

Towards a Robust Framework for Analyzing Random Telegraph Signals (RTS): Application to 2-level RTS in a Semiconductor Quantum Dot

by

Tonghe Bai

A thesis
presented to the University of Waterloo
in fulfillment of the thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2024

© Tonghe Bai 2024

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

This thesis proposes a multi-stage, robust framework for analyzing Random Telegraph Signals (RTS), characterized by temporal fluctuations, within semiconductor research. The framework is tailored to applications across nanoscale solid-state technologies and addresses the growing need for precise and scalable RTS analysis as quantum and semiconductor technologies progress. Beyond the commonly used pulse-based measurements, this work focuses on extracting insights from steady-state measurements, enabling the study of non-equilibrium states—a critical aspect of understanding semiconductor phenomena. Specifically, the framework meets the demand of mitigating noise at high resolutions, as well as future support of real-time monitoring, and enabling automated tuning in devices such as quantum dots (QDs) and some nanoscale CMOS devices where RTS arises from single-carrier actions. Through stages of pre-processing, denoising, digitization and mean dwell time extraction, this method supports high-resolution analysis of RTS, addressing the complexities of experimental semiconductor data.

The framework is validated with the real-world data of a specific QD holding 2-level RTS, demonstrating a robust 20-fold resolution increase, achieving a time bin reduction from 2 μ s to 100 ns, with further explorations reaching 50 ns. This enhanced resolution uncovers hidden patterns within RTS. By accurately characterizing tunneling rates and transition dynamics, this research yields insights critical for high-fidelity quantum devices, potentially impacting applications like field-programmable gate arrays (FPGAs) and superconducting qubits, where RTS influences operational stability and performance.

Beyond immediate applications, this thesis establishes a flexible RTS processing platform adaptable across various nanoscale semiconductor technologies. Future work will explore broader integration of theoretical and experimental insights to further enhance this framework, creating a versatile toolset aimed at improving robustness and adaptability in semiconductor and quantum devices operating in environments with complex noise and temporal fluctuations.

Keywords: Random telegraph signals, signal denoising, wavelet transforms, kernel density estimation, digitization, semiconductor, quantum dot.

Acknowledgements

I would like to express my sincere gratitude to everyone whose contributions made this thesis possible.

To my parents, whose invaluable support—mentally, physically, and financially—has allowed me to stay in Canada and pursue my studies with confidence and focus.

To my supervisor, Prof. Na Young Kim, for her unwavering guidance, supervision, and countless meetings, which have shaped this project and provided a clear path forward at every stage.

To graduated master student Bowen Deng, whose foundational work laid the basis for this thesis. His insights were instrumental in my understanding and further development of this research.

To graduated master student Ayush Kapoor, for his help in preparing phase 2 pre-processing data and for implementing additional digitization variants.

Finally, we thank Hendrik Mannel, Dr. Martin Paul Geller, and Prof. Axel Lorke from the University of Duisburg-Essen, whose insightful discussions and collaboration enriched our understanding of the device from which the data were taken from. The datasets from these experiments are available upon request from the corresponding author.

Dedication

This is dedicated to the one I love.

Table of Contents

Author's Declaration	ii
Abstract	iii
Acknowledgements	iv
Dedication	v
List of Figures	x
List of Tables	xiv
List of Abbreviations	xvi
List of Symbols	xvii
1 Introduction	1
1.1 Random Telegraph Signals	2
1.1.1 Simple and Complex RTSs	3
1.1.2 Noise Types	6
1.1.3 Mean Tau Statistics	7
1.2 General Logic	7
1.3 Data and Theories	8

1.3.1	Quantum Dot Semiconductors	8
1.3.2	Resolve Spin-Up and Spin-Down Electrons	11
1.3.3	Size and Storage	12
1.3.4	Decoupling Physical Process from Detector System	14
2	Multi-Stage RTS Processing Framework	20
2.1	Pre-Processing Stage	20
2.1.1	Black Box	20
2.1.2	Variants	21
2.1.3	Performance	29
2.2	Denoising Stage	30
2.2.1	Black Box	30
2.2.2	Kernel Density Estimation (KDE)	30
2.2.3	Variants	32
2.2.4	Performance	39
2.3	Digitization Stage	40
2.3.1	Black Box	40
2.3.2	Variants	42
2.3.3	Bit Flip Error Correction	50
2.3.4	Performance	54
2.4	Tau Statistics Stage	55
2.4.1	Black Box	55
2.4.2	Variant	56
2.4.3	Performance	59
3	Results and Discussions: Phase 1	60
3.1	Goal: Approaching An Exemplar Dataset	60
3.2	Proof of Sampling Mechanisms	61
3.3	Results	63
3.4	Summary	66

4	Results and Discussions: Phase 2	68
4.1	Goal: Primary Analysis on Full Datasets	68
4.2	Algorithm Comparison	70
4.3	Validation on Synthesized Data	75
4.4	Laser and Gate Voltages Dependency Towards Tunneling Rates	77
4.5	Laser and Gate Voltages Dependency Towards Tunneling Rates Asymmetry	79
4.6	Summary	85
5	Results and Discussions: Phase 3	86
5.1	Goal: Enhanced Analysis with Dense Samples	86
5.2	Appropriate Sampling Mechanism	86
5.3	Laser and Gate Voltages Dependency Towards Asymmetry	88
5.4	Summary	92
6	Unified Algorithm and Future Work	93
6.1	Overview	93
6.2	Unique Identifiers of Different Data	93
6.2.1	Quantum Dot	94
6.2.2	Transistor	94
6.2.3	Other Semiconductors Holding Complex RTS Data	96
6.3	Phase Map	96
6.4	Algorithm Optimization	98
6.5	High-Resolution Temporal Dynamics	99
6.6	Applications	105
6.6.1	Applications in CMOS and QD Domains	105
6.6.2	Directions in Quantum Science and Technology	106
6.7	Conclusions	107
	References	108

APPENDICES	112
A Modified Force Dense Sample Method	113
A.1 Briefing	113
A.2 Algorithm	114

List of Figures

1.1	A noiseless simple 2-level RTS with the definition of Δ_{RTS} , τ_{low} , and τ_{high}	4
1.2	Exemplar simple 2-level RTSs with ground truth (black) overlayed with different noise types (colored).	5
1.3	Schematic of the QD system.	9
1.4	Tunneling rates in QD perspective (physical process).	11
1.5	Tunneling rates in photon detector perspective (detector system).	11
1.6	Exemplar activity observed by photon detector with 81 ps resolution.	13
1.7	Raw dataset only stores timestamps of detector clicks for compact storage.	13
1.8	Histogram showing the distribution of intervals between photon detections at various laser intensities V_l . V_g is fixed at 0.3520 V. Each subplot examined the first 10 seconds of the dataset.	18
1.9	Overlay of distributions of intervals between photon detections at various laser intensities V_l . V_g is fixed at 0.3520 V. Each series examined the first 10 seconds of the dataset.	19
2.1	Overview of Pre-Processor Black Box (shown as yellow blocks in future diagrams).	20
2.2	Examples of the sufficiency of dense raw data before filtering (grey) for KDE to recognize at least 2 distinct signal levels. Quality of denoising (blue) is irrelevant when RTSs are too empty.	22
2.3	Exemplar binning process for $t_{\text{bin}} = 405$ ps from stored raw data.	24
2.4	Examples of the effect of moving average (MA) on raw data (grey) into processed (blue). Both subplots use the same raw data lasting for 0.1 ms but binned with different t_{bin} .	27

2.5	Overview of Denoiser Black Box (shown as green blocks in future diagrams).	30
2.6	Exemplar denoising and KDE process of a QD RTS sample.	31
2.7	Visualization of DTCWT threshold levels. Exemplar noisy signal (grey) overlaid with the denoised signals at threshold 1.5 (blue) and threshold 3.5 (dark blue).	38
2.8	Reconstructed/Denoised QD RTS (orange) overlaid with the raw RTS (blue) using all thresholded highpass layers in Algorithm 6.	38
2.9	Overview of Digitizer Black Box (shown as blue blocks in future diagrams).	40
2.10	Visualization of digitization on denoised 2-trap RTS.	41
2.11	Digitization variants visualized with RTS (blue) and digitized RTS (red). Thresholding (top left) as baseline method [1] assign states by comparison with a threshold (dotted line). Peak matching (top right) assigns states by signal proximity to computed KDE levels (dotted lines). Hidden markov model (bottom left) does not require KDE and does not require the signal to be denoised. Bayesian (bottom right) probabilistically assigns states to computed KDE levels (dotted lines) but based on neighboring information.	43
2.12	Error possibilities and classifications in the digitization process with denoised RTS (blue) and digitized RTS (red). True positive (top left) correctly identifies state transitions when there is no error. True negative (bottom left) correctly identifies error and corrected. False positive (top right) incorrectly identifies error, false alarm. False negative (bottom right) unable to detect and missed the error.	51
2.13	Overview of Tau Statistics Extractor Black Box (shown as purple blocks in future diagrams).	55
2.14	Example tau statistics fitting of a QD RTS sample with $V_l = 50$ nV, $V_g = 0.3743$ V, and $t_{\text{bin}} = 10$ μs	59
3.1	Method used to obtain QD tunneling rates by J. Kerski <i>et al.</i> [1] mapped in our framework. Pre-processing stage and variants (orange), denoising stage and variants (green), digitization stage and variants (blue), and tau statistics extraction (purple).	61
3.2	Method used in Phase 1 to study QD RTSs statistics in our framework. Thresholding chain (top path), wavelet chain (middle path), and autoencoder chain (bottle path).	63

3.3	Time lag plot of a portion of a dataset with $V_l = 50$ nV and $V_g = 0.3743$ V under $t_{\text{bin}} = 10$ μs . Noisy RTS as grey, digitized RTS as blue (left). Time lag plot based on noisy RTS (middle). Time lag plot based on digitized RTS (right).	65
3.4	Time lag plot of a portion of a dataset with $V_l = 50$ nV and $V_g = 0.3743$ V under $t_{\text{bin}} = 100$ μs binning. Noisy RTS as grey, digitized RTS as blue (left). Time lag plot based on noisy RTS (middle). Time lag plot based on digitized RTS (right)	66
4.1	Total photon detector counts per seconds (cps) for each dataset for each 20 V_g (as scatters along a series) and each 4 V_l (as series).	70
4.2	Methods used in Phase 2 to perform algorithm comparison by QD RTSs statistics in our framework. moving average (MA) chain (top path), wavelet chain (middle path), and autoencoder chain (bottle path). 4 different colors of blocks from left to right represent the 4 stages in order, namely pre-processing, denoising, digitization, and tau extraction.	71
4.3	Method used in Phase 2 to study QD RTSs statistics in our framework. Wavelet chain by DTCWT denoising.	72
4.4	A single randomly sampled QD RTS at $t_{\text{bin}} = 1$ μs . Tested with various pre-processing MA window size and DTCWT wavelet threshold (TH) levels.	73
4.5	Validation of effectiveness on proposed method compared with community method as the baseline on synthesized 2-level quantum dot (QD) RTS with interval of $t_{\text{bin}} = 10$ μs . Raw noisy RTS in grey, denoised RTS in blue, digitized RTS in red.	76
4.6	Scatter plot with linear regression fits for Γ_{in} and Γ_{out} vs V_g , across available t_{bin} results (legend series) for different V_l (subplots).	78
4.7	Scatter plot with linear regression fits of slope for Γ_{out} (left) and Γ_{in} (right) vs V_l	78
4.8	Scatter plot of asymmetry (offset to positives) vs V_g with fitted Fermi-Dirac (Equation 4.3). All t_{bin} and all V_l are grouped together.	81
4.9	Scatter plot of asymmetry (offset to positives) vs V_g with fitted Fermi-Dirac (Equation 4.3). All t_{bin} are grouped together. All V_l are grouped into subplots.	82
4.10	Scatter plot of asymmetry (offset to positives) vs V_g with fitted Fermi-Dirac (Equation 4.3). All t_{bin} are grouped into subplots (top to bottom). All V_l are grouped into subplots (left to right).	83

5.1	Method used in Phase 3 to study QD RTSs statistics in our framework. Changed from random sampling to force dense sample in contrast to Figure 4.3 in Phase 2.	89
5.2	Asymmetry (offset to positives) vs V_g with fitted Fermi-Dirac (Equation 4.3) for all V_l at $t_{\text{bin}} = 1 \mu\text{s}$	90
5.3	Box Plot of Asymmetry (offset to Positive) vs V_g with Fermi-Dirac Fit on median Asymmetry values for each V_l at Time Bin = $1 \mu\text{s}$, showing 95% confidence interval.	92
6.1	Raw (noisy) RTS samples from various available transistors.	95
6.2	A simple, conceptual phase map by categorizing RTSs based on white and pink noise intensities.	97
6.3	Digitization of $t_{\text{bin}} = 1 \mu\text{s}$ RTS sample specified in Table 6.1, only 10000 steps are displayed for visualization.	102
6.4	Method used in initial look towards high-resolution QD RTSs statistics in our framework.	103
6.5	Tau statistics fitting of QD RTS samples from dataset with $V_g = 0.3520 \text{ V}$, $V_l = 200 \text{ nV}$, and $t_{\text{bin}} = 100 \text{ ns}$ and 50 ns	104

List of Tables

1.1	Results across exponential bins on intervals between photon detections at $V_l = 50$ nV.	17
2.1	Pre-Processing Variants and Corresponding Execution Arguments.	21
2.2	Denoising Variants and Corresponding Execution Arguments.	33
2.3	Digitization Variants and Corresponding Execution Arguments.	43
2.4	Error in $\bar{\tau}_{\text{high}}$ and $\bar{\tau}_{\text{low}}$ with varying max filter widths.	53
2.5	Tau Extractor Variants and Corresponding Execution Arguments.	56
3.1	$\bar{\tau}_{\text{low}}$ and $\bar{\tau}_{\text{high}}$ of an exemplar time binned RTSs. No additional pre-processing besides time binning. DTCWT denoised and peak matching digitization, bit flip error correction with max error window size of 4.	62
4.1	RTS sampling specifications used for Phase 2 analysis.	69
4.2	Phase 2 RTS sample validity rate according to sampling specifications in Table 4.1.	77
4.3	Coefficients c_1 , c_2 , and c_3 fitted from Fermi-Dirac (Equation 4.3), for different combinations of V_l and t_{bin} values.	84
5.1	Average switching count per 1 K rows for RTS sample length of 300 K to 3 M rows used for Phase 2.	87
5.2	RTS sampling specifications used for Phase 3 analysis.	88
5.3	Phase 3 RTS sample validity rate according to sampling specifications in Table 5.2, in contrast to Phase 2.	89

6.1	RTS sampling specifications used for high resolution analysis from dataset with $V_g = 0.3520$ V, $V_l = 200$ nV.	100
-----	--	-----

List of Abbreviations

aRTS anomalous random telegraph signal 4, 41

DAEUBL denoising autoencoder model based on U-Net and bidirectional long short-term memory 33, 34, 39, 48, 54, 61, 64, 66, 71, 72, 93, 97–99

DTCWT dual-tree complex wavelet transform xi, xii, xiv, 1, 35–39, 48, 52, 54, 61, 62, 64, 66, 67, 71–74, 97, 99

HMM hidden markov model xi, 43, 46–48, 50

KDE kernel density estimation x, xi, 21, 22, 28, 30–32, 40–46, 50, 94, 97

MA moving average x, xii, 27, 28, 32–34, 39, 54, 62, 71–74

MSE mean squared error 25

nRTS normal random telegraph signal 4, 41

QD quantum dot x–xiii, 1, 3, 8–12, 14, 15, 21, 23, 24, 26, 29, 31, 32, 34, 38, 40, 48, 50, 52–56, 58–61, 63–68, 71–73, 75, 76, 79–81, 84, 89, 91, 93, 94, 96, 98, 103–107

RTS random telegraph signal x–xv, 1–8, 10, 13–15, 21–52, 54–69, 71–77, 81, 84, 86–89, 93–98, 100–107, 113, 114

List of Symbols

- Δ_{RTS} Transition amplitude [x](#), [2–4](#), [7](#), [27](#), [28](#), [40–42](#), [44](#), [55](#), [65](#), [72](#), [75](#)
- $\bar{\tau}_{\text{high}}$ Mean dwell time at high state [xiv](#), [3](#), [4](#), [7](#), [8](#), [10](#), [23](#), [25](#), [50](#), [52–58](#), [62](#), [77](#), [89](#), [101](#), [104](#)
- $\bar{\tau}_{\text{low}}$ Mean dwell time at low state [xiv](#), [3](#), [4](#), [7](#), [8](#), [10](#), [23](#), [25](#), [52–58](#), [62](#), [77](#), [89](#), [101](#), [104](#)
- τ_{high} Dwell time at high state [x](#), [2](#), [4](#), [7](#), [31](#), [50](#), [56–58](#), [76](#), [87](#), [101–105](#)
- τ_{low} Dwell time at low state [x](#), [2](#), [4](#), [7](#), [31](#), [54](#), [56–58](#), [76](#), [87](#), [101–105](#)
- t_{bin} Time bin [x–xiv](#), [12](#), [16](#), [21–29](#), [37](#), [52](#), [53](#), [58–66](#), [68–78](#), [81–91](#), [94](#), [98–104](#), [107](#), [113](#), [114](#)
- Γ_{in} Tunneling in rate [xii](#), [10](#), [60](#), [63](#), [64](#), [75](#), [76](#), [78](#), [79](#), [81](#), [89](#), [101–103](#), [105](#)
- Γ_{out} Tunneling out rate [xii](#), [10](#), [60](#), [63](#), [64](#), [75](#), [76](#), [78](#), [79](#), [81](#), [89](#), [101–105](#)
- V_g Gate voltage [x–xiii](#), [xv](#), [9](#), [10](#), [13](#), [15](#), [18](#), [19](#), [52](#), [59](#), [60](#), [65](#), [66](#), [68–70](#), [74](#), [77–79](#), [81–83](#), [86](#), [88–92](#), [99](#), [100](#), [104](#)
- V_l Voltage directly proportional to laser excitation power [x–xv](#), [9](#), [10](#), [13](#), [15–19](#), [52](#), [59](#), [60](#), [65](#), [66](#), [68–70](#), [74](#), [77](#), [78](#), [81–84](#), [86](#), [88–92](#), [99](#), [100](#), [104](#)

Chapter 1

Introduction

In semiconductor device research, characterizing and managing noise processes is crucial, as these fluctuations can inherently impact device sensitivity and overall performance. This work develops a novel framework for systematically analyzing noise-driven phenomena within real semiconductor devices, particularly using [quantum dot](#) as an example. Rather than merely adapting existing noise analysis techniques, this framework introduces a structured methodology that is robust and adaptable to signals in various settings. This study demonstrates a comprehensive evaluation of the given [random telegraph signal](#) data, underscoring the framework’s utility in practical semiconductor analysis.

The central innovation of this project lies in the development of the multi-stage analytical pipeline, where each stage is tailored to address specific complexities within the signal data. The methodology begins with a suite of pre-processing techniques targeting different needs for preparing the sample based on the raw dataset, followed by denoising approaches such as [dual-tree complex wavelet transform](#) based denoising. This wavelet-based denoising, alongside methods like denoising autoencoder [2], allows for a more nuanced removal of noise artifacts. The denoised signals then may be digitized and used for key parameter extractions. These refined processes not only enhance signal clarity but also establish a stronger foundation for detecting [random telegraph signal](#) patterns.

Beyond functionality, the framework is designed modularly for continuous work in scalability, with several pathways for optimization, including algorithmic enhancements and the potential migration to GPU-based kernels. This versatility is a significant step toward building a high-speed, high-accuracy analysis solution suitable for both experimental and

commercial semiconductor applications. Looking ahead, the project envisions an adaptive phase map capable of analyzing diverse real semiconductor [random telegraph signal](#) data autonomously, positioning this framework as a promising tool for semiconductor noise analysis and, ultimately, as a robust platform for advancing quantum device studies.

1.1 Random Telegraph Signals

[Random telegraph signals \(RTSs\)](#) are time-fluctuating signals characterized by discrete switching events between two or more well-defined states, resulting from single-particle actions within the device. These fluctuations occur randomly over time, producing a pattern of abrupt transitions between multiple distinct signal levels. The two main components of an [RTS](#) are the high and low dwell times, τ_{high} and τ_{low} , representing the duration the signal remains in each state before transitioning. The difference in amplitude between these states is commonly referred to as Δ_{RTS} . Since [RTSs](#) arise from single-particle dynamics, they are prominently observed in small devices, such as sub-micron channel-sized field-effect transistors (FETs), where individual particle interactions can notably impact the signal.

[RTSs](#) are time-fluctuating signals characterized by a series of discrete switching events between two or more well-defined states. These fluctuations occur randomly over time, leading to a pattern of abrupt transitions between multiple distinct levels of the signal. The two main components of an [RTS](#) are the high and low dwell times, denoted as τ_{high} and τ_{low} , respectively. These represent the duration for which the signal remains at the higher or lower state before switching to the other. The difference in amplitude between these two states is commonly referred to as Δ_{RTS} .

[RTSs](#) are a fundamental type of low-frequency stochastic fluctuations that can be observed across numerous fields, including physics (semiconductors, condensed matter), chemistry (kinetics, electrochemistry), biology (cellular, neuroscience), engineering (signal processing, sensors), finance (market models) [3, 4]. For instance, in electronic systems, [RTSs](#) are typically associated with charge trapping and de-trapping processes in semiconductor devices, while in biological systems, they can represent random transitions in molecular or cellular processes.

These signals are ubiquitous in various devices and systems, by analyzing the switching behavior in [RTSs](#) and extracting parameters such as $\bar{\tau}_{\text{high}}$, $\bar{\tau}_{\text{low}}$ and Δ_{RTS} , researchers can gain insight into the fundamental dynamics governing of single particles in these systems. Accurate measurement and analysis of [RTSs](#) are therefore essential for uncovering the source of these fluctuations and interpreting their broader implications across various fields, including the dynamics of the switching process, noise sources, and model of the temporal behavior of various systems. For this reason, [RTSs](#) are not only a diagnostic tool but also an avenue for investigating the fundamental noise characteristics of devices, particularly in semiconductor research.

1.1.1 Simple and Complex RTSs

[RTSs](#) can be categorized as simple or complex, depending on the number of signal levels they exhibit. In the simplest case, [RTSs](#) display two-level transitions, where a system alternates between two well-defined states. This type of signal is often associated with a single trap influencing the system's behavior. For example, in a small electronic device, a single carrier can stochastically move between a high and a low state, with characteristic dwell times $\bar{\tau}_{\text{high}}$ and $\bar{\tau}_{\text{low}}$, which follow Poisson statistics. These simple two-level [RTSs](#) are commonly observed in devices like metal-oxide-semiconductor field-effect transistors (MOSFETs) [5], transistors [6], complementary metal-oxide semiconductor (CMOS) image sensors, and [QD](#) semiconductors [1].

On the other hand, complex [RTSs](#) involves more than two distinct levels, corresponding to the presence of multiple traps. In such cases, the number of observable levels increases exponentially with the number of traps, as given by the relationship in [Equation 1.1](#). Here, N_{traps} represents the number of traps involved in the system, and N_{levels} is the number of distinguishable signal levels. If all levels are mutually exclusive, this represents the maximum signal levels.

$$N_{\text{levels}} = 2^{N_{\text{traps}}}. \quad (1.1)$$

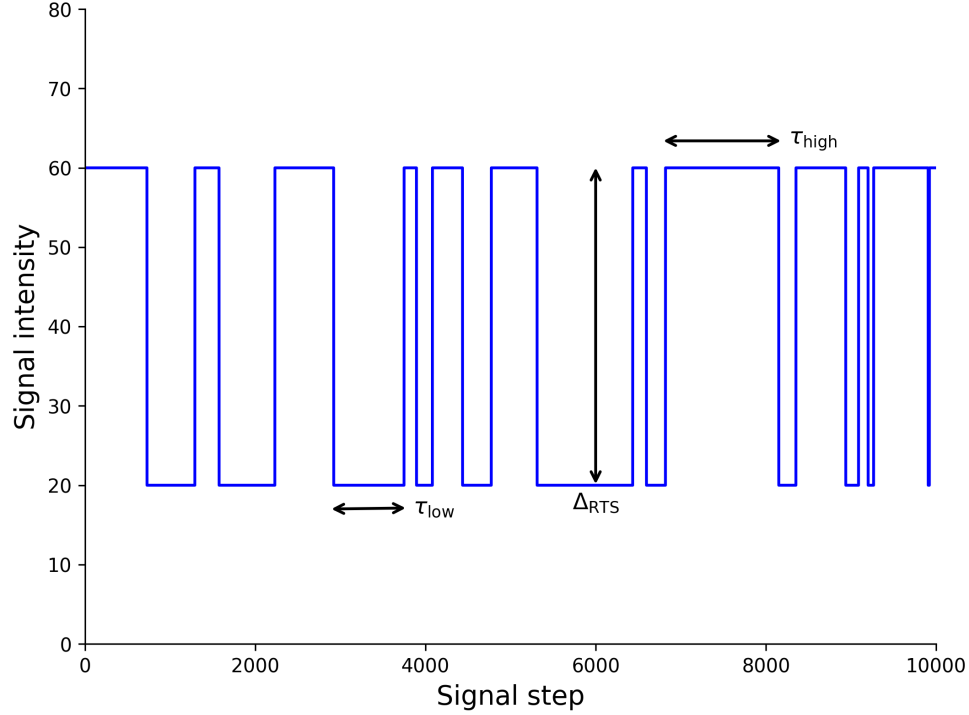
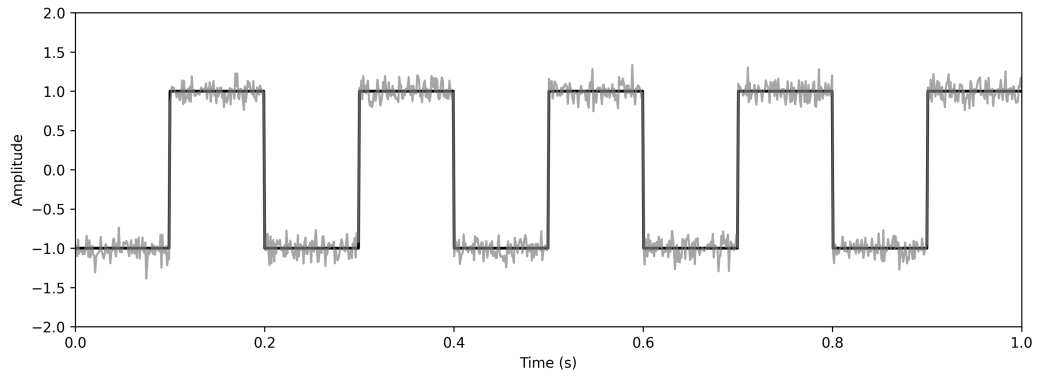
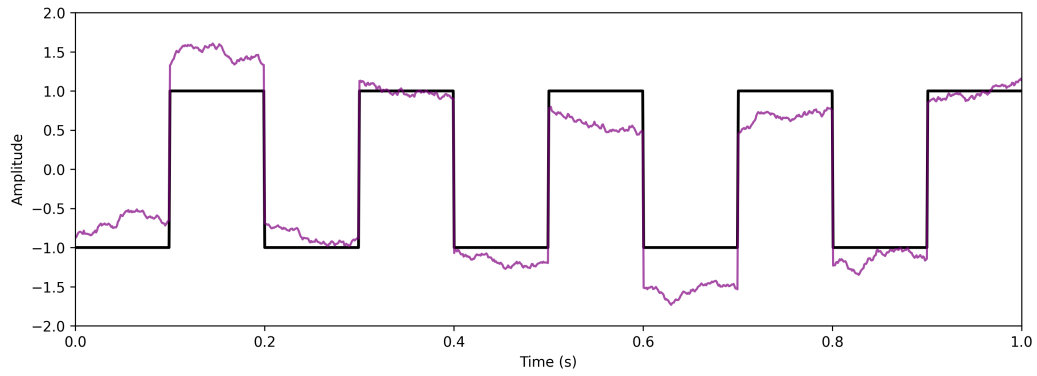


Figure 1.1: A noiseless simple 2-level RTS with the definition of Δ_{RTS} , τ_{low} , and τ_{high}

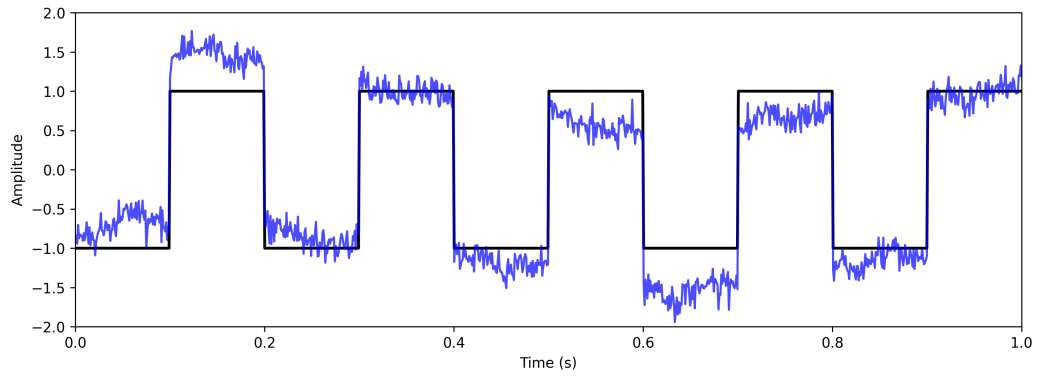
In this context, it is also important to distinguish between two main types of random telegraph signals: normal random telegraph signal and anomalous random telegraph signal. normal random telegraph signal refers to the typical response seen when multiple independent entities (traps) contribute to the signal, each behaving as an individual two-level signal with unique parameters for $\bar{\tau}_{\text{high}}$, $\bar{\tau}_{\text{low}}$ and Δ_{RTS} . Here, the combined output is a straightforward linear superposition of each individual random telegraph signals, making it simpler to analyze. On the other hand, anomalous random telegraph signal involves complex interactions or correlations between traps, leading to entangled signal levels and intricate dynamics that complicate straightforward analysis. For this thesis the real semiconductor random telegraph signals data to be applied exhibit single-trap RTSSs, eliminating the need to differentiate between normal random telegraph signal and anomalous random telegraph signal. Thus, all subsequent references to RTSSs pertain specifically to normal random telegraph signal. This restriction allows us to simplify the analysis by focusing on independent trap dynamics without needing to account for the additional complexities introduced by multi-trap or correlated behavior.



(a). RTS with white Gaussian noise only



(b). RTS with pink noise ($1/f$ noise) only



(c). RTS with all frequency noise combined

Figure 1.2: Exemplar simple 2-level RTSs with ground truth (black) overlaid with different noise types (colored).

While analyzing two-level [RTSs](#) is relatively straightforward, the complexity increases significantly when multiple traps are involved. The presence of multiple traps can lead to intricate switching patterns and may introduce additional fluctuations, making signal analysis more challenging. Understanding complex [RTSs](#) requires advanced techniques to differentiate between various states and account for unwanted disturbances. Substantial work has been done on synthetic complex [RTSs](#) explored in studies such as [\[7\]](#), and [\[8\]](#) [\[9\]](#) [\[10\]](#) [\[11\]](#) on real multi-level [RTSs](#). However, as we approach real semiconductor data in this study, we began with simple two-level [RTSs](#) from the datasets used by [\[1\]](#), focusing on more basic transitions before gradually progressing to more complex scenarios.

1.1.2 Noise Types

White Noise: White noise is characterized by a constant power spectral density across all frequencies, meaning it affects all frequency ranges equally and has no low-frequency, slowly varying components, but fluctuating amplitudes exist. It has a Gaussian amplitude distribution with no correlation between consecutive data points, often arising from thermal noise within electronic systems due to the random motion of charge carriers. As shown in [Figure 1.2 \(a\)](#), white noise obscures the [RTS](#) signal by adding rapid, uncorrelated fluctuations as a constant noise above a certain f_c .

Pink Noise: Pink noise, or $1/f$ noise, differs from white noise in that its power spectral density is inversely proportional to frequency, leading to more noticeable low-frequency components ($< f_c$) along with fluctuating amplitudes. This noise type is often observed in electronic devices at low frequencies and is typically associated with flicker noise. It can be attributed to slow, stochastic processes like charge trapping, which significantly impact [RTS](#) signals, as illustrated in [Figure 1.2 \(b\)](#).

All frequency noise combined: When all frequency noises are combined, as shown in [Figure 1.2 \(c\)](#), they create a complex noise profile with both high-frequency fluctuations (white noise) and low-frequency component (pink noise). This blended noise spectrum, which affects frequencies both above and below a critical cutoff f_c , introduces varying noise dynamics across the entire frequency range. For simple two-level [RTS](#), the higher noise fluctuations remain manageable, as small deviations may still align with the high-level state, as seen in [Figure 1.2 \(b\)](#) from approximately 0.1–0.2 seconds. However, for more complex, multi-level [RTS](#), distinguishing between signal levels becomes significantly

more challenging due to the masking effect of noise, which can obscure true transitions and complicate state identification.

1.1.3 Mean Tau Statistics

Tau statistics in the context of **RTSs** refer to the statistical analysis of the time intervals, or dwell times, that a signal spends in different discrete states before making a stochastic transition to another state. Traps of **RTSs** are characterized by rapid, random transitions between two discrete levels, often called “high” and “low” states, which are separated by a transition amplitude Δ_{RTS} . In this study, besides Δ_{RTS} , two other key parameters are of interest, annotated in [Figure 1.1](#):

- τ_{low} : This represents the time duration the signal remains in the high state before transitioning to the low state.
- τ_{high} : This is the time the signal remains in the low state before jumping back to the high state.

The mean dwell times in the high and low states, denoted as $\bar{\tau}_{\text{low}}$ and $\bar{\tau}_{\text{high}}$, are determined by averaging the individual τ_{low} and τ_{high} values over the entire signal trace. These dwell times follow a Poisson distribution, which reflects the memoryless nature of the process, meaning that the probability of transitioning to another state is independent of how long the signal has been in the current state [\[12\]](#).

RTSs analysis methods focus on extracting quantitative values of three key parameters: Δ_{RTS} , $\bar{\tau}_{\text{low}}$, and $\bar{\tau}_{\text{high}}$. These values are typically derived from digitized **RTS** by analyzing the dwell times in each state. One challenge in accurately estimating these parameters is the presence of unwanted noise, such as white Gaussian or pink noise, which can obscure the underlying **RTS**.

1.2 General Logic

The analysis of **RTS** presented in this thesis follows a systematic approach that can be conceptualized as a black-box staged process. This methodology allows for a clear separation of different stages of analysis while providing flexibility in handling various datasets and noise characteristics.

Initially, the raw signal is processed through a series of pre-processing steps. These steps aim to extract meaningful information from the original signal, resulting in a time series of amplitudes that closely resemble typical [RTS](#) traces. For more details on the pre-processing techniques, see [Sec. 2.1 Pre-Processing Stage](#).

Following pre-processing, the denoising stage focuses on minimizing the impact of pink and white noises, enabling a more accurate representation of the [RTS](#). This stage is crucial for distinguishing between true signal transitions and noise-induced fluctuations. For further insights into the denoising methods, refer to [Sec. 2.2 Denoising Stage](#).

The subsequent step involves digitizing the processed signal, where the continuous amplitude values are converted into discrete states. This digitization process is essential for quantitative evaluation and for the application of statistical methods. More detailed information about the digitization approach can be found in [Sec. 2.3 Digitization Stage](#).

Lastly, key parameters of interest in [RTS](#) analysis include $\bar{\tau}_{\text{high}}$, and $\bar{\tau}_{\text{low}}$ to be extracted by statistical tau fitting. For a comprehensive discussion on these tau statistics, see [Sec. 2.4 Tau Statistics Stage](#). By utilizing these tau statistics, it becomes possible to investigate device-specific behaviors, such as asymmetry plots for [QD](#) semiconductors. For additional details on asymmetry plots and their significance, refer to [Chapter 4 Results and Discussions: Phase 2](#).

1.3 Data and Theories

1.3.1 Quantum Dot Semiconductors

In semiconductors [quantum dot](#), [RTSs](#) often arise due to charges tunneling in and out of the [QD](#), which cause fluctuations in conductance or current as electron charges randomly tunnel in and out over time. Analyzing these signals provides insights into fundamental processes such as single electron tunneling and its transport statistics. J. Kerski *et al.* [1] discuss the unique dynamics of QDs, emphasizing their use in understanding quantum behaviors. While we had the option to explore other data sources, such as transistors, we chose to specialize in [QD](#) first as we started applying our methods to real semiconductor data. This is also largely due to our collaboration with A. Lorke’s German research group

has existing expertise in studying QD dynamics, giving us a strong reference point for comparison. The schematic of the QD system is given in Figure 1.3.

In the QD system, the QD (middle layer) is excited by a tuneable laser with its power proportional to a voltage denoted as V_l . The charge state of the QD can be tuned by applying a voltage V_g to the gate diode (top layer) that sets the ground state of the QD in resonance with the charge reservoir diode (bottom layer). Upon electron transitions in/out of the QD, photons will be emitted and captured by a photon detector [1], leading us to obtain the data as detector counts by timestamps.

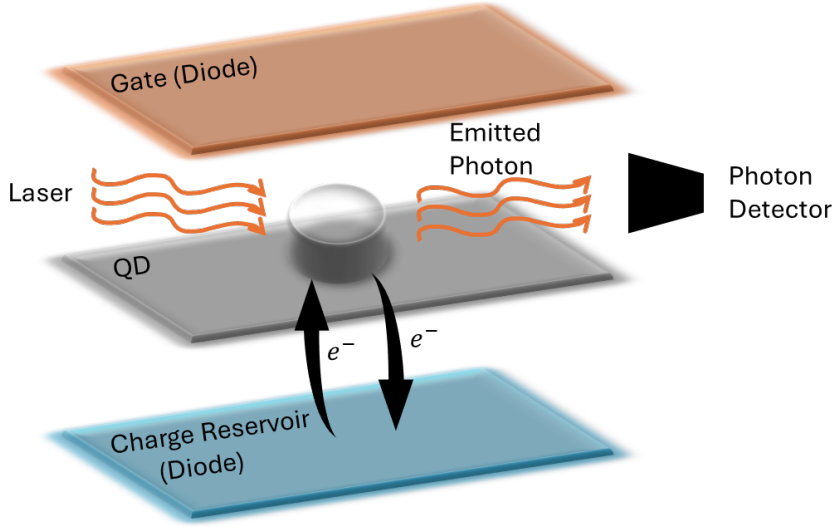


Figure 1.3: Schematic of the QD system.

Data acquisition from the QD experiments follows a specific setup. In this study, the magnetic field was kept constant. The source data is extremely dense, storing only the timestamps of each detected photon within the QD over approximately 300 seconds. With a photon detector resolution of 81 ps (a boolean detection of photon once per 81 ps) and only recording the timestamps of the on states, this dataset compresses to around 2 GB. However, if expanded to continuous stream data by both on and off states, it would balloon to approximately 60 TB, demonstrating a 30,000x increase in size. To manage this, we introduce larger time bins, such as 100 μ s, aiming to maintain as high a resolution as possible. This raises challenges related to algorithmic efficiency and file management, given the immense data volume.

Tunneling rates, defined as the rates at which electrons transition in and out of the quantum dot, offer a more accurate method for quantifying the behavior of the QD device, as they closely represent the physical electron dynamics within the system. These rates provide a clear measure of the time the QD spends in its high and low emission states. The tunneling rates (Γ_{in} and Γ_{out}) of QD are directly related to the average dwell times as a RTS ($\bar{\tau}_{\text{high}}$ and $\bar{\tau}_{\text{low}}$) spent in these states. Specifically, the tunneling rate is the inverse of the average dwell time in each state. The formulas that describe this relationship are given as Equation 1.2. Note that when the photon detector is not detecting photon transitions, RTS is in the low state, indicating Γ_{out} of the QD, and vice versa. Normally speaking, since the analysis methods are designed to output $\bar{\tau}_{\text{high}}$ and $\bar{\tau}_{\text{low}}$ statistics for any generic RTS signals, it is more common to convert the results into Γ_{in} and Γ_{out} from the perspective of the QD device.

$$\begin{aligned}\Gamma_{\text{in}} [\text{Hz}] &= \frac{1}{\bar{\tau}_{\text{high}} [\text{s}]} \\ \Gamma_{\text{out}} [\text{Hz}] &= \frac{1}{\bar{\tau}_{\text{low}} [\text{s}]}.\end{aligned}\tag{1.2}$$

QD Perspective: In the physical system, a simplified 3-state transition diagram is shown in Figure 1.4, where electrons tunnel into the QD as either spin-up or spin-down electrons (Γ_{in}). Spin-down electrons can tunnel out of the QD (Γ_{out}). While electrons can also transition between spin-up and spin-down states within the QD itself [13], this internal spin-flip is beyond the scope of this thesis and not considered in the current analysis.

Detector Perspective: In the detector system, photon emissions correspond directly to electron transitions, providing an alternative method to monitor electron activity through photon detection, as illustrated in Figure 1.5. When the detector is in the on state, it indicates that an electron has tunneled into the QD (Γ_{in}), and when the detector is in the off state, it suggests an electron has tunneled out (Γ_{out}), facilitating charge activity monitoring.

The data (hence statistics) varies based on parameters like laser excitation power, gate voltage, and magnetic fields. While magnetic fields remain fixed in this study, the laser power (V_l) and gate voltage (V_g) are varied, altering the tunneling rates and time resolution of the system. Higher laser excitation improves time resolution (by encouraging electron transitions hence detecting more emitted photons per unit time), allowing us to push the

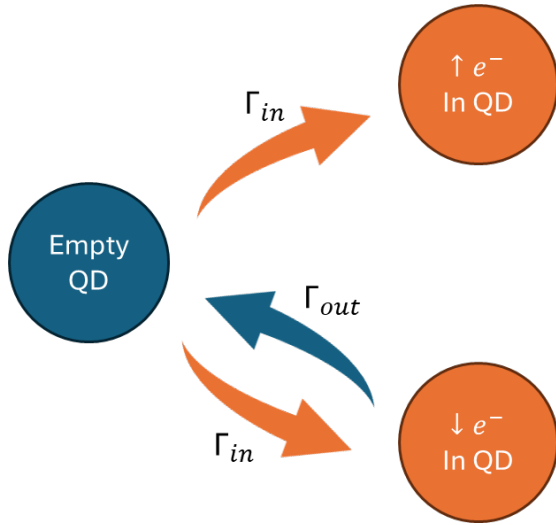


Figure 1.4: Tunneling rates in QD perspective (physical process).

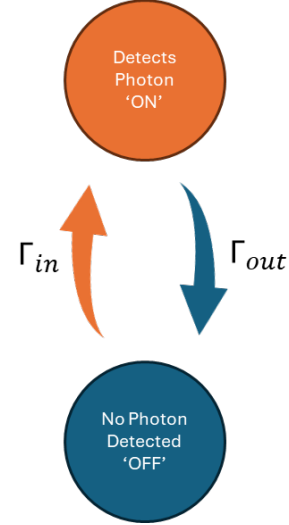


Figure 1.5: Tunneling rates in photon detector perspective (detector system).

limits of signal processing [1]. We aim to push the resolution boundaries, tackling challenges of data volume and algorithmic precision, while seeking to uncover additional insights into QD dynamics.

Within the scope of this thesis, example QD data provided by collaborating research group in Germany [1] will be used. Systematic studying of our analysis and testing methodologies will be conducted. By rigorously evaluating and refining our existing algorithmic framework on real semiconductor data, we aim to enhance its capabilities to generate high-quality results from a signal-processing perspective. Throughout this process, we will compare our findings with established understandings of the underlying physics, as well as results obtained through alternative processing techniques. This comparative analysis will not only validate the quality of our studies but will also provide insights into the effectiveness and reliability of our algorithms.

1.3.2 Resolve Spin-Up and Spin-Down Electrons

In QD, it is theoretically possible to resolve state transitions between spin-up and spin-down electrons through the application of an external magnetic field. The Zeeman effect (effect of splitting of a spectral line into several components in the presence of a static

magnetic field) causes energy level splitting between these two spin states, allowing for their potential observation. When an electron flips from spin-up to spin-down (or vice versa), this can be detected as a discrete transition in the QD’s electronic state. However, in our given dataset, we are limited to a simplified case given in Figure 1.4 and a fixed magnetic field of $B = 2$ T.

At a magnetic field strength of $B = 2$ T, it is not possible to resolve the spin-up and spin-down states of a charged quantum dot. This issue can be attributed to the relationship between Zeeman splitting energy and the system’s energy bandwidth. Specifically, if the energy level bandwidth is comparable to or larger than the Zeeman energy splitting, it becomes impossible to resolve individual spin states. As explained in [14], the problem is intensified by the temperature broadening of the electron reservoir. Additionally, at $B = 2$ T, the spin relaxation rate becomes smaller than the electron tunneling rates, as noted in [15]. This means that the electron will tunnel out of the quantum dot before we can observe a spin-flip. In this case, the tunneling rate essentially defines the bandwidth of the detector system, limiting the ability to detect the spin relaxation. If the tunneling rate does not align with the spin-flip rate, it becomes impossible to resolve individual spin states [14]. Therefore, at $B = 2$ T, we can only model the signal as a single-exponential decay (for both spin-up and spin-down electrons), where the magnetic field remains fixed throughout the phases of the study. Future research could explore the effects of varying magnetic fields to potentially resolve spin-up and spin-down states, using a bi-sectional exponential fitting model, as suggested by [13].

Previous studies, such as [1], extended the time resolution to $t_{\text{bin}} = 20 \mu\text{s}$ for reliable results. The goal of this thesis is to push for higher temporal resolution in an effort to uncover additional information that could enhance our understanding of the dynamics in QD semiconductors.

1.3.3 Size and Storage

Mentioned earlier, the photon detector tracks electron transitions in the quantum dot by recording the on states over time, as illustrated in Figure 1.6. These detected events are stored in a highly compact format, reducing the data’s storage requirements. For example, rather than storing the continuous time series as Figure 1.7 (top), only the timestamps of detected photons are saved as Figure 1.7 (bottom).

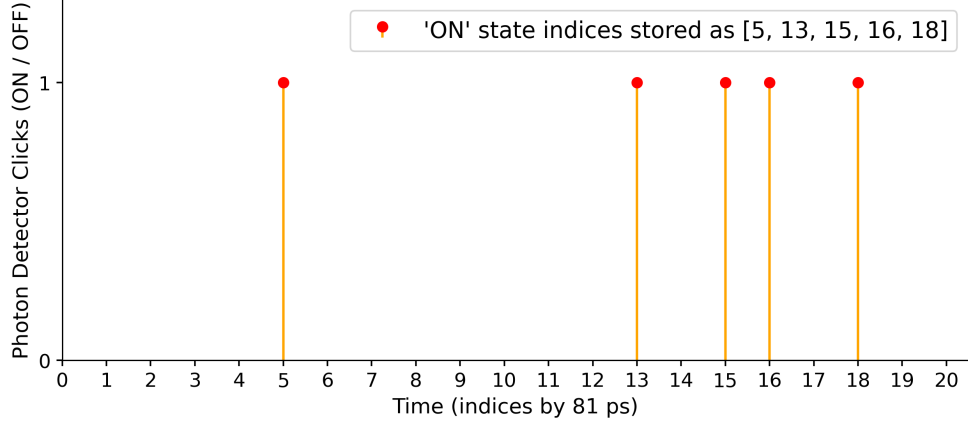


Figure 1.6: Exemplar activity observed by photon detector with 81 ps resolution.

Detector Output 81ps bins	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	1	0	1	0	0	...	amplitude
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	...	index

Text-based Storage	5	13	15	16	18	...	index
--------------------	---	----	----	----	----	-----	-------

Figure 1.7: Raw dataset only stores timestamps of detector clicks for compact storage.

This compact storage approach is crucial for managing the large datasets generated by the photon detector. For instance, consider the dataset with a V_g of 0.3743 V and V_l of 16 nV. In this case, the stored data contains 113 M rows of on states, while storing a continuous signal for the entire 308-second measurement at an 81-ps resolution would result in approximately 3.8 T rows. This method saves significant storage space, reducing a potential 65 TB dataset to just 2 GB. When we increase the time binning to 10 μ s, the dataset shrinks further. For example, binning by 10 μ s reduces the number of rows to 30 M and the file size to approximately 466 MB. The choice of time bin size—whether 10 μ s, 1 μ s, or 100 ns—significantly affects both the data size and resolution of analysis, with file sizes ranging from 466 MB to 65 TB on the entire dataset depending on binning.

For the future storage of RTSs, we propose a more efficient binary format such as HDF5, which allows for variable-sized data storage. By focusing on essential data like the amplitude (sum of counts) and using compact time representations, file size can be significantly

reduced without losing key information. For example, with a $1\ \mu\text{s}$ bin size, a [RTS](#) can be stored in less than 500 MB, in contrast to 466 MB for $10\ \mu\text{s}$ bin size with 10 times less the number of rows. Although the current algorithms are not yet optimized for handling very large datasets (limited to processing up to 300 M rows), this approach can enable efficient handling of large datasets in future studies, supporting higher time resolutions with minimal file size increases. For the [RTSs](#) storage within the scope of this thesis, we will use a simple comma-separated values (CSV) storage method for the [RTSs](#), consisting of two columns: one for time (with timestamps at constant intervals) and the other for amplitude (the sum of counts across each interval).

1.3.4 Decoupling Physical Process from Detector System

Recall the distinction between the [QD](#) perspective ([Figure 1.4](#)) and the detector perspective ([Figure 1.5](#)). In this context, some preliminary statistical analysis of the detector's off state has been conducted, corresponding to the intervals between individual photon detections.

A key component in this detection setup is the avalanche photodiode (APD) photon detector, a highly sensitive photodiode commonly used in quantum dot experiments. An APD is a specialized type of semiconductor diode designed for detecting low levels of light. Unlike conventional photodiodes, which are used in devices like solar panels, APDs work with a high voltage that boosts their sensitivity. When light hits the APD, it creates tiny charged particles, and the high voltage helps to amplify this signal, making it easier to detect even a single photon (the smallest unit of light).

Time Scales in APD Photon Detection: The specific photon detector used in the dataset has three key time scales:

1. 81 ps Resolution: This refers to the time precision with which each photon detection is recorded, enabling highly detailed tracking of photon events at fine intervals.
2. 300 ps Timing Jitter: Timing jitter represents the uncertainty in the exact time of each photon detection. Due to the inherent noise in the APD system, there is a 300 ps variability in when an event is recorded, relative to its true occurrence. This introduces a slight timing error but is less likely to deviate the analysis at much higher time bins.

3. 25 ns Dead Time: After detecting a photon, the APD requires a recovery period, known as the dead time, during which it cannot register additional photon events. This recovery time is approximately 25 ns, and during this period, any photons that arrive will not be detected, which can affect the measured photon statistics if the photon arrival rate is high.

These time scales are critical for understanding both the limitations of the APD in terms of photon detection and the accuracy of the data used to analyze the QD system. By isolating the intervals between photon detections, we can infer physical processes in the QD while also accounting for the influence of the detector’s operational characteristics.

In the raw datasets, we observe prolonged intervals of inactivity. We are interested in determining whether these gaps signify dead time or a lack of electron transitions. This observation underscores the ultimate need to move beyond random sampling of data subsets for the RTS analysis; it encourages us to focus on collecting subsamples from regions of high density in future studies. To investigate the periodic long gaps in the dataset, we will employ a statistical approach using histograms that analyze the time intervals between consecutive photon detections, specifically in determining whether the observed dead time is voltage-dependent or an intrinsic characteristic of the detector.

1. If we maintain constant excitation V_l , the dead time is likely a fixed hardware limitation of the detector.
2. If we set V_g at a constant level and examine the statistics of dead time in relation to laser intensity V_l , we can determine if the statistical variations at higher V_l (which encourage more electron transitions and thus increased detector counts) are attributable to detector response rather than the absence of physical activity (i.e., no electron transitions). This distinction allows us to differ between the physical processes and the detector’s performance.

We will choose the lowest gate voltage $V_g = 0.3520$ V, as it exhibits the most significant variation in counts per second (cps) across different laser intensities $V_l = [3.7$ nV, 16 nV, 50 nV, 200 nV], as illustrated later in Figure 4.1 later. In contrast, selecting the highest V_g yields consistently low counts despite variations in V_l . To store the differences between each consecutive timestamp in the raw dataset, considering that the dataset only records timestamps when the detector observes a photon, would be time-consuming and potentially

unnecessary. Therefore, we will analyze the first 10 seconds of these four datasets, which encompass approximately 3% of the total data. Since the physical conditions remain consistent throughout the dataset, this duration is sufficient for observing trends.

The choice of using this plot arises from the need to decide the number of bins before examining the dataset because storing all the interval sizes between every single timestamp would be computationally intensive (taking at least a week for each dataset without parallel processing). Instead, the algorithm, while iterating through timestamps in the dataset, increments the count within the appropriate predetermined bin based on interval size, reducing the amount of data writing required significantly. A log-scale x-axis was chosen due to the large range of interval sizes, covering multiple orders of magnitude (making equal-width histogram bins impossible). After multiple tests, an appropriate bin size and range were determined, with logarithmic bins spaced exponentially (powers of 10) even from 1 tick to 1 B ticks, resulting in 20 bins, with most data concentrated in the middle five bins. 1 tick corresponding to the t_{bin} at 81 ps. As an example:

- **1st bin:** From 1 tick to 2 ticks
- **11th bin:** From 32 K ticks to 89 K ticks
- **20th bin:** From 355 M ticks to 1 B ticks

The number of 20 bins was selected to ensure sufficient resolution. For the maximum bin limit, we chose 1 B ticks, as the highest count-per-second (cps) is approximately 3 M and the data collection time was 10 s, making the maximum required bin to cover 30 M. A few extra bins were added beyond 30 M to ensure both the smallest and largest bins captured all values with total counts at exactly zero. An exemplar collected results of $V_l = 50$ nV is given in Table 1.1.

Start (Closed Bracket) [ticks]	End (Open Bracket) [ticks]	Counts
1	2	0
2	7	0
7	22	0
22	63	0
63	177	0
177	501	631910
501	1412	1840474
1412	3981	3999738
3981	11220	5671581
11220	31622	2881541
31622	89125	193381
89125	251188	2059
251188	707945	679
707945	1995262	1377
1995262	5623413	1736
5623413	15848931	672
15848931	44668359	36
44668359	125892541	0
125892541	354813389	0
354813389	1000000000	0

Table 1.1: Results across exponential bins on intervals between photon detections at $V_l = 50$ nV.

For plotting in [Figure 1.8](#), ticks were converted to seconds, and the x-axis is represented in a log scale. The y-axis shows the counts of intervals between photon detections across various durations, with a fitted univariate spline to visualize the trend. As expected based on physical principles, a higher count of intervals correlates with higher V_l . To facilitate comparison, overlaying the distributions for each V_l provides a clear visualization.

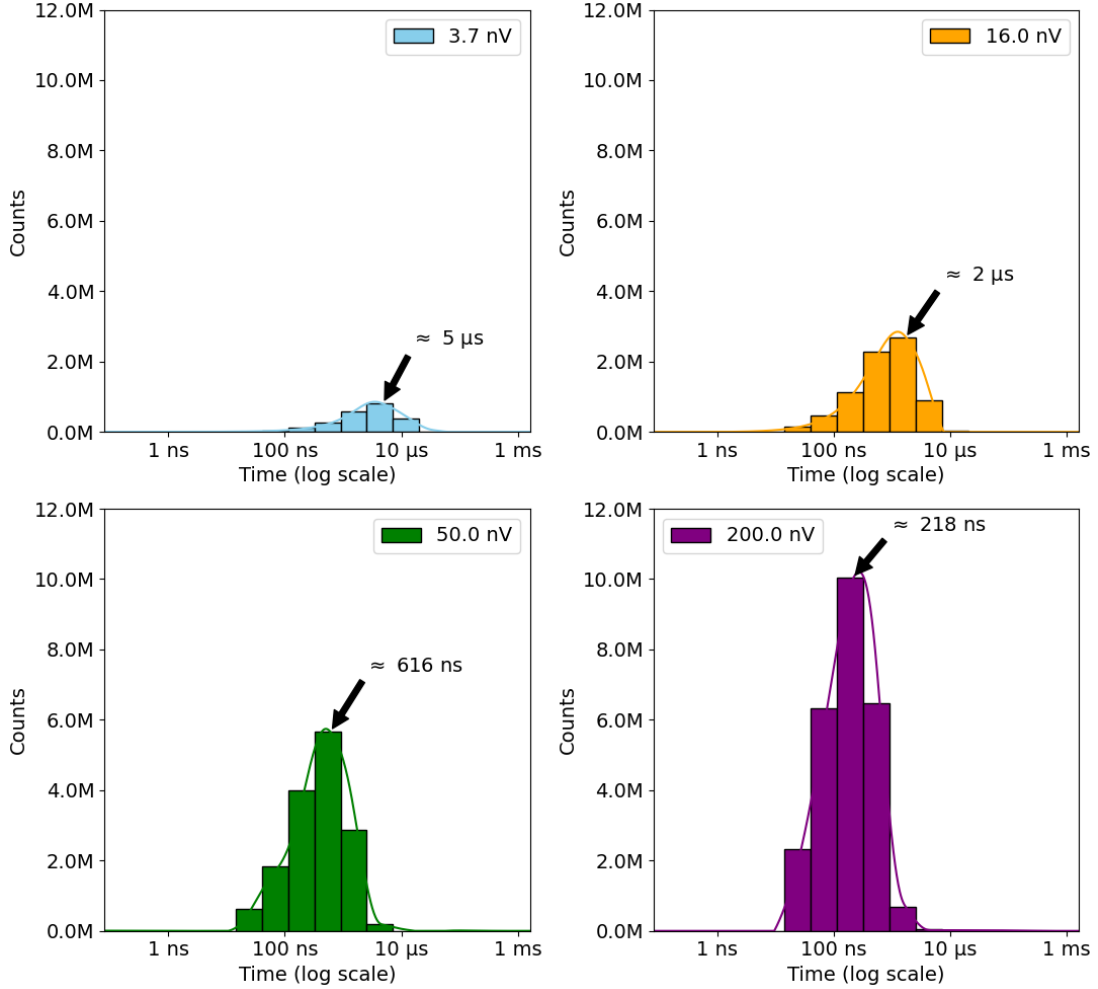


Figure 1.8: Histogram showing the distribution of intervals between photon detections at various laser intensities V_l . V_g is fixed at 0.3520 V. Each subplot examined the first 10 seconds of the dataset.

As shown in Figure 1.9, the overlay emphasizes that shorter intervals are more prevalent at higher V_l , reflecting the increased frequency of detector activity. This observation aligns with our expectations and indicates that the patterns we see are not influenced by the detector's dead time. In future work, this analysis will be repeated for different V_g values to determine whether similar trends can be observed, further validating the consistency of these patterns across varying conditions.

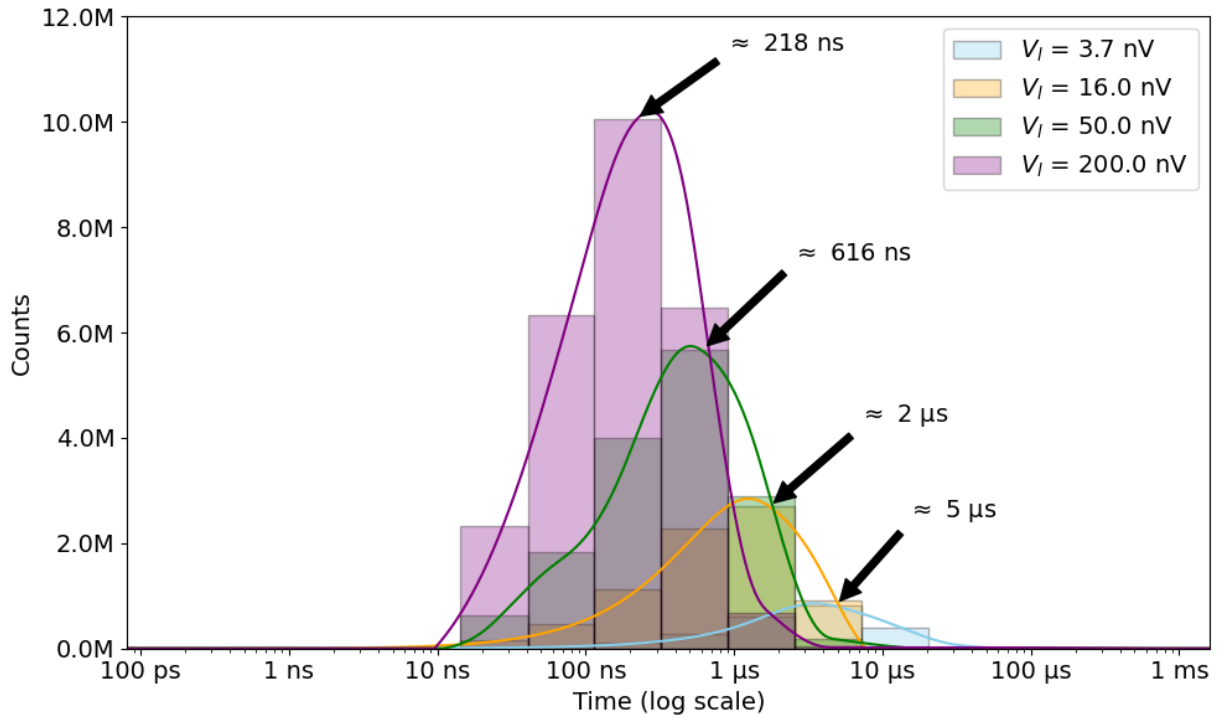


Figure 1.9: Overlay of distributions of intervals between photon detections at various laser intensities V_l . V_g is fixed at 0.3520 V. Each series examined the first 10 seconds of the dataset.

Chapter 2

Multi-Stage RTS Processing Framework

2.1 Pre-Processing Stage

2.1.1 Black Box

The idea of a black box is used for all the execution stages, where depending on the nature of the data, some stages may be optional or required to be used multiple times (i.e. multiple pre-processing actions are required). Hence, the stages are designed to intake and output data streams based on a specific description in a signal processing approach. That is, without needing any prior information such as the relevant background physics or any theory regarding the system that generated the experimental data.

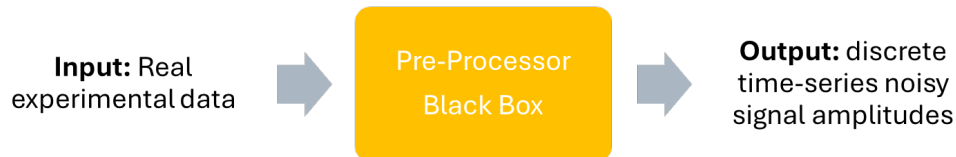


Figure 2.1: Overview of Pre-Processor Black Box (shown as yellow blocks in future diagrams).

Input: real experimental data, usually in the format of a continuous stream of signal amplitudes, although sources can be non-continuous due to data storage mechanisms as described in [Sec. 1.3.3](#).

Output: must be discrete (equally spaced time-intervals) time-series of signal amplitudes that contain sufficiently enough data for multiple [kernel density estimation \(KDE\)](#) levels (see more about [KDE](#) in [Sec. 2.2.2](#)), presence of both white and pink background noise is acceptable and expected at this stage. The emptiness of pre-processed [RTSs](#) can be visualized in [Figure 2.2](#) below. While it is possible to enforce a limit to govern the emptiness and withhold it from passing to the next stage in [Sec. 2.2 Denoising Stage](#), it was decided to continue to execute regardless and observe whether denoised [RTS](#) will be returned by denoising.

2.1.2 Variants

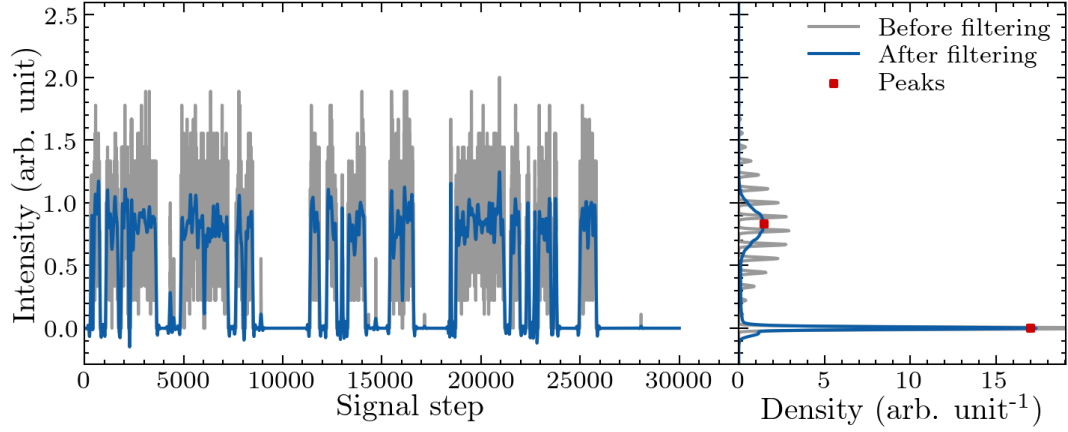
While the use of a black box is suitable for algorithm design and parameter tuning, the core methods internal to the black box contain various options to be chosen, [Table 2.1](#) below outlines the variants of pre-processing methods that will be covered next. The choice of different pre-processing techniques will be evaluated in [Chapter 3](#) and [Chapter 4](#).

Pre-Processor Variants	Execution Arguments
Time Binning	Target t_{bin}
Random Sampling	Target RTSs length (rows)
Moving Average	Window size
Force Dense Sample	Target RTSs length (rows)

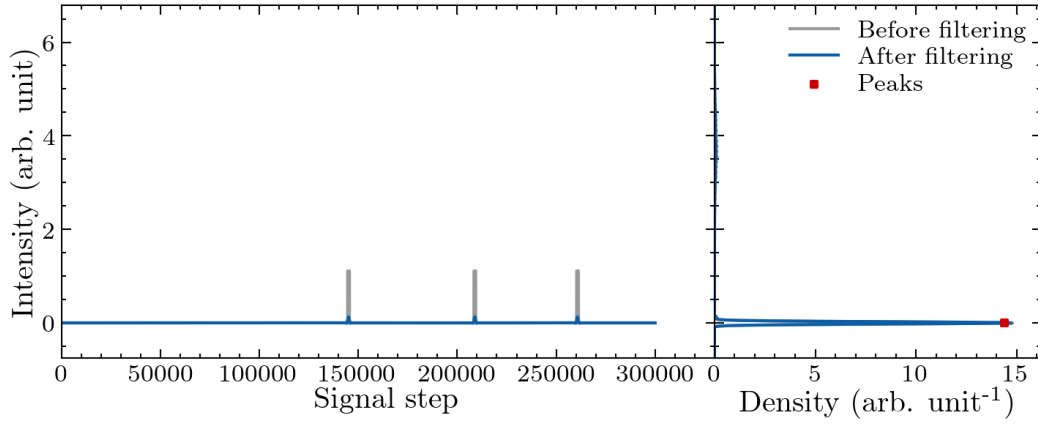
Table 2.1: Pre-Processing Variants and Corresponding Execution Arguments.

Time Binning

Time binning is one of the most crucial pre-processing actions required for data obtained from a high-sampling-frequency device, 81 ps for the [QD](#) as an example, where down-scaling of the resolution from the detector is mandatory to compact the data for analysis.



(a). Sufficiently dense, 2 KDE levels (peaks) recognized



(b). Insufficiently dense, 1 KDE level (peak) recognized

Figure 2.2: Examples of the sufficiency of dense raw data before filtering (grey) for KDE to recognize at least 2 distinct signal levels. Quality of denoising (blue) is irrelevant when RTSs are too empty.

The selection of time bin size directly impacts the sampling frequency or frequency bandwidth, that can be accessed. As an example, for a t_{bin} range from 1 ns to 100 μs , the accessible sampling frequency bandwidth is defined by the reciprocal of the time bin size. This gives a sampling frequency range as follows:

- $t_{\text{bin}} = 1 \text{ ns}$, the sampling frequency is:

$$f_{\text{sampling}} = \frac{1}{1 \text{ ns}} = 1 \text{ GHz}$$

To satisfy the Nyquist theorem, this allows us to sample frequencies up to:

$$f_{\text{max}} = \frac{f_{\text{sampling}}}{2} = 500 \text{ MHz}$$

- $t_{\text{bin}} = 100 \text{ } \mu\text{s}$, the sampling frequency is:

$$f_{\text{sampling}} = \frac{1}{100 \text{ } \mu\text{s}} = 10 \text{ kHz}$$

To satisfy the Nyquist theorem, this allows us to sample frequencies up to:

$$f_{\text{max}} = \frac{f_{\text{sampling}}}{2} = 5 \text{ kHz}$$

Thus, by setting the time bin range between 1 ns and 100 μs , we theoretically access a frequency bandwidth from 5 kHz to 500 MHz. Realistically, however, the range of time binning is constrained by some limitations. In this setup, the detector's dead time of 25 ns places an upper cap on achievable resolution, meaning that time bins finer than 25 ns would exceed the detector's capacity to capture separate photon events reliably. Furthermore, capturing meaningful [RTS](#) state transitions sets a lower cap on the resolution, as resolutions too coarse (e.g., beyond 100 μs) may miss these transitions entirely, as discussed in [Chapter 4](#). Therefore, a time bin resolution of approximately 50 ns to 50 μs is ideal for effectively monitoring [RTS](#) dynamics without feature loss or overload.

It is also essential to distinguish the sampling frequency from the [QD](#) tunneling rates, which correspond to the actual [RTS](#) state transitions. These transitions typically occur at much wider time intervals, resulting in significantly lower tunneling rates, often observed in the kHz range. The time binning, therefore, serves only as a preparatory data handling step and does not reflect the intrinsic rates of tunneling events, which are governed by quantum processes and are considerably slower. Besides this, when the same raw data are binned by different t_{bin} ; let's say the same 0.1 seconds of data binned to $t_{\text{bin}} = 100 \text{ } \mu\text{s}$ for 10K rows, compared with binned to $t_{\text{bin}} = 10 \text{ } \mu\text{s}$ for 100K rows, are essentially different [RTSs](#) that may exhibit different $\bar{\tau}_{\text{high}}$ and $\bar{\tau}_{\text{low}}$ statistics.

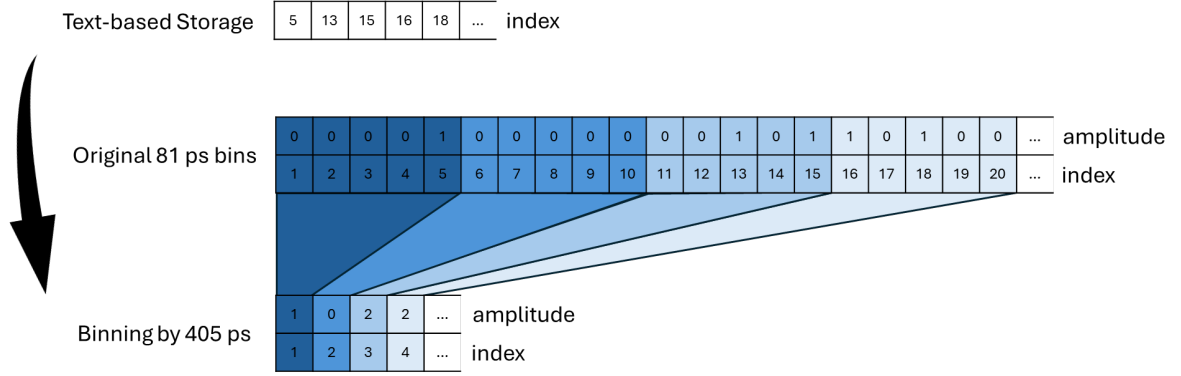


Figure 2.3: Exemplar binning process for $t_{\text{bin}} = 405$ ps from stored raw data.

Figure 2.3 demonstrates an exemplar time binning process of the QD data. Consider the detector with $t_{\text{bin}} = 81$ ps has obtained 20 consecutive data points (overall duration for about 1.6 ns). For compact storage practice as mentioned in Sec. 1.3.3, the indices with detector clicks on are stored as $\{5, 13, 15, 16, 18\}$. The time binning process is to directly bin the raw data from its text-based storage to the desired goal time bin, $t_{\text{bin}} = 405$ ps in this case, without the need for expansion to the original $t_{\text{bin}} = 81$ ps by taking up memory and storage space. The pseudocode of the time-binning process is outlined in Algorithm 1 below.

Algorithm 1 Time Binning

Input: $t_{\text{bin}} = \{1 \text{ ns}, 10 \text{ ns}, \dots, 100 \text{ us}\}$, total duration = 300 s

Output: noisy RTSS for each time bin

- 1: **for** each bin size in bin sizes **do**
 - 2: Estimate the total number of rows for the given bin size
 - 3: **for** each line in the raw data file **do**
 - 4: Parse the timestamp and calculate in seconds
 - 5: Adjust the timestamp relative to the start time and determine its bin number
 - 6: **if** bin number is the same as the previous bin number **then**
 - 7: Increment the count for the current bin
 - 8: **else**
 - 9: Write out the count for the previous bin
 - 10: Write zeros for any empty bins between the previous and current bin
 - 11: Set previous bin = current bin and count = 1
 - 12: Write out the count for the last bin and close input/output files
-

Random Sampling

While approaching higher resolutions (t_{bin} in ns), a time-binned signal for the entire raw data duration (300 s) would result in the generated RTSs being too long to be processed. As an example, full data binned to $t_{\text{bin}} = 10$ ns is about 300B rows, being unrealistic to analyze at once. Additionally, our methods towards RTS analysis do not require such long raw data in hundreds of seconds to generate results that are a good representation of the signal, rather a small portion in the range of seconds is sufficient, to be covered in [Sec. 3.2](#).

Considering the size of the raw dataset, it is unrealistic to perform the analysis as a single RTS, there are some options for taking a sub-sample, in which a random block (of the desired length) ensures that every region in the dataset has an equal chance of being selected. This is especially important as it helps in maintaining the overall characteristics of the dataset within the smaller sample. A randomly selected block also reduces the risk of introducing bias. The expectation of this approach is if the ultimate $\bar{\tau}_{\text{high}}$ and $\bar{\tau}_{\text{low}}$ statistics of the randomly sampled RTSs are giving a relatively small MSE compared with its averages, and the averages of the random samples are similar to the $\bar{\tau}_{\text{high}}$ and $\bar{\tau}_{\text{low}}$ statistics of the full signal, we would then conclude that the random sub-samples are a sufficient representation of the full signal. Hence, this approach achieves the generalizability of random sampling, allowing inferences to be made about the entire dataset with greater confidence.

The algorithm used to generate sampled data is presented in [Algorithm 2](#). The algorithm iterates through different bin sizes, estimates the number of rows for each, and determines viable sampling ratios within predefined constraints. For each viable sampling ratio, five samples are generated by selecting random seeds and calculating time ranges. The timestamp for each line in the raw data is parsed, adjusted, and binned, with the final output saved to files.

Algorithm 2 Random Sampling

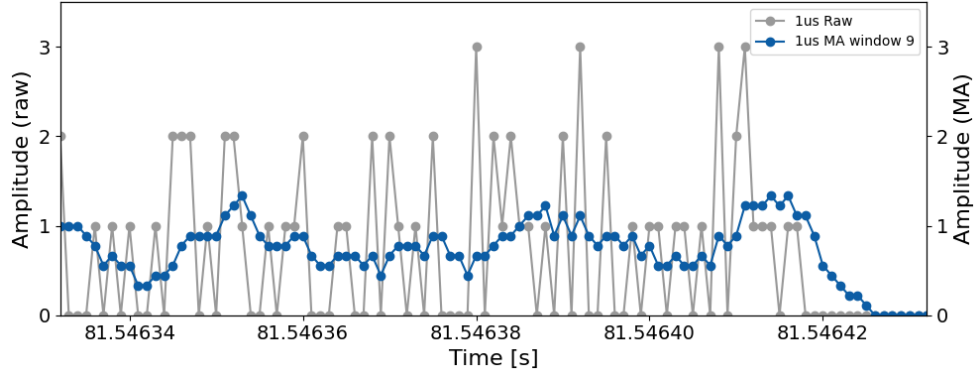
Input: total duration = 300 s, row constraints = 3K ~ 30M, number of samples = 5

Output: 5 noisy RTSs for each induced sampling ratio

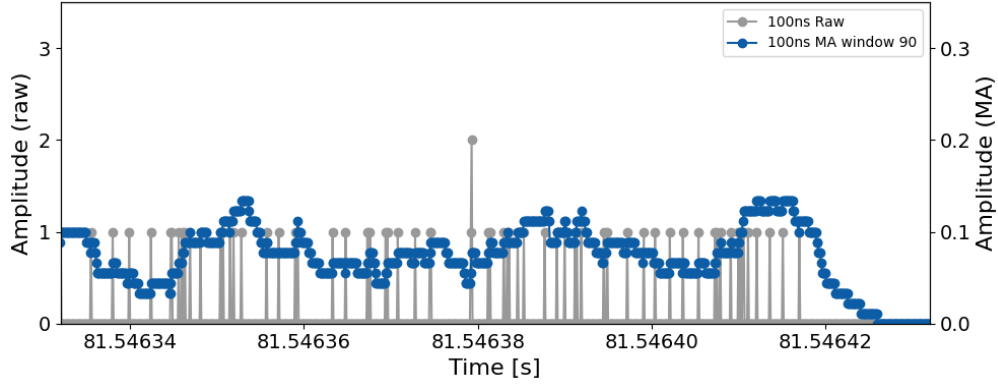
- 1: Estimate the total number of rows in the raw data
 - 2: Determine viable sampling ratios spaced by 10 times (i.e. 0.1%, 1%, 10%) that result in row counts within the constraints
 - 3: **for** each viable sampling ratio **do**
 - 4: **for** each sample [number of samples = 5] **do**
 - 5: Randomly select a seed and start percentage to calculate the start and end times
 - 6: **for** each line in the raw data file **do**
 - 7: Parse the timestamp from the line and calculate in seconds
 - 8: Adjust the timestamp relative to the start time
 - 9: **if** timestamp is within the specified time range **then**
 - 10: Save the line to the output sample
 - 11: **else if** timestamp exceeds the end time **then**
 - 12: Stop reading the file and close the output sample
 - 13: Exit the loop
-

Moving Average

While approaching smaller t_{bin} , where in a QD fewer photon detector clicks would be included in the same bin. For example, $t_{\text{bin}} = 100 \mu\text{s}$ usually have time series amplitudes of less than 100 in each bin, $t_{\text{bin}} = 10 \mu\text{s}$ usually have less than 20 in each bin. While these are still acceptable, the scenario would dramatically change when approaching high-resolution time bins. As shown by the binned data (grey) in Figure 2.4, $t_{\text{bin}} = 1 \mu\text{s}$ is most likely to have aggregated amplitudes per bin for 0, 1, 2 only; $t_{\text{bin}} = 100 \text{ ns}$ is even worse, resembles a binary series with frequent switches.



(a). $t_{\text{bin}} = 1\mu\text{s}$ and lasting for 100 indices



(b). $t_{\text{bin}} = 100\text{ns}$ and lasting for 1000 indices to accommodate same boundary of time, moving average series has been visually scaled up x10 times

Figure 2.4: Examples of the effect of moving average (MA) on raw data (grey) into processed (blue). Both subplots use the same raw data lasting for 0.1 ms but binned with different t_{bin} .

The primary reason for these RTSs to be considered inappropriate is not only because of its binary-like shape but most importantly the frequent sudden switches that do not have a period of stationary amplitude, hence cannot be treated as a normal step, or change in transition amplitude by Δ_{RTS} in RTSs. Passing these signals directly for denoising and digitization would result in a very high likelihood of failing, and even if succeeded, the results would not be meaningful. The incentive for applying MA in pre-processing is to smooth out the data stream and make the amplitudes less spiky, hence improving the success chance of denoising RTSs with few unique discrete values of amplitude. Ultimately

in Chapter 3, a significantly higher digitization success chance by implementing MA pre-processing at high resolution ($t_{\text{bin}} \leq 1 \mu\text{s}$) is observed.

Processing the time binned RTSs with $t_{\text{bin}} = 1 \mu\text{s}$ (grey) in Figure 2.4a with appropriate MA window size (blue) significantly smoothed the spiking amplitudes. From visual inspection, there is a much higher likelihood that KDE would recognize Δ_{RTS} approximately between amplitude 1 and 0. Moving to Figure 2.4b with a different $t_{\text{bin}} = 100 \text{ ns}$, the MA window size must be increased accordingly to maintain the effect of smoothing. By default, a 10-time increase in resolution (a 10-time decrease in t_{bin}) will have a 10-time increase in MA window size.

By inspecting the MA processed series in both t_{bin} , it can be observed that both series resemble an extremely similar shape. This is required and expected as they hold the same raw data which they were binned from. Recalling time binning is the process of changing the resolution for RTS analysis, hence binning the same raw data differently shall not alter the physics and must carry similar RTS traces [1]. The similarity of shape towards the MA processed signals between different t_{bin} in Figure 2.4, and the fact that Figure 2.4b with a 10-time higher resolution provides indices x10 as dense, which validate the importance of studying different t_{bin} : different resolution, but same physics.

Force Dense Sample

Utilizing random sampling (for $100 \mu\text{s} \leq t_{\text{bin}} \leq 1 \mu\text{s}$) was great practice for processing raw data into appropriate RTSs for analysis, not only saving up data storage space but also processing efficiently by using a portion of the entire sequence of data. However, as we approach much smaller time bins ($t_{\text{bin}} \leq 100 \text{ ns}$), the extracted RTSs have become too empty for the following stages of the algorithm to process as mentioned in Sec. 2.1.1. More on how the method was utilized will be elaborated in Chapter 5.

Dense regions, or blocks, can be extracted using a strategy that involves identifying the window with the highest density. As an example, to extract the densest RTS spanning 300K rows of data (a window) out of the $t_{\text{bin}} = 100 \text{ ns}$ full 300 s signal which would last for 3B rows, a sweep of the 10,000 windows will be performed to identify the window that has the highest aggregated counts as shown in Algorithm 3. Once the window boundary is localized, the time binning Algorithm 1 will be applied for RTS extraction.

Algorithm 3 Finding the Most Dense Window

Input: window size (desired [RTS](#) length), desired t_{bin} , total duration = 300 seconds

Output: Identification of the window with the maximum total counts

- 1: Initialize an empty list to store the count of hits in each bin
 - 2: Read line by line of the raw data file ($t_{\text{bin}} = 81$ ps)
 - 3: **for** each line **do**
 - 4: Convert the timestamp to seconds and adjust by the start time
 - 5: Determine the bin number and increment the count
 - 6: Find and print the bin with the maximum count (relevant start and end time)
-

2.1.3 Performance

In the following stages of [RTS](#) analysis, the performance will be quantitatively evaluated with the following hardware specifications on Beluga Compute Canada: allocating 10 GB of RAM, utilizing 10 cores of an Intel Gold 6148 Skylake CPU @ 2.4 GHz and 1 NVIDIA V100SXM2 GPU with 16 GB of VRAM.

To be elaborated in [Sec. 6.4](#), the performance of any stages at this era of the project has yet to be optimized and is for functionality only, as no optimizations have been implemented when there are no such requirements for processing real-time [RTS](#) streams. Specifically for pre-processing, the execution time is not monitored, since [QD RTSs](#) that hold different time resolution dynamics (or any generic [RTSs](#) that hold a different number of traps or noise compositions) would require different pre-processing techniques. Plus, the current stage is more closely related to the concept of preparation, hence in nature, it would not be appropriate to incorporate the execution performance towards the chain of [RTS](#) processing, although optimizations are still considered and to be developed.

2.2 Denoising Stage

2.2.1 Black Box

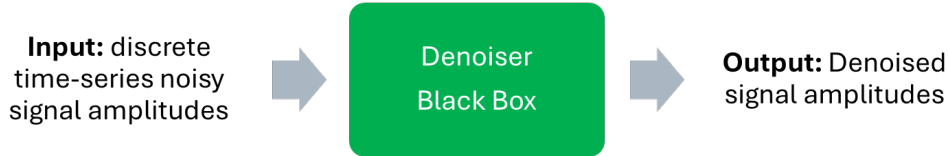


Figure 2.5: Overview of Denoiser Black Box (shown as green blocks in future diagrams).

Input: An output of sufficiently dense time-series signal amplitudes given from [Sec. 2.1.1](#). [RTSs](#) at this stage is expected to be noisy and may not necessarily show instantaneous switching events of [RTSs](#) to be recognized by visual inspection, primarily due to white noise and pink noise.

Output: Significantly denoised time-series signal amplitudes will be generated through the denoising algorithms, to the point where a visual inspection should show the obvious presence of an [RTS](#) trace. The signals now resembles discrete step-like shapes and is ready to be digitized into binary (on and off) combinations of different traps in [Sec. 2.3 Digitization Stage](#).

2.2.2 Kernel Density Estimation (KDE)

[KDE](#) is a technique used to estimate the probability density function of a random variable, briefly introduced in [Sec. 2.1.1](#). [KDE](#) smooths the data by placing a kernel (usually a Gaussian function) over each data point and then summing these kernels to produce a density function. The key advantage of [KDE](#) over a traditional discrete histogram is its ability to generate a continuous, smooth representation of the data distribution, which is particularly helpful for noisy signals, where high variability and fluctuations are common.

Unlike histograms, which are limited by fixed bin boundaries and can produce artificial peaks or gaps due to varying binning choices, [KDE](#) provides a smooth curve that better represents the true distribution, reducing the influence of bin selection on the visual output.

This smoothness enhances our interpretation of the specific [RTS](#) by downplaying random fluctuations and noise, making it easier to identify trends and characteristics within the data. Moreover, [KDE](#) provides a clearer view of multi-modal distributions (typical in [RTS](#), i.e., 1-trap [RTS](#) gives 2 modes shown in right subplot of [Figure 2.6](#)) and patterns that might be missed or misrepresented in histograms, providing a more precise and visually intuitive analysis of complex data sets.

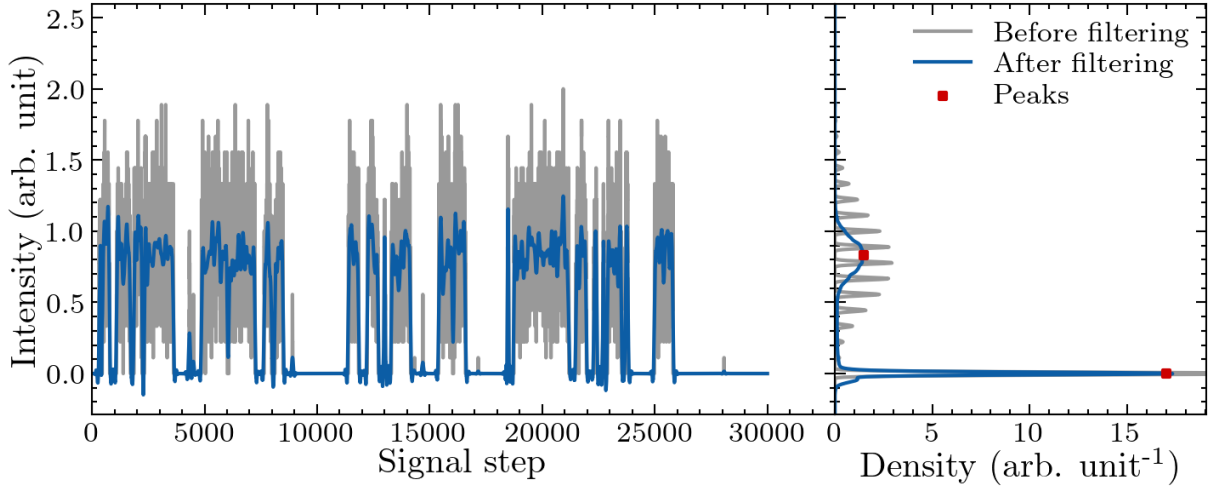


Figure 2.6: Exemplar denoising and [KDE](#) process of a [QD RTS](#) sample.

The primary goal of applying [KDE](#) in this context (as mentioned in [Sec. 2.1.1](#)) is to identify the [RTS](#) levels with high confidence and accuracy. [RTSs](#) often exhibit fluctuating intensity between their discrete states/levels, which must be precisely recognized to characterize τ_{high} and τ_{low} after digitization. Through the denoising stage, usually followed by [KDE](#) regardless of the denoising method, [KDE](#) helps to identify the [RTSs](#)' distinct levels by locating the density peaks from the intensity-density plot, i.e., the red points in [Figure 2.6](#).

In [Algorithm 4](#), once a denoising method has been applied to clean the raw [RTS](#) (from before filtering to after filtering in [Figure 2.6](#)), [KDE](#) is employed to analyze the resulting denoised [RTS](#). The peaks (red points) in the estimated density function correspond to the [RTS](#) levels of the current trap. These peaks are identified using peak-finding methods that examine the prominence and relative intensity of each peak. The main difference to a basic peak-finding method is the use of prominence, which is a measure of how much a peak

stands out from the surrounding noise, allowing some extra robustness if another peak is close by. Additionally, When the data is from a known 1-trap source, such as in [QD \[1\]](#), the “force 1-trap” option can be enabled to improve recognition of the most prominent peaks. This helps focus on the two most significant peaks, enhancing the detection of [RTS](#) levels in such cases.

Algorithm 4 Kernel Density Estimation (KDE) Process

Input: Noisy [RTS](#)

Output: Denoised [RTS](#) and its [KDE](#)

- 1: Normalize signal if required (real or synthetic data)
 - 2: Estimate white noise from the normalized signal
 - 3: Denoising algorithms, to be covered in [Sec. 2.2.3](#)
 - 4: Perform [KDE](#) using Gaussian kernel on the [RTS](#)
 - 5: Estimate signal density and find peaks in the [KDE](#) result
 - 6: **if** force 1-trap is enabled **then**
 - 7: Select top 2 peaks by the intensity
 - 8: **else**
 - 9: Filter out low-prominence peaks from the KDE result
 - 10: **if** number of peaks is abnormal (e.g., less than 4 for 2-trap, less than 8 for 3-trap) **then**
 - 11: Search for extra peaks in the KDE result
 - 12: Merge new peaks with existing peaks
 - 13: Save and exit
-

2.2.3 Variants

Similar to the concept of variants in the pre-processing stage, there are different options studied to denoise [RTS](#) too. The denoising variants internal to the black box are outlined in [Table 2.2](#) below. The most suitable option for [RTS](#) analysis is heavily dependent on the characteristics of the given [RTS](#), where effects of denoising by different methods will be discussed in [Chapter 3](#).

Moving Average

[MA](#) denoising, to be distinguished by [MA](#) in [Sec. 2.1.2](#) pre-processing, is a fast and effective baseline method for reducing noise in signals, particularly useful when white noise

Denoiser Variants	Execution Arguments
Moving Average	Window size
Denoising Autoencoder	None
Wavelet Transform	Decomposition levels, threshold

Table 2.2: Denoising Variants and Corresponding Execution Arguments.

is the primary background fluctuations. The principle behind [MA](#) is the use of a rolling window to smooth the raw [RTS](#), by averaging out fluctuations thereby capable of reducing the effect of random noise. To alter the effect of smoothing through [MA](#), increasing the window size generally results in greater smoothing, which helps to flatten noise and reduce fluctuations in the signal. However, care must be taken when selecting the window size. Too large of a window size can over-smooth the signal, potentially flattening out critical features such as the discrete steps in [RTS](#) levels. Excessive smoothing may blur or entirely mask these important signal characteristics as these steps represent the true transitions between different states in a [RTS](#). More on how the window size would be chosen will be covered in [Chapter 3](#).

[MA](#) is especially advantageous due to its simplicity and computational efficiency (to be compared with other methods in [Sec. 2.2.4](#)), making it a popular baseline for denoising. As noted in Deng’s work [2] when testing synthetic [RTSs](#), [MA](#) excels at mitigating background fluctuations, whether local, regional, or global, without introducing significant distortion to the underlying signal, making [MA](#) helpful to the realization of distinct states. While more advanced methods may be required for more complex [RTS](#) patterns, [MA](#) remains to be a quick solution for denoising.

Denoising Autoencoder

In the deep learning approach towards denoising, the general process involves feeding a noisy [RTS](#) into a model that is trained to infer and output a clean, denoised signal. One such deep learning model, the [denoising autoencoder model based on U-Net and bidirectional long short-term memory \(DAEUBL\)](#), developed by Deng [2], is specifically designed to denoise synthetic [RTSs](#). This model is built using convolutional layers, max-pooling layers, and bidirectional Long Short-Term Memory (Bi-LSTM) layers. The convolutional layers capture features of the signal, while max-pooling reduces dimensionality. The Bi-LSTM layers, on the other hand, are designed to handle time dependencies, making the

model effective at processing sequential data like [RTSs](#) [11]. In Deng’s work, the model was trained on synthesized [RTS](#) data, in an attempt to generalize towards both synthetic and real-world [RTSs](#), although the latter has yet to be tested [2].

[DAEUBL](#) is particularly well-suited for handling complex denoising tasks [2], such as the presence of pink noise and multi-trap [RTSs](#) with more than two levels. This is expected, as the model was trained with these types of complex data. Its effectiveness excels in scenarios where traditional methods like [MA](#) struggle, especially when there are global-trend fluctuations or fast switching [RTS](#) levels. For example in Deng’s work [2], when the relative intensity of pink noise was high enough, [DAEUBL](#) shows a significant advantage over [MA](#) in suppressing global trends that could otherwise distort the signal.

However, [DAEUBL](#) does have some drawbacks, particularly in terms of computational efficiency (> 10 hours for [RTSs](#) length in 30M rows). Inference using [DAEUBL](#) can take a considerable amount of time, even when using a GPU, making it less practical for real-time applications compared to the much faster [MA](#) approach. This trade-off between accuracy and speed is a key consideration when selecting a denoising method for different use cases. As an example for [QD](#) data, which will be elaborated in [Chapter 3](#), the data is known to contain minimal pink noise and is certainly simple 2-level [RTS](#), then a favor may be given towards other alternative approaches if the quality of the results can still be guaranteed.

Wavelet Transform

The incentive for introducing this wavelet denoising technique is not to replace or supersede the machine learning approach in all characteristics but to offer a much faster and equally effective method for simpler [RTSs](#). By applying the method below, we can achieve denoising results that are efficient in computation while maintaining the integrity of the signal for simpler tasks that do not require the complexity of deep learning models.

Wavelet transform is a versatile mathematical technique used for analyzing and processing digital signals. It essentially decomposes a signal into multiple levels, with each level offering a distinct time and frequency resolution. This allows wavelet transforms to capture both time and frequency characteristics of the signal at the same time. Denoising tasks can also be achieved with wavelet transform by applying thresholding at each level to attenuate noise-related coefficients, then the signal may be reconstructed as a denoised signal.

In wavelet transforms, there are various options for signal decomposition, each with different characteristics and trade-offs [16]. The discrete wavelet transform (DWT) decomposes the signal by downsampling low-frequency components at each level for a compact representation but losing shift-invariance by downsampling. The wavelet packet transform (WPT) extends DWT by also decomposing at high-frequency components. However, both DWT and WPT suffer from the lack of shift-invariance, which means that small shifts in the input signal (noisy RTS) can result in significant changes of the corresponding signal steps in the output (denoised RTS).

The stationary wavelet transform (SWT) addresses this issue by avoiding downsampling, thus maintaining shift-invariance. However, SWT introduces a high level of redundancy with increased computational cost and memory usage. In contrast, the dual-tree complex wavelet transforms (DTCWTs) [17] offers an ideal balance by achieving shift invariance with limited redundancy, making it more computationally efficient for tasks that require both high accuracy and low processing time. Furthermore, DTCWT has been shown to outperform other wavelet transforms in large white noise scenarios, as evidenced by results in Mortazavi’s work [16].

As shown in Algorithm 5 for DTCWT developed by Kingsbury [17]. DTCWT is implemented using two parallel trees, each containing a lowpass (scaling) and a highpass (wavelet) filter. The key to DTCWT’s performance is the relationship between the filters in the two trees: the filters in one tree are designed to approximate Hilbert transforms of the filters in the other tree [18]. This pairing creates complex-valued wavelets that can capture both amplitude and phase information, giving DTCWT a more detailed and accurate representation of signals.

Algorithm 5 Dual-tree complex wavelet transform [17]

Input: Signal values at the n -th time step $x[n]$, number of decomposition levels L

Output: Complex wavelet coefficients for real and imaginary parts

- 1: Initialize two parallel filter trees: Tree A and Tree B
 - 2: Define the low-pass filters H_{0a} and H_{0b} , and high-pass filters H_{1a} and H_{1b} for Tree A and Tree B respectively
 - 3: **Step 1: First Level Decomposition**
 - 4: Apply H_{0a} and H_{1a} to $x[n]$ to generate low-pass and high-pass outputs for Tree A
 - 5: Apply H_{0b} and H_{1b} to $x[n]$ to generate low-pass and high-pass outputs for Tree B
 - 6: Form complex coefficients: $\text{highpass} = H_{1a} + j \cdot H_{1b}$
 - 7: Store highpass coefficients
 - 8: **Step 2: Multi-Level Decomposition (for $l = 2$ to L)**
 - 9: **for** each decomposition level l **do**
 - 10: Downsample lowpass outputs from previous level by 2
 - 11: Apply low & high-pass filters H_{0a}, H_{1a} to downsampled lowpass signal in Tree A
 - 12: Apply low & high-pass filters H_{0b}, H_{1b} to downsampled lowpass signal in Tree B
 - 13: Store highpass complex wavelet coefficient: $\text{highpass}_l = H_{1a} + j \cdot H_{1b}$
 - 14: **Step 3: Reconstruct the Signal**
 - 15: Initialize reconstruction by combining highpass and lowpass coefficients from each level
 - 16: Apply inverse filters for both Tree A and Tree B
 - 17: Combine the real and imaginary parts to obtain the reconstructed signal
 - 18: Output complex wavelet coefficients or reconstructed signal
-

The redundancy factor is also an important aspect to acknowledge considering the size of the data mentioned in [Sec. 1.3.3](#). In the context of [DTCWT](#) for [RTS](#) denoising, it refers to the extra data generated during the transform beyond the minimal amount needed to represent the original signal. For 1D signals (i.e., [RTSs](#)), [DTCWT](#) has a redundancy factor of 2, meaning that twice the amount of data is produced in the memory. For higher-dimensional data, the redundancy factor increases exponentially, with a factor of 2^d for d -dimensional data (i.e., a redundancy factor of 4 for 2D images). Despite this redundancy, [DTCWT](#) remains significantly more efficient than transforms like the SWT, which is redundant at every scale and can be computationally expensive. [DTCWT](#) is hence selected to represent the wavelet method, due to having a decent balance between redundancy, shift-invariance, and computational efficiency.

The high-level algorithm shown in Algorithm 6 outlines the steps for denoising a normalized RTS signal using the DTCWT method as implemented in the Python package [19]. First, the algorithm initializes the 1D DTCWT transform and applies a forward transform to the input signal. Thresholding is then applied to the highpass components to eliminate noise, where values below a pre-determined wavelet threshold are set to zero. After modifying the highpass levels, the inverse DTCWT is performed to reconstruct the denoised RTS.

Algorithm 6 DTCWT Denoising

Input: Normalized RTS in Algorithm 4, DTCWT decomposition levels, DTCWT threshold

Output: Denoised signal

- 1: Initialize DTCWT 1D transform
 - 2: Perform forward DTCWT on normalized RTS
 - 3: Apply thresholding to highpass components:
 - 4: **for** each highpass level in `highpasses` **do**
 - 5: Zero out values below wavelet threshold
 - 6: Replace highpass levels in wavelet result
 - 7: Perform inverse DTCWT to reconstruct the signal
 - 8: Output denoised RTS
-

The choice of decomposition levels and threshold values is critical to the effectiveness of the DTCWT denoising algorithm towards different t_{bin} , as these parameters directly influence how effectively noise is suppressed without distorting the true signal. Decomposition levels determine the number of highpass and lowpass filter banks applied to the signal, where more levels allow for finer isolation of noise at varying frequencies but risk over-separating the signal into unintended frequency bands. The threshold, meanwhile, controls the intensity of noise removal within these highpass components: setting it too low may allow residual noise to remain, while setting it too high risks erasing relevant details in the underlying signal, visualized in Figure 2.7. Together, these settings enable a balance between preserving signal integrity and achieving effective noise reduction, which is especially important when managing fluctuations or bit flips commonly present in RTS data.

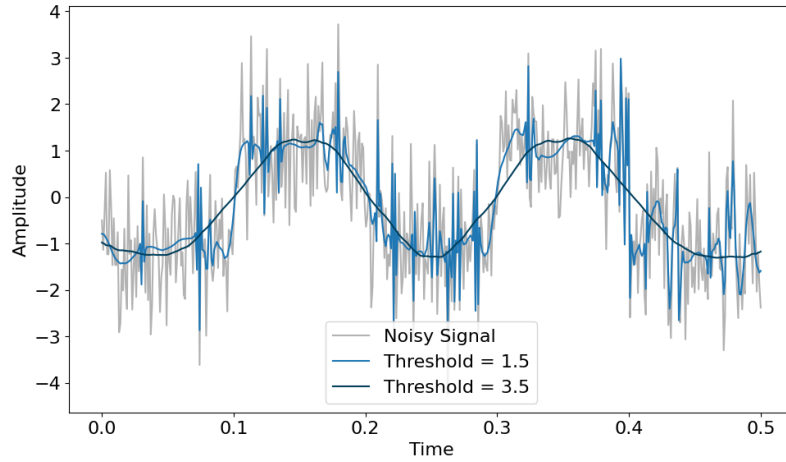


Figure 2.7: Visualization of [DTCWT](#) threshold levels. Exemplar noisy signal (grey) overlaid with the denoised signals at threshold 1.5 (blue) and threshold 3.5 (dark blue).

In a practical demonstration using exemplar [RTS](#) data, the denoising algorithm is capable of effectively reducing noise, particularly removing a short, unwanted peak at index 750, while preserving the underlying signal structure in [Figure 2.8](#). This demonstrates the utility of [DTCWT](#) in ignoring insignificant features, like isolated noise spikes, and providing cleaner signal reconstruction, which is significant considering the sudden peaks outlined in [Sec. 2.3.3](#).

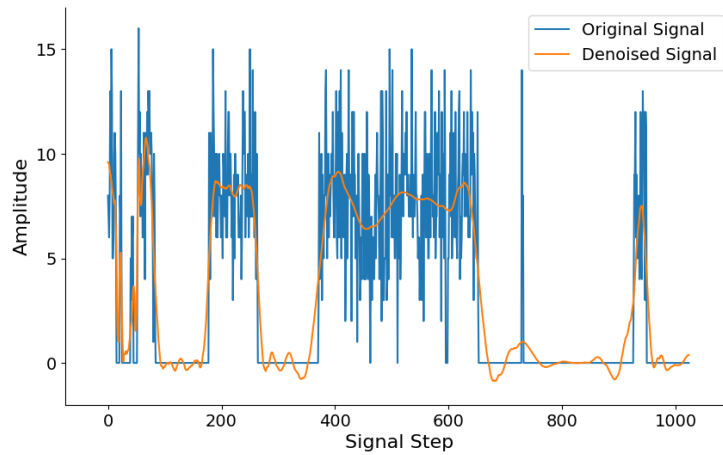


Figure 2.8: Reconstructed/Denoised [QD RTS](#) (orange) overlaid with the raw [RTS](#) (blue) using all thresholded highpass layers in [Algorithm 6](#).

2.2.4 Performance

Using the benchmarking standards mentioned in [Sec. 2.1.3](#), the performance of each denoising variant was benchmarked over jobs in the denoising stage during Phases 2 (in [Chapter 4](#)) and Phase 3 (in [Chapter 5](#)) studies. The average processing speed for the [DAEUBL](#) method was approximately 30 K rows per minute, while the [DTCWT](#) reached more than 566 K rows per minute. The [MA](#) method performed similarly to [DTCWT](#) but slightly slower.

Notably, [DTCWT](#) exhibited a performance advantage of about 20 times compared to [DAEUBL](#). These benchmarks are based on extracted [RTSs](#) at 100 K rows, and the performance gap is expected to widen with larger signals, as [DAEUBL](#)'s scalability is inferior to [DTCWT](#) through testing [RTSs](#) at different sizes. Currently, without further optimizations, [DAEUBL](#) is limited in its ability to denoise input data exceeding 3M rows due to significantly higher RAM requirements. This is largely due to the memory needed for activations across fifteen intermediate layers, which include three convolutional and three max-pooling layers in the encoder, two Bi-LSTM layers, and three convolutional transpose and four convolutional layers in the decoder [2]. On the other hand, [DTCWT](#) requires only about more than twice the size of the input data as memory, which comes from its redundancy factor of 2 when processing 1D signals. This makes [DTCWT](#) a more memory-efficient option for denoising large datasets.

2.3 Digitization Stage

2.3.1 Black Box

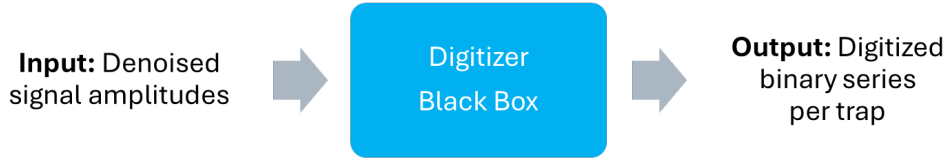


Figure 2.9: Overview of Digitizer Black Box (shown as blue blocks in future diagrams).

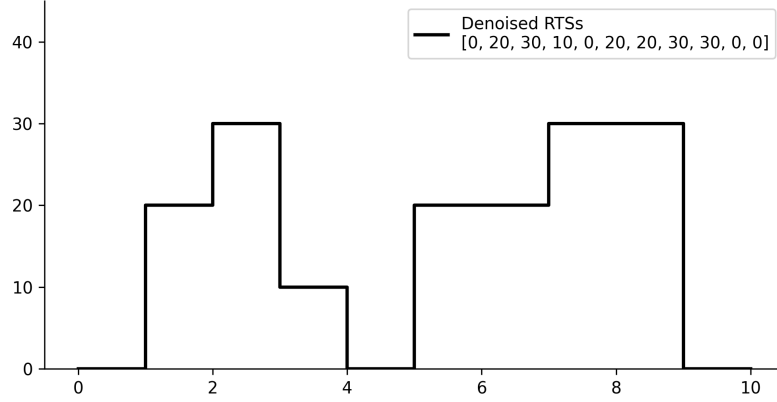
The concept of traps in [RTS](#) was introduced in [Sec. 1.1](#), where we highlighted the challenges in disentangling which specific [KDE](#) levels correspond to each trap’s state in increasingly complex [RTS](#). While a 1-trap (2-level) [RTS](#) presents a straightforward mapping, identifying which level corresponds to the “on” or “off” state of a single trap, the digitization stage becomes essential when handling more complex signals. For [RTS](#) with multiple traps, digitization determines which of the [KDE](#) levels (≥ 2) correspond to specific combinations of Δ_{RTS} levels across different traps. Each level may represent a unique configuration of “on” or “off” states among the various traps present.

Input: From the output of the last denoiser black-box, [RTS](#) has been successfully denoised and contains very minimal background noise from [Sec. 2.2.1](#). The present [RTS](#) as input should visually resemble clear switching steps between Δ_{RTS} .

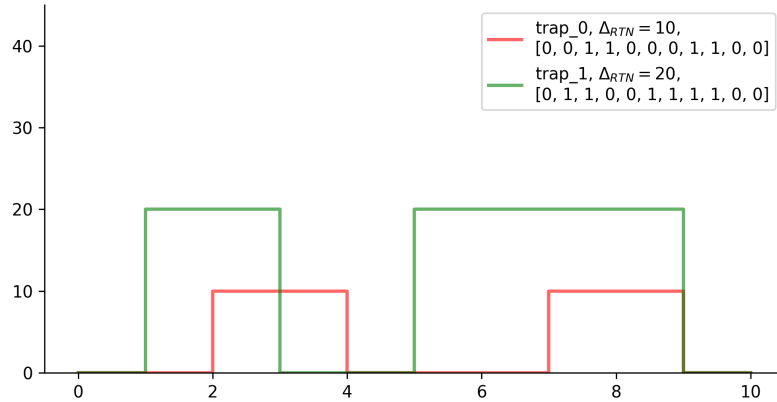
Output: Recognize the respective Δ_{RTS} for each individual trap and convert the one time-series amplitudes of the current [RTS](#) into a combination of time-series binary states, where each one of the binary series corresponds to the [RTS](#) traps.

The digitization process grows increasingly intricate with the addition of traps in a [RTS](#). To illustrate, we start with the simplest scenario: a 1-trap signal, which has only two distinct levels—one for the trap in the “on” state and the other for the “off” state. Such signal resembles a basic binary switching process, where the amplitude switches between two levels, typically representing the transition of charge carriers in the system, as in the case of the [QD](#) data. Here, each amplitude level is directly associated with one of the trap’s

two possible states, making it easy to determine the trap's occupancy status. However, as we analyze **RTS** systems beyond 1-trap setups, the digitization process must account for each potential state combination across multiple traps, a straightforward approach for **nRTS** but not always applicable for **aRTS** systems, where inter-trap correlations can introduce additional complexity.



(a) Exemplary denoised **RTS** with 4 **KDE** levels: 0, 10, 20, 30 respectively. The signal is a superposition of individual trap amplitudes.



(b) Exemplary digitized **RTS** decomposed into trap_0 and trap_1, with corresponding “1” & “0” states and Δ_{RTS} .

Figure 2.10: Visualization of digitization on denoised 2-trap **RTS**.

The complexity of 2-trap [RTS](#) increases significantly. In this case, the signal becomes a superposition of two individual traps, each switching between their respective “on” and “off” states. As a result, the signal now displays multiple levels, corresponding to the combination of states from both traps. Referring to [Equation 1.1](#), where $N_{\text{levels}} = 2^{N_{\text{traps}}}$, a 2-trap system produces a maximum of four distinct levels. This is clearly visible in [Figure 2.10\(a\)](#), where four distinct [KDE](#) levels (0, 10, 20, and 30) are observed in the exemplary denoised [RTS](#) signal.

The challenge in the 2-trap case lies in distinguishing which trap is in the “on” or “off” state at any given time, as the combined signal represents the superposition of both traps’ contributions. Without digitization, it is difficult to tell how the individual traps are combined with the overall signal, as the four amplitude levels represent various combinations of “on” and “off” states from the two traps. For example, one trap being “on” while the other is “off” results in a certain amplitude level in the [KDE](#), while both traps being “on” yields a higher level.

After applying the digitization procedure, as shown in [Figure 2.10\(b\)](#), the original [RTS](#) signal is decomposed into two separate binary time series corresponding to the individual traps. Each binary time series represents the “on” and “off” states of the respective traps, allowing us to trace their contributions over time. By doing this, the signal is transformed from a multi-level time series into a more interpretable format as N_{traps} number of binary series, where each binary series corresponds to the state of a single trap. Furthermore, we obtain the associated Δ_{RTS} values for each trap, giving a clear representation of how the combined signal’s amplitude is constructed from the individual trap contributions.

2.3.2 Variants

Similar to the concept of variants in the denoising stage, different options are studied for digitizing [RTS](#) signals as well. The digitization variants are outlined in [Table 2.3](#). The best option for digitization depends on the number of traps in the [RTS](#) signal. In the 1-trap case, the methods perform very similarly, as the binary state transitions are straightforward. However, as the number of traps increases, the complexity of the signal grows, making some methods more effective than others. Each method’s performance is influenced by the number of traps and the [KDE](#) peak distribution in the signal. The performance of different digitization methods will be further discussed in [Sec. 2.3.4](#).

Digitizer Variants	Execution Arguments
Thresholding	Threshold
Peak Matching	Whether force 1-trap
Hidden markov models	Number of components, number of iterations
Bayesian	Estimated noise σ , window size

Table 2.3: Digitization Variants and Corresponding Execution Arguments.

To help intuitively convey the unique qualities of each method, Figure 2.11 provides a preliminary visual comparison of these digitization variants. The figure displays both the raw RTS signal (blue) and the digitized output (red) for each method. As shown, thresholding (top left) serves as a baseline, classifying states by comparing signal amplitude against a fixed threshold [1]. Peak matching (top right) uses proximity to KDE levels for state assignment. Unlike other methods, HMM (bottom left) does not require KDE and can digitize unprocessed signals directly. Bayesian digitization (bottom right) probabilistically assigns states based on computed KDE levels and surrounding data points, providing robustness against noise.

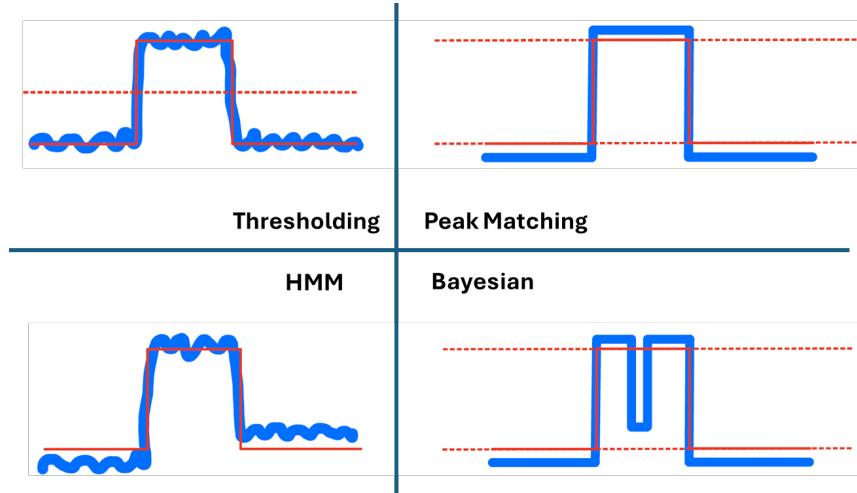


Figure 2.11: Digitization variants visualized with RTS (blue) and digitized RTS (red). Thresholding (top left) as baseline method [1] assign states by comparison with a threshold (dotted line). Peak matching (top right) assigns states by signal proximity to computed KDE levels (dotted lines). Hidden markov model (bottom left) does not require KDE and does not require the signal to be denoised. Bayesian (bottom right) probabilistically assigns states to computed KDE levels (dotted lines) but based on neighboring information.

Thresholding

The thresholding method is one of the simplest approaches to achieve digitization of **RTSs**. In this technique, a predefined threshold is applied directly to the signal without any prior denoising process, as demonstrated in the case discussed by Kerski [1]. Every data point in the signal that lies above the threshold is assigned to the “on” state, while data points below the threshold are assigned to the “off” state. In their work, [1] used arbitrary threshold values, which were estimated by positioning the threshold approximately in the middle of the step amplitudes. This is conceptually similar to averaging our **KDE** levels, where the steps in amplitude correspond to changes in the states of the traps. However, the approach lacks the precision of using **KDE** or more advanced techniques for identifying the exact levels in **RTSs**. While this method was implemented as the baseline for digitization comparisons, to further formalize it, the threshold value would be determined by approximately half of the estimated Δ_{RTS} obtained from **KDE** process in the denoising stage as discussed in [Sec. 2.2.2](#).

One of the primary limitations of this direct thresholding method is its inability to handle sudden jumps in amplitude that cross the fixed threshold. For example, if noise or other fluctuations cause the signal to cross the threshold momentarily, the method may classify the state incorrectly. This issue is particularly problematic in multi-trap signals where more than two states are involved. A fixed threshold is unable to distinguish between more than two distinct amplitude levels and may result in significant misclassification when the amplitude shifts between more complex combinations of trap states. Additionally, thresholding is highly sensitive to the selection of the threshold value. For **RTS** with higher noise levels (average magnitude of white noise exceeds the average of two **KDE** levels) or more than one trap, thresholding becomes increasingly ineffective, making it a less robust method compared to techniques that incorporate probabilistic or dynamic modeling to be discussed shortly.

In summary, while thresholding is straightforward and computationally inexpensive, it is best suited for 1-trap **RTSs** with clear step transitions and minimal noise. For multi-trap **RTSs** or those with significant noise, more advanced methods are required to accurately digitize the signal.

Peak Matching

The peak matching method for digitization was used in Deng’s work [2], which offers a practical solution for the rapid digitization of RTS data, particularly for signals with up to 3 traps, corresponding to a maximum of 8 levels (see Equation 1.1). This method is straightforward and leverages the filtered RTS’s intensity by matching it to the nearest KDE peak values. The KDE peaks represent the possible amplitude levels that result from the “on” and “off” states of the individual traps in the RTS.

In this approach, as shown in Algorithm 7, the method works by first identifying the KDE peaks in the denoised RTSs from the denoising stage. These peaks act as reference points, indicating the possible amplitude levels in the RTS data. For each data point in the signal, the nearest KDE peak is determined based on the current signal intensity. Once the closest peak is identified, the signal intensity is assigned to that corresponding peak, ensuring that the signal is effectively quantized into its most probable amplitude level. This method directly associates each point in the signal with one of the KDE-derived levels, making it a computationally efficient approach for digitization but with the requirement of prior KDE results.

Algorithm 7 Peak Matching Digitization

Input: Denoised RTS

Output: Digitized RTS with binary trap states

- 1: Calculate the number of traps based on the number of KDE peaks
 - 2: **for** each point in the filtered signal **do**
 - 3: Find the closest KDE peak to the current signal intensity
 - 4: Assign the nearest KDE peak value to the signal, similar to Figure 2.10(a)
 - 5: **for** each point in the digitized signal **do**
 - 6: **if** number of traps is 1 **then**
 - 7: Convert the KDE peak index to binary state for 1-trap
 - 8: **else if** number of traps is 2 **then**
 - 9: Use predefined levels (e.g., '00', '01', '10', '11') for 2-trap states
 - 10: Convert the KDE peak index to binary states
 - 11: **else if** number of traps is 3 **then**
 - 12: Use appropriate level combinations for 3-trap states and assign accordingly
 - 13: Save the digitized RTS
-

The next step involves converting the KDE peak index into a binary state. The number of traps in the signal determines how the peak index is mapped to binary combinations of trap states. For instance, in a 1-trap RTS, the process is simple as there are only two levels: “on” and “off.” In a 2-trap signal, the method must handle four possible levels, which represent the various combinations of the two traps being in the “on” or “off” state (e.g., both off, one on and the other off, or both on). The method scales similarly for signals with 3 traps, where 8 possible levels exist.

Peak matching is highly efficient and performs well for RTSs with up to three traps in Deng’s prior work [2]. However, the method relies on the distinctness of KDE peaks to achieve accurate digitization. If the signal is noisy or if the KDE peaks are too close together, the method may struggle to assign the correct peak, resulting in potential misclassifications. Additionally, as the number of traps in the signal increases beyond 3, the complexity of the RTSs increases and a simple peak matching may no longer be sufficient to capture the intricate transitions between the different amplitude levels. In such cases, more advanced methods may be required to handle the signal’s complexity.

Hidden Markov Models (HMMs)

The hidden markov model (HMM) approach is another method for digitizing RTS data, particularly useful when dealing with more complex signals that exhibit noise or intricate transitions between states [20, 21]. Unlike simpler methods such as thresholding or peak matching, HMM can account for the probabilistic nature of transitions between different states, making them a better-suited option for RTS where the distinction between “on” and “off” states is not always clear-cut.

HMM treats the RTS as a sequence of observations generated by hidden states [22]. These hidden states correspond to the underlying trap states in the signal, and the goal of HMM digitization is to uncover these hidden states and map them to the observable RTS amplitude levels. By modeling the signal in this probabilistic method, HMM can capture the likelihood of transitions between different trap states and predict the most probable sequence of hidden states that generated the observed RTS.

The general process of HMM digitization begins by reshaping the denoised RTS signal for input into a Gaussian HMM. The model is initialized with a specified number of components, where each component represents a possible trap state, and is trained through

an iterative fitting process. Once fitted, the model predicts the hidden states for each data point in the [RTS](#) signal. These predicted states are then mapped to the [RTS](#) amplitude levels, resulting in a time series of binary trap states, where each state corresponds to whether a trap is in the “on” or “off” position. The pseudocode for this process is provided in [Algorithm 8](#).

Algorithm 8 [HMM](#) Digitization

Input: Denoised [RTS](#), number of components n , number of iterations n_{iter}

Output: Digitized [RTS](#) with binary trap states

- 1: Reshape the [RTS](#) into a 2D array for [HMM](#) input
 - 2: Initialize GaussianHMM model with n components and n_{iter} iterations
 - 3: Fit the model to the 2D array
 - 4: Predict the hidden states using the fitted model
 - 5: **for** each predicted state **do**
 - 6: Map the predicted state to the corresponding [RTS](#) level based on the model’s means
 - 7: Save the digitized [RTS](#)
-

Fundamentally speaking, a standard [HMM](#) setup for a 1-trap [RTS](#) requires only a 2x2 matrix with a small number of unknowns, a multi-trap system dramatically increases the matrix size and number of parameters, making parameter estimation significantly more challenging. Unlike 1-trap models where initial values are easily assigned, multi-trap [HMMs](#) are sensitive to initial parameter values, often yielding varying results depending on these choices. For handling multi-level states, factorial HMMs (fHMMs) [23] was introduced as an extension of traditional HMMs, allowing multiple hidden chains to represent different traps or transitions. However, fHMMs also suffer from the challenge of assigning initial values to an expanded set of parameters, which can negatively impact model reliability and make the approach difficult to implement. This complexity has led to limited applications of fHMMs in [RTS](#) analysis, as seen in [7], and thus they were not integrated into the current framework.

While [HMM](#)-based digitization remains highly effective, it presents notable trade-offs. The probabilistic structure of [HMMs](#) allows it to capture transitions more robustly than methods such as thresholding or peak matching. However, [HMMs](#) are considerably slower and computationally more demanding than peak matching [2], especially towards multi-traps. Consequently, [HMM](#) digitization is best suited for cases prioritizing high accuracy over processing speed.

The characteristics of 1-trap QD RTSs lend themselves well to simpler, efficient digitization techniques. For straightforward, 1-trap signals, HMMs digitization indeed offers a powerful approach, even capable of digitizing noisy signals without requiring a prior denoising process. However, given that one of this framework’s goals is to develop and test effective denoising methods—particularly in demanding cases, where existing DAEUBL is insufficient for high-resolution long QD RTS, applying HMMs on 1-trap denoised signals brings limited additional benefit. While the data is already denoised, simple peak matching provides the most computationally efficient solution that is comparably effective in this context, making it a practical choice for denoised 1-trap RTS where high accuracy is achievable without the additional resource demands of HMM.

Nevertheless, a planned study beyond the scope of this thesis will evaluate HMM’s reliability on noisy signals directly, against denoising with DTCWT followed by peak matching for simple 1-trap RTSs. However, for the QD data we are currently testing, DTCWT denoising was prioritized as it is computationally lighter than other methods, such as DAEUBL while offering significant performance benefits. As a result, peak matching was selected as the primary digitization method.

Bayesian

Bayesian methods have been previously explored in the context of RTS noise analysis [24, 25], as they offer powerful tools for probabilistically modeling and interpreting fluctuating signals. Bayesian approaches, particularly suited for dealing with uncertain and noisy data, are valuable for extracting meaningful information from RTSs, where noise often obscures state transitions.

The Bayesian digitization method in the proposed framework, implemented by Kapoor [26], provides another probabilistic approach for analyzing multi-level, complex RTS data, similar to HMM. This method is advantageous when dealing with intricate and RTS with noises that could not be fully removed from the denoising stage. A key aspect of this Bayesian digitization implementation is the construction of priors using Dempster-Shafer Theory [27], which fuses the prior probabilities of RTS signal states based on the posterior probabilities derived from the previous n data points. This approach allows for a bit more robust digitization in comparison to simpler techniques like thresholding or peak matching.

Dempster-Shafer theory [27] is a method for fusing evidence from different sources—in this case, fusing posterior probabilities from previous data points. By doing so, the method becomes increasingly effective as the size of the data window increases, making the digitization more resilient to sudden changes in the RTS and effectively against the remaining noises. This approach is advantageous in scenarios where simple averaging of previous posteriors does not yield reliable results, as observed in other windowed techniques.

In Algorithm 9, Bayesian digitization first calculates the likelihood of each RTS amplitude level based on the denoised RTS. It then uses Bayes' theorem to update the posterior probabilities, which are combined over a sliding window to create a more stable estimate of the current state. Dempster-Shafer's theory is then applied to combine these probabilities, and lastly, the prior probabilities are updated iteratively with a smoothing factor, which further improves prediction consistency over time.

Algorithm 9 Bayesian Classification with Smoothing and Dempster-Shafer Fusion

Input: Denoised RTS y with amplitudes L , estimated noise standard deviation $\hat{\sigma}$, window size w

Output: Digitized RTS s'

- 1: Initialize prior probabilities P as equal for all levels
- 2: Initialize an empty list to store posterior probabilities within a sliding window
- 3: **for** each data point $y_i \in y$ **do**
- 4: **for** each $L_j \in L$ **do**
- 5: Compute likelihoods $P(y_i|L_j)$ using $\hat{\sigma}$
- 6: Calculate and normalize posterior probabilities $P(L_j|y_i) \propto P(L_j) \cdot P(y_i|L_j)$
- 7: Add the new posterior probabilities to the list and maintain a window of size w
- 8: Combine the posterior probabilities within the window using Dempster-Shafer:

$$\text{Combined probabilities} \leftarrow \frac{\prod_{k=1}^w \text{posterior probabilities}}{\sum}$$

- 9: Classify y_i based on the combined probabilities and update s'
 - 10: Update prior probabilities using smoothing: $P \leftarrow \alpha P + (1 - \alpha) \times \text{combined probabilities}$
 - 11: Save the digitized RTS
-

Alternatively, Dempster-Shafer fusion can be replaced with Maximum Likelihood Estimation (MLE) for constructing prior probabilities [26]. In this variant, the prior is calculated using the KDE-derived signal levels and estimated standard deviations, predicting the state for each data point based on maximum likelihood. While these approaches in Bayesian digitization have potential, it has yet to be thoroughly tested and benchmarked against methods like peak matching or HMM in the context of RTSs. For the scope of this project, peak matching has been used for 1-trap QD RTSs data, as it is more than sufficient, and the focus has been placed on optimizing the denoising process rather than the digitization approach. Expanding on this, separate paper will be conducted specifically targeting Bayesian methods in RTS analysis, which can provide further insights into their applications and advantages.

2.3.3 Bit Flip Error Correction

Lorke, author of [1], highlighted that with the current QD and detector setup, it is possible to observe sudden jumps between states that quickly reverse. These rapid switches can lead to inaccuracies in the digitized results, particularly in the calculation of key metrics like $\bar{\tau}_{\text{high}}$. We classify the error possibilities as Figure 2.12. When a sudden switch is mistakenly identified as a valid state transition, it can split a single “high” state into two shorter segments (false negative) shown in Figure 2.12, artificially lowering τ_{high} and distorting the overall statistical analysis. Ideally, if the digitization process were flawless, such bit-flip corrections would not be necessary (true positive). However, in practice, post-processing steps are required to correct these brief erroneous transitions.

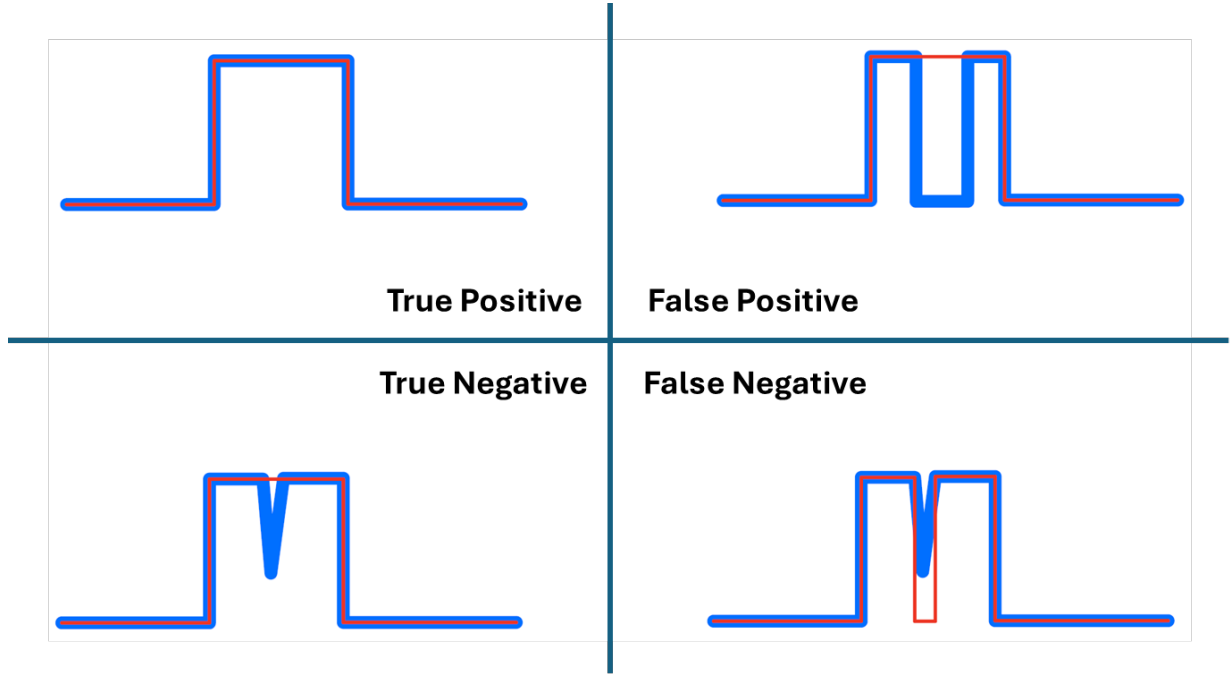


Figure 2.12: Error possibilities and classifications in the digitization process with denoised RTS (blue) and digitized RTS (red). True positive (top left) correctly identifies state transitions when there is no error. True negative (bottom left) correctly identifies error and corrected. False positive (top right) incorrectly identifies error, false alarm. False negative (bottom right) unable to detect and missed the error.

To mitigate this issue, we apply a simple post-processing method, a bit flip error correction, on the binary digitized series. This method filters out fast, short-duration switches that are likely noise rather than true state changes, as detailed in [Algorithm 10](#). The algorithm identifies short-lived changes in the digitized signal. If the duration of the change is shorter than a defined threshold, it is replaced with the preceding or following state value, depending on which is available. This approach usually improves the digitization accuracy that has been affected by sudden fluctuations.

Algorithm 10 Bit Flip Error Correction

Input: Digitized RTS, maximum allowed error window

Output: Digitized RTS with sudden switches removed

- 1: Initialize an index to start scanning the RTS
 - 2: **while** there are still unprocessed values in the RTS **do**
 - 3: Mark the starting point of the current scan (a group of consecutive identical values)
 - 4: Identify the value of this scan (e.g., “on” or “off” state)
 - 5: Move forward through the RTS until the state changes (to find the end of this scan)
 - 6: Calculate how long the scan lasted (number of consecutive values)
 - 7: **if** length of this scan $\leq$ the maximum allowed length for sudden switches **then**
 - 8: Check the state immediately before the scan, if it exists
 - 9: Check the state immediately after the scan, if it exists
 - 10: Decide on a replacement state: either the state before or after, depending on which exists
 - 11: Replace all the values in this short scan with the chosen replacement state
 - 12: Save the RTS
-

One critical parameter in this process is the maximum width of a state that can be considered as an error. This parameter is kept constant throughout the processing of QD RTSs, regardless of the t_{bin} used. While this strategy works well for lower-resolution data, it becomes less effective as resolution increases. The maximum width for error correction does not scale with time resolution, as doing so would risk skewing τ statistics. It is better to treat such short switches as noise in the data, rather than allowing them to bias the results.

Upon testing, it became clear that the key statistical metrics of interest, $\bar{\tau}_{\text{high}}$ and $\bar{\tau}_{\text{low}}$ (ultimately extracted from the digitized signal, which we will discuss in more detail in Sec. 2.4), will be affected by the choice of the width of the error correction filter. For now, the focus is on evaluating how the width of the error correction filter influences the accuracy of these metrics. Specifically, we assess how the filter affects the accuracy of the extracted RTS time constants when compared to known reference values. Using a sample where we have prior knowledge of the correct $\bar{\tau}_{\text{high}}$ and $\bar{\tau}_{\text{low}}$ values, we applied DTCWT denoising followed by peak matching digitization on the RTS. We then applied bit flip error correction with varying filter widths to determine its impact on the error margin. The results are summarized in Table 2.4 with the first 300 K rows of $V_l = 50$ nV, $V_g = 0.3743$ V dataset binned at $t_{\text{bin}} = 10$ μs .

Filter Window	$\bar{\tau}_{\text{high}}$ Error (%)	$\bar{\tau}_{\text{low}}$ Error (%)
No Error Correction	260%	1%
Filter Window of 3	18%	6%
Filter Window of 4	6%	7%

Table 2.4: Error in $\bar{\tau}_{\text{high}}$ and $\bar{\tau}_{\text{low}}$ with varying max filter widths.

The filter window size of 3 or 4 indicates the number of rows processed per index, which can be interpreted as a multiple of the chosen t_{bin} (10 μs in this case). Therefore, a filter window of 3 means max false positives correction size is across 3 consecutive time bins, or 30 μs , while a window of 4 means 40 μs . The filter window size in Table 2.4 is defined as per Algorithm 10 by the number of indices included in the correction. This intuitive window size selection increases the smoothing effect for higher resolutions (i.e., smaller t_{bin} values). Optimal window size obtained from testing with a t_{bin} of 10 μs is maintained for smaller t_{bin} values to minimize the risk of introducing false information. In this approach, the filter primarily reduces false positives (incorrectly identifying a state transition error when none exists) rather than false negatives (failing to detect an actual state transition error). The error deviation in Table 2.4 is calculated as the percentage difference of $\bar{\tau}_{\text{high}}$ and $\bar{\tau}_{\text{low}}$ relative to the reference values provided with the source dataset, as documented in Chapter 3. Since QD data lacks a ground truth, this reference serves as a benchmark rather than an absolute measure of accuracy.

The results show that the $\bar{\tau}_{\text{low}}$ error remains consistently low across all filter window sizes, which is expected since bit flip errors typically occur during the high states, where sudden, brief jumps are more likely to be misinterpreted as separate steps. The significant $\bar{\tau}_{\text{high}}$ error observed without error correction (260%) indicates that many high states were wrongly split into multiple segments due to these sudden jumps. As the filter window size increases, the $\bar{\tau}_{\text{high}}$ error decreases significantly, with a window size of 4 reducing the error to just 6%. Importantly, this improvement in $\bar{\tau}_{\text{high}}$ accuracy does not introduce substantial error in $\bar{\tau}_{\text{low}}$, indicating that the correction method effectively mitigates the impact of sudden switches without compromising the overall accuracy of the low-state durations. Increasing the filter window size beyond 4 (e.g., to 5 or 6) introduces greater deviation in $\bar{\tau}_{\text{high}}$ and $\bar{\tau}_{\text{low}}$. This increased deviation likely arises from more frequent false negatives, as larger windows may start to smooth out true state transitions, leading to missed detections instead of just correcting for false positives. Thus, a window size of 4 appears to balance error correction without introducing significant errors from over-smoothing.

Multiple segments may negatively impact τ_{low} . To estimate $\bar{\tau}_{\text{high}}$ and $\bar{\tau}_{\text{low}}$, we removed the first bin from the τ_{low} histogram, which contains values close to zero. This bin likely represents intervals distorted by detector limitations or noise, which could bias the τ_{low} estimate if included. Removing it provides a more accurate calculation of $\bar{\tau}_{\text{low}}$ by focusing on intervals that better represent the underlying dynamics. However, it is important to note that the smaller error in $\bar{\tau}_{\text{low}}$ observed in Table 2.4 is due not to the removal of this first bin but rather to the change in filter window size.

It is important to note that bit flip error correction is a post-processing step, applied after digitization, rather than a variant of the digitization method itself. Further testing revealed that this correction method is only beneficial when using denoising techniques such as MA and DTCWT, where sudden switches are not fully handled during the denoising process. In contrast, techniques such as DAEUBL, which employ machine learning models, tend to naturally account for such noise during training and are less affected by bit flip errors, making additional correction unnecessary.

2.3.4 Performance

The performance of digitization algorithms is not as intensive as other stages of analysis when processing large volumes of QD RTS data. No formal benchmarking comparison of different digitization methods has been conducted thus far, mainly because peak matching has proven to be more than sufficient for single-trap RTS studies, and its performance does not present a bottleneck for the overall processing pipeline.

At the same computing specifications given in Sec. 2.2.4, formal executions of peak matching through 4000 samples in Phase 3 indicate that it can process data at a rate of approximately 1.8 M rows per minute, and bit flip error correction at a rate of 8.5 M rows per minute. They are significantly faster than denoising algorithms, which are the primary contributors to computational delays in the full processing chain. While more advanced methods like Bayesian digitization could offer improved accuracy for more complex, multi-level RTS, they come with a higher computational cost. However, in the context of single-trap studies, the speed and simplicity of peak matching make it the optimal solution.

2.4 Tau Statistics Stage

2.4.1 Black Box

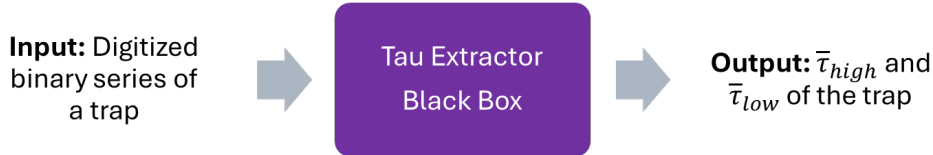


Figure 2.13: Overview of Tau Statistics Extractor Black Box (shown as purple blocks in future diagrams).

Figure 2.13 illustrates the tau statistics extractor as a black box within the [RTS](#) processing framework, which operates on binary time-series signals derived from the previous digitization stage. This extractor’s role is to capture the temporal behavior of each trap by computing dwell-time statistics that quantify how long a trap remains in each state.

Input: The input to the tau statistics extractor consists of the processed data from the previous digitization stage. Specifically, the input is a set of binary time-series signals, where each signal represents the activity of an individual trap in the [RTS](#). These binary series capture the trap’s state over time, where a “high” (“on”, “1”) binary value represents the [QD](#) being empty (photon detector in a high state), and a “low” (“off”, “0”) binary value represents the [QD](#) being occupied (photon detector in a low state). Additionally, the amplitude difference Δ_{RTS} between the high and low states for each trap has been determined. The goal at this stage is to analyze these binary time-series signals to extract meaningful dwell time statistics, specifically the time each trap spends in the high and low states.

Output: The output of the extractor is the quantitative characterization of the trap’s dwell times in each state, represented by the statistical parameters $\bar{\tau}_{\text{low}}$ and $\bar{\tau}_{\text{high}}$. These two metrics correspond to the average time the trap spends in the low and high states, respectively. The extracted statistics are crucial for understanding the temporal behavior of each trap, as outlined in [Sec. 1.1](#). The results of this stage provide a concise summary of the trap dynamics in terms of its characteristic time constants, allowing further analysis of the system’s behavior.

2.4.2 Variant

Currently, only one tau statistics extraction variant is implemented: Exponential Decay Fitting. This method fits an exponential decay function to the dwell-time histograms for both high and low states, providing estimates of $\bar{\tau}_{\text{high}}$ and $\bar{\tau}_{\text{low}}$ values.

While log-scale linear regression could be considered as an alternative approach for analyzing dwell-time statistics, it remains outside the scope of this thesis. Under the fixed magnetic field conditions ($B = 2$ T) used in the QD datasets, the physical principles governing the system indicate that the decay dynamics follow an exponential trend as discussed in Sec. 1.3.2. Therefore, exponential decay fitting accurately reflects the underlying physics, reliably capturing the switching dynamics without requiring additional modeling variants.

Tau Extractor Variants	Execution Arguments
Exponential Decay Fitting	Whether to plot with log scale (visualization only)

Table 2.5: Tau Extractor Variants and Corresponding Execution Arguments.

Exponential Decay Fitting

In this variant, Algorithm 11 is designed to extract statistical information about the dwell times in high and low states from the digitized RTSs. The algorithm identifies the transitions between the high and low states of a signal and measures the time intervals that the signal spends in each state, referred to as τ_{high} and τ_{low} . These dwell times are then used to compute the mean values, $\bar{\tau}_{\text{high}}$ and $\bar{\tau}_{\text{low}}$, by fitting an exponential decay model to the histograms of these intervals.

Algorithm 11 Tau Statistics Extraction

Input: Digitized RTS

Output: Extracted τ_{high} and τ_{low} statistics

- 1: **for** each signal column (a trap) in the digitized RTS **do**
 - 2: Compute the difference between consecutive values in the signal
 - 3: Identify the transition points where the signal switches from low to high (τ_{up}) and from high to low (τ_{down})
 - 4: Match the up and down transitions, ensuring equal lengths of transition pairs
 - 5: For each transition pair:
 - 6: **if** the signal starts in a high state **then**
 - 7: Calculate τ_{high} as the duration between a high-to-low transition and the preceding low-to-high transition
 - 8: Calculate τ_{low} as the duration between a low-to-high transition and the preceding high-to-low transition
 - 9: **else if** the signal starts in a low state **then**
 - 10: Calculate τ_{high} and τ_{low} accordingly
 - 11: Plot histograms of τ_{high} and τ_{low} and fit an exponential decay function to each, optional to plot in log scale
 - 12: Extract the mean values for τ_{high} and τ_{low} from the fitted functions
 - 13: Save the results as $\bar{\tau}_{\text{high}}$ and $\bar{\tau}_{\text{low}}$
-

The exponential fitting function [Algorithm 12](#) is used in the context of RTS to model how the probability of staying in the high (or low) state decreases exponentially over time. The fit of this function to the histogram of dwell times helps determine the average dwell time in each state (i.e., $\bar{\tau}_{\text{high}}$ and $\bar{\tau}_{\text{low}}$). Thus, this function plays a crucial role in capturing the fundamental dynamics of RTS by providing a mathematical model to describe how long the signal tends to stay in a particular state before transitioning.

Algorithm 12 Exponential Decay Fitting Function

Input: Dwell times from RTS

Output: Fitted exponential decay function for τ_{high} and τ_{low} histograms

- 1: Define the fitting function as:

$$\text{func}(x, t, a) = \frac{1}{t} \exp\left(-\frac{x - a}{t}\right)$$

where:

- x is the dwell time in either the high or low state,
 - t is the time constant representing the mean dwell time,
 - a is the shift parameter adjusting the starting point of the decay.
- 2: Fit the function to the histogram of τ_{high} and τ_{low} using non-linear least squares or another fitting method.
 - 3: The fitted time constant t represents the mean value of τ_{high} or τ_{low} .
-

In this variant, the only customizable execution argument relates to the visualization aspect of the extracted statistics. Specifically, the user can choose whether to plot the dwell time histograms on a logarithmic scale, which can be helpful for visualizing exponential decay trends more clearly over a wide range of time scales. However, this option only affects how the results are displayed and does not impact the extraction process itself.

As an example, as demonstrated in Figure 2.14, QD data can be used to showcase the typical outcomes of this extraction process, where the fitted exponential decay curves closely follow the histogram of dwell times. Note that since the algorithm is used to fit RTSs with different t_{bin} , the unit of the fitted $\bar{\tau}_{\text{high}}$ and $\bar{\tau}_{\text{low}}$ is by the index (i.e. t_{bin}). Hence, the resulting $\bar{\tau}_{\text{high}} = 0.56$ ms and $\bar{\tau}_{\text{low}} = 0.60$ ms.

Occasionally, a high occurrence appears in the first bin of the τ_{low} histogram, much more drastic than $\bar{\tau}_{\text{high}}$ in Figure 2.14. This initial peak is often a result of very short intervals between state transitions, potentially caused by detector limitations or noise rather than genuine switching events. To ensure more accurate fitting, the first bin is always excluded from the exponential decay fitting, as including these short, likely distorted intervals could bias the $\bar{\tau}_{\text{high}}$ and $\bar{\tau}_{\text{low}}$ estimation. By removing the first bin, the fitting procedure focuses on intervals that better represent the true decay dynamics of the RTS.

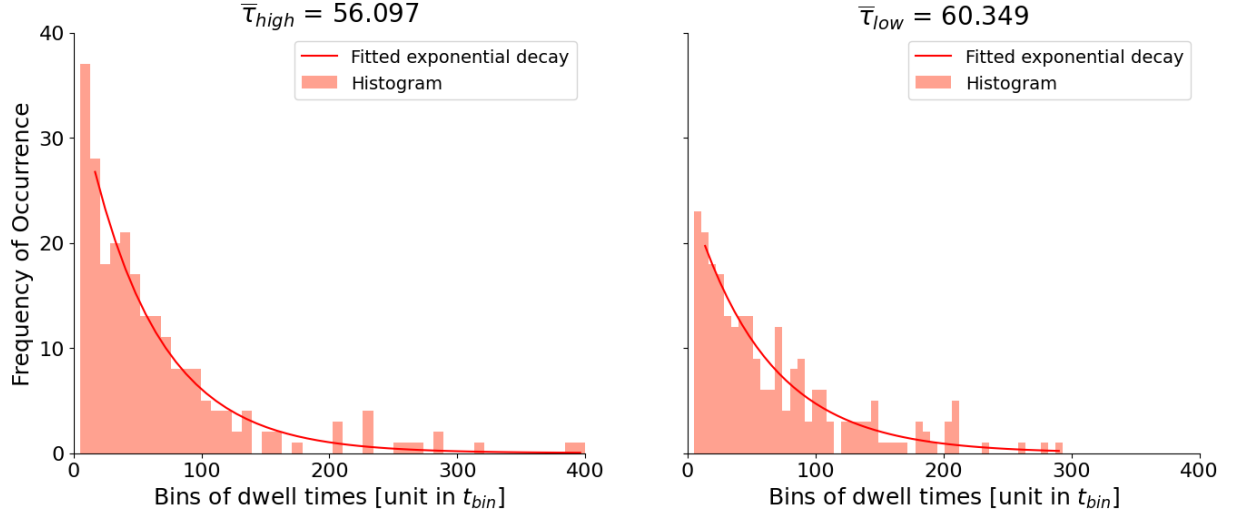


Figure 2.14: Example tau statistics fitting of a QD RTS sample with $V_l = 50$ nV, $V_g = 0.3743$ V, and $t_{bin} = 10$ μ s.

2.4.3 Performance

In our Phase 3 analysis of the tau statistics extraction from the digitized dataset, we evaluated the efficiency of the process using a specific dataset size. For extractions tested on a RTS dataset containing 100 K rows, the time required for the fitting procedure averaged around 2.3 M rows per minute. This indicates that even with more complex signals, the algorithm remains acceptable in performance, ensuring timely analysis.

Chapter 3

Results and Discussions: Phase 1

The studies on the experimental QD data were divided into three main phases, each contributing progressively to a deeper understanding of the QD RTSs data and facilitating the development of more advanced analysis methods. During these phases, several variants of the procedures described in Sec. 2.1 to Sec. 2.4 were developed and refined.

3.1 Goal: Approaching An Exemplar Dataset

Phase 1 focused on the initial exploration of a subset of data, where we were provided with a single exemplar dataset, 0.5e-7V_Excitation/3743.dat. This dataset was recorded with:

- laser excitation power $V_l = 50$ nV
- gate voltage $V_g = 0.3743$ V

and a reference QD tunneling rates of the following at $t_{\text{bin}} = 10$ μs :

- tunneling in rate $\Gamma_{\text{in}} = 2072$ Hz
- tunneling out rate $\Gamma_{\text{out}} = 1456$ Hz

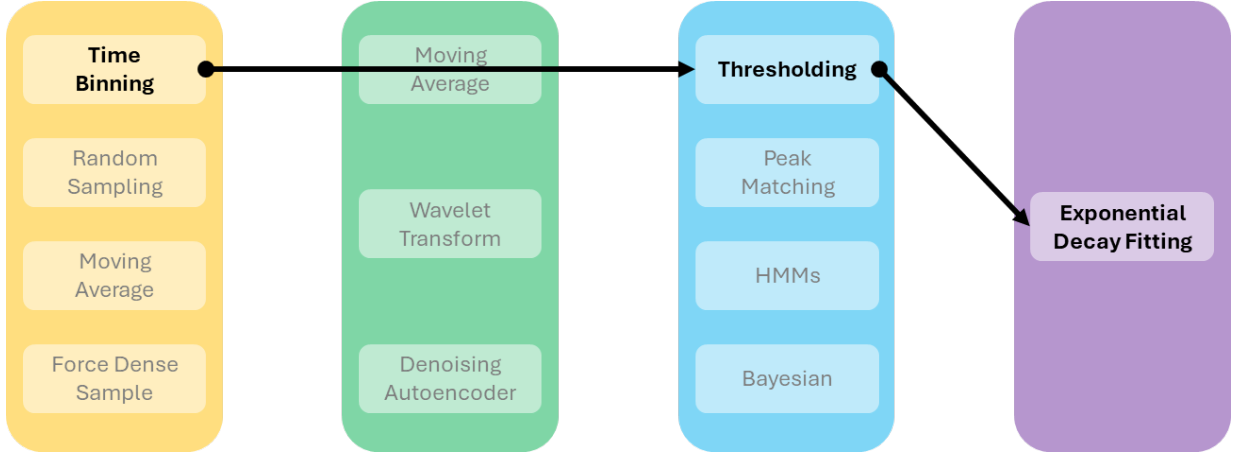


Figure 3.1: Method used to obtain **QD** tunneling rates by J. Kerski *et al.* [1] mapped in our framework. Pre-processing stage and variants (orange), denoising stage and variants (green), digitization stage and variants (blue), and tau statistics extraction (purple).

The reference values given by Mannel [1] were obtained by applying a direct thresholding approach (Sec. 2.3.2) without any prior denoising processes. An overview of the steps used is given in Figure 3.1 for easier comparisons with our developed methods.

During Phase 1, the gate voltage, laser excitation power, and magnetic field were held constant throughout the data file. Consequently, no dependencies related to the tau statistics (tunneling rates) were expected. A comprehensive analysis was conducted, examining all known parameters of our algorithm. One of the primary findings of this phase was that the deep learning-based denoising model (**DAEUBL**) was not necessary to achieve satisfactory results. Instead, the dual-tree complex wavelet transform (**DTCWT**) denoising method proved equally effective, offering a less computationally expensive solution. Additionally, several sensitive parameters in the execution process were identified during Phase 1. These parameters will be subject to further testing and evaluation in Phase 2.

3.2 Proof of Sampling Mechanisms

Through the testing of **QD RTSs** at different t_{bin} , we encountered the challenge of processing large datasets, often when exceeding 10M rows. To address this, we explored two main approaches with promising results: (1) random sampling from the full dataset and

(2) processing data in chunks, followed by chaining the results. One major concern was whether dividing the [RTSs](#) into chunks and then chaining the results would accurately represent the original signal. After some testing, we confirmed that this method is effective. Specifically, the average results from 10 chained processes closely mirrored those obtained from processing the entire dataset as a single unit. The standard deviation across the chained processes was minimal, confirming that this method can preserve the integrity of the [RTS](#). This approach later led to the development of the force-dense region method, which will be discussed in detail in Phase 3 ([Chapter 5](#)).

For the current phase, we tested random sampling without incorporating any prior domain knowledge (and without any intervention). The primary goal was to determine whether randomly sampled data could be a reliable representation of the entire dataset. Our analysis showed that random sampling yielded effective results, as the derived $\bar{\tau}_{\text{low}}$ and $\bar{\tau}_{\text{high}}$ statistics exhibited sufficiently low standard deviations, suggesting that random samples could indeed serve as proxies for the full data set.

t_{bin}	Sampled Percentage	Equivalent RTS Length	$\bar{\tau}_{\text{low}}$ [μs]	$\bar{\tau}_{\text{high}}$ [μs]
100 μs	100%	3M	1555 ± 0	1378 ± 0
	10%	300K	1702 ± 84	1354 ± 26
	1%	30K	1840 ± 136	1386 ± 36
10 μs	10%	3M	740.1 ± 6.4	435.1 ± 8.3
	1%	300K	735.8 ± 35.3	402.9 ± 25.0
	0.1%	30K	742.7 ± 116.2	440.3 ± 46.9

Table 3.1: $\bar{\tau}_{\text{low}}$ and $\bar{\tau}_{\text{high}}$ of an exemplar time binned [RTSs](#). No additional pre-processing besides time binning. [DTCWT](#) denoised and peak matching digitization, bit flip error correction with max error window size of 4.

[Table 3.1](#) presents the results of this random sampling experiment, comparing the $\bar{\tau}_{\text{low}}$ and $\bar{\tau}_{\text{high}}$ values for different sampling percentages across two different time bin widths ($t_{\text{bin}} = 100 \mu\text{s}$ and $10 \mu\text{s}$). These results were obtained using [DTCWT](#) denoising and peak matching digitization, with bit-flip error correction (maximum error window size of 4). In the absence of [MA](#) preprocessing (discussed in [Sec. 2.1.2](#)), differences in $\bar{\tau}_{\text{low}}$ and $\bar{\tau}_{\text{high}}$ across different t_{bin} values are evident. These discrepancies prompted further investigation, leading to the development of the [MA](#) preprocessing method, which will be

utilized in later phases. For now, the results indicate consistency between the different sampling percentages within the same t_{bin} , suggesting that random sampling is a viable strategy for processing large datasets that cannot be handled by the algorithm at once.

3.3 Results

In this phase, we aimed to test the established framework for processing the QD RTSs data by executing multiple analysis "chains" across different t_{bin} values and sampling approaches. The term "chain" here refers to a set of sequential steps comprising pre-processing, denoising, digitization, and tau statistics extraction, with each step executed using different variants. Three primary analysis chains were defined within the scope of Phase 1, as shown in Figure 3.2, which include the thresholding chain, wavelet chain, and autoencoder chain.

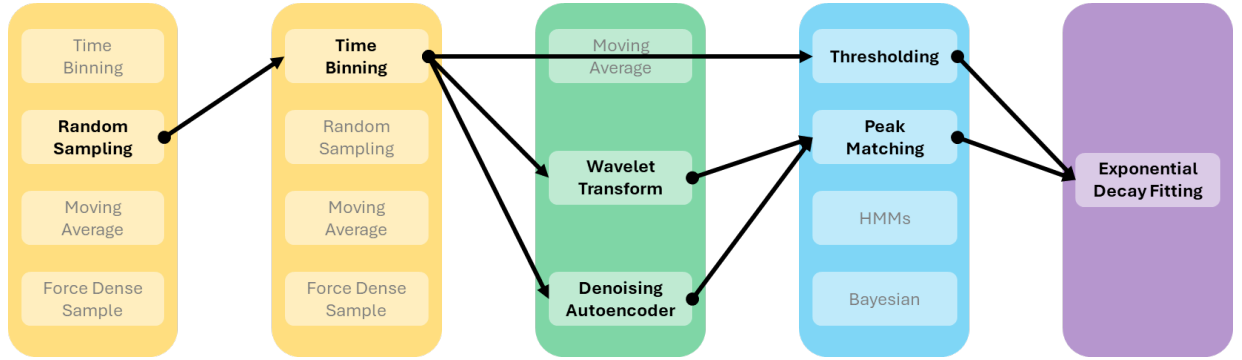


Figure 3.2: Method used in Phase 1 to study QD RTSs statistics in our framework. Thresholding chain (top path), wavelet chain (middle path), and autoencoder chain (bottle path).

We began by analyzing a RTS dataset binned at $t_{\text{bin}} = 10 \mu\text{s}$, as the reference tunneling rates ($\Gamma_{\text{in}} = 2072 \text{ Hz}$, $\Gamma_{\text{out}} = 1456 \text{ Hz}$) were derived at this resolution in prior studies by Kerski *et al.* [1]. This resolution was chosen because it allowed us to process the entire dataset without requiring any subsampling, thereby establishing a direct comparison with previous results.

Using the thresholding method, we replicated the conditions from [1] with the same threshold value (1.5). While the resolution of $t_{\text{bin}} = 10 \mu\text{s}$ was still low enough (i.e., the

data length was manageable), we were able to process the entire [RTS](#) dataset without taking any subsamples (top path in [Figure 3.2](#) without random sampling). This resulted in tunneling rates of $\Gamma_{\text{in}} = 2081$ Hz and $\Gamma_{\text{out}} = 1450$ Hz, which closely matched the reference values of $\Gamma_{\text{in}} = 2072$ Hz and $\Gamma_{\text{out}} = 1456$ Hz. These results confirmed that our method was valid and aligned with the benchmark data. While incorporating random sampling, the tunneling rates extracted from the tau statistics of approximately 20 subsamples were all within 5% of the full dataset values obtained using the thresholding method (top path in [Figure 3.2](#)). This further demonstrates that random sampling provides reliable results even when the full dataset is not processed, an important finding for handling larger datasets or higher resolutions.

Next, we evaluated two additional denoising variants, the denoising autoencoder-based variant ([DAEUBL](#)) and the wavelet transform variant ([DTCWT](#)), against the baseline thresholding variant. This led us to perform two more analysis chains: one following the bottom path in [Figure 3.2](#) and the other using the middle path. Both chains delivered results that were closely aligned with the thresholding method. Specifically, for the wavelet chain (with [DTCWT](#) denoising), we obtained tunneling rates of $\Gamma_{\text{in}} = 2246$ Hz and $\Gamma_{\text{out}} = 1329$ Hz, within 10% of the values obtained through the thresholding chain (top path) and the reference tunneling rates. In addition, when random sampling was applied to these methods, the subsamples also exhibited tunneling rates within 5% of the full dataset values, further validating the use of subsamples as a reliable means of analysis. These results indicate that as we move to higher resolutions (i.e., smaller t_{bin}) where processing the entire dataset becomes computationally impractical, it is safe to rely on non-empty (see [Figure 2.2](#)) subsamples for analysis. The consistency of the results across different denoising methods and random sampling approaches suggests that our framework is robust and flexible, also eliminating the potential that it was concerned in [DTCWT](#) denoising resulting in the inconsistency seen in tunneling rates across different t_{bin} in [Table 3.1](#).

To further explore the dynamics of the [RTS](#), we analyzed the data using time lag plots (TLPs), a powerful visualization tool that can highlight both level-corresponding peaks and between-level transitions. A time lag plot maps the signal at time t on the x-axis to the signal at time $t + \tau$ on the y-axis, where τ is the lag, making it useful for identifying patterns and correlations in signals with discrete state transitions, especially level-corresponding peaks (diagonal) and between level transitions (off-diagonal) discussed by the [RTSs](#) in [QDs \[7\]](#).

In the case of $t_{\text{bin}} = 10 \mu\text{s}$, the limited number of unique amplitude levels (approximately 20) made it difficult to observe any distinct features in the time lag plots. The aggregated detector counts ranged between 0 and 20 within each t_{bin} , which led to a highly discrete representation of the signal and less clarity in visualizing transitions. This is illustrated in Figure 3.3, where the noisy RTS (shown in grey) and the digitized RTS (shown in blue) are compared, and both time lag plots (middle and right) appear quite noisy and lacking obvious features.

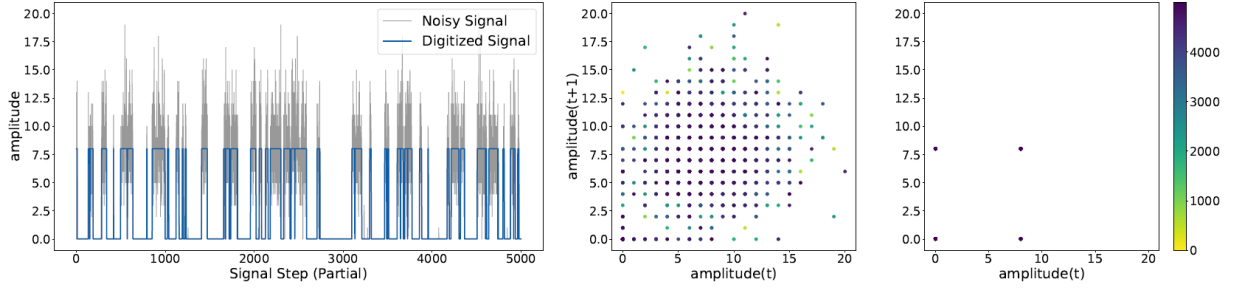


Figure 3.3: Time lag plot of a portion of a dataset with $V_l = 50 \text{ nV}$ and $V_g = 0.3743 \text{ V}$ under $t_{\text{bin}} = 10 \mu\text{s}$. Noisy RTS as grey, digitized RTS as blue (left). Time lag plot based on noisy RTS (middle). Time lag plot based on digitized RTS (right).

However, when we lowered the resolution to $t_{\text{bin}} = 100 \mu\text{s}$, with the maximum aggregated detector counts exceeding 100, clear features began to emerge in the time lag plots. Despite being a lower resolution, it allowed for a better representation of the transitions between the two levels, characteristic of a 1-trap RTS. As shown in Figure 3.4, the most dense regions in the plot are located around (0,0) and (80,80), corresponding to the two signal states, with $\Delta_{\text{RTS}} \approx 80$. The visual clarity of these transitions in the time lag plot indicates that $t_{\text{bin}} = 100 \mu\text{s}$ is a more suitable resolution for visualizing key RTS characteristics.

As discussed in Sec. 1.3.2, the study of QD data at higher resolutions is preferred, and prior studies such as Kerski *et al.* [1] explored time bins down to $t_{\text{bin}} = 10 \mu\text{s}$ while still achieving reliable results. However, our work in Phase 1 primarily focused on getting familiar with the dataset, and we have not yet thoroughly tested higher resolutions. Some preliminary testings while pushing beyond $t_{\text{bin}} = 10 \mu\text{s}$ brings new challenges. For instance, at a finer time resolution (i.e., smaller t_{bin}), fewer detector clicks are aggregated within each bin, leading to sparser data. This “emptiness” of the bins poses challenges for effective data analysis, as discussed in Figure 2.2. At extremely small time bins, such as $t_{\text{bin}} = 1$

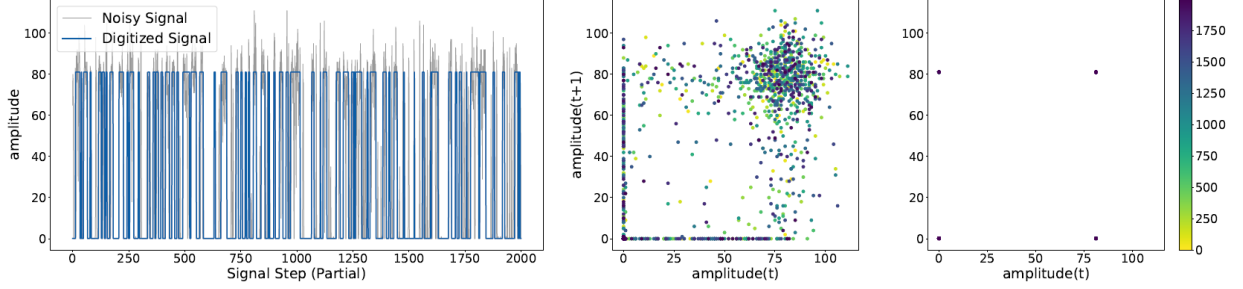


Figure 3.4: Time lag plot of a portion of a dataset with $V_l = 50$ nV and $V_g = 0.3743$ V under $t_{\text{bin}} = 100$ μ s binning. Noisy RTS as grey, digitized RTS as blue (left). Time lag plot based on noisy RTS (middle). Time lag plot based on digitized RTS (right)

ns or 10 ns, covering a large dataset (e.g., 300 seconds of signal) requires processing an immense number of rows. For instance, a RTS with $t_{\text{bin}} = 1$ ns that spans 3M rows would only cover 3 ms of the full dataset. Similarly, using $t_{\text{bin}} = 81$ ps, the same length in 3M rows would only span just 0.2 ms, resulting in a nearly zero-filled signal.

This extreme sparsity rendered smaller bins impractical for our current study, as the signals they produced were not representative of the full dataset and contained limited useful information. Moreover, the amount of computational cost of analyzing bins at this resolution, which would require processing upwards of 3.8T rows for $t_{\text{bin}} = 81$ ps, made it unfeasible. To address the sparsity issue encountered with smaller t_{bin} values, we developed a forced dense sampling mechanism, described in Sec. 2.1.2. This mechanism introduces human intervention (in contrast to random sampling) and focuses on extracting meaningful data from subsamples, ensuring that analysis can still be performed even when full datasets are impractical to process. This approach lays the groundwork for future studies involving higher-resolution RTS datasets and opens the door for more detailed analysis without sacrificing computational efficiency.

3.4 Summary

In this phase, we tested the framework for processing and analyzing QD RTSs data using three different analysis chains: thresholding, wavelet transform (DTCWT), and autoencoders (DAEUBL). While the execution arguments for each method—such as decomposition levels and wavelet thresholds in the DTCWT method—were specified manually

in this phase, one of the primary goals for Phase 2 is to develop automatic methods for determining these parameters. Additionally, Phase 2 will focus on pushing the analysis to higher time resolution to extract more detailed insights from the data.

Several key lessons were learned during this phase. First, we validated the use of random sampling, demonstrating that subsamples provide accurate tunneling rate estimates without requiring the analysis of the entire dataset. This significantly reduces the computational cost. Additionally, we confirmed that there was no “heating up” period during data collection, meaning that all recorded data correspond to the same physical conditions and are valid for taking a random sample. While this confirms we do not need to collect excessively long datasets (e.g., 300 s), recording too short a period (e.g., 0.3 s) could still miss essential signal characteristics.

Furthermore, we found that the [DTCWT](#) method was computationally less expensive than the other approaches for analyzing simple two-level [RTSs](#) with minimal pink noise ([QD RTSs](#)). This makes it a promising candidate for further development and automation. Overall, the framework we developed showed promise in effectively analyzing [RTSs](#) data, providing a strong foundation for the next phase of development, where the focus will be on automation and analysis at higher resolutions.

Chapter 4

Results and Discussions: Phase 2

4.1 Goal: Primary Analysis on Full Datasets

In Phase 2, the main objective is to conduct a large-scale analysis using the full dataset without the intervention of subsampling techniques: Random sampling was still employed within the full dataset to avoid potential biases introduced by human intervention. This approach allowed for a more objective analysis and gave us a broader picture of the data dynamics. This phase involved the comparison of algorithms across the entire data range, and primary results across the full datasets, allowing us to further test the capabilities of our developed framework for studying [QD RTSs](#).

The source data consisted of 80 datasets:

- 20 gate voltages $V_g \in [3520, 3603, 3635, 3648, 3666, 3683, 3698, 3713, 3743, 3752, 3761, 3766, 3781, 3796, 3811, 3823, 3852, 3870, 3920, 4010]$ V.
- 4 laser excitation powers $V_l \in [3.7, 16, 50, 200]$ nV.
- Fixed magnetic field conditions $B = 2$ T.

For Phase 2, 480 random samples were drawn, with each sample containing between 300 K to 3 M rows, across different combinations of V_g , V_l , and t_{bin} . i.e. 1 [RTS](#) for each of the 6 t_{bin} values in [Table 4.1](#) for each of the 80 datasets. These samples were analyzed to

evaluate their characteristics, but not all of them contributed to the final results analysis. The random sampling method on smaller t_{bin} often resulted in very sparse RTS samples, which impacted the quality of the results.

t_{bin}	Sampled Percentage	Sample Duration
100 μs	1.00%	≈ 3.00 s
10 μs	0.10%	≈ 0.30 s
1 μs	0.10%	≈ 0.30 s
100 ns	0.10%	≈ 0.30 s
50 ns	0.05%	≈ 0.15 s
10 ns	0.01%	≈ 0.03 s

Table 4.1: RTS sampling specifications used for Phase 2 analysis.

A key aspect of Phase 2 was examining the correlation between gate voltage (V_g) and t_{bin} in terms of sample validity rates. Validity here refers to whether a random sample contains enough meaningful data (i.e., non-empty photon detection intervals) to be interpretable for further analysis, briefly introduced in Figure 2.2. Based on the random sampling approach, we generally expect over 80% of the samples to be valid. However, we observed a noticeable decrease in sample validity as the t_{bin} decreased, which was expected.

Smaller time bins capture more fine-grained data, leading to longer full signal lengths when expanded from the base resolution $t_{\text{bin}} = 81$ ps. However, due to the limit of RTS sample length (in rows), this resulted in lower sampling percentages as the resolution increased. Moreover, as t_{bin} gets smaller, fewer clicks are detected per bin, leading to sparse or "empty" regions in the sampled data. This phenomenon significantly lowered the sample validity, particularly in smaller time bins like $t_{\text{bin}} = 10$ ns, where random sampling is much more likely to capture regions devoid of photon detections.

We also observed that higher V_g resulted in a decrease in quantum dot activity, leading to more empty regions in the data. This is consistent with the behavior of quantum dots, where higher gate voltages cause the quantum dot to become more occupied, resulting in fewer photons being detected. As illustrated in Figure 4.1, the total photon detector counts per second (cps) show clear dependencies on both V_g and V_l . Generally, higher laser excitation powers (V_l) and lower gate voltages (V_g) produced datasets with higher photon detection rates, leading to denser data.

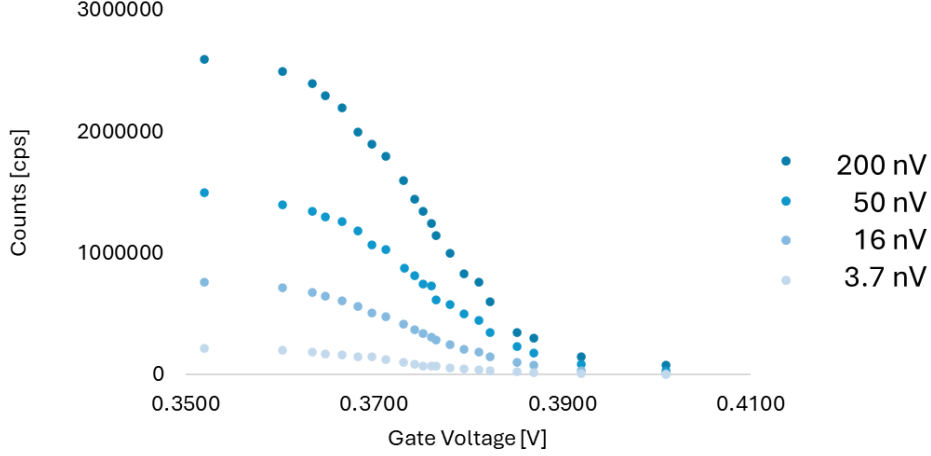


Figure 4.1: Total photon detector counts per seconds (cps) for each dataset for each 20 V_g (as scatters along a series) and each 4 V_l (as series).

This observation supports the idea that when the aim is to maximize data density (for high-resolution analysis), one would favor higher V_l and lower V_g settings. However, for this thesis, we are primarily interested in studying the dependencies between both V_l and V_g , and thus we will continue to evaluate all datasets equally, rather than focusing exclusively on those with the highest photon detection densities.

To improve the sample validity rate at smaller t_{bin} , a more targeted approach in Phase 3 will involve actively identifying and extracting dense signal regions, rather than relying on random sampling. This will help avoid the emptiness issue and improve the reliability of high-resolution analysis at nanoseconds t_{bin} , where random sampling struggles. Furthermore, intervention to extract regions with more photon events will be key to achieving better consistency and further refining the resolution improvements from 2 μs down to 100 ns and beyond.

4.2 Algorithm Comparison

A comparison of the algorithms applied to the full range of available laser voltages (V_l) and gate voltages (V_g) datasets will be performed, focusing on their effectiveness in

preprocessing and denoising the QD RTS data. A developed batch execution pipeline was implemented to streamline the process and compare results across different algorithmic approaches. We compared three distinct chains of algorithms given in Figure 4.2.

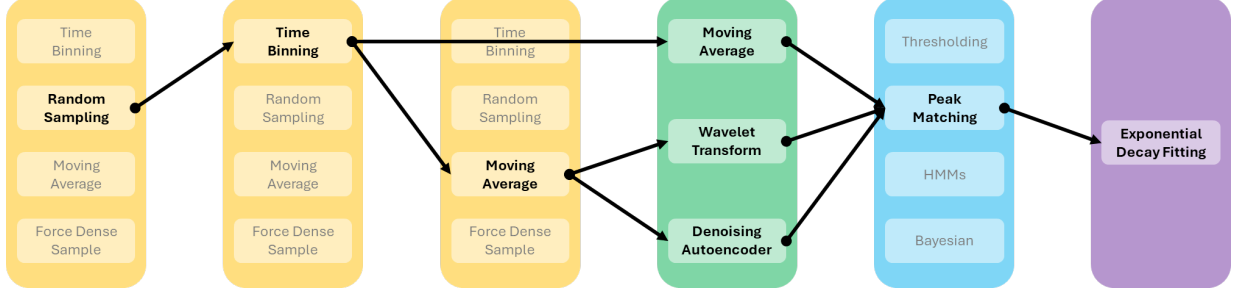


Figure 4.2: Methods used in Phase 2 to perform algorithm comparison by QD RTSs statistics in our framework. MA chain (top path), wavelet chain (middle path), and autoencoder chain (bottom path). 4 different colors of blocks from left to right represent the 4 stages in order, namely pre-processing, denoising, digitization, and tau extraction.

In Phase 1, both DTCWT and DAEUBL were found to be effective at denoising, although their relative performance was similar across different tested t_{bin} . However, Phase 2 provided clearer insights into the advantages of DTCWT, especially for smaller time bins. The DTCWT method was consistently more advantageous when denoising QD data at higher resolutions (i.e., smaller t_{bin}), highlighting its robustness in handling fine time scales where noise becomes more prominent.

In contrast, the combination of the MA and DAEUBL methods encountered limitations. They struggled to effectively denoise data starting at $t_{\text{bin}} = 100$ ns and smaller, even when preprocessing steps like MA were applied to support DAEUBL. This highlighted the superior benefits of the DTCWT method on QD data with minimal pink noise. The wavelet-based approach not only offered speed improvements but also demonstrated a greater ability to separate signal from noise at high resolutions. With its wavelets decomposition, DTCWT maintained denoising effectiveness, even when handling small time bins where traditional autoencoder approaches struggled. This difficulty was primarily due to the gap between the synthetic data on which the DAEUBL model was trained and the real-world semiconductor data, which presented more complex noise characteristics.

Overall, the results underscore the **DTCWT** method’s superior performance for high-resolution data (e.g., 100 ns and below). While **DAEUBL** performed well at larger t_{bin} sizes, the wavelet-based method demonstrated the ability to scale effectively across varying time resolutions, making it the preferred choice for denoising **QD** data in the context of smaller time bins. However, this does not imply that either denoising method is universally superior across all **RTS** datasets. **DAEUBL** was trained on synthetic data tailored to handle more complex **RTS** patterns with high levels of pink noise [7]. Based on the studies conducted in this thesis, it is safe to conclude that **DTCWT** is particularly well-suited for real semiconductor **RTS** data, where white noise is the dominant source, such as in the case of **QD**. The final determined method used for Phase 2 results and analysis is given in Figure 4.3.

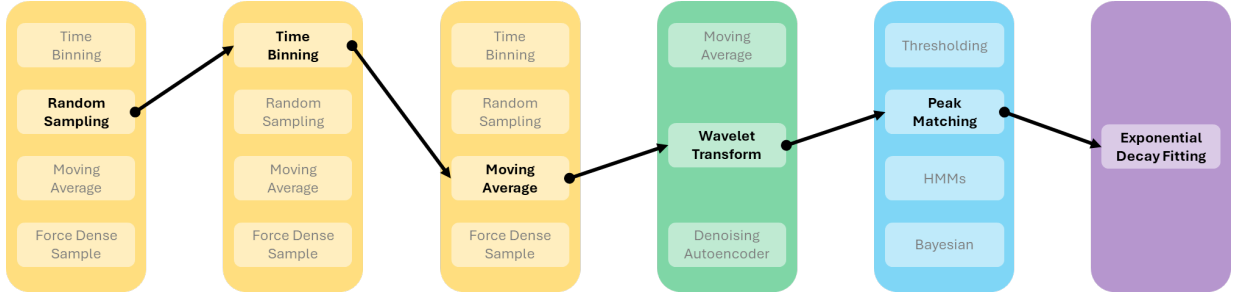


Figure 4.3: Method used in Phase 2 to study **QD RTSs** statistics in our framework. Wavelet chain by **DTCWT** denoising.

Before processing all **RTS** samples, we first tuned the key parameters for the **DTCWT** method, including the most suitable pre-processing **MA** window size, **DTCWT** threshold, and **DTCWT** decomposition levels. These parameters were adjusted to ensure optimal performance in denoising and extracting the underlying features.

As shown in Figure 4.4, certain trends emerged during the tuning process. For instance, when setting the **DTCWT** threshold to 10 (recall the effect of threshold levels in Figure 2.7), the effects of varying the **MA** window size became apparent. A larger **MA** window increased the distinction between the identified Δ_{RTS} levels, particularly evident between the intensity difference between the two red peaks in the density-intensity plots (right subplot of each grid). At low **MA** windows, the distinction between these peak values diminished, which was particularly noticeable in the corresponding τ statistics. These deviations became more pronounced when compared to reference values, indicating that insufficiently large **MA** windows can lead to less accurate denoising.

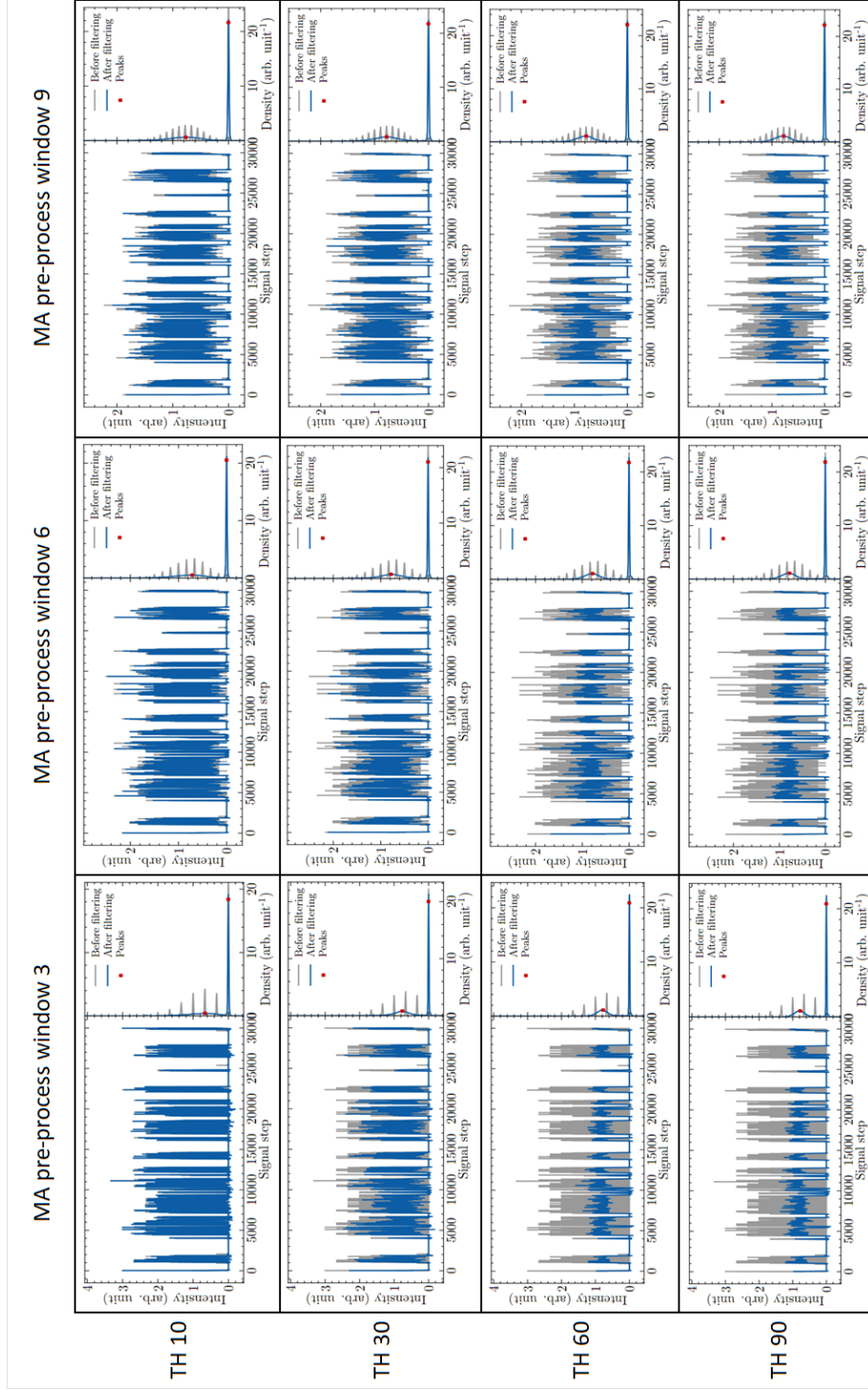


Figure 4.4: A single randomly sampled QD RTS at $t_{\text{bin}} = 1 \mu\text{s}$. Tested with various pre-processing MA window size and DTCWT wavelet threshold (TH) levels.

On the other hand, when examining an [MA](#) window size of 9, the effects of increasing the threshold were visually more distinct in the step-intensity plots (left subplot of each grid). Here, the denoised (blue) signal showed a significant reduction in white noise compared to the raw signal (grey). This was further confirmed by the density-intensity plot, where the high-state peak (higher intensity peak) showed an increased density, visually represented by a steeper slope. Similar to the trends observed with [MA](#) window size, increasing the threshold value resulted in more accurate τ statistics. Based on these visual observations in [Figure 4.4](#), we settled on using a pre-processing [MA](#) window size of 9 and a threshold of 90 for $t_{\text{bin}} = 1 \mu\text{s}$ in the [DTCWT](#) denoising process. Although studies like this across different t_{bin} may give more information on an optimal pre-processing [MA](#) window size and the threshold value for each individual t_{bin} , but these findings would be based on a specific dataset (a specific V_g and V_l). This is not ideal considering each different dataset has a varying density of detector activities.

Similar testings were conducted, only when the threshold was tripled at each [MA](#) window size on $t_{\text{bin}} = 100 \text{ ns}$, revealing consistent visual trends. We thus derived a preliminary scaling rule for threshold levels in [DTCWT](#): for every 10-fold increase in resolution, the threshold should be increased by a factor of 3. Similarly, for the [MA](#) window size, we determined that a 10-fold increase in resolution required a 10-fold increase in [MA](#) window size, as illustrated in [Figure 2.4](#) in prior discussions.

Another key parameter for the [DTCWT](#) denoising method is the wavelet decomposition level, which relates to the structure of the dual-tree complex wavelet transform as discussed in [Sec. 2.2.3](#). After conducting tests, we established a rule of thumb for determining the appropriate decomposition levels. For an [RTS](#) length of 300K rows, a decomposition level of 5 was suitable. For datasets with greater lengths, we applied the formula [Equation 4.1](#). This scaling formula allows us to adapt the decomposition level based on the length of the [RTS](#), ensuring effective denoising across different sample sizes.

$$\text{Decomposition level} = 5 + \log_2 \left(\frac{\text{rows}}{300000} \right). \quad (4.1)$$

4.3 Validation on Synthesized Data

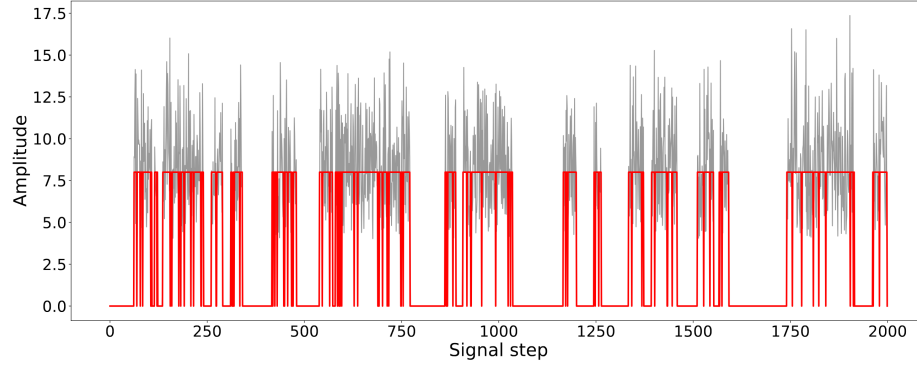
To demonstrate the correctness and effectiveness of our proposed method compared to the community’s established approach [1], we validate the framework using synthesized data under controlled conditions. This validation serves to highlight the method’s ability to achieve higher resolution while maintaining accuracy.

For the synthesized signal, we used parameters representative of real QD RTS, including tunneling rates of $\Gamma_{\text{in}} = 2000$ Hz and $\Gamma_{\text{out}} = 1500$ Hz, a $\Delta_{\text{RTS}} = 8$, and a time bin resolution of $t_{\text{bin}} = 10 \mu\text{s}$. The mean dwell times corresponding to the tunneling rates were used to construct a 2-level RTS, to which Gaussian noise was added. These parameters emulate the typical characteristics of real QD RTS data. We analyzed this synthesized signal using two methods:

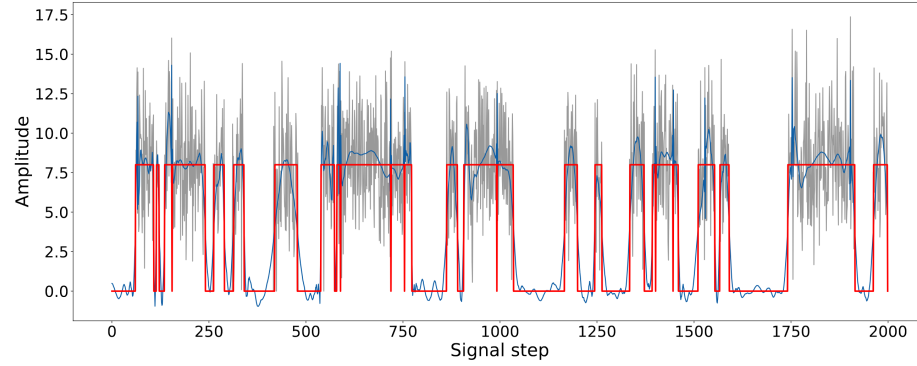
1. The baseline method based on the approach outlined in Figure 3.1, using a threshold of 4 (half the Δ_{RTS}).
2. The selected method from our proposed framework, as illustrated in Figure 4.3 with populated parameters based on pre-defined rules.

The RTS analysis processes for both methods are visualized in Figure 4.5. The raw noisy RTS is shown in grey, the denoised RTS in blue, and the digitized RTS in red. The results indicate that the baseline method, which lacks a robust denoising step, significantly underperforms, with noise skewing the digitization results. Misidentification of steps led to highly inaccurate estimates of tunneling rates, especially for Γ_{in} . In contrast, our proposed method effectively denoised the signal, enabling a more accurate digitization and tunneling rate extraction. The tunneling rates extracted by each method were compared to the reference values of $\Gamma_{\text{in}} = 2000$ Hz and $\Gamma_{\text{out}} = 1500$ Hz:

- **Baseline Method:** $\Gamma_{\text{in}} = 11949$ Hz and $\Gamma_{\text{out}} = 1536$ Hz.
- **Proposed Method:** $\Gamma_{\text{in}} = 2510$ Hz and $\Gamma_{\text{out}} = 1447$ Hz.



(a) Digitization using the baseline method as Figure 3.1.



(b) Denoising and digitization using the proposed method as Figure 4.3.

Figure 4.5: Validation of effectiveness on proposed method compared with community method as the baseline on synthesized 2-level QD RTS with interval of $t_{\text{bin}} = 10 \mu\text{s}$. Raw noisy RTS in grey, denoised RTS in blue, digitized RTS in red.

The baseline method suffered from a severe overestimation of Γ_{in} , with a 500% deviation due to the misidentification of τ_{high} as separated transitions. This highlights the inadequacy of the baseline method in handling high-resolution, noisy signals. In contrast, our proposed method reduced this error to 25%, providing a much closer estimate. For Γ_{out} , both methods achieved errors below 4%, as τ_{high} misidentifications have less impact on the τ_{low} exponential decay fitting when the shortest binned intervals are ignored, as discussed in Sec. 2.4. While this comparison was conducted on a single synthesized RTS, beyond the scope of this thesis, a broader statistical analysis over multiple RTS with different controlled conditions will provide a more comprehensive evaluation of the method's robustness and reliability.

4.4 Laser and Gate Voltages Dependency Towards Tunneling Rates

The emptiness of RTS sample can result in either or both of $\bar{\tau}$ failing to fit. We define it as valid when both $\bar{\tau}_{\text{high}}$ and $\bar{\tau}_{\text{low}}$ can be fitted. As shown in Table 4.2, out of the 480 RTS samples analyzed, most samples succeeded at the resolution of 1 μs and coarser, whereas samples at a resolution of 50 ns and finer have mostly failed. Hence $t_{\text{bin}} = [50 \text{ ns}, 10 \text{ ns}]$ has been excluded for remaining visualizations in Phase 2. Additionally, upon investigating the results of $t_{\text{bin}} = 100 \mu\text{s}$, it is decided to be removed as well due to the lack of resolution to provide meaningful $\bar{\tau}$ statistics. Although $t_{\text{bin}} = 100 \mu\text{s}$ gives the highest sample valid rate or density, it has become too coarse and is no longer able to show distinct $\bar{\tau}$ transitions.

t_{bin}	Sample Validity Rate
100 μs	83.75%
10 μs	75.00%
1 μs	70.00%
100 ns	41.25%
50 ns	11.25%
10 ns	0.00%
Overall	46.88%

Table 4.2: Phase 2 RTS sample validity rate according to sampling specifications in Table 4.1.

The initial approach was to perform a linear fit on the tunneling rates based on the observed data trends. The most obvious observation is a general increase in tunneling-in rates and a corresponding decrease in tunneling-out rates at higher V_g as shown in Figure 4.6. While the results obtained did not show highly distinguishable tunneling rates across different time bins, some interesting trends may emerge when focusing on higher resolutions with targeted datasets, as discussed in Chapter 6. The primary goal here was to test and enhance the algorithms while applying them to real-world data. Upon examining the changes across different laser voltages (V_l), we generally observe smaller slopes at higher V_l , and vice versa, shown in Figure 4.7. However, as mentioned earlier, no obvious or consistent trends were noted at the intersection points. Further investigation is required with higher-resolution data and targeted sampling.

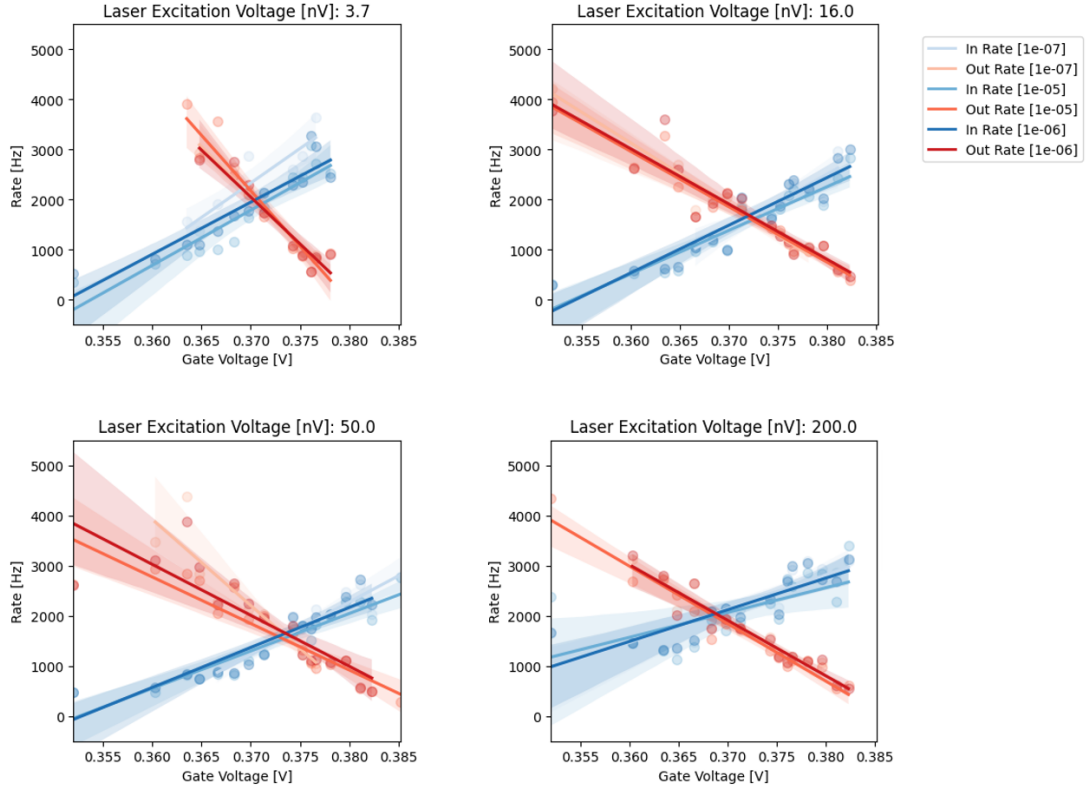


Figure 4.6: Scatter plot with linear regression fits for Γ_{in} and Γ_{out} vs V_g , across available t_{bin} results (legend series) for different V_l (subplots).

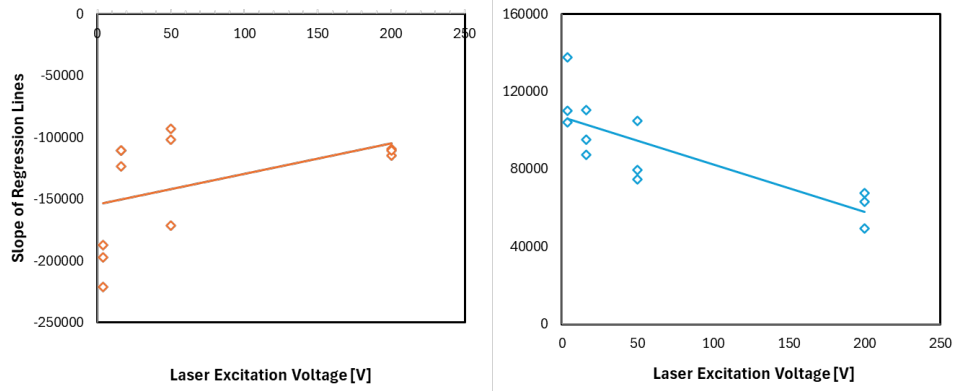


Figure 4.7: Scatter plot with linear regression fits of slope for Γ_{out} (left) and Γ_{in} (right) vs V_l .

4.5 Laser and Gate Voltages Dependency Towards Tunneling Rates Asymmetry

While the initial linear fitting of the tunneling rates revealed some promising trends, it is essential to apply the fitting method more commonly used within the QD community [1, 28] for a more comprehensive analysis. Specifically, we focus on the asymmetry between tunneling-in and tunneling-out rates, which provides insight into the system's behavior under varying gate voltages. The asymmetry, denoted as A , is computed from the tunneling rates Γ_{out} and Γ_{in} of the QD, as shown in Equation 4.2:

$$A = \frac{\Gamma_{\text{out}} - \Gamma_{\text{in}}}{\Gamma_{\text{out}} + \Gamma_{\text{in}}}. \quad (4.2)$$

Once computed, the asymmetry is plotted against the gate voltage V_g , and a Fermi-Dirac function is applied to fit the data. The fitting parameters depend on the magnetic field (which is constant for the datasets used in this thesis) and potentially other factors. The Fermi-Dirac fitting function is given by Equation 4.3, with V_g and 3 fitting parameters:

$$f_{\text{FD}}(V_g, c_1, c_2, c_3) = \frac{c_1}{1 + \exp\left(\frac{V_g - c_3}{c_2}\right)}. \quad (4.3)$$

Both tunneling rates and asymmetry are expected to follow a Fermi-Dirac-like behavior, as discussed in prior study [28]. In this analysis, we focus on the asymmetry for simplicity. The rationale behind why A (asymmetry) follows a Fermi-Dirac distribution is due to the behavior of the QD's tunneling rates, specifically γ_{01} and γ_{10} . In the context of QD in A. Kurzmann et al.'s definition [28], these tunneling rates are influenced by external factors, such as V_g and the energy level structure of the QD relative to the electron reservoir (Fermi energy).

- γ_{01} is the Γ_{in} in our definition, or the rate at which an electron tunnels into the QD (going from state 0, or unoccupied, to state 1, or occupied).
- γ_{10} is the Γ_{out} in our definition, or the rate at which an electron tunnels out of the QD (going from state 1, or occupied, back to state 0, or unoccupied).

The rates γ_{01} and γ_{10} for electron tunneling in and out of a QD are intrinsically tied to Fermi's Golden Rule, where the transition rate depends on the density of available states in the reservoir and the occupation probability. Crucially, the tunneling probability describes the likelihood of an electron tunneling, whereas the tunneling rate depends on the starting state (occupied or unoccupied) and whether the final state is empty. These dependencies arise because the tunneling rate must account for the Fermi-Dirac distribution, which governs the occupation probability of states in the reservoir.

Shifting the quantum dot energy level relative to the reservoir's Fermi energy leads to exponential suppression of tunneling as fewer states become accessible. Therefore, tunneling rates are described by a Fermi-Dirac framework, accurately capturing the temperature and energy dependence of the process. This relationship is particularly evident in the context of spin-resolved tunneling, where the behavior becomes more intricate, as spin states introduce an additional energy splitting (Δ). The QD community widely employs the Fermi-Dirac framework to model these rates, as shown in Equation 4.4, where the transition rates for spin-up ($\uparrow$) and spin-down ($\downarrow$) states are given by:

$$\begin{aligned}\gamma_{0\uparrow} &= \Gamma f_{\text{FD}} \left(\epsilon + \frac{\Delta}{2} \right) \\ \gamma_{0\downarrow} &= \Gamma f_{\text{FD}} \left(\epsilon - \frac{\Delta}{2} \right) \\ \gamma_{\uparrow 0} &= \Gamma \left[1 - f_{\text{FD}} \left(\epsilon + \frac{\Delta}{2} \right) \right] \\ \gamma_{\downarrow 0} &= \Gamma \left[1 - f_{\text{FD}} \left(\epsilon - \frac{\Delta}{2} \right) \right].\end{aligned}\tag{4.4}$$

- f_{FD} is the Fermi distribution
- ϵ is the orbital level energy relative to the Fermi energy of the electron reservoir
- Zeeman splitting Δ
- Temperature T
- Tunnel coupling strength Γ
- $\uparrow$ and $\downarrow$ to resolve spin up and spin down

By fitting the A using this framework, we aim to extract the relevant parameters that could potentially describe the underlying physics of the system, particularly the effects of V_g on tunneling rates in QD systems. Before categorizing our tunneling rates into sub-groups based on other variables such as V_l and t_{bin} , we first compute the asymmetry based on all Γ_{in} and Γ_{out} values obtained from the RTS samples and fit them using Equation 4.3. It is important to note that the fitting process requires all asymmetry values A to be positive. Therefore, we applied an offset of $+\min(A)$, such that the minimum y -value in the plot becomes 0. The resulting plot is shown in Figure 4.8, where each data point represents the result from a single RTS sample.

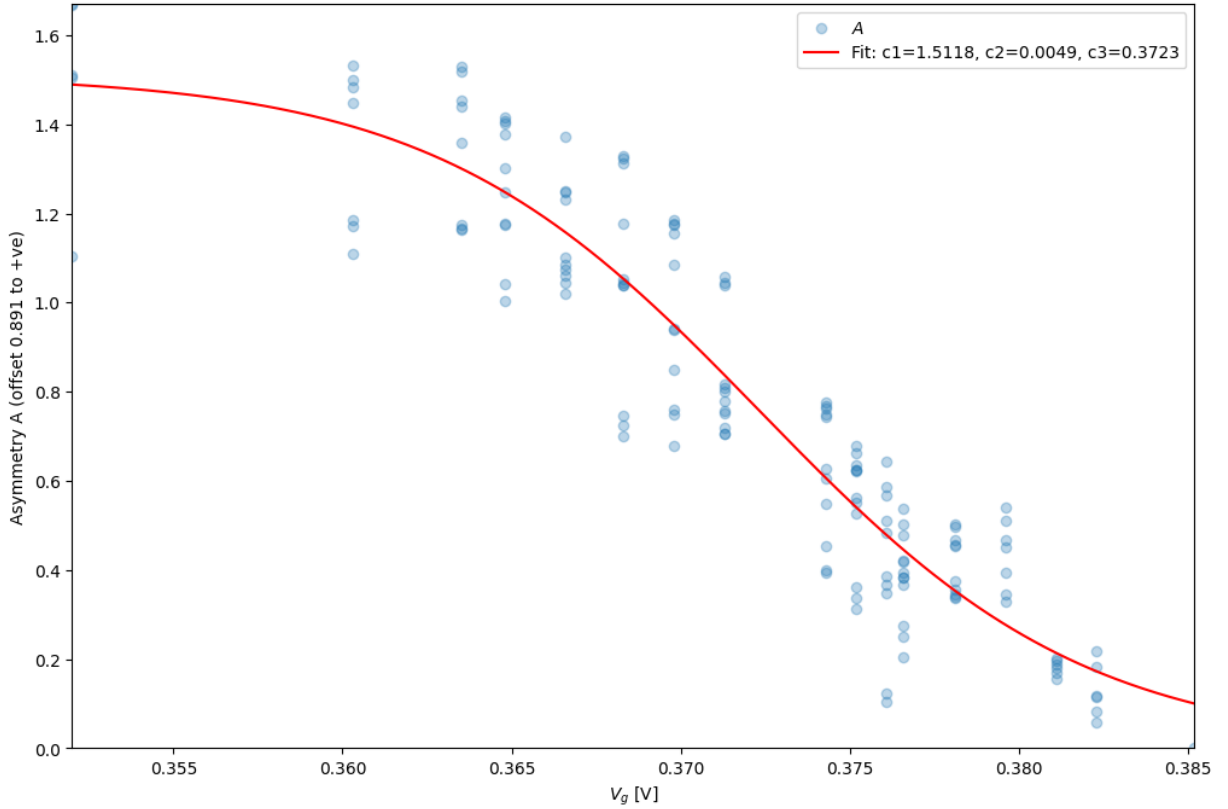


Figure 4.8: Scatter plot of asymmetry (offset to positives) vs V_g with fitted Fermi-Dirac (Equation 4.3). All t_{bin} and all V_l are grouped together.

The fitted curve follows the general Fermi-Dirac shape reasonably well, although there are some outliers. To further investigate the behavior, we maintained the same global $+\min(A)$ offset and split the plotting by V_I , as shown in Figure 4.9. It is worth noting that this offset leads to subplots where the minimum x-value may not necessarily be 0 in each case.

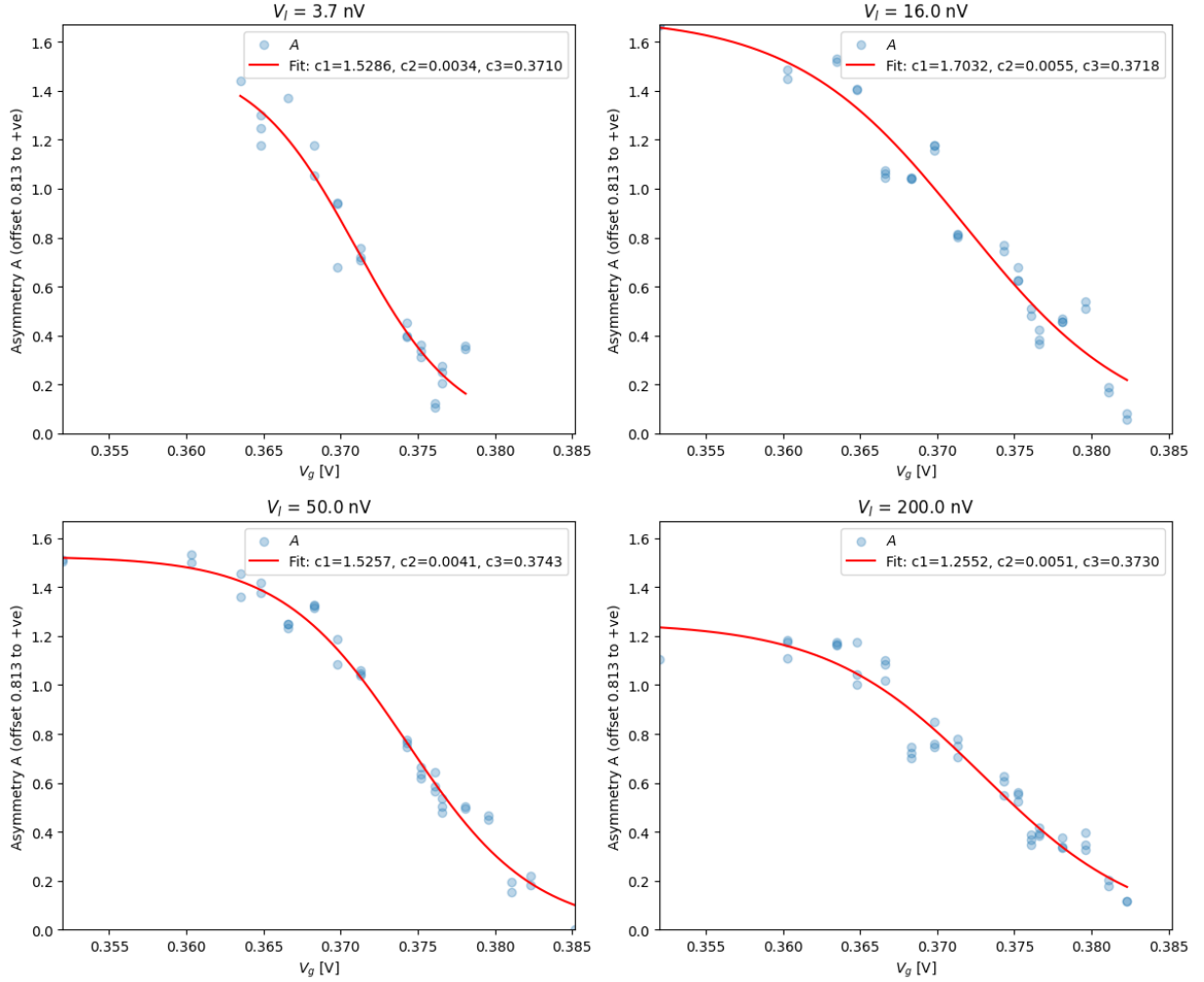


Figure 4.9: Scatter plot of asymmetry (offset to positives) vs V_g with fitted Fermi-Dirac (Equation 4.3). All t_{bin} are grouped together. All V_I are grouped into subplots.

From this split, we observed that while most subplots display smooth fits, some data points, especially in the top left ($V_l = 3.7$ nV), appear scattered or less smoothly fitted. This could indicate a need for better sampling or more representative data points. Our goal is to further split the data by t_{bin} as well and inspect the trends of the fitting parameters c_1 , c_2 , and c_3 . However, as seen in Figure 4.10, when we perform additional splits by t_{bin} , there are not enough plotted points to derive representative trends in the fitting parameters due to high variability in the random samples. The top left fitting in Figure 4.10 has even failed even with fine tuning of initial parameters.

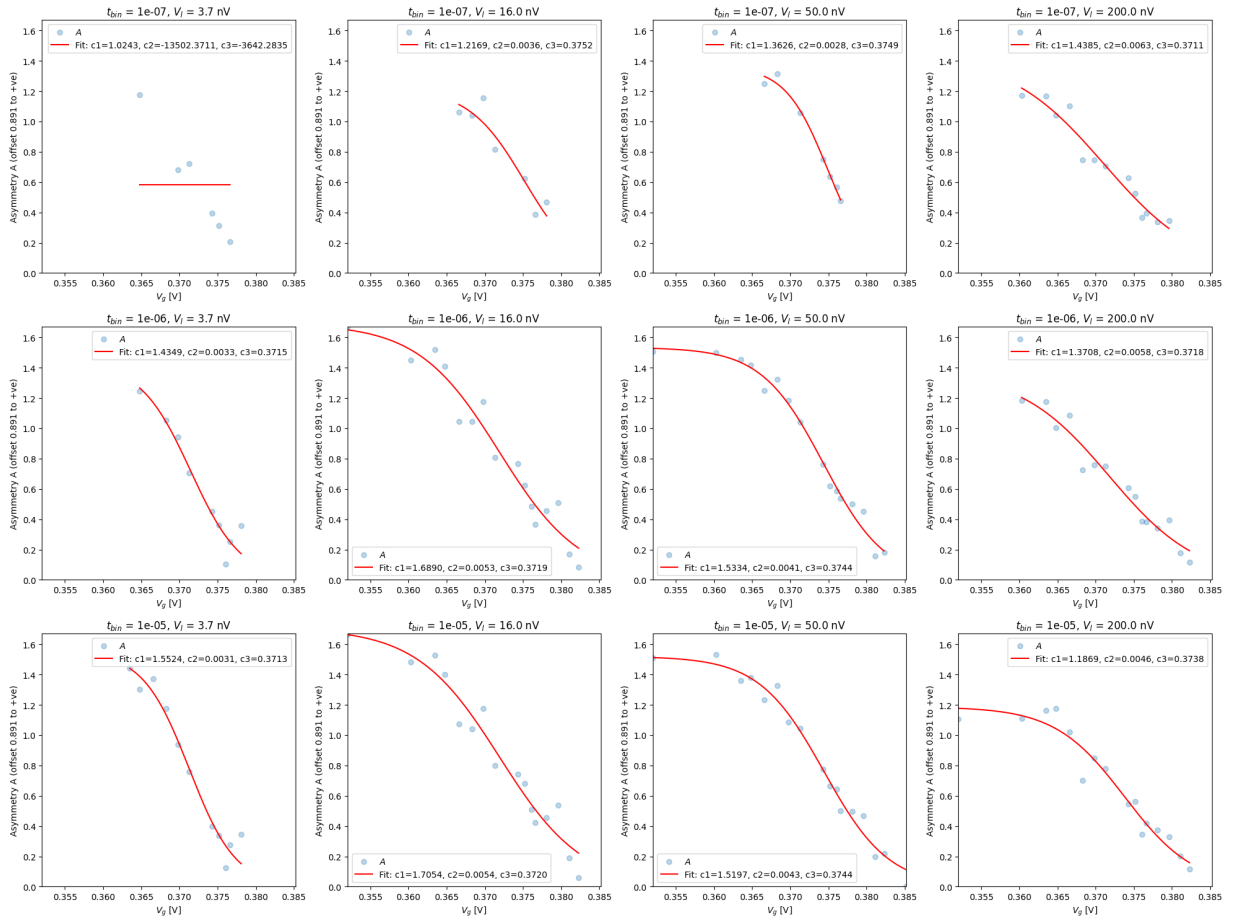


Figure 4.10: Scatter plot of asymmetry (offset to positives) vs V_g with fitted Fermi-Dirac (Equation 4.3). All t_{bin} are grouped into subplots (top to bottom). All V_l are grouped into subplots (left to right).

	$V_l = 3.7 \text{ nV}$	$V_l = 16.0 \text{ nV}$	$V_l = 50.0 \text{ nV}$	$V_l = 200.0 \text{ nV}$
$t_{\text{bin}} = 100 \text{ ns}$	Fitting failed see top left of Figure 4.10	$c_1 = 1.2169$ $c_2 = 0.0036$ $c_3 = 0.3752$	$c_1 = 1.3626$ $c_2 = 0.0028$ $c_3 = 0.3749$	$c_1 = 1.4385$ $c_2 = 0.0063$ $c_3 = 0.3711$
$t_{\text{bin}} = 1 \mu\text{s}$	$c_1 = 1.4349$ $c_2 = 0.0033$ $c_3 = 0.3715$	$c_1 = 1.6890$ $c_2 = 0.0053$ $c_3 = 0.3719$	$c_1 = 1.5334$ $c_2 = 0.0041$ $c_3 = 0.3744$	$c_1 = 1.3708$ $c_2 = 0.0058$ $c_3 = 0.3718$
$t_{\text{bin}} = 10 \mu\text{s}$	$c_1 = 1.5524$ $c_2 = 0.0031$ $c_3 = 0.3713$	$c_1 = 1.7054$ $c_2 = 0.0054$ $c_3 = 0.3720$	$c_1 = 1.5197$ $c_2 = 0.0043$ $c_3 = 0.3744$	$c_1 = 1.1869$ $c_2 = 0.0046$ $c_3 = 0.3738$

Table 4.3: Coefficients c_1 , c_2 , and c_3 fitted from Fermi-Dirac (Equation 4.3), for different combinations of V_l and t_{bin} values.

Despite the limited number of data points and the random sampling approach used for datasets each spanning 300 seconds (summarized in Table 4.3), some identifiable trends are emerging in the fitting parameters c_1 , c_2 , and c_3 . For example, higher V_l generally correlates with an increase in c_2 . This trend may reflect thermal effects induced by stronger laser excitation, where c_2 could be indicative of a temperature-dependent characteristic of the QD system. Since $c_2 \approx \frac{kT}{e} \backslash 15.2$, with both k and e being constants and a lever arm scaling of 15.2 according to the author of [1], we can infer an effective temperature of approximately 4 K for a value of $c_2 \approx 0.0050$, suggesting a similar temperature to the actual QD environment, bath cryostat at 4.2 K. This observation, while speculative given the random sampling of RTS data, highlights an interesting avenue for further exploration.

These initial results indicate potential insights that, with further refinement, could deepen our understanding of the QD system's response under different experimental conditions. Increasing the sampling density and employing multiple repetitions for the same sampling specifications can enhance the fitting quality. This refined data acquisition in Phase 3 will be instrumental in achieving a more reliable understanding of the physical mechanisms and in determining the robustness of trends in c_1 , c_2 , and c_3 in QD systems.

4.6 Summary

One of the most significant achievements in this phase was improving the time resolution by an order of magnitude. We successfully pushed the resolution from $t_{\text{bin}} = 2 \mu\text{s}$ [1] down to $t_{\text{bin}} = 100 \text{ ns}$, achieving a 20-fold improvement while maintaining high consistency in the results, even by taking random subsamples. However, starting at $t_{\text{bin}} = 50 \text{ ns}$ resolution (200 times improvement), the consistency of the results began to decrease significantly, primarily due to the random sampling of sparse data. Despite this, the algorithm still demonstrated promising potential at these smaller time bins.

Overall, Phase 2 confirmed that the framework could process larger datasets and achieve significantly higher resolutions. The algorithm framework showed excellent consistency at $t_{\text{bin}} = 100 \text{ ns}$ resolution and below, but further refinements will be necessary to handle the low consistency starting at $t_{\text{bin}} = 50 \text{ ns}$ resolution. In the next phase, we will focus on directly targeting dense signal regions for analysis, aiming for more stable and reliable fitting results before rushing to target higher resolutions.

Chapter 5

Results and Discussions: Phase 3

5.1 Goal: Enhanced Analysis with Dense Samples

In Phase 3, our primary objective is to evaluate the effectiveness of a dense sampling approach for RTS analysis, rather than solely focusing on maximizing time resolution. While previous phases achieved a time resolution of $t_{\text{bin}} = 100$ ns, aiming for higher resolution presents challenges due to limited data density at finer scales. Consequently, we prioritize establishing the reliability of the force-dense sampling mechanism under fixed time resolution (particularly on $t_{\text{bin}} = 1$ μs) across various values of V_g and V_l . Once validated, this framework can be expanded for future high-resolution studies focusing on a single dataset at the highest V_l and lowest V_g configurations.

The chosen approach reflects our goal of refining the sampling algorithm, minimizing empty or low-density samples, and improving sample validity rates to obtain clearer trends that align with underlying physical behaviors.

5.2 Appropriate Sampling Mechanism

Random sampling of the data at smaller t_{bin} has shown limitations, as denoising algorithms often fail with sparse data that lack sufficient signal density. To address this, we target blocks of the dataset exhibiting higher detector activity, tailoring block length by evaluating the relationship between row count and switch count across different t_{bin} settings.

Using 480 samples from Phase 2 (spanning 300 K to 3 M rows each across varying t_{bin} values; see Table 4.1), we assessed the switching frequency, defined here as transitions between non-zero and zero counts. i.e. `(amplitude != 0).astype(int).diff().abs().sum()`. Although this switch count does not precisely measure transition rates, it provides a preliminary view of signal density across bins. The resulting average switch rates per 1 K rows, shown in Table 5.1, help guide block size selection by t_{bin} . Lower switch rates at smaller t_{bin} values indicate sparser samples, justifying a denser sampling approach for these settings. Note that these are relatively simple measurements based on random sampled RTS so we don't expect to see any consistent trends. There could be more empty random samples in a certain t_{bin} than others, resulting lower average switch rate such as when $t_{\text{bin}} = 10 \mu\text{s}$ case.

t_{bin}	Average Switches [per 1 K rows]
100 μs	146.53
10 μs	37.10
1 μs	123.89
100 ns	89.79
50 ns	54.82
10 ns	13.31

Table 5.1: Average switching count per 1 K rows for RTS sample length of 300 K to 3 M rows used for Phase 2.

Overall, these observations reveal switch rates of approximately 10–150 per 1K rows, with generally lower rates at smaller t_{bin} values, supporting the need for finer segmentation of dense blocks. By increasing the number of sampled blocks at smaller t_{bin} values, we anticipate an improved sample validity rate to aid the RTS analysis. Given that the observed transitions from zero to non-zero will likely exceed actual counts of τ_{high} and τ_{low} transitions (due to the effects of denoising), the required sample length must be large enough to capture at least several thousand τ transitions for accurate statistical fits. To meet this threshold, we preliminarily set the block length at 100 K rows for the $t_{\text{bin}} = 1 \mu\text{s}$ configuration, as outlined in Table 5.2. This block size should yield the necessary density while remaining computationally manageable. Notably, the resolution limit of 25 ns, due to the detector's deadtime rating, precludes the analysis of any resolution finer than 25 ns, a constraint that Phase 2 did not fully account for.

t_{bin}	Sample Length	Sampled Percentage	Sample Duration
1 μs	100 K	0.03%	≈ 0.10 s

Table 5.2: RTS sampling specifications used for Phase 3 analysis.

Based on this configuration, approximately 50 RTS samples are to be analyzed per dataset, covering each of the 20 V_g and 4 V_l settings, totalling 4000 RTS samples. This approach aims to enhance the data density and enable more consistent tracking of trends within the collected data, as a foundation for future studies that may extend this framework to explore even higher-resolution datasets. Regarding switch event density, this setup results in an average of 148 switch events per 1 K rows across the 4000 RTS samples, an increase from the previous average of 124 as shown in Table 5.1 for the 1 μs bin. Although being an informal measurement, this increase of approximately 20% demonstrates an improvement in detecting switch events, supporting the enhanced data quality and consistency achieved by denser sampling.

5.3 Laser and Gate Voltages Dependency Towards Asymmetry

The primary method used to assess laser and gate voltage dependencies on asymmetry in Phase 3 is illustrated in Figure 5.1. In contrast to the random sampling approach in Phase 2 (Figure 4.3), this phase employs a forced dense sampling approach to improve data validity, especially for the focused t_{bin} of 1 μs .

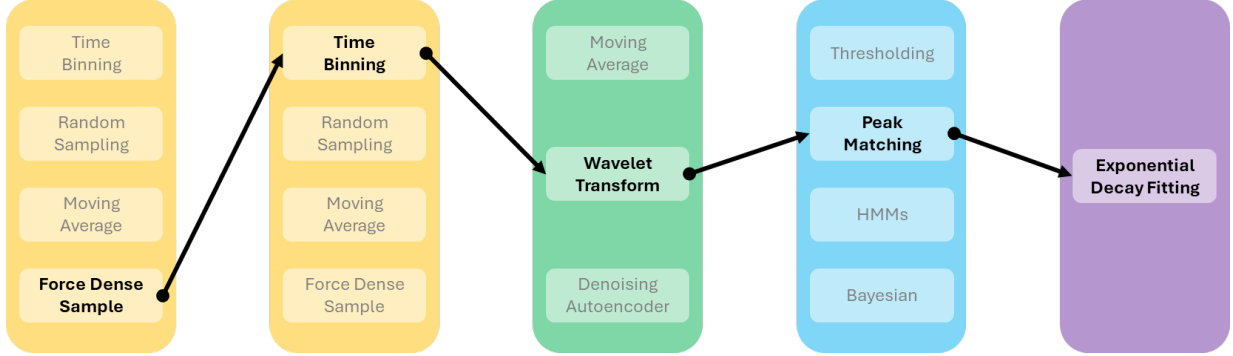


Figure 5.1: Method used in Phase 3 to study QD RTSs statistics in our framework. Changed from random sampling to force dense sample in contrast to Figure 4.3 in Phase 2.

In comparison to the sample validity rates from Phase 2, the Phase 3 results indicate a notable improvement in sample quality at $t_{\text{bin}} = 1 \mu\text{s}$, with validity rates increasing from 70% to 83% (Table 5.3). Here, a sample is only considered valid if both $\bar{\tau}_{\text{high}}$ and $\bar{\tau}_{\text{low}}$ can be accurately fitted, since the key metric under inspection—tunneling asymmetry—relies on both Γ_{in} and Γ_{out} data. The 83% validity rate in Phase 3, while an improvement, is influenced by several factors. One primary limitation is the dataset density across varying V_l and V_g settings. Since the extracted RTS samples cover a broad range of V_l and V_g combinations, some datasets inherently lack sufficient density, which reduces the likelihood of accurate fitting for both $\bar{\tau}_{\text{high}}$ and $\bar{\tau}_{\text{low}}$. Higher-density samples, often achievable at high V_l and low V_g settings, would likely yield validity rates closer to 100%. However, even in these cases, full validity is not guaranteed due to denoising effectiveness which can affect the consistency of tunneling rates.

t_{bin}	Sample Validity Rate
$1 \mu\text{s}$	70.000% in Phase 2
$1 \mu\text{s}$	82.975% in Phase 3

Table 5.3: Phase 3 RTS sample validity rate according to sampling specifications in Table 5.2, in contrast to Phase 2.

The scatter distributions in Figure 5.2 reveal a tighter alignment with the Fermi-Dirac (FD) fit at higher V_l , while lower V_l samples exhibit greater variation and less fidelity

in matching the FD distribution. This difference is inherited from lower density datasets at lower V_l , despite the sample quality improvements achieved through forced dense sampling, which enhances fit consistency. The improved validity rate and strengthened FD fits confirm that forced dense sampling allows for more accurate trend detection in asymmetry vs. V_g . This dense sampling method thereby enables us to explore how V_l influences the distribution's spread and narrowness, which may reveal dependencies between V_l and the asymmetry profile.

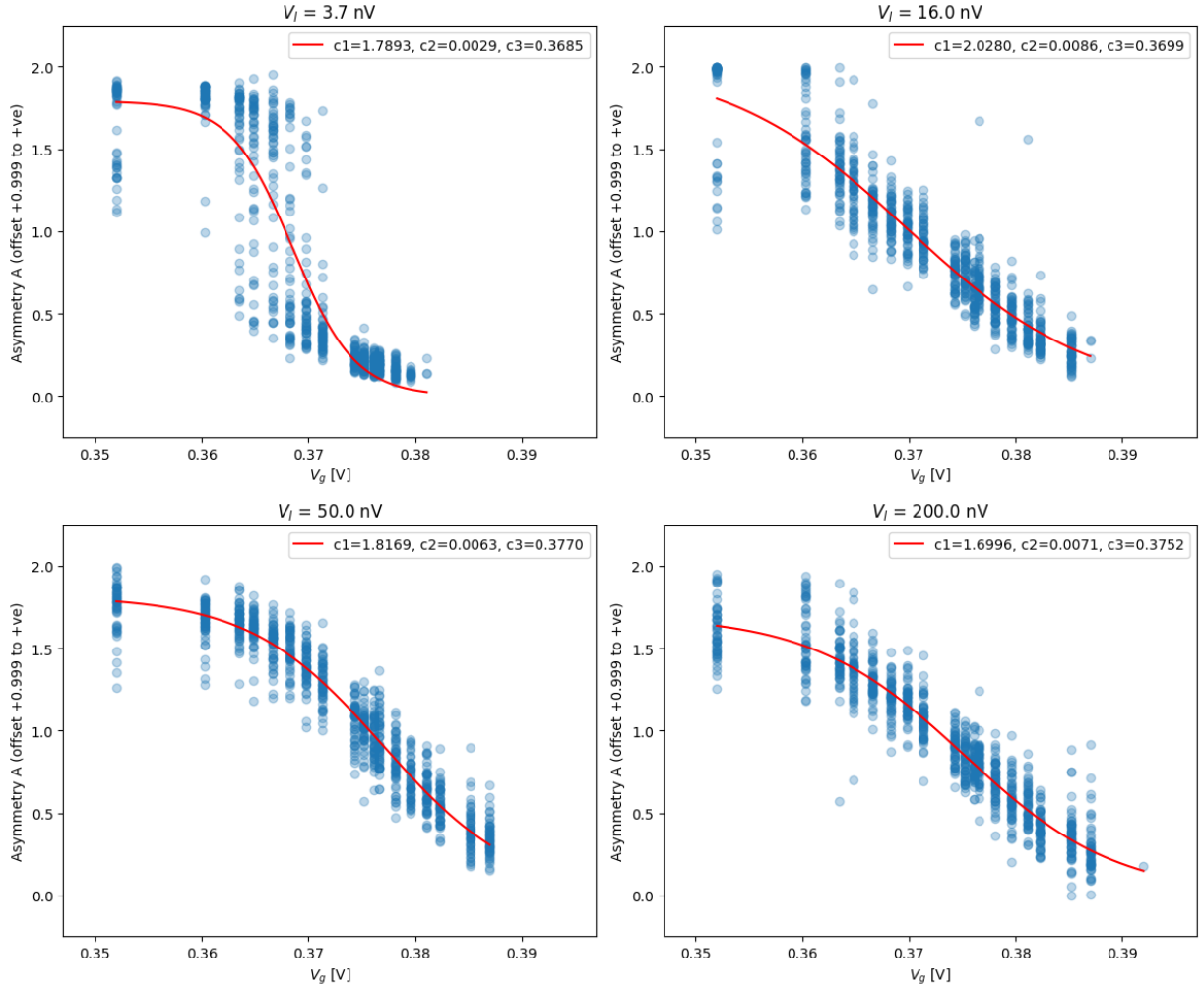


Figure 5.2: Asymmetry (offset to positives) vs V_g with fitted Fermi-Dirac (Equation 4.3) for all V_l at $t_{\text{bin}} = 1 \mu\text{s}$.

The refined plot in Figure 5.3, constructed from the (close to) 4,000 scatter points by consolidating each set of 50 data points with identical x-values (V_g) into box plots, provides a more statistically robust visualization. This approach improves clarity by displaying quartiles Q_0 through Q_4 , which illustrate the spread and central tendency of asymmetry values (A) for each gate voltage. The fitted Fermi-Dirac (FD) curve, based on the medians (Q_2) of the 20 unique gate voltages, captures the primary trend while minimizing the influence of outliers. The inclusion of an opaque 95% confidence interval, constructed from the standard deviations of fit parameters obtained via the covariance matrix, reflects the uncertainty in the fit with increased reliability. This is especially relevant for parameter c_2 , which remains close to ≈ 0.006 , supporting the results from Phase 2 with increased confidence.

When comparing Figure 5.3 with the Phase 2 results at the same t_{bin} (second row of Figure 4.10), the improved fit reveals more consistent trends across all V_g and V_l . This consolidated approach not only enhances the reliability of trends identified across datasets but also strengthens our confidence in the parameter estimates, particularly for c_2 , while providing a visual assessment of data variability across different V_g and V_l settings.

Although the parameters c_1 , c_2 , and c_3 in the Fermi-Dirac distribution formula (Equation 4.3) do not exhibit consistent trends across datasets, this variability aligns with the primary focus of Phase 3, which is to validate forced dense sampling, rather than to capture cross-time-bin variability. However, it is notable that the values of c_2 observed in Phase 3 remain in a similar range, around $c_2 \approx 0.006$, lending greater confidence to this parameter's stability and its potential to reflect an effective thermal signature in the QD system.

The current data constraints, particularly the limited parameter sampling across time bins, limit our capacity to draw definitive conclusions regarding trends in c_1 , c_2 , and c_3 . Nonetheless, as the methodology is applied across more diverse t_{bin} settings, we anticipate that more consistent trends in c_1 , c_2 , and c_3 will emerge, providing deeper insights into the physical mechanisms governing the QD behavior.

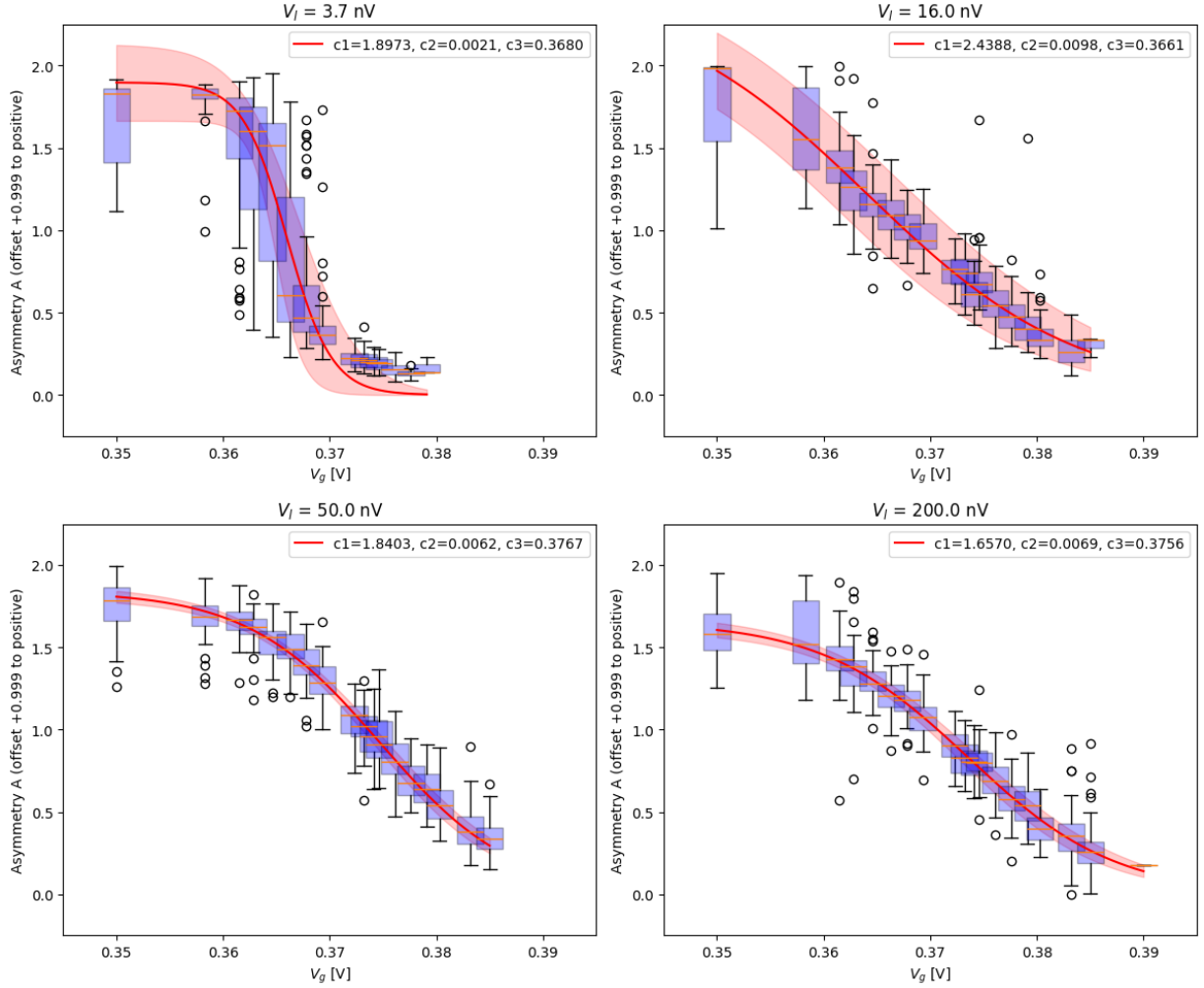


Figure 5.3: Box Plot of Asymmetry (offset to Positive) vs V_g with Fermi-Dirac Fit on median Asymmetry values for each V_l at Time Bin = 1 μ s, showing 95% confidence interval.

5.4 Summary

The Phase 3 analysis demonstrated that the forced dense sampling method reliably improves sample validity rates, yielding higher consistency in fitted results. This approach proves valuable as a foundation for systematically studying higher resolution dynamics in future work, with preliminary insights available in [Sec. 6.5](#).

Chapter 6

Unified Algorithm and Future Work

6.1 Overview

The objective is to gradually improve the so-far-developed framework that integrates algorithms tailored to various semiconductor types. The studies conducted throughout this thesis systematically contributed to a foundational framework that demonstrates how specific algorithms can enhance analysis across semiconductor [RTS](#) data. However, while [DAEUBL](#) has proven useful in several applications, especially for [RTSs](#) with pink noise [2, 7], it is not an optimal choice for all data types due to its high time complexity and other limitations. Therefore, the aim is to refine and specialize methods that can adapt efficiently to specific data characteristics, enabling the framework to automatically determine and deploy the most suitable techniques based on the data provided.

6.2 Unique Identifiers of Different Data

It is insufficient to classify data solely as originating from [QDs](#), transistors, or synthetic sources. Instead, the framework should be designed to first determine if the data aligns with well-characterized types such as [QDs](#) or transistors. For data that does not clearly align with these predefined types, the framework should then delve into further categorization, evaluating distinct attributes like the predominance of white or pink noise and differentiating between high and low noise intensity. This enables more precise classification, which in turn guides the selection of specific analysis techniques and parameters.

Ultimately, this framework contributes toward a broader goal—what could be described as a "unification" effort, as reflected in the proposed phase map outlined in [Sec. 6.3](#). This phase map, developed as part of future work, aims to guide researchers in matching data complexity with appropriate analysis methods. By systematically mapping noise types and levels to processing techniques, the phase map will serve as a comprehensive reference that enables the framework to autonomously determine the best methods and parameters for any given dataset. This modular, adaptable structure will ensure that even unfamiliar data types can be effectively analyzed, promoting a flexible approach across semiconductor research.

6.2.1 Quantum Dot

Before all, it is worth noting that not all [QD RTSs](#) exhibit only white noise backgrounds, and multi-level [RTSs](#) can also occur in [QDs](#). The simple two-level [RTS](#) with a white noise background is characteristic of the specific [QD](#) sample studied here, but not necessarily universal across all [QD](#).

To accurately identify a specific [RTS](#) from a [QD](#), it is first necessary to determine whether the observed [RTS](#) reflects a simple two-level signal predominantly influenced by white noise. This is particularly relevant for [QD](#) types such as the [QD](#) studied in this thesis, which is primarily characterized by these features. Verification can be achieved by applying a [KDE](#) analysis on raw data before any denoising steps, allowing an assessment of the dataset's noise characteristics to confirm alignment with typical behavior for this specific [QD](#) type. Moreover, because the underlying physics remains consistent within a given dataset, selecting a short time interval and calculating the average detector count can provide an initial estimation of the optimal t_{bin} . This estimation then supports fine-tuning denoising parameters specifically for the [RTS](#) in question, enhancing the accuracy of subsequent analysis.

6.2.2 Transistor

A preliminary examination of some specific transistor data we have available reveals distinct characteristics compared to [QD](#) data. Notably, the time resolution is considerably lower, reflecting different physical processes and timescales within the transistor samples. Other than that, it is more common to see multi-level [RTS](#) from transistors, unlike [QD](#) that are fixed to 2-level.

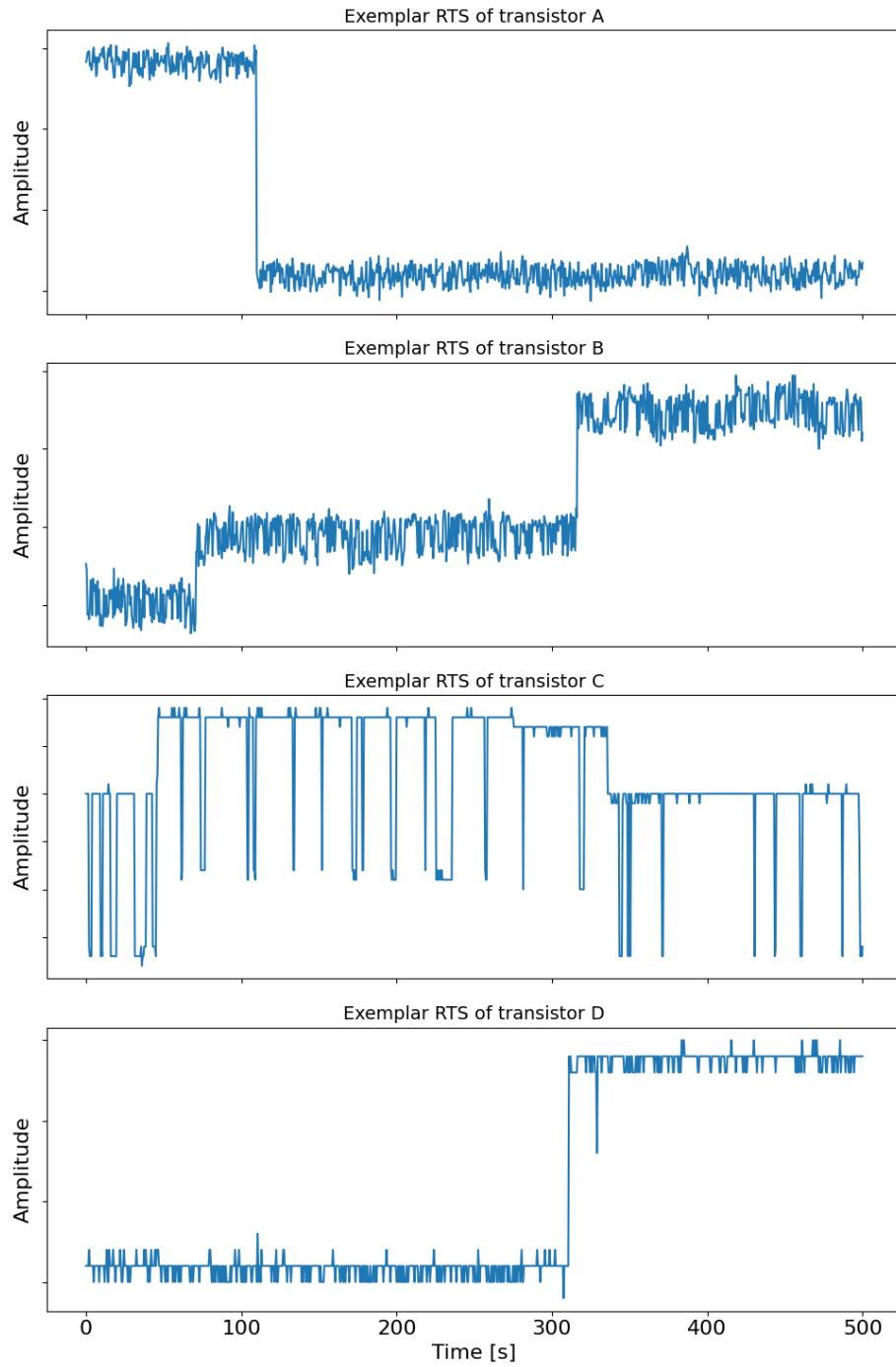


Figure 6.1: Raw (noisy) **RTS** samples from various available transistors.

Observations from the initial analysis include:

- **Transistor A:** 2-level (1-trap) [RTS](#) with minimal white noise
- **Transistor B:** 3-level (2-trap) [RTS](#) with moderate white noise
- **Transistor C:** 4-level (2-trap) [RTS](#) with barely any noise
- **Transistor D:** Either 2-level (1-trap) [RTS](#) or multi-level (multi-trap) [RTS](#) which consultant to the backgrounds physics is required.

Upon completing further [QD](#) and transistor studies, the framework should be advanced enough to differentiate between [QD](#) and transistor-generated [RTS](#). This distinction is required to enable targeted analysis and improved data handling based on the unique physical properties of each device.

6.2.3 Other Semiconductors Holding Complex RTS Data

Expanding the framework to handle more complex [RTS](#) data, including multi-level dynamics and data with significant pink noise components could provide additional insights and enhance its robustness. Access to such datasets, especially those with more complex noise structures, would allow the framework to be refined across a wider range of noise and signal behaviors. After thoroughly studying various real-world semiconductor data types, the existing framework could be adapted to evaluate synthetic data more effectively (if we treat it as a non-labeled type of real data). However, the emphasis will remain on real data analysis, as the primary goal is to develop a framework optimized for actual semiconductor behaviors rather than synthetic scenarios.

6.3 Phase Map

In linking the unique identifiers of different semiconductor [RTS](#) data, we can establish a comprehensive phase map that enables the framework to autonomously select optimal processing methods across stages without needing explicit information about the signal type. This phase map approach is particularly valuable for handling new semiconductor data for which detailed background physics may be unavailable. Through such an automated, adaptable process, the framework can analyze diverse data types with minimal intervention, relying instead on data characteristics to guide the choice of algorithms.

Automated classification of data through statistics-based evaluations or labeling assessments (such as KDE on raw RTS) can serve as a preliminary step in identifying the best processing method. This classification step must be computationally feasible and efficient only if its time cost is significantly lower than the formal analysis time. The phase map would cover various scenarios where different processing methods excel, optimizing for both efficiency and effectiveness depending on the noise levels and signal properties:

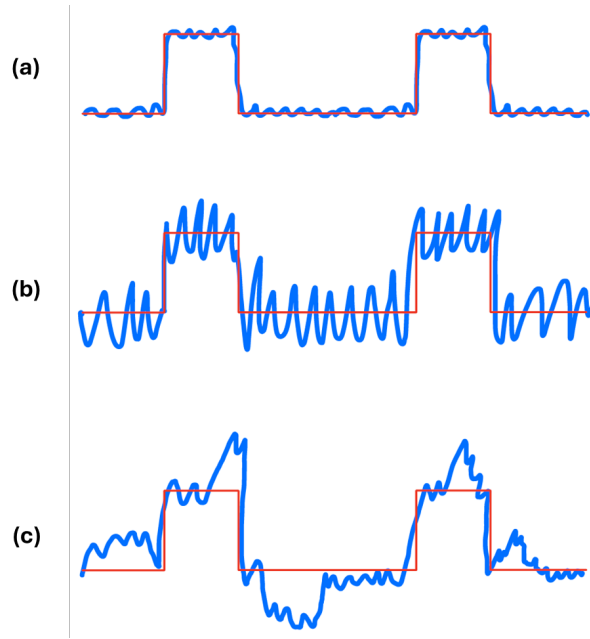


Figure 6.2: A simple, conceptual phase map by categorizing RTSs based on white and pink noise intensities.

- **Low White Noise, Low Pink Noise Figure 6.2 (a):** Optimal for certain standard denoising methods that avoid over-processing clean signals. Possible to jump to the digitization stage directly.
- **High White Noise, Low Pink Noise Figure 6.2 (b):** Requires algorithms tailored to maintain high sensitivity while filtering out substantial white noise. Ideal methods include DTCWT denoising.
- **High Pink Noise Figure 6.2 (c):** Demands robust filtering strategies like DAEUBL to handle the low-frequency components associated with pink noise.

The phase map, therefore, should support both automated parameter adjustment and method selection based on initial data labeling. For known data types, the map can retrieve relevant pre-set labels to use the best method for such semiconductors (Figure 4.3 for QD); for unknown data types, it uses general noise characteristics to approximate optimal parameters and choose best-suited variants. The utility of this phase map lies in facilitating the effective study of both existing and new semiconductor materials. Ultimately, as our understanding of these various device types grows and we collect sufficient real-world examples, we can retrain a machine learning model capable of recognizing, categorizing, and optimizing the analysis of all forms of RTS data. This fully autonomous model would integrate phase map principles with learned patterns to provide accurate and efficient processing across semiconductor applications. However, significant optimizations must be implemented to overcome current DAEUBL and any other methods’ performance limitations.

6.4 Algorithm Optimization

Algorithm optimization within the framework has significant potential for improvement, especially for handling extensive and high-resolution RTS data such as those from QD systems. Currently, the DAEUBL method has a maximum row size limit that is suboptimal for long-duration, high-resolution datasets. For example, processing raw data with $t_{\text{bin}} = 81$ ps over a 300-second interval involves approximately 3.8T rows. While this resolution exceeds practical limits due to timing jitter and dead time considerations (as discussed in Sec. 1.3.4), being able to achieve analysis at this theoretical resolution would be useful for evaluating and pushing the computational capabilities of the framework. In the scope of high-performance computing (HPC), we can leverage Tensor and CUDA cores on NVIDIA GPUs to significantly boost computational efficiency and address limitations in processing power and memory capacity. Using NVIDIA Tesla A100 GPU as an example:

Tensor Cores provide specialized hardware for tensor and matrix operations. Each of the 6912 Tensor Core performs 64 mixed-precision computations per GPU clock cycle (1.4 GHz at peak performance), typically using FP16/FP32 precision levels, where input matrices are processed in FP16 but accumulate in FP32 (e.g., $x+ = y \times z$). This is especially advantageous for tasks like DAEUBL inference, which inherently involve matrix operations that map well onto the Tensor Core’s structure. Tensor Cores are particularly efficient for operations typically encountered in machine learning tasks, enabling high-throughput computations that are necessary for handling extensive RTS data.

CUDA Cores are the fundamental building blocks of NVIDIA GPUs. They are specifically designed to execute parallel computing tasks, making them suitable for a wide range of applications, including graphics rendering, simulations, and scientific computations. 432 CUDA cores in A100 can handle thousands of threads simultaneously due to their parallel architecture. This allows them to perform multiple operations at once, leveraging data parallelism effectively, especially suited for **DTCWT** decompositions in parallel. In contrast to **DAEUBL**, **DTCWT** rely heavily on Fourier transforms and filtering, making the algorithm less compatible with Tensor Core optimizations. GPU-based Fast Fourier Transform (FFT) calculations are typically memory-bound, which limits the potential acceleration from Tensor Core matrix operations alone. To address this, we can implement CUDA FFT Device Extensions (cuFFTDx), which allow custom kernels to be fused with FFT operations, thus eliminating memory bottlenecks by keeping data within CUDA cores throughout the computation. This approach maximizes CUDA core utilization and reduces memory access latency, providing substantial speed gains for the **DTCWT** algorithm on GPUs.

For computationally intensive tasks across the framework, an effective optimization strategy requires first assessing if a process can be mapped onto matrix computations. If so, leveraging Tensor Cores is generally the most efficient path. For non-matrix tasks or processes bound by parallel computation, CUDA cores offer an adaptable solution to facilitate high-speed operations.

6.5 High-Resolution Temporal Dynamics

Exploring high-resolution temporal dynamics will be a key area of focus in future studies, particularly by varying the time resolution (t_{bin}) across datasets, and attempts to push to the highest resolution possible, namely close to the detector limit, dead time of 25 ns. This enables us to probe the dynamic behaviors more precisely, using the latest tools developed in the current framework to examine specific regions with dense sampling. For example, by selecting the dataset with the highest V_l and lowest V_g that have the highest counts per second, we can study sample blocks at different t_{bin} settings at the best possible chance.

Here we introduce a slight twist of the force-dense sample method used in Phase 3, see Appendix A. The idea is to not just pick the best 1 block across the whole dataset, but rank all blocks and pick multiple of the top ones, which not only ensures the captured region

is sufficiently dense but also has a longer overall duration. By referencing Table 5.1, the strategy adjusts the number of blocks at smaller t_{bin} settings and decreases the number of blocks as t_{bin} increases, compensating for duration differences. While an ideal setup would hold the sample duration constant across all t_{bin} values, this approach would generate data lengths of tens of millions of rows for high-resolution RTS, resulting in prohibitively high I/O times. Additionally, we observed that a 3M-row sample suffices for reliable τ statistics extraction in previous sections, rendering excessively large datasets unnecessary.

The optimized sampling specifications are summarized in Table 6.1. In this setup, the sample length reflects the number of data blocks included at each t_{bin} level, which allows for different sample durations without disproportionately long processing times. For instance, at a t_{bin} of 50 ns, the strategy involves combining the 20 densest blocks into a single RTS sample, which approximates 0.020 seconds in duration.

t_{bin}	Sample Length	Sampled Percentage	Sample Duration
10 μs	100K $\times$ 1 = 100K	0.333%	$\approx$ 1.000 s
1 μs	50K $\times$ 5 = 250K	0.083%	$\approx$ 0.250 s
100 ns	25K $\times$ 10 = 250K	0.008%	$\approx$ 0.025 s
50 ns	20k $\times$ 20 = 400K	0.007%	$\approx$ 0.020 s

Table 6.1: RTS sampling specifications used for high resolution analysis from dataset with $V_g = 0.3520$ V, $V_l = 200$ nV.

To mitigate issues related to the segmentation and merging of blocks, we maintain awareness of potential discrepancies at block junctions, where step transitions may misrepresent τ values. For instance, at a 50 ns t_{bin} with 20 blocks totaling 400 K rows, there are 19 block joints, potentially leading to a maximum of 38 transition errors. Given that there are approximately 220 K transitions across 400 K rows (based on estimates from Table 5.1 where $400 \text{ K} / 1 \text{ K} \times 55 \approx 220 \text{ K}$), this 38:220,000 ratio indicates a negligible impact on the τ statistics. Additionally, transitions from zero to non-zero states will be more frequent than true τ transitions, further reducing the ratio, i.e., error introduced from block junctions.

Each dataset will yield one sample per t_{bin} , totaling four RTS samples, with 2 in μs resolutions, and 2 in ns resolutions. Instead of focusing on numerical precision with a large number of samples like Phase 3, the primary goal here is to evaluate the insights that

high-density samples offer, particularly by observing trends and patterns that emerge at higher temporal resolutions. Notably, considering the detector’s 25 ns dead time, there is no practical need to increase resolution beyond 50 ns.

The selection of dense block sizes has been intentionally crafted to avoid capturing empty regions, which could potentially distort statistical results. Specifically, the block sizes for the first two sample sets at lower resolutions ($t_{\text{bin}} = 10 \mu\text{s}$ and $1 \mu\text{s}$) are chosen to ensure they are broad enough to capture numerous τ_{high} and τ_{low} features. Ensuring to reflect the statistics of the overall $\bar{\tau}_{\text{high}}$ and $\bar{\tau}_{\text{low}}$, from which tunneling rates Γ_{in} and Γ_{out} can be accurately derived. In contrast, the 2 higher-resolution samples ($t_{\text{bin}} = 100 \text{ ns}$ and 50 ns) cover shorter durations, with block sizes smaller than the typical τ durations (generally around 1 ms for kHz tunneling rates demonstrated from previous phases of analysis). For instance, the 100 ns sample block has a total duration of 2.5 ms, allowing for more precise capture of individual events within a single τ_{high} in an attempt to capture high-resolution tunneling dynamics.

This block design essentially divides the four samples into two functional groups. The $10 \mu\text{s}$ and $1 \mu\text{s}$ RTS samples effectively represent the conventional τ_{high} and τ_{low} states commonly observed in previous studies, while the higher-resolution samples offer a finer-grained temporal view, isolating more rapid, detailed fluctuations within the tunneling dynamics. This categorization hopefully provides some preparation work until the formal analysis to scale across temporal dynamics.

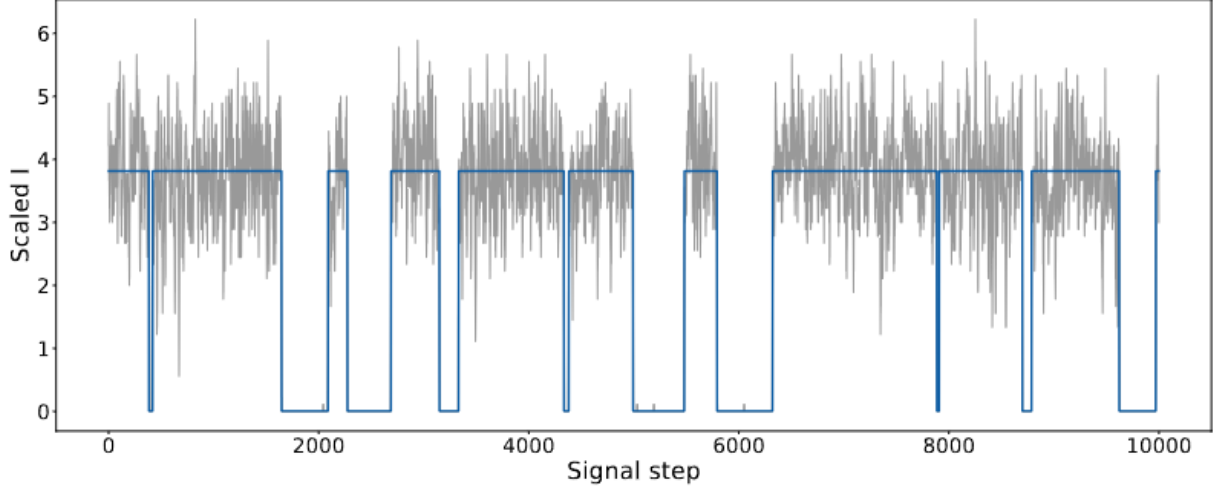


Figure 6.3: Digitization of $t_{\text{bin}} = 1 \mu\text{s}$ RTS sample specified in Table 6.1, only 10000 steps are displayed for visualization.

In Figure 6.3, we present a sample of digitized data for the $1 \mu\text{s}$ RTS signal, as detailed in Table 6.1. The computed Γ_{in} and Γ_{out} for two resolutions are as follows. These values, derived from a single sample for each specification, show that Γ_{in} remain within approximately 4% of one another, and Γ_{out} about 50% within another. This consistency across resolutions, even within a single sample, indicates an acceptable estimation prior to a more systematic study.

- $t_{\text{bin}} = 10 \mu\text{s}$: $\Gamma_{\text{in}} \approx 1288 \text{ [Hz]}$, $\Gamma_{\text{out}} \approx 3388 \text{ [Hz]}$
- $t_{\text{bin}} = 1 \mu\text{s}$: $\Gamma_{\text{in}} \approx 1332 \text{ [Hz]}$, $\Gamma_{\text{out}} \approx 4938 \text{ [Hz]}$

In the $1 \mu\text{s}$ resolution depicted in Figure 6.3, τ_{low} intervals appear as fully zero regions, likely either representing deadtimes or periods without actual electron transition events. With forced-dense sampling in high-resolution cases (such as $t_{\text{bin}} = 50 \text{ ns}$), the RTS almost entirely derive from periods initially identified as τ_{high} in lower-resolution samples. Consequently, these ns resolution samples reflect finer transitions within the τ_{high} states, where smaller-scale dynamics may be evident.

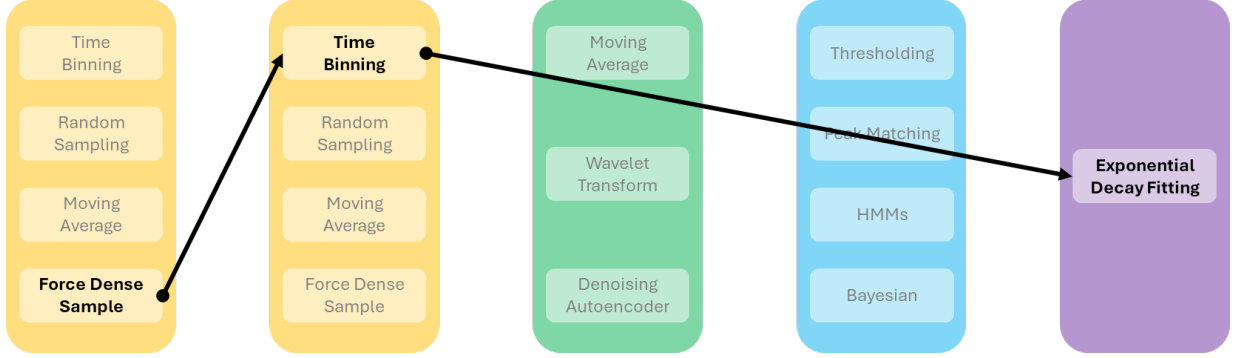
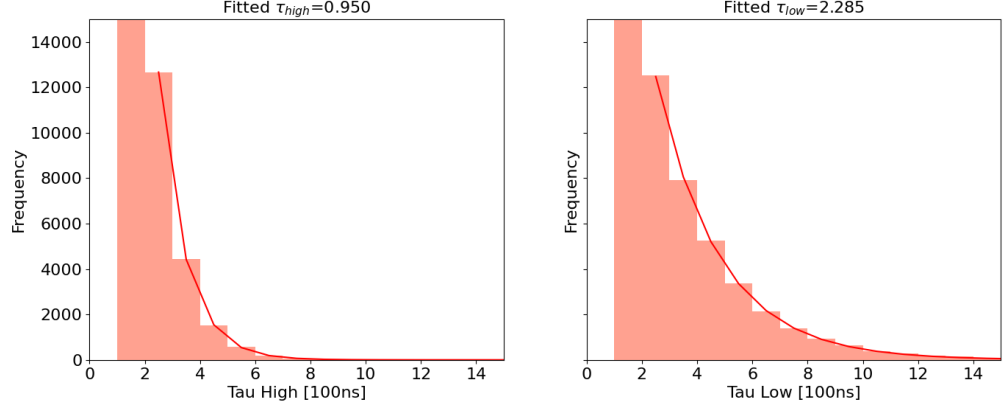


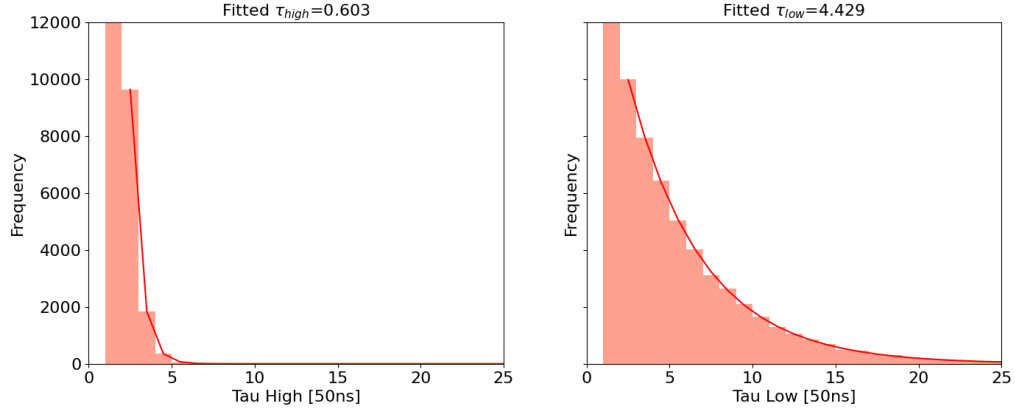
Figure 6.4: Method used in initial look towards high-resolution QD RTSs statistics in our framework.

In this context, any observed τ_{low} at the ns level resolution should not be conflated with τ_{high} deadtimes seen at μs resolutions. Instead, each dense block in the ns resolution RTS is a zoomed-in view of one or more τ_{high} regions, capturing its internal transitions into smaller τ_{high} and τ_{low} fluctuations. This approach provides a more close-up look at the tunneling dynamics by focusing within the broader τ_{high} states, allowing us to observe underlying patterns at ns resolutions not apparent in μs sampling. The high-resolution RTS samples at ns resolutions are effectively binary (1s and 0s) without significant deadtime regions. This inherently digitized format allows for the straightforward fitting of τ statistics, as illustrated by the chain in Figure 6.4. Just like previous exponential fittings, we exclude the lowest bin to minimize noise interference in τ statistics fitting given in Figure 6.5. The derived Γ_{in} and Γ_{out} values for each resolution reveal patterns unique to these ns scale resolutions:

- $t_{\text{bin}} = 100 \text{ ns}$: $\Gamma_{\text{in}} = 10 \text{ [MHz]}$, $\Gamma_{\text{out}} = 4.4 \text{ [MHz]}$
- $t_{\text{bin}} = 50 \text{ ns}$: $\Gamma_{\text{in}} = 33 \text{ [MHz]}$, $\Gamma_{\text{out}} = 4.5 \text{ [MHz]}$



(a). 100 ns



(b). 50 ns

Figure 6.5: Tau statistics fitting of QD RTS samples from dataset with $V_g = 0.3520$ V, $V_l = 200$ nV, and $t_{\text{bin}} = 100$ ns and 50 ns.

It is essential to note that these tunneling rates are distinct from the physical rates of the QD device; they represent dynamics specifically within the τ_{high} states given the zoomed-in focus at nanosecond resolutions. The short durations of τ_{high} events affect the quality of $\bar{\tau}_{\text{high}}$, which is likely less stable compared to $\bar{\tau}_{\text{low}}$ as indicated in Figure 6.5. Because the reduced density of detector counts (inherently at such high resolutions even at the densest part of the dataset), τ_{high} events act as separators for the τ_{low} intervals, the Γ_{out} values are more consistent across resolutions and are potentially more reliable.

These findings hint at some complex dynamics in high-resolutions, a pattern that may only be visible at extended ns scale observations. Further work could explore if deviations appeared from the drastic change, i.e., 3 orders of magnitudes of difference for τ that reveal alternative physical mechanisms or more intricate dynamics within the QD device's tunneling process, potentially uncovering previously unseen sub-structures in τ behavior.

6.6 Applications

6.6.1 Applications in CMOS and QD Domains

Along with the studies presented in this thesis and the aforementioned plan, the robustness and analytical capabilities of the framework for processing RTS data make it a valuable tool across a range of real-time applications. With its significantly improved performance, it may allow for real-time analysis and immediate feedback on semiconductor performance. This can be particularly useful in cases where fine-tuning device characteristics, such as voltage adjustments, is essential to achieving the desired operational parameters of semiconductors.

- CMOS and QD Devices:** The framework developed in this thesis should support dynamic voltage adjustments in QD devices, allowing real-time monitoring of tunneling statistics such as τ_{high} , τ_{low} , Γ_{in} , and Γ_{out} . This immediate feedback loop facilitates optimized device performance by achieving desired electron tunneling rates or specified frequencies, which is highly beneficial for tuning targeted electronic and optical responses. Additionally, the framework achieved a significant stepping stone of RTS noise studies in complementary metal-oxide-semiconductor (CMOS) and QD domains, a critical factor for ensuring stability, reliability, and performance in semiconductor applications. Capable of handling high-resolution random telegraph noise (RTN) analysis, this framework may even offer real-time monitoring and feedback in fields like field-programmable gate array (FPGA) broadband noise analysis and noise filtering.
- Multifunction Quantum Devices:** This research also has broad implications for multifunction quantum devices, where RTN plays a crucial role in setting electron tunneling rates, coherence times, and overall system stability. By advancing noise measurement and reduction capabilities, the tools and methods developed here should support improved control over charge noise, which is essential for high-performance applications in quantum communication and computation.

6.6.2 Directions in Quantum Science and Technology

The framework and findings in this thesis also demonstrated promising developments toward a valuable tool for quantum science and technology. In the ongoing development of quantum devices and systems, noise reduction and precise real-time monitoring are essential steps toward the realization of practical quantum computing technologies.

- **Support for Theoretical and Experimental Quantum Research:** Through future work, by providing a powerful tool for real-time noise analysis in quantum devices, this research hopefully enables experimental physicists to conduct more accurate studies of quantum phenomena, improving our theoretical understanding of solid-state quantum systems. The methodologies for precise measurement and categorization of tunneling rates and electron transition behaviors, as developed here, could provide insights into phenomena that are critical for advancing research on quantum systems like QD spin qubits and semiconductor-based quantum devices. Additionally, recent studies show that RTS also appears in superconducting qubits, underscoring the need for comprehensive noise-handling tools across various quantum platforms [10, 11, 29].
- **Toward Full-Stack Solid-State Quantum Platforms:** As solid-state quantum technology progresses, the ability to mitigate and adapt to noise in real time will become a fundamental requirement for scaling up quantum systems into full-stack platforms. This thesis supports these developments by contributing a framework that is both adaptable and scalable to the increasing complexity and demands of next-generation solid-state quantum systems. The results and methodologies presented here and future work in the next phases hopefully serve as a stepping stone towards the development of integrated, full-stack solid-state quantum platforms that can operate stably and reliably in complex, noisy environments.
- **Applying Quantum Algorithms to RTS Analysis:** Quantum algorithms offer exciting potential for advancing RTS analysis. For instance, quantum denoising autoencoders could be used to reduce noise in RTS data effectively. Initial work has explored quantum resources like NVIDIA’s cuQuantum SDK [30], enabling more efficient computations, with preliminary studies showing the feasibility of quantum-assisted neural networks [31]. Applying such quantum algorithms in quantum computing could assist in noise analysis and quantum device diagnostics.

6.7 Conclusions

This work presents a framework that could evolve into a high-speed, adaptable solution for precise and efficient semiconductor tuning, potentially benefiting both experimental and commercial applications. A core advancement achieved here is the enhanced resolution—refined from a t_{bin} of $2\ \mu\text{s}$ down to 100 ns, achieving a 20-fold improvement on the tested QD. Further explorations reach a resolution of 50 ns, opening new avenues for studying high-resolution dynamics and uncovering critical trends hidden by background noise.

Beyond speed and resolution, the framework is capable of examining information concealed under background noise, assisting the characterization of RTS behaviors. This enhanced sensitivity not only contributes directly to CMOS and QD device performance analysis but also establishes a foundational step toward scalable solid-state quantum systems. By supporting real-time analysis and adaptive noise mitigation, this thesis builds toward the development of resilient, multifunctional, and integrated quantum platforms crucial for advancing the broader field of quantum technology.

References

- [1] J. Kerski, H. Mannel, P. Lochner, E. Kleinherbers, A. Kurzmann, A. Ludwig, A. D. Wieck, J. König, A. Lorke, and M. Geller. Post-processing of real-time quantum event measurements for an optimal bandwidth. *Scientific Reports*, 13:1105, 2023. Faculty of Physics and CENIDE, University of Duisburg-Essen. 2nd Institute of Physics, RWTH Aachen University. Chair for Applied Solid State Physics, Ruhr-Universität Bochum.
- [2] Bowen Deng, HeeBong Yang, and Na Young Kim. A denoising autoencoder based on u-net and bidirectional long short-term memory for multi-level random telegraph signal analysis. Master’s thesis, University of Waterloo, 200 University Ave. West, Waterloo, N2L 3G1, Ontario, Canada, 2024. Department of Electrical and Computer Engineering, Institute for Quantum Computing, Waterloo Institute for Nanotechnology, and Department of Chemistry.
- [3] D. M. Fleetwood. *Origins of 1/f Noise in Electronic Materials and Devices: A Historical Perspective*, pages 1–31. Springer International Publishing, Cham, 2020.
- [4] Roberto da Silva and Gilson Wirth. RTS noise in semiconductor devices: time constants estimates and observation window analysis. *Journal of Statistical Mechanics: Theory and Experiment*, 2022(4):043201, apr 2022.
- [5] Nuditha Vibhavie Amarasinghe, Zeynep Çelik Butler, Anna Zlotnicka, and Fang Wang. Model for random telegraph signals in sub-micron mosfets. *Solid-State Electronics*, 47(9):1443–1449, 2003.
- [6] S Guo, R Wang, and D et al. Mao. Anomalous random telegraph noise in nanoscale transistors as direct evidence of two metastable states of oxide traps. *Scientific Reports*, 7(1):6239, 2017.
- [7] Marcel Robitaille, HeeBong Yang, Lu Wang, Bowen Deng, and Na Young Kim. Deep neural network analysis models for complex random telegraph signals. *Scientific Re-*

ports, 2024. Institute for Quantum Computing, Department of Electrical and Computer Engineering, Waterloo Institute for Nanotechnology, Department of Chemistry, and Perimeter Institute, University of Waterloo.

- [8] K.K. Hung, P.K. Ko, C. Hu, and Y.C. Cheng. Random telegraph noise of deep-submicrometer mosfets. *IEEE Electron Device Letters*, 11(2):90–92, Feb 1990.
- [9] Xinyang Wang, Padmakumar R. Rao, Adri Mierop, and Albert J.P. Theuwissen. Random telegraph signal in CMOS image sensor pixels. In *2006 International Electron Devices Meeting*, pages 1–4, Dec 2006.
- [10] M. D. Shaw, R. M. Lutchyn, P. Delsing, and P. M. Echternach. Kinetics of nonequilibrium quasiparticle tunneling in superconducting charge qubits. *Phys. Rev. B*, 78:024503, Jul 2008.
- [11] N. J. Lambert, A. A. Esmail, M. Edwards, A. J. Ferguson, and H. G. L. Schwefel. Random telegraph signal analysis with a recurrent neural network. *Phys. Rev. E*, 102:012312, Jul 2020.
- [12] A Theodorsen, OE Garcia, and M Rypdal. Statistical properties of a filtered poisson process with additive random noise: distributions, correlations and moment estimation. *Physica Scripta*, 92(5):054002, 2017.
- [13] Eric Kleinherbers, Hendrik Mannel, Jens Kerski, Martin Geller, Axel Lorke, and Jürgen König. Unraveling spin dynamics from charge fluctuations. *Physical Review Research*, 5(4):043103, 2023.
- [14] A. Kurzmann, P. Stegmann, J. Kerski, R. Schott, A. Ludwig, A. D. Wieck, J. König, A. Lorke, and M. Geller. Optical detection of single-electron tunneling into a semiconductor quantum dot. *Physical Review Letters*, 122(24):247403, 2019.
- [15] Miro Kroutvar, Yann Ducommun, Dominik Heiss, Max Bichler, Dieter Schuh, Gerhard Abstreiter, and Jonathan J. Finley. Optically programmable electron spin memory using semiconductor quantum dots. *Nature*, 432(7013):81–84, 2004.
- [16] S. H. Mortazavi and S. M. Shahrtash. Comparing denoising performance of dwt,wpt, swt and dt-cwt for partial discharge signals. In *2008 43rd International Universities Power Engineering Conference*, pages 1–6, 2008.
- [17] Nick Kingsbury. Dual-tree complex wavelets: Part 1. Talk presentation, PDF format, February 2005. Accessed: 2024-05.

- [18] MathWorks. *Dual-Tree Complex Wavelet Transforms*. The MathWorks, Inc., 2024. Available at <https://www.mathworks.com/help/wavelet/ug/dual-tree-complex-wavelet-transforms.html>.
- [19] Rich Wareham, Nick Kingsbury, and Cian Shaffrey. Dual-tree complex wavelet transform (dtcwt) python package, 2013–2014. © Copyright 2013, 2014, Rich Wareham, Nick Kingsbury, Cian Shaffrey. Revision f6953ff5.
- [20] Sean R. Eddy. What is a hidden markov model? *Nature Biotechnology*, 22(10):1315–1316, 2004.
- [21] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. 3rd edition, 2024. Online manuscript released August 20, 2024.
- [22] Alperen Değirmenci. Introduction to hidden markov models. 2015.
- [23] Zoubin Ghahramani and Michael Jordan. Factorial hidden markov models. volume 29, pages 245–273, 11 1997.
- [24] Hiromitsu AWANO, Hiroshi TSUTSUI, Hiroyuki OCHI, and Takashi SATO. Bayesian estimation of multi-trap rtn parameters using markov chain monte carlo method. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E95.A(12):2272–2283, 2012.
- [25] Xiaoming Chen, Lin Wang, Boxun Li, Yu Wang, Xin Li, Yongpan Liu, and Huazhong Yang. Modeling random telegraph noise as a randomness source and its application in true random number generation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35:1–1, 01 2015.
- [26] Ayush Kapoor. Ece699 final report, 2024. QuIN Lab, Department of Electrical and Computer Engineering, University of Waterloo.
- [27] Karl Sentz and Scott Ferson. Combination of evidence in dempster-shafer theory. Technical Report SAND2002-0835, Sandia National Laboratories, Albuquerque, New Mexico and Livermore, California, April 2002. Approved for public release; further dissemination unlimited. Prepared under Contract DE-AC04-94AL85000.
- [28] A. Kurzmann, P. Stegmann, J. Kerski, R. Schott, A. Ludwig, A. D. Wieck, J. König, A. Lorke, and M. Geller. Optical detection of single-electron transport dynamics. *Physical Review Letters*, 122(24):247403, 2019. Supplemental Information.

- [29] Łukasz Cywiński, Roman M. Lutchyn, Cody P. Nave, and S. Das Sarma. How to enhance dephasing time in superconducting qubits. *Phys. Rev. B*, 77:174509, May 2008.
- [30] Harun Bayraktar, Ali Charara, David Clark, Saul Cohen, Timothy Costa, Yao-Lung L. Fang, Yang Gao, Jack Guan, John Gunnels, Azzam Haidar, Andreas Hehn, Markus Hohnerbach, Matthew Jones, Tom Lubowe, Dmitry Lyakh, Shinya Morino, Paul Springer, Sam Stanwyck, Igor Terentyev, Satya Varadhan, Jonathan Wong, and Takuma Yamaguchi. cuquantum sdk: A high-performance library for accelerating quantum science, 2023.
- [31] Tonghe Bai and Yazi Chen. Unifying quantum circuits for versatile neural network architectures, 2024. ORCID: Tonghe Bai 0009-0003-7338-2986.
- [32] R. Paschotta. Avalanche photodiodes. RP Photonics Encyclopedia. Available online at https://www.rp-photonics.com/avalanche_photodiodes.html.

APPENDICES

Appendix A

Modified Force Dense Sample Method

A.1 Briefing

Before diving into the details of this algorithm, it is important to note that extracting data—that is, creating a [RTS](#) sample from the raw dataset is a time-consuming process, with complexity dependent on how the raw dataset is stored. We should always avoid binning the full dataset (where binning refers to aggregating data into intervals and writing the sum of detector counts as each row for an [RTS](#) sample). Instead, we should only bin the regions of interest, applicable to both random sampling and force dense sampling. For random sampling, the minimum number of passes (where each pass is an iteration through each line of the raw dataset) is 1, as the desired region can be identified beforehand, allowing us to bin only when needed. For force-dense sampling, the minimum number of passes is 2, as we need one additional pass before extraction to identify the relatively denser regions for extraction. Midway identification of dense regions is not possible because the entire dataset must be considered for density assessment. With these principles in mind, let us proceed with an example where a sample [RTS](#) is extracted at a 100 ns , where the top 10 blocks will be selected each at 25 K rows in length (see [Table 6.1](#)).

A.2 Algorithm

1. We segment the dataset into blocks of the desired size. For a 100 ns sample, each block would contain 25 K rows (0.0025 s of data), so a dataset of 300 s would have approximately 120 K blocks.
2. Next, we iterate through the dataset to obtain the sum of detector counts for each of the 120 K blocks. This initial pass is unavoidable.
3. We then rank the 120 K blocks by their counts, marking the 10 blocks (for a 100 ns sample) of interest with the highest counts.
4. We make a second pass through the dataset for the actual extraction, line by line. When we encounter one of the 10 marked blocks, which are not necessarily consecutive, we begin aggregating detector counts within each 100 ns (each row of the [RTS](#) sample). This approach is efficient, ensuring still only 1 pass is required for data extraction.
5. Upon finishing the extraction of one marked block, we estimate the number of bytes to skip to reach the start of the next marked block using `seek()`. This command allows us to skip a specified number of bytes without needing to read each byte. To ensure accuracy, this skip operation is conservative: it skips a portion of the estimated distance and then reads the next byte to check its position, adjusting and continuing as necessary. For example, by `read()` a line we find the next block is 8 M bytes away, we may first skip 4 M bytes, `read()` a line again then skip 2 M bytes, and so forth, until the correct position is reached. This significantly reduces the need for repeated `read()` operations, which are comparatively slower.

Quantitatively, for another timed [RTS](#) sample extraction, the extraction process was completed in 50 seconds using this skipping mechanism, with only 0.17% of the data being read. In comparison, without the skipping mechanism and reading line by line, the file reading process had processed less than 2% of the data before even reaching the first block (at least 50 times performance difference, note that the extraction at the dense blocks also takes time). This demonstrates that `seek()` requires substantially less time than `read()`, despite the significant reduction in the number of `seek()` calls.