ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ
ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ
ΜΑΘΗΜΑΤΙΚΩΝ
ΚΑΙ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΣΧΟΛΗ ΜΗΧΑΝΟΛΟΓΩΝ
ΜΗΧΑΝΙΚΩΝ

ΕΚΕΦΕ «ΔΗΜΟΚΡΙΤΟΣ»

ΙΝΣΤΙΤΟΥΤΟ ΝΑΝΟΕΠΙΣΤΗΜΗΣ
ΚΑΙ ΝΑΝΟΤΕΧΝΟΛΟΓΙΑΣ

ΙΝΣΤΙΤΟΥΤΟ ΠΥΡΗΝΙΚΗΣ
ΚΑΙ ΣΩΜΑΤΙΔΙΑΚΗΣ ΦΥΣΙΚΗΣ

Διατμηματικό Πρόγραμμα Μεταπτυχιακών Σπουδών
'Φυσική και Τεχνολογικές Εφαρμογές'

Interdepartmental Postgraduate Program
'Physics and Technological Applications'

Ανάλυση Χρονοσειρών του Στοχαστικού Θορύβου στον Αισθητήρα DECAL για τη Δημιουργία Πραγματικά Τυχαίων Αριθμών

Time Series Analysis of Stochastic Noise in the DECAL Sensor for the Generation of Truly Random Numbers

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ
του Δήμου Ασλάνη

MSc THESIS
Dimos Aslanis

Επιβλέπων: Καθηγητής Αλέξανδρος Κεχαγιάς

Supervisor: Professor Alexandros Kehagias

Αθήνα, Ιούλιος 2024

Acknowledgements

I would like to thank my supervisor, Professor Alexandros Kehagias, for assigning me this project and providing supervision. I am also thankful to Assistant Professor Ioannis Kopsalis and Dr. Ioannis Theodonis for their co-supervision and consistent availability throughout this year.

Special thanks are due to Professor Vlasis Mavrantzas and Dr. Loukas Peristeras for their patience and understanding during the completion of this thesis.

Moreover, I am profoundly grateful to my parents for their unwavering support of my decision to pursue an MSc in Physics, despite the much safer path in Chemical Engineering. Their encouragement has been invaluable, even when my choices seemed irrational. I also want to thank the people who I am extremely lucky to call friends, who have supported me throughout the past seven years of my undergraduate and postgraduate studies. Finally, my deepest appreciation goes to Katerina, without whom these past two years would have been significantly more challenging.

Abstract

This thesis explores the stochastic characteristics of noise in the Depleted Monolithic Active Pixel Sensor (DMAPS) Digital Electromagnetic Calorimeter (DECAL) sensor and its application as a True Random Number Generator (TRNG) through time series analysis. The study aims to (1) comprehensively analyze the noise properties of the DECAL sensor, (2) establish a methodology for generating random numbers from noisy signals using time series analysis, and (3) evaluate the feasibility of using the DECAL sensor as a TRNG. The sensor's output, recorded without any external stimulus, is analyzed using statistical and time series techniques and fitted into an Autoregressive Integrated Moving Average (ARIMA) model. Through this model, the sensor's noise is transformed into Gaussian white noise, which is then used to generate random bits. The generated random numbers and bits are evaluated using existing statistical tests, a novel diffusion test, and the USA National Institute of Standards and Technology (NIST) test suite. The results demonstrate that this method can be effectively applied to the DECAL sensor to produce random numbers with a high degree of randomness. However, the low rate of random bit generation presents a limitation, affecting both its testing and application as a TRNG. Possible solutions to this issue are discussed. The study concludes that time series analysis is a viable method for generating random numbers from the DECAL sensor and that, with further improvements, the sensor can serve as a TRNG.

Περίληψη

Η παρούσα εργασία διερευνά τα στοχαστικά χαρακτηριστικά του θορύβου του ψηφιακού ηλεκτρομαγνητικού αισθητήρα DECAL τεχνολογίας DMAPS και την εφαρμογή του ως γεννήτρια πραγματικά τυχαίων αριθμών (ΓΠΤΑ) μέσω ανάλυσης χρονοσειρών. Στόχος της μελέτης είναι (1) η συνολική ανάλυση των ιδιοτήτων θορύβου του αισθητήρα DECAL, (2) η δημιουργία μιας μεθοδολογίας για τη δημιουργία τυχαίων αριθμών από αναλογικούς θορύβους με τη χρήση ανάλυσης χρονοσειρών και (3) η αξιολόγηση του ενδεχομένου χρήσης του αισθητήρα DECAL ως ΓΠΤΑ. Ο θόρυβος του αισθητήρα αναλύεται με τη χρήση στατιστικών τεχνικών και τεχνικών χρονοσειρών και προσαρμόζεται σε ένα μοντέλο ARIMA (Autoregressive Integrated Moving Average). Μέσω αυτού του μοντέλου, ο θόρυβος του αισθητήρα μετατρέπεται σε Γκαουσιανό λευκό θόρυβο, ο οποίος στη συνέχεια χρησιμοποιείται για τη δημιουργία τυχαίων bits. Οι παραγόμενοι τυχαίοι αριθμοί και bits αξιολογούνται με τη χρήση στατιστικών δοκιμών, μιας δοκιμής διάχυσης και της ομάδας ελέγχου τυχαιότητας αριθμών του Εθνικού Ινστιτούτου Προτύπων και Τεχνολογίας των ΗΠΑ. Τα αποτελέσματα δείχνουν ότι η μέθοδος αυτή μπορεί να εφαρμοστεί στον αισθητήρα DECAL για την παραγωγή τυχαίων αριθμών με υψηλό βαθμό τυχαιότητας. Ωστόσο, ο χαμηλός ρυθμός παραγωγής τυχαίων bits αποτελεί περιορισμό, επηρεάζοντας τόσο τις δοκιμές όσο και την εφαρμογή του ως ΓΠΤΑ. Συζητούνται πιθανές λύσεις για το ζήτημα αυτό. Η μελέτη καταλήγει στο συμπέρασμα ότι η ανάλυση χρονοσειρών είναι μια έγκυρη μέθοδος για την παραγωγή τυχαίων αριθμών από τον αισθητήρα DECAL και ότι, με περαιτέρω βελτιώσεις, ο αισθητήρας μπορεί να χρησιμοποιηθεί ως ΓΠΤΑ.

Contents

List of Figures

List of Tables

Glossary

ACF Autocorrelation Function. 11, 12, 14–18, 27, 32–35
AIC Akaike Information Criterion. 25, 33, 34
AR Autoregressive. 13–16, 32, 33
ARIMA Autoregressive Integrated Moving Average. iii, 16, 25–27, 33–36, 39–41, 47, 48
ARMA Autoregressive Moving Average. 15, 16
ATLAS A Toroidal LHC Apparatus. 5

BIC Bayesian Information Criterion. 25, 33, 34
BMLE Binned Maximum Likelihood Estimation. 9, 24, 44

CDF Cumulative Distribution Function. 7
CERN European Organization for Nuclear Research. 5, 22
CMS Compact Muon Solenoid. 5
CSV Comma Separated Values. 23, 32, 39

DAC Digital-to-Analog Converter. 21, 22, 31
DECAL Digital Electromagnetic Calorimeter. iii, 4, 6, 18–21, 23, 26, 38, 41, 43, 44, 46
DMAPS Depleted Monolithic Active Pixel Sensor. iii, 20

ECAL Electromagnetic Calorimeter. 4, 5
ERNIE Electronic Random Number Indicator Equipment. 2, 3

FFT Fast Fourier Transform. 29
FPGA Field Programmable Gate Array. 22

HCAL Hadronic Calorimeter. 4
HEP High Energy Physics. 4, 7, 43

ITSDAQ ITk Strips Data Acquisition. 22, 39

LHC Large Hadron Collider. 5

MA Moving Average. 14–16, 32, 33
MAPS Monolithic Active Pixel Sensors. 20
MLE Maximum Likelihood Estimation. 8, 9, 25
MSD Mean Squared Displacement. 28, 29, 36, 37

NIST USA National Institute of Standards and Technology. iii, 2, 29, 37, 41

PACF Partial Autocorrelation Function. 12, 14–16, 32–35
PDF Probability Density Function. 7–9, 11
PMF Probability Mass Function. 7, 8
PRNG Pseudo-Random Number Generator. 1–3

RNG Random Number Generator. 1, 2

SNR Signal-to-Noise Ratio. 20

TRNG True Random Number Generator. iii, 1–3, 19, 38–41

Theoretical Background 1.

1.1. Random Number Generation

Introduction

Random number generation has always been important for humanity. In modern times, some main applications of random numbers are: simulations of natural phenomena [1, 2], mathematics [3], computer programming [4], and cryptography [5].[1]

Before computers were invented, true random numbers were generated using physical devices such as dice.[2] The first specialized Random Number Generator (RNG) was made in 1939 [7], and many followed, especially after the advent of electronic computers.[3] Until the widespread use of personal computers, these random numbers were mainly published in books and tables for scientists to use. However, the need for quick and easy access to random numbers led to the rapid development of two types of RNGs.

- *True Random Number Generators (TRNGs)*: Fast physical devices that generate random numbers using physical processes. These are mainly used in cryptography and security applications, where reproducibility is not important, and is even undesirable.
- *Pseudo-Random Number Generators (PRNGs)*: Algorithms that generate random numbers using deterministic processes. These are mainly used in simulations and computer programming, where reproducibility is important.

The main goal of a RNG is to generate uniformly distributed random numbers.[4]

1: For a detailed historical overview of random number generation, the reader is referred to: [6]

2: Dice have been found in archaeological sites in Mesopotamia dating back to 3000 BC. [6]

3: The Ferranti Mark I computer, built in 1951, had a hardware random number generator. [4]

4: Generating random numbers from other distributions is usually done by transforming uniformly distributed random numbers. [8]

Tests for Randomness

In addition to the generation of random numbers, it is also important to test their quality. It is impossible to prove that a sequence of numbers is random, since every sequence of numbers should have the same probability of occurring. However, Kendall and Babington-Smith [9] proposed the notion of *local randomness*, which states that every reasonably long segment of a random sequence should appear random, and pass statistical tests for randomness. The aim of a statistical test is to detect evidence against the null hypothesis that the sequence is random. Some of the most common suites of tests for randomness are:

- *Diehard tests* [10]: A suite of tests developed by George Marsaglia, which encompasses a wide range of tests for randomness he developed over the years.
- *Dieharder tests* [11]: A suite of tests developed by Robert G. Brown, which is an extension of the Diehard tests.

- *NIST tests* [12]: A suite of tests developed by the USA National Institute of Standards and Technology (NIST) to test the quality of RNGs.

For a standard introduction to statistical tests for randomness, the reader is referred to [4]. For historical and conceptual aspects of randomness testing, [6, 13] provide a comprehensive overview.

Pseudo-Random Number Generators

Even though this work is focused on TRNGs, it is important to have a basic understanding of PRNGs as they are widely used in simulations and computer programming. PRNGs are algorithms that generate random numbers using deterministic processes. A general framework for PRNGs is the following structure: [6]

- $\mathcal{S}$: a finite set of states, called the *state space*.
- μ: a probability distribution on $\mathcal{S}$, for the *initial state* (also called the *seed*), s_0.
- f: a deterministic function that maps the state space to itself, called the *state transition function*.
- $\mathcal{U}$: the output space, which is usually the interval $[0, 1)$.
- g: a deterministic function that maps the state space to the output space, called the *output function*.

The state at step i, s_i, is updated using the state transition function: $s_{i+1} = f(s_i \in \mathcal{S})$. The output at step i, u_i, is generated using the output function: $u_i = g(s_i) \in \mathcal{U}$.

An early attempt to generate pseudo-random numbers was through the calculation of digits of π [14]. The first proper PRNG was developed by John von Neumann in 1951 [15], known as the *middle-square method*. Von Neumann also suggested that the output of a PRNG should be periodic in a finite precision computer. A very important development in the field of PRNGs was the invention of the *linear congruential generator* by Lehmer in 1951 [16], which is thoroughly described in [4]. Other important PRNGs are the *Mersenne Twister*,[5] [17] the *Xorshift* family [18], and the *MIXMAX* family [19]. For a comprehensive treatment of PRNGs, the reader is referred to [4, 6, 20].

5: which is used in this work when pseudo-random numbers are needed explicitly

True Random Number Generators

As mentioned before, specialized hardware devices that generate random numbers using physical processes have existed for over 70 years [7, 21]. Perhaps the most famous TRNG is the Electronic Random Number Indicator Equipment (ERNIE) [22], which is used in the British Premium Bonds lottery. At its first generation, it produced approximately 50 random digits per second [6].[6] TRNGs are used in cryptography security applications, and in any case where the random numbers should not be reproducible.[7] However, they are avoided in simulations and computer programming, because of their relatively slow speed and their lack of reproducibility.

Unlike PRNGs, TRNGs are usually not made *a priori* to generate uncorrelated random bits. In order to generate proper random numbers, the

6: ERNIE still exists today, but its technology has been significantly changed.

7: In cryptography, this is called the *Kirchoff's principle*, which states that the security of a cryptographic system should not depend on the secrecy of the algorithm, but only on the secrecy of the key.

output of the device is usually post-processed, either by software or by hardware, to remove any bias and correlation, and its output is continuously tested for randomness. The simplest post-processing method is the *von Neumann extractor* [15], which takes pairs of bits from the TRNG output, and outputs a 0 if the pair is 01, a 1 if the pair is 10, and discards the pair if it is 00 or 11.[8] In most cases, the output of a TRNG is used to seed a PRNG at random intervals, in order to generate uniformly distributed random numbers faster.

8: The von Neumann extractor has been improved since then: [23].

Stipčević and Koç [24] classify TRNGs into four categories:

- *Noise-based TRNGs*: These TRNGs generate random numbers using the noise of a physical process, such as Johnson, Zener, or laser phase noise. They generally compare an analog voltage to a reference voltage, and output a 1 if the analog voltage is higher than the reference voltage, and a 0 otherwise.[9]
- *Chaotic TRNGs*: These TRNGs generate random numbers using chaotic systems, such as lasers. Of course, these systems are deterministic, and are completely reproducible if the initial conditions are known.
- *Free-running oscillator TRNGs*: These TRNGs generate random numbers using free-running oscillators. A free-running oscillator is made when a digital signal is fed back to the input of a digital circuit, which then oscillates.
- *Quantum TRNGs*: These TRNGs generate random numbers using quantum processes, such as Geiger counters or quantum optics. These are fundamentally random, and have increased in popularity in recent years.

9: This is the type of TRNG we will consider in this work.

We refrain from providing examples of TRNGs in this work, as there are excellent and recent reviews on the subject. Specifically, for a general overview of TRNGs, the reader is referred to [24],[10] and for a focus on quantum TRNGs, the reader is referred to [25, 26].

10: in which examples of commercial TRNGs are also given

1.2. Calorimetry

The aim of this section is not to describe calorimetry in detail, but to provide the reader with the necessary information to understand the basic principles of the Digital Electromagnetic Calorimeter (DECAL) sensor. For a review of calorimetry in High Energy Physics (HEP), the reader is referred to [27–29].[11]

11: On which the information in this section is based, unless otherwise stated.

Calorimetry in the context of HEP refers to the measurement of the energy of particles. Calorimeters are instruments in which particles to be measured are fully absorbed, and their energy is transformed into a measurable quantity. A calorimeter can be an Electromagnetic Calorimeter (ECAL) or a Hadronic Calorimeter (HCAL). In this essay, we will focus on the ECAL. An ECAL usually measures the energy of electrons or photons through their electromagnetic interactions. A calorimeter can be either *sampling*, which is consisted of alternating layers of an absorber, a dense material used to degrade the energy of the incident particle, and an active medium that provides the detectable signal, or *homogeneous*, which is consisted of a single material that both degrades the energy of the incident particle and provides the detectable signal.

Electrons and photons interact with the matter of an ECAL mainly through a few processes. For energies larger than $\sim$ 10 MeV, the dominant process of electron energy loss is bremsstrahlung (where the electron interacts with the nuclei of the absorber material and emits a photon [30]). For the same energy regime, photons lose their energy mainly through electron-positron pair production. At lower energies, the dominant process for electron energy loss is ionization and thermal excitation through collision, whereas photons lose their energy mainly through Compton scattering and the photoelectric effect. Electrons and photons with high energy ($\geq$ 1 GeV) produce secondary photons by bremsstrahlung, or secondary electrons and positrons by pair production. These secondary particles in turn produce other particles by the same mechanisms, thus giving rise to a cascade (shower) of particles with progressively degraded energies. At one point, the energy of the particles in the shower is low enough that the energy is degraded mainly through ionization and thermal excitation, and the shower stops. This is evident for electrons in Figure 1.1. An important quantity that characterizes electromagnetic showers is the *radiation length* X_0:

$$X_0\left(\mathrm{g/cm^2}\right) = \frac{716.4\,\mathrm{g\,cm^{-2}}A}{Z(Z+1)\ln\left(287/\sqrt{Z}\right)} \tag{1.1}$$

where A is the atomic mass number of the absorber material, and Z is the atomic number of the absorber material.

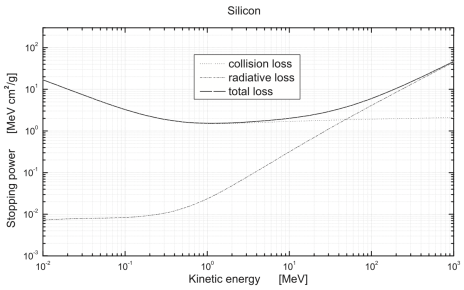


Figure 1.1.: Total, collision and radiation stopping powers as a function of the incoming electron kinetic energy in Si. (Reprinted from [28])

The radiation length is the mean distance over which the energy of an electron is degraded by a factor of $1/e$. Using the radiation length and certain well understood empirical functions, electromagnetic showers can be universally described. In the case of an analog ECAL, the measurement is based on the principle that the energy released in the detector material by the charged particles of the shower, mainly through ionization and

excitation, is proportional to the energy of the incident particle. The energy resolution of such an ECAL is given by:

$$\frac{\sigma_E}{E} = \frac{a}{\sqrt{E}} \oplus \frac{b}{E} \oplus c \tag{1.2}$$

The first term, $\frac{a}{\sqrt{E}}$ is the *stochastic* term. It is due to physical phenomena with statistical nature. Its main contribution is the fluctuations related to the physical development of the shower. In the case of sampling calorimeters, this term also includes the *sampling* contribution, as the energy deposited in the active medium fluctuates event by event because the active layers are interleaved with absorber layers. Finally, it includes the uncertainty in the energy loss through ionization of the active medium, as described by the Landau straggling function [29], which is visualized in Figure 1.2. The second term, $\frac{b}{E}$ is the *noise* term. It is due to the fluctuations in the detector response and depends on the detector technique and on the features of the readout circuit. The third term, c, is the *constant* term. It is due to the non-uniformity of the detector response. In special cases, more terms may be added to the energy resolution equation. For the sampling calorimeter of the A Toroidal LHC Apparatus (ATLAS) experiment of the Large Hadron Collider (LHC) in European Organization for Nuclear Research (CERN), the energy resolution is given by: [31]

$$\frac{\sigma_E}{E} = \frac{10.5\%}{\sqrt{E}} \oplus \frac{170\,\text{MeV}}{E} \oplus 0.7\% \tag{1.3}$$

whereas for the homogeneous calorimeter of the Compact Muon Solenoid (CMS) experiment of the LHC in CERN, the energy resolution is given by: [31]

$$\frac{\sigma_E}{E} = \frac{2.8\%}{\sqrt{E}} \oplus \frac{120\,\text{MeV}}{E} \oplus 0.3\% \tag{1.4}$$

It is noted that the lower stochastic term of the CMS calorimeter is due to the fact that it is a homogeneous calorimeter, in contrast to the sampling calorimeter of the ATLAS experiment.

Theoretical details about analog calorimeters and the interactions between particles and matter can be found in [27–29, 32], whereas a more detailed description of the technical aspects of calorimeters is developed in [27, 33].

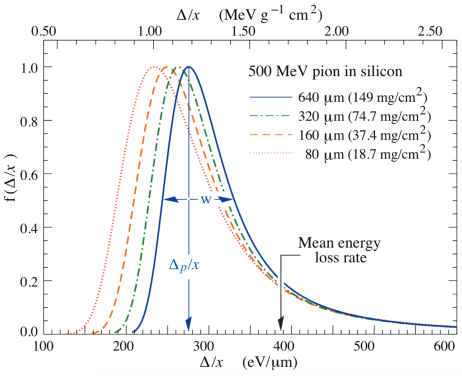


Figure 1.2.: Straggling functions in silicon for 500 MeV pions, normalized to unity at the most probable value $\Delta p/x$. The width w is the full width at half maximum. (Reprinted from [29])

Digital Electromagnetic Calorimeter

The idea of a digital calorimeter is to count the number of particles instead of measuring their energy, and reconstruct the energy of the incident particle from the number of particles. It can be thought of as a sampling calorimeter, since it is consisted of layers of active[12] and sensitive[13] layers [34]. By counting the number of particles, the contribution described by the Landau straggling function is eliminated, and the stochastic term is reduced [35]. A DECAL should have high granularity, be radiation tolerant, have low noise and be constructed with a quick charge collection

12: usually made from tungsten or lead

13: usually made from silicon

and summation logic [32]. Previous works on such calorimeters as well as more requirements are listed in [34, 35].

1.3. Statistics and Maximum Likelihood Estimation

This section will provide a brief introduction to the statistical concepts that are necessary for the rest of this work. For a comprehensive treatment of statistics in the context of HEP, the reader is referred to [36, 37].

Probability Mass and Density Functions

For a discrete random variable X, the Probability Mass Function (PMF) $P(X = x)$ is a function that gives the probability that X takes the value x. For a continuous random variable X, the Probability Density Function (PDF) $f(x)$ is a function that gives the probability that X takes a value in the interval $[x, x + dx]$. The PMF and PDF must satisfy the following properties:

$$\begin{aligned} P(X = x) &\geq 0, \quad \forall x \\ \sum_x P(X = x) &= 1 \\ f(x) &\geq 0, \quad \forall x \\ \int_{-\infty}^{\infty} f(x)dx &= 1 \end{aligned} \tag{1.5}$$

The Cumulative Distribution Function (CDF) of a random variable X, denoted as $F(x)$, is defined as the probability that X takes a value less than or equal to x:

$$F(x) \equiv P(X \leq x) = \begin{cases} \sum_{x_i \leq x} P(X = x_i) & \text{for discrete } X \\ \int_{-\infty}^{x} f(x)dx & \text{for continuous } X \end{cases} \tag{1.6}$$

The distributions that are relevant for this work are the normal distribution, the Poisson distribution, the χ^2 distribution, and the Rayleigh distribution.

A random variable X is said to follow a normal distribution with mean μ and variance σ^2, denoted as $X \sim N(\mu, \sigma^2)$, if its PDF is:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right] \tag{1.7}$$

The normal distribution is the asymptotic limit of many other distributions with physical relevance, and it is widely encountered in nature.[14]

A random variable X is said to follow a Poisson distribution with parameter λ, denoted as $X \sim \text{Poisson}(\lambda)$, if its PMF is:

$$P(X = k) = \frac{\lambda^k}{k!} e^{-\lambda} \tag{1.8}$$

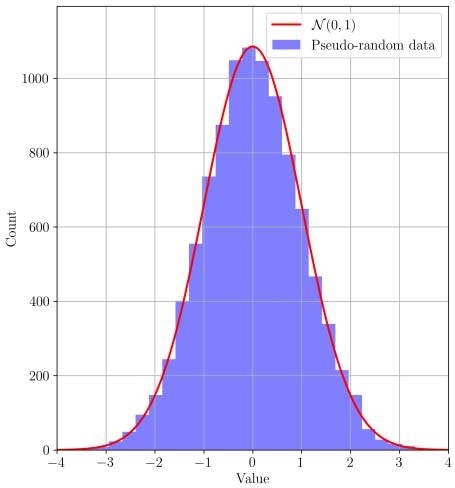


Figure 1.3.: Normal distribution with mean $\mu = 0$ and variance $\sigma^2 = 1$. 10000 simulated samples (blue bins) and the theoretical PDF (red line).

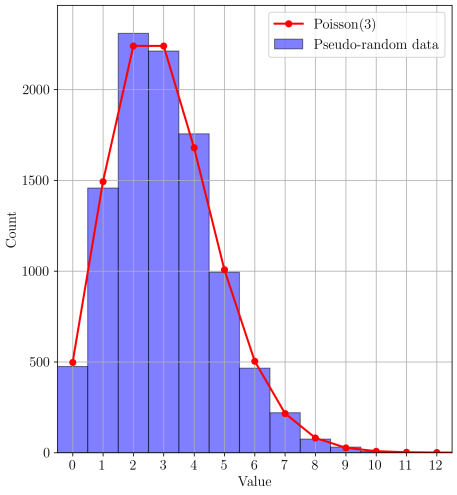


Figure 1.4.: Poisson distribution with parameter $\lambda = 3$. 10000 simulated samples (blue bins) and the theoretical PMF (red line).

14: The central limit theorem states that the sum of a large number of independent and identically distributed random variables converges to a normal distribution. We will not delve into the details of the central limit theorem in this work, but it is one of the most important theorems in probability theory.

The Poisson distribution is used to model the number of events in a fixed interval of time or space, given that the events occur with a constant rate and are independent of each other.

A random variable X is said to follow a χ^2 distribution with k degrees of freedom, denoted as $X \sim \chi^2(k)$, if its PDF is:

$$f(x) = \frac{1}{2^{k/2}\Gamma(k/2)} x^{k/2-1} e^{-x/2} \tag{1.9}$$

where $\Gamma(\cdot)$ is the gamma function, defined as:

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt \tag{1.10}$$

The χ^2 distribution is used to model the sum of the squares of k independent standard normal random variables.

A random variable X is said to follow a Rayleigh distribution with parameter σ, denoted as $X \sim \text{Rayleigh}(\sigma)$, if its PDF is:

$$f(x) = \frac{x}{\sigma^2} e^{-x^2/(2\sigma^2)} \tag{1.11}$$

The Rayleigh distribution is used to model the magnitude of a two-dimensional vector whose components are independent and identically distributed normal random variables.

Maximum Likelihood Estimation

The Maximum Likelihood Estimation (MLE) is a method used to estimate the parameters of a statistical model. Given a set of observations $\{x_1, x_2, \ldots, x_n\}$ that are assumed to be independent and identically distributed, the MLE of the parameters $\theta = (\theta_1, \theta_2, \ldots, \theta_k)$ of the model is the set of parameters that maximize the likelihood function $\mathscr{L}(\theta)$:

$$\mathscr{L}(\theta) = \prod_{i=1}^{n} f(x_i|\theta) \tag{1.12}$$

where $f(x_i|\theta)$ is the PDF (or PMF) of the model. In practice, it is more convenient to maximize the log-likelihood function $\ell(\theta)$:

$$\ell(\theta) = \ln \mathscr{L}(\theta) = \sum_{i=1}^{n} \ln f(x_i|\theta) \tag{1.13}$$

Therefore, the MLE of the parameters is the solution of the following optimization problem:

$$\hat{\theta} = \text{argmax}_\theta \ell(\theta) \tag{1.14}$$

The likelihood function satisfies the following properties:

- The MLE is consistent, i.e., it converges in probability to the true value of the parameters as the number of observations increases.
- The MLE is unbiased, i.e., the expected value of the MLE is equal to the true value of the parameters.
- The MLE is efficient, i.e., it has the smallest possible variance among all unbiased estimators.
- The MLE is invariant, i.e., it is invariant under one-to-one transformations of the parameters.

Binned Maximum Likelihood Estimation

In the case of binned data, we assume that each bin count follows a Poisson distribution. The likelihood function for binned data that are expected to follow the PDF $f(x|\theta)$ is the product of the Poisson probabilities of the observed counts in each bin, and is called the *Binned Maximum Likelihood Estimation (BMLE)*. The log-likelihood function for binned data is:

$$\ell(\theta) = \sum_{i=1}^{n} [k_i \ln \lambda_i - \lambda_i - \ln(k_i!)] \tag{1.15}$$

where k_i is the observed count in bin i, and $\lambda_i = \int_{x_i}^{x_{i+1}} f(x|\theta)dx$.

1.4. Time Series

This section briefly presents some fundamental concepts of time series analysis. The area of time series analysis is vast, and this section will only cover some concepts that are relevant for the rest of this work. The books by Box, Jenkins and Reinsel[15] [38] and Hamilton[16] [39] are excellent and standard references for linear time series analysis, while the book by Kantz and Schreiber [40] provides a detailed introduction to nonlinear time series analysis. The material presented in this section is based on these standard references, unless otherwise stated.

15: for a comprehensive treatment of time series analysis in their entirety

16: for a more mathematical treatment of time series analysis

Time Series Basics

A time series is a sequence of observations $z_1, z_2, \dots, z_n$ ordered in time (or space), which are typically assumed to be realizations of a stochastic process [41]. They can be univariate or multivariate, and can be discrete or continuous. Usually, adjacent observations are dependent. The goal of time series analysis is to model the underlying stochastic and dynamic structure of the time series. There are two kinds of mathematical models that describe any physical phenomenon: *deterministic* and *stochastic*. Deterministic models allow for exact predictions of the future, given the initial conditions, while stochastic models only allow for probabilistic predictions of the future. A time series is regarded as a realization of a stochastic process from an infinite population of possible realizations. Therefore, although it might have a deterministic component, time series are usually modeled as stochastic processes.[17] In this work, we will focus on univariate time series, with discrete time and continuous values:

17: The stochastic component can also be rooted in nonlinear or chaotic dynamics.

$$\{z_t\} \tag{1.16}$$

where z_t is the value of the time series at time t. Moreover, we will assume that the time interval between observations is constant, i.e., $\Delta t = t_{i+1} - t_i =$ constant. Under these assumptions:

$$z_t = z(t) = z(t_0 + i\Delta t), \quad i = 0, 1, 2, \dots \tag{1.17}$$

where t_0 is the initial time and Δt is the time interval between successive observations, numbered by i. From now on, we will interchangeably use the index t or i to refer to the time index.[18] We also define the *backward difference operator* ∇ as:

18: i.e., $z_t = z_i$

$$\nabla z_t = z_t - z_{t-1} \tag{1.18}$$

and the *backward shift operator* B as:

$$Bz_t = (1 - \nabla)z_t = z_{t-1} \tag{1.19}$$

We define the *expectation* of the t-th observation of a time series $\{z_t\}$ as:

$$\mu_t \equiv E[z_t] = \int_{-\infty}^{\infty} z_t f(z_t) dz_t = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} z_i(t) \tag{1.20}$$

where $f(z)$ is the PDF of the time series, and $E[\cdot]$ denotes the expectation operator. The variance of the t-th observation is defined as:

$$\gamma_{0t} \equiv E[(z_t - \mu_t)^2] = \int_{-\infty}^{\infty} (z_t - \mu_t)^2 f(z_t) dz_t = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} (z_i(t) - \mu_t)^2 \tag{1.21}$$

We can also define the j-th *autocovariance* of the t-th observation as:

$$\begin{aligned}\gamma_{jt} \equiv E[(z_t - \mu_t)(z_{t-j} - \mu_{t-j})] = \int_{-\infty}^{\infty} (z_t - \mu_t) \\ (z_{t-j} - \mu_{t-j}) f(z_t, z_{t-1}, \dots, z_{t-j}) dz_t dz_{t-1} \dots dz_{t-j} = \\ \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} (z_i(t) - \mu_t)(z_i(t - j\Delta t) - \mu_{t-j})\end{aligned} \tag{1.22}$$

where $f(z_t, z_{t-1}, \dots, z_{t-j})$ is the joint PDF of the time series at times $t, t-1, \dots, t-j$ and j is the *lag* of the autocovariance. The *autocorrelation* is defined as:

$$\rho_{jt} \equiv \frac{\gamma_{jt}}{\sqrt{\gamma_{0t}\gamma_{0t-j}}} \tag{1.23}$$

Using the autocorrelation, we can define the Autocorrelation Function (ACF) as the function that describes the autocorrelation of the time series at different lags.

Stationarity

A time series is said to be *weakly stationary*[19] if its mean and autocovariance are constant over time, i.e.:

19: or *covariance stationary*

$$\begin{aligned}\mu_t &= \mu, \quad \forall t \\ \gamma_{jt} &= \gamma_j, \quad \forall t\end{aligned} \tag{1.24}$$

where μ and γ_j are constants. A time series is said to be *strictly stationary* if the joint PDF of any set of observations is invariant under time shifts, i.e., for any set of observations $z_{t_1}, z_{t_2}, \dots, z_{t_k}$ and any time shift τ:

$$f(z_{t_1}, z_{t_2}, \dots, z_{t_k}) = f(z_{t_1+\tau}, z_{t_2+\tau}, \dots, z_{t_k+\tau}) \tag{1.25}$$

A strictly stationary time series is also weakly stationary.[20] In this text, the term *stationary* will refer to weak stationarity, unless otherwise stated.

20: Provided that the mean and autocovariances are finite.

Another important concept[21] is the *ergodicity* of a time series. A sta-

21: closely related to stationarity

tionary time series is said to be *ergodic for the mean* if the sample mean converges to the true mean as the number of observations increases, i.e.:

$$\lim_{n\to\infty} \frac{1}{n} \sum_{t=1}^{n} z_t = \mu \tag{1.26}$$

It can be proven [39] that if the autocovariances of a stationary time series are absolutely summable, then the time series is ergodic for the mean.

A useful quantity for linear time series is the *Partial Autocorrelation Function (PACF)*.[22] The PACF of a stationary time series can be theoretically[23] computed from the ACF using the *Durbin recursive algorithm* [42]:

$$\begin{aligned} \phi_{p+1,j} &= \phi_{p,j} - \phi_{p+1,p+1}\phi_{p,p+1-j}, \quad j = 1, 2, \dots, p \\ \phi_{p+1,p+1} &= \frac{\rho_{p+1} - \sum_{j=1}^{p} \phi_{p,j}\rho_{p+1-j}}{1 - \sum_{j=1}^{p} \phi_{p,j}\rho_j} \end{aligned} \tag{1.27}$$

where $\phi_{p,p}$ is the PACF at lag j.

22: It is a measure of the correlation between two observations at different lags, after removing the effect of the intermediate observations. It is rigorously defined in [39].

23: practical computations are usually done using more efficient algorithms

White Noise

A *white noise* time series $\{\alpha_t\}$ is a stationary time series with zero mean, constant variance, and uncorrelated observations at different times, i.e.:

$$\begin{aligned} \mu &= 0 \\ \gamma_j &= \sigma^2 \delta_{j0} \\ \rho_j &= 0, \quad j \neq 0 \end{aligned} \tag{1.28}$$

where σ^2 is the variance of the white noise, and δ_{j0} is the Kronecker delta[24]. If the white noise is also Gaussian, then it is called *Gaussian white noise*:

$$\alpha_t \sim N(0, \sigma^2) \tag{1.29}$$

where $N(\mu, \sigma^2)$ denotes a normal distribution with mean μ and variance σ^2. White noise is a fundamental concept in time series analysis, as it is the building block for many time series models.

24: $\delta_{ji} = 1$ if $j = i$, and $\delta_{ji} = 0$ otherwise

Linear Filter Models

For most time series models, the observations are assumed to be dependent on the previous observations and on a Gaussian white noise term (called *shock*).[25] A *linear filter model* is a model that describes the observations as a linear combination of the previous shocks:

$$z_t = \mu + \psi(B)\alpha_t \tag{1.30}$$

25: These dependencies are not necessarily linear.

where μ is the mean of the time series, α_t is the white noise shock at time t, B is the backward shift operator,[26] and $\psi(B)$ is a polynomial in B that describes the linear dependence of the observations on the shocks. The polynomial $\psi(B)$ is called the *transfer function* of the filter, and can be written as:

26: defined in Eq. (1.19)

$$\psi(B) = 1 + \psi_1 B + \psi_2 B^2 + \ldots = \sum_{j=0}^{\infty} \psi_j B^j \tag{1.31}$$

where ψ_j are the coefficients of the transfer function. If the sequence of these coefficients is absolutely summable, then the time series is called *stable*, and it is stationary [38]:

$$\sum_{j=0}^{\infty} |\psi_j| < \infty \tag{1.32}$$

At this point, it would be useful to define the transformed time series $\tilde{z}_t$ as:

$$\tilde{z}_t \equiv z_t - \mu \tag{1.33}$$

Autoregressive Models

An *Autoregressive (AR)* model of order p, denoted as AR(p), is a model where the observations are a linear combination of the previous observations and a white noise shock:

$$\phi(B)\tilde{z}_t = \alpha_t \quad \text{or} \quad \tilde{z}_t = \phi_1 \tilde{z}_{t-1} + \phi_2 \tilde{z}_{t-2} + \ldots + \phi_p \tilde{z}_{t-p} + \alpha_t \tag{1.34}$$

where $\tilde{z}_t$ is the transformed time series defined in Eq. (1.33), α_t is the white noise shock at time t, and $\phi(B)$ is a polynomial in B of order p:

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \ldots - \phi_p B^p \tag{1.35}$$

We can express the AR model using the linear filter model notation (1.30) as:

$$\tilde{z}_t = \phi^{-1}(B)\alpha_t \equiv \psi(B)\alpha_t = \sum_{j=0}^{\infty} \psi_j B^j \alpha_t = \alpha_t + \psi_1 \alpha_{t-1} + \psi_2 \alpha_{t-2} + \ldots \tag{1.36}$$

provided that the right-hand side of Eq. (1.36) converges. By factorizing the polynomial $\phi(B)$, we get:

$$\phi(B) = (1 - G_1 B)(1 - G_2 B) \ldots (1 - G_p B) \tag{1.37}$$

It can be proven [38] that the AR model is weakly stationary[27] if the roots G_j^{-1} of the polynomial $\phi(B)$[28] lie outside the unit circle in the complex plane.

27: i.e. it satisfies the condition of Eq. (1.32)

28: $\phi(B) = 0$, also called the *characteristic equation*

The ACF of a stationary AR model satisfies the difference equation:

$$\rho_j = \phi_1 \rho_{j-1} + \phi_2 \rho_{j-2} + \ldots + \phi_p \rho_{j-p} \quad j = 1, 2, \ldots \tag{1.38}$$

This system of equations for $j = 1, 2, \ldots, p$ is also called the *Yule-Walker equations* [43, 44].[29] For a known ACF of a stationary AR model, the ϕ_j coefficients of the model can be computed using the Yule-Walker equations. Therefore, along with the mean of the time series and the variance of the white noise shock, the AR model can be completely defined by its ACF.

29: These are important equations for the estimation of the AR model parameters. Detailed discussions can be found in [38, 39, 45].

By solving Eq. (1.38) [38], and assuming distinct roots for the characteristic equation (1.37), we get:

1. A damped exponential decay term proportional to G_i^j for every real root G_i of the characteristic equation.
2. A dampened sinusoidal term proportional to $|G_i|^j \cos(2\pi f_i j + \theta_i)$ for every complex root G_i of the characteristic equation, where f_i is the frequency of the sinusoid and θ_i is the phase.

Therefore, the ACF of a stationary AR model is a linear combination of damped exponentials and dampened sinusoids.

The PACF of an AR model can be computed using the Yule-Walker equations [43, 44], along with Eq. (1.27). From these computations, it can be shown that the PACF of an AR model is zero for lags greater than the order p of the model.

Moving Average Models

A *Moving Average (MA)* model if order q, denoted as MA(q), is a linear filter model where the transfer function is truncated after q terms:

$$z_t = \mu + \theta(B)\alpha_t \tag{1.39}$$

where μ is the mean of the time series, α_t is the white noise shock at time t, and $\theta(B)$ is a polynomial in B of order q:

$$\theta(B) = 1 - \theta_1 B - \theta_2 B^2 - \ldots - \theta_q B^q \tag{1.40}$$

Since the shocks are uncorrelated and are assumed to be Gaussian white noise, the mean and ACF of the MA(q) model are:

$$\mu = E[z_t] = \mu \tag{1.41}$$

$$\rho_j = \begin{cases} \frac{-\theta_j + \theta_1\theta_{k-1} + \theta_2\theta_{k-2} + \ldots + \theta_{q-j}\theta_q}{1 + \theta_1^2 + \theta_2^2 + \ldots + \theta_q^2} & \text{if } j \leq q \\ 0 & \text{if } j > q \end{cases} \tag{1.42}$$

Therefore, the MA model is weakly stationary and ergodic.[30] Note that the q first autocovariances are the equations needed to solve for the q unknowns $\theta_1, \theta_2, \ldots, \theta_q$. Therefore, the MA model can also be completely

30: If the shocks are Gaussian white noise.

defined by its ACF, along with its mean (1.41) and the variance of the white noise shock.

Since the shocks are unknown in principle, it would be useful to express the MA model of Eq. (1.39) in terms of the observed transformed time series $\tilde{z}_t$. This can be done by inverting the polynomial $\theta(B)$:

$$\alpha_t = \theta^{-1}(B)\tilde{z}_t \tag{1.43}$$

It can be shown [38] that $\theta(B)$ is invertible if the roots of the characteristic equation[31] $\theta(B) = 0$ lie outside the unit circle in the complex plane.

A MA model is always stationary, since the series $\psi(B) = \theta(B)$ is finite, and therefore converges.[32]

From Eq. (1.42), it is clear that the ACF of a MA model is zero for lags greater than the order q of the model. It is possible to show [38, 46] that this is not true for the PACF of an invertible MA model, which is nonzero for all lags, and is dominated by exponential decay and dampened sinusoids.

31: note that this is different from the characteristic equation of the AR model

32: see Eq. (1.32)

Autoregressive Moving Average Models

In the previous sections, we have seen that the AR and MA models have different properties. However, an AR model can be expressed as an MA of infinite order, and vice versa. To avoid the infinite order problem, we can combine the AR and MA models into a single model, called the *Autoregressive Moving Average (ARMA)* model. An ARMA model of order p, q, denoted as ARMA(p, q), is a model where the observations are a linear combination of the previous observations and the previous shocks:

$$\phi(B)\tilde{z}_t = \theta(B)\alpha_t \tag{1.44}$$

It can be seen as either a p-order AR model with white noise that follows a q-order MA model, or a q-order MA model with white noise that follows a p-order AR model. This lets us combine the properties of the AR and MA models. Both its ACF and PACF are infinite in length, and behave as a combination of damped exponentials and dampened sinusoids [38].

Linear Nonstationary Models

The AR, MA, and ARMA models are all stationary models.[33] However, many time series are nonstationary, usually due to nonconstant mean. There are many ways a time series can be nonstationary. Specifically, if the roots of the characteristic equation $\phi(B) = 0$ of the general ARMA model lie inside the unit circle in the complex plane, then the time series exhibits *explosive behavior*.[34] However, if they lie on the unit circle, then the time series exhibits *homogeneous nonstationary behavior*.[35] In this case, the time series (1.44) can be expressed as:

$$\phi(B)(1 - B)^d\tilde{z}_t = \theta(B)\alpha_t \tag{1.45}$$

33: They satisfy the condition of Eq. (1.32).

34: An example is the exponential growth of bacteria. It will not be considered in this work.

35: also called *unit root behavior*

where d is the order of the unit root. Since we defined the backward difference operator ∇ in Eq. (1.18), we can rewrite Eq. (1.45) as:

$$\phi(B)\nabla^d \tilde{z}_t = \theta(B)\alpha_t \tag{1.46}$$

The order d of the unit root is called the *order of integration* of the time series, and the model defined in Eq. (1.46) is called an *Autoregressive Integrated Moving Average (ARIMA)* model. Note that $\nabla^d \tilde{z}_t = \nabla^d z_t$ for $d \geq 1$.

The reason this kind of nonstationarity is often encountered in practice is that it models stochastic processes that behave similarly on different levels. For example, for $d = 1$,[36] the backward difference operator ∇ makes the process invariant under constant shifts:

36: which will be the case later in this work

$$\nabla(\tilde{z}_t + c) = \nabla \tilde{z}_t \tag{1.47}$$

Similarly, for $d = 2$, the process is invariant under constant and linear shifts.

Box-Jenkins Methodology

The *Box-Jenkins methodology* [38] is the most widely used approach for identifying, estimating, and diagnosing ARIMA models. It consists of the following steps:

1. **Difference the time series** as many times as needed to make it stationary. This is done by computing the backward difference operator ∇ of the time series. A nonstationary time series can be identified by constant or linearly decreasing ACF and PACF.
2. **Identify the AR and MA orders** of the stationary time series. This is done by examining the ACF and PACF of the time series.
3. **Estimate the parameters** of the ARIMA model. This is done by fitting the model to the data.
4. **Perform diagnostic checks** on the residuals of the model.

The *identification* step is done by examining the ACF and PACF of the stationary time series. In general, we can have the following cases:[37]

37: which are discussed in detail in the previous subsections of this section

1. If the ACF of the stationary time series decays exponentially or sinusoidally, and the PACF cuts off after a certain lag p, then the time series can be modeled as an AR model of order p.
2. If the PACF of the stationary time series decays exponentially or sinusoidally, and the ACF cuts off after a certain lag q, then the time series can be modeled as an MA model of order q.
3. If both the ACF and PACF of the stationary time series decay exponentially or sinusoidally, then the time series can be modeled as an ARMA model.

The *estimation* and *diagnostic checks* steps can be done via a variety of methods, depending on the case under study.

Seasonality and trends

Many time series exhibit *seasonality*. Seasonality is a pattern that repeats at regular intervals, and can be identified by the presence of peaks of the ACF at regular lags. A time series can also have a *deterministic trend*. There can also be quadratic or higher-order terms in the trend.

Periodogram

An alternative way to analyze time-series is to assume that it made up of a sum of sinusoids of different frequencies. If the number of observations is odd, $n = 2q + 1$, then we can fit the Fourier series to the time series:

$$z_t = \alpha_0 + \sum_{k=1}^{q} [\alpha_k \cos(2\pi f_k t) + \beta_k \sin(2\pi f_k t)] + \epsilon_t \tag{1.48}$$

where $f_k = k/n$ is the frequency of the k-th sinusoid, and ϵ_t is the white noise shock.

The least squares estimates of the Fourier coefficients α_k and β_k are given by:

$$\begin{aligned} \hat{\alpha}_0 &= E[z_t] \\ \hat{\alpha}_k &= \frac{2}{n} \sum_{t=1}^{n} z_t \cos(2\pi f_k t) \\ \hat{\beta}_k &= \frac{2}{n} \sum_{t=1}^{n} z_t \sin(2\pi f_k t) \end{aligned} \tag{1.49}$$

The *periodogram* is the squared magnitude of the Fourier coefficients:

$$I(f_k) = \hat{\alpha}_k^2 + \hat{\beta}_k^2 \tag{1.50}$$

If the number of observations is even, $n = 2q$, then the only difference is that the equations remain the same for $k = 1, 2, \ldots, q - 1$, and the Fourier coefficients for $k = q$ are:

$$\begin{aligned} \hat{\alpha}_q &= \frac{1}{n} \sum_{t=1}^{n} (-1)^t z_t \\ \hat{\beta}_q &= 0 \end{aligned} \tag{1.51}$$

with the periodogram for $k = q$ being:

$$I(f_q) = I(0.5) = n\hat{\alpha}_q^2 \tag{1.52}$$

It is worth noting that the maximum frequency that can be estimated is $f_{\max} = 0.5$, since the smallest period that can be estimated is two intervals.

The periodogram and the ACF are transformations of each other [38]. The use of the periodogram is more common in time-series that contain periodic components, and its use in the context of time-series is called

spectral analysis. In this work, we will not use the periodogram,[38] since we are not dealing with periodic time series.

38: with the exception of the cumulative periodogram in Section 2.4

Nonlinear Time Series

The models discussed so far are all linear models, since each observation is a linear combination of the previous observations and the white noise shock. However, many time series are nonlinear, and cannot be modeled by linear models. Nonlinear time series can exhibit complex behavior, such as chaos, bifurcations, and strange attractors. Nonlinear time series analysis is a vast field, and is beyond the scope of this work.[39] A standard and excellent reference is the book by Kantz and Schreiber [40].

39: Since, as we will see in the following sections, the noise of the DECAL is well modeled by a linear model.

1.5. Purpose of the Study

In this study, time series analysis is used to analyze the behavior of the noise of a DECAL, that was originally designed to be used in collider experiments[34, 47, 48].[40] The purpose of this work is threefold:

1. To thoroughly analyze the noise of the DECAL using time series analysis, and to characterize the stochastic properties of the noise. This full characterization of the noise can then be used to improve the performance and calibration of the DECAL.
2. To present the method of using time series analysis in order to generate random numbers from a noisy analog signal, using the noise of the DECAL as a case study. This method can be used in any TRNG with similar output.
3. To explore whether the DECAL in its current, or in a revised form, can be used as a TRNG.

40: from this point on, *DECAL* will be used to refer to this specific sensor

2. Methods

2.1. Setup

The DECAL Sensor

The Digital Electromagnetic Calorimeter (DECAL) sensor [34] is a Monolithic Active Pixel Sensors (MAPS) (i.e. it combines sensor and readout circuit on the same substrate) designed for use as both a digital electromagnetic calorimeter (by calculating the initial energy of the incident particles) and a tracker (by reconstructing the trajectories of the particles). More specifically, it is a Depleted Monolithic Active Pixel Sensor (DMAPS) prototype consisting of 64×64 pixels with 55 µm pitch and an epitaxial layer of 25 µm that is expected to be fully depleted when a low bias voltage is applied. Its pixel readout comprises a comparator that detects a hit when the shaper output falls below a globally set threshold voltage. Only this binary information is readout per 40 MHz clock cycle. The reconfigurable readout groups the pixel hits in either 64 strips, each of size 1×64 pixels, or in four pads of size 16×64 pixels.

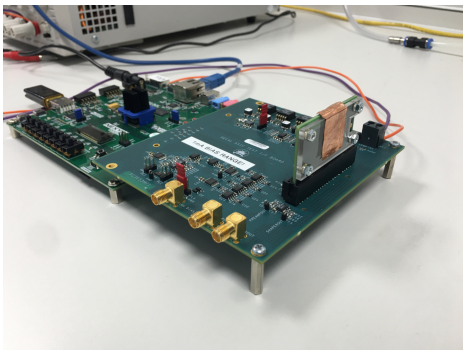


Figure 2.1.: Photograph of the DECAL sensor on its board. Setup similar to the one used in this work. Reprinted from [34].

The core of the sensor is the pixel array. Within a column, hits from the pixel are summed to provide a per column total to the readout logic at the bottom of the column. This either sums the number of hits across all columns (pad mode) or outputs the number of hits per column (strip mode). The chip also includes a test register. This injects 5 bits of data into the summation scheme, which permits the column and periphery circuitry to be tested independently of the pixels. Each of these steps (pixel hit detection, column summation and peripheral readout) requires 25 ns to complete.

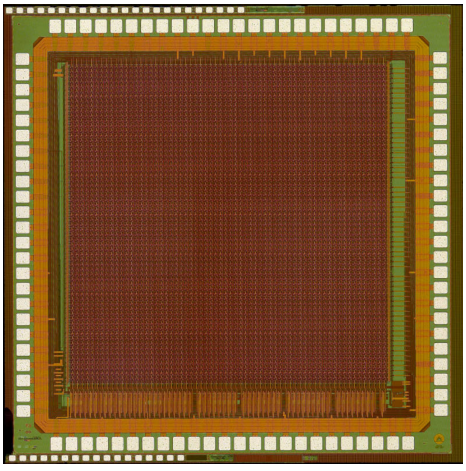


Figure 2.2.: Photograph of the DECAL chip. Reprinted from [34].

Pixel circuity

The circuity of DECAL will be now briefly described. The reader can find much more detail in reference [34]. Each DECAL pixel is composed of a monolithic front-end circuit and comprises an amplification, shaping and discrimination unit (see Figure 2.3). An input voltage pulse induces a step increase in the amplifier, which in turn produces an amplified signal that is transmitted to the shaper. The shaper shapes the pulse coming from the amplifier and helps to filter out noise, improving the Signal-to-Noise Ratio (SNR). To determine whether a hit has occurred, the output of the shaper is then passed to a comparator. This detects a hit if the incoming signal value has passed an externally set detection threshold. Since manufacturing tolerances mean that the shaper output level and the comparator's offset will vary slightly from pixel to pixel, each pixel effectively experiences a slightly different threshold. However, the threshold voltage can only be set globally, for all the pixels. To account for this, a capacitor is placed between the shaper output and the comparator input. The voltage on this capacitor can be set using an in-pixel Digital-to-Analog Converter (DAC).

The value applied to the capacitor allows pixel-to-pixel variations to be tuned out, ensuring that all pixels have the same threshold. The six bits values of the tuning DAC are:

- **Bit 1**: Masks or unmasks the pixel. A masked pixel is deactivated in the sense that its shaper output signal does not reach the discriminator.
- **Bit 2**: Determines the polarity of adding or subtracting to the shaper output.
- **Bits 3-6**: Determine the magnitude of the voltage shift. It is this voltage that is added or subtracted via the second bit setting.

After the tuning, the global threshold voltage can be set and the shaper output of each pixel can be used as the input of its comparator, which compares its voltage with the globally set threshold voltage. Only binary information is stored, when the shaper output drops below the threshold. This way, the discriminator separates the analogue front-end from the digital processing that follows.

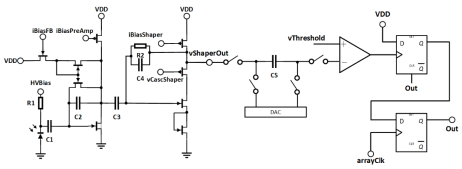


Figure 2.3.: Schematic of the pixel front end. From left to right: amplification, shaping and discrimination unit. Reprinted from [32].

Readout

In this subsection, the summation of the DECAL pixels is summarized. The reader can find many more details about the terms and methods that are presented here in [34]. The pixels are summed over by a mixture of two approaches, in which groups of pixels use the cascade approach (i.e. pixels are iteratively summed in pairs), and the results of these sums combine in a waterfall fashion (i.e. pixels are summed one by one). More specifically, the cascade logic is used in blocks of 16 pixels and then the resulting four blocks are added together in waterfall logic. The DECAL chip has two modes of operation, *strip* and *pad* mode. These modes are described below:

- **Strip mode**: All 64 pixels in one column are readout as one strip. A maximum of three hits per strip and clock cycle can be summed up, with the loss of information about which pixels have detected a hit. In strip mode, the chip reads out each column in turn, sending out 2 bits for each, representing 0, 1, 2, or many hits. Explicitly, $(00)_2 = (0)_{10}$ hits, $(01)_2 = (1)_{10}$ hit, $(10)_2 = (2)_{10}$ hits, $(11)_2 = (3)_{10}$ or more hits. Therefore, for 64 strips, 128 bits are sent out per clock cycle, which means that 4 32-bit numbers can be read per clock cycle.
- **Pad mode**: The pad mode consists of four blocks (pads) of strips (of 64 bits). Each pad corresponds to a block of 16×64 pixel column arrays and can record up to a maximum of 15 hits per column or a maximum of 240 (15×16) total counts. Further, each strip provides an overflow bit, the sum of the overflows (16 per pad at most) is passed via a second 8-bit. Therefore, each pad provides an 8-bit number for the total number of hits and another 8-bit number for the total number of overflows. In total, $2 \times 8 \times 4 = 64$ bits are sent out per clock cycle, which means that two 32-bit numbers can be read per clock cycle.

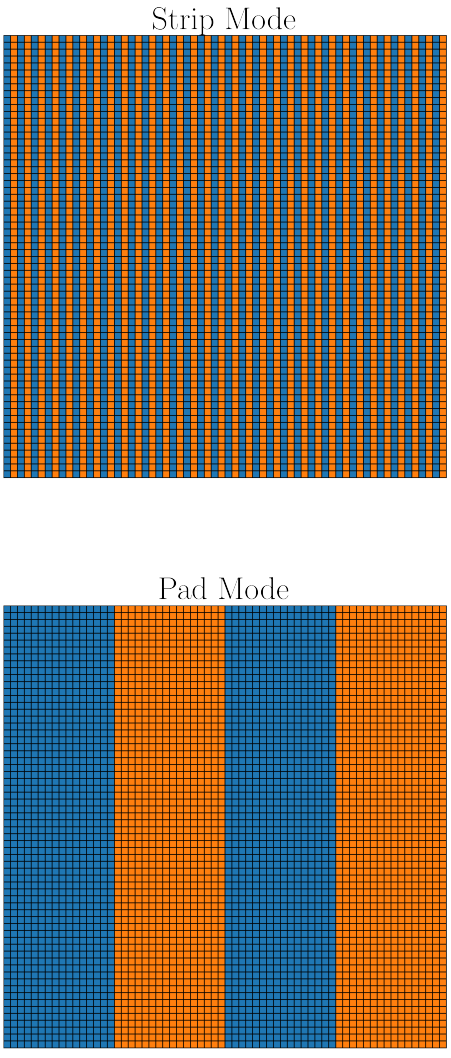


Figure 2.4.: The two readout modes of DECAL. Top: strip mode, bottom: pad mode. The changing colours represent the different readout groups.

The communication between DECAL sensor and computer functions via the Nexys Video Field Programmable Gate Array (FPGA) board and follows the ROOT-based framework of ITk Strips Data Acquisition (ITSDAQ) [49].

Tuning

Tuning of each pixel is necessary because from fabrication the voltage output level varies among pixels and the threshold voltage can only be set globally for all pixels together. Each pixel is tuned through a dedicated 6-bit DAC.[1] It is applied by disconnecting shaper output and comparator input from a capacitor and instead charging it to a specific voltage using the tuning DAC. Once that voltage differential is applied, the tuning DAC gets disconnected the shaper output is reconnected to the comparator input. Therefore, the aim of the tuning procedure is to find the DAC bit configuration per pixel that aligns the shaper outputs in the absence of a signal. The tuning process is applied through a *threshold scan*. A threshold scan is done for each pixel independently. During a threshold scan, the threshold voltage is varied from a low to a high value, and the comparator of the pixel counts only when the shaper output is near the threshold voltage. Therefore, a distribution of counts of different voltages is being generated. This process is repeated for the $2^5 = 32$ different DAC configurations, and 32 different distributions are obtained for each pixel. From configuration 0 to 15 the polarity is negative. That means that the voltage shift is subtracted from the shaper output. On the contrary, from configuration 16 to 31, the polarity is positive, and the voltage shift is added to the shaper output. It is clear that for the same conditions, the response should be linear. However, there is a jump when the polarity changes. The reason is that adding and subtracting a shift of zero results in a slightly shifted voltage output and the drop is preferred in the chip design because it leads to an overlap in the voltage range of both polarities. In order to tune the pixels, a nominal value is chosen and the DAC configuration whose mean is closest to this value is set.

1: which has been outlined in subsection Pixel circuity

Software and Computer

The whole board is connected with a computer with a 1000 Mbps Ethernet connection. The computer is equipped with two Intel Core i3-2120 @ 3.30 GHz processors and 8 GB of RAM. The storage is a 1 TB Seagate HDD. The operating system is Ubuntu 18.04.6, and the communication is done using the ITSDAQ software, which is based on European Organization for Nuclear Research (CERN)'s ROOT framework.

2.2. Data Acquisition and Preprocessing

Primary Data

The primary data are obtained by performing consecutive threshold scans[2] on a strip[3]. In this work, we will focus on data obtained by one of the 64×64 pixels of the sensor. We choose the pixel at column/strip 31 and row 39.[4] There is no specific reason for this arbitrary choice, and we have observed similar results for other pixels as well.[5] For this application, there is no need to tune the sensor, as we are only interested about its noise and do not want to filter it out. However, it is more efficient to tune the pixels to a certain voltage value, so that we can scan over a smaller range of threshold voltages. We choose to tune them around the value of 1.16 V. Each threshold scan is controlled by the following parameters:

- **Strip** (`row`)[6]: The strip that is being scanned. In this case, it is 31.
- **Offset** (`offset`): The offset of the threshold voltage, i.e. the minimum threshold voltage. In this case, it is 1.10 V.
- **Range** (`range`): The range of the threshold voltage, i.e. the difference between the maximum and the minimum threshold voltage. In this case, it is 0.08 V.
- **Bin edges** (`nSteps`): The number of bin edges[7] inside the range. The space is divided into `nSteps-1` equal parts of width `range / (nSteps-1)`. Each threshold voltage is saved in the middle of the corresponding bin of width `range / (nSteps-1)`. In this case, `nSteps` is 41, and therefore the width of each bin is 0.002 V.[8]
- **Number of scans** (`nStrobes`): The number of scans that are performed for each threshold voltage scan. In this case, it is 2000.[9]
- **Number of cycles between scans** (`nSleep`): The number of cycles that are waited between two consecutive scans. This is necessary because the sensor needs some time to reset after each scan.[10] In this case, it is 300.
- **Maximum number of cycles** (`MaxCap`): The maximum number of cycles that are waited before a reset is performed. This parameter exists in order to reset the sensor in the middle of a threshold scan, in order to avoid data corruption due to a voltage drift that has been documented in previous works [32, 48]. For this application, it is not necessary to perform long scans, since we need quick consecutive scans to estimate the noise points. Therefore, this parameter is disabled. However, the pixel resets anyway before each scan.[11]

2: introduced in subsection Tuning

3: using the strip mode of the DECAL sensor

4: Therefore, we consecutively perform threshold scans for the strip 31, and only process the data of the pixel at row 39.

5: See Appendix B

6: these parentheses indicate the name of the variable in the code

7: since the data are saved in a histogram format

8: i.e. we sample the threshold voltage at 40 different values starting from 1.10 V and ending at 1.18 V with a step of 0.002 V

9: The number of scans is chosen to be large enough so that the noise can be estimated accurately. Of course, increasing the number of scans will increase the time needed for the data acquisition.

10: It can be found by decreasing the value until the data are corrupted, and is dependent on the setup

11: This is not strictly necessary, and might slow down the data acquisition. However, since the source and the consequences of the voltage drift are not well understood, for this work we choose to reset the sensor before each scan.

Data Preprocessing

As mentioned in the previous subsection, the data are saved in a histogram format. Each histogram is saved in a Comma Separated Values (CSV) file, where the first two columns are the bin edges and the third column is the number of counts in each bin. As mentioned in previous works [32, 47], the histogram of each pixel is expected to be normally distributed. Therefore, we can estimate the mean of the noise by fitting a Gaussian to the histogram using the Binned Maximum Likelihood Estimation (BMLE) method, presented in subsection Binned Maximum Likelihood Estimation. Detailed expressions and their derivations can be found in Appendix A.2.

2.3. Data Analysis and Random Number Generation

ARIMA Fitting

By following the procedure outlined in Section 2.2, we obtain a time series of the mean of the noise of the pixel under study. In principle, these values are expected to be temporarily correlated. In order to model this correlation, we use the Autoregressive Integrated Moving Average (ARIMA) model. By fitting the ARIMA model (Eq. (1.46)) to the time series, we can estimate the parameters of the model and extract the white noise component. We fit the time series using the *innovations Maximum Likelihood Estimation (MLE)* algorithm, details of which can be found in [50]. For the computation, we use the `statsmodels` library [51] in Python.

ARIMA Model Selection

The ARIMA model is defined by three parameters: the autoregressive order p, the differencing order d, and the moving average order q. There are many methods to select the optimal values of these parameters.[12] In this work, we use the Akaike Information Criterion (AIC) [52] and the Bayesian Information Criterion (BIC) [53] to select the optimal model. The AIC is defined as:

12: Many of them are discussed in [38, 50].

$$\mathrm{AIC}_{p,q} = -2\frac{\ln \mathscr{L}_{\max} + 2(p+q+1)}{n} \approx \ln\left(\hat{\sigma}_\alpha^2\right) + \frac{2(p+q+1)}{n} + \text{constant} \tag{2.1}$$

where $\mathscr{L}_{\max}$ is the maximum likelihood of the model after fitting, p is the autoregressive order, q is the moving average order, n is the number of observations and $\hat{\sigma}_\alpha^2$ is the estimated variance of the white noise. The BIC is given by:

$$\mathrm{BIC}_{p,q} = \ln\left(\sigma_\alpha^2\right) + \frac{(p+q+1)\ln(n)}{n} + \text{constant} \tag{2.2}$$

According to the AIC and the BIC, the optimal model is the one that minimizes these values. The BIC gives a higher penalty for the number of parameters, and therefore tends to select simpler models than the AIC.

Random Number Generation

After fitting the ARIMA model to the time series, we can extract the white noise component of the model, using Eq. (1.46):

$$\hat{\alpha}_t = \hat{\theta}^{-1}(B)\hat{\phi}(B)\tilde{w}_t \tag{2.3}$$

where $\hat{\theta}^{-1}(B)$ and $\hat{\phi}(B)$ are the inverse autoregressive and moving average polynomials, respectively, $\tilde{w}_t$ is the observations after differencing and

subtracting the mean, and $\hat{\alpha}_t$ are the residuals. The $\hat{\cdot}$ symbol indicates that these are the estimated values of the parameters.

The white noise component is expected to be a time series of uncorrelated, normally distributed random numbers. In principle, this is the end goal of this work: these are hardware generated true random numbers. There are many approaches in which someone can create random binary numbers from these residuals. In order to fully exploit the residuals, one should use all the information that they contain, i.e. the full resolution of the residuals. However, since this work is focused on exploring the potential of the DECAL sensor to generate random numbers, we will use a more conservative approach.[13] Since the residuals are normally distributed around zero, the simplest way to create random numbers is to assign a bit on each residual. If the residual is positive, the bit is set to 1, otherwise it is set to 0. This way, we can create a binary string of length equal to the length of the time series.

This should make completely random and uncorrelated random bits, provided that the time series is an ARIMA process and the fit is done correctly. However, in order to tackle any possible small nonlinearities or deviations of the estimated residuals from the true residuals, an extra shuffle is performed. The shuffle is done using only information from the generated binary string, and can be done continuously and computationally efficiently while generating the random bits. The algorithm is this:

1. **Segment Division**: Divide the binary string into segments of length $L = 8$ bytes. Iterate over each segment to perform the bit manipulation operations.
2. **Determine Shuffle Factor**:
   a) Within each segment, locate the first occurrence of the bit '1'
   b) Convert the next two bits following this '1' into a decimal value and add 1 to this value. This results in a number between 1 and 4, which will be denoted as N_{shuffle}.
3. **Skip Segments**: Continue iterating over the next N_{shuffle} segments without any modifications. When the N_{shuffle}-th segment is reached, proceed to the next step.
4. **Determine Swap Factor**:
   a) In the N_{shuffle}-th segment, locate the first occurrence of the bit '0'.
   b) Convert the next two bits following this '0' into a decimal value and add 1 to this value. This results in a number between 1 and 4, which will be denoted as N_{swap}.
5. **Bit Swapping**: Perform bit swapping within the next segments as follows:
   a) Swap the first bit of the N_{shuffle}-th segment with the first bit of the $(N_{\text{shuffle}} + N_{\text{swap}})$-th segment.
   b) Swap the second bit of the N_{shuffle}-th segment with the second bit of the $(N_{\text{shuffle}} + 2 \times N_{\text{swap}})$-th segment.
   c) Continue this process for all bits in the segment.

   In order to do this, a maximum of 32 bytes are needed to be stored in memory.
6. **Repeat the Process**: Repeat the above steps until the end of the binary string is reached.

13: In which aspects such as round off and measurement errors are not expected to significantly affect the results

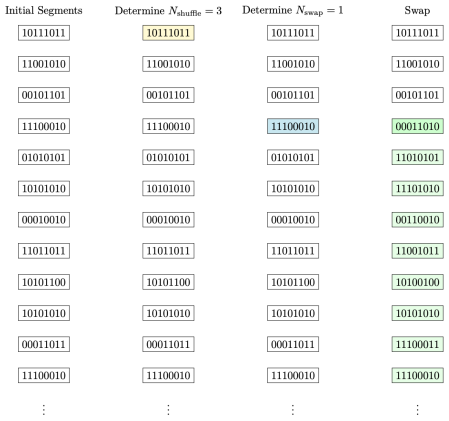


Figure 2.5.: Example for the swap algorithm. See text for details.

2.4. Random Number Testing

After generating the random numbers, we should test them to ensure that they are indeed random. We will first test the residuals, and then the binary numbers that are generated from them.

ARIMA Model Diagnostics

The residuals of the ARIMA model are expected to be normally distributed, with zero mean and constant variance, and to be temporally uncorrelated. First, we can visually inspect the residuals by plotting their histogram and fit a normal distribution to them. We can also plot the residuals over time to check for any temporal correlation. For a white noise process, the Autocorrelation Function (ACF) is expected to follow a normal distribution around zero, with standard deviation equal to $1/\sqrt{n}$, where n is the number of observations.[38] We can therefore plot the ACF of the residuals and check if it follows these limits. However, in practice we do not know the true residuals α_t, but only the estimated residuals $\hat{\alpha}_t$. It can be proven [54] that the errors of the estimated parameters significantly affect the standard errors of the residuals' ACF for small lag values. Box and Pierce [54] have shown how to correct the ACF standard errors for the estimated parameters.

In order to assess the ACF of the residuals as a whole, Box and Pierce [54] introduced a test, later modified by Ljung and Box [55], called the *Ljung-Box-Pierce* test. The test statistic is defined as:

$$Q = n(n+2)\sum_{k=1}^{K}\frac{\hat{\rho}_k^2}{n-k} \tag{2.4}$$

where n is the number of observations, K is the number of lags,[14] $\hat{\rho}_k$ is the sample ACF at lag k, and Q is the test statistic. The test statistic is expected to follow a χ^2 distribution with $K-p-q$ degrees of freedom, where p is the autoregressive order and q is the moving average order. The null hypothesis is that the residuals come from a white noise process.

14: in this work, we choose $K = 100$

Another check that is performed is the *cumulative periodogram* test.[38] Since the ACF is not a sensitive indicator of seasonality, the cumulative periodogram test is used to detect periodicity in the residuals. We define the *normalized cumulative periodogram* as:

$$C(f_k) = \frac{1}{ns^2}\sum_{j=1}^{k} I(f_j) \tag{2.5}$$

where f_k is the frequency, s^2 is the variance estimate of the residuals, and $I(f_j)$ is the periodogram[15] at frequency f_j. For a white noise process, the cumulative periodogram is expected to be close to a straight line, joining the points $(0,0)$ and $(0.5,1)$. For a process characterized by low frequency components, the cumulative periodogram will be above this line, and for a process characterized by high frequency components, the cumulative periodogram will be below this line. For seasonality, the cumulative periodogram will show spikes at the corresponding frequencies. The

15: see Subsection Periodogram

standard error of the cumulative periodogram using the Kolmogorov-Smirnov test are parallel lines at distances $\pm K_\epsilon/q$ from the cumulative periodogram, where $K_\epsilon = 1.36$ for a 95% confidence interval, and $q = (n-2)/2$ for n even and $q = (n-1)/2$ for n odd.

Recurrence Plot

The *recurrence plot* is a graphical representation of the recurrence of a state in a time series. It is used to detect periodicity and other patterns[16] in the time series. The recurrence plot is defined as:

16: even nonlinear

$$R(i,j) = \Theta(\epsilon - \|x_i - x_j\|) \tag{2.6}$$

where $R(i,j)$ is the recurrence plot, Θ is the Heaviside step function[17], ϵ is the threshold distance, and x_i and x_j are the time series at times i and j, respectively. The recurrence plot is a binary matrix, where $R(i,j) = 1$ if the distance between the two points is less than ϵ, and $R(i,j) = 0$ otherwise. The recurrence plot is symmetric, and the diagonal line represents the recurrence of the same state. The recurrence plot can be used to detect periodicity, deterministic chaos, and other patterns in the time series. The threshold distance ϵ is a free parameter, and its value depends on the time series. In this work, we choose $\epsilon = 0.001$. An introduction to the recurrence plot for time series analysis can be found in [56].

17: which is equal to 1 if the argument is positive and 0 otherwise

Diffusion Simulation

In order to test the bits that are generated from the residuals, we can simulate a diffusion process. The diffusion process is a random walk process, in which the position of a particle is determined by a series of random steps. We simulate two versions of the diffusion process.

- **One-dimensional diffusion**: The particle starts at position 0 and at each step it moves up if the random bit is 1 and down if the random bit is 0. The position of the particle is recorded at each step.
- **Two-dimensional diffusion**: The particle starts at position $(0,0)$ and at each step it moves up if the next two random bits are 00, down if they are 10, left if they are 11 and right if they are 01. The position of the particle is recorded at each step.

We also use the Mean Squared Displacement (MSD) to quantify the diffusion process. The MSD is defined as:

$$\mathrm{MSD}(t) = \langle (x(t) - x(0))^2 \rangle \approx \frac{1}{N} \sum_{i=1}^{N} (x_i(t) - x_i(0))^2 \tag{2.7}$$

where $x(t)$ is the position of the particle at time t, $x(0)$ is the initial position, N is the number of particles, and the brackets $\langle \cdot \rangle$ denote the average over all particles. The MSD is expected to follow a linear relationship with time, with a slope equal to the diffusion coefficient.

We break the random bits into segments, and for each segment we simulate the random walk process. We then gather the statistics using these segments and compare them to the expected values.

For the one-dimensional case, we expect the position of the particle to be normally distributed[18] around the origin, with a standard deviation equal to $\sqrt{t}$, where t is the number of steps. Approximately $2/3$ of the particles are expected to be within one standard deviation from the origin, and 95% of the particles are expected to be within two standard deviations from the origin. The MSD is expected to follow a linear relationship with time, with a slope equal to 1. We calculate it using an efficient Fast Fourier Transform (FFT) based algorithm, described in [57], and perform similar pseudo-random number simulations to create the 66% and 95% confidence intervals. This test checks whether the number of 1s and 0s are balanced in multiple segments of the random bits.

18: for large number of steps; for small number of steps, the distribution is binomial

For the two-dimensional case, the particle should perform a one dimensional random walk in the x and y directions. For large number of steps, the position of the particle should be symmetrically distributed around the origin, with a distance from the origin that asymptotically follows the Rayleigh distribution, with scale parameter equal to $\sqrt{t}$.[58] This test checks whether the number of 00s, 01s, 10s and 11s are balanced in multiple segments of the random bits.

NIST Test Suite

The USA National Institute of Standards and Technology (NIST) test suite is a collection of tests that are used to assess the randomness of a sequence of random numbers.[12] It is considered a standard in the field of random number testing, and is widely used to test and validate random number generators. The NIST test suite consists of 15 tests:

1. **Frequency (Monobit) Test**: To test the proportion of 0s and 1s in the entire sequence. It checks if the number of 0s and 1s are approximately the same, indicating a random sequence.
2. **Frequency Test within a Block**: Similar to the monobit test but within smaller blocks of the sequence. It checks if the proportion of 0s and 1s is approximately the same within each block.
3. **Runs Test**: To test the number of runs[19] in the sequence. It checks if the number of runs is approximately the same as expected in a random sequence.
4. **Longest Run of Ones in a Block**: To test the longest run of 1s in the sequence. It checks if the longest run of 1s is approximately the same as expected in a random sequence.
5. **Binary Matrix Rank Test**: To check for linear dependencies among fixed-length substrings of the original sequence by analyzing the rank of matrices constructed from the sequence.
6. **Discrete Fourier Transform (Spectral) Test**: To identify periodic features (i.e., repetitive patterns that are near each other) in the sequence.
7. **Non-overlapping Template Matching Test**: To count the occurrences of pre-specified target strings in the sequence. This test checks for the randomness of specific patterns.

19: sequences of consecutive 0s or 1s

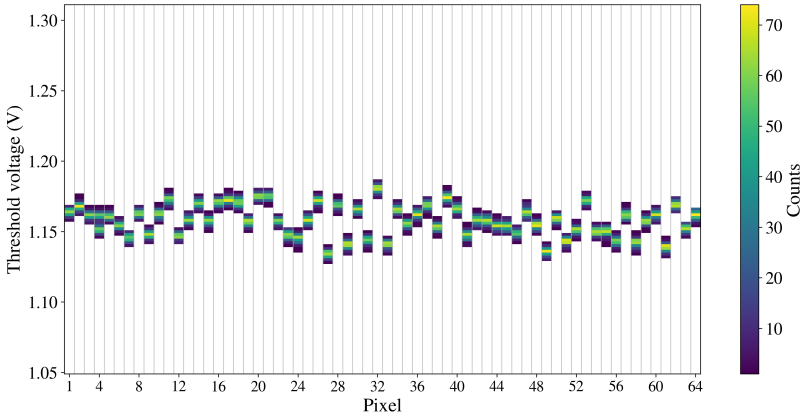
8. **Overlapping Template Matching Test**: Similar to the non-overlapping template matching test, but with overlapping substrings.
9. **Maurer's Universal Statistical Test**: To test the compressibility of the sequence. It checks if the sequence is compressible, which is an indication of non-randomness.
10. **Linear Complexity Test**: To test the linear complexity of the sequence. It checks the length of a linear feedback shift register that produces the sequence.
11. **Serial Test**: To test the number of occurrences of specific patterns in the sequence. It checks for the randomness of specific patterns.
12. **Approximate Entropy Test**: To test the approximate entropy of the sequence. It checks the unpredictability of the sequence.
13. **Cumulative Sums (Cusum) Test**: To test the cumulative sums of the sequence. It checks if the cumulative sums are approximately zero, indicating randomness.
14. **Random Excursions Test**: To count the number of cycles between zero crossings and the distribution of visits to a particular state within each cycle. Evaluates the number of visits to a specific state within a cycle of cumulative sums.
15. **Random Excursions Variant Test**: Similar to the random excursions test, but with additional states.

3. Results and Discussion

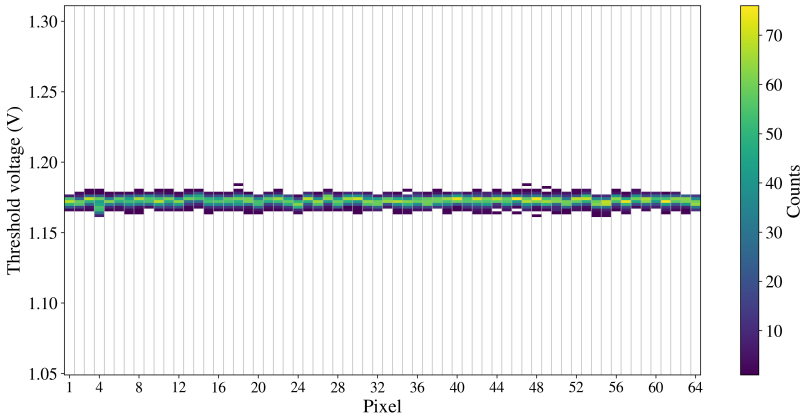
3.1. Tuning Results

As mentioned in Section 2.2, before any measurements were taken, the pixels were tuned (see Subsection Tuning) to limit the range of the threshold scans and decrease the measurement time. The pixel polarities and Digital-to-Analog Converter (DAC) values after the tuning procedure can be seen in Figure 3.1. The histograms for a row of pixels before and after tuning are shown in Figures 3.2a and 3.2b, respectively.[1] It is evident that after tuning, the threshold scan range is significantly reduced, and the pixels are more uniform. This lets us focus the threshold scans on the relevant range and reduce the measurement time.

1: The tuning parameters remain sufficiently constant over time for each pixel, which considerably reduces the measurement time, as there is no need to retune the pixels before each measurement.



(a) Histogram of a row of pixels before tuning.



(b) Histogram of a row of pixels after tuning.

Figure 3.2.: Comparison of row of pixels before and after tuning.

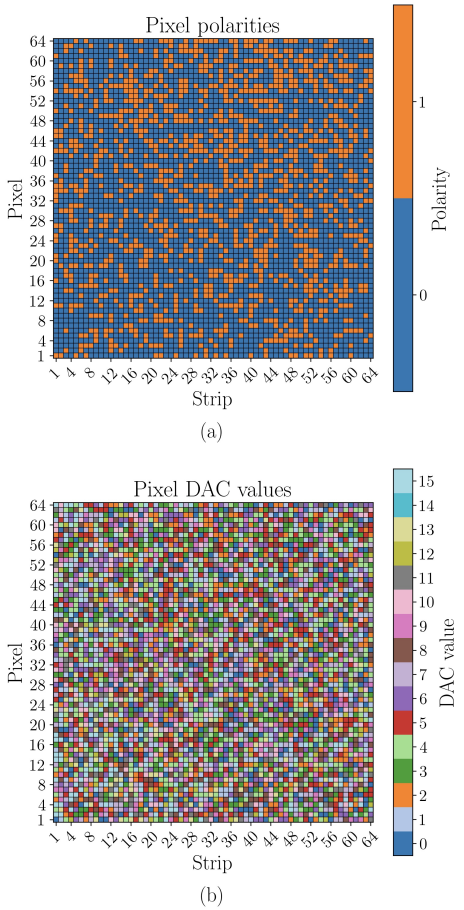


Figure 3.1.: The pixel polarities (a) and DAC values (b) after the tuning process.

3.2. Threshold Scan Mean Time Series

Consecutive threshold scans were performed on the same row of pixels, as described in Subsection Primary Data. This created a series of Comma Separated Values (CSV) files, each containing the threshold scan histogram for a single scan. These files were then processed to extract the mean and standard deviation of the threshold distribution for each scan, following the procedure described in Subsection Data Preprocessing. An example of this fitting can be seen in Figure A.2. The mean for each scan is plotted in Figure 3.3. From this figure, it is evident that the threshold scan mean series[2] quickly drops for the first few scans, after which it stabilizes. This is likely due to the warm-up of the sensor, and has also been noted and discussed in previous studies [32]. In our case, it seems that the sensor stabilizes after around 10000 scans. We therefore discard the first 10000 scans in the rest of the analysis. It can also be seen that the behavior of the time series of the different pixels is similar, as is evident from certain common peaks and valleys in Figure 3.3. The same observation can be made from similar plots in the literature [32]. This suggests that the time series of the different pixels of the same strip are correlated.

2: from now on, the term *time series* will be used for the threshold scan mean time series

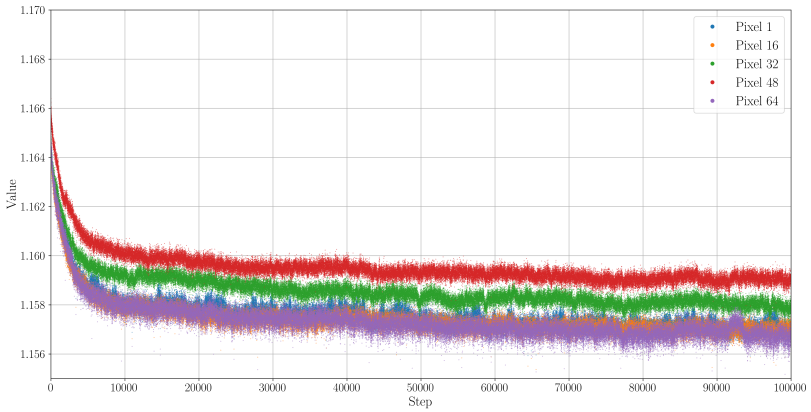


Figure 3.3.: Threshold scan time series for four pixels in a single strip.

The time series of a single run of the sensor[3] after discarding the first 10000 values, along with its histogram, Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF), can be seen in Figure 3.4.

3: under the conditions described in Subsection Primary Data

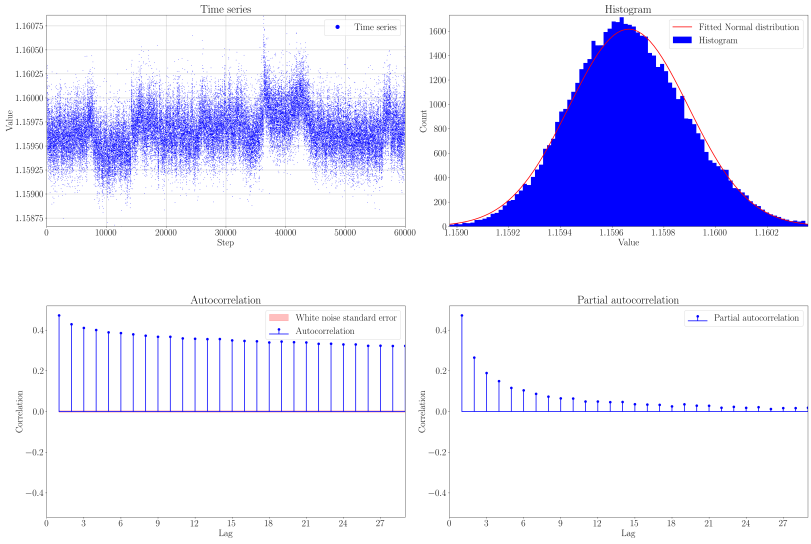


Figure 3.4.: Threshold scan time series (top left), histogram (top right), ACF (bottom left) and PACF (bottom right) for a single pixel after discarding the first 10000 scans. The white noise standard error of the ACF (in which 66.6% of the values should lie) is calculated as $1/\sqrt{N}$, where N is the number of samples.[38]

From this figure, it is clear that the time series is heavily autocorrelated, as the ACF and PACF show significant values. Specifically, the slow and linear decrease of the ACF and PACF suggest non-stationarity. Therefore, the time series is differenced to make it stationary. The differenced time series, along with its histogram, ACF and PACF, can be seen in Figure 3.5.

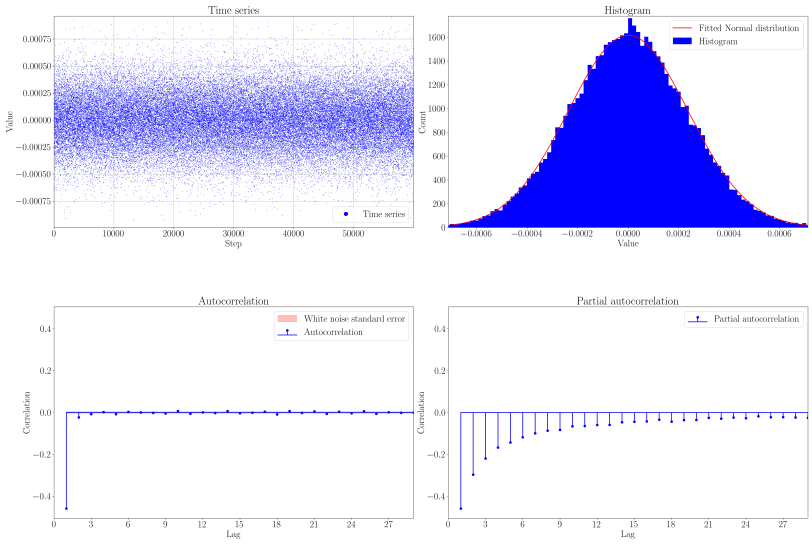


Figure 3.5.: Differenced threshold scan time series (top left), histogram (top right), ACF (bottom left) and PACF (bottom right) for a single pixel after discarding the first 10000 scans. The white noise standard error of the ACF (in which 66.6% of the values should lie) is calculated as $1/\sqrt{N}$, where N is the number of samples.[38]

From this figure, it is evident that the differenced time series is stationary, as the ACF and PACF do not exhibit slow and linear decreases. The differenced time series is then used for the rest of the analysis. Moreover, it seems that the histogram of the differenced time series is approximately Gaussian. However, the ACF and PACF still show significant values, suggesting that the time series is not yet white noise. Specifically, the ACF shows significant values at lags 1, 2 and 3, and the PACF seems to decay exponentially. This would suggest that the time series is an Autoregressive (AR)(3) process. Nevertheless, since there are some deviations from this behavior that might suggest contributions from other AR and Moving Average (MA) terms, we will explore different models and determine the best one using the Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC). The resulting AIC and BIC values for different AR and MA orders can be seen in Figure 3.6.

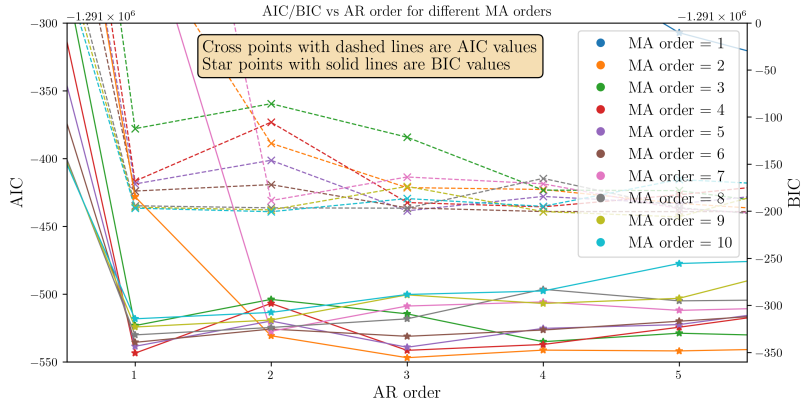


Figure 3.6.: AIC and BIC values for different AR and MA orders for a single pixel.

Using the AIC and BIC values of Figure 3.6, we choose the Autoregressive Integrated Moving Average (ARIMA)(3, 1, 5) model for the time series of

the pixel in question, as it has a low AIC[4] and BIC[5] value. The residuals of this model, along with their histogram, ACF and PACF, can be seen in Figure 3.7.

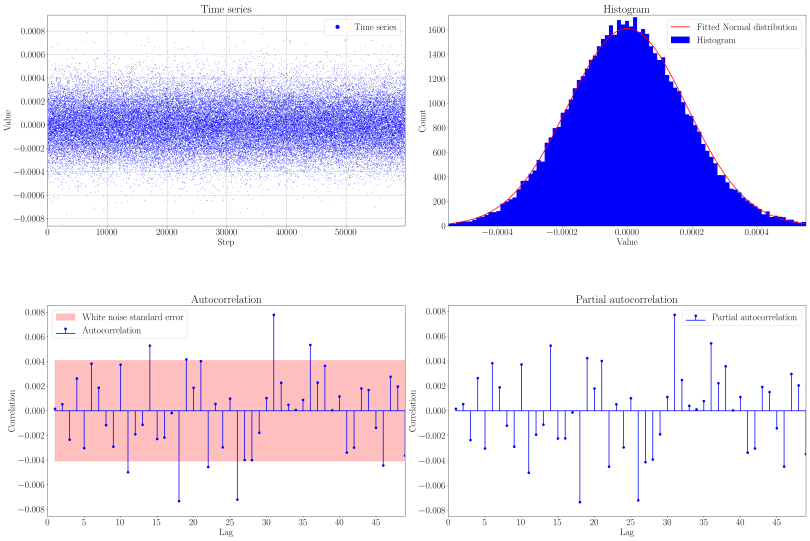


Figure 3.7.: Residuals of the ARIMA(3,1,5) model for a single pixel. The histogram (top right), ACF (bottom left) and PACF (bottom right) are shown. The white noise standard error of the ACF (in which 66.6% of the values should lie) is calculated as $1/\sqrt{N}$, where N is the number of samples.[38]

From this figure, it seems that the residuals are Gaussian white noise, as the ACF and PACF are significantly close to zero, and the histogram is Gaussian. This suggests that the ARIMA(3,1,5) model is a good fit for the time series of the pixel in question.

4: which gives more weight to goodness of fit

5: which gives more weight to model complexity

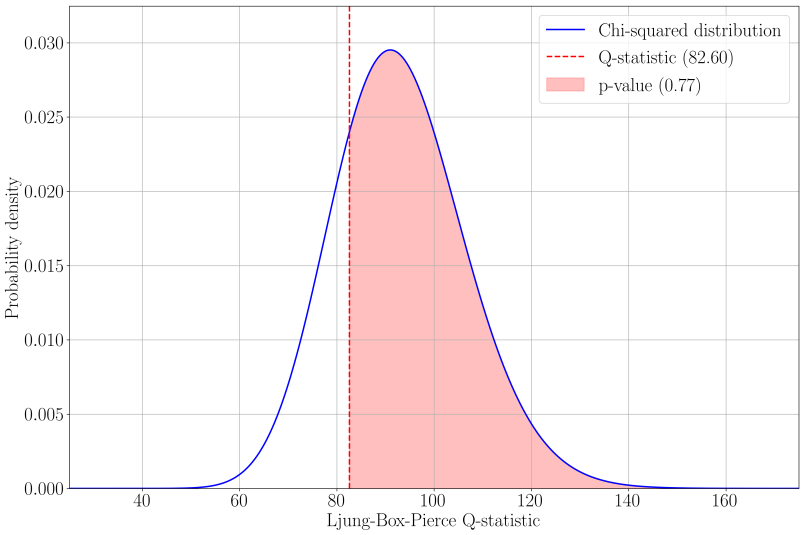
3.3. Random Number Validation

ARIMA Model Diagnostics

The ARIMA(3,1,5) model was fitted to the time series of the chosen pixel. The residuals of this model were then analyzed to determine if the model was a good fit. These tests address whether the residuals are Gaussian white noise[6].

The normality of the residuals is assessed using a Q-Q plot.[7] The Q-Q plot of the residuals can be seen in Figure 3.8. From this figure, it is evident that the residuals are approximately normally distributed, as the points lie close to the line.

The autocorrelation of the residuals is assessed by visually inspecting the ACF and PACF of the residuals in Figure 3.7, and by the Ljung-Box-Pierce test, which was presented in the Subsection ARIMA Model Diagnostics. The Ljung-Box-Pierce result for the residuals is visualized in Figure 3.9. From this figure, it is evident that the p-value is larger than 0.05, and the null hypothesis that the residuals are white noise cannot be rejected. This further suggests that the ARIMA(3,1,5) model is a good fit for the time series of the pixel in question.



The seasonality of the residuals is assessed using the cumulative periodogram, as described in Subsection ARIMA Model Diagnostics. The cumulative periodogram of the residuals can be seen in Figure 3.10. From this figure, it is evident that the data do not show any significant seasonality, as there are no significant peaks, and have an excess of high frequencies. After the ARIMA fitting, the residuals follow the expected line, which suggests the lack of seasonality and autocorrelation.

Finally, the residuals are checked for nonlinear behavior using recurrence plots, also described in Subsection ARIMA Model Diagnostics. The recurrence plot of the data and the residuals can be seen in Figure 3.11. From this figure, it is evident that the residuals are less autocorrelated than the data, as the recurrence plot of the residuals is more uniform than the recurrence plot of the data.

6: i.e. normally distributed with zero mean and constant variance, and uncorrelated

7: Due to the large sample size, most quantitative tests are not very useful, because they are overly sensitive to small deviations from normality

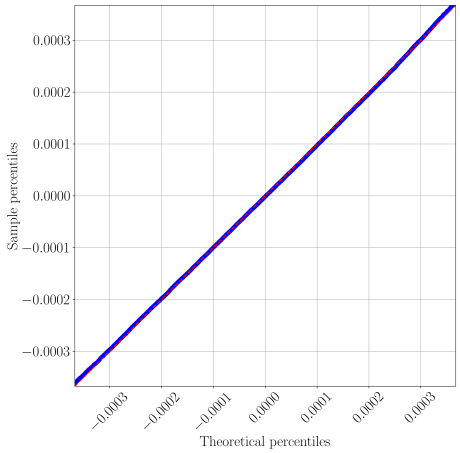


Figure 3.8.: Q-Q plot of the residuals of the ARIMA(3,1,5) model for a single pixel.

Figure 3.9.: Ljung-Box-Pierce test for the residuals of the ARIMA(3,1,5) model for a single pixel. The blue line is the relevant χ^2 distribution, the red dashed line is the Q statistic, and the shaded area is the p-value.

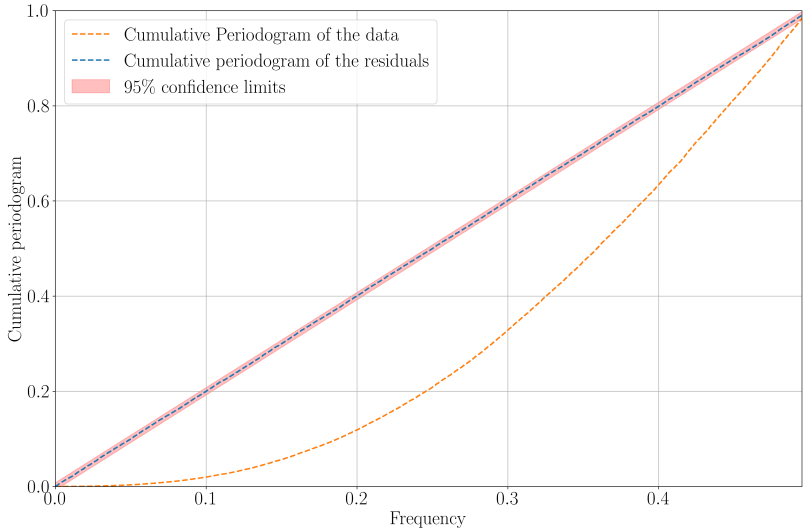
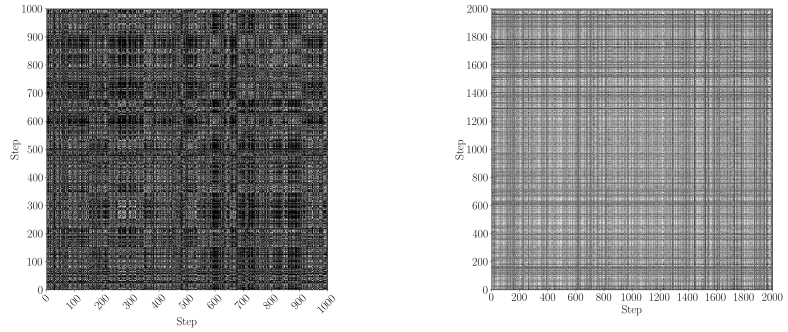


Figure 3.10.: Cumulative periodogram of the residuals of the ARIMA(3,1,5) model for a single pixel. The orange dashed line is the cumulative periodogram of the data, the blue dashed line is the cumulative periodogram of the residuals, and the red shaded area is the 95% confidence interval of the white noise.



(a) Recurrence plot of the data.

(b) Recurrence plot of the residuals.

Figure 3.11.: Recurrence plots of the data and the residuals of the ARIMA(3,1,5) model for a single pixel.

After these checks, the residuals are transformed into binary values using the method described in Subsection Random Number Generation. The binary results taken from different days of the same pixel are then concatenated to form a single binary string of size 3 Mb.

Diffusion Tests

As described in Subsection Diffusion Simulation, the diffusion tests were constructed to test the relative frequency of 0 and 1 in segments of the binary results[8] and the relative frequency of 00, 01, 10 and 11 in segments of the binary results[9]. In order to properly assess the statistics, the binary data were divided into segments (1) of size 1000, and 3000 Mean Squared Disploacements (MSDs) were computed from these segments for the one-dimensional test, and (2) of size 500, for the two-dimensional test. The results of the one-dimensional and two-dimensional tests can be seen in Figures 3.12 and 3.13, respectively.

8: for the one-dimensional test

9: for the two-dimensional test

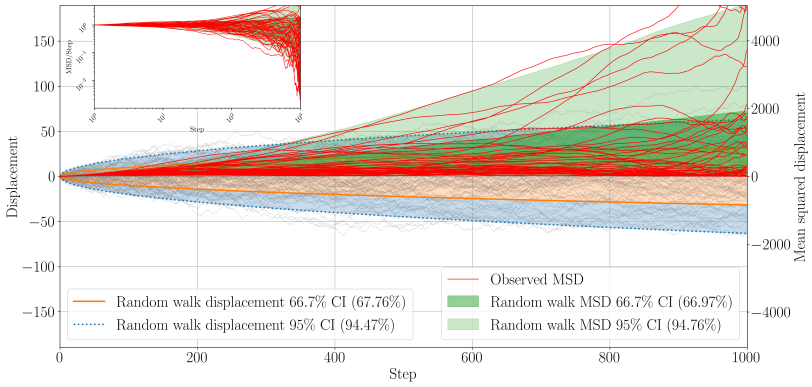


Figure 3.12.: Results of the one-dimensional diffusion test. The faint black lines are some simulated random walks, the orange area is the 2/3 confidence interval, the blue area is the 95% confidence interval (analytically computed). The red lines are some simulated MSDs, the green area is the 2/3 confidence interval, and the faint green area is the 95% confidence interval (numerically computed). Inside the parentheses of the legends, the actual percentage of simulations that lie inside the confidence intervals is shown. The inset shows the same MSD plots, along the x axis and with logarithmic scale to better visualize the behavior of the MSDs.

From these figures, it is evident that the diffusion tests are passed, as all the confidence intervals are consistent with the expected behaviors. This suggests that the binary data are indeed random.

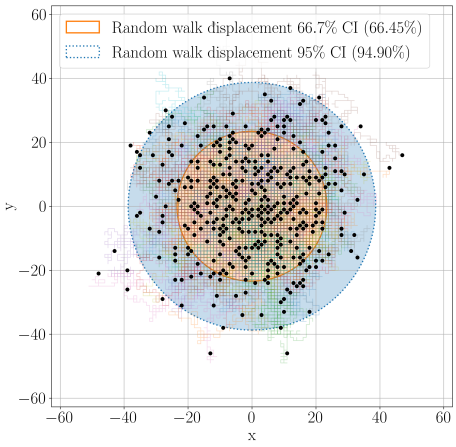


Figure 3.13.: Results of the two-dimensional diffusion test. The faint lines are some simulated random walks, the orange area is the 2/3 confidence interval, the blue area is the 95% confidence interval (analytically computed). Inside the parentheses of the legends, the actual percentage of simulations that lie inside the confidence intervals is shown.

NIST Tests

The binary data were also tested using the USA National Institute of Standards and Technology (NIST) tests, as described in Subsection NIST Test Suite. The results of these tests are summarized in Table 3.1. For the symbol convention, we use the one in the original documentation [12]: n is the number of bits in the binary data to be assessed at each repetition, m or M is the relevant parameter for the test, the p-value is proportional to the probability that the underlying p-values follow the expected distribution, and the number of passes is the number of times the p-value is larger than 0.01 in 100 repetitions. Some of these tests pose limitations on the sample size, which is why the results are either unavailable or limited for some tests. See the NIST documentation for more information on the tests and their limitations.

Table 3.1.: Results of the NIST tests. For details on the tests and their limitations, see the NIST documentation.

Test	n	m/M	p-value	Passes (Minimum)
Frequency	10000	-	0.05655	291/300 (291)
Block Frequency	10000	10	0.02408	293/300 (291)
Runs	10000	-	0.52744	295/300 (291)
Longest Run	10000	-	0.38383	296/300 (291)
Rank	38912	-	0.02075	76/77 (73)
Fourier	10000	-	0.00001	298/300 (291)
Non-overlapping	10^6	9	-	433/441 (427)
Overlapping	10^6	9	-	3/3 (2)
Universal	10^6	-	-	3/3 (2)
Linear Complexity	10^6	500	-	3/3 (2)
Serial	10000	2	0.48142, 0.19516	292, 296/300 (291)
Approx. Entropy	10000	2	0.21709	296/300 (291)
Cumulative Sums	10000	-	0.77276, 0.01265	293, 291/300 (291)

From Table 3.1, it is evident that the binary data pass the tests. The small sample size forbids the proper assessment of the random excursions tests.

3.4. Performance and Efficiency

Including Other Pixels

As described in Subsection Primary Data, the Digital Electromagnetic Calorimeter (DECAL) sensor works in *strip* mode during the measurements. This means that the data from all 64 pixels of a strip are read out simultaneously. In order to improve the rate of random bits, it would be beneficial to include the data from all pixels. However, as was briefly noted in Section Threshold Scan Mean Time Series, the time series of the different pixels of the same strip seem to be correlated. This is likely due to the fact that the pixels are read out simultaneously, while the sensor is subject to the same conditions.

The correlation between the time series of the different pixels of the same strip is visualized in Figure 3.14. From this figure, it is evident that the time series of the different pixels of the same strip are indeed correlated. This suggests that the data from the different pixels of the same strip should not be concatenated to form a single binary string, as this would significantly worsen the quality of the random bits. Instead, the data from the different pixels should be treated separately, as was done in the previous sections.

If the binary data from the different pixels of the same strip are concatenated, the diffusion tests are not passed, as can be seen in Figures 3.16 and 3.15.

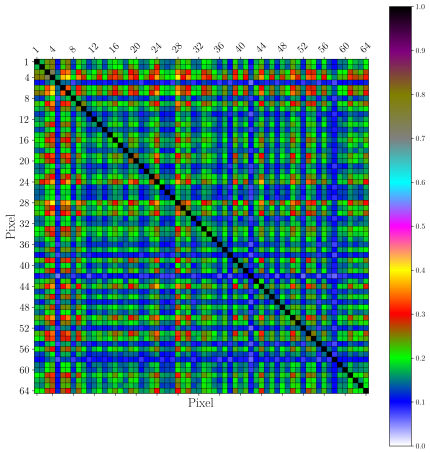


Figure 3.14.: Correlation matrix of the time series of the different pixels of the same strip running simultaneously.

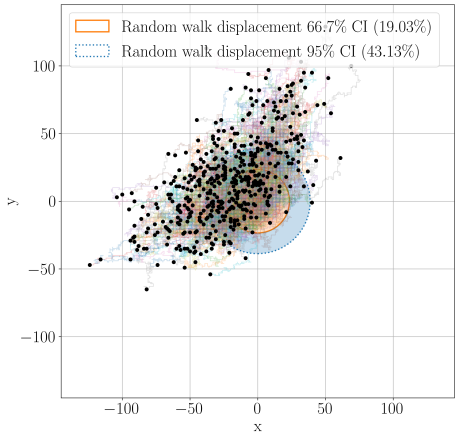


Figure 3.15.: Results of the two-dimensional diffusion test for the concatenated data of the different pixels of the same strip. The convention is the same as in Figure 3.13.

Efficiency

Asides from the quality of the random bits, the efficiency, as measured by the rate of high-quality bits generated, of a True Random Number Generator (TRNG) is also important. Existing TRNGs can generate random bits at rates higher than 10 Mbps [24, 59–62]. At the current DECAL setup, the significantly slowest step is the data acquisition, which takes around 310 min for 100000 scans.[10] This corresponds to a rate of 5.4 bps, which is significantly slower than the commercial TRNGs. However, the current setup is not optimized for speed, as it has been designed to test the DECAL as a calorimeter, and not for producing random bits. It is likely that the rate can be improved by optimizing the data acquisition process, either by redesigning the setup to be suitable for a TRNG, or by

10: The post-processing for this amount of data takes less than one minute.

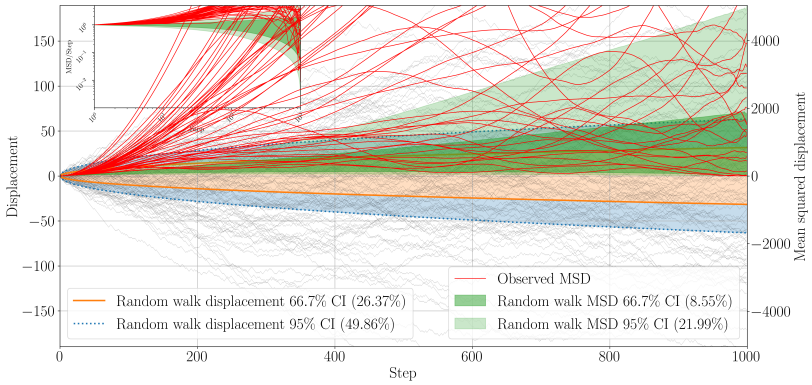


Figure 3.16.: Results of the one-dimensional diffusion test for the concatenated data of the different pixels of the same strip. The convention is the same as in Figure 3.12.

improving the firmware of the sensor. In this work, the focus has been on assessing the quality of the random bits, and the optimization of the rate has been left for future work. Some suggestions for improving the rate are given below:

- Take advantage of the fact that the sensor can read out all pixels of a strip simultaneously. This could potentially increase the rate by a factor of 64. In order to decorrelate the data from the different pixels, an additional step could be added to the post-processing, using either univariate or multivariate time series analysis, or other methods. Alternatively, the sensor or its firmware could be redesigned in a way that the data from the different pixels are decorrelated before being read out. For example, the pixels could be read with a small delay between them.
- Optimize the data acquisition process and setup. Currently, the user has to manually start the data acquisition process, through the ROOT-based ITk Strips Data Acquisition (ITSDAQ) software. The threshold scans are saved in CSV files, which are then read and processed by a Python script. This is a significantly slow process, as the writing and reading of the CSV files on a computer equipped with a hard drive takes a long time. In a final setup, the data acquisition and processing should be done on a single device using compiled code, without the need for intermediate files. At the same time, a lot of information is printed to the console during the data acquisition, which can be turned off in a future setup. This would significantly reduce the time needed for the data acquisition and processing.
- Avoid resetting the sensor after each measurement. Because of the voltage drift reported in previous studies [32, 48], the sensor is hard coded to reset after each measurement. However, this is a slow process, reported to take as much as 85% of the total measurement time [32]. If the drift is not significant, the sensor could be left running for a longer time, until the values fall outside the relevant range. This would significantly reduce the time needed for the data acquisition. In our case, the ARIMA model could take care of the drift by including a constant term.
- Reduce the number repetitions of the threshold scans and the range of the scans. As described in Subsection Primary Data, in this work the range is set from 1.10 V to 1.18 V, and the number of repetitions is set to 2000. This is done to ensure that the data is of the highest quality, to provide a good basis for the analysis. However, the range and the number of repetitions could be reduced to increase the rate of random bits. Specifically, from Figure 3.3, it is evident that the values are located between 1.156 and 1.166. Therefore, the chosen range is unnecessarily large. The number of repetitions could also be reduced, until the error in the calculation of the mean of the threshold distribution is acceptable. This would significantly reduce the time needed for the data acquisition.
- Optimize the post-processing. The post-processing of the data is currently done in Python,[11] with many checks. The post-processing could be done in a faster language, such as C++, or the Python code could be optimized. The post-processing could also be parallelized, as the data from the different pixels are independent.
- Use all the available information from the residuals. In principle, the

11: mainly using statsmodels and custom code

random numbers are not the bits, but the residuals of the ARIMA model. The residuals are then transformed into binary values, choosing 0 or 1 depending on their sign. However, the residuals contain more information than just their sign. For example, the magnitude of the residuals could be used to generate more random bits from each residual. This would increase the rate of random bits.

The combination of these suggestions could significantly increase the rate of random bits, and bring it closer to the rates of commercial TRNGs.

4. Conclusions

The feasibility of using the Digital Electromagnetic Calorimeter (DECAL) sensor as a True Random Number Generator (TRNG) was evaluated through a comprehensive analysis of its noise properties and the application of time series analysis. The key findings and implications of this research are summarized as follows:

- **Effective Noise Modeling**: The Autoregressive Integrated Moving Average (ARIMA) model was effectively fitted to the time series data obtained from the DECAL sensor. Diagnostic tests, including Q-Q plots, the Ljung-Box-Pierce test and spectral analysis, confirmed that the residuals of the model exhibited properties consistent with Gaussian white noise. This indicates that the ARIMA model is well-suited for characterizing the sensor's noise and transforming it into a format suitable for random number generation.
- **Generation and Validation of Random Bits**: By converting the residuals of the ARIMA model into binary sequences, 3 Mb of random numbers were successfully generated. The randomness of these sequences was rigorously tested using a variety of statistical tests, including the USA National Institute of Standards and Technology (NIST) test suite and novel diffusion tests. The sequences passed these tests, demonstrating high-quality randomness suitable for cryptographic and other security applications.
- **Diffusion Tests Confirmation**: The diffusion tests, which assess the relative frequency of binary pairs and the mean squared displacement of binary sequences, provided additional validation. The results from both one-dimensional and two-dimensional diffusion tests were consistent with the expected behavior of random sequences, further affirming the reliability of the generated random numbers.
- **Performance and Practicality**: The process of generating random numbers from the DECAL sensor data was found to exhibit a low rate of random bit generation, which may limit the sensor's practical application as a TRNG. However, the computational efficiency of the model fitting, residual analysis, and binary conversion processes supports the sensor's use in real-time or near-real-time systems.
- **Recommendations for Improvement**: To enhance the DECAL sensor's performance as a TRNG, future work should focus on optimizing the sensor's design and its communication with the data acquisition and post-processing systems. At this stage, the setup was a prototype, and further possible refinements should be considered to improve the rate of random bit generation and broaden the sensor's applicability as a TRNG.

Appendix

A. MLE of the Normal Distribution

A.1. Count Error Analysis

The first step is to determine the error of each binned count. As is common practice in event count statistics in High Energy Physics (HEP), the hits count of each bin is assumed to follow a Poisson distribution (Probability Mass and Density Functions)

For each bin, a specific count $k_{\text{obs},i}$ is recorded by the Digital Electromagnetic Calorimeter (DECAL) sensor. The true value of the count is $k_{\text{true},i} = \lambda_i$. Therefore, the error of the count is assumed to be equal to the confidence interval of the parameter λ_i. The confidence interval is defined as the range of values within which the true value of the parameter is expected to lie with a certain probability, $1 - \alpha$, given the observed data. In order to determine the confidence interval, the cumulative distribution function of the Poisson distribution is used. The cumulative distribution function of the Poisson distribution is given by:

$$F(k) \equiv P(X \leq k) = \sum_{i=0}^{k} \frac{\lambda^i e^{-\lambda}}{i!} = 1 - F_{\chi^2}\left(2\lambda; 2(k+1)\right) \tag{A.1}$$

where F_{χ^2} is the cumulative distribution function of the chi-squared distribution, which is given by:

$$F_{\chi^2}(x;k) = \frac{\gamma\left(\frac{k}{2}, \frac{x}{2}\right)}{\Gamma\left(\frac{k}{2}\right)} = \frac{1}{\Gamma\left(\frac{k}{2}\right)} \int_0^x t^{\frac{k}{2}-1} e^{-\frac{t}{2}} dt \tag{A.2}$$

where γ is the lower incomplete gamma function and Γ is the gamma function. Assuming central confidence intervals (i.e. equal probability on both sides of the interval), the confidence interval of the parameter λ_i is given by:

$$\begin{aligned} \lambda_{\text{low},i} &= \frac{1}{2} F^{-1}_{\chi^2}\left(1 - \frac{\alpha}{2}; 2(k_{\text{obs},i} + 1)\right) \\ \lambda_{\text{high},i} &= \frac{1}{2} F^{-1}_{\chi^2}\left(\frac{\alpha}{2}; 2(k_{\text{obs},i} + 1)\right) \end{aligned} \tag{A.3}$$

All of these can also be seen in Figure A.1. This interval has a probability of $1 - \alpha$ to contain the true value of the parameter λ_i. It is worth noting that for very small values of λ_i, the confidence continues to give central intervals, and therefore the lowest possible value of λ_i is never zero. In order to account for noise and avoid a phenomenon which in

HEP is called "flip-flopping", the procedure described in [63] must be followed. In this case, the signals we are dealing with are produced by the noise, and therefore there is no need to account for this phenomenon. However, if at some point the DECAL sensor is used to generate numbers from a source, this methodology must be revised. As far as efficiency is concerned, for large number of events, the Poisson distribution can be approximated by a Gaussian distribution with mean λ_i and standard deviation $\sqrt{\lambda_i}$. Even though this would save computational time, at this point the analysis is performed using the Poisson distribution, in order to avoid any approximation. For this report, the observed counts are used as the most probable true values of the parameters λ_i. The error of each bin is then calculated as the difference between the observed value and the edges of the Poisson distribution confidence interval with $1 - \alpha = 0.68$.

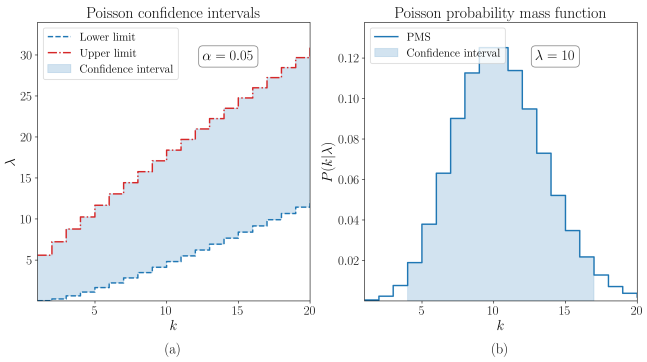


Figure A.1.: Poisson distribution confidence intervals for different values of λ (a) and the interval for $\lambda = 25$ (b). In both cases, $\alpha = 0.05$.

A.2. Gaussian Fit of the Histograms

According to previous work on the DECAL sensor, these histograms are expected to be normally distributed [47]. Therefore, the next step is to fit the histograms in order to determine the mean and the standard deviation of each one of them. The histograms are fitted with a Gaussian distribution. The probability density function of the Gaussian distribution is given by Equation **??**. The Gaussian distribution is defined by two parameters, the mean μ and the standard deviation σ. In order to fit the histograms, the Binned Maximum Likelihood Estimation (BMLE) method is used (see Subsection **??**). This method assumes that every bin is independent of the others and that the probability of a count k_i in a bin is given by the Poisson distribution. The efficiency can be increased by approximating the Poisson distribution with a Gaussian distribution. However, in this case the Poisson distribution is used, in order to avoid any approximation. Specifically, the likelihood function is given by the product of the probability of each bin, as given in Equation (**??**). For each bin, the expected count λ_i of the Gaussian distribution is given by:

$$\lambda_i = \int_{x_i}^{x_{i+1}} f(x)dx = \frac{N}{2}\left[\text{erf}\left(\frac{x_{i+1} - \mu}{\sqrt{2}\sigma}\right) - \text{erf}\left(\frac{x_i - \mu}{\sqrt{2}\sigma}\right)\right] \tag{A.4}$$

where erf is the error function and N the total hit count. The partial

derivatives of λ_i with respect to μ and σ are given by:

$$\begin{aligned}\frac{\partial \lambda_i}{\partial \mu} &= -\frac{N}{\sqrt{2\pi}\sigma}\left[e^{-\frac{(x_{i+1}-\mu)^2}{2\sigma^2}} - e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}\right] \\ \frac{\partial \lambda_i}{\partial \sigma} &= -\frac{N}{\sqrt{2\pi}\sigma^2}\left[(x_{i+1}-\mu)e^{-\frac{(x_{i+1}-\mu)^2}{2\sigma^2}} - (x_i-\mu)e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}\right]\end{aligned} \tag{A.5}$$

The second partial derivatives of λ_i with respect to μ and σ are given by:

$$\begin{aligned}\frac{\partial^2 \lambda_i}{\partial \mu^2} &= -\frac{N}{\sqrt{2\pi}\sigma^3}\left[(x_{i+1}-\mu)e^{-\frac{(x_{i+1}-\mu)^2}{2\sigma^2}} - (x_i-\mu)e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}\right] \\ \frac{\partial^2 \lambda_i}{\partial \sigma^2} &= -\frac{N}{\sqrt{2\pi}\sigma^5}\left\{\left[(x_{i+1}-\mu)^2 - 2\sigma^2\right](x_{i+1}-\mu)e^{-\frac{(x_{i+1}-\mu)^2}{2\sigma^2}} - \right. \\ &\quad \left.\left[(x_i-\mu)^2 - 2\sigma^2\right](x_i-\mu)e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}\right\} \\ \frac{\partial^2 \lambda_i}{\partial \mu \partial \sigma} &= -\frac{N}{\sqrt{2\pi}\sigma^4}\left[(\mu^2 + x_{i+1}^2 - 2x_{i+1}\mu - \sigma^2)e^{-\frac{(x_{i+1}-\mu)^2}{2\sigma^2}}\right. \\ &\quad \left.-(\mu^2 + x_i^2 - 2x_i\mu - \sigma^2)e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}\right]\end{aligned} \tag{A.6}$$

Maximizing the likelihood function is equivalent to maximizing the logarithm of the likelihood function (Equation 1.15).

The partial derivatives of the logarithm of the likelihood function with respect to μ and σ are given by:

$$\begin{aligned}\frac{\partial \ln L(\mu,\sigma)}{\partial \mu} &= \sum_{i=1}^{n}\left[\left(\frac{k_i}{\lambda_i} - 1\right)\frac{\partial \lambda_i}{\partial \mu}\right] \\ \frac{\partial \ln L(\mu,\sigma)}{\partial \sigma} &= \sum_{i=1}^{n}\left[\left(\frac{k_i}{\lambda_i} - 1\right)\frac{\partial \lambda_i}{\partial \sigma}\right]\end{aligned} \tag{A.7}$$

The second partial derivatives of the logarithm of the likelihood function with respect to μ and σ are given by:

$$\begin{aligned}\frac{\partial^2 \ln L(\mu,\sigma)}{\partial \mu^2} &= \sum_{i=1}^{n}\left[\left(\frac{k_i}{\lambda_i} - 1\right)\frac{\partial^2 \lambda_i}{\partial \mu^2} - \frac{k_i}{\lambda_i^2}\left(\frac{\partial \lambda_i}{\partial \mu}\right)^2\right] \\ \frac{\partial^2 \ln L(\mu,\sigma)}{\partial \sigma^2} &= \sum_{i=1}^{n}\left[\left(\frac{k_i}{\lambda_i} - 1\right)\frac{\partial^2 \lambda_i}{\partial \sigma^2} - \frac{k_i}{\lambda_i^2}\left(\frac{\partial \lambda_i}{\partial \sigma}\right)^2\right] \\ \frac{\partial^2 \ln L(\mu,\sigma)}{\partial \mu \partial \sigma} &= \sum_{i=1}^{n}\left[\left(\frac{k_i}{\lambda_i} - 1\right)\frac{\partial^2 \lambda_i}{\partial \mu \partial \sigma} - \frac{k_i}{\lambda_i^2}\frac{\partial \lambda_i}{\partial \mu}\frac{\partial \lambda_i}{\partial \sigma}\right]\end{aligned} \tag{A.8}$$

The error of the parameters μ and σ are assumed to be one standard deviation from the maximum likelihood value. Therefore, the errors of the parameters μ and σ are given by the contour that is defined by the equation:

$$\ln L(\mu,\sigma) = \ln L_{\max} - \frac{1}{2} \tag{A.9}$$

where $L_{\max}$ is the maximum value of the likelihood function. The logarithm of the likelihood function can be approximated by a quadratic function around the maximum likelihood value. Therefore, equation A.9 can be approximated by:

$$\mathrm{H}_{1,1}\left(\mu-\mu_{\max}\right)^2+2\mathrm{H}_{1,2}\left(\mu-\mu_{\max}\right)\left(\sigma-\sigma_{\max}\right)+\mathrm{H}_{2,2}\left(\sigma-\sigma_{\max}\right)^2=0 \tag{A.10}$$

where $\mathrm{H}_{i,j}$ is the i,j-th element of the Hessian matrix, which is given by:

$$\mathrm{H}=\begin{bmatrix} -\left.\dfrac{\partial^2 \ln L(\mu,\sigma)}{\partial\mu^2}\right|_{\mu=\mu_{\max},\sigma=\sigma_{\max}} & -\left.\dfrac{\partial^2 \ln L(\mu,\sigma)}{\partial\mu\partial\sigma}\right|_{\mu=\mu_{\max},\sigma=\sigma_{\max}} \\ -\left.\dfrac{\partial^2 \ln L(\mu,\sigma)}{\partial\mu\partial\sigma}\right|_{\mu=\mu_{\max},\sigma=\sigma_{\max}} & -\left.\dfrac{\partial^2 \ln L(\mu,\sigma)}{\partial\sigma^2}\right|_{\mu=\mu_{\max},\sigma=\sigma_{\max}} \end{bmatrix} \tag{A.11}$$

Therefore, for a given histogram (i.e. sets of bins x_i and counts k_i), the maximum likelihood value of the parameters μ and σ are found by the computational solution of the system of equations A.7 and the errors of the parameters μ and σ by the solution of equation A.10. The p − value of the Pearson's chi-squared test is also calculated.

This procedure is applied to all the histograms generated by the DECAL sensor. An example can be seen in Figure A.2.

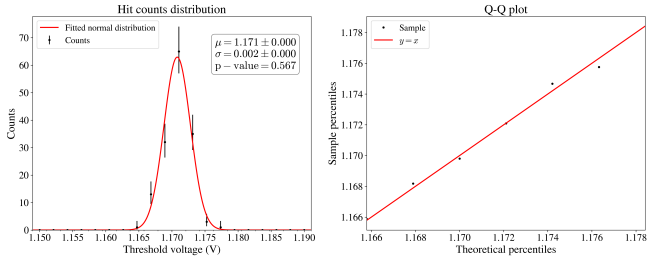


Figure A.2.: Example of a Gaussian fit of a threshold voltage scan histogram.

B.

ARIMA Analysis of Another Pixel

In the main text, the Autoregressive Integrated Moving Average (ARIMA) analysis of the pixel located at column/strip 31 and row 39 was presented. In order to show that the results are similar for other pixels as well, the ARIMA analysis of the pixel located at column/strip 31 and row 51 is presented here.[1] These measurements were taken during a different day than the ones in the main text. The results are shown in Figures B.1, B.2, B.3, and B.4. It can be seen that the analysis is consistent with the one presented in the main text.

1: Which was chosen arbitrarily.

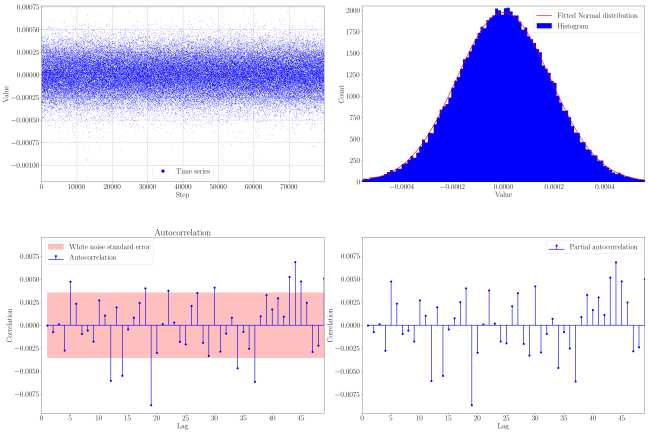


Figure B.1.: Residuals of the ARIMA analysis of the pixel located at column/strip 31 and row 51.

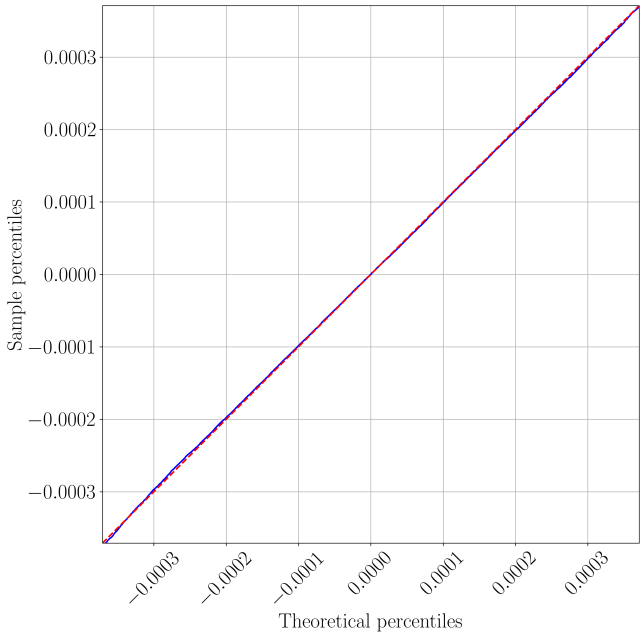


Figure B.2.: Q-Q plot of the residuals of the ARIMA analysis of the pixel located at column/strip 31 and row 51, assuming a normal distribution.

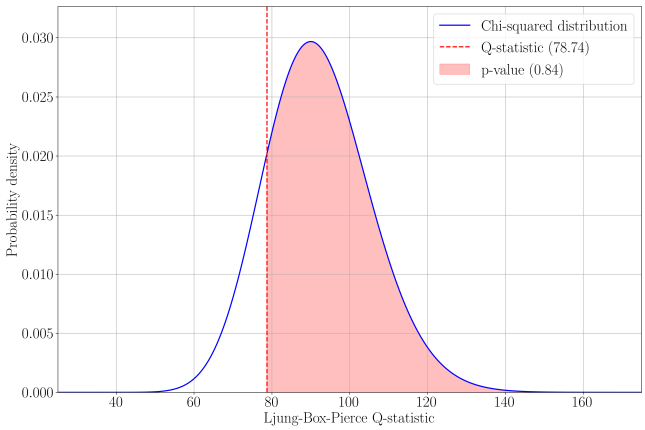


Figure B.3.: Ljung-Box-Pierce Q-statistic of the residuals of the ARIMA analysis of the pixel located at column/strip 31 and row 51.

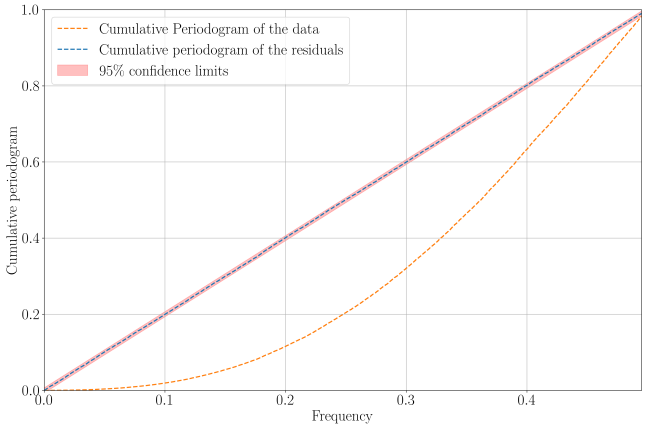


Figure B.4.: Cumulative periodogram of the residuals of the ARIMA analysis of the pixel located at column/strip 31 and row 51.

Bibliography

Here are the references in citation order.

[1] David P. Landau and K. Binder. *A guide to Monte Carlo simulations in statistical physics*. Fourth edition. Cambridge, United Kingdom: Cambridge University Press, 2015. 519 pp. (cited on page 1).

[2] Mark E. J. Newman and Gerard Tonnis Barkema. *Monte Carlo methods in statistical physics*. Oxforg: Clarendon press, 1999 (cited on page 1).

[3] James E. Gentle. *Random number generation and Monte Carlo methods*. 2. edition, corrected second printing. Statistics and computing. New York: Springer, 2005. 381 pp. (cited on page 1).

[4] Donald Ervin Knuth. *The art of computer programming. Volume 2: Seminumerical algorithms*. Third edition. Vol. 2. Boston: Addison-Wesley, 2021. 764 pp. (cited on pages 1, 2).

[5] A. J. Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. *Handbook of applied cryptography*. CRC Press series on discrete mathematics and its applications. Boca Raton: CRC Press, 1997. 780 pp. (cited on page 1).

[6] Pierre L'Ecuyer. "History of uniform random number generation." In: *2017 Winter Simulation Conference (WSC)*. 2017 Winter Simulation Conference (WSC). Las Vegas, NV: IEEE, Dec. 2017, pp. 202–230. DOI: 10.1109/WSC.2017.8247790. (Visited on 01/08/2024) (cited on pages 1, 2).

[7] M. G. Kendall and B. Babington-Smith. "Second Paper on Random Sampling Numbers." In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 6.1 (Jan. 1, 1939), pp. 51–61. DOI: 10.2307/2983623. (Visited on 06/24/2024) (cited on pages 1, 2).

[8] Luc Devroye. *Non-Uniform Random Variate Generation*. New York, NY: Springer New York, 1986. (Visited on 06/24/2024) (cited on page 1).

[9] M. G. Kendall and B. Babington Smith. "Randomness and Random Sampling Numbers." In: *Journal of the Royal Statistical Society* 101.1 (1938), p. 147. DOI: 10.2307/2980655. (Visited on 06/24/2024) (cited on page 1).

[10] George Marsaglia. *The Marsaglia Random Number CDROM including the Diehard Battery of Tests of Randomness*. 1995 (cited on page 1).

[11] Robert G. Brown. *Dieharder: A Random Number Test Suite*. Version 3.31.1. 2024 (cited on page 1).

[12] Andrew Rukhin et al. "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications." In: *NIST Special Publication 800-22* (2010) (cited on pages 2, 28, 36).

[13] Alexander Shen. "Randomness Tests: Theory and Practice." In: *Fields of Logic and Computation III*. Ed. by Andreas Blass et al. Vol. 12180. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 258–290. DOI: 10.1007/978-3-030-48006-6_18. (Visited on 06/24/2024) (cited on page 2).

[14] N. C. Metropolis, G. Reitwiesner, and J. Von Neumann. "Statistical treatment of values of first 2,000 decimal digits of e and of π calculated on the ENIAC." In: *Mathematics of Computation* 4.30 (1950), pp. 109–111. DOI: 10.1090/S0025-5718-1950-0037598-8. (Visited on 06/24/2024) (cited on page 2).

[15] J. Von Neumann. "Various techniques used in connection with random digits, notes by G E Forsythe." In: *National Bureau of Standards Applied Math Series*. National Bureau of Standards Applied Math Series 12 (1951), pp. 36–38 (cited on pages 2, 3).

[16] Derrick H. Lehmer. "Mathematical Methods in Large-scale Computing Units." In: *Proceedings of the Second Symposium on Large Scale Digital Computing Machinery*. Symposium on Large Scale Digital Computing Machinery. Harvard University: Harvard University Press, 1951, pp. 141–146 (cited on page 2).

[17] Makoto Matsumoto and Takuji Nishimura. “Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator.” In: *ACM Transactions on Modeling and Computer Simulation* 8.1 (Jan. 1998), pp. 3–30. DOI: 10.1145/272991.272995. (Visited on 01/08/2024) (cited on page 2).

[18] George Marsaglia. “Xorshift RNGs.” In: *Journal of Statistical Software* 8.14 (2003). DOI: 10.18637/jss.v008.i14. (Visited on 06/24/2024) (cited on page 2).

[19] G.K Savvidy and N.G Ter-Arutyunyan-Savvidy. “On the Monte Carlo simulation of physical systems.” In: *Journal of Computational Physics* 97.2 (Dec. 1991), pp. 566–572. DOI: 10.1016/0021-9991(91)90015-D. (Visited on 06/24/2024) (cited on page 2).

[20] Frederick James and Lorenzo Moneta. “Review of High-Quality Random Number Generators.” In: *Computing and Software for Big Science* 4.1 (Dec. 2020), p. 2. DOI: 10.1007/s41781-019-0034-3. (Visited on 01/08/2024) (cited on page 2).

[21] J.D. Cobine and J.R. Curry. “Electrical Noise Generators.” In: *Proceedings of the IRE* 35.9 (Sept. 1947), pp. 875–879. DOI: 10.1109/JRPROC.1947.229646. (Visited on 06/25/2024) (cited on page 2).

[22] W. E. Thomson. “ERNIE - A Mathematical and Statistical Analysis.” In: *Journal of the Royal Statistical Society. Series A (General)* 122.3 (1959), p. 301. DOI: 10.2307/2342795. (Visited on 06/25/2024) (cited on page 2).

[23] Yuval Peres. “Iterating Von Neumann’s Procedure for Extracting Random Bits.” In: *The Annals of Statistics* 20.1 (Mar. 1, 1992). DOI: 10.1214/aos/1176348543. (Visited on 06/25/2024) (cited on page 3).

[24] Mario Stipčević and Çetin Kaya Koç. “True Random Number Generators.” In: *Open Problems in Mathematics and Computational Science*. Ed. by Çetin Kaya Koç. Cham: Springer International Publishing, 2014, pp. 275–315. DOI: 10.1007/978-3-319-10683-0_12. (Visited on 06/24/2024) (cited on pages 3, 37).

[25] Miguel Herrero-Collantes and Juan Carlos Garcia-Escartin. “Quantum random number generators.” In: *Reviews of Modern Physics* 89.1 (Feb. 22, 2017), p. 015004. DOI: 10.1103/RevModPhys.89.015004. (Visited on 06/24/2024) (cited on page 3).

[26] Xiongfeng Ma et al. “Quantum random number generation.” In: *npj Quantum Information* 2.1 (June 28, 2016), p. 16021. DOI: 10.1038/npjqi.2016.21. (Visited on 01/08/2024) (cited on page 3).

[27] Christian W. Fabjan and Fabiola Gianotti. “Calorimetry for particle physics.” In: *Reviews of Modern Physics* 75.4 (Oct. 15, 2003), pp. 1243–1286. DOI: 10.1103/RevModPhys.75.1243. (Visited on 10/18/2023) (cited on pages 4, 5).

[28] Claude Leroy and Pier-Giorgio Rancoita. *Principles of radiation interaction in matter and detection*. 2nd ed. Singapore: World Scientific, 2009 (cited on pages 4, 5).

[29] Particle Data Group et al. “Review of Particle Physics.” In: *Progress of Theoretical and Experimental Physics* 2022.8 (Aug. 8, 2022), p. 083C01. DOI: 10.1093/ptep/ptac097. (Visited on 10/24/2023) (cited on pages 4, 5).

[30] Werner Nakel. “The elementary process of bremsstrahlung.” In: *Physics Reports* 243.6 (July 1994), pp. 317–353. DOI: 10.1016/0370-1573(94)00068-9. (Visited on 10/23/2023) (cited on page 4).

[31] Francesca Cavallari. “Performance of calorimeters at the LHC.” In: *Journal of Physics: Conference Series* 293 (Apr. 1, 2011), p. 012001. DOI: 10.1088/1742-6596/293/1/012001. (Visited on 10/23/2023) (cited on page 5).

[32] Lucian Fasselt. “Characterization of the DECAL sensor - a CMOS MAPS prototype for digital electromagnetic calorimetry and tracking.” MSc Thesis. Berlin: Humboldt U. Berlin, Jan. 23, 2023. 77 pp. (cited on pages 5, 6, 21, 23, 31, 38).

[33] Erika Garutti. “Overview on calorimetry.” In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 628.1 (Feb. 2011), pp. 31–39. DOI: 10.1016/j.nima.2010.06.281. (Visited on 10/18/2023) (cited on page 5).

[34] Philip Patrick Allport et al. “DECAL: A Reconfigurable Monolithic Active Pixel Sensor for Tracking and Calorimetry in a 180 nm Image Sensor Process.” In: *Sensors* 22.18 (Sept. 10, 2022), p. 6848. DOI: 10.3390/s22186848. (Visited on 09/24/2023) (cited on pages 5, 6, 19–21).

[35] P.P. Allport et al. “First tests of a reconfigurable depleted MAPS sensor for digital electromagnetic calorimetry.” In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 958 (Apr. 2020), p. 162654. DOI: 10.1016/j.nima.2019.162654. (Visited on 09/24/2023) (cited on pages 5, 6).

[36] J. Ocariz. “Probability and Statistics for Particle Physicists.” In: (2014). Publisher: arXiv Version Number: 1. DOI: 10.48550/ARXIV.1405.3402. (Visited on 11/07/2023) (cited on page 7).

[37] Roger John Barlow. “Practical Statistics for Particle Physics.” In: *CERN Yellow Reports: School Proceedings* Vol. 5 (Sept. 19, 2020), 149 Pages. DOI: 10.23730/CYRSP-2020-005.149. (Visited on 11/07/2023) (cited on page 7).

[38] George E. P. Box et al. *Time series analysis: forecasting and control*. Fifth edition. Wiley series in probability and statistics. Hoboken, New Jersey: John Wiley & Sons, Inc, 2016. 669 pp. (cited on pages 10, 13–17, 24, 26, 31–33).

[39] James D. Hamilton. *Time series analysis*. Princeton, N.J: Princeton University Press, 1994. 799 pp. (cited on pages 10, 12, 14).

[40] Holger Kantz and Thomas Schreiber. *Nonlinear Time Series Analysis*. 2nd ed. Cambridge University Press, Nov. 27, 2003. (Visited on 06/02/2024) (cited on pages 10, 18).

[41] Geoffrey Grimmett, David Stirzaker, and David Stirzaker. *Probability and random processes*. 3. ed., repr. with corr., [Nachdr.] Oxford: Oxford Univ. Press, 2009. 596 pp. (cited on page 10).

[42] J. Durbin. “The Fitting of Time-Series Models.” In: *Revue de l’Institut International de Statistique / Review of the International Statistical Institute* 28.3 (1960), p. 233. DOI: 10.2307/1401322. (Visited on 06/08/2024) (cited on page 12).

[43] G. U. Yule. “VII. On a method of investigating periodicities disturbed series, with special reference to Wolfer’s sunspot numbers.” In: *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* 226.636 (Jan. 1927), pp. 267–298. DOI: 10.1098/rsta.1927.0007. (Visited on 06/06/2024) (cited on page 14).

[44] G. Walker. “On periodicity in series of related terms.” In: *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character* A131.818 (June 3, 1931), pp. 518–532. DOI: 10.1098/rspa.1931.0069. (Visited on 06/06/2024) (cited on page 14).

[45] Walter Enders. *Applied econometric time series*. Fourth edition. Hoboken, NJ: Wiley, 2015. 485 pp. (cited on page 14).

[46] Robert H. Shumway and David S. Stoffer. *Time Series Analysis and Its Applications: With R Examples*. Springer Texts in Statistics. Cham: Springer International Publishing, 2017. (Visited on 06/08/2024) (cited on page 15).

[47] I. Kopsalis et al. “Evaluation of the DECAL Fully Depleted monolithic sensor for outer tracking and digital calorimetry.” In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 1038 (Sept. 2022), p. 166955. DOI: 10.1016/j.nima.2022.166955. (Visited on 09/24/2023) (cited on pages 19, 23, 43).

[48] Lucian Fasselt et al. “Energy calibration through X-ray absorption of the DECAL sensor, a monolithic active pixel sensor prototype for digital electromagnetic calorimetry and tracking.” In: *Frontiers in Physics* 11 (Oct. 17, 2023), p. 1231336. DOI: 10.3389/fphy.2023.1231336. (Visited on 10/18/2023) (cited on pages 19, 23, 38).

[49] ATLAS Collaboration. *Technical Design Report for the ATLAS Inner Tracker Strip Detector*. Artwork Size: pages 556 Publication Title: 556 pp. (2017). Deutsches Elektronen-Synchrotron, DESY, Hamburg, 2017, pages 556. DOI: 10.3204/PUBDB-2017-09975. (Visited on 10/25/2023) (cited on page 22).

[50] Peter J. Brockwell and Richard A. Davis. *Introduction to Time Series and Forecasting*. Springer Texts in Statistics. Cham: Springer International Publishing, 2016. (Visited on 06/18/2024) (cited on page 24).

[51] Skipper Seabold and Josef Perktold. “Statsmodels: Econometric and Statistical Modeling with Python.” In: Python in Science Conference. Austin, Texas, 2010, pp. 92–96. DOI: 10.25080/Majora-92bf1922-011. (Visited on 06/18/2024) (cited on page 24).

[52] H. Akaike. “A new look at the statistical model identification.” In: *IEEE Transactions on Automatic Control* 19.6 (Dec. 1974), pp. 716–723. DOI: 10.1109/TAC.1974.1100705. (Visited on 06/18/2024) (cited on page 24).

[53] Gideon Schwarz. “Estimating the Dimension of a Model.” In: *The Annals of Statistics* 6.2 (Mar. 1, 1978). DOI: 10.1214/aos/1176344136. (Visited on 06/18/2024) (cited on page 24).

[54] G. E. P. Box and David A. Pierce. “Distribution of Residual Autocorrelations in Autoregressive-Integrated Moving Average Time Series Models.” In: *Journal of the American Statistical Association* 65.332 (Dec. 1970), pp. 1509–1526. DOI: 10.1080/01621459.1970.10481180. (Visited on 02/22/2024) (cited on page 26).

[55] G. M. Ljung and G. E. P. Box. “On a measure of lack of fit in time series models.” In: *Biometrika* 65.2 (Aug. 1, 1978), pp. 297–303. DOI: 10.1093/biomet/65.2.297. (Visited on 06/23/2024) (cited on page 26).

[56] Bedartha Goswami. “A Brief Introduction to Nonlinear Time Series Analysis and Recurrence Plots.” In: *Vibration* 2.4 (Dec. 8, 2019), pp. 332–368. DOI: 10.3390/vibration2040021. (Visited on 05/08/2024) (cited on page 27).

[57] V. Calandrini et al. “nMoldyn - Interfacing spectroscopic experiments, molecular dynamics simulations and models for time correlation functions.” In: *École thématique de la Société Française de la Neutronique* 12 (2011), pp. 201–232. DOI: 10.1051/sfn/201112010. (Visited on 02/20/2024) (cited on page 28).

[58] Rayleigh. “The Problem of the Random Walk.” In: *Nature* 72.1866 (Aug. 3, 1905), pp. 318–318. DOI: 10.1038/072318a0. (Visited on 06/23/2024) (cited on page 28).

[59] *Quantis Quantum Random Number Generator (QRNG) PCIe - IDQ*. URL: https://www.idquantique.com/random-number-generation/products/quantis-qrng-pcie/ (visited on 07/02/2024) (cited on page 37).

[60] M. Bucci et al. “A high-speed oscillator-based truly random number source for cryptographic applications on a smartcard IC.” In: *IEEE Transactions on Computers* 52.4 (Apr. 2003), pp. 403–409. DOI: 10.1109/TC.2003.1190581. (Visited on 07/02/2024) (cited on page 37).

[61] Kyungduk Kim et al. “Massively parallel ultrafast random bit generation with a chip-scale laser.” In: *Science* 371.6532 (Feb. 26, 2021), pp. 948–952. DOI: 10.1126/science.abc2666. (Visited on 07/03/2024) (cited on page 37).

[62] Xiufeng Xu and Yuyang Wang. “High Speed True Random Number Generator Based on FPGA.” In: *2016 International Conference on Information Systems Engineering (ICISE)*. 2016 International Conference on Information Systems Engineering (ICISE). Los Angeles, CA, USA: IEEE, Apr. 2016, pp. 18–21. DOI: 10.1109/ICISE.2016.14. (Visited on 07/03/2024) (cited on page 37).

[63] Gary J. Feldman and Robert D. Cousins. “Unified approach to the classical statistical analysis of small signals.” In: *Physical Review D* 57.7 (Apr. 1, 1998), pp. 3873–3889. DOI: 10.1103/PhysRevD.57.3873. (Visited on 11/07/2023) (cited on page 43).