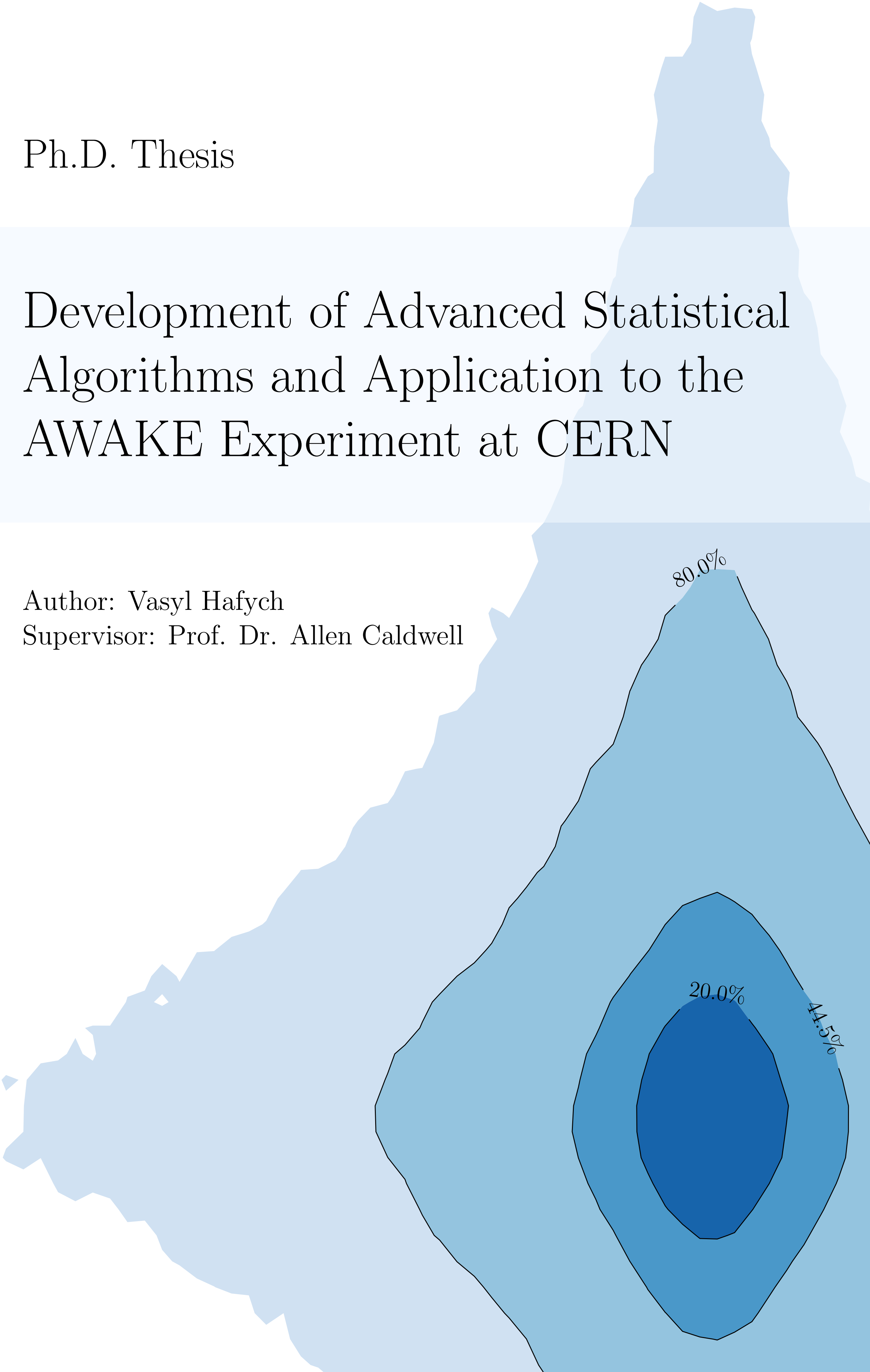
Ph.D. Thesis

Development of Advanced Statistical Algorithms and Application to the AWAKE Experiment at CERN

Author: Vasyl Hafych
Supervisor: Prof. Dr. Allen Caldwell



TECHNISCHE UNIVERSITÄT MÜNCHEN
Fakultät für Physik

Development of Advanced Statistical Algorithms and Application to the AWAKE Experiment at CERN

Vasyl Hafych

Vollständiger Abdruck der von der Fakultät für Physik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzende:

Prof. Dr. Katharina Krischer

Prüfer der Dissertation:

1. Hon.-Prof. Dr. Allen C. Caldwell
2. Hon.-Prof. Dr. Frank Jenko

Die Dissertation wurde am 27.10.2021 bei der Technischen Universität München eingereicht und durch die Fakultät für Physik am 18.01.2022 angenommen.

Abstract

Analysis of data with the help of statistical methods is a key element of modern scientific research. A powerful framework for such analyses is Bayesian inference, which allows testing models and updating information about their parameters in an easy-to-interpret manner. The developments presented in this thesis contribute to the field of Bayesian computations and show a practical application of Bayesian inference to a problem formulated by the Advanced Wakefield Experiment (AWAKE) at CERN.

The first part of this thesis presents two algorithms that improve Bayesian numerical methods. The first algorithm, called Integration with an Adaptive Harmonic Mean (AHMI), allows the estimation of the integral of the target density function — often denoted as the Bayesian evidence — using samples drawn from the target density. The second algorithm presents an approach to parallelize and improve Markov Chain Monte Carlo sampling of complex target densities using space partitioning. These two algorithms are implemented in the Bayesian Analysis Toolkit software package, whose functionalities are discussed. Both algorithms are tested on multiple benchmark densities and show a reliable performance on problems with up to 20 dimensions.

The second part of this thesis presents a detailed statistical analysis of the incoming proton bunch parameters used in the AWAKE experiment. Our analysis combines the data from multiple beam imaging systems located at different positions along the beamline to reconstruct the envelop trajectory of the bunch profile. The parameters of two different models are fitted to represent the observed data using Bayesian inference. The analysis is tested on simulated data and then applied to the experimental data. Multiple advanced statistical algorithms are used to compare the accuracy of the fits and to post-process posterior distributions, including the AHMI algorithm, a two-sample classifier test, and Hamiltonian Monte Carlo sampling. The self-modulation of the proton bunches in the plasma is studied using numerical plasma modeling with the measured proton bunch parameters as input.

Acknowledgements

First of all, I am immensely grateful to my supervisor Prof. Dr. Allen Caldwell, for his excellent guidance during the past three years of my work as a doctoral student at the Max-Planck Institute for Physics in Munich. His great support, readiness to help, enthusiastic encouragement showed me the academic world at its best. His efforts have been very much appreciated.

Secondly, I would like to extend my thanks to Dr. Philipp Eller for sharing his expertise and help. I am especially thankful to him for carefully reading this thesis and helping to make it more clear and understandable.

I am also thankful to Dr. Oliver Schultz for helping me with numerous large and small problems on the computing and other aspects of my projects, and I am especially thankful for introducing me to the Julia programming language.

I would like to thank Dr. John Farmer for performing plasma modeling that is described in the last section of this thesis.

I would also like to extend my thanks to my colleagues from BAT, AWAKE, INSIGHTS, ODSL, and C2PAP teams for teaching me many interesting things about physics and data science and helping me to understand what I want to do next in my career.

I must express my very profound gratitude to Prof. Dr. Glen Cowan for coordinating this amazing INSIGHTS ITN program. My special gratitude to the European Union's Framework Programme for Research and Innovation Horizon 2020 (2014-2020) under the Marie Sklodowska-Curie (Grant Agreement No.765710) for providing me with the fundings that made this all possible.

Зрештою, успіх — це не завжди результат наполегливих старань та роботи. Це також результат віри, підтримки — та головне — мотивації від всх тих людей, які є для нас важливими. Я дякую своїм батькам, братові, дівчині, та всім тим, хто вірив не так сильно в результат моєї роботи, як в мене самого. Ваша заслуга у моїх успіхах завжди була та завжди буде визначальною.

Publications

Included in the thesis:

[1] A. Caldwell, P. Eller, V. Hafych, R. Schick, O. Schulz, M. Szalay. Integration with an adaptive harmonic mean algorithm. *International Journal of Modern Physics A*, 35 (24), 2050142

[2] V. Hafych, P. Eller, O. Schulz, A. Caldwell. Parallelizing MCMC Sampling via Space Partitioning. Submitted to: *Springer — Statistics and Computing*, *arXiv preprint:* 2008.03098

[3] O. Schulz, F. Beaujean, A. Caldwell, C. Grunwald, V. Hafych, K. Kröninger, S. La Cagnina, L. Röhrig, L. Shtembari. BAT. jl: A Julia-Based Tool for Bayesian Inference. *SN Computer Science 2*, (3), 1-17

[4] V. Hafych, A. Caldwell, et al. (AWAKE Collaboration). Analysis of Proton Bunch Parameters in the AWAKE Experiment. *JINST Journal of Instrumentation*, 16.11 (2021): P11031

Not included in the thesis:

[1] A. Caldwell, V. Hafych, O. Schulz, L. Shtembari. Infections and Identified Cases of COVID-19 from Random Testing Data. *arXiv preprint:* 2005.11277

Contents

Abstract i

Publications v

Motivation and Overview 1

1 Introduction to Bayesian Inference 3

1.1 Bayes’ Theorem . . . 3
1.2 Bayesian Parameter Inference . . . 4
1.3 Sampling Techniques . . . 7
1.3.1 Independent and Identically Distributed Samples . . . 8
1.3.2 Markov Chain Monte Carlo Sampling . . . 9
1.3.3 Hamiltonian Monte Carlo . . . 13
1.3.4 Nested Sampling . . . 15
1.4 Enhancing the Efficiency of Sampling Methods . . . 16

2 Development of Advanced Techniques for MCMC Sampling 19

2.1 Integration With an Adaptive Harmonic Mean . . . 19
2.1.1 Reduced Volume Harmonic Mean Estimate . . . 20
2.1.2 An Adaptive Harmonic Mean Integration Algorithm . . . 24
2.1.3 Benchmark Examples . . . 28
2.1.4 Harmonic Mean with Spherical Volumes . . . 34
2.1.5 AHMI Conclusions . . . 35
2.2 Sampling Parallelization via Space Partitioning . . . 37
2.2.1 Algorithm Overview . . . 37
2.2.2 Implementation . . . 39
2.2.3 Performance Benchmark . . . 41
2.2.4 Space Partitioning Conclusions . . . 49
2.3 BAT.jl: A Julia-Based Tool for Bayesian Inference . . . 49
2.3.1 Motivation and Overview . . . 49
2.3.2 Functionalities . . . 50

3 Introduction to Particle Acceleration and AWAKE 53

3.1 Conventional Particle Acceleration . . . 53
3.1.1 Historical Overview . . . 53
3.1.2 Future Accelerators . . . 54
3.2 Plasma-Based Particle Acceleration . . . 56
3.2.1 A Digression Into Plasma Theory . . . 56
3.2.2 Plasma Wakefield Acceleration . . . 58
3.3 The AWAKE Experiment . . . 60
3.3.1 Experiment Overview . . . 60
3.3.2 Proton Bunch Diagnostics in AWAKE . . . 62
3.3.3 Impact of the Proton Bunch on the Wakefield Amplitude . . . 63
3.3.4 Importance of the Proton Bunch for Plasma Modelling . . . 64

4 Analysis of Proton Bunch Parameters in the AWAKE Experiment 65
4.1 Proton Bunch Measurements . . . 65
4.1.1 Experimental Setup . . . 65
4.1.2 The Dataset . . . 66
4.2 Preliminary Investigations . . . 67
4.2.1 Pixel Calibration Factor . . . 67
4.2.2 Resolution Functions . . . 69
4.2.3 Background Distribution . . . 71
4.3 Statistical Model . . . 74
4.3.1 Likelihood Definition . . . 74
4.3.2 Bunch Propagation Equation . . . 75
4.3.3 Camera Modeling . . . 77
4.3.4 Likelihood Implementation . . . 77
4.3.5 Model Validation . . . 78
4.4 Parameters . . . 81
4.4.1 Model Parameters . . . 81
4.4.2 Sampling Parameters . . . 82
4.5 Sampling Results . . . 83
4.5.1 Analysis of Experimental Data: Single Gaussian Model . . . 83
4.5.2 Analysis of Experimental Data: Double Gaussian Model . . . 83
4.5.3 Stability Analysis . . . 90
4.5.4 Bunch Drift . . . 95
4.6 Self-Modulation of the Proton Bunch in Plasma . . . 96
4.6.1 Description of the QV3D Code . . . 96
4.6.2 Plasma Modeling Results . . . 96
4.7 Conclusions . . . 101

Afterword 103

Bibliography 105

Motivation and Overview

Throughout the history of science, data — the outcome of experimental observations — has been a crucial element that helped scientists to advance our understanding of Nature and the Universe. In fact, the scientific method used since at least the 17th century consists of the validation of certain subjective hypotheses based on the observed data; and the process of hypotheses validation is one of the key elements of scientific research. One of the earliest historical examples of this scientific approach is the discovery of laws of planetary motion made by J. Kepler. Namely, Kepler used the data recorded by the astronomer T. Brahe to develop equations of planet motion and showed that the orbit of Mars could be described as an ellipse.

A lot of time has passed since then. Rapid growth in technologies in the last few decades significantly affected almost all aspects of humans life. In the research world, the complexity of experiments, the amount of data, and the number of available analysis tools have drastically increased. Yet, the main scientific paradigm remained largely unchanged — for making discoveries, we still need to test empirical hypotheses using now often much larger and more complex datasets.

Just to give a feel of what modern experimental data frontiers are, let us consider a few oft-encountered examples. The experiments at the European Organization for Nuclear Research (CERN) that perform investigations in particle and high energy physics produce approximately 115 PB of data per year [1]. The data from the Large Hadron Collider (LHC) represent the results of several billion recorded collisions each year. This data is transmitted and stored around the Globe using a worldwide computing grid, with data processing rates of several PB/day. To make important discoveries, such as the discovery of the Higgs Boson, the data should be carefully analyzed to find an order of tens of collisions of interest out of billions recorded. Another example is the Square Kilometre Array [2], an intergovernmental radio telescope planned to consist of thousands of small telescopes spread over an area of several thousand square kilometers. The data from these telescopes can be combined to produce datasets of extremely high sensitivity and angular resolution; with data rates estimated to be of the order of 130 PB per year [3]. Earth sciences is another field of science which deals with huge data volumes and requires advanced analyses. Namely, satellite observations, seismograph stations, weather and buoy stations, and tiltmeters generate databases used to make predictions over time-scales from minutes (such as earthquakes predictions) to hundreds of years (such as global warming).

Hence, just over a few hundred years after Kepler's discovery, it became common for scientists to analyze large volumes of data to make new valuable discoveries and predictions. The need to deal with this data also changes the conventional way of doing science — theory and experiments are now heavily supported by computations. New disciplines such as computational physics, computational biology, and data science emerged to respond to the need of having experts in these interdisciplinary fields. Dedicated research institutions and programs, such as the Alan Turing Institute in London or the Origins Data Science Laboratory in Munich, have been established that aim to solve problems for both science and industry.

The work on this thesis was conducted in the scope of one such program, called the INSIGHTS Marie Sklodowska-Curie Innovative Training Network. The goal of this program is to develop advanced statistical methods, implement them in software, and apply them to solve problems in physics and other fields. The training network consists of multiple work packages, such as statistical methods, software tools, physics applications, and statistics for society. These research projects are conducted at 10 partner institutions across Europe, with the Max Planck Institute for Physics in Munich among them. The developments presented in this thesis belong to the work package Statistical Methods. They can be divided into two categories: Development of algorithms for Bayesian computations and Markov Chain Monte Carlo (MCMC) methods, and the application of statistical methods to the analysis of proton bunch parameters in the Advanced Proton Driven Plasma Wakefield Acceleration Experiment (AWAKE) at CERN.

This thesis is organized as follows: Chapter 1 provides a general introduction to Bayesian analysis and MCMC methods and gives a brief description of current developments to enhance the efficiency of MCMC methods. This is followed by Chapter 2, in which the algorithms for numerical evidence estimation and MCMC parallelization are presented, and their implementation in the Bayesian Analysis Toolkit (BAT.jl) software package. The second part of the thesis starts with Chapter 3, in which a brief introduction to conventional and plasma-based particle accelerators is given, and the AWAKE experiment is discussed. In Chapter 4, a detailed statistical analysis of the proton bunch parameters in the AWAKE experiment is presented and the self-modulation of the proton bunches with measured parameters is studied using plasma modeling. Finally, the thesis is concluded with a brief summary of presented developments and a discussion of their recent applications.

1 Introduction to Bayesian Inference

There are two main schools of statistical reasoning: Bayesian and frequentist. A long philosophical debate about the primacy of one of the two approaches has been conducted by generations of scientists, and this debate is yet not finished. The reason for this is rooted in different interpretations of the meaning of probability.

In the frequentists schools, developed by R. Fisher [4], J. Neyman and E. Pearson [5], probability shows the frequency of various outcomes of an experiment given a large number of trials. This probability is independent of opinion, and it can be determined by a repeatable, objective process.

In the Bayesian school, developed originally by T. Bayes and R. Price [6] and later by P. Laplace [7], probability represents an abstract concept that measures a state of knowledge or a degree of belief in a given proposition. It does not assign a single value for the probability of an outcome, and instead, a range of values is considered, each with its probability of being true. A Bayesian approach requires one to specify a subjective prior that expresses the initial knowledge about the hypothesis.

For most of the 20th century, the frequentist school has been dominant, especially in fields like medicine, biology, and the social sciences. The rapid growth in computing power and big data renovated interest in Bayesian approaches. Moreover, it made it very promising for many practical applications, especially those with a need to continuously update knowledge about the model parameters when the new data becomes available.

Bayesian analysis plays a central role in this thesis; therefore, its theoretical background together with the common numerical techniques for practical applications are discussed next.

1.1 Bayes' Theorem

In 1933 A. Kolmogorov introduced the basic axioms of the mathematical formulation of probability [8] from which many useful rules for studying probabilities can be deduced. To describe them, we define a measure space as (Ω, F, P), where Ω is a non-empty set of possible states, F is the set of subsets of Ω that has Ω as a member (that is closed under complementation and union), and $P(A)$ is the probability of some event $A \in F$. In the following, 'event' will denote occurrence of a state in F.

The Kolmogorov axioms are defined as follows:

Axiom 1. The probability of A that belongs to a subset of F is a non-negative real number

$$P(A) \in \mathbb{R},\ P(A) \geq 0,\ \forall A \in F. \tag{1.1}$$

Axiom 2. The probability that at least one event from the entire state space Ω will occur is one

$$P(\Omega) = 1. \tag{1.2}$$

Axiom 3. The probability of mutually exclusive events is equal to the sum of their probabilities

$$P(A \cup B) = P(A) + P(B),\ \forall A, B \in F\ \exists A \cap B = \varnothing. \tag{1.3}$$

The conditional probability describes the probability of one event, given that some other event occurred. For example, if we denote two events by A and B, then the conditional probability of $P(A|B)$ denotes a probability of event A being true given that B is true. The conditional probability can be expressed together with the joint probability $P(A \cap B)$ as

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \text{ if } P(B) \neq 0. \tag{1.4}$$

The same reasoning can be applied to the probability $P(B|A)$. Taking into account that the joint probabilities $P(A \cap B)$ and $P(B \cap A)$ are equal, one can write

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)} \text{ if } P(A) \neq 0, P(B) \neq 0, \tag{1.5}$$

and the last equation is called Bayes' theorem. We can consider the case when Ω is partitioned into n sets, i.e., $\{B_1, ..., B_n\}$, $B_i \cap B_j = \varnothing$ for $i \neq j$, and $\sum_{i=1}^{n} B_i = \Omega$. Then $P(A)$ can be described by the law of total probability

$$P(A) = \sum_{i=1}^{n} P(A|B_i)P(B_i), \tag{1.6}$$

Using this, we can rewrite Eq. 1.5 as

$$P(B_j|A) = \frac{P(A|B_j)P(B_j)}{\sum_{i=1}^{n} P(A|B_i)P(B_i)}, \tag{1.7}$$

which is also called the Bayes-Laplace Theorem, acknowledging that Laplace was the first who used it for scientific purposes.

Eq. 1.7 has a particular meaning if instead of considering events A and B we consider a data $\mathcal{D}$ and a hypothesis H that describes the data. In that case, $P(H|\mathcal{D})$ is the posterior that describes the probability of a hypothesis given the data, $P(H)$ is the prior probability of H being true, $P(\mathcal{D}|H)$ is called the likelihood, and it presents the probability of data given that the hypothesis is true, and $P(\mathcal{D})$ is called evidence, representing a probability of the data.

1.2 Bayesian Parameter Inference

In scientific experiments, we typically have some data $\mathcal{D}$ that we want to use to draw conclusions about the underlying process or processes. The standard procedure for doing this in a Bayesian context is, for example, to develop a model M that is characterized by the parameter vector $\boldsymbol{\lambda} = \{\lambda_i, ..., \lambda_d\} \in \mathbb{R}^d$ where d is the number of dimensions[1], and find the parameters that represent the data most precisely. The process of determining the best parameter values is called 'parameter inference', and if it uses Bayes' theorem, then

[1] In the following, λ will represent a scalar, and $\boldsymbol{\lambda} = \{\lambda_i, ..., \lambda_d\}$ will represent a d-dimensional vector unless otherwise stated.

'Bayesian parameter inference'. In this context, we can reinterpret Bayes' theorem as

$$P(\boldsymbol{\lambda}|\mathcal{D}, M) = \frac{P(\mathcal{D}|\boldsymbol{\lambda}, M)P(\boldsymbol{\lambda}|M)}{P(\mathcal{D}|M)}. \tag{1.8}$$

The denominator is called 'Bayesian evidence' or 'marginal likelihood' and similarly to Eq. 1.6 it is given by

$$P(\mathcal{D}|M) = \int P(\mathcal{D}|\boldsymbol{\lambda}, M)P(\boldsymbol{\lambda}|M)d\boldsymbol{\lambda}. \tag{1.9}$$

Bayes' theorem described by Eq. 1.8 is well known for both frequentist and Bayesian schools of statistics, and it should be used by everyone if the prior $P(\boldsymbol{\lambda}|M)$ is known exactly. If the prior is not known, Bayesians develop one using the best knowledge that they have. After that, they use the data to draw conclusions about their hypothesis. In contrast to this, frequentists draw conclusions about their data using only the likelihood $P(\mathcal{D}|\boldsymbol{\lambda}, M)$.

The prior probability density is a key aspect of the Bayesian approach as it allows to incorporate information about parameter distributions, constraints, etc., into inference using mathematical representation. To utilize the prior information and update its posterior, the following procedure is used:

1. Define a prior probability density $P(\boldsymbol{\lambda}|M)$ that expresses prior belief about parameters $\boldsymbol{\lambda}$ of the model M.

2. Build a statistical model that represents the probability of the data, $P(\mathcal{D}|\boldsymbol{\lambda}, M)$, given the model parameters $\boldsymbol{\lambda}$.

3. Use observed data $\mathcal{D}$ to update a prior probability distribution to posterior probability distribution, $P(\boldsymbol{\lambda}|\mathcal{D}, M)$, using Bayes' theorem.

The posterior probability distribution is the solution to the Bayesian inference problem. It contains the information about the model parameters, given the measured data, and it can be used as a prior for the next analysis if the new data becomes available.

It is sometimes more convenient to summarize the posterior distribution using a point estimate and the probability ranges. There are multiple standard approaches, e.g., the mean and the standard deviation, the median and the central interval, or the mode and the shortest interval. For example, for one-dimensional λ the mean and the variance can be defined as

$$\begin{aligned} \mu &\equiv \langle\lambda\rangle_P = \int_{-\infty}^{\infty} \lambda P(\lambda|\mathcal{D}, M)d\lambda, \\ \sigma^2 &\equiv \left\langle(\mu-\lambda)^2\right\rangle_P = \int_{-\infty}^{\infty} (\mu-\lambda)^2 P(\lambda|\mathcal{D}, M)d\lambda. \end{aligned} \tag{1.10}$$

Alternatively, one can compute the posterior median, λ_{med}, and the central interval, $C = [\lambda_{\text{min}}, \lambda_{\text{max}}]$, defined as

$$\begin{aligned} \int_{-\infty}^{\lambda_{\text{med}}} P(\lambda|\mathcal{D}, M)d\lambda &= \int_{\lambda_{\text{med}}}^{\infty} P(\lambda|\mathcal{D}, M)d\lambda = 0.5, \\ \int_{-\infty}^{\lambda_{\text{min}}} P(\lambda|\mathcal{D}, M)d\lambda &= \int_{\lambda_{\text{max}}}^{\infty} P(\lambda|\mathcal{D}, M)d\lambda = \alpha/2, \end{aligned} \tag{1.11}$$

where $1-\alpha$ represents probability enclosed by the central interval, so the parameter λ belongs to this interval with probability $1-\alpha$.

Bayesian inference allows to compare which one of two models represents the data more accurately. Assuming that we have two models, denoted as M_1 and M_2, with the prior probabilities $P(M_1)$ and $P(M_2)$, the posterior odds can be defined as

$$\begin{aligned} K &\equiv \frac{P(M_1|\mathcal{D})}{P(M_2|\mathcal{D})}, \\ &= \frac{P(\mathcal{D}|M_1)P(M_1)}{P(\mathcal{D}|M_2)P(M_2)}, \end{aligned} \tag{1.12}$$

where $\frac{P(M_1)}{P(M_2)}$ is called the prior odds, and $B = \frac{P(\mathcal{D}|M_1)}{P(\mathcal{D}|M_2)}$ is called the Bayes factor. If the two models are given by the same prior probabilities, then the Bayes factor is equal to the posterior odds. Values of B larger than one indicate that the model M_1 represents the data more accurately than the model M_2.

Bayesian model comparison considers integrated support of the posterior probability distribution, and therefore the choice of parameter priors should be carefully considered. For example, suppose initial knowledge about the model parameters is absent, and one uses the uniform priors to model it. In that case, the resulting value of the Bayes factor can vary in favor of one model or another depending on the range of the prior parameters. Additionally, Bayesian evidence has an intrinsic property to satisfy Occam's razor principle [9], penalizing unnecessarily complex models. Namely, the inclusion of additional parameters to otherwise equivalent models requires one to specify their priors. Broad and non-informative priors on these parameters can introduce a penalty to the Bayesian evidence that can play in favor of another, simpler model. Thus, indicating that the simpler model that does not overfit the data should be preferred. A detailed explanation of the Bayesian Occam's razor principle can be found in [10, 11].

For many practical applications, a solution of Eq. 1.8 cannot be obtained analytically. This is the case, e.g., when the likelihood is given by a separate program that models the performance of some experimental device, binned histograms represent the prior knowledge, or just a closed-form solution to the integral cannot be evaluated. In such situations, one has to consider numerical algorithms to approximate the posterior distributions.

1.3 Sampling Techniques

Let us assume that we have an unnormalized target density function given, for instance, by a product of the likelihood and prior in Eq. 1.8, i.e.,

$$f(\boldsymbol{\lambda}) \sim P(\mathcal{D}|\boldsymbol{\lambda}, M)P(\boldsymbol{\lambda}|M), \tag{1.13}$$

where $\mathcal{D}$ and M are fixed. Let us also assume that our goal is to compute expectation values of a form

$$r \equiv \langle g \rangle_f = \frac{\int g(\boldsymbol{\lambda}) f(\boldsymbol{\lambda}) d\boldsymbol{\lambda}}{\int f(\boldsymbol{\lambda}) d\boldsymbol{\lambda}}, \tag{1.14}$$

where g represents a function of interest, such as the mean or standard deviation. Depending on the dimensionality of the parameter space $\boldsymbol{\lambda}$ and the form of the target density $f(\boldsymbol{\lambda})$, different approaches to approximate the target density and the expectation values $\langle g \rangle_f$ exist.

The simplest one is to construct d-dimensional grid of uniformly distributed points and use them to evaluate local values of the target density. This approach is effective for problems with a small number of dimensions. If the number of dimensions is large, then the overall number of points will scale as N^d, where N is a number of points per dimension. The exponential scaling of the number of evaluation points makes the grid approximation extremely inefficient.

Another approach is to approximate the target density by drawing random samples from it. The class of algorithms that uses random sampling to approximate numerically density functions is called Monte Carlo (MC) sampling. This name was given by N. Metropolis and S. Ulam [12] in 1949, intuitively referring to the city in Monaco, which has a large number of casinos.

According to the law of large numbers [13], the average of a function $g(\boldsymbol{\lambda})$ over the set of independent and identically distributed[2] (*i.i.d.*) samples $\{\boldsymbol{\lambda}\} = \{\boldsymbol{\lambda}_1, ..., \boldsymbol{\lambda}_N\}$ drawn from a distribution $f(\boldsymbol{\lambda})$, approaches the expectation value r when the number of samples N is large[3], i.e.

$$\hat{r} = \frac{1}{N}\sum_{i=1}^{N} g(\boldsymbol{\lambda}_i). \tag{1.15}$$

The convergence of $\hat{r}$ to r is characterized by the average error proportional to $\sim N^{-1/2}$, and it is independent of the number of dimensions [14]. This makes the sampling approach much more computationally efficient compared to the d-dimensional grid approximation in high dimensional spaces. Hence, the problem of estimation of the expectation values reduces to the problem of accurate sampling from the target densities. A variety of techniques exist for drawing samples from the target density, and a description of a few of them is given next.

[2] The definition of *i.i.d.* samples is given in Section 1.3.1.

[3] In the following, $\boldsymbol{\lambda}_i$ will denote one sample drawn from a target density, and $\{\boldsymbol{\lambda}\}$ will denote a set of N samples.

1.3.1 Independent and Identically Distributed Samples

A set of N one-dimensional samples $\{\lambda_1, ..., \lambda_N\}$ drawn from a distribution $f(\lambda)$ is defined as *i.i.d.* if the following two conditions hold

$$\begin{aligned} &F_{\lambda_i}(x) = F_{\lambda_j}(x), \text{ for } i, j = 1...N, \ i \neq j, \\ &F_{\lambda_1,...,\lambda_N}(x_1, ..., x_N) = \prod_{i=1,...,N} F_{\lambda_i}(x_i), \end{aligned} \tag{1.16}$$

where $F_{\lambda_i}(x)$ denotes a cumulative distribution function (CDF):

$$F_{\lambda_i}(x) = \frac{\int_{-\infty}^{x} f_{\lambda_i}(t)dt}{\int_{-\infty}^{\infty} f_{\lambda_i}(t)dt}, \tag{1.17}$$

and $F_{\lambda_1,...,\lambda_N}(x_1, ..., x_N)$ denotes a joint cumulative distribution function, defined, e.g., in [15].

In general, generating *i.i.d.* samples from arbitrary distributions is a non-trivial and often not possible task. However, there are a couple of approaches that can be used for specific distributions. Their detailed description is given in [16], and some of the methods are listed below:

- In the 'inversion method', the explicit expression for the CDF is required. Given it, the inverse CDF can be defined as

  $$F^{-}(u) = \inf\{x : F(x) \geq u\}. \tag{1.18}$$

  To generate samples according to $F(x)$, one can generate samples $\{U_1, ..., U_N\}$ according to the uniform probability distribution $\mathcal{U}(x|0, 1)$ where

  $$\mathcal{U}(x|a, b) = 1/(b - a) \text{ for } a < x < b \text{ and } x \in \mathbb{R}, \tag{1.19}$$

  and then make the transformation $x = F^{-}(u)$. Even though this approach requires only uniform random number generator, the practical application of it is limited because the explicit expression for CDF is not always available.

  For example, the exponential distribution, $\mathcal{E}(x|\nu) = \nu \exp(-\nu x)$ for $x > 0, \nu > 0$, has a CDF $F(x|\nu) = 1 - \exp(-\nu x)$. The uniform samples $\{U_1, ..., U_N\}$ drawn from $\mathcal{U}(x|0, 1)$ can be transformed into exponential as $X = -\log(1 - U)/\nu$.

- The 'relationships method' takes advantage of some known relationships between different distributions. For example, the Box-Muller algorithm allows generating samples from a unit Normal distribution[4], $\mathcal{N}(x|0, 1)$, defined as

  $$\mathcal{N}(x|\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(\mu - x)^2}{2\sigma^2}\right], \tag{1.20}$$

  where μ represents the mean and σ^2 represents the variance. Two samples $\{U_1, U_2\}$ drawn from $\mathcal{U}(x|0, 1)$ can be transformed into samples $\{X_1, X_2\}$ from $\mathcal{N}(x|0, 1)$ using the following transformation

  $$\begin{aligned} X_1 &= \sqrt{-2\log U_1} \cos 2\pi U_2, \\ X_2 &= \sqrt{-2\log U_1} \sin 2\pi U_2. \end{aligned} \tag{1.21}$$

[4] In the following, the terms 'Normal distribution' and 'Gaussian distribution' will be used interchangeably.

Another example is the generation of samples from the Poisson distribution defined as

$$\mathcal{P}(x|\nu) = e^{-\nu}\frac{\nu^x}{x!}, \text{ for } \nu > 0, x \in \mathbb{Z}. \tag{1.22}$$

One can first generate samples $\{X_1, ..., X_N\}$ from $\mathcal{E}(x|\nu)$ and then use the fact that they are connected to the Poisson distribution through the Poisson process [17, 18]. The samples $N \sim \mathcal{P}(x|\nu)$ can be obtained from the following condition

$$\mathcal{P}(N = k|\nu) = \mathcal{P}(X_1 + ...X_k \leqslant 1 < X_1 + ...X_{k+1}|\nu), \tag{1.23}$$

which requires generating exponential random samples until their sum exceeds 1. This approach is simple for implementation, but it is not computationally efficient for large ν (see [16] for detailed discussion).

- The ‘rejection method’ uses the geometrical probability interpretation of density functions when it is not possible to exploit their probabilistic properties directly. This approach can be used to generate samples from the densities that: (a) Have a finite support range, (b) Function values are bounded.

Due to their statistical properties, *i.i.d.* samples should be always preferred from other, possibly correlated, samples. If the generation of *i.i.d.* samples is not possible, then one has to consider other methods, that sacrifice the condition of independence but still resemble the exact target density.

1.3.2 Markov Chain Monte Carlo Sampling

Markov Chain Monte Carlo (MCMC) is a class of algorithms that allows generating samples with a distribution proportional to a given target density function using properties of Markov Chains. It was first developed approximately with the development of the first computers, and it has been used to solve problems required for the Manhattan project in 1939 [19].

Markov Chain

There exist a variety of random processes such as Gaussian processes [20, 21], Poisson processes [17, 18], autoregressive models [22], Markov chains [23, 24]. Each of these processes has its unique properties that can be used to study them.

The Markov chain is a random process in which, for any given iteration, the conditional distribution of the future state is completely determined by the current state of the system and not by the past states. In other words, the process is memoryless, and it is not possible to get any additional information about the future behavior of the process by collecting information about the past.

To discuss some properties of the Markov process, let us assume that we wants to generate a sequence of N vectors in a discrete state space $\boldsymbol{\lambda}_i \rightarrow \boldsymbol{\lambda}_j \rightarrow ... \rightarrow \boldsymbol{\lambda}_N$. Each of these states is characterized by the transition matrix W, where the matrix element $W(\boldsymbol{\lambda}_j|\boldsymbol{\lambda}_i)$ gives the probability that the system makes a transition from state $\boldsymbol{\lambda}_i$ to state $\boldsymbol{\lambda}_j$, where $j - i = 1$. The dynamics of such transitions in the state space can be described in terms of the master equation [25]

$$\frac{\mathrm{d}f(\boldsymbol{\lambda}_i)}{\mathrm{d}t} = \sum_j f(\boldsymbol{\lambda}_j)W(\boldsymbol{\lambda}_i|\boldsymbol{\lambda}_j) - f(\boldsymbol{\lambda}_i)\sum_j W(\boldsymbol{\lambda}_j|\boldsymbol{\lambda}_i). \tag{1.24}$$

Given a time-homogeneous state space, i.e.,

$$\sum_j f(\boldsymbol{\lambda}_j) W(\boldsymbol{\lambda}_i|\boldsymbol{\lambda}_j) = f(\boldsymbol{\lambda}_i) \sum_j W(\boldsymbol{\lambda}_j|\boldsymbol{\lambda}_i), \tag{1.25}$$

a Markov chain has 'stationary distribution' if the following condition holds:

$$f(\boldsymbol{\lambda}_i) = \sum_j f(\boldsymbol{\lambda}_j) W(\boldsymbol{\lambda}_i|\boldsymbol{\lambda}_j). \tag{1.26}$$

A Markov chain is called 'irreducible' if there exist a finite number of steps, $N < \infty$, needed to reach any two points in a state space, i.e., $W^N(\boldsymbol{\lambda}_j|\boldsymbol{\lambda}_i) > 0$. If a Markov chain is irreducible and has a stationary distribution, then this stationary distribution is unique, and the chain is 'positive recurrent'. An aperiodic Markov chain that is positive recurrent and has a stationary distribution is 'asymptotic', meaning that the stationary distribution can be reached from any point of a state-space in a limit of a large number of iterations. Irreducible Markov chains with unique stationary distributions are called 'ergodic', and they allow to approximate expectation values defined in Eq. 1.14 using the estimator defined in Eq. 1.15.

If the dynamic of the chain remains unchanged if the chain runs backward while being at a stationary distribution, then we can say that the chain is time reversible and it satisfies the condition of detailed balance:

$$f(\boldsymbol{\lambda}_j) W(\boldsymbol{\lambda}_i|\boldsymbol{\lambda}_j) = f(\boldsymbol{\lambda}_i) W(\boldsymbol{\lambda}_j|\boldsymbol{\lambda}_i). \tag{1.27}$$

The condition of time reversibility should not necessarily be satisfied to guarantee the existence of an asymptotic stationary distribution. But if this condition is satisfied, then it guarantees that the limiting stationary distribution exists. This allows using the detailed balance condition to construct algorithms for generating MCMC samples.

Due to their great properties, Markov chains are widely used in many fields such as statistics [16, 23], biology [26, 27], and queueing theory [28, 29].

Sampling using Markov Chains

MCMC methods are typically used to solve problems with intractable many-dimensional target densities. There are various MCMC algorithms, and their main difference is in the way how they define a transition matrix $W(\boldsymbol{\lambda}_i|\boldsymbol{\lambda}_j)$ of the Markov chain. These algorithms have in common that (1) the representative samples can be generated after reaching a certain number of burn-in iterations needed for the Markov chain to approach stationary distribution, and (2) that the resulting samples are correlated.

Convergence To determine whether the samples represent target distribution correctly, M chains can be run from different starting points, and their distributions can be compared. There are no methods to detect a convergence of samples from multidimensional target densities, and the convergence criteria can only be used to detect convergence failure. One of the most popular criteria was proposed by A. Gelman and D. B. Rubin in [30], and it compares variance within chains and the pooled variance estimate. For example, if we have a one-dimensional parameter space and each of M chains produced N samples, i.e., $\{\lambda_{i,j}\}$ for $i = 1, ..., M$, $j = 1, ..., N$, then we can compute the estimator

$$\hat{R} = \frac{\hat{V}}{W}, \tag{1.28}$$

where

$$
\begin{aligned}
W &= \sum_{i=1}^{M}\sum_{j=1}^{N} \frac{(\lambda_{i,j} - \langle\lambda_i\rangle)^2}{M(N-1)}, \\
\hat{V} &= \frac{(N-1)W}{N} + \sum_{i=1}^{M} \frac{(\langle\lambda_i\rangle - \langle\lambda\rangle)^2}{M-1},
\end{aligned}
\tag{1.29}
$$

and $\langle\lambda_i\rangle$ represents i-th chain mean and $\langle\lambda\rangle$ is the overall mean. The degree of convergence is measured by the closeness of $\hat{R}$ to the value 1, and values that are larger than 1 indicate that chains did not converge. There exist other convergence approaches whose review is presented in [31].

Effective sample size Samples that originated from one MCMC chain are correlated. The degree of sample correlation depends on the acceptance probability, number of chains, complexity of the target density function, etc. The effective number of samples can be estimated as

$$N_{eff} = \frac{N}{\hat{\tau}}, \tag{1.30}$$

where N is the number of samples and $\hat{\tau}$ is the integrated autocorrelation time, estimated via the normalized autocorrelation function $\hat{\rho}(\tau)$:

$$\hat{\tau}_k = 1 + 2\sum_{\tau=1}^{\infty} \hat{\rho}_k(\tau), \tag{1.31}$$

$$\hat{\rho}_k(\tau) = \frac{\hat{c}_k(\tau)}{\hat{c}_k(0)}, \tag{1.32}$$

$$\hat{c}_k(\tau) = \frac{1}{N-\tau} \sum_{i=1}^{N-\tau} \left(\lambda_{k,i} - \langle\lambda_k\rangle\right)\left(\lambda_{k,i+\tau} - \langle\lambda_k\rangle\right), \tag{1.33}$$

where k refers to the one of the d dimensions of the multivariate sample $\{\lambda_{k,i}\}$, i reefers to the one of the N samples, and $\langle\lambda_k\rangle$ is the average of the k-th component of all the N samples.

The two most famous sampling algorithms in Bayesian calculations are the Metropolis-Hastings sampler and the Gibbs sampler.

Metropolis-Hastings Algorithm

This algorithm is the oldest and most general algorithm for constructing a sequence of random Markov chain samples from the target density. It has been proposed by Metropolis et al. in 1953 [32] and later generalized by K. Hastings in 1970 [33].

The proposed idea consists of defining a transition probability $W(\boldsymbol{\lambda}_i|\boldsymbol{\lambda}_j)$ in the following way

$$W(\boldsymbol{\lambda}_i|\boldsymbol{\lambda}_j) = T_{gen}(\boldsymbol{\lambda}_i|\boldsymbol{\lambda}_j)A_{acc}(\boldsymbol{\lambda}_i|\boldsymbol{\lambda}_j), \tag{1.34}$$

where $T_{gen}(\boldsymbol{\lambda}_i|\boldsymbol{\lambda}_j)$ is a proposal probability distribution, i.e., a probability to generate a trial move from state $\boldsymbol{\lambda}_j$ to state $\boldsymbol{\lambda}_i$, and $A_{acc}(\boldsymbol{\lambda}_i|\boldsymbol{\lambda}_j)$ is a probability to accept this move. Taking into account Eq. 1.27, one can rewrite the detailed balance condition:

$$f(\boldsymbol{\lambda}_j)T_{gen}(\boldsymbol{\lambda}_i|\boldsymbol{\lambda}_j)A_{acc}(\boldsymbol{\lambda}_i|\boldsymbol{\lambda}_j) = f(\boldsymbol{\lambda}_i)T_{gen}(\boldsymbol{\lambda}_j|\boldsymbol{\lambda}_i)A_{acc}(\boldsymbol{\lambda}_j|\boldsymbol{\lambda}_i). \tag{1.35}$$

The acceptance probability that satisfies the solution of this equation can be written as

$$A_{acc}(\boldsymbol{\lambda}_i|\boldsymbol{\lambda}_j) = \min\left\{1, \frac{f(\boldsymbol{\lambda}_i)T_{gen}(\boldsymbol{\lambda}_j|\boldsymbol{\lambda}_i)}{f(\boldsymbol{\lambda}_j)T_{gen}(\boldsymbol{\lambda}_i|\lambda_j)}\right\}, \tag{1.36}$$

and it has been shown by A. Barker [34] that this solution results in the fastest mixing rate of samples compared to other methods. The algorithm can be presented by the following pseudo-code:

Listing 1.1: Metropolis-Hastings Algorithm

~~~~
```
Input:
    target   : Target probability density
    pr_prob  : Proposal probability density
    startpt  : Starting point
    nsteps   : Number of steps
Output:
    samples  : Vector of samples
```
function sample(target, pr_prob, startpt, nsteps)

    λ_i = startpt
    samples = []

    for t in 1:nsteps
        λ_j = draw_sample(pr_prob(λ_j|λ_i))
        accept = target(λ_j) * pr_prob(λ_i|λ_j) / (target(λ_i)*pr_prob(λ_j|λ_i))
        a = draw_sample(Uniform from 0 to 1)
        if a < minimum([1, accept])
            λ_i = λ_j
        else
            λ_i = λ_i
        end
        push!(samples, λ_i)
    end

    return samples
end
~~~~

A proposal probability distribution can be defined in many different ways, resulting in different versions of the Metropolis-Hastings algorithm. The most typical are Gaussian, Student's *t*, and Uniform distributions. The width of the proposal probability distribution is the tuning parameter, and it affects the correlation between MCMC samples. Choosing a broad width of the proposal distribution makes the samples less correlated and the acceptance rate smaller. By making the width of the proposal distribution narrow, the correlation between samples and the acceptance rate increase. The parameters of the proposal distribution are typically tuned to have the acceptance rate in the range of $[0.15, 0.35]$, which has been proved to be optimal [35].

Gibbs Sampling

This algorithm was described by brothers Geman in 1984 [36], and it is named after the physicist J. W. Gibbs.

The idea of the Gibbs sampler, which is a special case of the Metropolis-Hastings algorithm [16], is to replace sampling from a full joint distribution by sampling over conditional distributions. To define the transition probability of the simplest Gibbs sampler, let us assume that the parameter vector $\boldsymbol{\lambda}_i$ consist of d components, i.e., $\boldsymbol{\lambda}_i = \{\lambda_{1,i}, \lambda_{2,i}, ..., \lambda_{d,i}\}$.

The probability to construct a displacement from $\boldsymbol{\lambda}_i$ to $\boldsymbol{\lambda}_j$ is then given by

$$W(\boldsymbol{\lambda}_j|\boldsymbol{\lambda}_i) = \prod_{k=1}^{d} P(\lambda_{k,j}|\lambda_{1,j}, ..., \lambda_{k-1,j}, \lambda_{k+1,i}, ..., \lambda_{d,i}), \tag{1.37}$$

and it assumes that one can generate samples from the distribution of each variable in turn, conditional on the current values of the other variables.

This definition of the transition probability does not satisfy the time reversibility condition. This can be fixed, for instance, by picking random components of the parameter vector and sampling them consecutively. The following pseudo-code summarizes the simplest time-reversible Gibbs algorithm:

Listing 1.2: Gibbs Sampling Algorithm

~~~
```
Input:
    target   : Target probability density
    startpt  : Starting point
    nsamples : Number of samples
Output:
    samples  : Vector of samples
```
function sample(target, startpt, nsamples)
    λ = startpt
    samples = []
    for t in 1:nsteps
        ind = draw_sample(integer from 1 to number of dimensions)
        λ[ind] = draw_sample(target(|all parameters are constant despite λ[int]))
        push!(samples, λ)
    end
    return samples
end
~~~

The Gibbs sampler is usually preferred from the Metropolis-Hastings sampler for problems where the decomposition of the joint distribution into conditionals is easy to implement and fast to run. In such problems, the Gibbs sampler generates samples with the acceptance probability of one. If such decomposition includes multi-modality, then the convergence of the Gibbs sampler will be slower than the convergence of the Metropolis-Hastings sampler because the variables in the Gibbs sampler cannot evolve jointly [16]. In this scenario, tuning the proposal in the Metropolis-Hasting algorithm might produce a higher sampling efficiency.

1.3.3 Hamiltonian Monte Carlo

One of the most sophisticated sampling methods is Hamiltonian Monte Carlo (HMC) [37, 38, 39]. By using a proposal function that is adjusted to the shape of the target distribution, HMC algorithms can yield higher acceptance rates and less correlated samples than other sampling algorithms based on random walks, thus reducing the number of samples required to fully explore the target distribution.

In HMC, the d-dimensional parameter space is expanded to $2d$ dimensions by introducing so-called momenta $\boldsymbol{p}$ as hyperparameters, moving from the original phase space to the canonical phase space $\boldsymbol{q} \to (\boldsymbol{q}, \boldsymbol{p})$. In order to conform to standard notation when discussing HMC, $\boldsymbol{q}$ is used here to represent the parameters of the model in place of $\boldsymbol{\lambda}$.

In the HMC formalism, the target distribution $f(\boldsymbol{q})$ is lifted to the canonical phase space using a joint probability distribution

$$f(\boldsymbol{q},\boldsymbol{p}) = f(\boldsymbol{p}|\boldsymbol{q})f(\boldsymbol{q}) = \mathrm{e}^{-H(\boldsymbol{q},\boldsymbol{p})}\,, \tag{1.38}$$

where the probability distribution of the momenta $f(\boldsymbol{p}|\boldsymbol{q})$ is chosen to be conditional. The last equality in Eq. (1.38) comes from defining the so-called Hamiltonian as

$$H(\boldsymbol{q},\boldsymbol{p}) = -\log f(\boldsymbol{q},\boldsymbol{p}) = -\log f(\boldsymbol{p}|\boldsymbol{q}) - \log f(\boldsymbol{q})\,. \tag{1.39}$$

The differential equations

$$\frac{dq_i}{dt} = \frac{\partial H}{\partial p_i}, \quad \frac{dp_i}{dt} = -\frac{\partial H}{\partial q_i}\,, \tag{1.40}$$

are well known from classical mechanics and referred to as Hamilton's equations of motion. Solving the equations of motion for a certain time T allows moving along trajectories ϕ and gives a transition in the canonical phase space

$$(\boldsymbol{q},\boldsymbol{p}) \to \phi_T(\boldsymbol{q},\boldsymbol{p}) = (\boldsymbol{q}^*,\boldsymbol{p}^*)\,, \tag{1.41}$$

resulting in the new point $(\boldsymbol{q}^*,\boldsymbol{p}^*)$. By marginalizing over the momenta $\boldsymbol{p}$, we obtain a new proposal point $\boldsymbol{q}^*$ in the original parameter space. This proposal is then either accepted as a new sampling point or rejected by calculating an acceptance ratio, similar to the Metropolis-Hasting algorithm. Since the proposal points are generated using information of the target distribution, their acceptance rates are higher than samples using non-problem-specific proposal distributions.

The key steps of the HMC algorithm can be described by the following pseudo-code:

Listing 1.3: Hamiltonian Monte Carlo Algorithm

~~~~
```
Input:
    target   : Target probability density
    pr_prob  : Proposal probability density
    startpt  : Starting point
    max_time : The time during which trajectory evolves
Output:
    samples  : Vector of samples
```
function sample(target, pr_prob, startpt, max_time)

    q = startpt
    samples = []

    for t in 0:max_time

        p = draw_sample(initial momentum)
        q*, p* = run Leap Frog algorithm starting at q, p
        accept = target(q*) * pr_prob(q|q*) / (target(q) * pr_prob(q*|q))
        a = draw_sample(Uniform from 0 to 1)
        if a < minimum([1, accept])
            q = q*
        else
            q = q
        end
        push!(samples, q)
    end

    return samples
end
~~~~

Since HMC requires gradient information and introduces multiple hyperparameters (such as momenta and integration times) into the sampling process, performing Bayesian analyses with HMC samplers is usually not as straight-forward as using the MH algorithm as it requires additional computational steps such as the numerical integration of the equations of motions and the selection and tuning of the hyperparameters. In modern computing languages like Julia, there exist packages that allow automatic differentiation of arbitrary code, allowing to increase the applicability of HMC algorithms.

1.3.4 Nested Sampling

Nested Sampling (NS) is another very efficient approach for Bayesian calculations presented in 2004 by J. Skilling [40]. It inverses the traditional procedure for Bayesian calculation, where the emphasis is placed on evaluating posterior samples. Instead, NS allows an estimate of the Bayesian evidence directly, providing posterior samples at no extra cost. The NS algorithm does not require a gradient of the target density compared to HMC, and it proved to be very efficient for sampling multimodal and many-dimensional target densities.

The motivation of the algorithm is to estimate the following integral

$$I = \int P(\mathcal{D}|\boldsymbol{\lambda}, M)P(\boldsymbol{\lambda}|M)d\boldsymbol{\lambda}, \tag{1.42}$$

given constant $\mathcal{D}$ and model M. The NS algorithm defines the so-called cumulative probability of the likelihood restricted prior (also called the prior mass) as

$$\begin{aligned} X(L_{min}) &= \text{Prob}\left\{P(\mathcal{D}|\boldsymbol{\lambda}, M) > L_{min}\right\} \\ &= \int_{P(\mathcal{D}|\boldsymbol{\lambda}, M) > L_{min}} P(\boldsymbol{\lambda}|M)d\boldsymbol{\lambda}. \end{aligned} \tag{1.43}$$

where L_{min} denotes the minimum likelihood value. Given that, a possibly many-dimensional integral I can be transformed into a one-dimensional integral of form

$$I \equiv \int_0^1 L_{min}(X)dX, \tag{1.44}$$

where the integrand is a positive and decreasing function. Eq 1.44 can be approximated, e.g., using standard quadrature methods,

$$\hat{I} = \sum_{i=1}^{j} L_i(X_{i-1} - X_i), \tag{1.45}$$

where $L_i = L_{min}(X_i)$, and $0 < X_j < ... < X_0 < 1$ is an arbitrary grid in the range $[0, 1]$. The likelihood values L_i are selected by the following procedure: (1) Draw N independent samples from the prior distribution. (2) Find the sample with the smallest likelihood and assign its likelihood value to L_1. (3) Replace this sample with another sample that is drawn from the likelihood restricted prior. (4) Repeat (1)-(3) until a given stopping rule is satisfied, recording L_i for each iteration.

In the above description, the values of X_i are generally unknown. Skilling proposed to approximate them as $X_i = \exp(-i/N)$, such that $\log X_i$ is the expectation of $\log X(L_i)$ (see [40]). A simple version of the nested sampling algorithm can be described by the following pseudo-code:

Listing 1.4: Nested Sampling Algorithm

~~~
```
Input:
    prior       : Prior probability density
    likelihood  : Likelihood
    niter       : Number of iterations
    npoints     : Number of points
Output:
    samples     : Vector of samples
    weights     : Weights of the samples
    I           : Bayesian evidence
```
function sample(prior, likelihood, niter, npoints)

    λ = draw_sample(prior, npoints)  # generate N samples from prior
    samples = []
    weights = []
    I = 0
    x_0 = 1

    for i in 1:niter
        λ_i = find sample with smallest likelihood
        L_i = likelihood(λ_i)
        x_i = exp(-i/N)
        w_i = x_{i-1} - x_i
        I = I + L_i * w_i

        push!(samples, λ_i) # save sample with weight w_i
        push!(weights, w_i)

        Update λ_i with with a sample that keep the likelihood above L_i

    end
    return samples, weights, I
end
~~~

The most challenging part of the NS algorithm is to replace the sample with the smallest likelihood with another sampled from the likelihood restricted prior. There exist different variants of the NS implementations, such as those that are using MCMC or HMC to generate new samples more efficiently. A detailed review of recent developments in this field is presented in [41].

1.4 Enhancing the Efficiency of Sampling Methods

Overall, one of the major strengths of MCMC techniques is that they can converge to the target density even if target functions are highly multidimensional and multimodal. A major difficulty is that convergence is reached only asymptotically, and approaching the stationary distribution can require a very large number of sampling steps. For many real-world applications, evaluation of a target density can be very computationally costly, and there is usually a limit to how far a single target evaluation can be parallelized efficiently; this can make MCMC sampling very costly.

A further complication stems from the fact that a large number of burn-in steps need to be performed for each MCMC chain before representative samples can be generated. The burn-in duration can even exceed the sampling time, especially for target densities that have a complex shape. While separate Markov chains can be run independently and in parallel, simply increasing their number while producing fewer samples from each chain is therefore not an effective parallelization strategy as the length of the burn-in process for each chain would not change.

Significant research has been conducted to enhance the efficiency of MCMC methods.

The developments in this field can be divided into several categories [42].

The first is based on exploiting the geometry of the target density function. As discussed earlier, the HMC sampling belongs to this category, and it introduces an auxiliary variable momentum that proposes displacements solving a Hamiltonian equation of motion. A numerical solution of the Hamiltonian equation using, for instance, standard Euler's method creates an unstable approximation to the trajectory of motion. This induces a bias, and the approximated trajectory drifts away from its true values. Symplectic integrators of different orders of accuracy have been developed to approximate equations of motions more accurately [43], and one of the most famous in terms of speed and accuracy is the so-called leapfrog approximation. The discretization of the Hamiltonian equation introduces two free parameters, the step size and the trajectory length, and one of the most widely used heuristics to chose these parameters are implemented in the 'no-U-turn sampler' (NUTS) [44]. A further attempt to speed up HMC sampling has been proposed in [20] and [45] where the exact target density is replaced by a computationally faster approximation. Overall, the HMC provides less correlated samples than the Metropolis-Hastings algorithm; however, the gradient of the density is not always readily available or cannot be computed in a reasonable time, remaining a practical interest for improving the performance of non-gradient based samplers.

The second approach of accelerating MCMC is based on improving the proposal function. Techniques such as simulated tempering [46, 47], adaptive MCMC [48], and multi-proposal MCMC [49, 50] are available and have been shown to be effective for many applications [51, 52, 53, 54, 55]. The simulated tempering has the drawback that it requires a run time that scales exponentially with a number of dimensions to reach a convergence if the modes of the target density have very different structures. The adaptive MCMC has the drawback that it relies too much on the existing samples to tune the proposal, reinforcing the exclusion of those parts of the space that has not been previously explored. The multi-proposal MCMC methods require generating many additional samples to make a proposal, which can be very expensive if the target density is computationally costly.

The third approach is based on breaking initially complicated problems into simpler pieces. For example, separate MCMC chains explore the parameter space in parallel, and the resulting samples are merged together [56, 57]. As discussed earlier, this approach does not simplify convergence of chains to the stationary distribution. Another approach is to partition the data space [57, 58, 59] or parameter space [60, 61, 62] into simpler pieces that can be processed independently. An effective partitioning can thereby change the task from sampling from a complicated target distribution to sampling from many, simpler target distributions. The approach of space partitioning requires knowledge of the integrated normalization factor to stitch multiple posteriors together; evaluation of this normalization factor is a numerically challenging task that will be discussed in the next chapter.

2 Development of Advanced Techniques for MCMC Sampling

In this chapter, I present the development and numerical implementation of two algorithms for Bayesian computations.

In the first section, the Adaptive Harmonic Mean Integration Algorithm (AHMI) is presented, following closely its description published in [63]. This algorithm has been proposed and developed by joint efforts of my colleagues, A. Caldwell, P. Eller, R. Schick, O. Schulz, M. Szalay, and me. Given samples drawn according to a probability distribution proportional to the function, the AHMI algorithm estimates the integral of the function and the uncertainty of the estimate by applying a harmonic mean estimator to adaptively chosen regions of the parameter space.

In the second part, an approach to improve the efficiency of the MCMC sampling is described that is based on the division of parameter space into simpler regions. An effective partitioning of the parameter space can change the task from sampling from a complicated target distribution to sampling from many, simpler target distributions. This approach requires evaluation of the integral over the target density, which is performed using the AHMI algorithm. This algorithm is presented in [64], and it has been proposed and developed jointly by my colleagues, P. Eller, O. Schulz, A. Caldwell, and me.

Both of the algorithms are implemented in the BAT.jl [65] package that can be found on the following GitHub page [66]. A short overview of the BAT.jl package is given in the last section, following its original description published in [65].

In this Chapter, by mentioning 'we' in the text, I will refer to those colleagues with whom the collaborative work has been performed, who supervised me and helped with different aspects of my work.

2.1 Integration With an Adaptive Harmonic Mean

As it was briefly mentioned earlier, estimating the integral of density functions in high dimensional spaces is a nontrivial task. However, there exist a number of application where such integration is needed, such as Bayesian evidence calculation, or evaluation of thermodynamic potentials. In context of this Chapter, we will consider a problem of target density normalization using a samples drawn from an unnormalized target density

$$\{\boldsymbol{\lambda}\} \sim f(\boldsymbol{\lambda}),\ f(\boldsymbol{\lambda}) > 0\ \forall \boldsymbol{\lambda} \in \mathbb{R}^d, \tag{2.1}$$

where $f(\boldsymbol{\lambda})$ is given by Eq. 1.42. The integral that we want to evaluate is

$$I \equiv \int_\Omega f(\boldsymbol{\lambda}) d\boldsymbol{\lambda}, \tag{2.2}$$

where Ω described the support of f. The underlying assumption of the integration algorithm that is presented in this section is that it will not provide a correct result if the sampling algorithm has failed in correct sampling. The AHMI algorithm is aimed to provide integral estimate without the need to regenerate new samples.

A variety of techniques to estimate evidence in Bayesian calculations have been successfully developed. A summary can be found in [67], where a number of MCMC related techniques are reviewed, including Laplace's method [68], Harmonic Mean Estimation (HME) [69], Chib's method [70], annealed importance sampling techniques [71], NS [72] and thermodynamic integration methods [73, 74].

Only the HME (see Section 2.1.1) and Laplace techniques allow the direct estimation of the evidence from available samples, and the Laplace technique makes the unwanted assumption that the target density is a single multivariate Gaussian. The Laplace approximation fails badly for multimodal distributions or distributions with significant probability mass in the tails of the distribution. The HME method has been strongly criticized (even called 'worst Monte Carlo Method ever' [75]), since its estimator can easily diverge. The algorithm presented in this section is aimed to improve the performance of HME by adaptively finding subvolumes where the integration can be performed without a divergence of HM estimator.

2.1.1 Reduced Volume Harmonic Mean Estimate

We are interested in estimating the integral I from Eq. 2.2. We start by defining the integral

$$I_\Delta \equiv \int_\Delta f(\boldsymbol{\lambda}) d\boldsymbol{\lambda} \tag{2.3}$$

with $\Delta \subset \Omega$ a finite integration region, and the ratio

$$r \equiv \frac{I_\Delta}{I}. \tag{2.4}$$

Given our assumption that the sampling algorithm has successfully sampled from $f(\boldsymbol{\lambda})$, we use the following as an estimator to our ratio

$$\hat{r} = \frac{N_\Delta}{N_\Omega}, \tag{2.5}$$

which is the fraction of the total number of samples that fall within $\Delta \subset \Omega$. Defining the normalized density over Δ

$$\tilde{f}_\Delta(\boldsymbol{\lambda}) = \frac{f(\boldsymbol{\lambda})}{I_\Delta} \qquad \boldsymbol{\lambda} \in \Delta, \tag{2.6}$$

allows us to perform a harmonic mean calculation as follows:

$$E\left[\frac{1}{f(\boldsymbol{\lambda})}\right]_{\tilde{f}_\Delta(\boldsymbol{\lambda})} = \int_\Delta \frac{1}{f(\boldsymbol{\lambda})} \cdot \tilde{f}_\Delta(\boldsymbol{\lambda}) d\boldsymbol{\lambda} = \int_\Delta \frac{1}{f(\boldsymbol{\lambda})} \cdot \frac{f(\boldsymbol{\lambda})}{I_\Delta} d\boldsymbol{\lambda} = \frac{V_\Delta}{I_\Delta} \tag{2.7}$$

where V_Δ is the volume of the region defined by Δ. An estimator for this expectation value is the harmonic mean

$$\hat{X} = \frac{1}{N_\Delta} \sum_{\boldsymbol{\lambda}_i \in \Delta} \frac{1}{f(\boldsymbol{\lambda}_i)}. \tag{2.8}$$

The HME for the reduced volume integral then follows as

$$\hat{I}_\Delta = \frac{V_\Delta}{\hat{X}} = \frac{N_\Delta V_\Delta}{\sum_{\boldsymbol{\lambda}_i \in \Delta} \frac{1}{f(\boldsymbol{\lambda}_i)}} \ . \tag{2.9}$$

This calculation is performed directly from the values of the target density $f(\boldsymbol{\lambda}_i)$ given by the sampling algorithm, and does not require any extra sampling. An estimator for the integral over the full space Ω can then be written down as

$$\hat{I} = \frac{\hat{I}_\Delta}{\hat{r}} = \frac{N_\Omega V_\Delta}{\sum_{\boldsymbol{\lambda}_i \in \Delta} \frac{1}{f(\boldsymbol{\lambda}_i)}} \, . \tag{2.10}$$

The task of estimating our integral therefore reduces to choosing one or several subspaces Δ—typically small regions around local modes of $f(\boldsymbol{\lambda})$. The full space Ω over which the integration ought to be performed can be large or even infinite, while this does not affect the outcome of our integral estimate. We discuss the bias and uncertainty of this estimator in the following subsection.

In general, MCMC samples come with weights (e.g. repeated samples, with the weight being the number of repetitions). We therefore rewrite Eq. 2.10 as

$$\hat{I} = \frac{W_\Omega V_\Delta}{\sum_{\boldsymbol{\lambda}_i \in \Delta} \frac{w_i}{f(\boldsymbol{\lambda}_i)}} \tag{2.11}$$

with w_i the weights assigned to the samples at parameter values $\boldsymbol{\lambda}_i$ and $W_\Omega = \sum_i w_i$ the sum of all weights. The use of weights also allows this technique to be applied to samples obtained from, for example, importance sampling.

Illustration of the Technique

To illustrate our technique for applying harmonic mean integration, we consider the unit normal distribution $P(x) = \mathcal{N}(x|0,1)$. A fixed number of samples ($3 \cdot 10^3$) was generated from directly sampling the unit normal distribution, and Eq. 2.10 was used to calculate the integral for different sub-regions Δ. These regions are defined as windows of x centered on 0 and varied in width from 0.02 up to 9. This was repeated $1.5 \cdot 10^4$ times, and the mean and standard deviation were evaluated. Figure 2.1 shows the results of the integration as a function of window size. As is seen in the figure, harmonic mean integration applied to a finite region gives an accurate value for the integral over a wide range of sampling windows. The variation in the integral results is largest for small windows due to the small number of samples used, and for large windows due to the divergence of the harmonic mean estimator. We discuss the biases in the next section.

Bias and Uncertainty of the Estimator

We can estimate the bias and uncertainty on $\hat{I}$ given in Eq. 2.10 by separately analyzing the behavior of $\hat{r}$ and $\hat{X}$. As described below, we choose regions Δ for which the range of target density values is moderate. For *i.i.d.* sampling, this would imply, via the Central Limit Theorem, that $\hat{X}$ follows a Normal distribution. Assuming we can approximate the distribution of $\hat{X}$ with a Normal distribution, we have

$$P(\hat{X}) \approx \mathcal{N}\left(\hat{\mu}_X = \hat{X}, \hat{\sigma}_X^2 = \frac{\sum_{\boldsymbol{\lambda}_i \in \Delta}(\frac{1}{\boldsymbol{\lambda}_i} - \hat{X})^2}{N_\Delta(N_\Delta - 1)}\right)$$

where $P(\hat{X})$ is the probability distribution for $\hat{X}$ with mean μ and variance σ^2 estimated from the observed samples. Since $\hat{X}$ appears in the denominator in Eq. 2.10, this produces a bias in our integral of size $\hat{\sigma}_X^2/\hat{\mu}_X^2$. The fractional uncertainty in our integral estimator

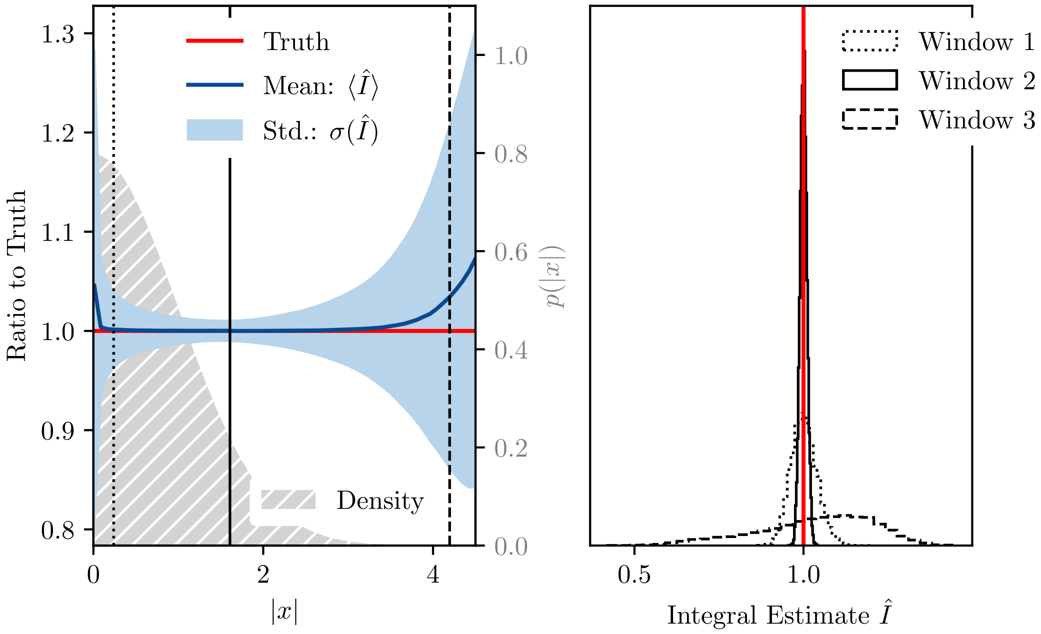


Figure 2.1: Demonstration of the reduced harmonic mean technique with the unit normal distribution. The left panel shows in blue the mean (solid line) and the region covering 68 % of results (from $1.5 \cdot 10^4$ repeated trials) of the integral estimates as a function of the window extent. The true integral value is 1.0, indicated in red. The gray shaded distribution shows the unit normal pdf (right y-axis scale). For the three window extents indicated by the black, vertical lines (Window 1: $|x| < 0.24$, Window 2: $|x| < 1.61$, Window 3: $|x| < 4.20$) the full distribution of integral estimates from the $1.5 \cdot 10^4$ trials are plotted on the right side as histograms.

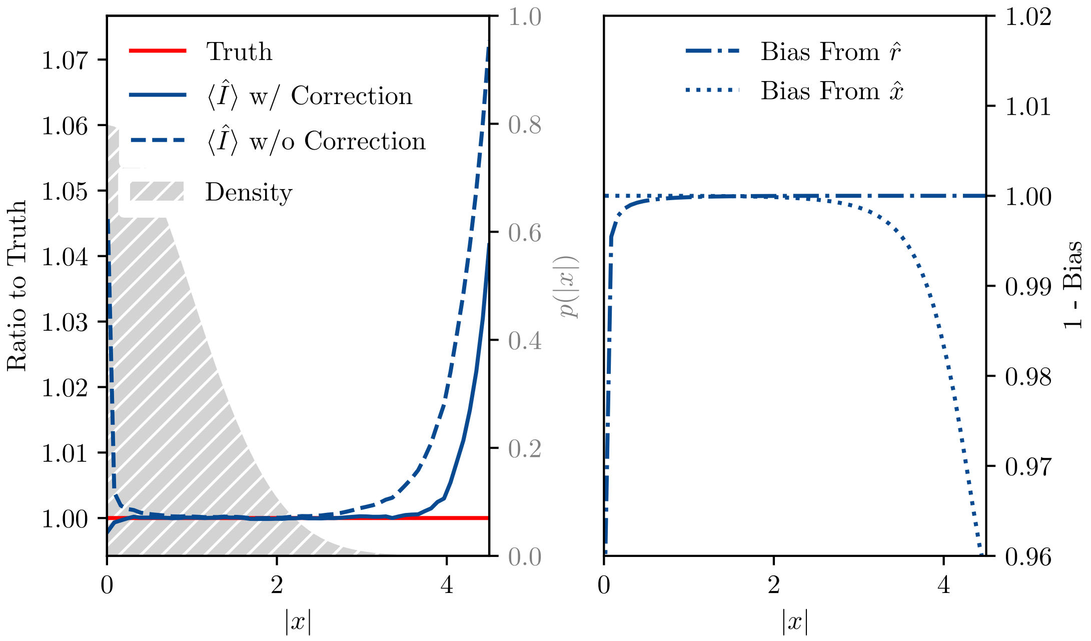


Figure 2.2: Left: The average uncorrected (Eq. 2.10) and corrected ($b \cdot \hat{I}$) integrals as a function of the extent of the window used to accept samples. The results are from averaging $1.5 \cdot 10^4$ integration results, where each integration test was performed from $3 \cdot 10^3$ samples in the full function range. Right: Individual contributions to the bias correction from the Binomial ($\hat{r}$) and the $1/f$ terms.

is $\hat{\sigma}_X/\hat{\mu}_X$.

The estimator $\hat{r}$ will also typically follow approximately a Normal distribution with parameters that can be estimated from *i.i.d.* sampling and Binomial statistics as

$$P(\hat{r}) \approx \mathcal{N}\left(\hat{\mu}_r = \hat{r}, \hat{\sigma}_r^2 = \frac{\hat{r}(1-\hat{r})}{N_\Omega}\right)$$

Since $\hat{r}$ also appears in the denominator, it will also produce a bias in our integral of size $\hat{\sigma}_r^2/\hat{\mu}_r^2$. The fractional uncertainty in our integral estimator from $\hat{r}$ is, in the approximation of *i.i.d.* sampling and a Normal distribution, $\hat{\sigma}_r/\hat{\mu}_r$.

We can therefore write down an explicit correction factor

$$b = (1 - \frac{\hat{\sigma}_X^2}{\hat{\mu}_X^2} - \frac{\hat{\sigma}_r^2}{\hat{\mu}_r^2}) \tag{2.12}$$

that we apply to the integral estimate $\hat{I}$ by multiplication.

The correction is illustrated in Fig. 2.2 using the same numerical experiments discussed in section 2.1.1. The resulting uncorrected and corrected average integral values are displayed. Focusing on small window sizes, it can be seen that the term from $\hat{r}$ dominates, as the binomial uncertainty is largest for small numbers of samples. With the bias correction applied, already for as few as ≈ 20 samples inside the integration volume (corresponding to roughly $|x| < 0.01$), accurate results are produced, while without the correction factor applied much large windows, starting at around $|x| < 0.5$, are necessary.

Towards larger windows the bias produced from $\hat{X}$ become dominant, as the range of values of f of the contained samples grows. The bias correction successfully mitigates this effect as illustrated.

Once the window size exceeds the space where samples are present, the integral starts diverging. For our example using $3 \cdot 10^3$ samples, we expect those to cover a region up to only $|x| \approx 3.58$, which explains well the observed trend.

Many samplers such as MCMC algorithms generate strong correlations amongst samples and using the binomial uncertainty discussed here can be inaccurate. We therefore also numerically evaluate the uncertainty as described in detail below. The integration regions are chosen such that the bias correction can be neglected.

2.1.2 An Adaptive Harmonic Mean Integration Algorithm

AHMI uses the HME on multiple subregions Δ_i to estimate the integral of $f(\boldsymbol{\lambda})$ over its full support Ω. In this subsection we present our example algorithm in detail and will show benchmark tests on several distributions in Sec. 2.1.3. As discussed previously, defining a set of suitable regions is crucial in obtaining a robust and unbiased estimate of the integral of $f(\boldsymbol{\lambda})$. In particular, to avoid biasing the result, it is essential not to use the same elements of the sample set $\{\boldsymbol{\lambda}\}$ for both the definition of Δ_i and estimates of the integral $\hat{I}_i$.

The general flow of the AHMI algorithm, including the procedure of defining the Δ_i, is summarized in Fig. 2.3. The various involved steps are discussed in more technical detail in the following subsections.

Samples, Preprocessing and Splitting

We start with a given set of samples $\{\boldsymbol{\lambda}\}$ that we assume are drawn according to the probability distribution proportional to our function $f(\boldsymbol{\lambda})$, obtained for example from MCMC sampling.

In order to de-correlate the sample space we apply a whitening transformation. In general, a whitening transformation maps a set of random variables with a known non-singular covariance matrix to a new set of variables with a covariance matrix equal to $\mathbb{I}$. A Cholesky Decomposition is used to whiten the samples, and the AHMI estimator for the integral becomes

$$\hat{I} = \frac{W_\Omega V'_\Delta}{\det R \cdot \sum_{\boldsymbol{\lambda}'_i \in \Delta} \frac{w_i}{f(\boldsymbol{\lambda}'_i)}} \tag{2.13}$$

where $\det R$ is the determinant of the whitening matrix and the primed symbols represent the quantities in the transformed space. In the following we drop this explicit addition of prime symbols and work in the whitened space (unless otherwise stated).

The full set of samples is then divided into two equally sized and mutually exclusive subsets A and B.

Hyper-rectangle Generation

We illustrate the hyper-rectangle generation steps in more detail using a two-dimensional Gaussian shell example with distribution

$$f(\boldsymbol{\lambda}|\boldsymbol{c}, r, \omega) = \frac{1}{\sqrt{2\pi\omega^2}} \exp\left(-\frac{(|\boldsymbol{\lambda} - \boldsymbol{c}| - r)^2}{2\omega^2}\right) . \tag{2.14}$$

In our examples, we use the following settings: radius $r = 5$, width $\omega = 2$ and $\boldsymbol{c} = \vec{0}$. The integration region extends from $[-25, 25]$ in each dimension. Samples from this distribution

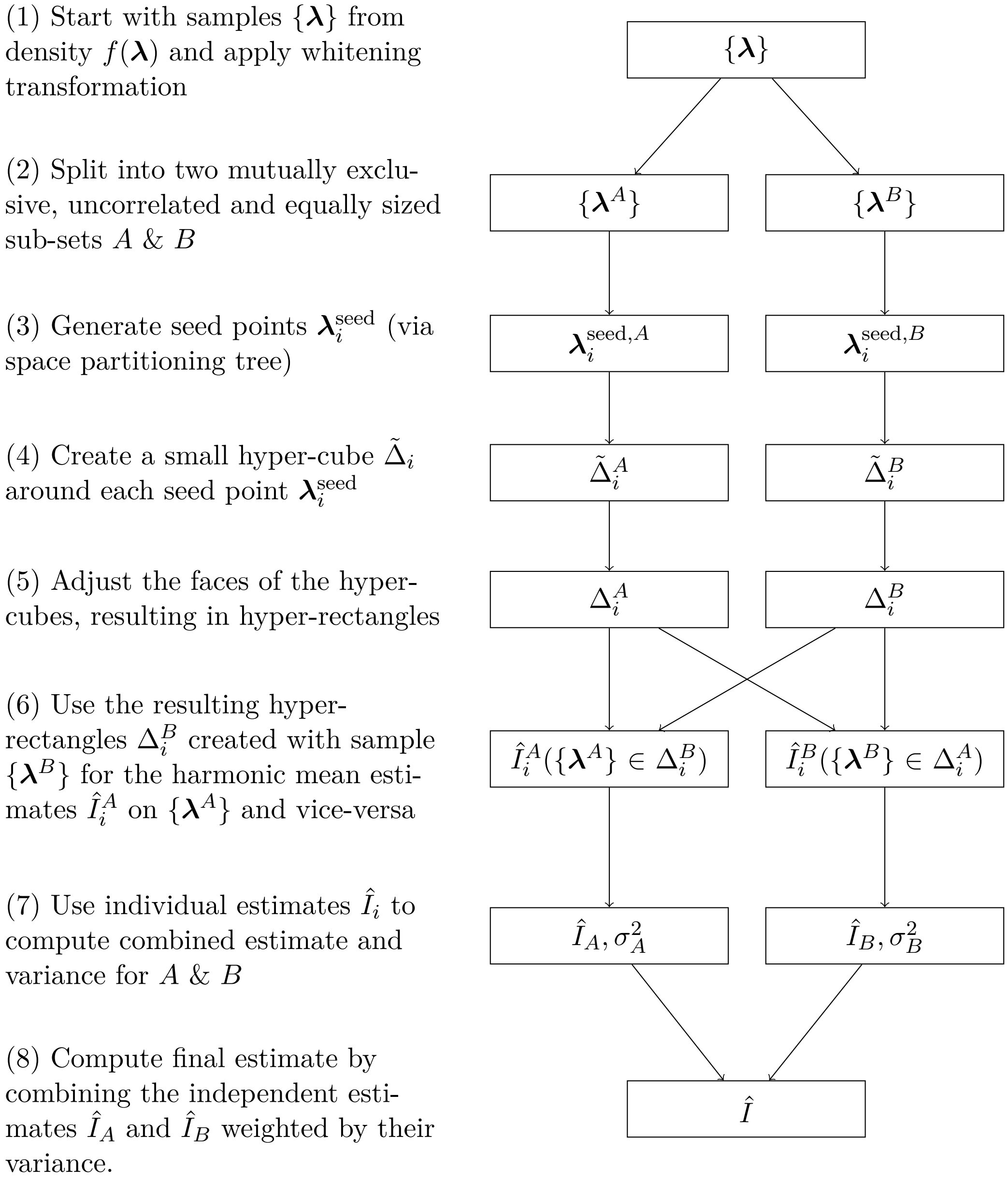
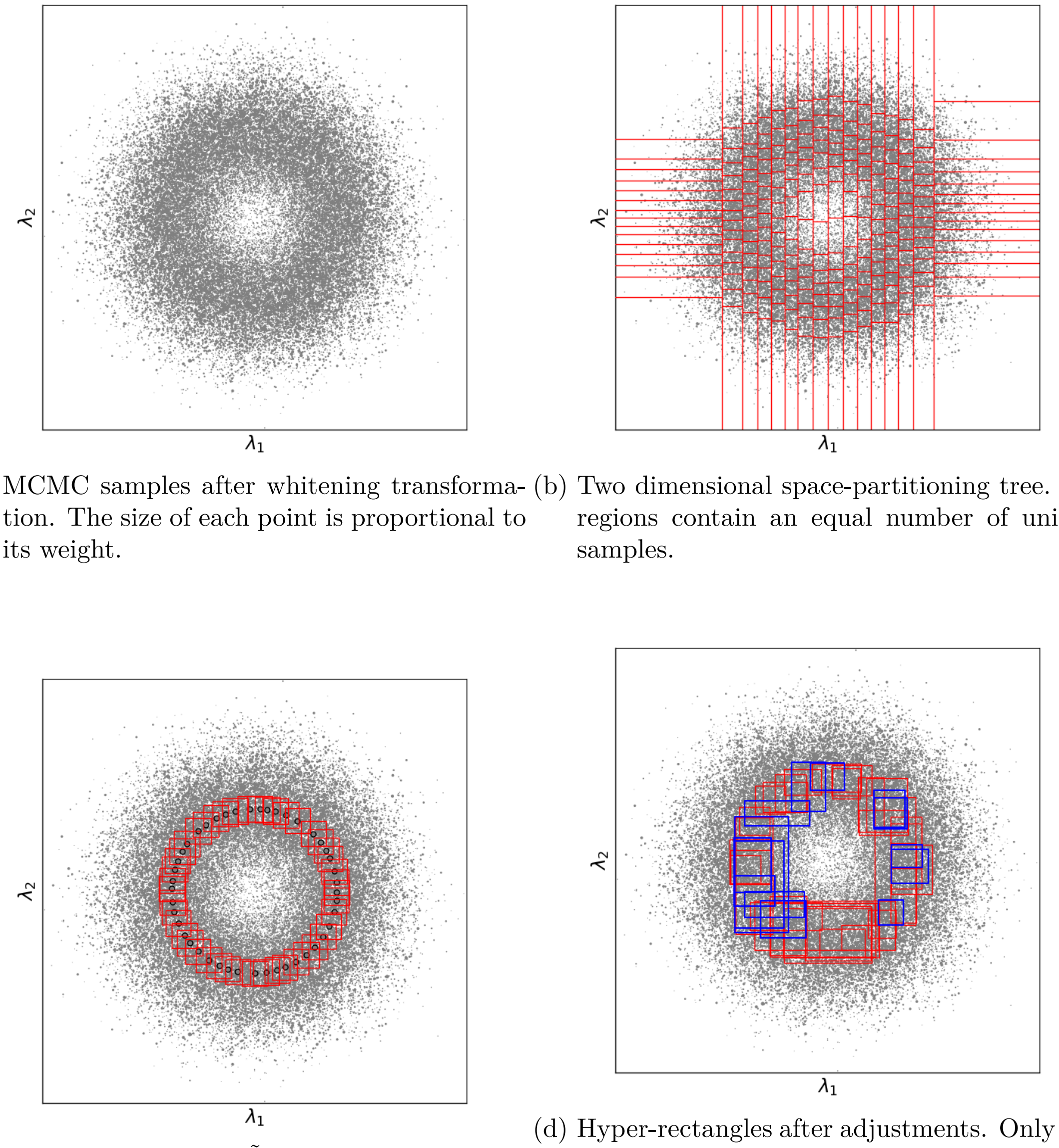

(1) Start with samples $\{\boldsymbol{\lambda}\}$ from density $f(\boldsymbol{\lambda})$ and apply whitening transformation
(2) Split into two mutually exclusive, uncorrelated and equally sized sub-sets A & B
(3) Generate seed points $\boldsymbol{\lambda}_i^{\text{seed}}$ (via space partitioning tree)
(4) Create a small hyper-cube $\tilde{\Delta}_i$ around each seed point $\boldsymbol{\lambda}_i^{\text{seed}}$
(5) Adjust the faces of the hyper-cubes, resulting in hyper-rectangles
(6) Use the resulting hyper-rectangles Δ_i^B created with sample $\{\boldsymbol{\lambda}^B\}$ for the harmonic mean estimates $\hat{I}_i^A$ on $\{\boldsymbol{\lambda}^A\}$ and vice-versa
(7) Use individual estimates $\hat{I}_i$ to compute combined estimate and variance for A & B
(8) Compute final estimate by combining the independent estimates $\hat{I}_A$ and $\hat{I}_B$ weighted by their variance.
$\{\boldsymbol{\lambda}\}$
$\{\boldsymbol{\lambda}^A\}$
$\{\boldsymbol{\lambda}^B\}$
$\boldsymbol{\lambda}_i^{\text{seed},A}$
$\boldsymbol{\lambda}_i^{\text{seed},B}$
$\tilde{\Delta}_i^A$
$\tilde{\Delta}_i^B$
Δ_i^A
Δ_i^B
$\hat{I}_i^A(\{\boldsymbol{\lambda}^A\} \in \Delta_i^B)$
$\hat{I}_i^B(\{\boldsymbol{\lambda}^B\} \in \Delta_i^A)$
$\hat{I}_A, \sigma_A^2$
$\hat{I}_B, \sigma_B^2$
$\hat{I}$

Figure 2.3: Overview of the different steps in the AHMI algorithm, including the procedure of finding subvolumes Δ_i and computing integral estimates $\hat{I}_i$.

are shown in Fig. 2.4a.

The algorithm starts by creating seed points around which to construct the integration regions Δ_i. These points should lie in areas of high density and should result in broadly distributed starting points.

In order to limit computation time, a simple space-partitioning tree is used to divide the whitened space into subsets of non-overlapping regions. A tree is constructed by performing cuts in every dimension in such a way to have an equal number of samples on the left and right leaves. The number of cuts in each parameter axes is determined by the total number of samples and the number of dimensions and is chosen in a way, that the number of samples in each leaf does not exceed 200. An example of such a space-partitioning tree is shown in Fig. 2.4b. For each partition i, the sample with the largest



(a) MCMC samples after whitening transformation. The size of each point is proportional to its weight.

(b) Two dimensional space-partitioning tree. All regions contain an equal number of unique samples.

(c) Initial hypercubes $\tilde{\Delta}$ around seed points.

(d) Hyper-rectangles after adjustments. Only the red hyper-rectangles are used in the final integral estimate.

Figure 2.4: Process of finding integration regions in a two dimensional example for the Gaussian shells test function.

function value contained in that partition is defined as seed point $\boldsymbol{\lambda}_i^{seed}$.

The following steps produce hyper-rectangle-shaped regions suitable for AHMI. In order to limit this variance and ensure numerical stability, the ratio between the highest and the lowest probability of samples inside a hyper-rectangle is bound by the following condition:

$$\frac{f_{\max}}{f_{\min}} \leq t \; . \tag{2.15}$$

Although this threshold is user-selectable, by default it is equal to 500 (an example of integration with different threshold values is shown in Sec. 2.1.3).

To create M regions for integration, we select the seed point $\boldsymbol{\lambda}_i^{\text{seed}}$ with the overall largest value $f(\boldsymbol{\lambda}_i^{\text{seed}})$ and follow the steps below. Then we recursively repeat the same procedure $M-1$ times using the remaining seed points.

1. The algorithm starts by building a small hyper-cube $\tilde{\Delta}_i$ around the selected seed point $\boldsymbol{\lambda}_i^{\text{seed}}$, see Fig. 2.4c.

2. This hyper-cube is then incrementally either increased or decreased in size, until the probability ratio of contained samples matches the threshold t within some tolerance, or until it contains more than one percent of the total samples.

3. The faces of the hyper-cube are then iteratively adjusted (expand or contract), to adapt to the density of the contained samples, while enforcing the condition $f_{max}/f_{min} \leq t$. This step turns the d-dimensional hyper-cube into a d-dimensional hyper-rectangle. This hyper-rectangle adaptation algorithm continues as long as changes to the hyper-rectangle's faces are accepted. The stopping criterion is based on the fraction of samples accepted or rejected compared to expectation from the volume change. However, the hyper-rectangle adaption algorithm always ensures that no modification to the hyper-rectangle's faces are made if such a modification would result in $f_{\max}/f_{\min} > t$.

Figure 2.4d shows the resulting set of M hyper-rectangles for our example. A detailed description of the rectangle optimization procedure can be found in [76].

Integral Estimates

Once M integration regions are defined, we can compute the integral estimates $\hat{I}_i^A$ for each Δ_i^B, according to Eq. 2.11. The procedure is the same for $\hat{I}_i^B$, so we shall drop the superscripts A and B in the following. The two resulting, separate estimates $\hat{I}^A$ and $\hat{I}^B$ will then be combined to obtain the final estimate.

From the distribution of all estimates $\hat{I}_i$ we select only the 68% central percentile to reject outliers—a procedure that was empirically found to work well. This is indicated in Fig. 2.4d labeled as 'accepted' and 'rejected' rectangles. We proceed to combine the remaining estimates $\hat{I}_i$ into a single estimate $\hat{I}$ using a robust and unbiased estimator for the combination of correlated measurements as suggested in [77].

$$\hat{I} = \sum_i w_i \hat{I}_i \qquad\qquad \sigma^2(\hat{I}) = \sum_{i,j} w_i w_j \bar{\sigma}_{ij}, \tag{2.16}$$

where the weights w_i are defined as:

$$w_i = \frac{\frac{1}{\bar{\sigma}_i^2}}{\sum_j \frac{1}{\bar{\sigma}_j^2}} . \tag{2.17}$$

The variances $\bar{\sigma}_i^2 \equiv \bar{\sigma}_{ii}$ and covariances $\bar{\sigma}_{ij}$ of the mean assigned to integration regions Δ_i and Δ_j are estimated in the next section.

Covariance Estimate

The following procedure is used to estimate the covariance between individual integral estimates $\hat{I}_i$:

1. We partition $\{\boldsymbol{\lambda}\}$ into a number S of subsets $\{\boldsymbol{\lambda}_1\}$, $\{\boldsymbol{\lambda}_2\}$ ... $\{\boldsymbol{\lambda}_S\}$, chosen in a way that reduces their correlation. The default value for S is 10.

2. Separate estimates $\hat{I}_{i,k}$ (k indexes the S partitions) of the integral are then performed for all sample subsets $\{\boldsymbol{\lambda}_k\}$ resulting in S integral estimates for each subspace Δ_i:

$$\begin{bmatrix}\hat{I}_{i,1} & \hat{I}_{i,2} & ... & \hat{I}_{i,S}\end{bmatrix}$$

3. The covariance of the separate integral values then is

$$\sigma_{ij}^2 = \frac{1}{S-1}\sum_{k=1}^{S}(\hat{I}_{i,k} - \bar{I}_i)(\hat{I}_{j,k} - \bar{I}_j)$$

with

$$\bar{I}_i = \frac{1}{S}\sum_{k=1}^{S}\hat{I}_{i,k}$$

4. The estimate of the covariance of $\bar{I}_i \approx \hat{I}_i$ and $\bar{I}_j \approx \hat{I}_j$ is then $\bar{\sigma}_{ij}^2 = \frac{\sigma_{ij}^2}{S}$.

Final AHMI Integral Estimate

As we have now obtained two values of $\hat{I}$ and two variances $\sigma^2(\hat{I})$ from the two sets $\{\boldsymbol{\lambda}^A\}$ and $\{\boldsymbol{\lambda}^B\}$, we combine those into the final result like

$$\hat{I} = \frac{\hat{I}^A/\sigma_A^2 + \hat{I}^B/\sigma_B^2}{1/\sigma_A^2 + 1/\sigma_B^2}$$

with variance estimate

$$\sigma^2 = \left(\frac{1}{\sigma_A^2} + \frac{1}{\sigma_B^2}\right)^{-1}.$$

2.1.3 Benchmark Examples

To validate our algorithm, we apply it to estimate the integral of several test functions in varying dimensionality (up to $d = 25$) for which an analytic (or accurate numerical) solution of the integral value is available.

The test functions were chosen to pose different challenges to the algorithm. For our first test problem, we start with the canonical example of a multivariate normal distribution. For the second test case we look at Gaussian shells, for which the mode of the distribution does not lie on a single point but has infinite modes on $d-1$-dimensional surface. Next we

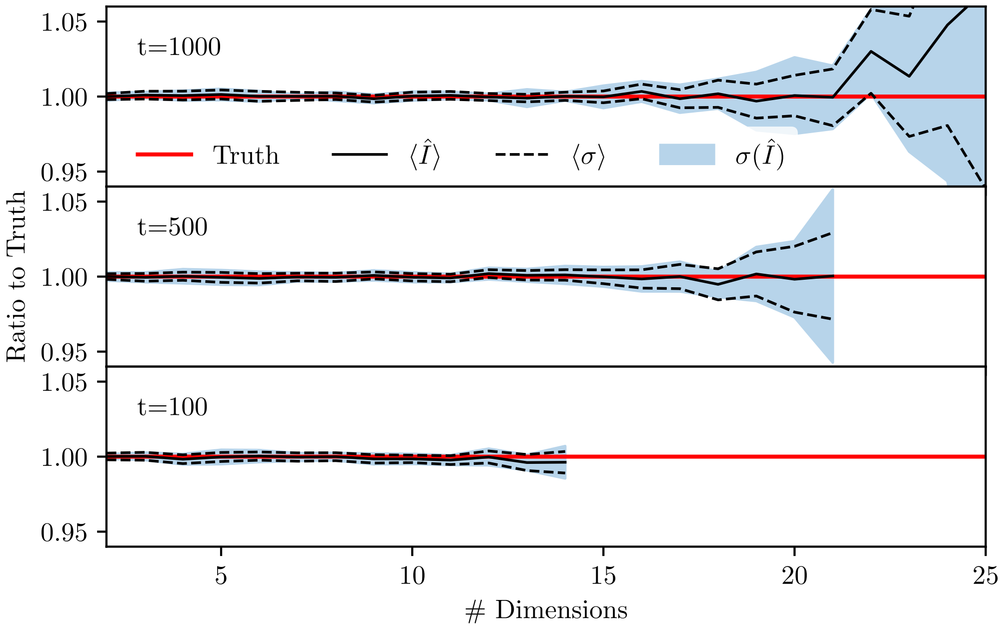


Figure 2.5: AHMI value for the multivariate normal distribution as ratio to the true integral shown as a function of dimensionality for three threshold values $t = [100, 500, 1000]$. The solid black lines give the mean result over ten independent trials and the shaded bands show the standard deviation of these trials. The dashed lines show the average errors reported by AHMI. Samples are obtained from *i.i.d.* sampling, 10^6 for each run.

study the heavy-tailed Cauchy distribution with multiple modes, and in the end explore the asymmetric 'Funnel' function. Additional information of the number of integration volumes used and the computation time are only provided for the first example, as these are very similar for the other three examples.

The samples $\{\boldsymbol{\lambda}\}$ on which our integration is based are obtained from Metropolis-Hastings MCMC, and in the case of the multivariate normal from *i.i.d.* sampling. The sample size is fixed to $2 \cdot 10^6$ for Gaussian shells example and it is equal to 10^6 for other examples.

Multivariate Normal Distribution

The first test case is a unit normal distribution (centered at zero, width one) in two up to 25 dimensions. The samples input to AHMI are obtained from *i.i.d.* sampling. The resulting integral estimates are shown in Fig. 2.5 as a function of the dimensionality for three different threshold values $t = [100, 500, 1000]$.

For both threshold values, $t = [500, 1000]$, we get unbiased and consistent results up to around 21 dimensions, after which results start to become positively biased for $t = 1000$. This bias seems to be intrinsic to the method itself or our implementation of the algorithm and will be the subject of future studies. The coverage for one standard error for $t = 500$ (220 trials) is 0.52 ± 0.03. For the threshold value $t = 100$ we get an unbiased integral estimate up to 14 dimensions, and after that hyper-rectangles can no longer be created. The default value of $t = 500$ was used for the remaining examples below.

In Fig. 2.6 we show the execution time of the algorithm, and the number of hyper-

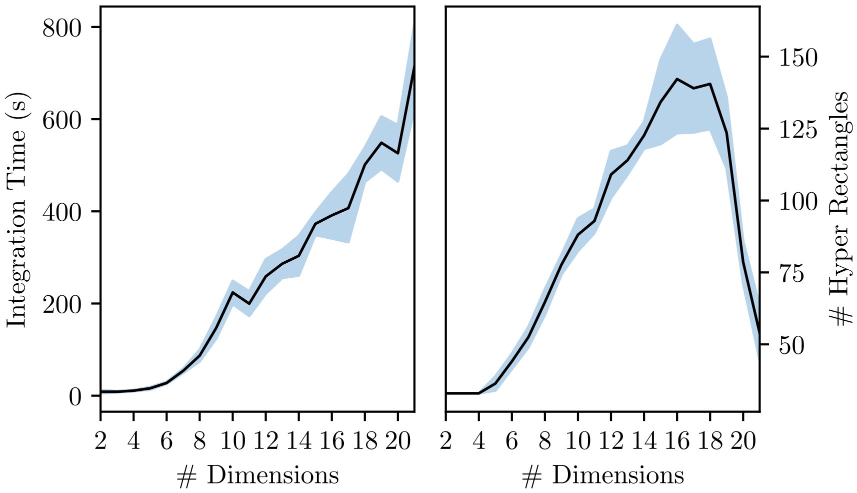


Figure 2.6: Left: Average AHMI execution time for the multivariate normal distribution ($t = 500$) in total CPU seconds (run on a system with a 2.3 GHz Intel Xeon 6140 processor, SPECrate2017-FP equivalent ca. 180) as a function of dimensionality. Right: Average number of hyper-rectangles used by AHMI for computing the integral estimate for the multivariate normal distribution as a function of dimensionality.

rectangles used for integration for the threshold value $t = 500$. The execution time rises with the number of dimensions almost linearly. The change in slope at low dimensionality is likely due to CPU caching behaviour. The number of hyper-rectangles starts to decay after 18 dimensions indicating that there exist fewer hyper-rectangles that satisfy Eq. 2.15.

Gaussian Shell Distribution

The functional form was given in Eq. 2.14 and an example distribution in the first two dimensions is shown in Fig. 2.7. The AHMI algorithm results (Fig. 2.8) shows a similar behaviour as for the multivariate normal distribution for this more complicated test function. However, integration was possible up to 17 dimensions. Up to that point the integral estimates, including errors, are well behaved, with a coverage for one standard error of 0.48 ± 0.04 (160 trials).

Multimodal Cauchy Distribution

The Cauchy distribution, with its heavy tails, is a notoriously difficult problem and used here to point out possible weaknesses of our algorithm. We further increase complexity for the hyper-volume creation process by using four separate, shifted Cauchy distributions creating multiple modes. The functional form can be written as

$$f(\boldsymbol{\lambda}) = \prod_{i=1}^{2} \frac{1}{2} \left[\text{Cauchy}(\lambda_i \mid \mu, \sigma) + \text{Cauchy}(\lambda_i \mid -\mu, \sigma)\right] \cdot \prod_{j=3}^{d} \text{Cauchy}(\lambda_j \mid 0, \sigma), \qquad (2.18)$$

where $\mu = 1, \sigma = 0.2$ and n is a dimensionality of $\boldsymbol{\lambda}$. An example of this target distributions is provided in Fig. 2.9. The integration region extends from $[-8, 8]$ in each dimension. Our results are collected in Fig. 2.10 and indicate that integration is possible up to seven

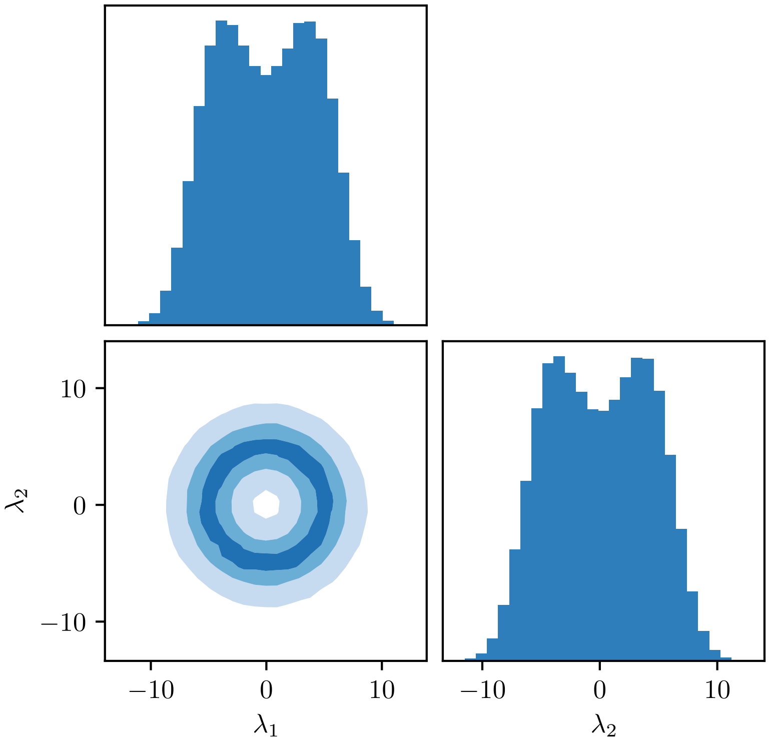


Figure 2.7: One and two dimensional distributions of samples along the first two dimensions of the Gaussian shell target function.

dimensions, given the fixed sample size of 10^6. For this range, the AHMI results are very reliable. The coverage (120 trials, one standard error) for this function is 0.42 ± 0.05.

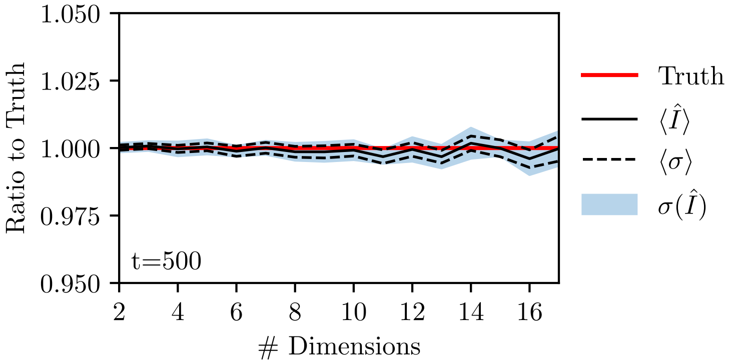


Figure 2.8: AHMI value for the Gaussian shell distribution as ratio to the true integral shown as a function of dimensionality. The solid line gives the mean result over ten independent trials and the shaded band show the standard deviation of these trials. The dashed lines show the average errors reported by AHMI. The samples are obtained from Metropolis-Hastings MCMC and the sample size is $2 \cdot 10^6$.

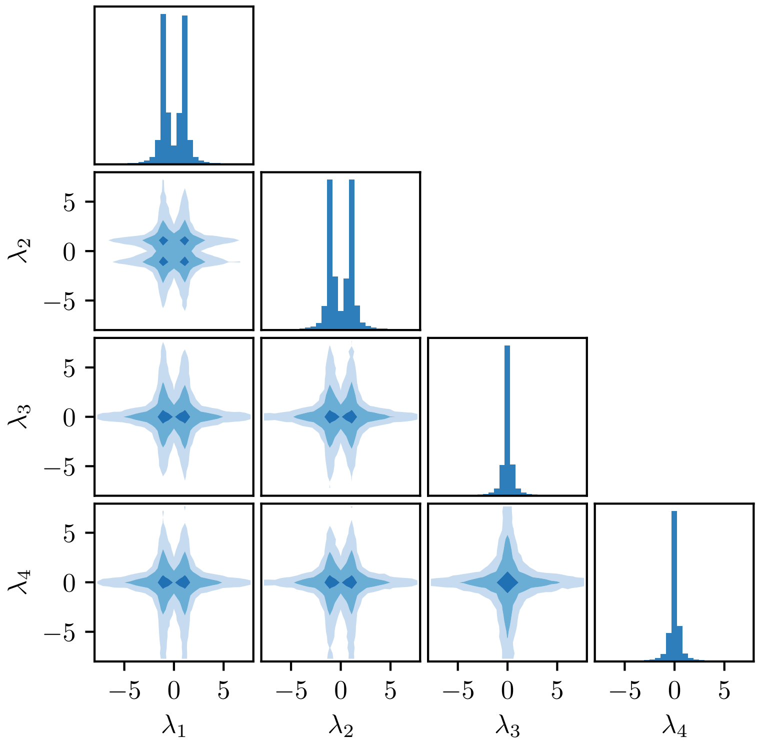


Figure 2.9: One and two dimensional distributions of samples along the first four dimensions of the multimodal Cauchy target function.

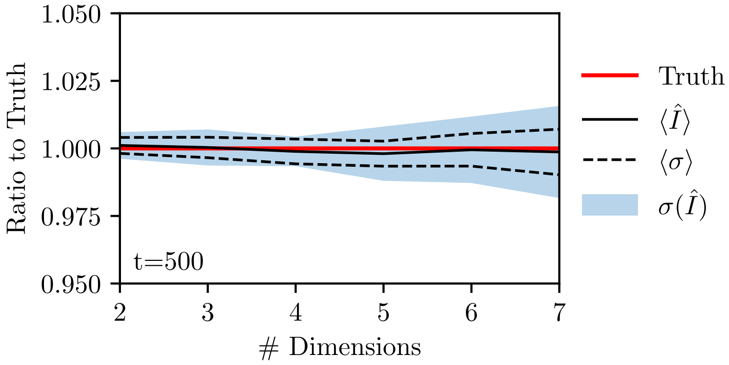


Figure 2.10: AHMI value for the multimodal Cauchy distribution as ratio to the true integral shown as a function of dimensionality. The solid line gives the mean result over twenty independent trials and the shaded band show the standard deviation of these trials. The dashed lines show the average errors reported by AHMI.

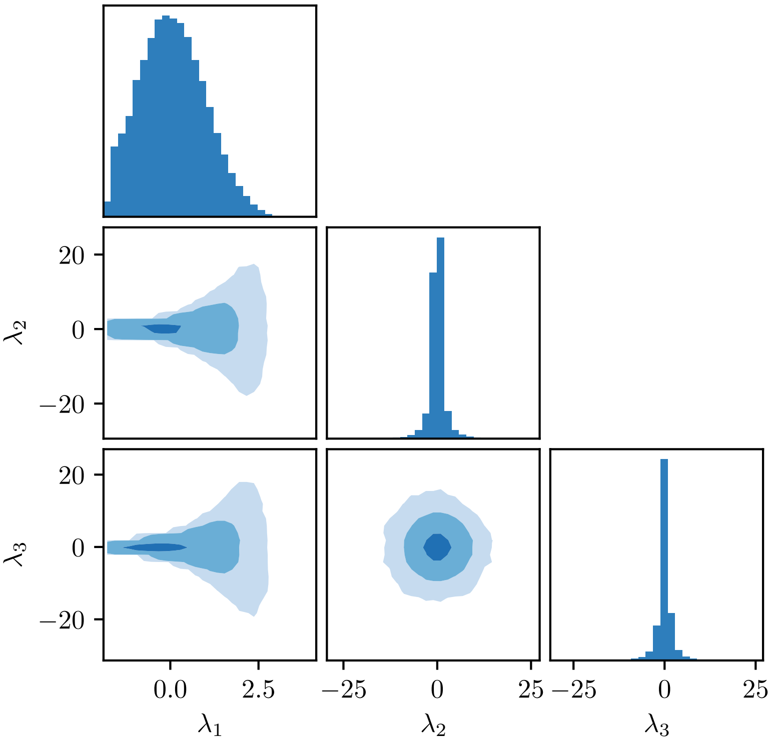


Figure 2.11: One and two dimensional distributions of samples along the first three dimensions of the Funnel target function.

Funnel Distribution

The final problem we study is the so-called 'Funnel' distribution, that is described in [78]. The functional form of this distribution can be written as

$$f(\boldsymbol{\lambda}) = \mathcal{N}\left(\lambda_1 \mid 0, a^2\right) \prod_{i=2}^{d} \mathcal{N}\left(\lambda_i \mid 0, \exp\left(2b\lambda_1\right)\right), \tag{2.19}$$

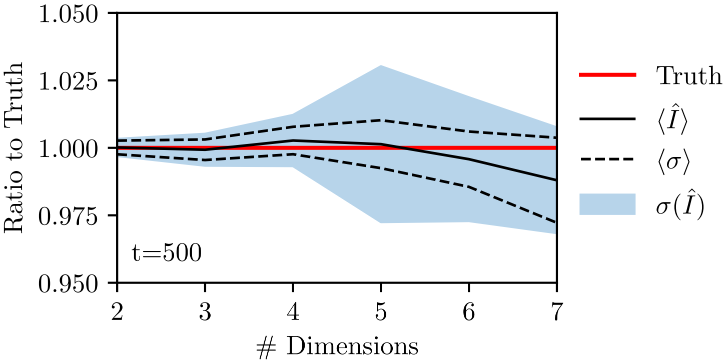


Figure 2.12: AHMI value for the 'Funnel' distribution as ratio to the true integral shown as a function of dimensionality. The solid line gives the mean result over twenty independent trials, with the shaded band showing the standard deviation of these trials. The dashed lines show the average errors reported by AHMI.

where $a = 1$, $b = 0.5$ and n is a dimensionality of $\boldsymbol{\lambda}$. An example of the distribution in its first three dimensions on the parameter range $[-50, 50]$ is provided in Fig. 2.11. The results (Fig. 2.12) show a similar performance as for the previous example with reliable estimates up to seven dimensions, with a coverage of 0.41 ± 0.04 (120 trials, one standard error).

2.1.4 Harmonic Mean with Spherical Volumes

As shown in the previous section, there is a limit on the number of dimensions for which one can use the AHMI integration with rectangular volumes and obtain accurate results. The integration algorithm fails for problems with more than 10-20 dimensions depending on the complexity of the target density, and it shows a non-linear increase in computational time. The reason for this is a curse of dimensionality that makes the use of hyperrectangular integration volumes inefficient. An approach to improve the AHMI algorithm has been investigated by J. Thiel in the scope of the Bachelor's thesis [79] that I co-supervised. The idea and key results are discussed briefly.

Often, typical target densities that one is interested in sampling and integrating are spherical-like or can be partitioned and transformed to be such. Fitting them with hyper rectangles is not efficient because most of the volume of the hyper rectangle is located in its corners, whose number scales non-linearly with the number of dimensions. To improve integration accuracy, one can use integration volumes in the form of spheres or spherical shells. In the case of a spherical volume, one has to tune only one parameter, i.e., the radius of the sphere. For the spherical shell, two parameters should be tuned, i.e., the inner and outer radius. To ensure that the samples can be efficiently fitted with hyperspheres, the samples of the target density should be whitened (see Sec. 2.1.2).

A comparison of the performance of the AHMI integration with hyper rectangles and hyperspheres is presented in Fig. 2.13. We draw 10^5 *i.i.d.* samples from the multivariate normal distribution with unitary covariance for a number of dimensions, d, in the range from 0 to 50. For each d, we evaluate the AHMI integral using different sizes of the hypervolume. As Fig. 2.13 shows, the hyperrectangular integration volume has limited applicability to problems with up to 20-30 dimensions. After that, the integral estimate tends to overestimate the true value significantly. In contrast to this, hyperspheres allow obtaining accurate integral estimates for the whole range of dimensions, i.e., from 0 o 50. Moreover, spherical volumes give a much wider range of optimal size parameters that result in an accurate integral estimate compared to hyperrectangular volumes. This shows that spherical integration volumes are more suitable for integration than rectangular volumes.

Finding an optimal integration volume is, in general, a challenging task. Adjustment of the hyperrectangular volumes requires tuning each edge separately, and this becomes extremely computationally costly with an increasing number of dimensions. In the case of spherical volumes, one needs to solve only a one-dimensional optimization problem, in which optimal radius size has to be obtained that satisfies a predetermined log-likelihood ratio (see Eq. 2.15). The following algorithm has been proposed in [79] to estimate best parameters for spherical shell:

1. Whiten samples to have the identity covariance matrix.
2. Compute radial distribution function of samples with respect to the mean of the sample.
3. Select ball with a certain fraction of samples inside (equivalent to the spherical shell with inner radius zero).

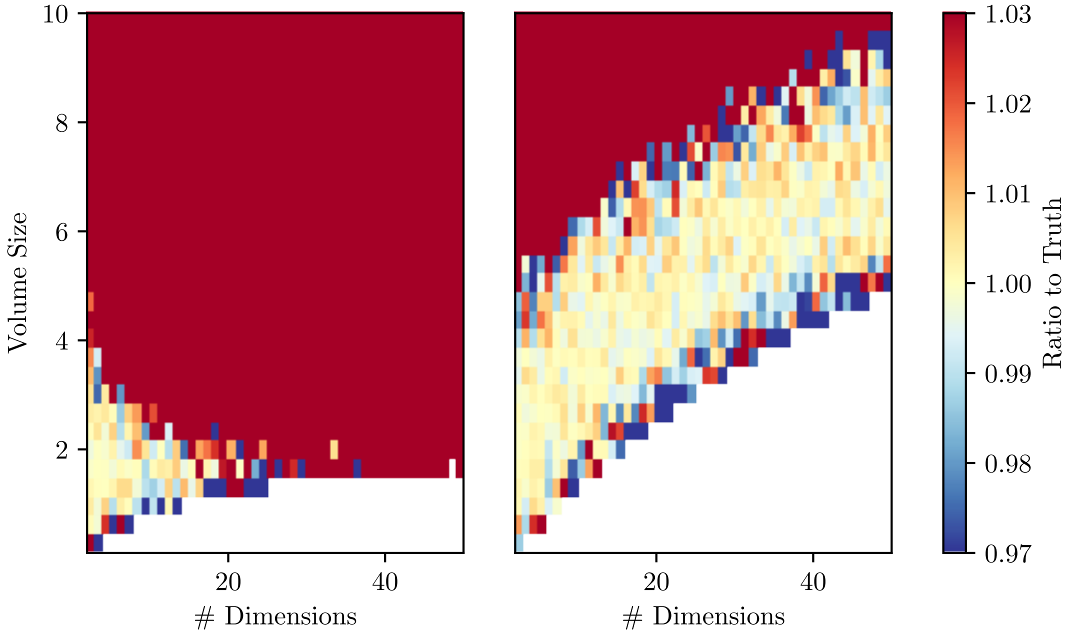


Figure 2.13: Integral estimates of the multivariate normal distribution using HM with rectangular (left) and spherical (right) volumes. The volume size denoted a half of the rectangle edge for rectangular volumes and the radius for spherical volume. Color coding represents deviations from the truth. The results are averaged over 30 independent trials.

4. Decrease volume to lower likelihood ratio, t, by removing 1% of samples from inner/outer radius of spherical shell separately and calculate the reduction in t for both options.

5. Apply that volume change that gives more reduction in t.

6. Repeat steps 4 and 5 until t is smaller than the given threshold or a maximum number of iterations is reached.

With this procedure, one can find a subset of initial integration volume that has the largest fraction of samples inside and satisfies the required log-likelihood ratio.

A difficulty with this approach arises when one uses correlated samples instead of *i.i.d.*. The algorithms for a whitening transformation are known to perform poorly when the number of dimensions is large, or the effective number of samples is small [80]. They introduce highly biased eigenvalues of the sample covariance matrix, and the inversion of the matrix gets numerically unstable. As seen from Eq. 2.13, the determinant of the whitening transformation appears in the denominator of the harmonic mean estimate, and this produces an unstable integral estimate. To overcome this problem, alternative techniques for the whitening transformation should be considered, such as Factor Analysis, LW-Shrinkage, or Nonlinear Shrinkage [80]. Although they provide an improved transformation matrix, they require additional — often significant — computational time.

2.1.5 AHMI Conclusions

We have developed an Adaptive Harmonic Mean Integration (AHMI) algorithm that can be used to integrate a non-normalized density function using the samples drawn according to the probability distribution proportional to the function. The fundamental assumption is that the sampling algorithm has faithfully produced samples from this distribution. Given this, the AHMI algorithm can be used to produce both an estimate of the integral of the function over its full support as well as an estimate of the uncertainty of the integral. In this first implementation of the AHMI algorithm, finite hyper-rectangles are generated in the whitened space of the samples covering the full support. The adaptive algorithm ensures that the range of function values enclosed by the hyper-rectangles is limited such that the variance of the integral results are moderate. This allows for reliable results both for the integral values as well as for reliable uncertainty estimates.

The algorithm has been tested on a number of examples and found to produce reliable and unbiased integral estimates up to around twenty dimensions for the first two test problems, and up to seven for the latter two, more difficult test cases. The reported errors provide a useful measure of uncertainty, while slightly under covering at around 40-50% (expecting 68%). The use of hyper-rectangles however limits the applicability to a not-too-large number of dimensions (≈ 20 in the case of the multivariate normal distribution) because a large fraction of the volume is in the corners of the hyper-rectangles. The AHMI algorithm with spherical volumes has demonstrated much better performance on a toy problem compared to the integration with rectangular volumes. However, to increase the applicability of AHMI integration to a larger number of dimensions, an accurate estimation of the whitening transformation is needed.

2.2 Sampling Parallelization via Space Partitioning

Now, as we discussed an approach to normalize a target density function using samples of the function, we can use it to improve the performance of MCMC samplers on multimodal target densities. The text presented in this section closely follows [64].

2.2.1 Algorithm Overview

As in the previous section, we consider generating samples according to a target density function $f(\boldsymbol{\lambda})$ where $\boldsymbol{\lambda} \in \mathbb{R}^d$ and Ω is the support of the function. To illustrate key steps of our method, we will use as example the sum of four bivariate normal distributions with $\boldsymbol{\lambda} = (\lambda_1, \lambda_2)$:

$$f(\boldsymbol{\lambda}) = \sum_{i=1}^{4} a_i \cdot \mathcal{N}(\boldsymbol{\lambda}|\mu_i, \Sigma_i), \tag{2.20}$$

where $a_1 = a_2 = 0.48$, $a_3 = a_4 = 0.02$, $\mu_i = (\pm 3.5, \pm 3.5)$, $\Sigma_1 = \Sigma_2 = (0.33, 0.17; 0.17, 0.33)$, and $\Sigma_3 = \Sigma_4 = (0.019, -0.003; -0.003, 0.017)$. Each bivariate normal distribution is individually normalized. Two, in the upper-right and lower-left quadrants, have large weights (0.48) and the other two, in the other quadrants, have small weights (0.02). The covariances are relatively small compared to the separations of the modes, making this a challenging target distribution to sample from for many MCMC algorithms. Probability contours of this test function are shown in Fig. 2.14-1.

Our approach consists of the following four steps illustrated in Fig. 2.14:

1. Generate a set of N_{exp} exploration samples $\{\boldsymbol{\lambda}_i^*\}_{i=1..N_{exp}} \in \Omega$, distributed amongst N_{chains}, where N_{exp} is a small number compared to the desired number of final MCMC samples and N_{chains} is the number of chains. The chains should have different (possibly randomly chosen) starting points and can be run in parallel. The samples are used to find regions of the parameter space with a high density and the MCMC chains are not required to converge. An initial sampling of our example function with $N_{exp} = 500$ generated using $N_{chains} = 25$ with 20 samples per chain is shown in Figure 2.14-1.

2. Partition the parameter space into N_{sp} mutually exclusive subspaces

$$\{\omega_k\}_{k=1..N_{sp}} \in \Omega \tag{2.21}$$

in such a way that $\cup\omega_k = \Omega$, $\omega_k \cap \omega_m = \varnothing$ if $k \neq m$ (see also Figure 2.14-2). While in general, the boundaries of the subspaces could be arbitrary shapes, in the following, 'N space partitions' will refer to N cuts along different, single parameter axes. This splits the parameter space into $N + 1$ rectangular subspaces.

3. Generate N_{samp}^k samples $\left\{\boldsymbol{\lambda}_i^k\right\}_{i=1..N_{samp}^k} \in \omega_k$ in each subspace k with the distribution proportional to $f(\boldsymbol{\lambda})$ using a sampling algorithm of choice (see Figure 2.14-3). Note that each sampler has to perform its burn-in cycle only in the reduced subspace ω_k, which significantly reduces tuning time. If the sampling algorithm failed for some reason in one of the subspaces, e.g., ω_j, generate an additional partition in that subspace using the existing samples N_{samp}^j. Afterward, generate new samples in each of two resulting subspaces, $\omega_{j,1}, \omega_{j,2}$. Repeat this procedure until the convergence criteria is passed in each subspace or a maximum number of recursive iterations is reached.

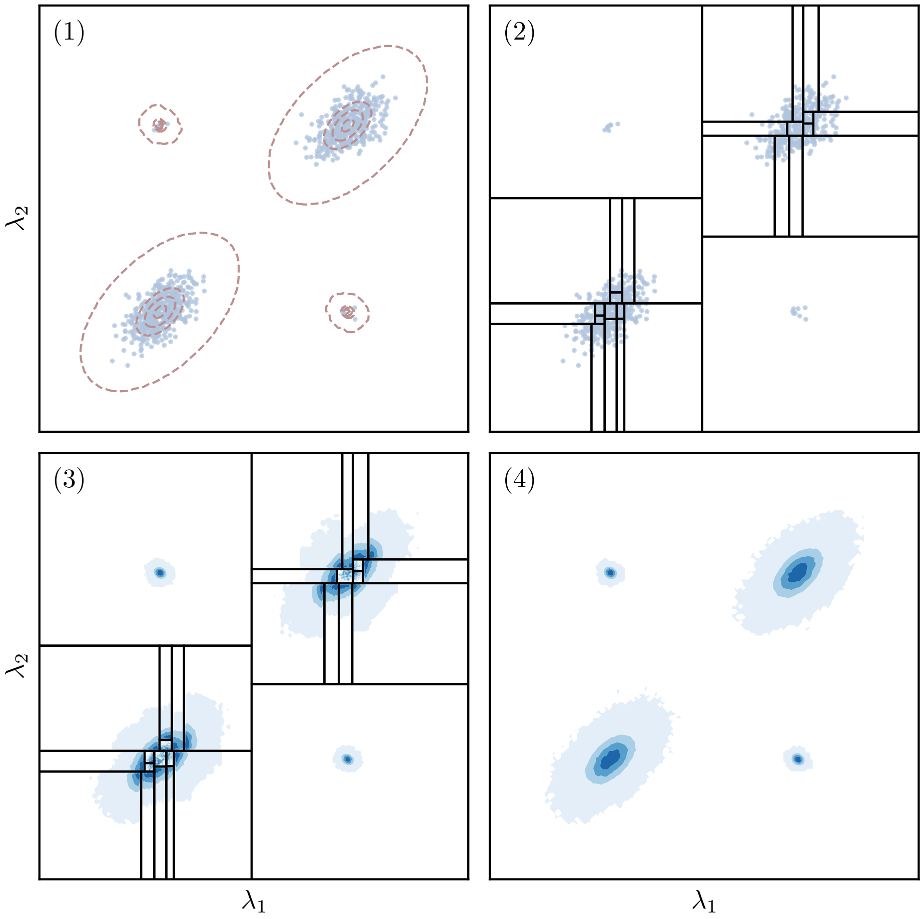


Figure 2.14: The subplots show the key steps in the partitioned sampling approach: (1) The 500 exploration samples are generated from the target density. The red dashed lines demonstrate contours of the true density. (2) The parameter space is partitioned into 30 subspaces. The black lines demonstrate the boundaries of subspaces. (3) The 10^4 samples are generated in each subspace from the individual MCMC chains. (4) The samples are reweighted in the full space.

4. Determine the integrated density of the target distribution in each subspace by computing $I_k = \int_{\omega_k} f(\boldsymbol{\lambda}) d\boldsymbol{\lambda}$ and assign the following weights to the sample of subspace k

$$w_k \propto \frac{I_k}{N^k_{samp}}.$$

5. Stitch the now weighted samples together resulting in the final sampling distribution (see Figure 2.14-4).

There are many ways of implementing the described idea based on choices of samplers, integrators and space partitioning strategies. In the following, we describe our implementation that is also made available in the BAT.jl package (see Section 2.3).

2.2.2 Implementation

Exploration samples

Exploration samples play an important role in this algorithm, since the parameter space is partitioned based on them. If exploration samples represent the structure of the target density closely enough — for example indicating the presence of multiple modes by clusters of spatially neighboring points — then the space partitioning algorithm can capture these features and generate partitions in such a way to split those clusters. This simplifies the target density in each subspace and thus allows for much faster burn-in and tuning procedures. In our implementation, we generate exploration samples by running a large number of MCMC chains, where each chain generates a few hundred samples. There is no tuning or convergence requirement for these chains, but a small set of samples are initially used to set the parameters of the proposal functions for each chain. Some knowledge of the form of the target distribution is useful in determining how many chains and how many samples will be necessary. While the morphology of the resulting sample clouds should resemble that of the target density as closely as possible, this initial exploration should be fast compared to the following sampling time in the partitioned space.

Space Partitioning

Given the discussed exploration samples, we partition our parameter space into rectangular subspaces in such a way as to split clusters of spatially neighboring samples. To do so, a binary tree is used where each node is determined by a cut that is orthogonal to parameter axes. For the sake of illustration, we consider a one-dimensional problem with the samples $\{\lambda\}$ shown in Fig 2.15. The cut position perpendicular to the λ axis is denoted as $\widetilde{\lambda}$ and it is selected by finding the minimum of the following cost function:

$$\widetilde{\lambda} = \inf_{a} \{W(a, \lambda)\} = \inf_{a} \left\{ \sum_{\lambda_i < a} |\lambda_i - \langle \lambda \rangle_{\lambda < a}|^2 + \sum_{\lambda_i > a} |\lambda_i - \langle \lambda \rangle_{\lambda > a}|^2 \right\}, \tag{2.22}$$

where $\langle \lambda \rangle$ denote the mean of samples. This process is then repeated iteratively resulting in the desired number of partitions.

The blue lines in Figure 2.15 demonstrate how this cost function depends on the cut positions for 3 partitioning steps. The partitioning procedure is ended when a minimal change in the cost function results from further partitioning or a maximum number of subspaces is reached. The evolution of the cost function for our example is also shown in Figure 2.15.

The partitioning procedure is analogous for higher dimensional space. If, for instance, a sample vector is d-dimensional, then we evaluate Equation 2.22 for every dimension which results in proposed cut positions $\widetilde{\lambda}_i$ with corresponding cost values $W_i(a_i, \widetilde{\lambda}_i)$ for dimension $i = (1, ..., d)$. The minimum cost value is selected and the cut along the corresponding dimension is accepted. Additionally, if preliminary knowledge about the structure of the target density is present, the user can specify manually along which parameters partitioning of the parameter space should be performed.

Sampling

Sampling in the subspaces is performed independently and does not require communication between MCMC processes. It can therefore be trivially divided into tasks and executed in parallel on multiple processors using distributed computing. In the following, we define a

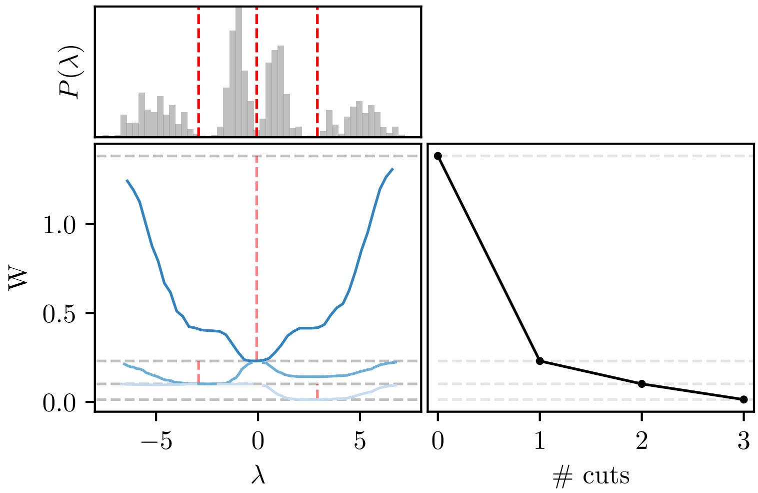


Figure 2.15: Illustration of the space partitioning algorithm using $5 \cdot 10^3$ one-dimensional exploration samples λ^* with a distribution demonstrated in the upper left histogram. The red dashed lines demonstrate the first three cut positions $\widetilde{\lambda}$. The blue lines show the value of $W(a, \lambda)$ as a function of the cut position for 3 iterations of space partitioning. The gray dashed lines show the value of the cost function at its minimum for each iteration. The bottom right subplot illustrates the dependence of W as a function of the number of cuts.

'worker' as a computing unit (this can either be a node of a cluster, networked machine, or a single machine) that consists of multiple CPU cores used to perform one task; i.e., sample the target density in one subspace. All the cores that belong to one worker are called threads and are used to run multiple MCMC chains in parallel within one subspace. Running multiple chains within one subspace is necessary for determining convergence of the MCMC process. Our implementation allows running subspaces on multiple remote hosts using Julia's support for compute clusters. Communication between workers can be performed via MPI/TCP/IP protocols. By default, to generate MCMC samples on each subspace the Metropolis-Hastings algorithm is used. However, it can be replaced by other samplers if needed.

It is possible that one or more modes of the target distribution were missed in the exploration step, or that the space partitioning algorithm did not choose the optimal mode separation locations. If the convergence criteria is not met in one or more of the subspaces, then these subspaces are further subdivided. The cut position is defined by using the already generated samples. These samples contain much more information about the target density structure than the initial exploration samples and thus can be used as a more accurate approximation of the target density. This procedure is repeated until samplers in all subspaces report successful convergence tests, or a maximum number of partitioning cycles is reached.

Reweighting

Samples that originate from different subspaces have different, and a priori unknown, normalizations with respect to each other. In order to correctly stitch those together, a weight proportional to the integral of the target density within the subspace needs to be applied. Given that samples are drawn from the target function $\left\{\boldsymbol{\lambda}^k\right\} \sim f(\boldsymbol{\lambda})$ in each subspace k, we compute the following integrals:

$$I_k = \int_{\omega_k} f(\boldsymbol{\lambda}) d\boldsymbol{\lambda} \; . \tag{2.23}$$

We perform this integration using the AHMI algorithm (see Sec. 2.1). However, alternative algorithms for numerical integral evaluation can be considered if needed.

Final sample

Once the sampling and weighting are performed, the weighted samples from multiple subspaces are concatenated and returned to the user. The total integral of the function $f(\boldsymbol{\lambda})$ is then estimated by summing the weights of the subspaces $I = \sum_{k=1..N_{sp}} I_k$.

Our implementation was used to generate the results shown in Figure 2.14.

2.2.3 Performance Benchmark

Example 1

In this example, we evaluate the performance of our algorithm on a more complicated test density function. The function was chosen in such a way to (a) have a known analytic integral, (b) allow generating *i.i.d* samples, and (c) have multiple modes in many dimensions and thus be challenging for a classical Metropolis-Hastings algorithm. With this aim, we have chosen a mixture of four multivariate normal distributions in 9-dimensional space:

$$f(\boldsymbol{\lambda}) = \sum_{i=1}^{4} a_i \mathcal{N}(\boldsymbol{\lambda}|\mu_i, \Sigma_i), \tag{2.24}$$

where all $a_i = 1/4$ and μ and Σ are randomly assigned mean vector and a covariance matrix. Figure 2.16 illustrates one and two dimensional distributions of 10^5 *i.i.d* samples drawn from this density.

There are two primary points that we demonstrate in this section. The first one is the ability to improve the wall-clock time spent on sampling by utilizing efficiently computational resources. The second one is the ability to improve the quality of samples once we increase the number of space partitions. Measurements of the performance were evaluated as follows:

- We use a varying number of subspaces $S = (1, 2, 4, 8, 16, 32)$. Sampling and integration in different subspaces are executed in parallel using 1 worker per subspace with 10 CPU cores per worker. All the CPU cores that are available for the worker are used for multithreaded chain execution.

- In addition, we also vary the wall-clock time that workers can spend on generating samples, considering time intervals of $3, 7, 11$, and 15 seconds.

- For every combination of space partitions and wall-clock times, we repeat the sampling process 3 times to evaluate statistical fluctuations.

The overall number of MCMC runs is 72. An example run is: 8 subspaces and 8 workers with 10 CPU cores each are used to sample 10 chains for 11 seconds of wall-clock time, after which samples are integrated and returned; sampling with this setting is repeated 3 times.

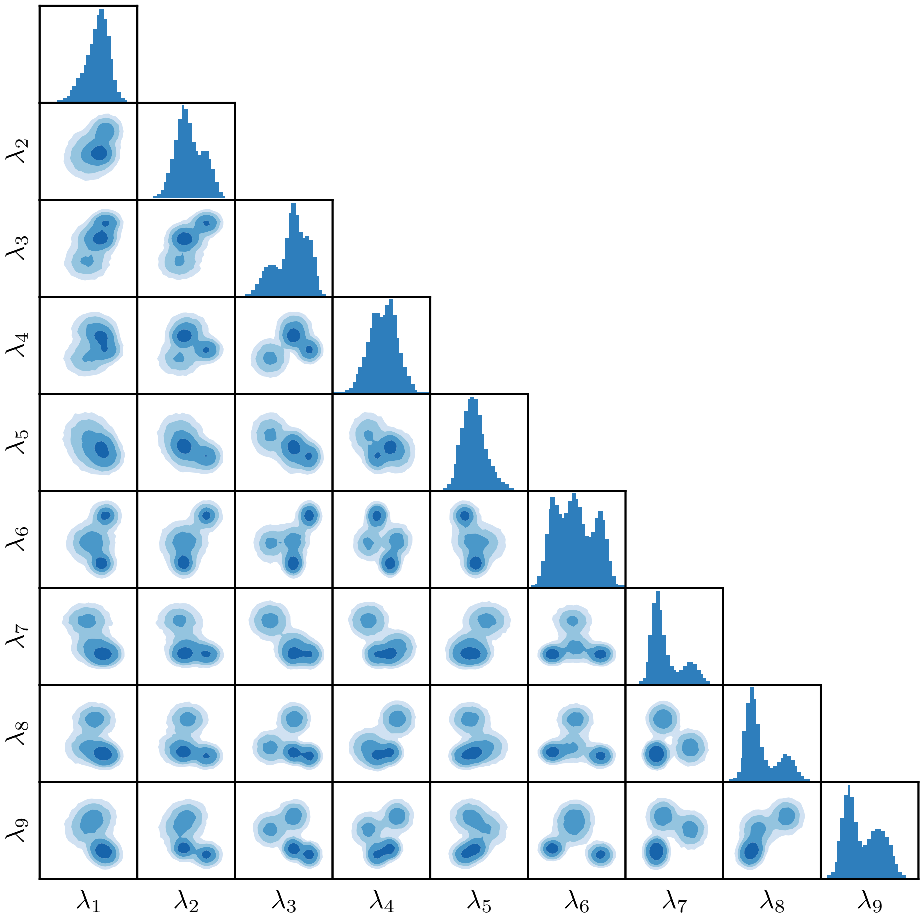


Figure 2.16: One and two dimensional distributions of the density function given by Eq. 2.20. Histograms are constructed using 10^5 *i.i.d* samples.

Sampling Rate A summary of the benchmark run is demonstrated in Fig. 2.17. We used the MPCDF HPC system DRACO[1] with Intel 'Haswell' Xeon E5-2698 processors (880 nodes with 32 cores @ 2.3 GHz each) to perform parallel MCMC executions. While sampling with space partitions, each subspace requires a different amount of time on sampling and integration, depending on the complexity of the underlying density region. The time of the slowest one is reported in Fig. 2.17. It can be seen that, by changing the number of subspaces from 1 to 32 (and the number of total CPU cores from 10 to 320), the number of generated samples increases almost two orders of magnitude while the wall-clock time remains constant.

Figure 2.17 can be rearranged into a slightly different form in order to demonstrate the sampling rate. We define N_0 as the number of samples that the sampler with no space partitions has generated during the time interval $\Delta t_0 = t_{stop} - t_{start}$ (t_{stop} is the wall-clock time when integration has finished and t_{start} is the wall-clock time when sampling on subspace has started). We further denote as N_k the total number of samples from the run with k subspaces, and the time spent on each subspace as Δt_k. The sampling rate is defined as

$$S = \frac{N_k}{\max\limits_k \Delta t_k} \cdot \frac{\Delta t_0}{N_0}. \tag{2.25}$$

[1] https://www.mpcdf.mpg.de/services/computing/draco/about-the-system

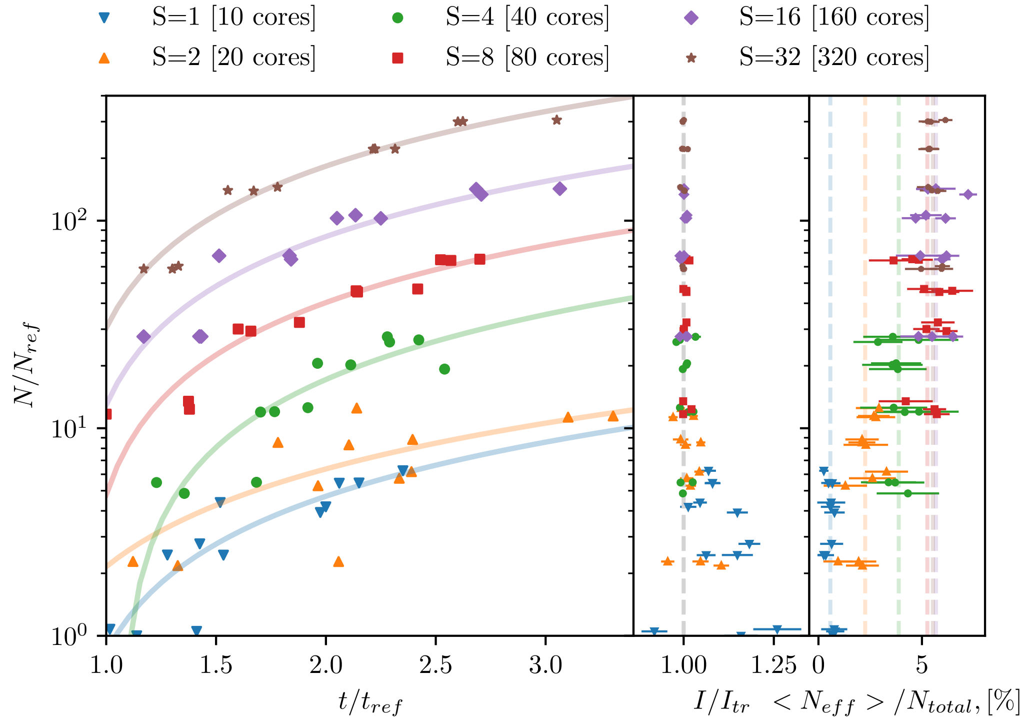


Figure 2.17: Summary of the benchmark runs for the target density function given by Eq. 2.20. The different colors represent runs with different numbers of partitions; the number of subspaces is denoted by S. The vertical axis is common for all subplots and gives the ratio of the total number of samples generated per single run to the number of samples that are generated if no space partition is performed ($N_{ref} = 3.3 \cdot 10^4$). *Left subplot*: The horizontal axis shows the ratio of the time spent on sampling and integration to the time that a single worker spent if no space partition is performed ($t_{ref} = 14.5$ s). The lines are from linear fits of measurements. *Middle subplot*: The horizontal axis shows the ratio of the integral to the true value; error bars are obtained from the integration algorithm. *Right subplot*: The horizontal axis illustrates the ratio of the effective number of samples (separately for each dimension) to the total number of samples. An effective number of samples is estimated per dimension and the error bars represent the standard deviation across dimensions. Dashed colored lines represent the average fraction over the runs with the same number of space partitions.

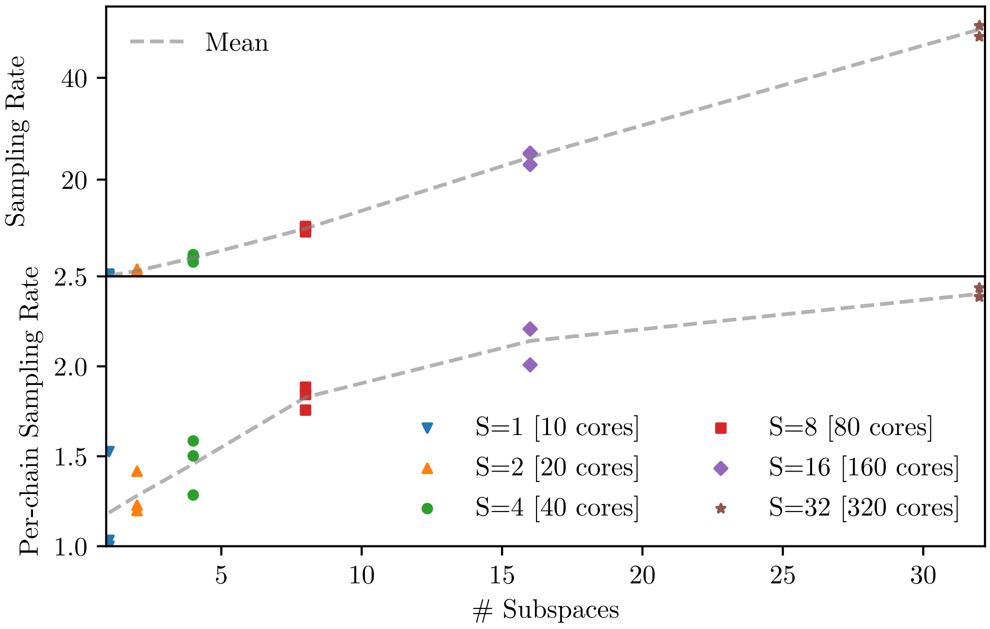


Figure 2.18: The figure illustrates sampling rate (upper subplot) and per-chain sampling rate (lower subplot) versus the number of space partitions. The gray lines represent average over 3 runs.

In addition to the wall-clock time, we measure a CPU time spent on sampling and integration on each subspace using the *CPUTime.jl* package[2]. We denote it as τ_i, where i is the subspace index, and τ_0 is a CPU time when no space partitions are used. The per-chain sampling rate is defined as

$$S_{per-chain} = \frac{N_k}{\sum_{i=1..k} \tau_i} \cdot \frac{\tau_0}{N_0}. \tag{2.26}$$

Eq. 2.25 and Eq. 2.26 do not include time spent on the generation of exploration samples and construction of the partition tree. A time spent on the generation of exploration samples depends primarily on the complexity of a likelihood evaluation, and for our problem, it is equal to 4 seconds. The time required to generate the space partition tree primarily depends on the number of exploration samples, it is about 2 seconds for our problem.

The sampling rate and the per-chain sampling rate versus the number of space partitions are presented in Fig. 2.18. The figure shows that the sampling rates are improved for both cases. Improvements in the per-chain sampling rate indicates that by partitioning the parameter space we simplify the target density function resulting in faster tuning and convergence (tuning and convergence are occurring in every subspace). While improvement in the sampling rate is expected due to the scaling of the number of CPU cores, its faster-than-linear behavior can be explained by a superposition of the improved per-chain sampling rate and the linear sampling rate.

Density Integration and Effective Sample Size Another important characteristic to track is the integral estimate of the target density function. If, for example, samples are not correctly representing the target function, then the integral will deviate from the truth.

[2] A detailed definition of the CPU time can be found in the package documentation.

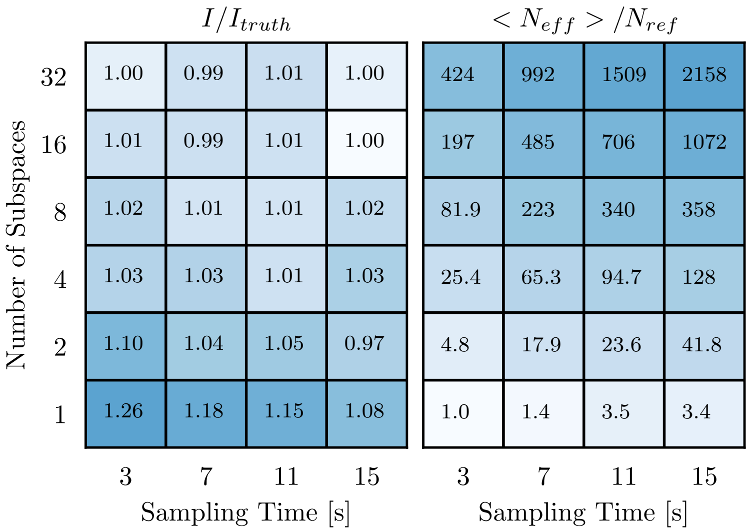


Figure 2.19: The ratio of the integral to the true value (left panel) and the average number of effective MCMC samples (right panel) for the different numbers of subspaces and sampling times. In the right panel, the values are referenced to the average effective number of samples generated for one subspace and a running time of 3 seconds: $N_{ref} = 267$.

By partitioning the parameter space we are simplifying tasks for both the sampler and integrator; the complicated problem has been split into a number of simpler ones. This results in better integral estimates, which can be seen in Fig. 2.17 (middle subplot).

In order to determine the effective number of samples, we evaluate the sum given by Eq. 1.31 using a heuristic cut-off given by Geyer's initial monotone sequence estimator [81]. This technique allows us to calculate an effective sample size for each dimension $N_{eff,k} = \frac{N}{\hat{\tau}_k}$. As it is shown in Fig. 2.17 (right subplot), the effective number of samples increases with the number of space partitions. It can also be seen that in our example there is no increase of the fraction of effective samples when the number of subspaces exceeds 8.

Summary of Scaling Performance A summary of the performance enhancement from partitioning and sampling in parallel is presented in Fig. 2.19. The accuracy of the integrals of the function for different numbers of subspaces as a function of sampling wall-clock time is shown in the left panel. There, we see that even with the longest running times tested, the integral estimate without partitioning deviates considerably from the correct value. Good results are seen already for the shortest running times with 4 subspaces, and running on 32 subspaces gives excellent results in all running times tested.

The right panel in Fig. 2.19 shows the average number of effective samples for different combinations of numbers of subspaces and running times. We find a dramatic increase in the number of effective samples: the factor achieved in the effective number of samples is an order of magnitude larger than the increase in the computing resources (number of processors). This is due to the much simpler forms of the distributions sampled in the subspaces.

Both of these results show that a strong scaling of the performance is achieved using the partitioning scheme that we have outlined.

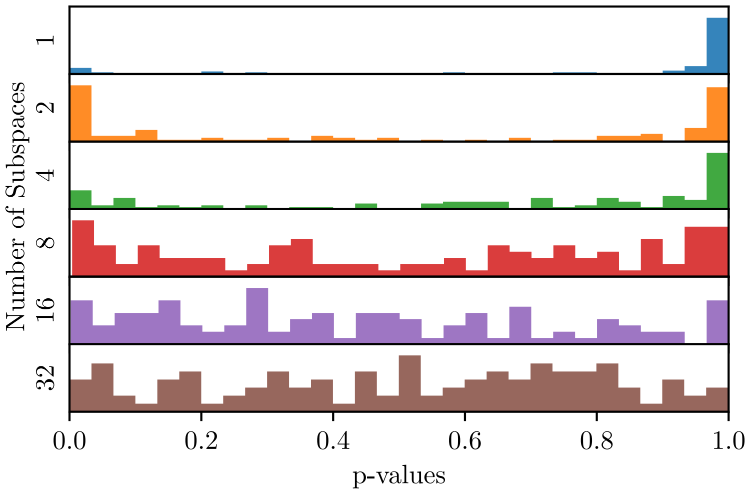


Figure 2.20: The plot illustrates histograms of the p-values from the two-sample Kolmogorov-Smirnov test to verify whether MCMC and *i.i.d* samples come from the same distribution. Different colors represent different numbers of space partitions. One set of MCMC samples results in 9 p-values for every dimension.

Sampling Accuracy Since our algorithm requires the weighting of samples from different subspaces and stitching them together, we test whether this results in a smooth posterior approximating the true target density function. For comparison, we also approximate the true density by directly generating *i.i.d* samples. In the following, we will use the two-sample Kolmogorov-Smirnov test [82] and a two-sample classifier test [83] as a quantitative assessment of how close our MCMC samples are to the *i.i.d* ones.

The two-sample Kolmogorov-Smirnov test is used to test whether two one-dimensional marginalized samples come from the same distribution. P-values from this test for every marginal and different number of space partitions are shown in Fig. 2.20. We use the effective number of samples to calculate p-values for the Kolmogorov-Smirnov test. If two sets of samples stem from the same distribution, then the Kolmogorov-Smirnov p-values should be uniformly distributed. It can be seen that for a small number of space partitions p-values are peaking around 0 and 1. Peaks close to 1 indicate that the effective number of MCMC samples is likely underestimated, demonstrating that samples are very correlated. In contrast to this, the peaks near p-values of zero indicate that marginals of *i.i.d* and MCMC distributions deviate from each other. When using more than 4 partitions, p-values are uniformly covering the range from 0 to 1, indicating that marginals of *i.i.d* and MCMC samples are in a good agreement.

A second approach to determining whether two samples are similar to one another uses a binary classifier aiming at distinguishing them. A training dataset is constructed by pairing MCMC and *i.i.d* samples with opposite labels. If samples are indistinguishable, the classification accuracy on the test dataset should be close to that obtained from a random guess. To train a classifier, we use a simple neural network model with two dense layers with sizes 9×20 and 20×2, and the sigmoid activation function. To construct training and testing datasets, we generate MCMC and *i.i.d* samples with equal weights. By default, *i.i.d* samples come with weight one. For our weighted MCMC however, we generate $3 \cdot 10^4$ samples with unit weights by using ordered resampling implemented in [65]. In total, $4 \cdot 10^4$ samples are used for training and $2 \cdot 10^4$ for testing with an equal fraction of

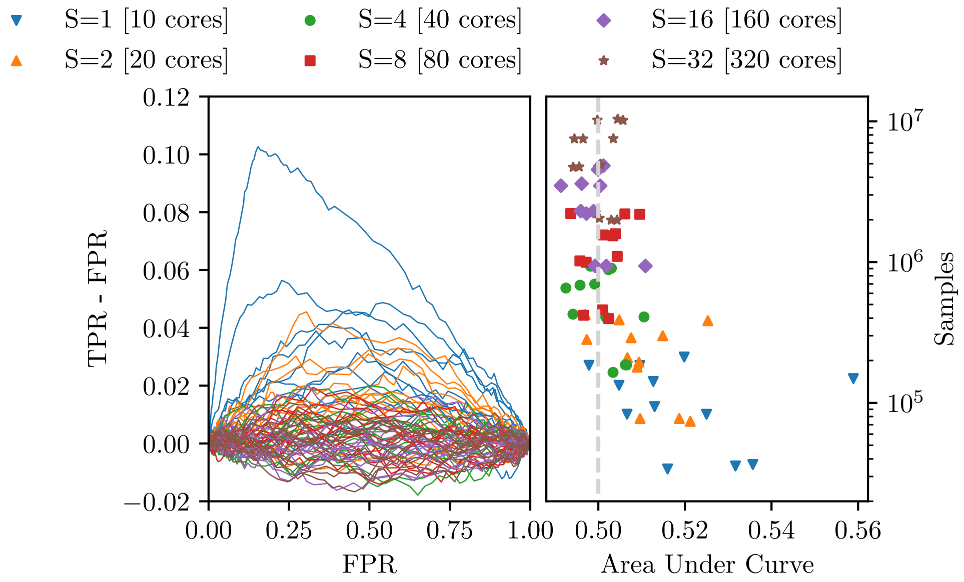


Figure 2.21: The figure illustrates the results of the classifier two-sample test performed to distinguish *i.i.d* samples and MCMC samples with space partitioning. The left subplot shows a modified receiver operating characteristic (ROC) where TPR stands for true positive rate and FPR for false positive rate. Different lines correspond to a different number of subspaces (S). The right subplot shows area under the ROC curve versus the number of samples.

MCMC and *i.i.d* samples. Training is performed for every MCMC run that is described in Fig. 2.17 individually and results are presented in Fig. 2.21. The left subplot in Fig. 2.21 shows the modified receiver operating characteristic (ROC), where the vertical axis is a difference between true positive rate (TPR) and false positive rate (FPR) for different MCMC runs. If the classifier cannot distinguish two samples, then the line will fluctuate around zero. The right subplot in Fig. 2.21 shows the integral under the ROC curve (expected to be close to 0.5 for indistinguishable distributions) versus the number of MCMC samples (before resampling was performed). It can be seen that even though the training datasets consist of the same number of MCMC samples, there is a difference in their distinguishability. Samples obtained from the runs with a large number of space partitions have ROC curve integrals much closer to 0.5. It was not possible to detect this difference in the one-dimensional Kolmogorov-Smirnov tests.

Example 2

In this example, we consider a problem in which the algorithm needs to detect subspaces where the convergence test was not passed and simplify and resample these subspaces until convergence is reached. We construct a target density function as a mixture of eleven bivariate normal distributions, in which covariances scale exponentially. The density function is defined as

$$f(x, y) = \sum_{i=0}^{10} a_i \mathcal{N}(x, y | \mu_{i,x}, \mu_{i,y}, \Sigma_i), \tag{2.27}$$

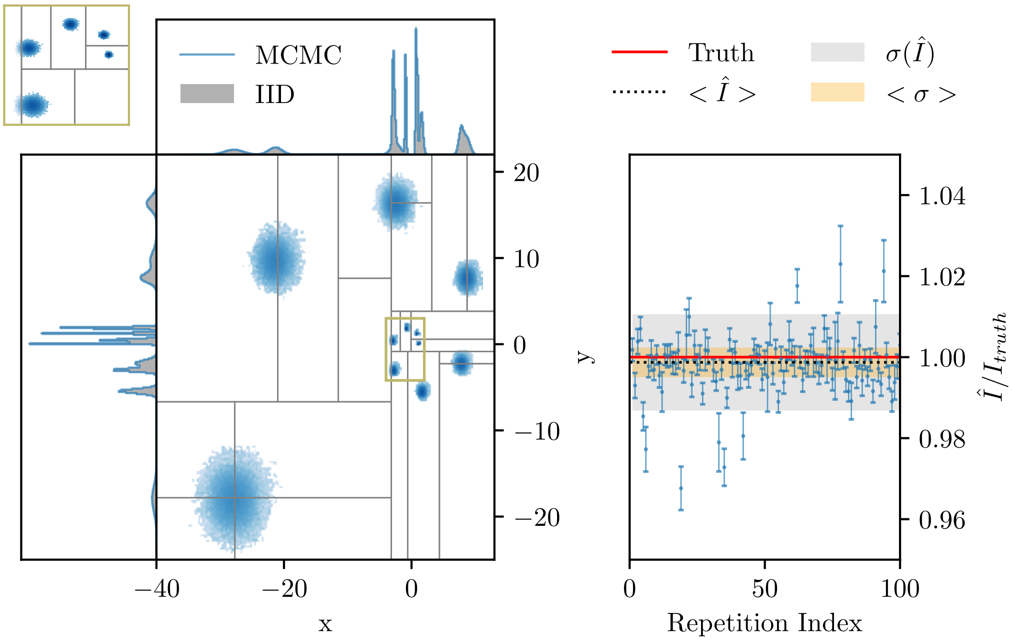


Figure 2.22: The left subplot shows the samples of the target density given by Eq. 2.27 obtained using the partitioned sampling. The gray lines show the boundaries of the subspaces. The horizontal and vertical histograms compare *i.i.d* and space partitioned samples. The area enclosed by a green color is zoomed in the upper left corner. The right subplot shows the density integral normalized by truth for 100 runs. The orange region shows the mean error, and the gray region shows the standard deviation of the results.

where

$$\begin{aligned} \mu_{x,i} &= e^{0.35i}\cos i, \\ \mu_{y,i} &= e^{0.35i}\sin i, \\ \Sigma_i &= \text{diag}(0.45\sqrt{\mu_{x,i}^2+\mu_{y,i}^2}, 0.45\sqrt{\mu_{x,i}^2+\mu_{y,i}^2}), \end{aligned} \tag{2.28}$$

and weights a_i are assigned randomly such that they are non-zero and their sum is equal to one. As in the previous example, this target density allows for *i.i.d* sampling, it is challenging for the Metropolis-Hastins algorithm, and the true value of the density integral is known.

We test our algorithm by first drawing exploration samples with 30 chains and 700 samples per chain. The chains typically get trapped in one of the modes of the target density, and their samples do not represent the entire space correctly. Our algorithm generates 17 initial space partitions using the exploration samples. Due to imperfect exploration sampling, some of these subspaces contain regions with multiple modes. We find that in repeated testing, typically 3 additional subspaces are generated due to a convergence failure. An example of posterior samples from one run and a summary of 100 runs are shown in Figure 2.22. It can be seen that the marginals of the MCMC and *i.i.d* samples overlap, indicating that the target density was sampled accurately. Also, the density integral for 100 runs is close to the true value, with a small average underestimation of 0.1%. The error bars are slightly underestimated, with coverage of 48%.

2.2.4 Space Partitioning Conclusions

We have presented an approach to both improve and accelerate MCMC sampling by partitioning the parameter space of the target density function into multiple subspaces and sampling independently in each subspace. These subspaces can be sampled in parallel and the resulting samples then stitched together with appropriate weighting. The scheme relies on a good space partitioning, which we achieve using a binary partitioning algorithm, that can recursively generate new space partitions in those subspaces where convergence tests of the samplers were unsuccessful; and a good integrator for determining the weights assigned to the samples in the different subspaces. The integrations in our examples were performed using the AHMI algorithm. This approach provides the user with a normalization constant of the target density function - the Bayesian evidence is provided at no extra cost.

We have benchmarked this technique by evaluating the quality of samples and the sampling rate for a mixture of four multivariate normal distributions in 9-dimensional space. We demonstrate that the space partitioning allows us to obtain a 50-fold increase in the sampling rate while increasing the number of CPUs by a factor 32. This increase is a superposition of two effects: a linear scaling with the number of CPU cores, and a CPU-time reduction due to the simplification of the target density function. In addition to the increase in the sampling rate, sampling with space partitioning also resulted in an increased quality of MCMC samples by reducing their correlations. This was evidenced in particular by more accurate integral values of the target density.

We have evaluated the correctness of the resulting sampling distributions by comparing the MCMC samples with *i.i.d* samples using a two-sample Kolmogorov-Smirnov test and with a two-sample classifier test. Both show that increasing the number of space partitions leads to a better agreement between MCMC and *i.i.d* samples.

We have also demonstrated an example of a recursive partitioning scheme using a target density with exponentially scaled covariances. In this problem, the algorithm was repartitioning recursively those subspaces where the MCMC chains failed convergence tests. The estimated evidence averaged over 100 runs is consistent with the true value.

2.3 BAT.jl: A Julia-Based Tool for Bayesian Inference

As disused above, there exist a large number of algorithms for Bayesian computations. To use them in practice, it is also important to have robust, reliable, and optimized implementations of them. This section discusses BAT.jl — a software toolkit for Bayesian inference written in the Julia programming language. The text presented in this section closely follows [65]. The reader is referred to the original publication for a more detailed description.

2.3.1 Motivation and Overview

Nowadays, a variety of automated statistical analysis tools are available, usually tailored to the needs of a particular field of research or a class of statistical models, such as STAN [84], PYMC [85], R [86] or OpenBUGS [87]. An important criterion to choose one tool over the others is its compatibility with the rest of the infrastructure used in a research field, typical data bases or programs used for processing the results obtained.

There is a lack of tools for Bayesian calculation in the field of particle physics. Due to this, the Bayesian Analysis Toolkit [88] C++ library was developed in 2008 under the open-source LGPL license. It features several numerical algorithms for optimization, integration and marginalization with a strong focus on the use of MCMC algorithms. BAT has been widely used over the years and examples of advanced applications in physics [89, 90, 91, 92, 93], cosmology [94], astrophysics [95], and nuclear physics [96].

Given the wide range of possible applications, there was a need to have a more easily portable version of BAT that does not come with the heavy dependencies on particle-physics software stacks and that also allows for smart parallelization. In 2017, the work on complete redesign of the package has started, and it resulted in a new package called BAT.jl implemented in Julia [97].

BAT.jl aims to help solve a wide range of complex and computationally demanding problems. The design of the implementation was guided by the requirement to support multi-threaded and distributed code and offers a choice of sampling, optimization and integration algorithms. BAT.jl has a user-facing interface that makes it easy to quickly solve comparatively simple problems, while offering the direct access to lower-level functionality and tuning parameters that an expert may need to solve very hard problems. In addition, the package is very easy for the user to interact with and visualize results of BAT.jl's algorithms.

2.3.2 Functionalities

Several algorithms for marginalization, integration and optimization are implemented in BAT.jl, giving it a toolbox character that also allows for the future inclusion of further methods, algorithms and software packages. The central algorithms available in BAT.jl are summarized in the following.

Sampling BAT.jl currently provides a choice of a few main sampling algorithms to the user: Metropolis-Hastings, HMC, NS, Grid Sampling, Sampling with Space partitioning.

The package will by default use the Metropolis-Hastings algorithm with four MCMC chains, which are iterated in parallel on multiple threads (and in the future, also on multiple compute nodes). Each MCMC chain is initialized with a random sample drawn from the prior, and we require that efficient sampling is possible for all priors. Typically, priors will be composed from common distributions provided by the Julia package Distributions.jl, which supports *i.i.d.* sampling for all of it's distributions.

In order to determine if the Markov chains have converged and the burn-in phase can stop, BAT.jl uses the Gelman-Rubin convergence test [98] and the Brooks-Gelman test [99].

Algorithms for point estimates The global mode of a posterior distribution is often a quantity of interest. While the MCMC sample with the largest value of the target density may come close to the true mode, it is sometimes not as close as required. It is, however, an ideal starting point for a local optimization algorithm than can then further refine the mode estimation. BAT.jl offers automatic mode-estimation refinement using the Nelder-Mead [100] and LBFGS [101] optimization algorithms, by building on the Optim.jl [102] package. When using LBFGS, a gradient of the posterior distribution is required. We utilize the Julia automatic-differentiation package ecosystem to automatically compute that gradient.

Another quantity that is often computed from samples is a marginal mode. To construct marginals, a binning of the samples is performed. The optimal number of bins can be determined by using Square-root choice, Sturges' formula, Rice Rule, Scott's normal reference rule, or the Freedman-Diaconis rule.

BAT.jl also provides functionality to estimate other quantities such as the median, the mean, quantiles and standard deviations, and to propagate errors on a fit function.

Integration algorithms There are two main algorithms implemented in BAT.jl that allow for evidence estimation. The first one is the AHMI integration with rectangular integration volumes. The second algorithm provides an interface to CUBA [103] integration library. Cuba implements multiple integration algorithms that cover a range of (Monte-Carlo and deterministic) importance sampling, stratified sampling and adaptive subdivision integration strategies. These will typically not scale to high-dimensional spaces, but can provide quick and robust results for low-dimensional problems.

Parameter space transformations Different algorithms have different requirements on the structure and domain of the densities they operate on. HMC, for example (like other gradient-based algorithms), requires a continuous target density. As a result, it does not perform well if the density is bounded. CUBA, on the other hand, can only operate on the unit hypercube, and so requires a bounded density. It is therefore often necessary to perform a change of variables, transforming the original density into one more suitable for the chosen algorithm. The prior distribution will typically contain sufficient information on the structure and domain of the posterior distribution to choose an suitable transformation.

BAT.jl will, by default, automatically try to internally transform posterior densities so that the prior becomes equivalent to a standard multivariate-normal or multivariate-uniform distribution in the transformed space, depending on the requirements of the algorithm chosen to operate on the posterior. Prior distributions are often product distributions and these are simply transformed element-wise. For univariate distributions, BAT.jl will transform according to their (inverse) CDF, multivariate normal distributions are transformed according to their covariance matrix, and hierarchical distributions are transformed iteratively. The mechanism can be extended by specialized transformations for complex or custom prior distributions.

Visualization of results A key element of all statistical analyses is the graphical representation of outcomes. BAT.jl includes functionalities to create visualizations of the analyses results in a user-friendly way. By providing a collection of plot recipes to be used with the Plots.jl package, several plotting styles for one-dimensional and two-dimensional representations of (marginalized) distributions of samples and priors are available through simple commands. Properties of the distributions, such as highest density regions or point estimates like mean and mode values, can be automatically highlighted in the plots. Further recipes to visualize the results of common applications, such as function fitting, are provided. While the plot recipes provide convenient default options, the details of the plotting styles can be quickly modified and customized. Since all information about the posterior samples and the priors are available to the user, completely custom visualizations are of course also possible.

3 Introduction to Particle Acceleration and AWAKE

This chapter introduces the second part of this thesis, which shows a solution to a practical problem from the AWAKE experiment. The chapter starts with a short overview of developments of conventional particle accelerators. This is followed by a discussion of plasma-based particle acceleration, and the chapter concludes with an overview of the AWAKE experiment at CERN.

3.1 Conventional Particle Acceleration

Since the late 19th century, particle accelerators have been the main driving force that advanced our knowledge about the structure of matter from early J. Thomson's theory of atoms [104] to a much more complicated scientific picture, the so-called 'standard model' [105], that describes fundamental particles and forces.

3.1.1 Historical Overview

The history of particle accelerators starts with an electrostatic generator constructed by the British physicists J. Cockcroft and E. Walton, in which charged particles could accelerate in a constant electric field to energies of 400 keV [106]. The energy of particles obtained in such accelerator was limited by a maximum voltage (approximately 700 kV) that could be generated in their system. To avoid the need to increase electrostatic potentials further, Swedish physicist G. Ising proposed to apply the same accelerating voltage periodically through a series of drift tubes connected to a Radio Frequency (RF) generatorr [107]. With this began the era of RF-based particle accelerators, which are still widely used today.

At the same time as the development of linear accelerators, E. Lawrence proposed another approach, called a cyclotron [108]. He suggested that particles can be accelerated along a spiral path where a static magnetic field controls the particles' trajectory and a rapidly varying RF electric field increases the particles' energy. The main limitation of this technology was that relativistic effects (that become significant at high energies) desynchronize the phase between RF and particles motion, thus preventing particles from further acceleration. The cyclotron was superseded in 1944 by a new, improved idea called synchrotron [109, 110]. In a synchrotron, particles are accelerated using RF voltage via a gap or cavity, and they stay in the stationary orbit as their velocity increases. To keep particles on a circular trajectory, Lorentz's force created by dipole magnets is used. In addition, quadrupole magnets are used to control the shape of the beams. The next advancement in particle acceleration was made by transitioning from fixed-target experiments to storage ring colliders, in which particles collide head-on in the center-of-mass system. This was followed by the use of superconducting magnets, which allowed to reach higher magnetic fields to keep particles on circular trajectories.

Overall, the field of particle accelerators has undergone significant progress since the developments of Cockcroft and Walton. Among the largest and most advanced machines that have been built since that time, there are the following: The linear accelerator at SLAC, which could accelerate electrons and positrons up to 50 GeV. A circular particle accelerator called Tevatron at Fermi National Accelerator Laboratory that could accelerate protons and antiprotons to energies of up to 1 TeV. Hadron-Electron Ring Accelerator at DESY in Hamburg in which electrons-positrons collisions were conducted at center-of-mass energies of 318 GeV. A detailed review of historical developments of these and other particle accelerators can be found in [111].

In 1951, there was an intergovernmental meeting of UNESCO in Paris, during which the resolution about the foundation of the European Council for Nuclear Research — also known by the acronym CERN — was established. Since then, CERN has become the largest particle physics laboratory in the world. CERN operates as a chain of accelerators where the largest one is called the Large Hadron Collider (LHC) — a synchrotron accelerator with a size of 27 kilometers in circumference. In the LHC, protons are accelerated in ultrahigh vacuum (pipes pressure $\sim 10^{-11}$mbar) using RF cavities and they can reach energies of centre-of-mass collisions of 14 TeV. To control particles trajectories, superconducting magnets at temperatures of 1.9 K are used. Accelerators at CERN helped to make many important achievements in particle physics, e.g., the discovery of W and Z bosons in the UA1 and UA2 experiments [112], the first creation of antihydrogen atoms in the PS210 experiment [113], discovery of the Quark-Gluon Plasma [114], discovery of direct CP violation in the NA48 experiment [115]. In 2012, the LHC was used to discover the Higgs Boson [116], which was predicted by P. Higgs and independently by F. Englert and R. Brout in 1966. For this discovery, P. Higgs and F. Englert were awarded the Nobel Prize in 2013.

Even though particle accelerators have already significantly advanced our knowledge about the structure of the Universe, there are still many unsolved problems in physics, e.g., quantised theory of gravity [117, 118], the hierarchy problem [119, 120, 121], search for dark matter and dark energy [122, 123], and the origin of neutrino mass [124, 125, 126]. Increasing the energy of collisions even further is one of the most promising directions that can help to test new fundamental theories.

3.1.2 Future Accelerators

Nowadays, the construction of another, more powerful particle accelerator requires making, among others, the following two choices: (a) Linear vs. Circular collider; (b) Lepton vs. Hadron collisions. Hadrons, by their nature, are composite particles, and they produce collisions with a significant background; this makes precise analysis of such collisions very difficult. In contrast, leptons are point-like particles; they generate much cleaner collisions and are more suitable for precision measurements.

The difficulty in using light particles, e.g., electrons, in circular colliders is that the particles would emit intense synchrotron radiation, which would result in a significant loss of energy. For example, the energy loss per turn ΔE of the charged particle when it moves at a circular trajectory with radius r_s is

$$\Delta E_s \sim \frac{E^4}{r_s m_0^4 c^8}, \tag{3.1}$$

where m_0 is the particle rest mass, E is the particle energy, and c is the speed of light. ΔE_s can be minimized by building accelerators with a large radius or by accelerating heavy particles, e.g., protons. In general, the synchrotron radiation severely limits the maximum achievable beam energy of light particles in circular colliders, making the use of linear colliders more suitable for precision measurements [127].

Another limitation of circular collides is the maximum magnetic field that can be created by magnets. The bending radius ρ produced by a dipole with magnetic field $|B|$ perpendicular to the trajectory of a particle with energy E can be estimated as [128]:

$$\frac{1}{\rho}\ [\mathrm{m}] = \frac{|B|\ [T]}{E\ [\mathrm{GeV}]}. \tag{3.2}$$

To keep particles on a constant trajectory, powerful magnets are needed. For example, the LHC uses 1232 dipole magnets with a field strength of 8.3 T each; they all are 15 m long and weigh up to 28 t. The overall space occupied by such magnets is 2/3 of the total length of the accelerator. They are challenging to construct and maintain (a current of ≈ 11 kA passes through a superconducting coil), and significant research is currently ongoing to develop more powerful magnets [129, 130]. Hence, reaching new frontier energies of accelerators also requires the development of new technologies to generate, control and maintain stronger magnetic fields.

Many new projects to extend the frontiers of particle accelerations are under construction or undergo detailed studies, e.g., Compact Linear Collider at CERN [131] (electron-positron center-of-mass energies up to 3 TeV), International Linear Collider [132] in Japan (center-of-mass energies up to 500 GeV). One of the most impressive proposals that currently undergo feasibility study is the Future Circular Collider [133] (FCC) with a circumference of 100 km proposed to be constructed at CERN. If constructed, it can produce hadron collisions with energies up to 100 TeV, or lepton collisions with energies up to 400 GeV.

In contrast to circular accelerators, to improve the maximum attainable energy of particles in linear accelerators, the size of the accelerators and/or accelerating gradients should be increased. To preserve the reasonable size of the accelerator facilities and avoid staging of multiple accelerators, one has to be able to increase accelerating gradients. Novel accelerating mechanisms for high energy physics applications have been proposed, such as laser-driven plasma wakefield accelerators (LWFA) [134], particle-driven plasma wakefield accelerators (PWFA) [135], structure wakefield accelerators (SWFA) [136], and dielectric laser accelerators (DLA) [137]. A detailed review of these approaches can be found, e.g., in [138]. As discussed in the next section, PWFA is one of the most promising ones for the design of future colliders

3.2 Plasma-Based Particle Acceleration

Plasma mediums consist of ionized particles, and they can sustain much higher electric fields compared to other medias. For example, the maximum electric field that can be created at a given plasma density can be estimated by the cold plasma wavebreaking field as

$$E_{max} \approx \frac{m_e c \omega_p}{e} \approx 100 \frac{V}{m} \sqrt{n_e[\mathrm{cm}^{-3}]}, \tag{3.3}$$

where m_e is the electron mass, e is the fundamental charge, n_e is the density of plasma electrons, and ω_p the plasma electron frequency (see Sec. 3.2.1). A plasma density of 10^{14} cm^{-3} can provide accelerating gradients up to $1\,\mathrm{GV\,m^{-1}}$. Hence, high electric fields can produce high accelerating gradients, making plasma-based particle accelerators very promising for many applications.

3.2.1 A Digression Into Plasma Theory

Plasma Definition

Plasma can be viewed as a state of matter, along other states such as solid, fluid, gaseous, or Bose–Einstein condensate, present in our Universe. In the book written by F. F. Chen [139] a plasma is defined as 'A quasineutral gas of charged and neutral particles which exhibits collective behavior'.

The condition of 'collective behavior' means that motion of plasma particles depend not only on local parameters but also on the state of the plasma overall. This is because interactions between charged particles are described by the long-ranged Coulomb force, which implies that the volume elements of plasma can affect one another at large distances.

The quasi-neutrality means that the dimension of the plasma system L should be much larger than the Debye length λ_D, defined[1] as

$$\lambda_D = \sqrt{\frac{\epsilon_0 k_b T_e}{n_e e^2}}, \tag{3.4}$$

where ϵ_0 is the vacuum permittivity, k_B is the Boltzmann constant, and T_e is a the temperatures of the electrons. The condition $\lambda_D \ll L$ means that local fluctuations of plasma density (and generated electric potentials) are vanished by the shielding and therefore plasma remains neutral in the large scale. An ideal plasma approximation implies that the number of particles per Debye sphere is large, i.e., $N_D = n_e \frac{4\pi}{3} \lambda_D^3 \gg 1$. Classical plasma theory assumes that $n_e \lambda_D^3 \gg 1$, which implies that collective effects in plasma are dominant over collisions between particles.

Oscillations in Plasma Fluid Theory

One of the very common approaches to describe plasma phenomena mathematically is to use so-called fluids model. In this model, plasma is assumed to be electromagnetic fluid, or a combination of fluids (if motion of mulitple charges is considered, e.g., electrons and ions). To define a set of closure equations, let us consider the case when the plasma is cold (i.e., the electron temperature is $T_e = 0.0$) and isotropic (without external magnetic fields); and the ions are slightly charged ($Z = 1$) and immobile ($v_i = 0$), with the equilibrium density $n_i = n_0$. Neglecting thermal fluctuations and considering that plasma electron

[1] Assuming that the mobility of ions is negligible.

density is slightly perturbed, $n = n_0 + n_e$, Maxwell's equations that describe electric, $\boldsymbol{E}$, and magnetic, $\boldsymbol{B}$, fields in plasma can be written as

$$\begin{aligned}
&\nabla \cdot \boldsymbol{E} = -\frac{e}{\epsilon_0} n_e, \\
&\nabla \times \boldsymbol{E} = -\frac{\partial \boldsymbol{B}}{\partial t}, \\
&c^2 \nabla \times \boldsymbol{B} = -\frac{e}{\epsilon_0} n_e \mathbf{v} + \frac{\partial \mathbf{E}}{\partial t}, \\
&\nabla \cdot \boldsymbol{B} = 0.
\end{aligned} \tag{3.5}$$

The motion of such plasma is affected by the Lorentz force and constrained by the continuity equation

$$\begin{aligned}
&\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} = -e\boldsymbol{E}/m, \\
&\frac{\partial n_e}{\partial t} + n_0 \nabla \mathbf{v} = 0,
\end{aligned} \tag{3.6}$$

where $\mathbf{v}$ is the perturbed velocity of the plasma. Eq. 3.5 and Eq. 3.6 form a closed set of equations that described linear and non-linear processes in a cold isotropic plasma.

One of the most fundamental processes that occur in plasmas are oscillations caused by perturbations of the electron density. Namely, if a small amount of electrons is displaced with respect to the immobile ions, then the Columb force attracts the electrons back, restoring equilibrium configuration. Differentiating Eq. 3.6 (2) over time, then substituting Eq. 3.6 (1) and Eq. 3.5 (1) gives the differential equation for the plasma density fluctuations:

$$\frac{\partial^2 n_e}{\partial t^2} + \omega_p^2 n_e = 0, \tag{3.7}$$

where $\omega_p^2 = e^2 n_0 / \epsilon m_e$ is the electron plasma frequency — the most fundamental time-scale in plasma. A solution to this equation is the harmonic oscillation of plasma density with the frequency ω_p.

Perturbation of Plasma by a Bunch of Charged Particles

In the context of this thesis, it is of particular interest to consider a response of the plasma on the perturbation caused by the bunch of relativistic particles (further denoted as a 'driver' bunch), e.g., protons, that propagates in plasma at velocity c. For this, let us define the cylindrical charge density of the bunch as

$$\rho_b(r, \zeta) = \rho_z(\zeta) \rho_r(r), \tag{3.8}$$

where $\zeta = z - ct$ is the distance along the bunch in the co-moving frame, $\rho_z(\zeta)$ and $\rho_r(r)$ are normalized longitudinal and radial charge densities. Let us also assume that the proton bunch has a density much smaller than the density of plasma n_0.

When the bunch penetrates plasma, it perturbs its equilibrium state, and the plasma responds to the perturbation by radiating electromagnetic fields also called 'wakefields'. To describe the perturbation, the right hand side of the Eq. 3.5 (1) can be rewritten as $\rho_b(r, \zeta)/\epsilon_0 - e n_e/\epsilon_0$. Keeping only linear therms in Eq. 3.5 and Eq. 3.6, the differential equation describing the response of the electron density on the external perturbation can be defined as

$$\frac{\partial^2 n_e}{\partial t^2} + \omega_p^2 n_e = \frac{\omega_p^2}{e} \rho_b(r, \zeta). \tag{3.9}$$

The last equation describes the motion of the harmonic oscillator affected by the external force, and the solution to it is given by the Green's function response (see [140] for detailed solution). Solving this equation together with Maxwell's equations for $\boldsymbol{E}$, and $\boldsymbol{B}$, allows to find wakefields that are produced by the driver bunch. The longitudinal and transversal components of the wakefield can be computed as

$$\begin{aligned} E_z(r, \zeta) &= Z'(\zeta) R(r), \\ E_r(r, \zeta) &= Z(\zeta) R'(r), \end{aligned} \tag{3.10}$$

where $Z' = \frac{\partial Z}{\partial \zeta}$, $R' = \frac{\partial R}{\partial r}$, and

$$\begin{aligned} Z'(\zeta) &= -\frac{1}{\epsilon_0} \int_\zeta^\infty d\zeta' \rho_z(\zeta') \cos k_p(\zeta - \zeta'), \\ R(r) &= \frac{k_p^2}{2\pi} \int_0^{2\pi} d\theta \int_0^\infty r' dr' \rho_r(r') K_0(k_p \left|\boldsymbol{r} - \boldsymbol{r}'\right|), \end{aligned} \tag{3.11}$$

and K_0 denotes zero-order modified Bessel function of the second kind. Eq. 3.10 and Eq. 3.11 describe the response of the plasma on the perturbation caused by the driver bunch, and they indicate that the strength of the radiated wakefields depend on the proton bunch density $\rho_b(r, \zeta)$.

3.2.2 Plasma Wakefield Acceleration

As discussed above, when the driver beam propagates in plasma, it creates wakefields. When the second beam, also called a witness beam, is passing behind the driver, it can gain the energy from the wakefield — like a wake surfer that rides down a watery hill. This simple idea underlines a powerful concept of plasma wakefield acceleration that is being actively developed in recent decades.

Lasers and particle beams can be considered as drivers of the wakefield. The idea that lasers can create wakefields that can trap and accelerate electrons was proposed by T. Tajima and J.M. Dawson in [141] and later tested by C. Clayton in [142]. In 2006, a regime known as the forced wakefield has been used to produce electron beams with finite energy spread applying intense laser pulses [143]. Later, in 2014, electrons were accelerated to energies of 4.2 GeV with the 300 TW laser system in Berkeley [144]. An alternative idea was proposed in 1985 by P. Chen et al. to use relativistic electron beam as a driver in wakefields [145]. This was successfully tested by J. Rosenzweig in 1988 [146]. Experiments performed at SLAC in 2007 reported acceleration of electrons from 42 GeV to 84 GeV in 85 cm of plasma [147].

In the former experiments, the energy gain was limited by the driver energy and by the propagation distance in the plasma. To reach better performance, the energies of several individual drivers can be combined by staging drivers across many separate stages [148, 149]. Staging of accelerating sections is a technically-challenging procedure. Namely, strong focusing of beams in wakefield results in high-divergence beams in the staging regions. To preserve the quality of the bunches with a finite energy spread, more beam optics is needed in between of the accelerating sections. This optics is space consuming — which results in an overall longer accelerator, making the effective accelerating gradient smaller.

To avoid staging of the multiple sections, an alternative approach is to use a driver bunch with higher energy. Protons have a larger mass, they can carry high energy, and they can drive wakefields over much longer distances than other drivers. Proton bunches with high energies are routinely used in many accelerators. For example, ideal candidates for the plasma wakefield experiment can be proton bunches produced at CERN's particle accelerators, such as PS, SPS, or LHC. These accelerators can produce bunches with energies of $24, 450$ or 7000 GeV, longitudinal length of $\approx 3 - 20$ cm and the radial size of $\approx 100 - 400$ µm. Theory predicts that to reach accelerating gradients of $\mathrm{GV\,m^{-1}}$ and above, the length of the proton bunch should be of the order of plasma wavelength (i.e., of the order of a millimeter) [150]. Therefore long proton bunches should be efficiently compressed into shorter ones.

A bunch-shortening mechanism based on the controlled self-modulation of the long proton bunch is of particular interest in the scope of this thesis. Namely, when the relativistic proton bunch enters the plasma, it seeds longitudinal and transversal wakefields at the plasma frequency. The wakefield acts on the bunch by modulating its radius. The proton distribution with a modulated density produces stronger wakefields. This creates a feedback loop that amplifies the modulation amplitude and eventually splits the long bunch into a group of many micro-bunches with a characteristic length given by the plasma wavelength. This process is known as self-modulation, and it can be used to utilize long proton bunches to drive wakefields in plasma. A relativistic ionization front that co-propagates with the driver bunch can be used to control the seeding of self-modulation, resulting in seeded self-modulation.

3.3 The AWAKE Experiment

The AWAKE experiment is a research and development, proof-of-principle experiment that investigates proton-driven plasma wakefield acceleration. The experiment is a part of the CERN accelerator complex, and it has used long proton bunches with the energy of $\approx 400\,\text{GeV}$ to accelerate a low-energy witness bunch of electrons from $\approx 20\,\text{MeV}$ to two GeV over short distances. To divide the long driver bunches into a group of shorter ones, a seeded self-modulation mechanism is used.

3.3.1 Experiment Overview

The AWAKE experiment is located at the Eastern side of the CERN accelerator complex, as schematically shown in Fig. 3.1. The acceleration of protons used in AWAKE starts with Linac 4, which is the first accelerator in the chain; it accelerates the proton bunch to the energy of 50 MeV. This is followed by the Proton Synchrotron Booster (PSB), which accelerates the protons to 1.4 GeV. The next in the chain is Proton Synchrotron (PS) which increases the bunch energy to 25 GeV. After this, protons are sent to Super Proton Synchrotron (SPS), pushing the bunch energy to 450 GeV. The final accelerating stage is LHC, where a maximum bunch energy of 6.5 TeV has been obtained. For the AWAKE experiment, bunches from SPS are optimal, as they have sufficiently high energy, and they

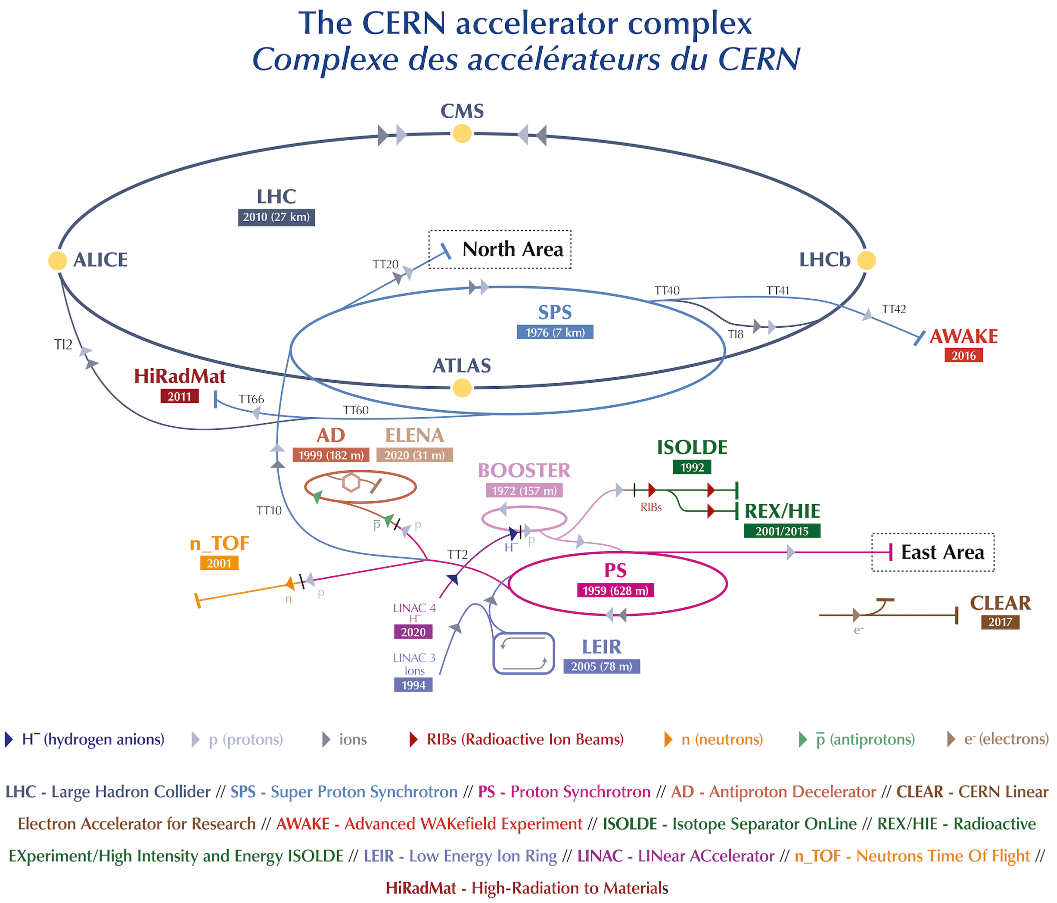

The CERN accelerator complex
Complexe des accélérateurs du CERN
CMS
LHC
2010 (27 km)
ALICE
North Area
TT20
LHCb
TT40
TT41
TT42
AWAKE
2016
SPS
1976 (7 km)
TI8
TI2
ATLAS
HiRadMat
2011
TT66
TT60
AD
1999 (182 m)
ELENA
2020 (31 m)
ISOLDE
1992
BOOSTER
1972 (157 m)
RIBs
REX/HIE
2001/2015
TT10
p̄
p
n_TOF
2001
TT2
H⁻
East Area
PS
1959 (628 m)
n
LINAC 4
H⁻
2020
CLEAR
2017
e⁻
LINAC 3
Ions
1994
LEIR
2005 (78 m)
H⁻ (hydrogen anions)
p (protons)
ions
RIBs (Radioactive Ion Beams)
n (neutrons)
p̄ (antiprotons)
e⁻ (electrons)
LHC - Large Hadron Collider // SPS - Super Proton Synchrotron // PS - Proton Synchrotron // AD - Antiproton Decelerator // CLEAR - CERN Linear Electron Accelerator for Research // AWAKE - Advanced WAKefield Experiment // ISOLDE - Isotope Separator OnLine // REX/HIE - Radioactive EXperiment/High Intensity and Energy ISOLDE // LEIR - Low Energy Ion Ring // LINAC - LINear ACcelerator // n_TOF - Neutrons Time Of Flight // HiRadMat - High-Radiation to Materials

Figure 3.1: Overview of the CERN accelerator complex. Retrieved from [151].

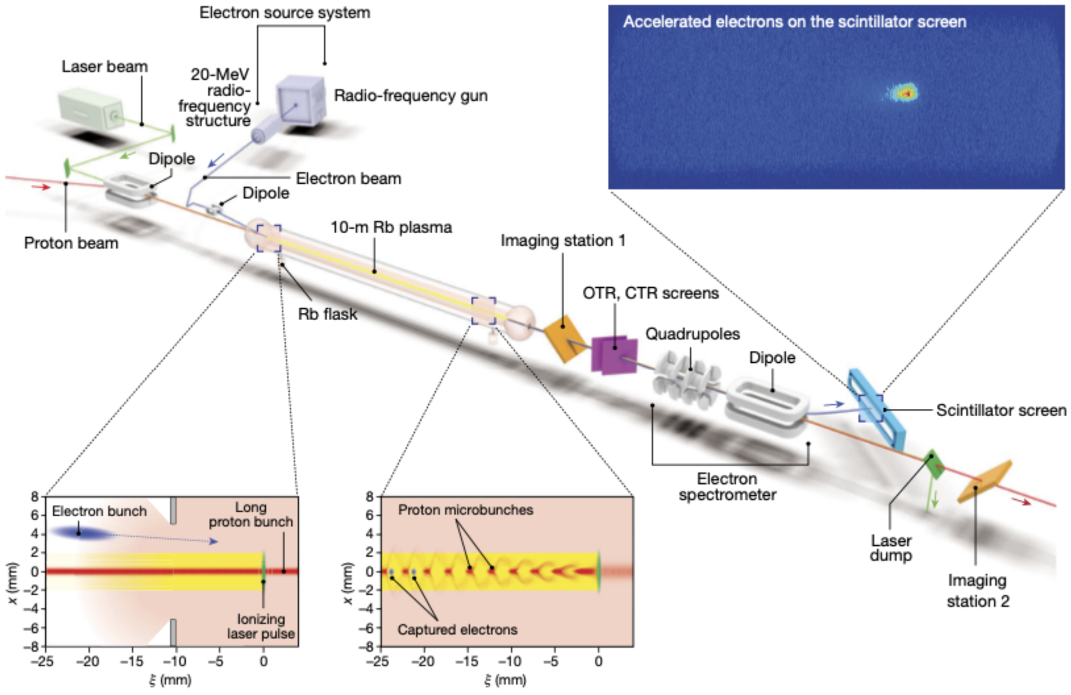


Figure 3.2: The layout of the AWAKE experiment. Electron, proton, and laser beams that propagate from left to right are shown in blue, red, and green colors, respectively. The bottom subplot shows the proton bunch at the plasma entrance and exit. Retrieved from [152].

are more easily extracted than the bunches from the LHC.

After reaching the desired energy, bunches are sent from SPS to AWAKE, which is located in the CERN Neutrinos to Gran Sasso (CNGS) facility — a deep underground tunnel designed for neutrino generation with high-energy proton bunches. The transfer line that connects SPS and AWAKE is $\sim$ 900-meter long, and the plasma section, electron, and laser beams together with various diagnostics are installed downstream of the CNGS tunnel. A layout of the AWAKE experiment is presented in Fig. 3.2.

In AWAKE, the proton bunch enters a ten-meter-long rubidium vapor section [153, 154] together with the co-propagating laser pulse (see Fig. 3.2). According to the Run 1 baseline [155], the bunch is focused at $z_w = (5 \pm 3)$ cm after the entrance to the rubidium section. The radial size is $\sigma_{x,y} = (0.20 \pm 0.04)$ mm, longitudinal size is $\sigma_z = 6 - 8$ cm and angular divergence is $\sigma'_{x,y} = (4 \pm 2) \times 10^{-5}$ rad, where σ represents a Gaussian standard deviation. The parameters of proton bunches, such as the bunch centroid and population, fluctuate from event to event.

The laser pulse [156] with a maximum energy of 450 mJ and a central wavelength of 780 nm ionizes the rubidium vapor, creating a relativistic ionization front that co-propagates with the proton bunch. The relativistic ionization front is much shorter than the typical period of the wakefields ($>$ 3 ps) and it is used to seed the controlled self-modulation instability of the proton bunch. The resulting microbunches then act resonantly to drive large wakefields during the plasma section.

The electron beam with an energy of 18 MeV, maximum charge of 656 pC and with length of the order of the plasma wavelength is injected near the plasma entrance. In this region, significant self-modulation of the proton bunch has not yet occurred. Due to this, some of the electrons are captured by focusing and defocusing phases of the field. Those in the focusing regions are accelerated up to 2 GeV by the wakefield and sent to the electron spectrometer diagnostic, where their energy can be measured [152]. To increase maximum energy attainable by the witness bunch and prevents destruction of the driver microbunches at the final stage of self-modulation, gradients in plasma density are proposed for future use [157].

The Run 1 period of the AWAKE experiment (2016-2018) has shown that self-modulation of a long proton beam in plasma is possible and that this is a fully deterministic and repeatable process [158, 159]. It has also demonstrated that injected electrons can be accelerated by the wakefield from 18 MeV to 2 GeV [160, 152]. After two years of CERN's Long Shutdown 2, the experiment is resuming its operation in the scope of the Run 2 program which is planned from 2021 onward. The goal for this run is to demonstrate the scalable acceleration of an electron bunch while controlling its emittance.

3.3.2 Proton Bunch Diagnostics in AWAKE

The diagnostic tools to control proton bunch quality are present in both the SPS-ring and the AWAKE facility. The SPS diagnostics measure the parameters of the bunch during the acceleration process and before the extraction to AWAKE. The AWAKE diagnostics is used to observe the structure of the proton bunch before and after the plasma section, allowing to analyze self-modulation of the bunch.

In the SPS accelerating ring, wire scanners are used to measure the transverse and longitudinal bunch emittances with the accuracy of 20% [150, 161, 162]. The Beam Quality Monitors [163, 164] are used to measure longitudinal bunch profiles, and Beam Current Transformers (BCT) [165] are used to determine the bunch intensity during the whole accelerating cycle.

In AWAKE, transversal and longitudinal structures of the bunch can be observed before and after the plasma section using a diagnostics based on the Transition Radiation (TR) or scintillating light, which is produced when the bunch crosses light-emitting screens placed at 45° to the beamline. The TR light is prompt as the electromagnetic radiation is produced when the charged particles pass the screen boundary, and it is used as a fast picosecond-scale diagnostic. The scintillating light is slow (up to tens of milliseconds) due to its luminescence nature; it follows an exponentially decaying light profile, and a decay constant is determined by the screen properties, such as material or thickness. The light yield of the scintillating screen is larger compared to the light yield of the TR screen. However, this comes with the price of increased smearing of the signal. A comparative study of various light-emitting screens used in the AWAKE experiment can be found in [166].

The TR light is produced in a broad spectrum. The incoherent part of the spectrum corresponds to the wavelengths of the visible range (400 − 800 nm), and is called the Optical Transition Radiation (OTR) light. It carries information about the bunch longitudinal and transverse structure in a temporal intensity of the light. It can be used to observe the modulation of the individual proton bunches of size $\approx 1.2\,\mathrm{mm}$, by projecting the light onto the slit of a streak camera for time-resolved measurements and onto a screen of a CCD/CMOS camera for time-integrated measurements.

The coherent part of the TR spectrum (CTR) produced by the modulated proton bunch radiates in the microwave frequency range 90 − 300 GHz. It can be mixed with a local oscillator signal to convert the CTR into a signal in the range of 5 − 20 GHz which can be detected by an oscilloscope with a high spectral resolution of 1 − 3 GHz. This allows to perform measurements of the modulation frequency of the proton bunch [167].

In addition to the beam observing systems, Beam Position Monitors (BPM) are used to control positions of the beam, and the beam losses are controlled by the Beam Loss Monitors (BLM).

3.3.3 Impact of the Proton Bunch on the Wakefield Amplitude

The driver bunch plays a central role in the wakefield acceleration. We can estimate a quantitative impact of its parameters on the wakefields using formulas provided in Sec. 3.2.1. For this, let us assume that the relativistic bunch density is defined as

$$\begin{aligned}\rho_b(r,\zeta) &= \rho_z(\zeta)\rho_r(r)\\ &= \frac{Nq}{(2\pi)^{3/2}\sigma_z\sigma_r^2}e^{-\frac{\zeta^2}{2\sigma_z^2}-\frac{r^2}{2\sigma_r^2}},\end{aligned} \tag{3.12}$$

where N is the number of particles in the bunch, and $\sigma_{r/\zeta}$ is the r.m.s. size of the bunch. The strength of the longitudinal wakefield $E_z(r,\zeta)$ can be obtained from Eq. 3.10 and Eq. 3.11. The integration shows that the maximum on-axis longitudinal field behind the bunch center, $\zeta \ll \sigma_z$, is equal to

$$\begin{aligned}E_{z,max} &= eNZ(k_p,\sigma_z)R(k_p\sigma_r)\\ &= eNk_p^2e^{-\frac{k_p^2\sigma_z^2}{2}+\frac{k_p^2\sigma_r^2}{2}}\Gamma(0,\frac{k_p^2\sigma_r^2}{2}),\end{aligned} \tag{3.13}$$

where

$$\begin{aligned}Z(k_p,\sigma_z) &= k_p^2e^{-\frac{k_p^2\sigma_z^2}{2}},\\ R(k_p\sigma_r) &= e^{\frac{k_p^2\sigma_r^2}{2}}\Gamma(0,\frac{k_p^2\sigma_r^2}{2}).\end{aligned} \tag{3.14}$$

In the former equations, $\Gamma(\alpha,\beta) = \int_\beta^\infty t^{\alpha-1}e^{-t}dt$ is the incomplete Gamma function, and $k_p = \omega_p/c$ denotes the plasma wave-number.

As discussed earlier, AWAKE operates in a self-modulation regime in which the initially long proton bunch, $\sigma_{z,0}$, is divided into a group of short microbunches with the size σ_z. To ensure that optimal wakefields are created, a few criteria should be met [168]: (a) The axisymmetric instability mode should develop faster than filamentation of the bunch; (b) the modulation of microbunches should be well-developed; (c) the wakefield amplitude given by Eq. 3.13 should be maximized. The first condition requires that the plasma skin depth is at least as large as the transverse size of the bunch. Examination of Eq. 3.14 and Eq. 3.13 shows that these conditions are satisfied when $\sigma_r k_p = 1$ and $\sigma_z^2 k_p^2 = 2$. In this case, the longitudinal size of the microbunch σ_z is fixed by the plasma wavelength, and the plasma wavelength is fixed by the transverse size of the proton bunch σ_r. To ensure that the self-modulation is well developed in the center of the bunch, bunches with a large aspect ratio should be considered.

The maximum electric field created by a group of microbunches can be estimated by assuming that fields from all microbunches add coherently and that each microbunch contains 1/2 of the initial number of charges within one plasma wavelength [168]. Summing up contributions from microbunches, and assuming that optimal plasma parameters are chosen, i.e., $k_p\sigma_r \approx 1$, $k_p^2\sigma_z^2 \approx 2$, gives

$$\begin{aligned} E_{z,max} &\approx \frac{eN}{4} Z(k_p, \sigma_z) R(k_p\sigma_r) \\ &\approx 0.085 \frac{eN}{\sigma_r^2} \approx 0.12 \text{ GV m}^{-1} \left(\frac{N}{10^{10}}\right) \left(\frac{100}{\sigma_r \ [\mu\text{m}]}\right). \end{aligned} \tag{3.15}$$

As follows from the last equation, the crucial element that determines the maximum attainable strength of the wakefield amplitude in the modulated proton bunch is the ratio of the bunch population to the square of the transverse bunch size N/σ_r^2. In another words, knowing bunch parameters is necessary to understand amplitudes of the experimental wakefields.

3.3.4 Importance of the Proton Bunch for Plasma Modelling

Computer simulations of plasma have been broadly used in various stages of the AWAKE experiment. One of the input parameters that should be provided for plasma modeling is the distribution of protons at the entrance of plasma. A recent study by A. Gorn et al. [169] compares the experimentally-measured distribution of defocused due to self-modulation protons with the results of plasma modeling. The study shows that agreement between experimental observations of the defocused protons and plasma modeling is sensitive to the parameters of the proton bunch. Namely, a 20% uncertainty in the beam radius results in visible disagreement with measured data. To make the comparative analysis more accurate, a smaller uncertainty on the bunch parameters is needed. For this, more precise and systematic event-to-event measurements of bunch parameters should be performed.

4 Analysis of Proton Bunch Parameters in the AWAKE Experiment

As discussed in the last section, proton bunch parameters play a central role in the AWAKE experiment. They impact the development of the proton bunch modulation and define the resulting plasma wakefields. To make detailed comparisons of the simulation and experimental data, an accurate proton bunch description is needed. This chapter contributes to the AWAKE experiment by performing a detailed statistical analysis of the parameters of the proton bunches that drive wakefields.

The work presented here has been submitted for publication [170] as an AWAKE collaboration-wide paper. In the following, by mentioning 'we' in the text, I will refer to those colleagues who supervised me and helped me with different parts of this project.

4.1 Proton Bunch Measurements

4.1.1 Experimental Setup

We use four beam imaging systems that capture images of the transverse profile of the unmodulated proton bunch before and after the rubidium vapor section. Parameters of the beam imaging systems are summarized in Table 4.1. The first three stations have CCD cameras with OTR screens, and the last station has a digital CMOS camera with a scintillating screen. Fig. 4.1 illustrates the relative positions of the measurement stations, the plasma section, and the envelope trajectory of the unmodulated bunch for the baseline parameters. It can be seen that the first two stations are located very close to the waist position, so they mainly carry information about the size of the bunch close to the focus. The last two stations are located much farther from the waist position, and they are primarily sensitive to the angular divergence of the bunch.

	Cam. 1	**Cam. 2**	**Cam. 3**	**Cam. 4**
Full Name	BTV412350	BTV412353	BTV412426	BTV412442
Position, z (m)	0.000	1.478	15.026	23.164
Screen Type	OTR	OTR	OTR	Scint.
Screen Material	Si coated Ag	Si coated Ag	Si coated Ag	Chromox
Screen Drawing	BTVWS0041	BTVWS0041	BTVWS0042	BTVAA0006
Camera Type	CCD	CCD	CCD	CMOS
Camera Res.	400 × 300	400 × 300	400 × 300	1280 × 960

Table 4.1: Description of the beam observing systems (denoted as Cam. 1-4) used in the measurements. The position along the beamline is denoted as z, and it is measured from the position of the first camera.

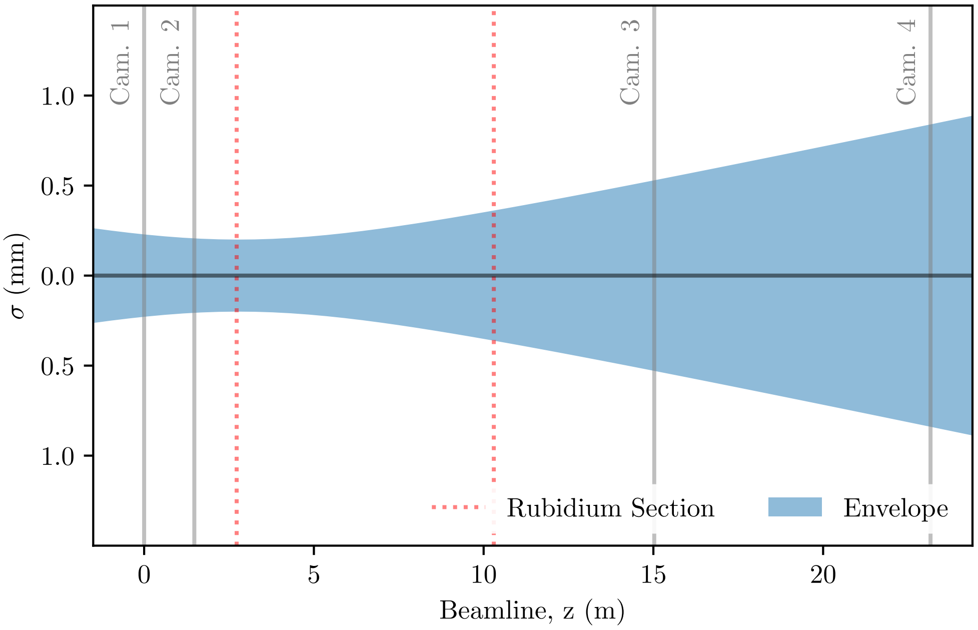


Figure 4.1: The standard deviation of the transverse proton bunch profile versus the beamline position for nominal bunch parameters without plasma. Gray solid lines show positions of four beam observation systems. The position of the plasma section is shown by a red dotted line.

4.1.2 The Dataset

We performed measurements on October 10, 2018, during which a dataset that consists of 672 events was collected. By 'event data' we denote 4 images from the beam imaging systems and the proton bunch population measured in the SPS using the BCT. Four types of proton bunches were requested from the SPS operators with the parameters summarized in Table 4.2. The first two types correspond to the bunches with small, $(7.77-10.30)\times10^{10}p^{+}$, and large, $(23.20-28.00)\times10^{10}p^{+}$, populations. These events are intended to study the impact of the bunch population on the bunch emittance and focal size. The second two types represent bunches with or without longitudinal compression. The bunch compression is achieved via a rotation in longitudinal phase space using a voltage step with a fast rise time [171, 172]. The typical longitudinal bunch length (rms) without the bunch rotation is ≈ 9.6 cm and with the bunch rotation ≈ 7.9 cm. The acquired dataset represents proton bunches commonly used in the later stages of Run 1 data-taking period.

Symbol	q $[1e^{10}, p^{+}]$	**Rotation**	**# Enevts**
D_{11}	7.77 - 10.30	ON	181
D_{12}	7.77 - 10.30	OFF	160
D_{21}	23.20 - 28.00	ON	139
D_{22}	23.20 - 28.00	OFF	192

Table 4.2: Events are divided into 4 categories, i.e., small and large bunch population and bunch rotation ON and OFF.

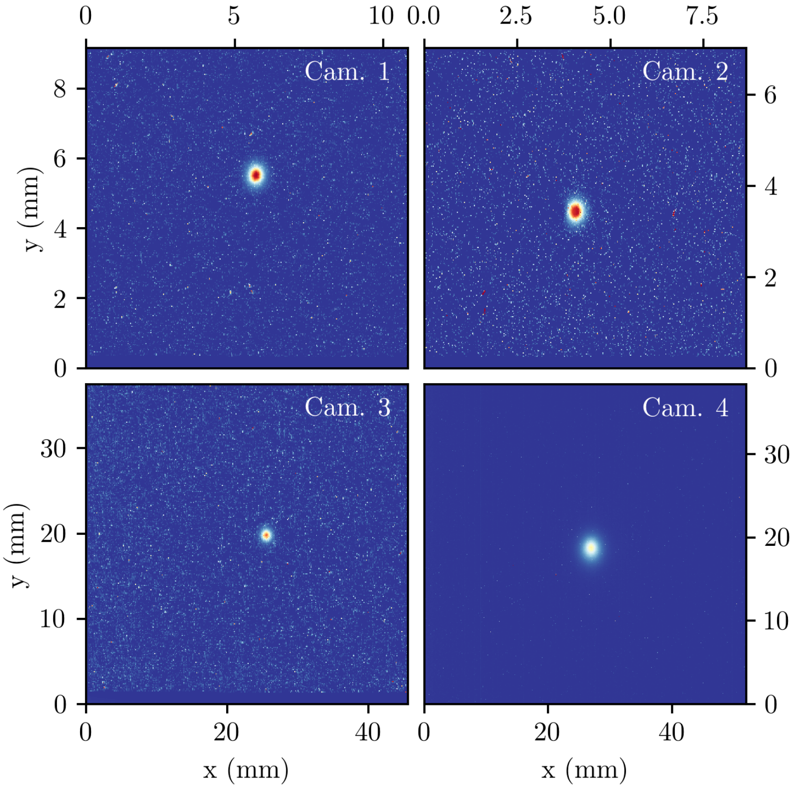


Figure 4.2: Images from one event with the proton bunch population $q = 24.04 \cdot 10^{10} p^{+}$. Subplots correspond to the four beam observing systems specified in Table 4.1. Color-coding represents the values recorded by the camera pixel, and it is in the range $[0, 4095]$.

An example of the event data with a population of $q = 24.04{\times}10^{10} p^{+}$ is shown in Fig. 4.2. The central part of the images represents the OTR light on the first three screens and the scintillating light on the last screen produced by the proton bunch. The background noise is produced primarily by secondary particles that are generated upstream of AWAKE.

The integrated intensity of the signal in four cameras is summarized in Fig. 4.3. It can be seen that the intensity is correlated with the bunch population. Namely, it increases with the increasing bunch population in Cam. 1, 2, 4, and decreases in Cam. 3. The decrease in Cam. 3 is explained by the change in the camera setting, that was made to avoid signal saturation.

4.2 Preliminary Investigations

The analysis relies on knowledge of calibration factors and resolutions scales for the devices used. These are described briefly.

4.2.1 Pixel Calibration Factor

To determine the calibration factor of the pixel sizes, we use calibration frames that are engraved on the surface of each light-emitting screen. Parameters of the frames, such as width, height, and sizes of the engraving lines, are known to high precision. The screens are placed at an angle of 45° with respect to the beamline and rotated on the horizontal

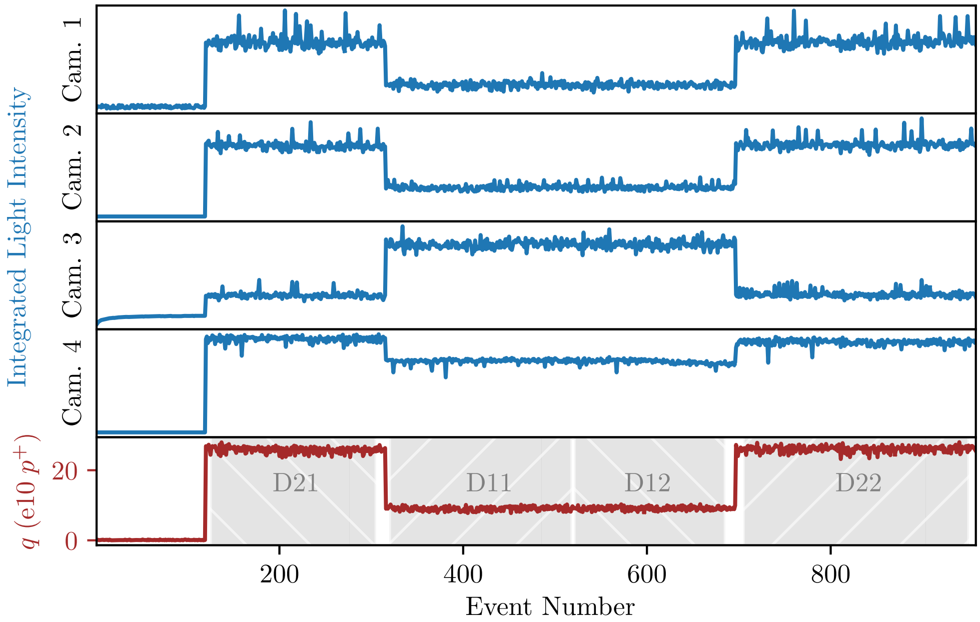


Figure 4.3: The plot illustrates the integrated light intensity and the proton bunch population versus the event number for four cameras. The shaded regions specify datasets that were measured with the same experimental settings.

axis. The images of the calibration frames that are taken from the viewports of each camera are shown in Fig. 4.4. To calculate the horizontal calibration factor, we divide the absolute size of the frame by the number of pixels that correspond to it. To determine the vertical calibration factor, the absolute size of the frame is multiplied by $\cos(\pi/4)$ and then divided by the number of corresponding pixels.

For illustration, let us compute the calibration factor for Cam. 1. It has a calibration frame with a technical drawing BTVWS0041, width 15 mm and height 16.97 mm. Since the frame is visible at 45°, its width is 15 mm and height is $16.97 \cdot \cos(45) = 8.91$ mm. The horizontal value is estimated as

$$\Delta x = \frac{0.5 \cdot \text{frame width}}{\text{number of pixels}} = \frac{7.5\text{ mm}}{277\text{ px}} = 0.0271\text{ mm}. \tag{4.1}$$

The thickness of the engraving line is 0.4 mm, and it is larger than the pixel size. This creates uncertainty on finding the optimal position of the measuring grid shown by the dashed lines in Fig. 4.4. This uncertainty can be estimated as a standard deviation of the uniform distribution given by

$$\sigma(\Delta x) = \frac{1}{\sqrt{12}} \cdot \frac{\text{line thickness}}{0.5 \cdot \text{frame width}} \cdot \Delta x = 42 \cdot 10^{-5}\text{ mm}. \tag{4.2}$$

This gives the calibration factor of $\Delta x = (27.1 \pm 0.42) \cdot 10^{-3}$ mm. The same procedure is performed for other cameras, and the results are summarized in Table 4.3. In the following, we assume that the pixel size includes the correct calibration factors.

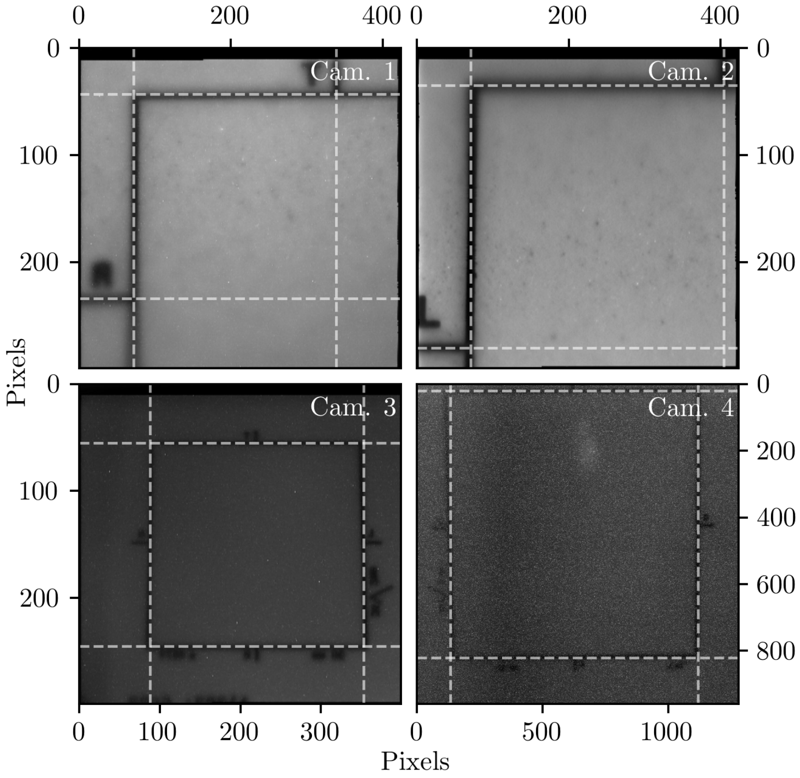


Figure 4.4: Calibration images used to determine pixel size. The dashed lines show an approximation to the frame edges. The width of the engraving lines is 0.4 mm. References to the detailed technical drawings are provided in Table 4.1.

Name	Δx (µm)	Δy (µm)
Cam. 1	27.1 ± 0.42	30.5 ± 0.59
Cam. 2	21.6 ± 0.33	23.4 ± 0.45
Cam. 3	114.0 ± 0.88	125.0 ± 1.2
Cam. 4	40.6 ± 0.23	40.0 ± 0.29

Table 4.3: Summary of the pixel size calibration.

4.2.2 Resolution Functions

Optical resolution is a characteristic of the system to resolve details of the signal that is being measured. The features of the signal that are smaller than the resolution size will be smeared and not detected by the measurement. For the one-dimensional signal, the resolution effect can be presented as a convolution

$$f(x) = \int_{-\infty}^{\infty} g(\tau)k(x-\tau)d\tau, \tag{4.3}$$

where $g(x)$ is the incoming signal, $f(x)$ is the response of the measurement device, and $k(x)$ is the resolution kernel. The resolution kernel can be presented using a Gaussian function centered at 0 with a standard deviation of $\tilde{\sigma}$

$$k(x) = \mathcal{N}(x|0, \tilde{\sigma}). \tag{4.4}$$

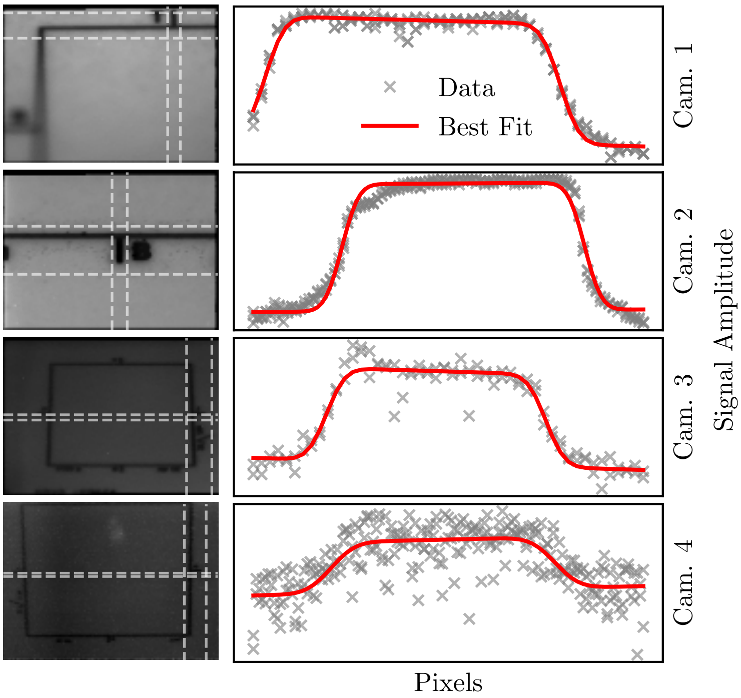


Figure 4.5: Left column: Calibration images used to estimate resolution functions. The regions enclosed by dashed lines show pixels that were used in the fit. Right column: The data points and the best-fit models for different rows/columns of the camera. The resulting resolution parameters are summarized in Table 4.4.

Images of the bunch profile obtained from the beam observing systems contain two types of resolution effects. The first one is the resolution of the optical system of the camera that is denoted as $\tilde{\sigma}_c$. The second one is the resolution of the light-emitting screen denoted as $\tilde{\sigma}_s$. We assume that they both are Gaussian smearings, and their superposing effect is given by a quadrature sum of individual components, i.e., $\tilde{\sigma}_{tot} = \sqrt{\tilde{\sigma}_s^2 + \tilde{\sigma}_c^2}$.

The camera's resolution, $\tilde{\sigma}_c$, depends on the parameters of the optical system, camera magnification, and signal-to-noise ratio. This resolution can be determined with high precision by sending into a camera a signal profile with a known shape and measuring the response of the camera, as it is done, e.g., in the USAF resolution test. In our case, this precise calibration was not performed, and therefore an upper limit of the resolution function is estimated using images of the calibration frame. Namely, similarly to the pixel calibration test, it is assumed that the calibration frame has perfectly shaped rectangular edges with known width of the lines. The rectangular edge can be presented as a product of Heaviside step functions $g(x) = H(x)H(w - x)$, with w denoting the width of the engraving line. Given this incoming signal, the response of the camera is

$$f(x|\tilde{\sigma}, w, \Delta x) = \int_{-\infty}^{\infty} H(\tau)H(w - \tau)\mathcal{N}(x - \tau|0, \tilde{\sigma} \cdot \Delta x)d\tau. \tag{4.5}$$

We perform a fitting procedure to determine the best resolution parameters that approximate the data (see Fig. 4.5). The resulting resolution parameters are summarized in Table 4.4.

The resolution of the light-emitting screen is a process that is determined by the mechanism of the OTR or scintillating light creation, and it depends on the screen material and

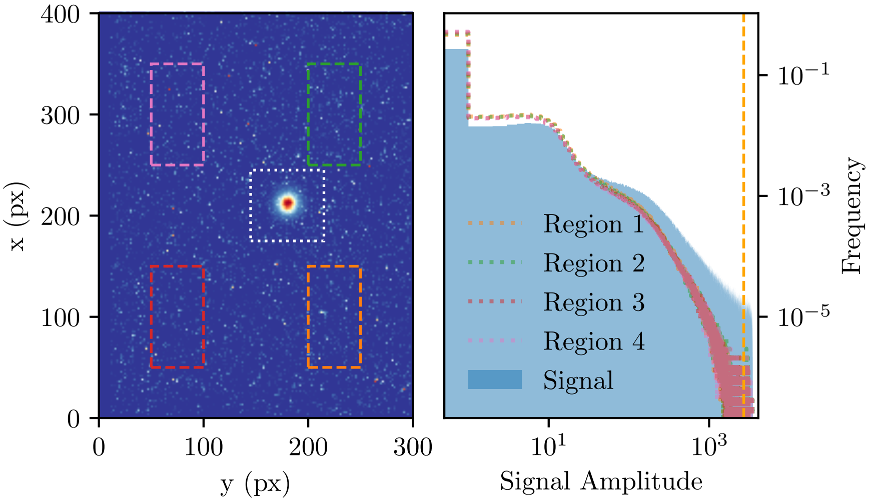


Figure 4.6: Background distributions from different regions of Cam. 1. The frequency represents the number of counts present in each bin normalized such that the sum of frequencies from each bin is equal to one. The pixels enclosed by the white dotted lines are used to determine the parameters of the proton bunch. The background distributions from four rectangular regions enclosed by the dashed lines are shown in the right subplot. The dashed line in the right subplot shows the truncation threshold. The filled region represent the distribution of the signal (enclosed by the white dotted line).

thickness. Precise calibration of the resolution of the light-emitting screens has not been performed in the AWAKE experiment. A study performed at the CERN HiRadMat test facility indicates that scintillating screens are characterized by a worse resolution compared to the OTR screens [166]. We include the additional smearing from the scintillating screen in our analysis using nuisance variables and consider constant resolution parameters for 3 cameras with OTR screens.

4.2.3 Background Distribution

We divide the image from each camera into different regions (see Fig. 4.6). Those pixels located in the central region — which is approximately 4-5 standard deviations around the bunch centroid — are used to infer the parameters of the proton bunch. The remaining pixels are combined from multiple events to approximate the background distribution via binned histograms. We use separate background distributions for each camera and datasets with small and large bunch populations. An example of the background distributions from

Name	$\tilde{\sigma}_{x/y}$ (px)	$\tilde{\sigma}_{x/y}$ (µm)
Cam. 1	1.91 ± 0.1	58.3 ± 3.05
Cam. 2	2.11 ± 0.16	49.37 ± 3.74
Cam. 3	1.76 ± 0.25	200.6 ± 28.5
Cam. 4	6.97 ± 1.54	283.0 ± 62.5

Table 4.4: The upper limits of the optical resolution function given in pixels and microns.

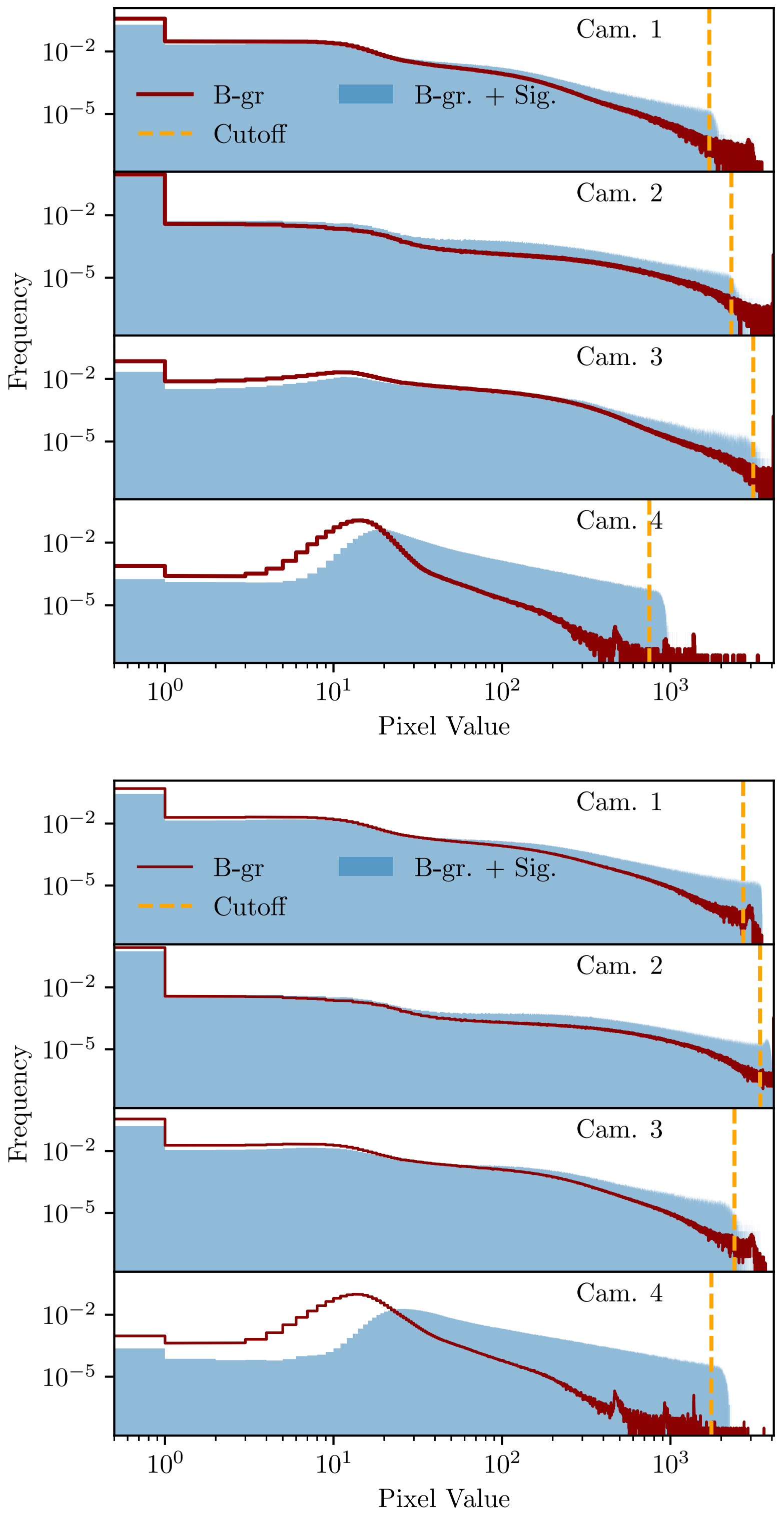


Figure 4.7: Background distributions corresponding to the dataset with a small (top) and large (bottom) bunch populations. The dark red colors show the distribution of the background only. The filled histograms show the distributions from pixels used in the analysis of bunch parameters, and they represent the superposition of signal and background. Orange lines show positions of the cutoff values.

four different regions of one camera is shown in Fig. 4.6. It can be seen that all histograms significantly overlap, demonstrating that the background follows a similar distribution in different parts of the camera.

Background distributions from the four different cameras are shown in Fig. 4.7 for events with a small and large bunch populations.. The histograms corresponding to cameras with the OTR screens have a similar structure. Namely, they all have a maximum at 0, long tails that extend to the amplitudes of > 2000, and a small bump of saturated pixels at the righthand side of the histograms. To avoid a possible negative impact of the saturation on the analysis, we discard those pixels that exceed the threshold values of 2700, 3400, 2400,1750, for each camera, respectively (see also dashed lines in Fig. 4.7). The histograms of signal and background show a smooth behavior up to these threshold values without signs of saturation.

The images produced by the last beam observing system have approximately ten times more pixels compared to the images from the three other cameras. To improve the runtime of the analysis, we average every 3×3 pixels from the last camera. Averaging of the pixels reduces the long tail of the distribution as can be seen in Fig. 4.7 (bottom subplot). In the following, we will assume that the background on each pixel of the last camera is well-modeled with a truncated (form 0 to 4095) Gaussian distribution. The distribution variance is determined from the histogram, and the mean is fitted with a free parameter.

4.3 Statistical Model

We perform the statistical analysis using a Bayesian approach. Prior probabilities about parameters $\boldsymbol{\lambda}, \boldsymbol{\nu}$ of the model M are updated to posterior probabilities using Bayes' theorem

$$P(\boldsymbol{\lambda}, \boldsymbol{\nu}|M, D) = \frac{P(D|\boldsymbol{\lambda}, \boldsymbol{\nu}) \cdot P(\boldsymbol{\lambda}, \boldsymbol{\nu}|M)}{\int P(D|\boldsymbol{\lambda}, \boldsymbol{\nu}) \cdot P(\boldsymbol{\lambda}, \boldsymbol{\nu}|M) d\boldsymbol{\lambda} d\boldsymbol{\nu}}, \tag{4.6}$$

where $P(D|\boldsymbol{\lambda}, \boldsymbol{\nu})$ is a likelihood that represents a probability of data given the model, $P(\boldsymbol{\lambda}, \boldsymbol{\nu}|M)$ is a prior, and $P(\boldsymbol{\lambda}, \boldsymbol{\nu}|M, D)$ is a posterior probability distribution. The data from one event is denoted as $D \equiv \left\{d^j_{x,y}\right\}$, where d is a signal from one pixel, x, y are the row, column of the pixel and j represents the camera index. The dataset consists of multiple events $\{D\}_i$ where i denotes the event index. In Eq. 4.6, $\boldsymbol{\lambda}$ represents parameters of interest, and $\boldsymbol{\nu}$ represents nuisance parameters. Some of the nuisance parameters are kept constant, and others are free parameters of the fit that will be marginalized at the final stage of the analysis.

4.3.1 Likelihood Definition

The measured datasets consist of multiple events, and their parameters (e.g., transverse size of the bunch, angular divergence, waist position, bunch alignment) fluctuate from event to event. The parameters of the experimental setups (e.g., pixel size, camera resolution) are also characterized by uncertainties, and their true (unknown) values were constant at the whole measurement process. Given this, there are two ways to define the likelihood function.

In the first approach, all events can be analyzed simultaneously. This allows the experiment-specific parameters to be free variables of the fit. In this case, experiment-specific parameters will be tuned globally, such that they represent the optimal values for every event of the dataset. The likelihood that corresponds to this case can be defined as

$$P(\{D\}|\boldsymbol{\lambda}, \boldsymbol{\nu}) = \prod_{D \in \{D\}} \prod_{j \in N_{cam}} \prod_{y \in N_{rows}} \prod_{x \in N_{columns}} p(d^j_{xy}|\boldsymbol{\lambda}, \boldsymbol{\nu}), \tag{4.7}$$

where $p(d^j_{xy}|\boldsymbol{\lambda}, \boldsymbol{\nu})$ denotes the likelihood of one pixel. The likelihood given by Eq. 4.7 requires approximately $16.8 \cdot 10^3$ free parameters of the model, and each likelihood execution require $6 \cdot 10^6$ evaluations of $p(d^j_{xy}|\boldsymbol{\lambda}, \boldsymbol{\nu})$. This large number of free parameters and expensive likelihood-evaluation would make the sampling procedure very challenging. Therefore, approximations should be considered to simplify the computations.

Another, simpler, approach is to analyze individual events consecutively. To ensure that all events are analyzed at the same experiment-specific parameters, the last should be kept constant. The likelihood that corresponds to this case can be written as

$$P(D|\boldsymbol{\lambda}, \boldsymbol{\nu}) = \prod_{j \in N_{cam}} \prod_{y \in N_{rows}} \prod_{x \in N_{columns}} p(d^j_{xy}|\boldsymbol{\lambda}, \boldsymbol{\nu}), \tag{4.8}$$

and it replaces one many-dimensional problem with many problems with fewer dimensions. Eq. 4.8 gives a compromise between numerical complexity of the likelihood evaluation and accuracy of the models, and it will be further considered in our analysis.

The d^j_{xy} are composed of a background noise, with a probability distribution denoted as $P_b(d^j_{xy})$, and a signal $P_p(d^j_{xy}|\boldsymbol{\lambda}, \boldsymbol{\nu})$ produced by the proton bunch. In the following, we will avoid indices x, y, j in d^j_{xy} for notational convenience if one pixel is considered. We assume that, for the given camera, the background is the same for all pixels and we approximate

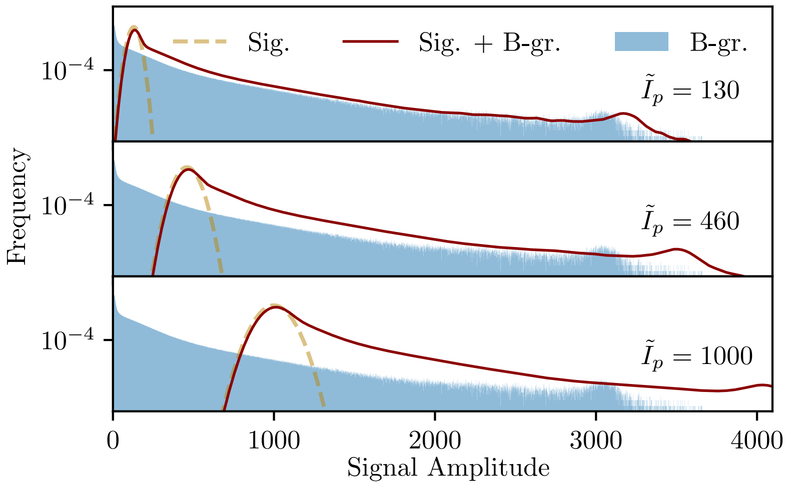


Figure 4.8: Convolutions of the probability distribution of the signal created by the proton bunch and the background distribution are shown for different assumed mean signal amplitudes and variances for Cam. 3. The signal from the proton bunch, $P_p(d|\boldsymbol{\lambda}, \boldsymbol{\nu})$, is assumed to be normally distributed (shown in dashed lines). The filled histograms show the background noise. The red lines show a numerical convolution represented by $p(d|\boldsymbol{\lambda}, \boldsymbol{\nu})$.

it by a binned histogram (see previous section). The resulting superposition of these two contributions is given by the convolution

$$p(d|\boldsymbol{\lambda}, \boldsymbol{\nu}) = \int P_p(\tau|\boldsymbol{\lambda}, \boldsymbol{\nu}) \cdot P_b(d - \tau) d\tau. \tag{4.9}$$

The convolution is computed numerically for the first three cameras due to the non-analytic form of the background, and it is computed analytically for the last camera, where the background is Gaussian. An example for such a convolution is shown in Fig. 4.8, where it is assumed that the signal from the proton bunch has a Gaussian distribution. The evaluation of the numerical convolutions is computationally expensive, but it is needed to include correctly the non-analytic features of the background.

The key component of the equation Eq. 4.9 is a probability distribution of the camera's signal intensity created by the proton bunch $P_p(d|\boldsymbol{\lambda}, \boldsymbol{\nu})$. To compute it, one needs to simulate the trajectory of the bunch along the beamline and then convert a distribution of protons at each screen position into a signal on a camera.

4.3.2 Bunch Propagation Equation

We consider a model in which each particle of the bunch follows a linear equation of motion defined as

$$\mathbf{r}_i = \mathbf{r_0}_i + \boldsymbol{r}'_i (z - z_w) \, , \tag{4.10}$$

where i denotes the particle's index, z_w denotes the waist position, i.e., the coordinate along the beamline in which the radial bunch size is minimal, $\mathbf{r}_0 = (x_0, y_0)$ is the particle position at the waist, and $\boldsymbol{r}' = (dx/dz, dy/dz) \approx (\theta_{xz}, \theta_{yz})$ is the particle angle with respect to the beamline in the $x - z$ and $y - z$ planes. The distance along the beamline is denoted by z and is defined as the trajectory of the bunch centroid.

We measure distances in the transverse directions relative to the center of the proton

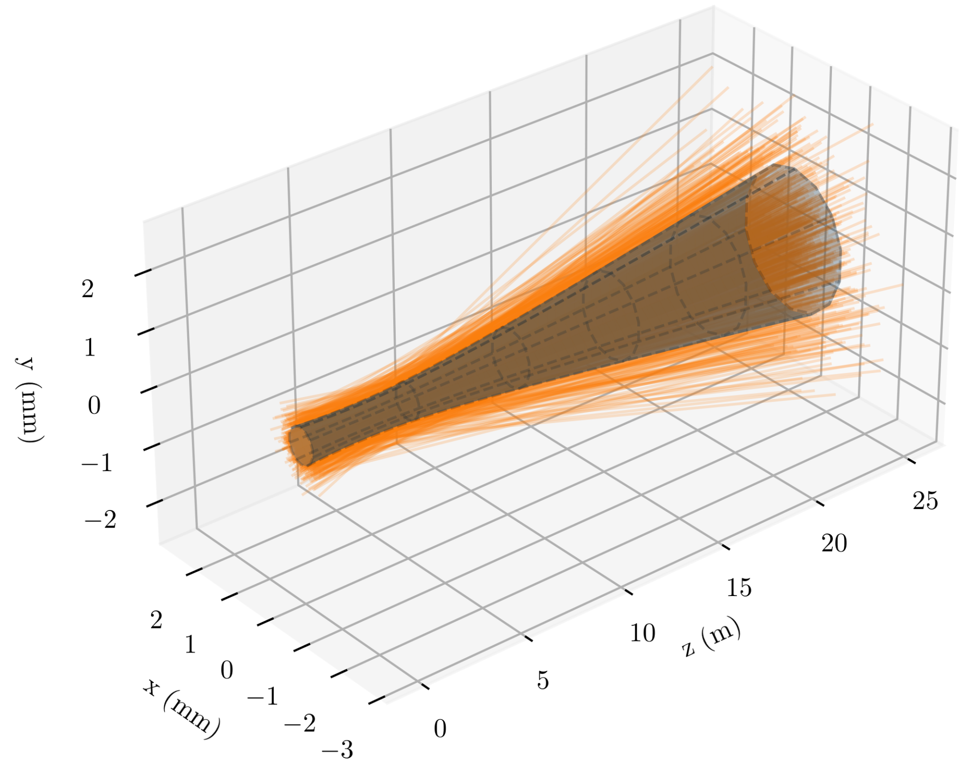


Figure 4.9: The figure illustrates trajectories of 10^4 particles and their envelope equation for the nominal bunch parameters. The envelope equation is computed using Eq. 4.12 and it is shown by the blue surface.

bunch. To determine the center of the bunch, we use two parameters per camera, one for the x and one for the y directions, labeled as $\mu_{j,x}, \mu_{j,y}$ with j denoting the camera index.

The envelope equation that describes the transverse size of such a bunch along the beamline can be defined as

$$\begin{aligned}\boldsymbol{\sigma}^2 &= \left\langle \boldsymbol{r}^2 \right\rangle \\ &= \left\langle \boldsymbol{r}_0^2 \right\rangle + 2 \left\langle \boldsymbol{r_0} \boldsymbol{r}' \right\rangle (z - z_w) + \left\langle \boldsymbol{r}'^2 \right\rangle (z - z_w)^2,\end{aligned} \tag{4.11}$$

where $\langle \cdot \rangle$ denotes the average over the ensemble of particles. It is assumed that there is no correlation between the x and y projections of the bunch, and that $\boldsymbol{r_0}$ and $\boldsymbol{r'}$ are not correlated at the waist position, i.e., $\langle \boldsymbol{r_0} \boldsymbol{r}' \rangle = 0$. For notational convenience, the bunch size at the waist will be denoted as $\boldsymbol{\sigma}_0 = \sqrt{\langle \boldsymbol{r}_0^2 \rangle}$, and the angular spread of the bunch as $\boldsymbol{\sigma'} = \sqrt{\langle \boldsymbol{r}'^2 \rangle}$. Eq. 4.11 can be rewritten as

$$\boldsymbol{\sigma}^2 = \boldsymbol{\sigma}_0^2 + \boldsymbol{\sigma}'^2 (z - z_w)^2. \tag{4.12}$$

We consider two models that describe the distribution of protons in the beamline.

In the first model, we assume that the proton bunch is represented by a single Gaussian distribution in the transverse and longitudinal directions. The transverse distribution of the protons at each light-emitting screen can be described by a bivariate Gaussian distribution with the bunch size, $\boldsymbol{\sigma}(\boldsymbol{\lambda}, \boldsymbol{\nu})$, described by Eq. 4.11

$$I_p(x, y|\boldsymbol{\lambda}, \boldsymbol{\nu}, M_1) = \mathcal{N}(x, y|\boldsymbol{\mu}(\boldsymbol{\lambda}, \boldsymbol{\nu}), \boldsymbol{\sigma}(\boldsymbol{\lambda}, \boldsymbol{\nu})), \tag{4.13}$$

and this model will be further denoted as 'Single Gaussian'. Trajectories of individual particles and the envelop equation for this model are shown in Fig. 4.9.

In the second model, denoted as 'Double Gaussian', we assume that the proton bunch is represented by a mixture of two Gaussian components. These components have the same alignment but different sizes, waist positions, and angular divergence. In this model, the transverse distribution of the protons at the light-emitting screen is defined as

$$\begin{aligned} I_p(x, y|\boldsymbol{\lambda}, \boldsymbol{\nu}, M_2) = \alpha \mathcal{N}(x, y|\boldsymbol{\mu_1}(\boldsymbol{\lambda}, \boldsymbol{\nu}), \boldsymbol{\sigma_1}(\boldsymbol{\lambda}, \boldsymbol{\nu})) + \\ +(1-\alpha)\mathcal{N}(x, y|\boldsymbol{\mu_1}(\boldsymbol{\lambda}, \boldsymbol{\nu}), \boldsymbol{\sigma_2}(\boldsymbol{\lambda}, \boldsymbol{\nu})) \end{aligned} \tag{4.14}$$

where α controls the significance of each contribution. The component with larger angular divergence will be called 'halo' and the smaller 'core', and their parameters will be denoted by subscripts 'c' and 'h', respectively. The single and double Gaussian models are nested, and they predict the same result if $\alpha = 1$.

4.3.3 Camera Modeling

Light with intensity proportional to the number of particles in the bunch is emitted when the proton bunch crosses the light-emitting screen. This light experiences optical smearing that can be represented by a convolution of $I_p(x, y|\boldsymbol{\lambda}, \boldsymbol{\nu})$ with a Gaussian kernel $\mathcal{N}(x, y|0, \tilde{\sigma})$ with zero mean and resolution parameters $\tilde{\sigma}$, i.e.

$$\begin{aligned} \tilde{I}_p(x, y|\boldsymbol{\lambda}, \boldsymbol{\nu}) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} I_p(x-\tau_1, y-\tau_2|\boldsymbol{\lambda}, \boldsymbol{\nu}) \times \\ \times \mathcal{N}(\tau_1, \tau_2|0, \tilde{\sigma}) d\tau_1 d\tau_2. \end{aligned} \tag{4.15}$$

The amount of light captured by one pixel is given by the integral over the pixel surface

$$I = \int_{x}^{x+\Delta x} \int_{y}^{y+\Delta y} \tilde{I}_p(x, y|\boldsymbol{\lambda}, \boldsymbol{\nu}) dx dy. \tag{4.16}$$

We assume that the signal at each pixel is described by a Gaussian probability distribution with the mean given by $i_j I$, and the standard deviation of $f_j \sqrt{I}$, where j denotes the camera index and i_j, f_j are coefficients of proportionality represented by free parameters, i.e.,

$$P_p(d|\boldsymbol{\lambda}, \boldsymbol{\nu}) = \mathcal{N}(d|i_j I, f_j \sqrt{I}). \tag{4.17}$$

This expression is used in Eq. 4.9 to compute the likelihood for one pixel. The likelihoods from all pixels are multiplied together to get the final event likelihood described by Eq. 4.8.

4.3.4 Likelihood Implementation

As discussed before, the likelihood $P(d|\boldsymbol{\lambda}, \boldsymbol{\nu})$ cannot be written in closed form, and it should be computed numerically. The following pseudo-code summarizes the key steps that are used in the likelihood implementation:

```
Input:
    data   : Images from four cameras
    λ      : Proton bunch parameters
    ν      : Nuisance parameters
Output:
    log_likelihood  : Logarithm of the likelihood
```
function log_likelihood(data, λ, ν)
    log_likelihood = 0.0
    for j in camera_ind
        I = compute_bunch_distribution(j, λ, ν)
        I = apply_smearing(I, λ, ν)
        for x in column_ind, y in row_ind
            pdf_signal = Normal(μ=ν[1]*I[x,y], σ=ν[2]*√I[x,y])
            pdf_signal_noise = conv(pdf_signal, pdf_noise)
            log_likelihood += log(pdf_signal_noise[data[j,x,y]])
        end
    end
    return log_likelihood
end
```

This function returns a logarithmic value of the likelihood for better numerical stability. To optimize the performance of the likelihood evaluation, the convolution given by Eq. 4.9 is precomputed in advance for all possible values of the camera signal and accessed from the lookup table.

We sample the posterior distribution given by Eq. 4.6 using Metropolis–Hastings MCMC algorithm. The likelihood is implemented in the Julia programming language, and the BAT.jl [65] package is used for the statistical inference.

### 4.3.5 Model Validation

To validate that the analysis leads to the correct reconstruction of the parameters, we performed the following procedure:

1. True parameters of the models were defined.

2. Simulated events were generated that correspond to these parameters. The simulated events include background noise, lights fluctuations, optical smearing of cameras.

3. The analysis algorithms were applied and the resulting parameters were compared to the true values.

The validation procedure was performed for both single and double Gaussian models. Excellent agreement was found between the extracted and true parameters for both models. An example of the simulated data and the fitted model for the single and double Gaussian bunch densities is presented in Fig. 4.10. A violin plot with the parameter distributions is shown in Fig. 4.11. It can be seen that true parameters are well within the 95% central interval of the posterior distribution for both models.

Fig. 4.11 shows that some of the parameters, such as the alignment of the bunch, can be reconstructed with great accuracy, with uncertainties smaller than 3% of pixel sizes. Another group of parameters, such as the transverse size and angular divergence, is characterized by uncertainties of a few percent. The nuisance parameters that describe the resolution function of the scintillating screen have the largest uncertainty. The camera with the scintillating screen is located at the largest distance from the waist position, and the bunch size is determined primarily by the angular divergence of the beam. The resolution is not a critical parameter, and it is not correlated significantly with the proton bunch parameters.
```

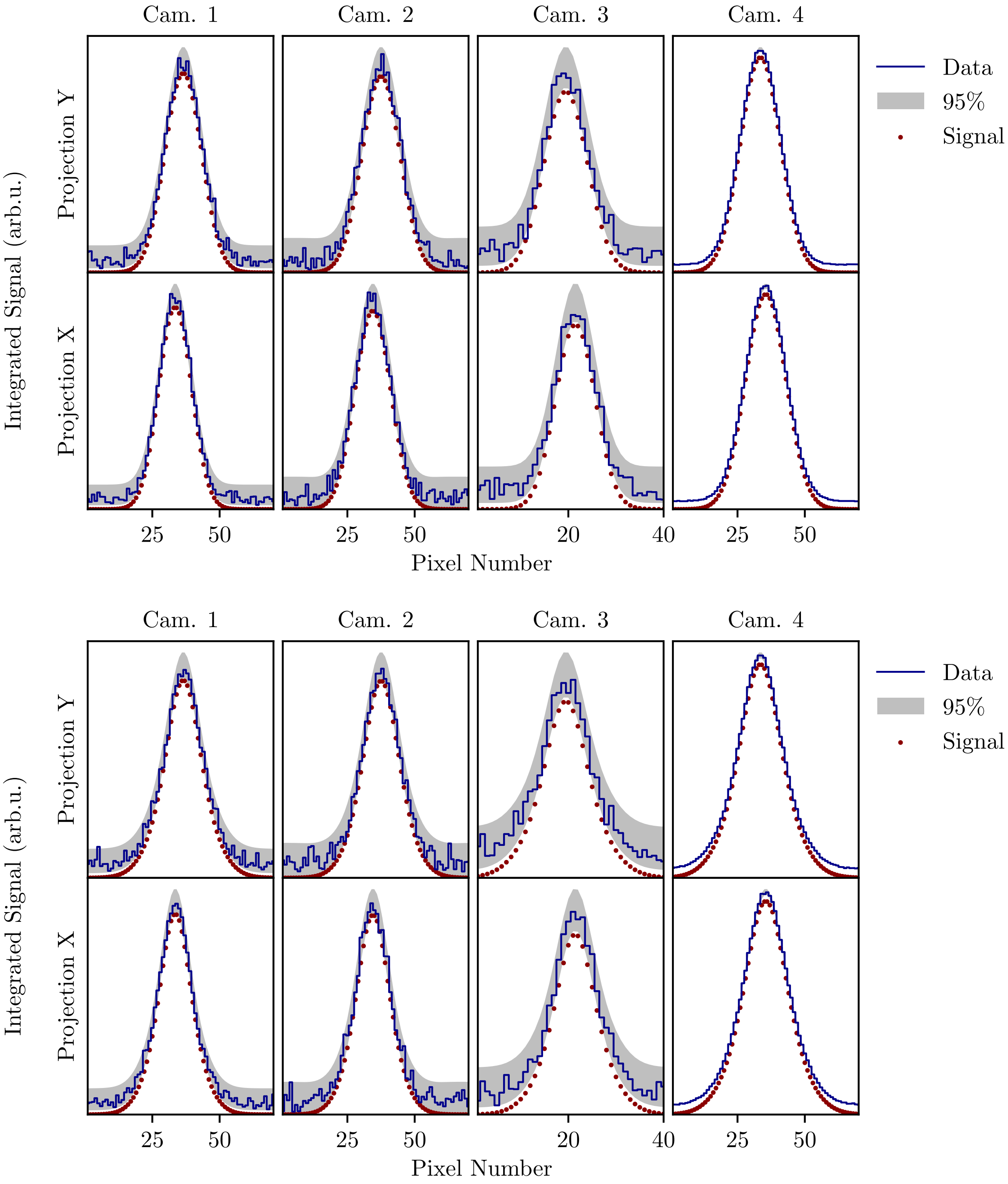


Figure 4.10: A comparison of the simulated data and the best-fit result is shown for the single (top) and double (bottom) Gaussian models. The blue step-lines show an integrated signal over rows/columns of the camera. The grey-filled regions show the 95% central probability intervals of the model including background and signal. The signal from the proton bunch is shown as points. The grey band for Camera 4 overlaps with the data and is not visible.

We have tested the sensitivity of the analysis to the truncation of the data. Pixels that exceed the threshold values determined from the experimental data were discarded from the analysis, and the change in the resulting posterior means is well within the

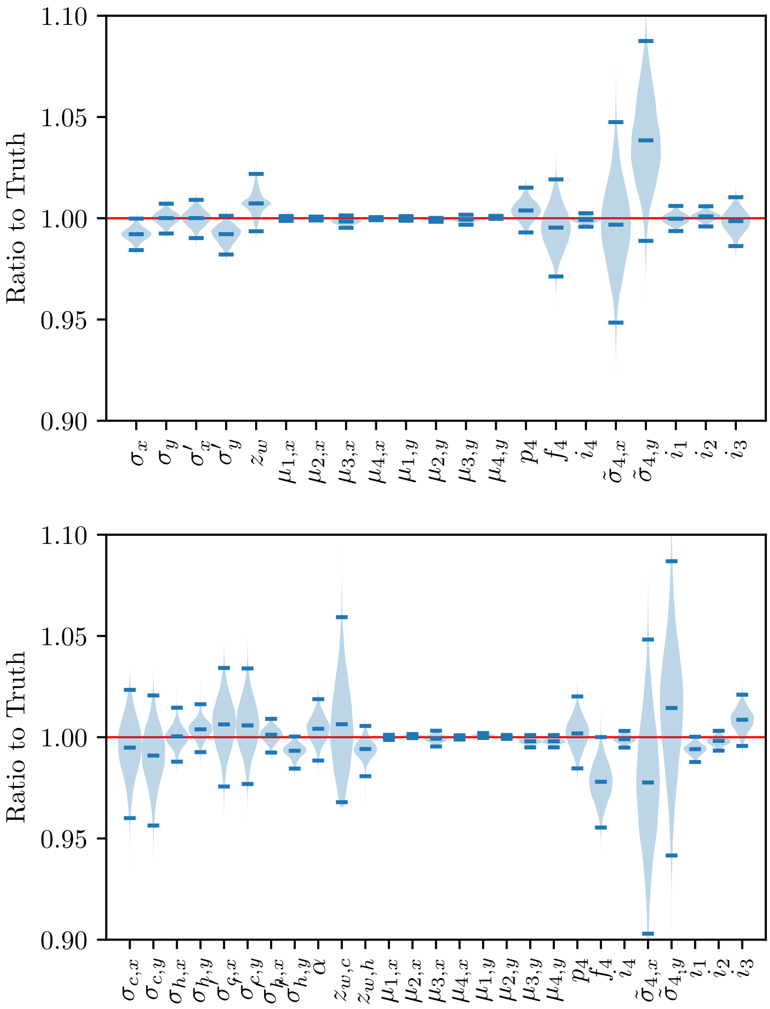


Figure 4.11: The figure illustrates the posterior distribution obtained from the simulated event analysis for the single (top) and double (bottom) Gaussian models. Each parameter is divided by truth to standardize the scale of the error bars. Blue horizontal ticks show 95% central probability intervals and means.

uncertainties of the parameter values.

4.4 Parameters

4.4.1 Model Parameters

The parameters that describe the proton bunch distribution are

$$\begin{aligned} \boldsymbol{\lambda}_{SG} &= \{\boldsymbol{\sigma}, \boldsymbol{\sigma}', z_w\}\,, \\ \boldsymbol{\lambda}_{DG} &= \{\boldsymbol{\sigma}_c, \boldsymbol{\sigma}_h, \boldsymbol{\sigma}'_c, \boldsymbol{\sigma}'_h, z_{w,c}, z_{w,h}, \alpha\}\,, \end{aligned} \tag{4.18}$$

where the two vectors correspond to the single and double Gaussian models, respectively. In addition, the nuisance parameters are

$$\boldsymbol{\nu} = \{\boldsymbol{\mu}_j, \Delta x_j, \Delta y_j, \tilde{\boldsymbol{\sigma}}_j, \boldsymbol{i}_j, \boldsymbol{f}_j, p_4\}\,, \tag{4.19}$$

where $j = 1 \,..\, 4$ denotes the camera index, and the bold font is used for the parameters that have the x and y components. A summary of all the parameters is given in Table 4.5 and Table 4.6 for the single and double Gaussian models, respectively.

The prior probability distributions for the proton bunch parameters are selected based on the AWAKE design report. The priors of the transverse size of the core and halo components of the bunch at the waist position (denoted as $\sigma_{c,x}, \sigma_{c,y}, \sigma_{h,x}, \sigma_{h,y}$) are described by Gaussian probability distributions with means of 0.2 mm and standard deviations of 0.04 mm.

The priors of the angular divergences of the bunch (denoted as $\sigma'_{c,x}, \sigma'_{c,y}, \sigma'_{h,x}, \sigma'_{h,y}$) are described by Gaussian probability distributions with means of 4×10^{-5} rad and standard deviations of 2×10^{-5} rad. Initially, very broad prior ranges were considered for the proton bunch parameters. After learning about the typical locations of the posteriors, the prior ranges were restricted to reduce computational time. We truncate the angular divergence of the core component, denoted as $\sigma'_{c,x}, \sigma'_{c,y}$, to the range $[1\times10^{-5}, 8\times10^{-5}]$ rad; and the halo component, denoted as $\sigma'_{h,x}, \sigma'_{h,y}$, to the range $[1 \times 10^{-5}, 4 \times 10^{-5}]$ rad to clearly identify the halo and core components. The prior probability distributions for the waist positions of the core and halo components, denoted as $z_{w,c}, z_{w,h}$, are described by a Gaussian distribution with means of 2.774 m and standard deviations of 0.03 m. The coefficient that defines the intensity ratio for the halo and core component is denoted as α and is described by a uniform prior.

Parameters that represent the bunch centroid at the camera j along the x and y directions are denoted as $\mu_{x,j}, \mu_{y,j}$, and they are given by uniform prior probability distributions in the ranges specified in Table 4.5 and Table 4.6. The pixel sizes along the x and y directions are denoted as Δx_j and Δy_j, and they are represented by the Dirac delta prior distributions (further denoted as 'constant prior'). The resolution parameters along the x and y directions, denoted as $\tilde{\sigma}_{j,x}, \tilde{\sigma}_{j,y}$, are assumed to be constant for the cameras that have OTR screens. The prior for the camera with the scintillating screen was modeled with a mean of 3 pixels and a standard deviation of 1.5 pixels. The conversion of the proton bunch distribution into the pixel signal is performed by defining proportionality coefficients i_j, that are described by uniform priors. The standard deviations of the Gaussian fluctuations of the light are defined by f_j (see section 4.3.3). This parameter has a constant prior for those cameras where the numerical convolutions of the background and signal are computed. For the last camera, the signal fluctuation (f_4) and the mean of the background (p_4) are kept as free variables with uniform priors.

4.4.2 Sampling Parameters

To approximate the posterior probability distribution given by Eq. 4.6, a Metropolis-Hastings sampler implemented in BAT.jl is used. The sampling process consists of initialization, tuning, and convergence steps, and they require tuning parameters that are listed below:

- To find optimal initial positions for MCMC chains, the MCMC pool initialization strategy is used with the following settings:

  ```
  init = MCMCChainPoolInit(init_tries_per_chain = 50 .. 150, nsteps_init = 1500)
  ```

- To adjust proposal probability distribution for optimal acceptance probability, we use the following tuning parameters:

  ```
  tuning = AdaptiveMHTuning(
          λ = 0.5, α = ClosedInterval(0.15,0.25),
          β = 1.5, c = ClosedInterval(1e-4,1e2), r = 0.5,)
  ```

- The Brooks-Gelman criteria with the following parameters is used to test convergence of multiple MCMC chains:

  ```
  convergence = BrooksGelmanConvergence(threshold = 1.15, corrected = false)
  ```

- During the burning cycle, a proposal distribution is tuned, and chains are tested for convergence. The following tuning settings are considered:

  ```
  burnin = MCMCMultiCycleBurnin(max_ncycles = 160, nsteps_per_cycle = 40000)
  ```

Approximately 60 tuning cycles are needed to tune and converge 4 MCMC chains. The sampling of 1 event with 10^6 samples requires approximately 40 minutes of run-time on a system with a 2.3 GHz Intel Xeon 6140 processor with 12 hyperthreaded CPUs.

4.5 Sampling Results

We analyzed each event summarized in Table 4.2 independently, using single and double Gaussian bunch models. A comparison of the data from one event (with a small bunch population and the bunch rotation OFF) to the best-fit predictions from the two models is shown in Fig. 4.12. It can be seen that the double Gaussian model fits the data more closely than the single Gaussian model. The prediction from the single Gaussian model shows a visible discrepancy with the data in the first, third, and fourth cameras. It is especially evident for the events with a small bunch population. A much better data-model agreement is reached for the double Gaussian model, showing that the data fluctuations are covered reliably by the 95% central interval of the posterior probability distribution. A small disagreement of the distribution for Cam. 4 (see the bottom subplot in Fig. 4.12) resulted from the fact that some of the pixels were discarded from the analysis due to their saturated signals.

In the following, we discuss the results obtained by using the single and double Gaussian proton bunch models.

4.5.1 Analysis of Experimental Data: Single Gaussian Model

Even though this model does not accurately represent the data, we will present the resulting parameters because they provide a simple approximation to the double Gaussian model, i.e., representing average parameters of the halo and core components.

The transverse sizes of the proton bunches at the waist position are illustrated in Fig. 4.13 (upper left). It can be seen that the bunch size is smaller than the prior expectations for all range of bunch populations, and the bunch size increases with the increasing bunch population. The bunch profile is slightly elliptical, with a larger vertical component. The angular spread for individual events is shown in Fig. 4.13 (upper right). It increases with the increasing bunch population, and it is relatively close to the value of the prior mode. The bunch emittance can be computed using the transverse size of the bunch and angular spread as

$$\epsilon = 426.0 \cdot 10^3 \cdot \sigma[\mathrm{mm}] \cdot \sigma'[10^{-5}\ \mathrm{rad}], \tag{4.20}$$

and it is shown in Fig. 4.13 (lower center). It can be seen that the emittance increases with the increasing bunch population. Events with the large population are close to the mode of the prior distribution. A detailed description of the average parameters for a small and large bunch populations is given in Table 4.5.

4.5.2 Analysis of Experimental Data: Double Gaussian Model

The transverse sizes of the halo and core components of the bunch at the waist position are shown in Fig. 4.13 (upper left). The size of the core is larger than the size of the halo for all events, and both components are significantly smaller than the mode of the prior distribution. There is a correlation of the bunch size with the increasing bunch population. The transverse bunch profile is elliptical.

The angular divergences for different events are shown in Fig. 4.13 (upper right). Events with a larger bunch population have a smaller angular divergence of the halo component.

The waist positions of the halo and core components are shown in Fig. 4.13 (lower right). The figure shows that the core component of the bunch is focused much closer to the expected waist position compared to the halo component, which is focused further downstream.

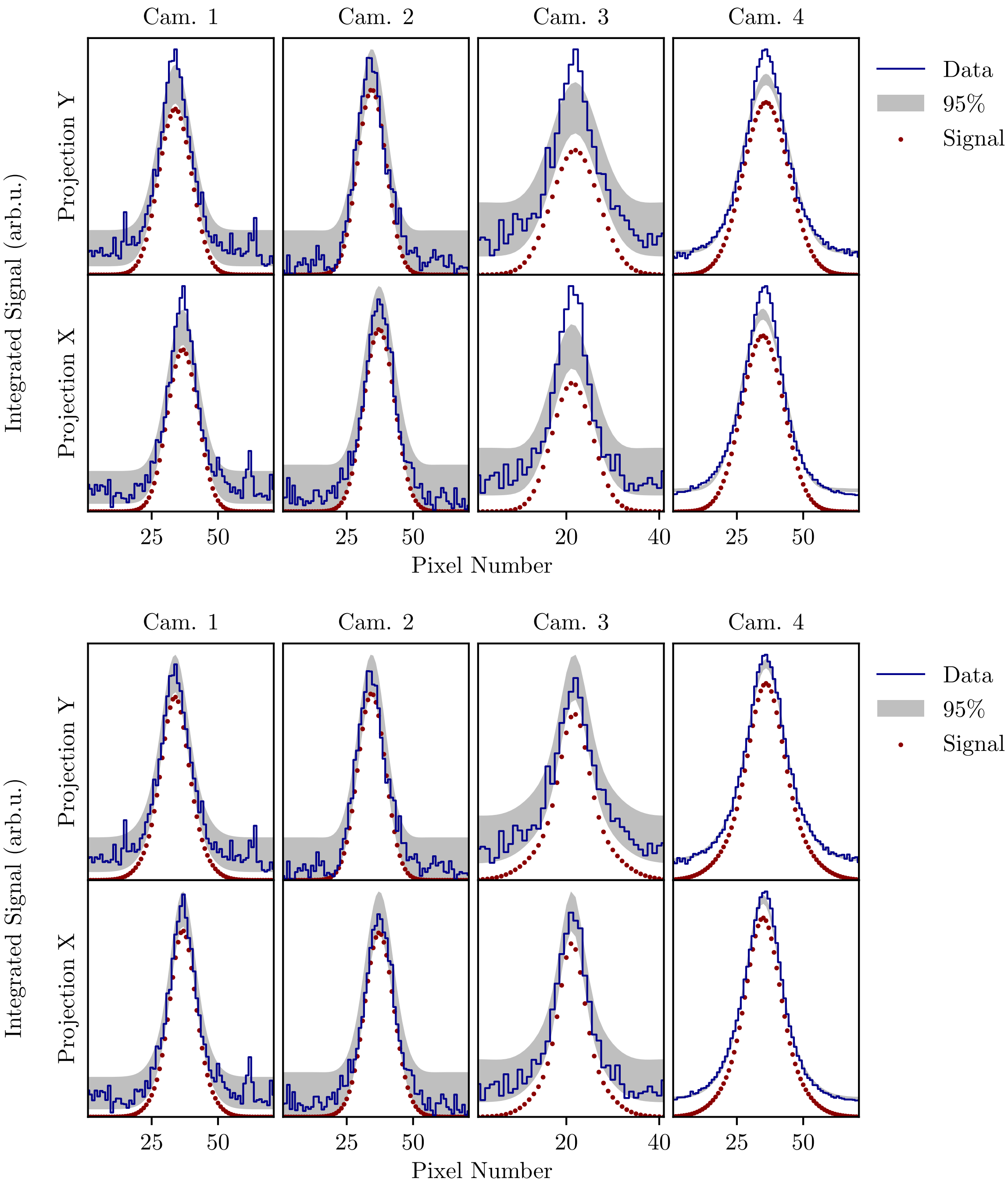


Figure 4.12: A comparison of the experimental data and the best-fit result for the single (top) and double (bottom) Gaussian models. The blue step-lines show data from one event integrated over rows/columns. The model prediction is plotted for the mean value of the posterior. The grey-filled regions show the 95% central probability intervals of the model including background and signal. The grey band for Camera 4 overlaps with the data and is not visible.

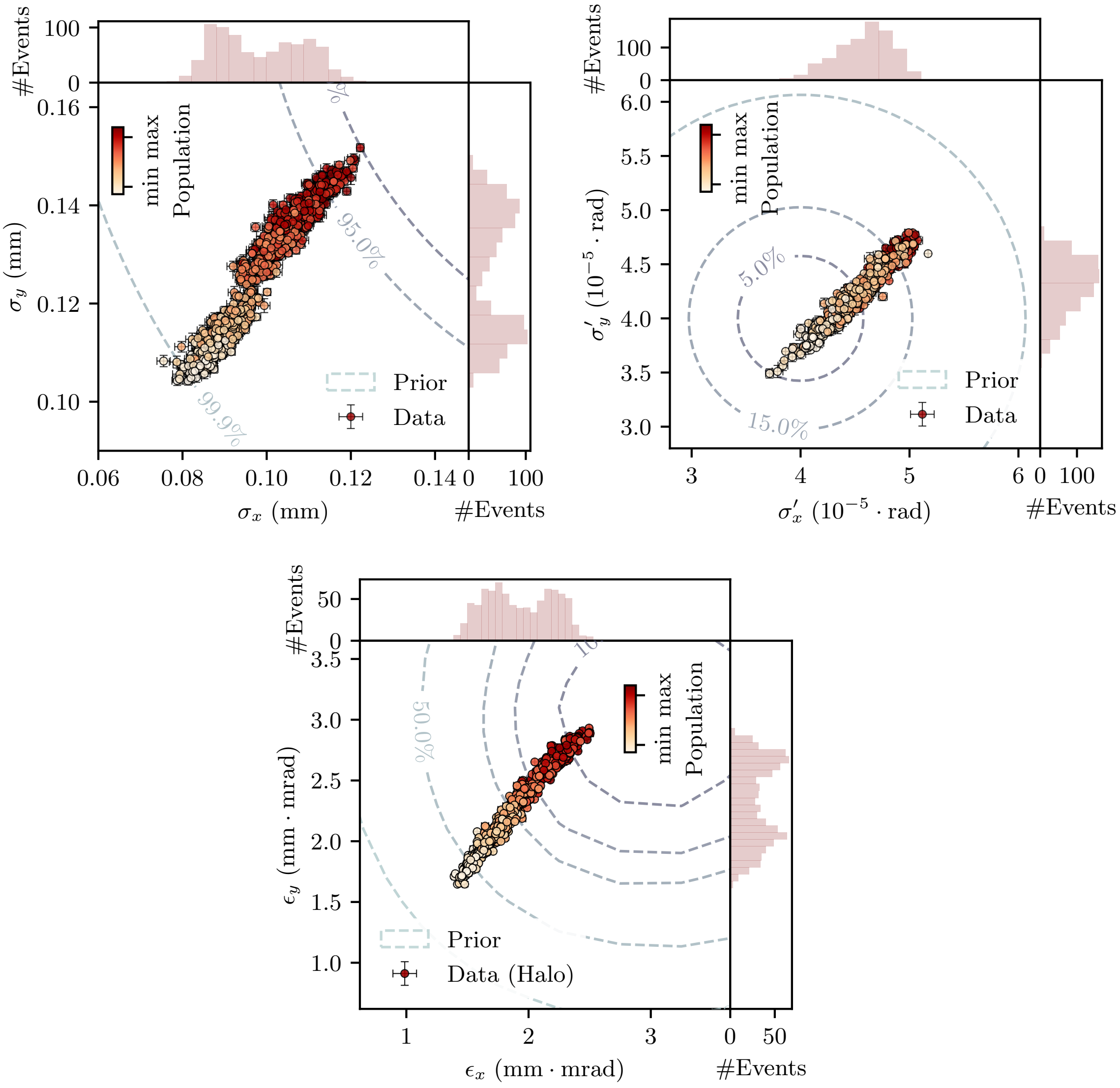


Figure 4.13: The transverse size of the proton bunch at the waist position (upper left), the angular divergence of the bunch (upper right), and the bunch emittance (lower center) are shown for events with different bunch populations using a single Gaussian model. Each event is represented by a symbol, and darker symbols correspond to a larger bunch population. The color scale is non-linear.

Parameter	**Symbol**	**Unit**	**Prior**	**Posterior 1**	**Posterior 2**
Proton Bunch					
Transverse size x	σ_x	mm	$\mathcal{N}(0.20, 0.04)$	0.089 ± 0.004	0.11 ± 0.006
Transverse size y	σ_y	mm	$\mathcal{N}(0.20, 0.04)$	0.11 ± 0.004	0.14 ± 0.006
Angular spread x	σ'_x	10^{-5} rad	$\mathcal{N}(4.0, 2.0)$	4.42 ± 0.24	4.73 ± 0.17
Angular spread y	σ'_y	10^{-5} rad	$\mathcal{N}(4.0, 2.0)$	4.14 ± 0.24	4.43 ± 0.17
Waist position	z_w	m	$\mathcal{N}(2.774, 0.03)$	2.98 ± 0.046	3.2 ± 0.055
Nuisance					
Alignment on Cam. 1, x	$\mu_{1,x}$	px	$[23.0, 48.0]$	35.2 ± 1.79	35.6 ± 2.06
Alignment on Cam. 2, x	$\mu_{2,x}$	px	$[23.0, 48.0]$	36.2 ± 2.06	36.7 ± 2.38
Alignment on Cam. 3, x	$\mu_{3,x}$	px	$[10.0, 30.0]$	21.4 ± 0.362	21.5 ± 0.373
Alignment on Cam. 4, x	$\mu_{4,x}$	px	$[23.0, 48.0]$	35.7 ± 0.508	35.3 ± 0.45
Alignment on Cam. 1, y	$\mu_{1,y}$	px	$[23.0, 48.0]$	35.4 ± 0.484	35.3 ± 1.32
Alignment on Cam. 2, y	$\mu_{2,y}$	px	$[23.0, 48.0]$	36.0 ± 0.483	36.0 ± 1.22
Alignment on Cam. 3, y	$\mu_{3,y}$	px	$[10.0, 30.0]$	20.6 ± 0.275	20.7 ± 0.356
Alignment on Cam. 4, y	$\mu_{4,y}$	px	$[23.0, 48.0]$	34.6 ± 0.201	34.7 ± 0.575
Pixel size on Cam. 1, x	Δx_1	µm	27.1	-	-
Pixel size on Cam. 2, x	Δx_2	µm	21.6	-	-
Pixel size on Cam. 3, x	Δx_3	µm	114.0	-	-
Pixel size on Cam. 4, x	Δx_4	µm	121.8	-	-
Pixel size on Cam. 1, y	Δy_1	µm	30.5	-	-
Pixel size on Cam. 2, y	Δy_2	µm	23.4	-	-
Pixel size on Cam. 3, y	Δy_3	µm	125.0	-	-
Pixel size on Cam. 4, y	Δy_4	µm	120.0	-	-
Resolution effect on Cam. 1, x	$\tilde{\sigma}_{1,x}$	px	1.0	-	-
Resolution effect on Cam. 2, x	$\tilde{\sigma}_{2,x}$	px	1.0	-	-
Resolution effect on Cam. 3, x	$\tilde{\sigma}_{3,x}$	px	1.0	-	-
Resolution effect on Cam. 4, x	$\tilde{\sigma}_{4,x}$	px	$\mathcal{N}(3.0, 1.5)$	4.09 ± 0.635	5.19 ± 0.333
Resolution effect on Cam. 1, y	$\tilde{\sigma}_{1,y}$	px	1.0	-	-
Resolution effect on Cam. 2, y	$\tilde{\sigma}_{2,y}$	px	1.0	-	-
Resolution effect on Cam. 3, y	$\tilde{\sigma}_{3,y}$	px	1.0	-	-
Resolution effect on Cam. 4, y	$\tilde{\sigma}_{4,y}$	px	$\mathcal{N}(3.0, 1.5)$	3.37 ± 0.787	4.9 ± 0.358
Signal amplitude on Cam 1	i_1	counts	$[1.0, 13.0]$	2.79 ± 0.171	8.01 ± 0.391
Signal amplitude on Cam 2	i_2	counts	$[1.0, 17.0]$	3.89 ± 0.238	11.1 ± 0.4
Signal amplitude on Cam 3	i_3	counts	$[1.0, 5.0]$	2.01 ± 0.142	2.51 ± 0.131
Signal amplitude on Cam 4	i_4	counts	$[1.0, 13.0]$	2.55 ± 0.159	8.16 ± 0.324
Signal fluctuations on Cam 1	f_1	one	2.0	-	-
Signal fluctuations on Cam 2	f_2	one	2.0	-	-
Signal fluctuations on Cam 3	f_3	one	2.0	-	-
Signal fluctuations on Cam 4	f_4	one	$[1.0, 3.0]$	1.75 ± 0.197	2.33 ± 0.276
Pedestal on Cam 4	p_4	counts	$[4.0, 40.0]$	22.2 ± 1.06	32.9 ± 2.41
Calculated					
Emittance x	ϵ_x	mm mrad	-	1.7 ± 0.12	2.1 ± 0.16
Emittance y	ϵ_y	mm mrad	-	2.0 ± 0.14	2.6 ± 0.17
Bunch population	q	$10^{10}p^+$	-	9.1 ± 0.6	25.9 ± 0.9

Table 4.5: The table summarizes the parameters used in the analysis of the proton bunch with the single Gaussian model. Parameters are separated into the proton bunch ($\boldsymbol{\lambda}$), nuisance ($\boldsymbol{\nu}$), and calculated categories. The fourth column describes prior probability distributions. If the distribution is uniform on a certain region or truncated, the corresponding region is specified in rectangular parentheses; a single number denotes the argument of the Dirac delta distribution, $\mathcal{N}(\mu, \sigma)$ stands for a Gaussian distribution with a mean μ and a standard deviation σ. The fifth and sixth columns show the mean and standard deviation of the parameters averaged over the datasets with small (1) and large (2) bunch populations.

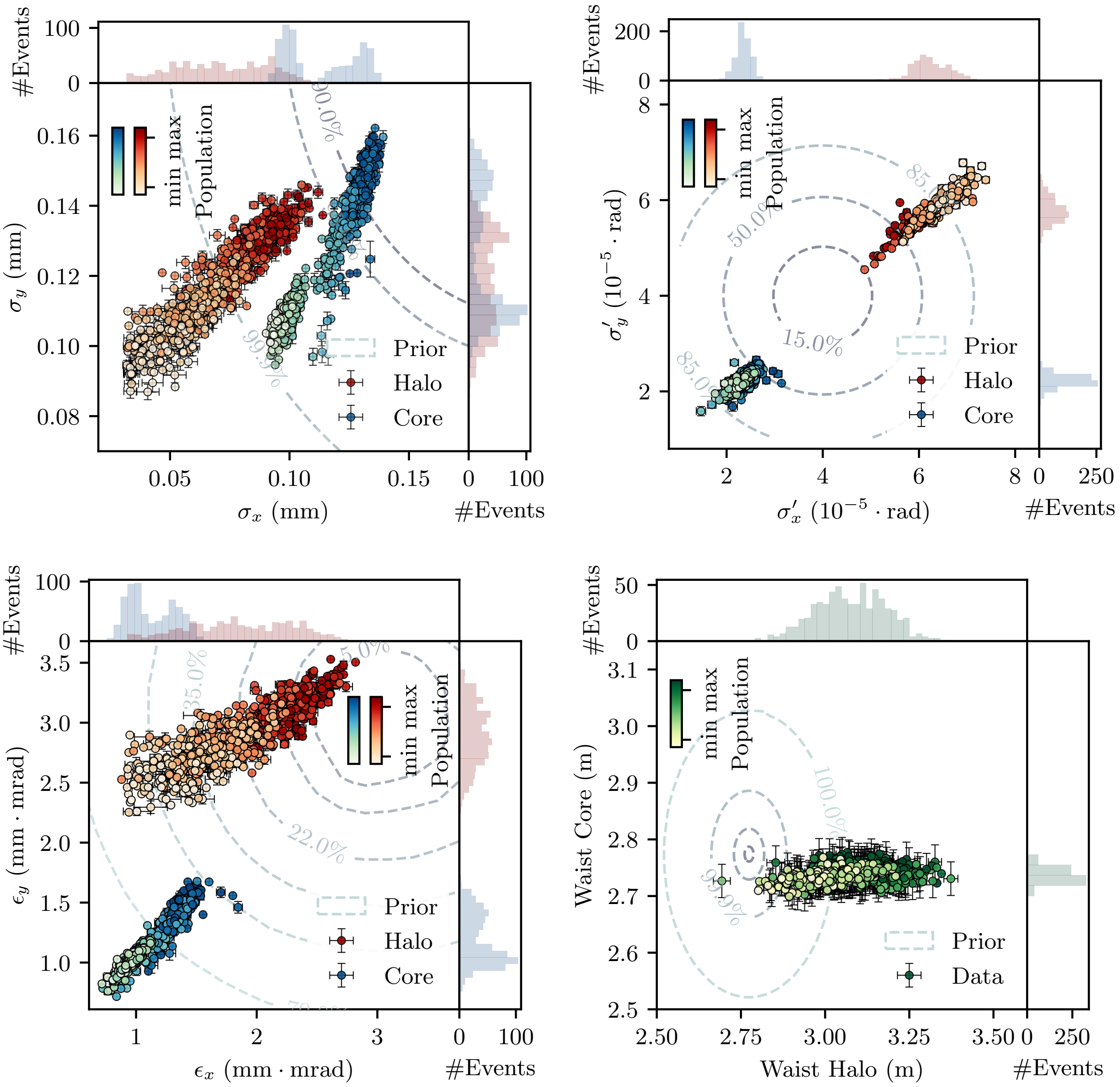


Figure 4.14: The transverse size of the proton bunch at the waist position (upper left), the angular divergence of the bunch (upper right), the bunch emittance (lower left), and the waist positions (lower right) are shown for events with different bunch populations using the double Gaussian model. Blue colors represent the core component, red colors represent the halo component, and green color denotes the bunch population. Darker colors represent a larger bunch population; the color scale is non-linear. .

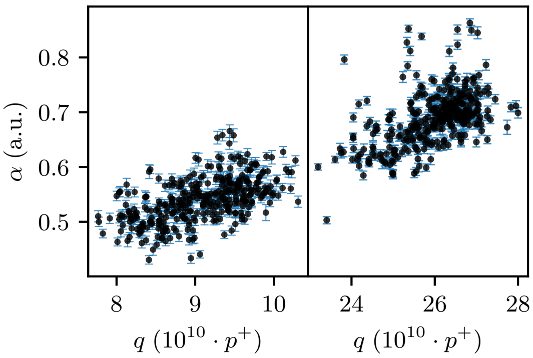


Figure 4.15: Parameter α that defines the significance of the halo component is shown versus the bunch population. A more significant halo component is observed in the bunches with a larger bunch population.

The posterior distributions of the halo and core emittances are summarized in Fig. 4.13 (lower left). It can be seen that the bunch emittance increases with the increasing bunch population for both components. The x component of the halo emittance is more strongly correlated with the bunch population than the y component.

The coefficient that shows the intensity of halo and core components is shown in Fig. 4.15. There is a correlation of α with the proton bunch population that shows that the halo component is more significant for the events with a larger bunch population.

A detailed description of the posterior parameters for small and large bunch populations is given in Table 4.6.

Parameter	**Symbol**	**Unit**	**Prior**	**Posterior 1**	**Posterior 2**
Proton Bunch					
Transverse size x, core	$\sigma_{c,x}$	mm	$\mathcal{N}(0.2, 0.04)$	0.099 ± 0.0031	0.13 ± 0.0065
Transverse size y, core	$\sigma_{c,y}$	mm	$\mathcal{N}(0.2, 0.04)$	0.11 ± 0.0041	0.14 ± 0.012
Transverse size x, halo	$\sigma_{h,x}$	mm	$\mathcal{N}(0.2, 0.04)$	0.056 ± 0.012	0.086 ± 0.011
Transverse size y, halo	$\sigma_{h,y}$	mm	$\mathcal{N}(0.2, 0.04)$	0.11 ± 0.0079	0.13 ± 0.0069
Angular spread x, core	$\sigma'_{c,x}$	10^{-5} rad	$\mathcal{N}(4.0, 2.0)$	2.28 ± 0.14	2.41 ± 0.18
Angular spread y, core	$\sigma'_{c,y}$	10^{-5} rad	$\mathcal{N}(4.0, 2.0)$	2.19 ± 0.11	2.25 ± 0.15
Angular spread x, halo	$\sigma'_{h,x}$	10^{-5} rad	$\mathcal{N}(4.0, 2.0)$	6.5 ± 0.31	5.99 ± 0.26
Angular spread y, halo	$\sigma'_{h,x}$	10^{-5} rad	$\mathcal{N}(4.0, 2.0)$	5.95 ± 0.27	5.6 ± 0.22
Waist position, core	$z_{w,c}$	m	$\mathcal{N}(2.774, 0.03)$	2.73 ± 0.011	2.74 ± 0.013
Waist position, halo	$z_{w,h}$	m	$\mathcal{N}(2.774, 0.03)$	3.01 ± 0.082	3.14 ± 0.081
Intensity ratio	α	one	$[0.25, 1.0]$	0.54 ± 0.041	0.69 ± 0.05
Nuisance					
Alignment on Cam. 1, x	$\mu_{1,x}$	px	$[23.0, 48.0]$	35.2 ± 1.79	35.6 ± 2.06
Alignment on Cam. 2, x	$\mu_{2,x}$	px	$[23.0, 48.0]$	36.2 ± 2.06	36.7 ± 2.38
Alignment on Cam. 3, x	$\mu_{3,x}$	px	$[10.0, 30.0]$	21.4 ± 0.362	21.5 ± 0.373
Alignment on Cam. 4, x	$\mu_{4,x}$	px	$[23.0, 48.0]$	35.7 ± 0.508	35.3 ± 0.45
Alignment on Cam. 1, y	$\mu_{1,y}$	px	$[23.0, 48.0]$	35.4 ± 0.484	35.3 ± 1.32
Alignment on Cam. 2, y	$\mu_{2,y}$	px	$[23.0, 48.0]$	36.0 ± 0.483	36.0 ± 1.22
Alignment on Cam. 3, y	$\mu_{3,y}$	px	$[10.0, 30.0]$	20.6 ± 0.275	20.7 ± 0.356
Alignment on Cam. 4, y	$\mu_{4,y}$	px	$[23.0, 48.0]$	34.6 ± 0.201	34.7 ± 0.575
Pixel size on Cam. 1, x	Δx_1	µm	27.1	-	-
Pixel size on Cam. 2, x	Δx_2	µm	21.6	-	-
Pixel size on Cam. 3, x	Δx_3	µm	114.0	-	-
Pixel size on Cam. 4, x	Δx_4	µm	121.8	-	-
Pixel size on Cam. 1, y	Δy_1	µm	30.5	-	-
Pixel size on Cam. 2, y	Δy_2	µm	23.4	-	-
Pixel size on Cam. 3, y	Δy_3	µm	125.0	-	-
Pixel size on Cam. 4, y	Δy_4	µm	120.0	-	-
Resolution effect on Cam. 1, x	$\tilde{\sigma}_{1,x}$	px	1.0	-	-
Resolution effect on Cam. 2, x	$\tilde{\sigma}_{2,x}$	px	1.0	-	-
Resolution effect on Cam. 3, x	$\tilde{\sigma}_{3,x}$	px	1.0	-	-
Resolution effect on Cam. 4, x	$\tilde{\sigma}_{4,x}$	px	$\mathcal{N}(3.0, 1.5)$	4.6 ± 0.15	4.7 ± 0.19
Resolution effect on Cam. 1, y	$\tilde{\sigma}_{1,y}$	px	1.0	-	-
Resolution effect on Cam. 2, y	$\tilde{\sigma}_{2,y}$	px	1.0	-	-
Resolution effect on Cam. 3, y	$\tilde{\sigma}_{3,y}$	px	1.0	-	-
Resolution effect on Cam. 4, y	$\tilde{\sigma}_{4,y}$	px	$\mathcal{N}(3.0, 1.5)$	4.1 ± 0.13	4.5 ± 0.2
Signal amplitude on Cam 1	i_1	counts	$[1.0, 13.0]$	3.0 ± 0.185	8.31 ± 0.329
Signal amplitude on Cam 2	i_2	counts	$[1.0, 17.0]$	3.9 ± 0.241	11.1 ± 0.381
Signal amplitude on Cam 3	i_3	counts	$[1.0, 5.0]$	2.51 ± 0.168	2.67 ± 0.117
Signal amplitude on Cam 4	i_4	counts	$[1.0, 13.0]$	2.8 ± 0.17	8.59 ± 0.303
Signal fluctuations on Cam 1	f_1	one	2.0	-	-
Signal fluctuations on Cam 2	f_2	one	2.0	-	-
Signal fluctuations on Cam 3	f_3	one	2.0	-	-
Signal fluctuations on Cam 4	f_4	one	$[1.0, 3.0]$	1.62 ± 0.22	1.78 ± 0.414
Pedestal on Cam 4	p_4	counts	$[4.0, 40.0]$	18.4 ± 1.04	27.7 ± 2.97
Calculated					
Emittance x, core	$\epsilon_{c,x}$	mm mrad	-	0.96 ± 0.07	1.3 ± 0.13
Emittance y, core	$\epsilon_{c,y}$	mm mrad	-	1.02 ± 0.074	1.4 ± 0.16
Emittance x, halo	$\epsilon_{h,x}$	mm mrad	-	1.5 ± 0.3	2.2 ± 0.27
Emittance y, halo	$\epsilon_{h,y}$	mm mrad	-	2.7 ± 0.18	3.1 ± 0.17
Bunch population	q	$10^{10} p^+$	-	9.1 ± 0.6	25.9 ± 0.9

Table 4.6: The table summarizes the parameters used in the analysis of the proton bunch with the double Gaussian model. Parameters are separated into the proton bunch ($\boldsymbol{\lambda}$), nuisance ($\boldsymbol{\nu}$), and calculated categories. The fourth column describes prior probability distributions. If the distribution is uniform on a certain region or truncated, the corresponding region is specified in rectangular parentheses; a single number denotes the argument of the Dirac delta distribution, $\mathcal{N}(\mu, \sigma)$ stands for a Gaussian distribution with a mean μ and a standard deviation σ. The fifth and sixth columns show the mean and standard deviation of the parameters averaged over the datasets with small (1) and large (2) bunch populations.

4.5.3 Stability Analysis

Data Truncation

To estimate how the truncation of saturated pixels affects posterior distributions of the bunch parameters, we applied the analysis to one simulated and one experimental event with and without data truncation. The result of the simulated event analysis is shown in Fig. 4.16 (top), and it indicates that the mean of the posterior has changed with an absolute value of less than 2%. The result of the experimental event analysis is shown in Fig. 4.16 (bottom). It can be seen that the data truncation does not impact significantly on the posterior bunch parameters.

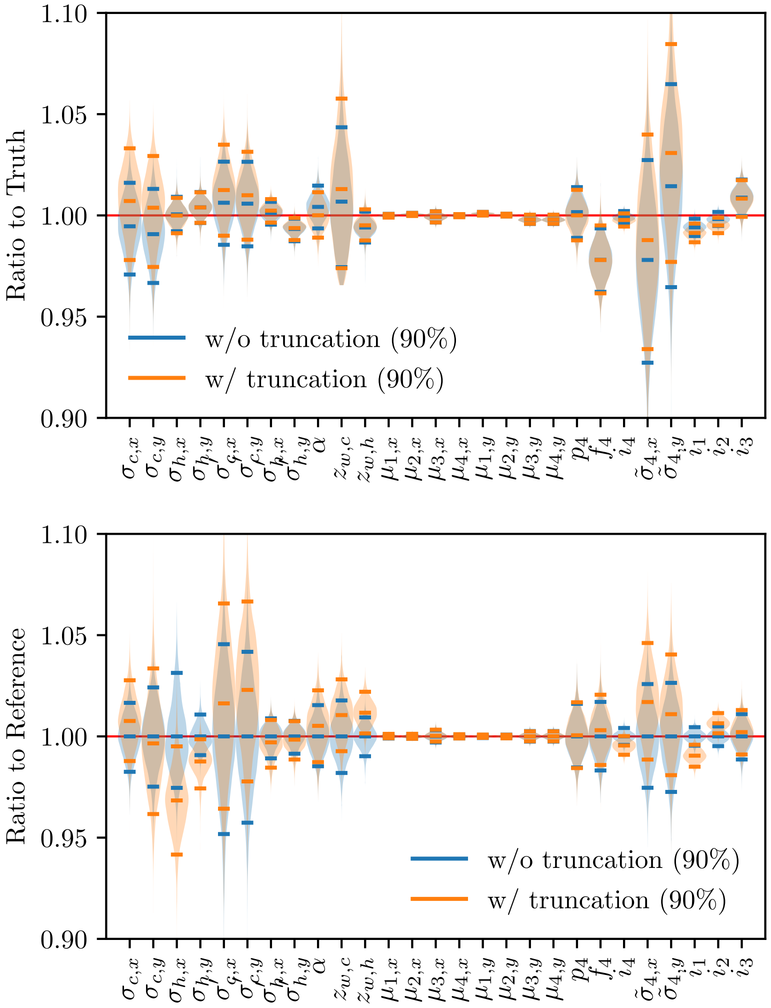


Figure 4.16: The figure illustrates the posterior distribution with and without truncation of saturated pixels for the simulated event (top), and experimental event (bottom). Each parameter is divided by truth to standardize the scale of the error bars. Horizontal ticks represent 5%, 50%, 95% quantiles of the probability distributions. The reference in the bottom subplot represents the mean of the distribution.

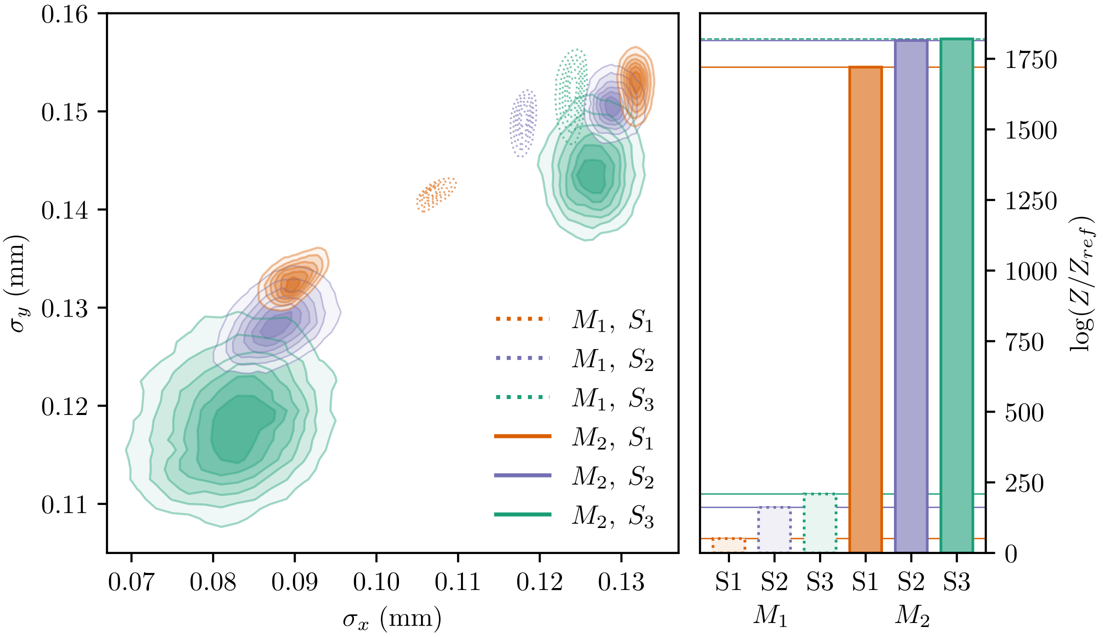


Figure 4.17: Posterior distributions of the transverse bunch size using different priors on experiment-specific parameters. S_1 denotes a setting where pixel sizes and resolution functions are constant. S_2 denotes a setting where pixel sizes are free parameters and resolution functions are constant. S_3 denotes a setting where pixel sizes and resolution functions are free parameters. The single Gaussian model (M_1) is shown in dotted lines, and the double Gaussian model (M_2) is shown in solid lines. Evidence values for each setting are presented in the right subplot.

Stability and Postprocessing

To determine whether the bunch profile has a rotation in the transverse plane, the model was extended with an additional rotation parameter ϕ, that acts on the coordinates x, y as

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos(\phi) & -sin(\phi) \\ sin(\phi) & cos(\phi) \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}, \tag{4.21}$$

where x' and y' show a position in the rotated coordinate system. We analyzed a few randomly selected events to determine posterior distributions of ϕ. The results indicate that the rotation parameter has absolute values close to zero and does not significantly impact posterior distributions of other parameters of the model.

Nuisance Parameters

To determine the impact of the experiment-specific parameters on the posterior distributions of the proton bunch parameters, we analyzed one event without constraining pixel size and resolution function to the constant values. The following configurations of parameters are considered: (S1) Pixel sizes and resolution functions are constant; (S2) Resolution functions are free parameters, and pixel sizes are constant; (S3) Resolution and pixel sizes are free parameters. These settings are used with both single and double Gaussian models. The prior probability distributions for the pixel sizes and resolution

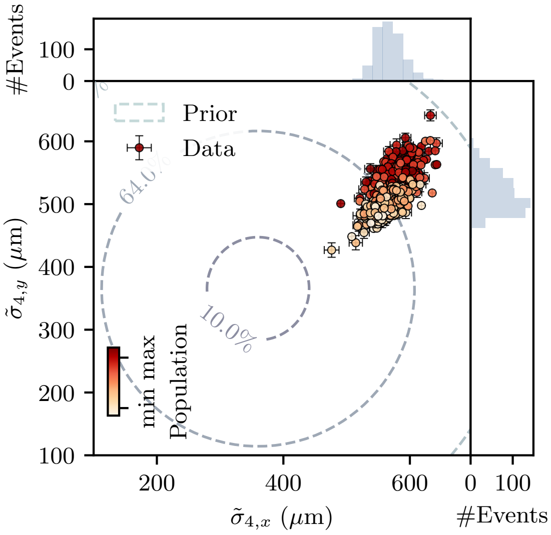


Figure 4.18: Posterior distributions of the resolution parameters for Cam. 4. Each event is represented by one symbol that indicates the posterior mean and standard deviation. Darker colors correspond to a larger bunch population. The color scale is non-linear.

functions are constructed using Table 4.3 and Table 4.4. The resulting posterior distributions for the transverse bunch size for one event are shown in Fig. 4.17. As the figure shows, the size of the halo and core components decreases with the increasing number of free parameters for the double Gaussian model; and it increases for the single Gaussian model. At the maximal number of free parameters, the size of the bunch predicted by the single Gaussian model is equal to the size of the core component of the double Gaussian model. The uncertainty of transverse bunch size increases with an increasing number of free parameters.

To determine which model best fits our data, we estimate Bayesian evidence using the AHMI integration algorithm. The single and double Gaussian models are nested; therefore, absolute values of evidence integrals can be compared. As can be seen from Fig. 4.17, the double Gaussian model has much larger evidence compared to the single Gaussian model. The inclusion of free resolution slightly increases the evidence, but it saturates when the pixel sizes are unconstrained.

The prior and posterior distributions of the resolution parameters are presented in Fig. 4.18. The average resolution effect is $\sim$ 568 µm and $\sim$ 518 µm in the x and y projection, respectively. These values are larger than prior expectations, and the systematic study of the optical and scintillating resolutions is a subject for future study.

Correlations Between Posterior Parameters

Given a multidimensional dataset, correlation matrices can be used to visualize relations between model parameters, i.e., which parameter influence each other and how strongly. A correlation matrix constructed from the samples of one randomly selected event is shown in Fig. 4.19 (top). It indicates that the sizes of the halo and core components of the bunch are inversely proportional; the waist position of the core component is weakly affected by other parameters compared to the waist position of the halo component, which is strongly affected by the halo size and angular divergence.

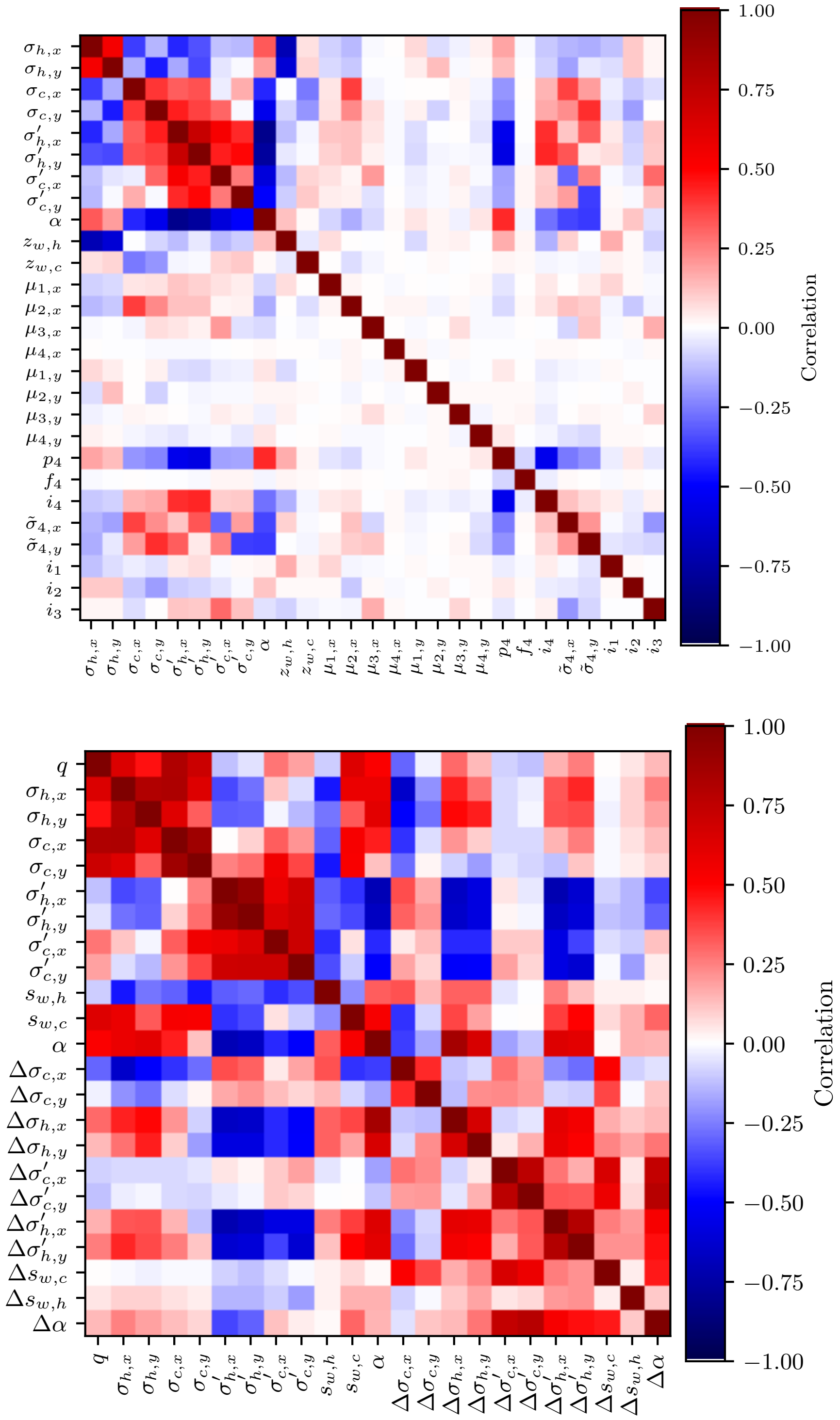


Figure 4.19: Top: A correlation matrix of posterior parameters from one randomly selected event using the double Gaussian model. Bottom: A correlation matrix of the mean posterior parameters constructed from multiple events with a large bunch population using the double Gaussian model.

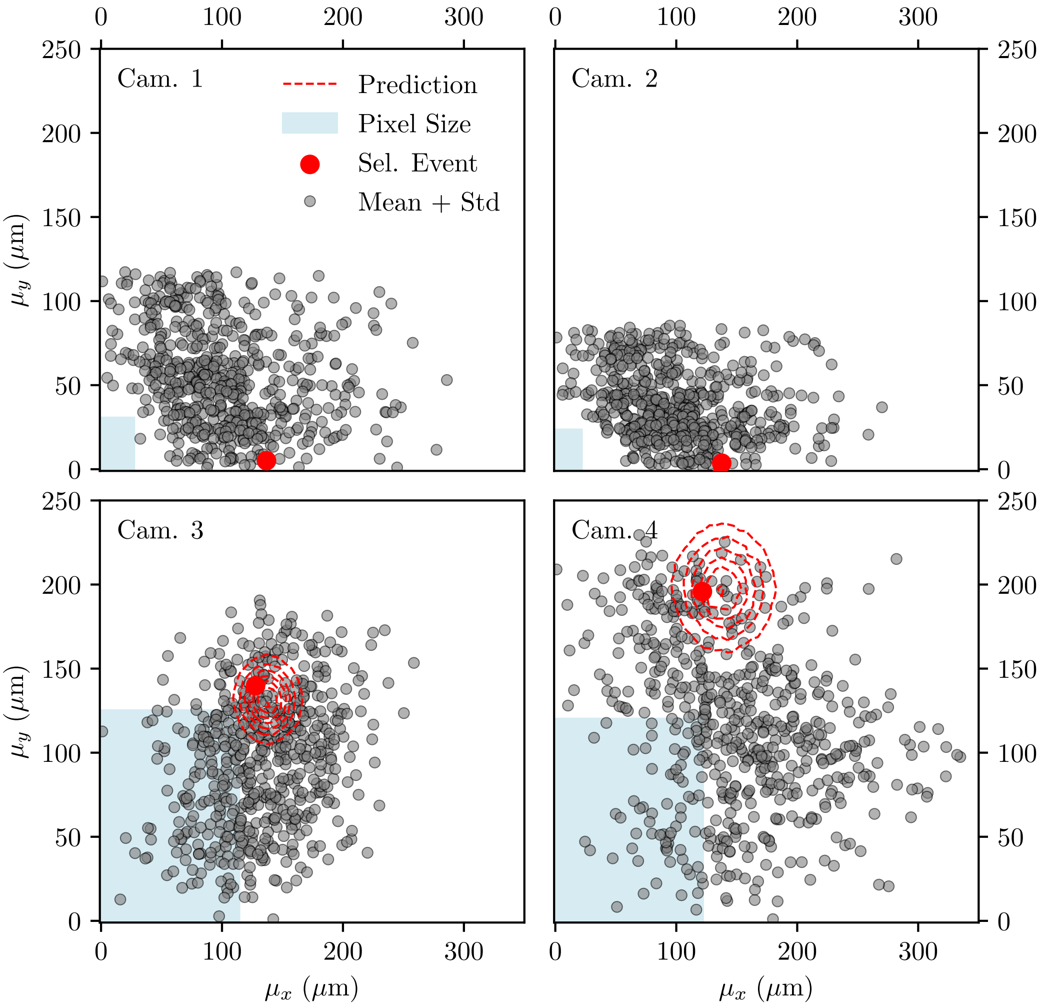


Figure 4.20: The figure illustrates the positions of the bunch centroids at each screen. An example of trajectory prediction for one event (marked by the red point) is shown in dashed contours. Blue shaded regions show the sizes of the pixels.

A correlation matrix constructed from the mean parameters of the dataset with a large bunch population is shown in Fig. 4.19 (bottom). It demonstrates that the sizes of the halo and core components of the bunch are directly proportional to the bunch population. It also shows that bunches with larger charge are typically focused further downstream for the core component and further upstream for the halo component.

Effect of the Longitudinal Bunch Rotation on Bunch Parameters

As discussed in Sec. 4.1, the dataset $\{D\}$ has been measured with 4 experimental settings, i.e., small and large bunch populations and bunch rotation ON and OFF.

It is easy to notice differences in parameters between events with large and small bunch populations by looking at posterior distributions. Namely, events from these two categories are characterized by different transverse sizes, angular divergences, and emittances, and they can be trivially distinguished by eye, as shown, for example, in Fig. 4.14. However, it is not possible to see if there are some trends in parameters if the longitudinal bunch shortening is applied simply by looking at 2-dimensional posterior distributions.

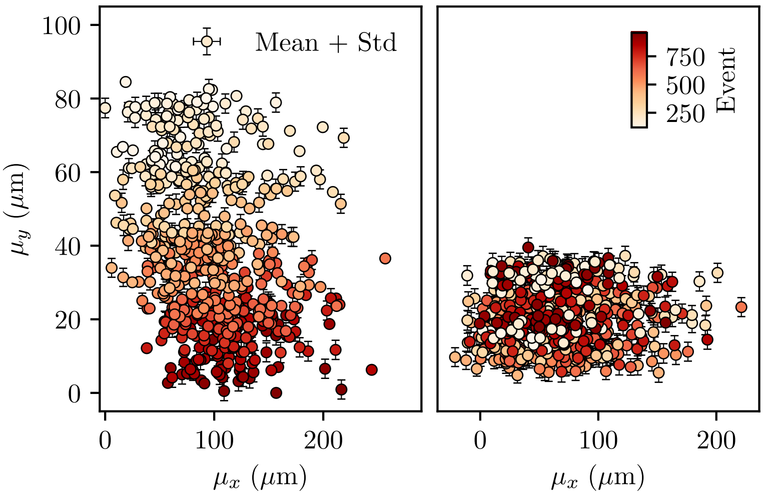


Figure 4.21: Positions of the bunch centroids at the waist. The left plot shows the drift of the bunch centroid with the time of the measurement. The right plot shows the same data with subtracted time drift.

Multivariate analysis can be more effective for such a problem by looking at samples in many-dimensional spaces instead of two-dimensional. A binary classifier that consists of 2 dense layers was trained to distinguish events with bunch rotation ON and OFF. If the bunch rotation has no impact on the samples, then the classification accuracy should equal a random guess, i.e., 50% (see also Section 2.2.3). Otherwise, the classification accuracy should be more than 50%.

The classification test indicated that the ROC curve is close to the diagonal line for training and testing datasets. This tells that neither single nor double Gaussian models allow detecting differences in the bunch profile when the longitudinal bunch shortening is applied. To include information about the longitudinal structure of the bunch, a more detailed model should be constructed, e.g., utilizing information from the streak cameras.

4.5.4 Bunch Drift

In the analysis, we use nuisance parameters to determine the coordinates of the bunch at each screen. These coordinates can be combined to determine the relative alignments of the screens together with polar and azimuthal angles of the individual bunch centroids. We use Hamiltonian Monte Carlo to fit individual bunch centroids and approximate 1350-dimensional posterior distribution. An example of the resulting trajectory for one event is shown in Fig. 4.20. As the figure shows, the model predicts the position of the bunch centroid accurately, with uncertainty smaller than the pixel size.

By propagating individual bunch centroids to the waist position, the drift and jitter of the bunch can be computed. Fig. 4.21 shows that the bunch center drifts as a function of event number, which is proportional to the time at which measurement occurred. The standard deviation at the waist is $\sigma(\mu_x) \approx 43\,\mu\text{m}$, $\sigma(\mu_y) \approx 20\,\mu\text{m}$. Once the time-drift of the bunch is subtracted, the resulting jitters are $\sigma(\mu_x) \approx 41\,\mu\text{m}$ and $\sigma(\mu_y) \approx 8\,\mu\text{m}$.

4.6 Self-Modulation of the Proton Bunch in Plasma

As shown above, the measurement of proton bunch parameters indicates that the bunch has a non-Gaussian radial distribution and considerably smaller radial size at the waist compared to the AWAKE baseline model. In this section, by using numerical plasma modeling we show how the bunches described by the measured parameters self-modulate when they propagate in the plasma. We also demonstrate the difference between the wakefields created by the protons with the baseline and measured parameters.

The modeling is performed by John Farmer (the results are presented in [173, 174]) using the QV3D code [175] that is built on the VLPL platform [176].

4.6.1 Description of the QV3D Code

The QV3D code is a three-dimensional quasi-static Particle-in-Cell (PIC) code that allows modeling non-linear processes in plasmas. The PIC codes rely on the kinetic model of plasma, which treats plasma as an ensemble of charged particles. These particles deposit charge or current, which can be used to calculate the fields, which are then used to update the particles. In the PIC codes, however, particles are not physical, and they represent clusters of real particles with the same charge-to-mass ratio. Using macro-particles instead of the real ones allows a drastic decrease in the number of degrees of freedom, thus speedup performance and applicability of the code to large-scale problems. Over the past decade, plasma modeling that uses PIC codes has proven to be very reliable and successful in many applications including AWAKE [177, 178, 179, 180].

The quasi-static condition imposed in the QV3D code allows separating slow and fast processes in plasma to accelerate simulations [181]. For example, the smallest scale of the plasma-based accelerators typically corresponds to the laser wavelength (order of micrometers) or the plasma wavelength (tens of micrometers to millimeters). The driver bunches can be of a medium scale, e.g., $6 - 12$ cm. Finally, the largest scale is typically the length of the accelerating regions, and it can be from a few centimeters to kilometers [182]. The quasi-static approximation assumes that the driver changes slowly over distances compared to its length. Therefore the driver can be advanced with the larger time steps compared to the plasma particles. This approximation allows simulating the evolution of drivers in plasma over longer distances compared to the plasma wavelength, extending the applicability of codes to real-world problems. The limitation of this approach is that it cannot describe fast in time radiation processes, limiting applications only to static electromagnetic fields.

Codes that use two-dimensional plasma models are much faster than three-dimensional, and they give the same accuracy as three-dimensional for axisymmetrical problems. These codes, however, do not allow to simulate off-axis witness bunch injection (as used in part of the AWAKE Run 1), drivers with non-round shape, and other non-axisymmetrical problems. As discussed earlier, measurements show that the transverse shape of the proton bunch is elliptical. To test how these bunches self-modulate in plasma, a three-dimensional code should be used.

4.6.2 Plasma Modeling Results

Plasma simulations are performed for three proton bunch models: (1) Single Gaussian with measured parameters, (2) double Gaussian with measured parameters, and (3) single Gaussian with baseline parameters (further referred to as ‘baseline model’). To select optimal input parameters for the first two models, we average the parameters over events

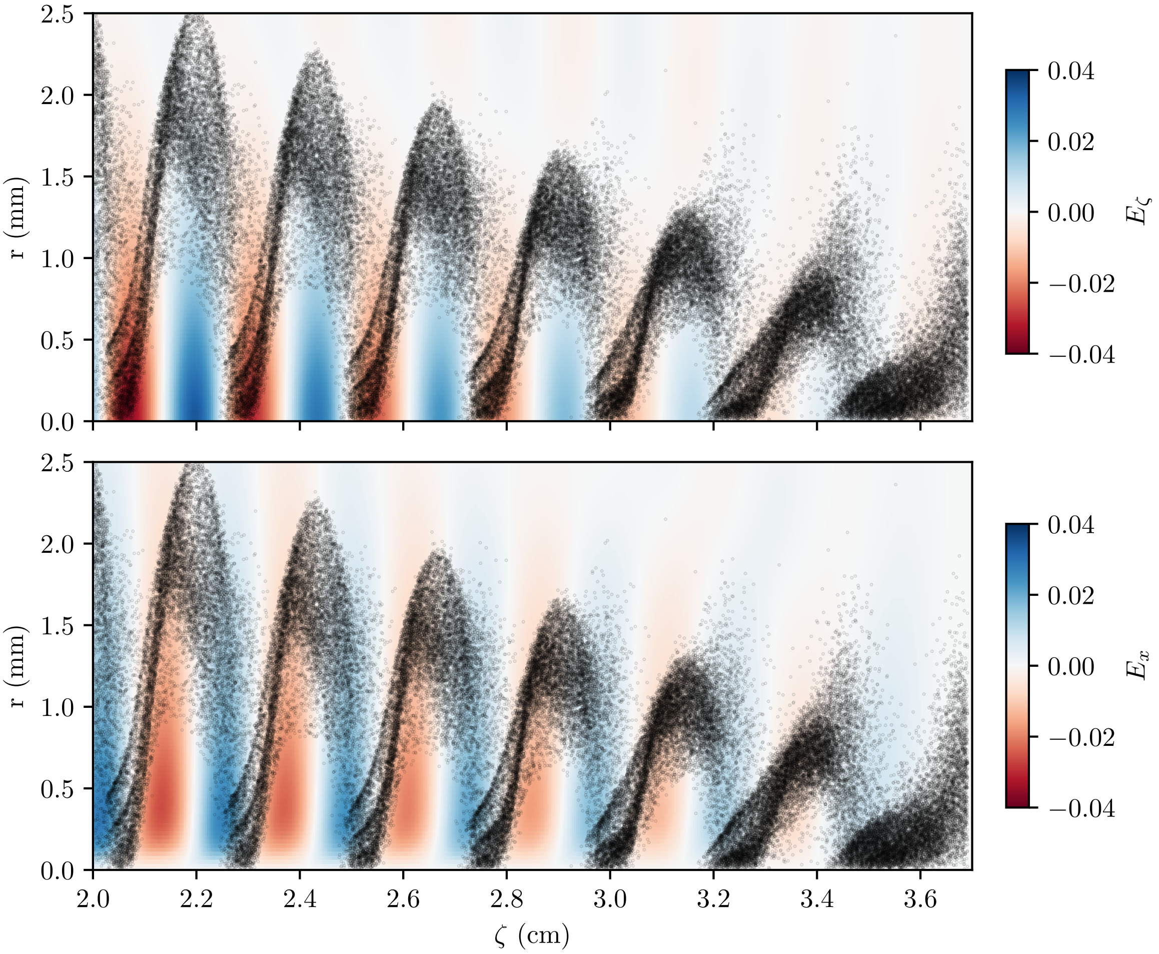


Figure 4.22: The figure illustrates the proton bunch self-modulation at the plasma exit for the single Gaussian bunch model. The upper and bottom subplots show longitudinal and radial plasma electric fields (in the Cartesian coordinate system). Black dots show the radial positions of protons.

with a large bunch population. All models assume that the bunch is Gaussian in the longitudinal direction with $\sigma_z = 325\,\mathrm{ps}$ ($\sim 9.74\,\mathrm{cm}$). The relativistic ionization front is simulated by a sharp cut at the midpoint of the bunch, so only half of the bunch travels in the plasma. The window length is $100/k_p$ ($\sim 3.73\,\mathrm{cm}$), and it represents only a section of the bunch. The density of the plasma is $2.03\,\mathrm{cm}^{-3}$. The cell size is $(0.1 \times 0.025 \times 0.025)/k_p$, with a plasma timestep of $0.025/\omega_p$ and a beam timestep of $500/\omega_p$. The window size is $(100 \times 12.8 \times 12.8)/k_p$, and the plasma has a radius $5.36/k_p$ ($\sim 2\,\mathrm{mm}$). There are on average 400 beam particles per cell on the axis at the focus for the single Gaussian bunch model.

An example of self-modulation of the proton bunch in plasma is shown in Fig. 4.22. It can be seen that an initially long proton bunch is divided into a group of micro bunches located on-axis in the focusing phases of the wakefields. Particles that appeared in the defocusing phases of the field are repulsed from the center of the bunch, forming a defocused halo. It can be seen that the amplitude of the longitudinal field is stronger than the amplitude of the transverse field, and that the strength of the field increases from the head of the bunch to its tail.

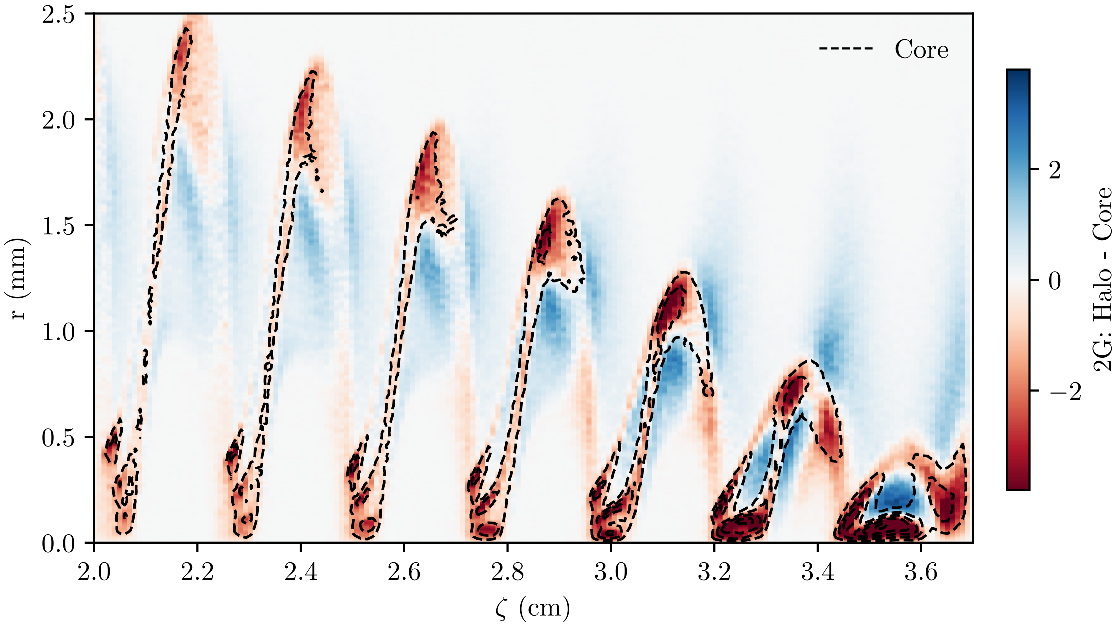


Figure 4.23: The density difference between the halo and core components of the modulated bunch represented by the double Gaussian model. Dashed lines illustrate the contours of the core component.

Halo and core components of the double Gaussian model are treated in the plasma modeling as particles of two different species, and the self-modulation of both of them can be visualized separately. Histograms of the radial coordinates of the self-modulated halo and core components are compared in Fig. 4.23. The core component of the unmodulated bunch has a smaller angular spread, and the analysis predicts that it is focused more strongly at the plasma exit than the halo component. As follows from Fig. 4.23, core microbunches have a more dense structure compared to the halo microbunches. It can be also seen that the core component has a smaller defocusing region at the head of the bunch and larger at the tail of the bunch compared to the halo component.

A comparison of self-modulated protons for different bunch models is shown in Fig. 4.24. The upper subplot shows that the single Gaussian model has a noticeably larger radial defocusing of protons in the bunch tail compared to the baseline model. It also shows that the first few microbunches of the single Gaussian model are more focused compared to the baseline. To visualize contribution from the radially focused protons and phasing of the bunch, the effective current can be computed

$$I_{eff}(\zeta) = 4\pi \int_0^\infty \rho(r,\zeta) K_0(k_p r) r dr. \tag{4.22}$$

As Fig. 4.25 shows, the amplitude of the effective current for the baseline model is smaller at the head of the bunch and larger at the tail of the bunch. As Fig. 4.24 and Fig. 4.25 show, the main difference in the self-modulation of the protons described by the single and double Gaussian models is in the first few microbunches. Namely, the double Gaussian model predicts a more tightly focused first two microbunches and slightly more defocused halo compared to the single Gaussian model.

A comparison of the maximum longitudinal electric fields versus a propagation distance in the plasma for different bunch models is shown in Fig. 4.26. It can be seen that

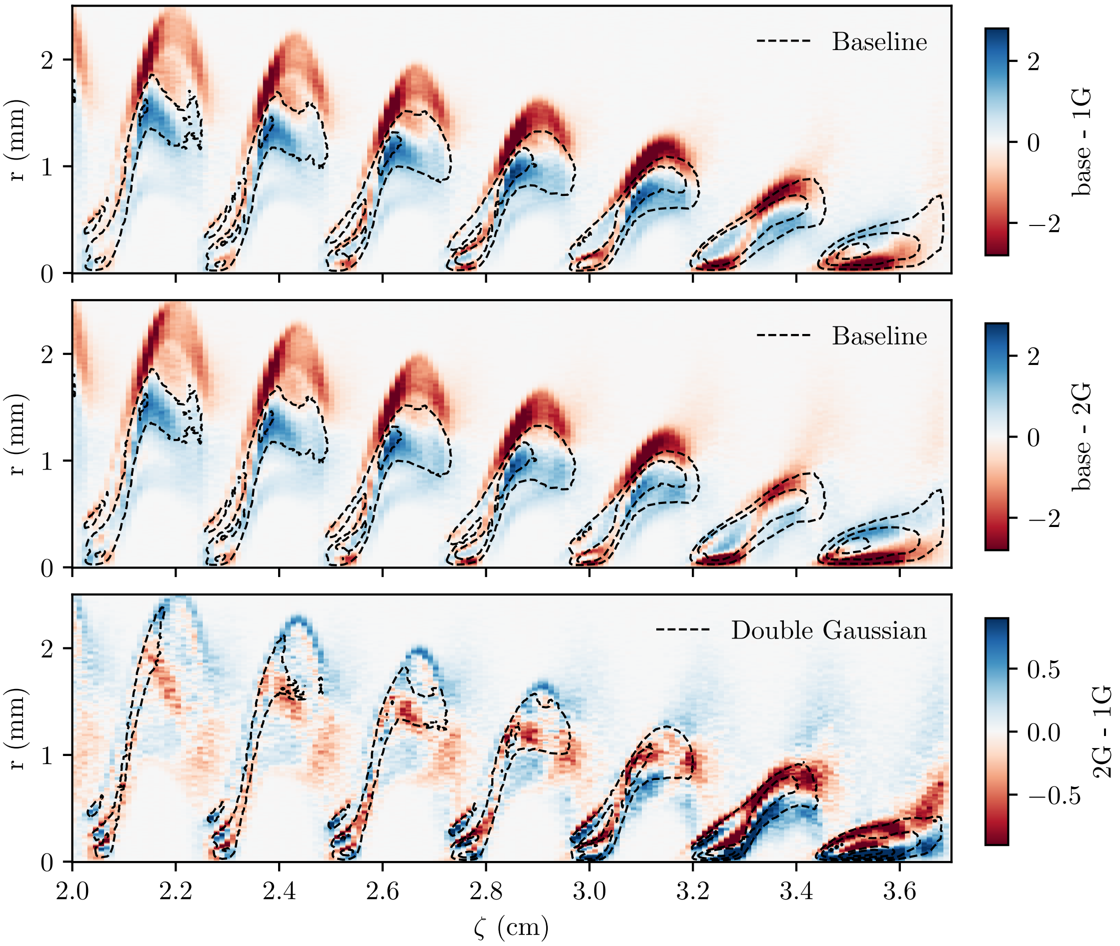


Figure 4.24: The density difference between modulated proton bunches described by three models. ‘base’ denotes the single Gaussian model with baseline parameters. ‘1G’ and ‘2G’ denote the single and double Gaussian models with measured parameters, respectively.

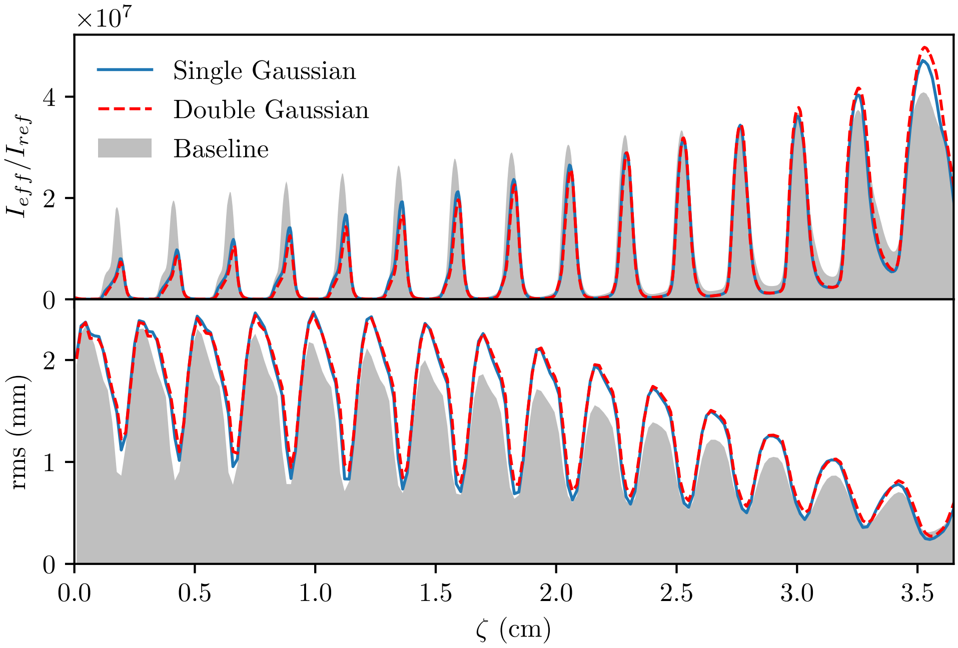


Figure 4.25: The effective current (top) and r.m.s. size (bottom) of the modulated proton bunch versus the position along the bunch for measured and baseline parameters.

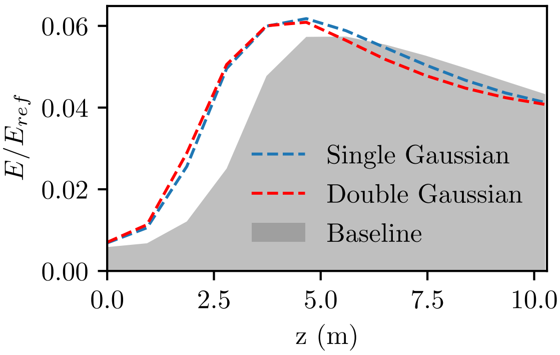


Figure 4.26: The longitudinal electric field versus the propagation distance for different bunch models.

the maximum field is 7% larger for the bunches with measured parameters compared to the bunches with baseline parameters. Also, the maximum longitudinal electric field for bunches with measured parameters is reached 1 m upstream compared to the baseline parameters. The saturated amplitude at the end of the plasma section is, however, the same for the three models. As shown by Eq. 3.15, a smaller radial bunch size should generate wakefields with larger amplitudes, and this agrees well with the results of plasma modeling.

4.7 Conclusions

We have developed and applied an approach to analyzing the parameters of the proton bunch in the AWAKE experiment. In this approach, the data from multiple beam imaging systems that capture integrated radial bunch profiles were used to determine optimal parameters of the model that describe proton bunch propagation along the beamline. The fitting procedure was performed using a Bayesian approach, and the MCMC sampling was used to extract the posterior distributions.

Two models that describe the radial bunch density were considered. In the first model, the transverse bunch profile was represented as a Gaussian function, in the second model — as a mixture of two Gaussians denoted as halo and core. By definition, the halo component has a larger emittance compared to the core. These models have been tested using simulated events, and the results show that reconstruction of true parameters is possible with typical uncertainties of a few percent.

We have acquired a dataset with 671 proton bunch extractions, each characterized by varying proton bunch parameters, and applied the developed analysis scheme to this experimental data. It has been demonstrated that the double Gaussian model gives much better agreement with the experimental data compared to the single Gaussian model. The resulting posterior parameters for the double Gaussian model indicate the following:

- The transverse size of the proton bunch at the waist position is smaller than the baseline parameters for all bunch populations.
- Sizes of the halo and core components increase with the increasing bunch population.
- The transverse bunch profile is elliptical (the horizontal size is smaller than the vertical by $\approx 8\%$ and $\approx 40\%$ for the core and halo components, respectively).
- The contribution of the halo component increases with the bunch population.
- The bunch emittance is smaller than the nominal parameters.

A systematic drift of the bunch centroid was observed of approximately 50 µm during 6 hours of measurements. The jitter of the bunch at the waist position after drift correction is $\sigma(\mu_x) \approx 41\,\mu\text{m}$ and $\sigma(\mu_y) \approx 8\,\mu\text{m}$.

Plasma modeling has indicated that the bunches with measured parameters produce a 7% larger longitudinal wakefield amplitude than those with baseline parameters. The maximum wakefield is reached 1 m upstream for measured parameters compared to the baseline. During the self-modulation, the bunches with measured parameters show more significant focusing in the head of the bunch and more significant defocusing in the tail of the bunch compared to the baseline parameters.

Afterword

Although conclusions of individual projects are provided in Section 2.1.5, Section 2.2.4, and Section 4.7, there are a few general points that I would like to address as an afterword to this thesis.

The analysis of the proton bunch parameters presented in Chapter 4 was recently successfully used in the AWAKE control room during the early phase of the second data-taking period. Now, this analysis is a part of the standard procedures performed before each run, and it allows monitoring of the stability of proton bunch parameters over long operating times. To make the analysis faster and more suitable for quick monitoring during the experiment run-time, two-dimensional projections of the data were considered and the MCMC sampler was replaced by an optimizer that finds the posterior mode without sampling the full space. The detailed MCMC analysis, however, remains as an offline diagnostic that can provide precise parameter estimations with corresponding uncertainties when the run-time of the analysis is not strictly limited. I believe that this development can contribute to AWAKE successes in a long-term perspective.

The development of the algorithm for sampling with space partitioning presented in Section 2.2 was motivated by the need to have an approach to parallelize MCMC techniques efficiently and improve sampling accuracy of complex multimodal (but not only) densities. The first attempts of the analysis of the proton bunch parameters in AWAKE indicated that resulting posterior distributions are multimodal, so the sampling with space partitioning was intended to be applied to this problem. However, a detailed preprocessing of the experimental data noticeably simplified the analysis, making posteriors easy to sample with the standard Metropolis-Hastings algorithm. As a result, sampling with space partitioning was not used in the AWAKE analysis, and it was rather parallel development as a part of the BAT.jl package. The current implementation of this algorithm is very generic: It allows utilizing arbitrary samplers implemented in BAT.jl (e.g., Metropolis-Hastings, HMC, NS) together with arbitrary integrators (e.g., AHMI, CUBA, and soon the Bridge Sampler). In addition, the algorithm is implemented using functionalities of distributed computing. It can utilize high-performance computing resources by default, without a requirement to write additional, parallelizable code. I believe that this tool can be of help to those who need to solve problems with complex, multimodal posteriors and time-expensive likelihoods in physics or elsewhere.

Bibliography

[1] CERN storage: What data to record? Retrieved from URL: https://home.cern/science/computing/storage on 29.09.2021.

[2] Robert Braun, Tyler L Bourke, James A Green, Evan Keane, and Jeff Wagg. Advancing astrophysics with the square kilometre array. In *Advancing Astrophysics with the Square Kilometre Array*, volume 215, page 174. SISSA Medialab, 2015.

[3] SKA Organisation. The square kilometre array. page 19. Retrieved from URL: https://www.dropbox.com/s/n7ghen7sb8h2tog/2016_SKA_Prospectus.pdf on 1.09.2021.

[4] Ronald A Fisher. On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 222(594-604):309–368, 1922.

[5] Jerzy Neyman and Egon S Pearson. On the use and interpretation of certain test criteria for purposes of statistical inference: Part ii. *Biometrika*, 20(3/4):263–294, 1928.

[6] Thomas Bayes. Lii. an essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, frs communicated by mr. price, in a letter to john canton, amfr s. *Philosophical transactions of the Royal Society of London*, (53):370–418, 1763.

[7] Pierre Simon Laplace. Memoir on the probability of the causes of events. *Statistical science*, 1(3):364–378, 1986.

[8] Andrei Nikolaevich Kolmogorov and Albert T Bharucha-Reid. *Foundations of the theory of probability: Second English Edition.* Courier Dover Publications, 2018.

[9] Phil Gregory. *Bayesian Logical Data Analysis for the Physical Sciences: A Comparative Approach with Mathematica® Support.* Cambridge University Press, 2005.

[10] Iain Murray and Zoubin Ghahramani. A note on the evidence and bayesian occam's razor. 2005.

[11] David JC MacKay. *Information theory, inference and learning algorithms.* Cambridge university press, 2003.

[12] Nicholas Metropolis and Stanislaw Ulam. The monte carlo method. *Journal of the American statistical association*, 44(247):335–341, 1949.

[13] Paul Erd. On a new law of large numbers. *J. Anal. Math*, 23(103111):8, 1970.

[14] Luca Martino, David Luengo, and Joaquín Míguez. *Independent random sampling methods.* springer, 2018.

[15] Michael J Evans and Jeffrey S Rosenthal. *Probability and statistics: The science of uncertainty.* Macmillan, 2004.

[16] Christian Robert and George Casella. *Monte Carlo statistical methods.* Springer Science & Business Media, 2013.

[17] John Frank Charles Kingman. *Poisson processes*, volume 3. Clarendon Press, 1992.

[18] Yu A Kutoyants. *Statistical inference for spatial Poisson processes*, volume 134. Springer Science & Business Media, 2012.

[19] Bruce Cameron Reed. *The history and science of the Manhattan Project.* Springer, 2014.

[20] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer, 2003.

[21] Christopher K Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[22] Timo Teräsvirta. Specification, estimation, and evaluation of smooth transition autoregressive models. *Journal of the american Statistical association*, 89(425):208–218, 1994.

[23] James R Norris and John Robert Norris. *Markov chains.* Number 2. Cambridge university press, 1998.

[24] Daniel Revuz. *Markov chains.* Elsevier, 2008.

[25] Johanne Hizanidis. The master equation. *TU Berlin*, 2002.

[26] Maya R Said, Alan V Oppenheim, and Douglas A Lauffenburger. Modeling cellular signal processing using interacting markov chains. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03).*, volume 6, pages VI–41. IEEE, 2003.

[27] Mukhtar Ullah and Olaf Wolkenhauer. Family tree of markov models in systems biology. *IET systems biology*, 1(4):247–254, 2007.

[28] Andreas Willig. A short introduction to queueing theory. *Technical University Berlin, Telecommunication Networks Group*, 21, 1999.

[29] Leonard CG Rogers. Fluid models in queueing theory and wiener-hopf factorization of markov chains. *The Annals of Applied Probability*, 4(2):390–413, 1994.

[30] Andrew Gelman and Donald B Rubin. Inference from iterative simulation using multiple sequences. *Statistical science*, 7(4):457–472, 1992.

[31] Mary Kathryn Cowles and Bradley P Carlin. Markov chain monte carlo convergence diagnostics: a comparative review. *Journal of the American Statistical Association*, 91(434):883–904, 1996.

[32] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[33] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

[34] Av A Barker. Monte carlo calculations of the radial distribution functions for a proton? electron plasma. *Australian Journal of Physics*, 18(2):119–134, 1965.

[35] Andrew Gelman, Walter R Gilks, and Gareth O Roberts. Weak convergence and optimal scaling of random walk metropolis algorithms. *The annals of applied probability*, 7(1):110–120, 1997.

[36] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6):721–741, 1984.

[37] Simon Duane, A.D. Kennedy, Brian J. Pendleton, and Duncan Roweth. Hybrid monte carlo. *Physics Letters B*, 195(2):216 – 222, 1987.

[38] Radford M. Neal. *MCMC Using Hamiltonian Dynamics*, chapter chapter5. CRC Press, 2011.

[39] Michael Betancourt. A Conceptual Introduction to Hamiltonian Monte Carlo. *arXiv e-prints*, page arXiv:1701.02434, January 2017.

[40] John Skilling. Nested sampling for general bayesian computation. *Bayesian analysis*, 1(4):833–859, 2006.

[41] Johannes Buchner. Nested sampling methods. *arXiv preprint arXiv:2101.09675*, 2021.

[42] Christian P Robert, Víctor Elvira, Nick Tawn, and Changye Wu. Accelerating mcmc algorithms. *Wiley Interdisciplinary Reviews: Computational Statistics*, 10(5):e1435, 2018.

[43] Michael Betancourt. A conceptual introduction to hamiltonian monte carlo. *arXiv preprint arXiv: 1701.02434*, 2017.

[44] Matthew D Hoffman, Andrew Gelman, et al. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *J. Mach. Learn. Res.*, 15(1):1593–1623, 2014.

[45] Mark Fielding, David J Nott, and Shie-Yui Liong. Efficient mcmc schemes for computationally expensive posterior distributions. *Technometrics*, 53(1):16–28, 2011.

[46] Charles J Geyer. Markov chain monte carlo maximum likelihood. 1991.

[47] Enzo Marinari and Giorgio Parisi. Simulated tempering: a new monte carlo scheme. *EPL (Europhysics Letters)*, 19(6):451, 1992.

[48] Randal Douc, Arnaud Guillin, J-M Marin, Christian P Robert, et al. Convergence of adaptive mixtures of importance sampling schemes. *The Annals of Statistics*, 35(1):420–448, 2007.

[49] Jun S Liu, Faming Liang, and Wing Hung Wong. The multiple-try method and local optimization in metropolis sampling. *Journal of the American Statistical Association*, 95(449):121–134, 2000.

[50] Mylène Bédard, Randal Douc, and Eric Moulines. Scaling analysis of multiple-try mcmc methods. *Stochastic Processes and their Applications*, 122(3):758–786, 2012.

[51] Eric Laloy and Jasper A Vrugt. High-dimensional posterior exploration of hydrologic models using multiple-try dream (zs) and high-performance computing. *Water Resources Research*, 48(1), 2012.

[52] Radford M Neal. Sampling from multimodal distributions using tempered transitions. *Statistics and computing*, 6(4):353–366, 1996.

[53] Yun Xie, Jian Zhou, and Shaoyi Jiang. Parallel tempering monte carlo simulations of lysozyme orientation on charged surfaces. *The Journal of chemical physics*, 132(6):02B602, 2010.

[54] JN Carter and DA White. History matching on the imperial college fault model using parallel tempering. *Computational Geosciences*, 17(1):43–65, 2013.

[55] Arun Nampally and CR Ramakrishnan. Adaptive mcmc-based inference in probabilistic logic programs. *arXiv preprint arXiv:1403.6036*, 2014.

[56] Per Mykland, Luke Tierney, and Bin Yu. Regeneration in markov chain samplers. *Journal of the American Statistical Association*, 90(429):233–241, 1995.

[57] Willie Neiswanger, Chong Wang, and Eric Xing. Asymptotically exact, embarrassingly parallel mcmc. *arXiv preprint arXiv:1311.4780*, 2013.

[58] Steven L Scott, Alexander W Blocker, Fernando V Bonassi, Hugh A Chipman, Edward I George, and Robert E McCulloch. Bayes and big data: The consensus monte carlo algorithm. *International Journal of Management Science and Engineering Management*, 11(2):78–88, 2016.

[59] Xiangyu Wang and David B Dunson. Parallelizing mcmc via weierstrass sampler. *arXiv preprint arXiv:1312.4605*, 2013.

[60] Douglas N VanDerwerken and Scott C Schmidler. Parallel markov chain monte carlo. *arXiv preprint arXiv:1312.7479*, 2013.

[61] Jonas Hallgren and Timo Koski. Decomposition sampling applied to parallelization of metropolis-hastings. *arXiv preprint arXiv:1402.2828*, 2014.

[62] Guillaume Basse, Aaron Smith, and Natesh Pillai. Parallel markov chain monte carlo via spectral clustering. In *Artificial intelligence and statistics*, pages 1318–1327, 2016.

[63] Allen Caldwell, Philipp Eller, Vasyl Hafych, Rafael Schick, Oliver Schulz, and Marco Szalay. Integration with an adaptive harmonic mean algorithm. *International Journal of Modern Physics A*, 35(24):2050142, 2020.

[64] Vasyl Hafych, Philipp Eller, Oliver Schulz, and Allen Caldwell. Parallelizing mcmc sampling via space partitioning. *arXiv preprint arXiv:2008.03098*, 2020.

[65] Oliver Schulz, Frederik Beaujean, Allen Caldwell, Cornelius Grunwald, Vasyl Hafych, Kevin Kröninger, Salvatore La Cagnina, Lars Röhrig, and Lolian Shtembari. Bat. jl: A julia-based tool for bayesian inference. *SN Computer Science*, 2(3):1–17, 2021.

[66] Oliver Schulz. BAT.jl. GitHub repository: https://github.com/bat/BAT.jl on 19.10.2021.

[67] Nial Friel and Jason Wyse. Estimating the evidence–a review. *Statistica Neerlandica*, 66(3):288–308, 2012.

[68] Luke Tierney and Joseph B Kadane. Accurate approximations for posterior moments and marginal densities. *Journal of the american statistical association*, 81(393):82–86, 1986.

[69] Michael A Newton and Adrian E Raftery. Approximate bayesian inference with the weighted likelihood bootstrap. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56(1):3–26, 1994.

[70] Siddhartha Chib and Ivan Jeliazkov. Marginal likelihood from the metropolis–hastings output. *Journal of the American Statistical Association*, 96(453):270–281, 2001.

[71] Christian P Robert and Darren Wraith. Computational methods for bayesian model choice. In *Aip conference proceedings*, volume 1193, pages 251–262. American Institute of Physics, 2009.

[72] John Skilling. Nested sampling for general bayesian computation. *Bayesian analysis*, 1(4):833–859, 2006.

[73] Andrew Gelman and Xiao-Li Meng. Simulating normalizing constants: From importance sampling to bridge sampling to path sampling. *Statistical science*, pages 163–185, 1998.

[74] Nial Friel and Anthony N Pettitt. Marginal likelihood estimation via power posteriors. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(3):589–607, 2008.

[75] Neal Radford. The harmonic mean of the likelihood: Worst monte carlo method ever. Retrieved from URL: https://radfordneal.wordpress.com/2008/08/17/the-harmonic-mean-of-the-likelihood-worst-monte-carlo-method-eve on 19.10.2021.

[76] R Schick. Adaptive harmonic mean integration technique and its application to neutrino physics. *MPP-2018-435*. URL: https://publications.mppmu.mpg.de/?action=search&mpi=MPP-2018-435.

[77] Michael Schmelling. Averaging correlated data. *Physica Scripta*, 51(6):676, 1995.

[78] He Jia and Uros Seljak. Normalizing constant estimation with gaussianized bridge sampling. In *Symposium on Advances in Approximate Bayesian Inference*, pages 1–14. PMLR, 2020.

[79] Thiel Joris. *Adaptive Harmonic Mean Integration with Spherical Volumes*. Technical University of Munich (Germany), 2021.

[80] Daniel Bartz. *Advances in high-dimensional covariance matrix estimation*. Technische Universitaet Berlin (Germany), 2016.

[81] Charles J Geyer. Practical markov chain monte carlo. *Statistical science*, pages 473–483, 1992.

[82] Jerome Klotz. Asymptotic efficiency of the two sample kolmogorov-smirnov test. *Journal of the American Statistical Association*, 62(319):932–938, 1967.

[83] David Lopez-Paz and Maxime Oquab. Revisiting classifier two-sample tests. *arXiv preprint arXiv:1610.06545*, 2016.

[84] Bob Carpenter, Andrew Gelman, Matthew Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: A probabilistic programming language. *Journal of Statistical Software, Articles*, 76(1):1–32, 2017.

[85] Christopher Fonnesbeck John Salvatier, Thomas Wiecki. Probabilistic programming in python using pymc3.

[86] R Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2017.

[87] David J. Lunn, Andrew Thomas, Nicky Best, and David Spiegelhalter. Winbugs - a bayesian modelling framework: Concepts, structure, and extensibility. *Statistics and Computing*, 10(4):325–337, 2000.

[88] Allen Caldwell, Daniel Kollar, and Kevin Kröninger. BAT: The Bayesian Analysis Toolkit. *Comput. Phys. Commun.*, 180:2197–2209, 2009.

[89] A. J. Bevan et al. The UTfit collaboration average of D meson mixing data: Winter 2014. *JHEP*, 03:123, 2014.

[90] Diptimoy Ghosh, Matteo Salvarezza, and Fabrizio Senia. Extending the Analysis of Electroweak Precision Constraints in Composite Higgs Models. 2015.

[91] Jorge de Blas et al. Global Bayesian Analysis of the Higgs-boson Couplings. In *International Conference on High Energy Physics 2014 (ICHEP 2014) Valencia, Spain, July 2-9, 2014*, 2014.

[92] Matteo Agostini, Giovanni Benato, and Jason Detwiler. Discovery probability of next-generation neutrinoless double-β decay experiments. *Phys. Rev.*, D96(5):053001, 2017.

[93] Allen Caldwell, Alexander Merle, Oliver Schulz, and Maximilian Totzauer. Global Bayesian analysis of neutrino mass data. *Phys. Rev.*, D96(7):073001, 2017.

[94] Orlando Luongo, Giovanni Battista Pisani, and Antonio Troisi. Cosmological degeneracy versus cosmography: a cosmographic dark energy model. 2015.

[95] Piero Ullio and Mauro Valli. A critical reassessment of particle Dark Matter limits from dwarf satellites. 2016.

[96] Christophe Rappold et al. Hypernuclear production cross section in the reaction of ^{6}Li + ^{12}C at 2A GeV. *Phys. Lett.*, B747:129, 2015.

[97] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B. Shah. Julia: A fresh approach to numerical computing. *CoRR*, abs/1411.1607, 2014.

[98] Andrew Gelman and Donal B. Rubin. Inference from Iterative Simulation Using Multiple Sequences. *Statistical Science*, 07(04):503–511, 1992.

[99] Andrew Gelman and Donal B. Rubin. Inference from Iterative Simulation Using Multiple Sequences. *Statistical Science*, 07(04):503–511, 1992.

[100] John A. Nelder and Ronald Mead. A simplex method for function minimization. *Comput. J.*, 7:308–313, 1965.

[101] Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45(1-3):503–528, 1989.

[102] Patrick K. Mogensen and Asbjørn N. Riseth. Optim: A mathematical optimization package for julia. *Journal of Open Source Software*, 3(24):615, 2018.

[103] Thomas Hahn. Cuba - a library for multidimensional numerical integration. *Computer Physics Communications*, 168(2):78–95, 2005.

[104] Joseph John Thomson. Xxiv. on the structure of the atom: an investigation of the stability and periods of oscillation of a number of corpuscles arranged at equal intervals around the circumference of a circle; with application of the results to the theory of atomic structure. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 7(39):237–265, 1904.

[105] Mary K Gaillard, Paul D Grannis, and Frank J Sciulli. The standard model of particle physics. *Reviews of Modern Physics*, 71(2):S96, 1999.

[106] John Douglas Cockcroft and Ernest TS Walton. Experiments with high velocity positive ions. *Proceedings of the royal society of London. Series A, containing papers of a mathematical and physical character*, 129(811):477–489, 1930.

[107] G Ising. Arkiv för matematik. *Astronomi och Fysik*, 18(1), 1924.

[108] Ernest O Lawrence and M Stanley Livingston. The production of high speed protons without the use of high voltages. *Physical Review*, 38(4):834, 1931.

[109] Edwin M McMillan. The synchrotron—a proposed high energy particle accelerator. *Physical Review*, 68(5-6):143, 1945.

[110] Vladimir Isaakovich Veksler. A new method of accelerating of relativistic particles. *J. Physics USSR*, 9(3):153–168, 1945.

[111] PJ Bryant. A brief history and review of accelerators. *CERN European Organization for Nuclear Research-Reports-CERN*, pages 1–1, 1994.

[112] Roger Cashmore, Luciano Maiani, and Jean-Pierre Revol. *Prestigious Discoveries at CERN: 1973 Neutral Currents 1983 W & Z Bosons*. Springer Science & Business Media, 2013.

[113] G ai Baur, G Boero, A Brauksiepe, A Buzzo, W Eyrich, R Geyer, D Grzonka, J Hauffe, K Kilian, M LoVetere, et al. Production of antihydrogen. *Physics Letters B*, 368(3):251–258, 1996.

[114] New state of matter created at CERN? Retrieved from URL: https://home.cern/news/press-release/cern/new-state-matter-created-cern on 29.09.2021.

[115] Viviana Fanti, A Lai, D Marras, L Musa, AJ Bevan, TJ Gershon, B Hay, RW Moore, KN Moore, DJ Munday, et al. A new measurement of direct cp violation in two pion decays of the neutral kaon. *Physics Letters B*, 465(1-4):335–348, 1999.

[116] Serguei Chatrchyan, Vardan Khachatryan, Albert M Sirunyan, Armen Tumasyan, Wolfgang Adam, Ernest Aguilo, Thomas Bergauer, M Dragicevic, J Erö, C Fabjan, et al. Observation of a new boson at a mass of 125 gev with the cms experiment at the lhc. *Physics Letters B*, 716(1):30–61, 2012.

[117] Carlo Rovelli. Loop quantum gravity. *Living reviews in relativity*, 11(1):1–69, 2008.

[118] Katrin Becker, Melanie Becker, and John H Schwarz. *String theory and M-theory: A modern introduction.* Cambridge university press, 2006.

[119] Csaba Csáki and Philip Tanedo. Beyond the standard model. *arXiv preprint arXiv:1602.04228*, 2016.

[120] Gia Dvali. Strong coupling and classicalization. In *The future of our physics including new frontiers: Proceedings of the International School of Subnuclear Physics*, pages 189–200. World Scientific, 2017.

[121] Gia Dvali. Cosmological relaxation of higgs mass before and after lhc and naturalness. *arXiv preprint arXiv:1908.05984*, 2019.

[122] Stefano Profumo, Leonardo Giani, and Oliver F Piattella. An introduction to particle dark matter. *Universe*, 5(10):213, 2019.

[123] Steven Weinberg. *Cosmology.* Oxford university press, 2008.

[124] Rabindra N Mohapatra and Goran Senjanović. Neutrino mass and spontaneous parity nonconservation. *Physical Review Letters*, 44(14):912, 1980.

[125] Murray Gell-Mann, Pierre Ramond, and Richard Slansky. Complex spinors and unified theories. In *Murray Gell-Mann: Selected Papers*, pages 266–272. World Scientific, 2010.

[126] Guido Fantini, Andrea Gallo Rosso, Francesco Vissani, and Vanessa Zema. The formalism of neutrino oscillations: an introduction. *arXiv preprint arXiv:1802.05781*, 2018.

[127] BJ Holzer. Introduction to particle accelerators and their limitations. *arXiv preprint arXiv:1705.09601*, 2017.

[128] Helmut Wiedemann. *Particle accelerator physics.* Springer Nature, 2015.

[129] Corinne Pralavorio. Fresca2: a record field for an extraordinary magnet. 2018.

[130] Glyn Kirby. Next generation materials for future magnet development at CERN. In *IOP Conference Series: Materials Science and Engineering*, volume 502, page 012002. IOP Publishing, 2019.

[131] Lucie Linssen, Akiya Miyamoto, Marcel Stanitzki, and Harry Weerts. Physics and detectors at clic: Clic conceptual design report. *arXiv preprint arXiv:1202.5940*, 2012.

[132] Philip Bambade, Tim Barklow, Ties Behnke, Mikael Berggren, James Brau, Philip Burrows, Dmitri Denisov, Angeles Faus-Golfe, Brian Foster, Keisuke Fujii, et al. The international linear collider: a global project. *arXiv preprint arXiv:1903.01629*, 2019.

[133] Frank Zimmermann, M Benedikt, M Capeans Garrido, F Cerutti, B Goddard, J Gutleber, JM Jimenez, M Mangano, V Mertens, JA Osborne, et al. Future circular collider. Technical report, CERN-ACC-2018-0059, 2018.

[134] Simon Martin Hooker. Developments in laser-driven plasma accelerators. *Nature Photonics*, 7(10):775–782, 2013.

[135] E Gschwendtner. AWAKE, a particle-driven plasma wakefield acceleration experiment. *arXiv preprint arXiv:1705.10573*, 2017.

[136] Chunguang Jing. Dielectric wakefield accelerators. *Reviews of Accelerator Science and Technology*, 9:127–149, 2016.

[137] R Joel England, Robert J Noble, Karl Bane, David H Dowell, Cho-Kuen Ng, James E Spencer, Sami Tantawi, Ziran Wu, Robert L Byer, Edgar Peralta, et al. Dielectric laser accelerators. *Reviews of Modern Physics*, 86(4):1337, 2014.

[138] Brigitte Cros and Patric Muggli. Towards a proposal for an advanced linear collider report on the advanced and novel accelerators for high energy physics roadmap workshop, CERN, geneva, april 2017. 2017.

[139] Francis F Chen. *Introduction to plasma physics*. Springer Science & Business Media, 2012.

[140] S Wilks, JM Dawson, TC Katsouleas, and JJ Su. Beam loading efficiency in plasma accelerators. *Part. Accel.*, 22:81–99, 1987.

[141] Toshiki Tajima and John M Dawson. Laser electron accelerator. *Physical Review Letters*, 43(4):267, 1979.

[142] Donna Strickland and Gerard Mourou. Compression of amplified chirped optical pulses. *Optics communications*, 55(6):447–449, 1985.

[143] Jérôme Faure, Yannick Glinec, A Pukhov, S Kiselev, S Gordienko, E Lefebvre, J-P Rousseau, F Burgy, and Victor Malka. A laser–plasma accelerator producing monoenergetic electron beams. *Nature*, 431(7008):541–544, 2004.

[144] WP Leemans, AJ Gonsalves, H-S Mao, K Nakamura, C Benedetti, CB Schroeder, Cs Tóth, J Daniels, DE Mittelberger, SS Bulanov, et al. Multi-gev electron beams from capillary-discharge-guided subpetawatt laser pulses in the self-trapping regime. *Physical review letters*, 113(24):245002, 2014.

[145] Pisin Chen, JM Dawson, Robert W Huff, and Thomas Katsouleas. Acceleration of electrons by the interaction of a bunched electron beam with a plasma. *Physical review letters*, 54(7):693, 1985.

[146] James Benjamine Rosenzweig, DB Cline, B Cole, H Figueroa, W Gai, R Konecny, J Norem, P Schoessow, and J Simpson. Experimental observation of plasma wakefield acceleration. *Physical review letters*, 61(1):98, 1988.

[147] Ian Blumenfeld, Christopher E Clayton, Franz-Josef Decker, Mark J Hogan, Chengkun Huang, Rasmus Ischebeck, Richard Iverson, Chandrashekhar Joshi, Thomas Katsouleas, Neil Kirby, et al. Energy doubling of 42 gev electrons in a metre-scale plasma wakefield accelerator. *Nature*, 445(7129):741–744, 2007.

[148] Wim Leemans and Eric Esarey. Laser-driven plasma-wave electron accelerators. *Phys. Today*, 62(3):44–49, 2009.

[149] Swapan Chattopadhyay and Roger M Jones. Radiative regime of linear colliders, high repetetion rate free electron lasers and associated accelerating structures. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 657(1):168–176, 2011.

[150] A Caldwell, E Gschwendtner, K Lotov, P Muggli, and M Wing. AWAKE design report: a proton-driven plasma wakefield acceleration experiment at CERN. Technical report, 2013.

[151] Mobs Esma. The CERN accelerator complex - 2019. Retrieved from URL: `https://cds.cern.ch/record/2684277` on 29.09.2021.

[152] Erik Adli, *et al.* (AWAKE Collaboration). Acceleration of electrons in the plasma wakefield of a proton bunch. *Nature*, 561(7723):363–367, 2018.

[153] E Öz and P Muggli. A novel rb vapor plasma source for plasma wakefield accelerators. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 740:197–202, 2014.

[154] Gennady Plyushchev, Roberto Kersevan, Alexey Petrenko, and Patric Muggli. A rubidium vapor source for a plasma source for AWAKE. *Journal of Physics D: Applied Physics*, 51(2):025203, 2017.

[155] Edda Gschwendtner, *et al.* (AWAKE Collaboration). AWAKE, the advanced proton driven plasma wakefield acceleration experiment at CERN. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 829:76–82, 2016.

[156] Valentin Fedosseev, Patric Muggli, Eric Chevallay, Alessandro Masi, Jan Hansen, Christoph Heßler, Krzysztof Szczurek, Edda Gschwendtner, Nicolas Chritin, Mikhail Martyanov, et al. Integration of a terawatt laser at the CERN sps beam for the AWAKE experiment on proton-driven plasma wake acceleration. Technical report, 2016. cern-acc-2016-209.

[157] Alexey Petrenko, Konstantin Lotov, and Alexander Sosedkin. Numerical studies of electron acceleration behind self-modulating proton beam in plasma with a density gradient. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 829:63–66, 2016.

[158] Michael Turner, *et al.* (AWAKE Collaboration). Experimental observation of plasma wakefield growth driven by the seeded self-modulation of a proton bunch. *Physical review letters*, 122(5):054801, 2019.

[159] Erik Adli, *et al.* (AWAKE Collaboration). Experimental observation of proton bunch modulation in a plasma at varying plasma densities. *Physical review letters*, 122(5):054802, 2019.

[160] Marlene Turner, *et al.* (AWAKE Collaboration). Experimental study of wakefields driven by a self-modulating proton bunch in plasma. *Physical Review Accelerators and Beams*, 23(8):081302, 2020.

[161] OE Berrig, F Caspers, C Vollinger, C Vuitton, J Kuczerowski, R Sautier, M Hamani, C Zannini, J Emery, B Dehning, et al. CERN-sps wire scanner impedance and wire heating studies. Technical report, 2014.

[162] G Baud, E Piselli, JJ Gras, A Guerrero, J Emery, and B Dehning. Performance assessment of wire-scanners at CERN. Technical report, 2013.

[163] G Papotti. A beam quality monitor for lhc beams in the sps. Technical report, 2008.

[164] G Papotti, T Bohl, F Follin, E Shaposhnikova, et al. The sps beam quality monitor, from design to operation. *Momentum*, 200(300):400, 2011.

[165] A Monera, M Andersen, D Belohrad, L Jensen, L Søby, and G Kasprowicz. Upgrade of the CERN psb/cps fast intensity measurements. In *DIPAC*, volume 11, page 185, 2011.

[166] S Burger, B Biskup, S Mazzoni, M Turner, et al. Scintillation and otr screen characterization with a 440 gev/c proton beam in air at the CERN hiradmat facility. *Proceedings of IBIC*, 2016, 2016.

[167] F Braunmueller, M Martyanov, S Alberti, and P Muggli. Novel diagnostic for precise measurement of the modulation frequency of seeded self-modulation via coherent transition radiation in AWAKE. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 909:76–79, 2018.

[168] Allen Caldwell and KV Lotov. Plasma wakefield acceleration with a modulated proton bunch. *Physics of Plasmas*, 18(10):103101, 2011.

[169] Alexander Gorn, *et al.* (AWAKE Collaboration). Proton beam defocusing in awake: comparison of simulations and measurements. *Plasma Physics and Controlled Fusion*, 62(12):125023, 2020.

[170] Vasyl Hafych, *et al.* (AWAKE Collaboration). Analysis of proton bunch parameters in the AWAKE experiment. *arXiv preprint arXiv: 2109.12893*, 2021.

[171] H Timko, T Argyropoulos, H Bartosik, T Bohl, J Esteban Müller, E Shaposhnikova, et al. Short high-intensity bunches for plasma wakefield experiment AWAKE in the CERN sps. *Proc. IPAC2013 (Shanghai, China)*, page 1820, 2013.

[172] Chiara Bracco, Edda Gschwendtner, Alexey Petrenko, Helga Timko, Theodoros Argyropoulos, Hannes Bartosik, Thomas Bohl, Juan Esteban Müller, Brennan Goddard, Malika Meddahi, et al. Beam studies and experimental facility for the AWAKE experiment at CERN. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 740:48–53, 2014.

[173] John Farmer. 132 AWAKE MPP group meeting. Presented at: `https://indico.cern.ch/event/1019873/` on 18.03.21.

[174] John Farmer. 133 AWAKE MPP group meeting. Presented at: `https://indico.cern.ch/event/1022280/` on 25.03.21.

[175] Alexander Pukhov. Particle-in-cell codes for plasma-based particle acceleration. *arXiv preprint arXiv:1510.01071*, 2015.

[176] Alexander Pukhov. Three-dimensional electromagnetic relativistic particle-in-cell code vlpl (virtual laser plasma lab). *Journal of plasma physics*, 61(3):425–433, 1999.

[177] T Zh Esirkepov. Exact charge conservation scheme for particle-in-cell simulation with an arbitrary form-factor. *Computer Physics Communications*, 135(2):144–153, 2001.

[178] John M Dawson. Particle simulation of plasmas. *Reviews of modern physics*, 55(2):403, 1983.

[179] Roger W Hockney and James W Eastwood. *Computer simulation using particles*. crc Press, 1988.

[180] Alexander Pukhov and John P Farmer. Stable particle acceleration in coaxial plasma channels. *Physical review letters*, 121(26):264801, 2018.

[181] Patrick Mora and Thomas M Antonsen, Jr. Kinetic modeling of intense, short laser pulses propagating in tenuous plasmas. *Physics of Plasmas*, 4(1):217–229, 1997.

[182] Eric Esarey, CB Schroeder, and WP Leemans. Physics of laser-driven plasma-based electron accelerators. *Reviews of modern physics*, 81(3):1229, 2009.