-statistical-similarity-between-two-samples-using-jensen-shannon-and-kullback-leibler-8d05af514b15 (visited on 08/18/2022).

Jül-4444 • Januar 2024
ISSN 0944-2952

Mitglied der Helmholtz-Gemeinschaft