technische universität
dortmund

Arbeit zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften

Development of a tool for Bayesian data analysis and its application in Monte Carlo tuning

Salvatore La Cagnina
geboren in Hemer

2023

Arbeitsgruppe Kröninger
Fakultät Physik
Technische Universität Dortmund

Diese Dissertation wurde der Fakultät Physik der Technischen Universität Dortmund zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften vorgelegt.

Erstgutachter:	Prof. Dr. Kevin Kröninger
Zweitgutachter:	PD Dr. Stefan Kluth
Vorsitzender der Prüfungskommission:	Prof. Dr. Götz Uhrig
Vertreter der wiss. Mitarbeiter:	Dr. Jörg Debus

Datum des Einreichens der Arbeit:	1. Dezember 2023
Datum der mündlichen Prüfung:	20. Februar 2024

Abstract

In this thesis, a novel approach to Monte Carlo event generator tuning, grounded in Bayesian reasoning, is presented. The Bayesian Analysis Toolkit (BAT.jl) is introduced as a modern tool for performing Bayesian inference. A numerical test suite that verifies the validity and performance of the BAT.jl package is developed. The test suite is used to evaluate the performance of the Markov chain Monte Carlo (MCMC) sampling algorithms implemented in BAT.jl, utilizing a selection of test functions and different metrics to quantify the quality of the samples. The results show that the MCMC algorithms are able to sample the posterior distributions of the test functions accurately. Utilizing the BAT.jl toolkit, two hadronization models within the Herwig Monte Carlo event generator (MCEG) are successfully tuned to data from the LEP experiments. Several aspects of the tuning procedure are investigated, such as parameter and observable selection and parametrization quality. Samples generated using the tuned parameters, obtained from the global mode of the posterior, are compared to data through a χ^2 test. The resulting p-values for the tuned simulations significantly outperform those from the nominal MCEG samples, indicating a successful tune and an improved description of the data. The posterior is also used to present a method for propagating the parameter uncertainties to the realm of the observables, providing a measure for the tuning uncertainty. Studies on the impact of assigning weights to the observables and the impact of correlations between measurements on the tuning are also presented. These show that weights can alter the tuning results, especially in cases with multiple modes in the posterior. However, their influence on the quality of the tune is minimal in this case. The correlation of measurements has less of an impact on the position of the global mode but substantially affects the associated parameter uncertainties estimates. Finally, a comparison of the two tuned hadronization models is presented, which indicates that the Lund string model describes the data slightly better than the cluster hadronization model for this set of observables.

Kurzfassung

In dieser Arbeit wird ein neuartiges Konzept für das Monte-Carlo-Ereignisgenerator-Tuning auf Grundlage des Bayes'schen Ansatzes präsentiert. Das Bayesian Analysis Toolkit (BAT.jl) wird als modernes Werkzeug zur Durchführung von Bayes'schen Inferenzen vorgestellt. Das Toolkit wird verwendet, um eine numerische Testsuite zu entwickeln, mit der die Gültigkeit und Leistung des Toolkits überprüft wird. Die Testsuite basiert auf ausgewählten Testfunktionen und verschiedenen Metriken zur Quantifizierung der Qualität der Stichproben und wird verwendet, um die Funktionsweise der in BAT.jl implementierten Markov-Chain-Monte-Carlo-(MCMC)-Verfahren zu überprüfen. Die Ergebnisse zeigen, dass die MCMC-Algorithmen in der Lage sind, die a-posteriori-Verteilungen der Testfunktionen genau abzubilden. Unter Verwendung des BAT.jl-Toolkits werden zwei Hadronisierungsmodelle mit dem Herwig Monte-Carlo-Ereignisgenerator erfolgreich an Daten der LEP-Experimente angepasst (Tuning). Verschiedene Aspekte des Tuning-Verfahrens, wie die Auswahl von Parametern und Observablen und die Qualität der Parametrisierung, werden untersucht. Die mit den angepassten Parametern erzeugten Monte-Carlo-Simulationen, die aus der globalen Mode der a-posteriori-Verteilung erhalten werden, werden mittels eines χ^2-Test mit Daten verglichen. Die resultierenden p-Werte der angepassten Simulationen übertreffen deutlich die der nominalen Simulationen, was auf eine erfolgreiche Anpassung und eine verbesserte Beschreibung der Daten hindeutet. Die a-posteriori-Verteilung wird zusätzlich verwendet, um eine Methode zur Fortpflanzung der Parameterunsicherheiten in den Raum der Observablen vorzustellen, welche als Maß für die Unsicherheit des Tunings dient. Studien über die Auswirkungen der Gewichtung von Observablen und den Einfluss von Korrelationen zwischen Messungen auf das Tuning werden ebenfalls präsentiert. Diese zeigen, dass die Gewichte die Ergebnisse des Tunings ändern können, insbesondere bei Verteilungen mit mehreren Moden, jedoch ist ihr Einfluss auf die Qualität des Tunes in diesem Fall minimal. Die Korrelation der Messungen hat einen geringeren Einfluss auf die Position der globalen Mode, beeinflusst jedoch die zugehörigen Parameterunsicherheitsschätzungen erheblich. Schließlich wird ein Vergleich der beiden angepassten Hadronisierungsmodelle vorgestellt, aus dem hervorgeht, dass das Lund-String-Modell die Daten für die gewählten Observablen etwas besser beschreibt als das Cluster-Hadronisierungsmodell.

Contents

List of contributions

This thesis presents the development of a tool for Bayesian inference and its application in the tuning of Monte Carlo event generators. The development of both the BAT.jl tool and the tuning procedure was performed in collaboration with other researchers and has been published as research articles. As part of this thesis, I have contributed to this research as outlined below.

- 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 COMPUT. SCI. 2 210 (2021) 210

  This article presents the development of the BAT.jl tool. My main contributions to this work were regarding the development of the numerical test suite. I was involved in the design decisions, including the selection of the test functions and metrics. I was also responsible for the implementation of the test suite and the evaluation of the results. This included the development of automated scripts for the execution of the tests in arbitrary dimensions, calculating the metrics, and the generation of the plots. The numerical test suite is incorporated into the package and is executed for every major version release to ensure the correctness of the tool. The details and results of the numerical test suite are presented in Chapter 6.

- S. La Cagnina, K. Kröninger, S. Kluth, A. Verbytskyi, *A Bayesian tune of the Herwig Monte Carlo event generator*, JINST 18 (2023) P10033

  In this article, the tuning of the `Herwig` event generator using Bayesian inference is presented. This study is an application of the BAT.jl tool and is the main outcome of this thesis. I worked on all aspects of this study, including the design of the tuning procedure, the implementation of the workflow, the execution of the tuning, and the evaluation of the results. This includes the setup of external tools such as the `Herwig` event generator and `Rivet` for the Monte Carlo generation and the implementation of the model in the EFTfitter.jl framework. The parametrization and its evaluation using statistical tests, the comparisons to data, the error propagation, as well as the studies on observable weights and measurement correlations were performed by myself. The details of the tuning procedure and the results are presented in Chapter 7. All plots and results presented in this chapter were produced by myself.

1 Introduction

The fundamental objective of scientific research is to deepen and expand our knowledge of the complex world around us. To achieve this goal, science relies on two core elements. First, theoretical models are developed to encapsulate existing knowledge and to propose new hypotheses. These models offer testable predictions that provide a way for empirical validation. Second, experiments are conducted to gather data, which is then compared with the prediction from these models. This comparison between theoretical predictions and experimental data is a crucial step and is made possible through the use of statistical methods. Through statistical inference, meaningful conclusions about both the model's parameters and its overall compatibility with the experimental data can be drawn. One approach for this type of statistical inference is the application of Bayesian statistics. In this approach, Bayes' theorem is employed to update the prior knowledge about the model's parameters using experimental data. As a result, this knowledge update directly provides the probability distributions of the model's parameters. Although interpreting these posterior distributions is straightforward, their computation in complex real-world scenarios demands specialized numerical techniques. To tackle this, the Bayesian Analysis Toolkit (BAT.jl) [1] has been developed, which provides a collection of tools and a user-friendly interface for the application of Bayesian statistics. This thesis will introduce the toolkit in detail, as well as the development of a numerical test suite designed to validate the functionality of the package.

A field of research that extensively utilizes computational methods is the field of particle physics. In this domain, the Standard Model of particle physics (SM) has been established as the leading theoretical model to describe fundamental particles and their interactions. However, the complexity and high-dimensional nature of the calculations involved necessitates the use of Monte Carlo event generators (MCEG) to obtain predictions based on the SM. While parton interactions — those involving quarks and gluons — can be calculated from first principles, MCEGs employ phenomenological models to account for the non-perturbative nature of the hadronization process and the evolution of the parton shower at low energy scales. These models introduce a set of free parameters that need to be fine-tuned to best describe the experimental data. This process of aligning model parameters with data is known as Monte Carlo tuning. The current state-of-the-art in Monte Carlo tuning is the Professor procedure [2], which systematically evaluates multiple event samples with varying parameter settings to refine the

model's alignment with experimental data. It parametrizes the MCEG response and employs minimization techniques to find the optimal parameter values that best align the model with experimental data. The Professor procedure has been effectively used in various applications to fine-tune MCEGs, resulting in improved alignment between theoretical predictions and experimental results [3–10]. Nonetheless, the Professor framework has some limitations that have motivated the development of tools like Apprentice [11], which aim to address some of these shortcomings. Additionally, alternative approaches to Monte Carlo tuning have been explored, such as Bayesian optimization [12] and the use of machine learning algorithms [13, 14]. In the scope of this thesis, a novel Monte Carlo tuning methodology grounded in Bayesian reasoning is presented. This procedure is implemented in the EFTfitter.jl package [15], a specialized extension built upon BAT.jl. By applying numerical algorithms like Markov chain Monte Carlo, the full posterior distribution of the MCEG parameters is obtained. This posterior serves multiple purposes: it identifies optimal parameter values while offering a comprehensive overview of the parameter space, potentially revealing limitations in the model. Furthermore, this procedure enables precise estimation of both uncertainties and correlations among model parameters and facilitates the propagation of these uncertainties to the model's predictions. In this approach, the tune uses data from LEP experiments [16–18], examining two different hadronization models implemented in the `Herwig7` [19] MCEG. To investigate the impact of different factors on the tuning process, the tune is repeated under varying conditions: first, by incorporating different correlations between the experimental data, and second, by applying alternative weighting schemes to the observables.

This thesis is structured as follows: In Chapter 2, the Standard Model of particle physics is introduced, and the role of QCD in collider experiments is discussed. An overview of the Monte Carlo event generation process with a focus on hadronization models is provided in Chapter 3. In Chapter 4, foundational concepts of Bayesian statistics are explored alongside numerical techniques like Markov Chain Monte Carlo and the Metropolis-Hastings algorithm. The BAT.jl framework is presented in Chapter 5, including descriptions of the implemented numerical algorithms. Chapter 6 covers the development of a numerical test suite for BAT.jl. There, the selection of test functions and test metrics is discussed, and the results of the test suite are presented. In Chapter 7, the development of a novel Monte Carlo tuning procedure based on Bayesian statistics is presented. The implementation of this approach in the EFTfitter.jl package is discussed, and both the parameterization process and the tuning results are examined. The influences of different correlation and weighting schemes on the tune are investigated, and the impact of error propagation of the tune on the predictions of the model is discussed. Furthermore, a comparison between two hadronization models is presented. Finally, Chapter 8 summarizes the results of this thesis and provides an outlook on future work.

2 The Standard Model of elementary particle physics

The goal of particle physics is to describe the properties and interactions of elementary particles. Throughout the history of particle physics, this description has undergone multiple iterations, becoming increasingly sophisticated over time. As of now, the Standard Model of particle physics (SM) represents the state-of-the-art in theoretical models [20–24]. It is a relativistic quantum field theory describing the fundamental particles and their interactions. The mathematical description of the Standard Model follows a compound $\mathrm{SU(3)_C} \times \mathrm{SU(2)_L} \times \mathrm{U(1)_Y}$ symmetry. A visual representation of the particles described by the Standard Model can be seen in Figure 2.1. These particles can generally be categorized based on properties like spin, charge, or mass. When sorted by spin, they fall into three distinct groups. Particles with spin ½ are called fermions, those with spin 1 are known as vector bosons, and those with spin 0 are termed scalar bosons.

2.1 Fermions

Fermions, with half-integer spin, are often referred to as 'matter' particles since they constitute the matter around us. They can be categorized into two groups: leptons and quarks. Leptons can be paired into tuples consisting of a charged lepton and a corresponding neutrino. Charged leptons, which include the electron (e), muon (μ), and tau (τ) leptons, carry an electric charge of $-1e$. In contrast, their corresponding neutrinos — electron neutrino (ν_e), muon neutrino (ν_μ), and tau neutrino (ν_τ) — are electrically neutral.

Quarks are organized in a manner similar to leptons, forming tuples that consist of an up-type and a down-type quark. Up-type quarks have an electric charge of $+2/3e$, while down-type quarks have a charge of $-1/3e$. Like the leptons, quarks also come in three distinct tuples: the up-quark (u) and down-quark (d), the charm-quark (c) and strange-quark (s), as well as the top-quark (t) and bottom-quark (b). Unlike leptons, all quarks possess a color charge, leading to additional interactions.

Finally, all fermions in the Standard Model have corresponding anti-fermions, which are identical in all quantum numbers except for charge.

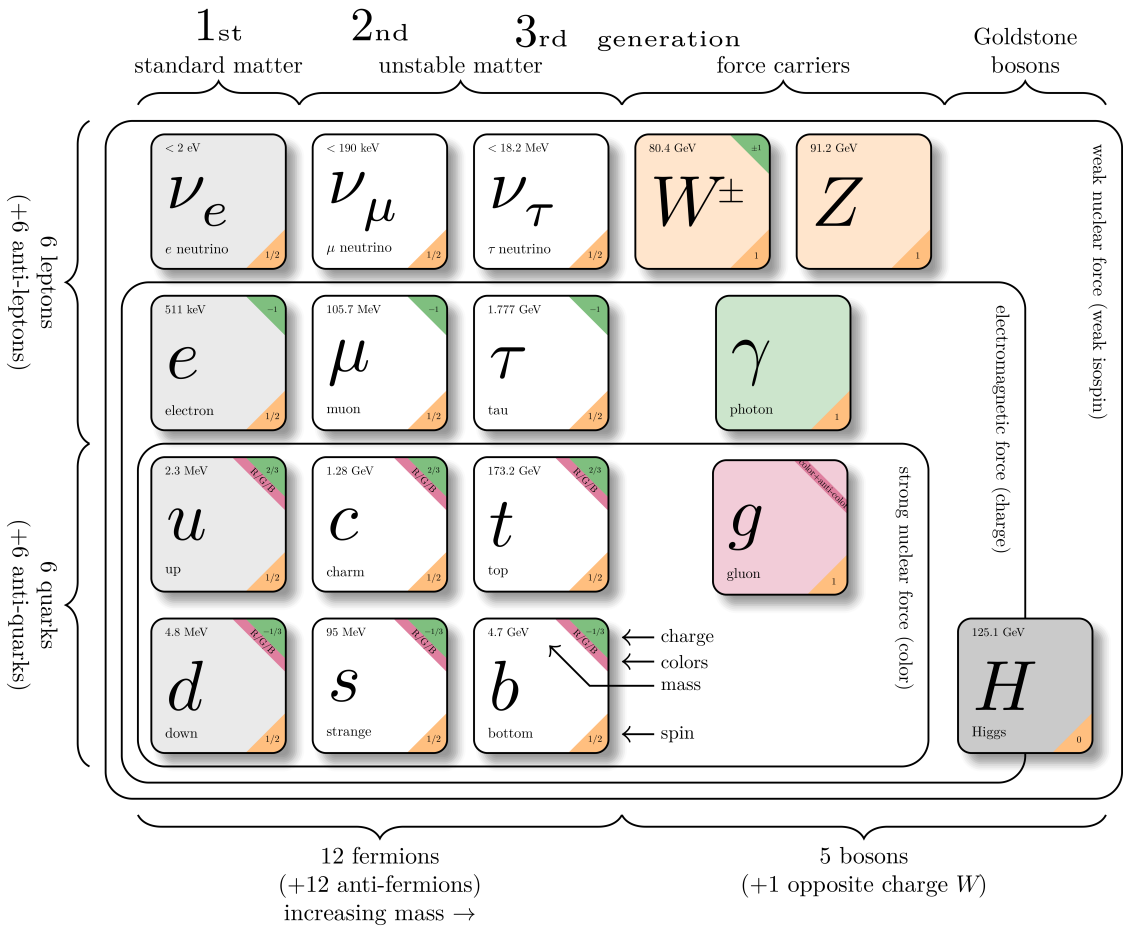


Figure 2.1: Scematic representation of the Standard Model and their interactions. The picture is based on [25] while the values of the particle properties are taken from [26].

2.2 Bosons and fundamental forces

Vector bosons, particles with a spin of 1, serve as the gauge bosons for the three fundamental forces described by the Standard Model, with the photon being the most prominent as it mediates the electromagnetic force. Each fermion and boson with an electric charge can interact through this force, a behavior described by quantum electrodynamics (QED).

The charged $W^\pm$ bosons and the electrically neutral Z^0 boson serve as the gauge bosons for the weak interaction and are unique among gauge bosons for having non-zero mass. The associated charge for this interaction is the weak isospin carried by all fermions in the Standard Model.

The last force accounted for in the Standard Model is the strong interaction governed by quantum chromodynamics (QCD). The gauge boson mediating this force is the gluon, and the charge associated with QCD is termed color charge. In contrast to QED, this quantum number is carried not only by the affected fermions (i.e., the quarks) but also by the mediating vector boson, the gluon. This attribute stems from the non-abelian symmetry of the associated $SU(3)_C$ symmetry group. As a result, gluons are capable of interacting with themselves. An additional property of QCD is the so-called confinement. While particles with non-zero electric charge and weak isospin have been observed as free particles, the same does not hold for color charge. Particles possessing color charge, like quarks and gluons, immediately form bonds with other color-charged particles to create color-neutral composite particles. These composite particles, known as hadrons, can be observed freely, whereas quarks and gluons cannot. The process by which quarks and gluons form these color-neutral hadrons is called hadronization. Finding a precise description of the confinement phenomenon remains one of the major unresolved questions in particle physics. Understanding this confinement is critical for experiments like the LHC, where quarks are generated and hadronization is inevitable. Given the significance of QCD, further details will be discussed in Section 2.3.

Finally, the Standard Model includes one more boson: the Higgs boson. Unlike other bosons, the Higgs is a scalar boson with spin 0. The Higgs boson was introduced to the Standard Model through the Higgs mechanism [27, 28]. This mechanism adds a new field, the Higgs field, which has a non-zero vacuum expectation value. This leads to the spontaneous symmetry breaking of the $SU(2)_L \times U(1)_Y$ group. As a result, the massless electroweak bosons mix to form the observed massive vector bosons of the weak force and the photon. In addition to the massive vector bosons, fermions also acquire mass through coupling with the Higgs field.

2.3 Quantum Chromo Dynamics in collider experiments

In current high-energy collider experiments, such as the Large Hadron Collider (LHC) at CERN [29], QCD plays a crucial role. The way protons collide is intrinsically linked to how they are described by QCD. Thus, understanding the initial state of the collision relies on factors like the proton's parton distribution function. On the other hand, QCD is also important for the description after the collision, as processes like particle showering and hadronization of the final state products and proton remnants need to be accurately modeled. The description of QCD in collider experiments is complicated by the running of the strong coupling constant α_s. While the coupling constant is small at high energy scales, resulting in asymptotically

free quarks and gluons, it increases with decreasing energy scale, neccesiating a non-perturbative description of the low energy regime.

In this thesis, data collected from experiments carried out at the Large Electron-Positron Collider (LEP) are utilized [16–18]. Since electrons have no substructure, the initial state is independent of QCD effects, which will hence only impact the final-state shower and hadronization processes. This is beneficial as the main focus of Monte Carlo tuning is the optimization of these shower and hadronization models. In general, the influence of QCD can be divided into three parts based on the energy scale of the processes.

This division is possible due to the QCD factorization theorem. It enables the separation of the hard scattering process from the hadronization step, as QCD operates on vastly different time scales in these two domains. Between these two domains lies an evolutionary step where additional partons are generated. These partons are part of both initial and final state parton showers and also contribute to the hadronization process. An overview of the steps involved in Monte Carlo event generation can be seen in Figure 2.2. The procedure of Monte Carlo event generation will be discussed in more detail in Section 3. More details about the foundation of QCD and factorization can be found in Ref. [30].

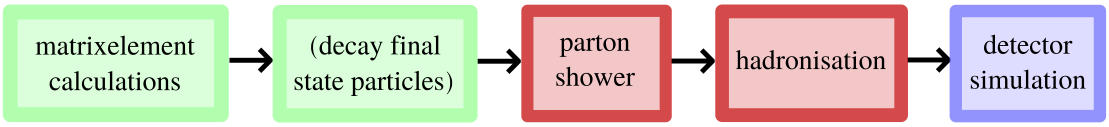


Figure 2.2: An overview of the steps involved in Monte Carlo event generation.

3 Monte Carlo event generation in particle physics

This chapter will discuss the event generation process in particle physics. The aim of Monte Carlo event generation is to accurately represent the production rate and the kinematic distribution of particles, thereby offering theoretical predictions for experiments. As briefly discussed in Section 2.3, Monte Carlo generation in high-energy physics can be separated into three steps. These are the hard process, the parton shower, and the hadronization. First, the evaluation of the hard process and the showering of quarks and gluons will be briefly discussed. The main focus will be on the different hadronization models, which are particularly relevant as tunable models in this thesis. These models are the default hadronization models for `Herwig7` [19] and `Pythia8` [31].The parameters that will be tuned in the subsequent chapters will be introduced and briefly described during this discussion. Finally, a brief introduction to `Rivet` [32] will be provided, as it supplies the data and analysis code for the tuning, followed by a discussion of the `ThePEG` [33] package.

3.1 Hard process

The hard process involves the highest momentum transfers in particle collisions, creating heavy particles and high-transverse-momentum hadrons, typically observed as collimated sprays of particles, referred to as jets. Therefore, accurately describing this process is crucial for creating Monte Carlo simulations of collider experiments. Due to the large momentum transfers, the QCD particles become asymptotically free, allowing their interactions to be described using perturbation theory. As a result, the hard scattering process can be represented using Feynman diagrams, for instance. There are several ways to implement the calculation of the matrix element. Pre-computed matrix elements are useful for simple processes involving only a few final-state particles. Alternatively, automated processes to calculate matrix elements can be used. A more in-depth look at how `Herwig7` implements the hard process generation will be provided in 3.4. Other techniques that are commonly used, as well as more details, are provided in Ref. [34].

3.2 Parton Shower

The hard process, outlined in Section 3.1, allows for an adequate description of outgoing jet momenta using fixed order calculations as the high-momentum final state particle mostly determines the jet momentum. However, higher-order calculations are needed to describe the internal structure of the jets and to obtain a complete picture of the produced hadrons. As a result, parton shower algorithms serve as an alternative to perturbative calculations for simulations. These algorithms evolve the momentum transfers from the typically high scales of the hard process down to low scales, around 1 GeV, where confinement leads to the hadronization of partons into hadrons.

One crucial free parameter in the parton shower is the cutoff for transverse momentum, Q. This cutoff is essential because the possibility of collinear emission of partons is divergent, resulting in an infinite number of partons. However, a minimum distance is required at which partons must be physically distinguishable as separate particles. This concept is somewhat equivalent to QED, where a high-energy photon splitting into a boosted e^+e^- pair does not ionize an atom as their proximity leads to the atom only seeing the combined, electrically neutral state. Therefore, a distance measure must be introduced to generate only physically meaningful processes. One such choice is the relative transverse momentum between the partons. A threshold value Q is then set, below which partons are not considered in the showering process.

In `Herwig7`, the implementation of this parameter has different variations depending on the possible color-connected particles. For the scope of this thesis, only the final state particles are color-connected. As such, the relevant parameters for massless and massive splitting are `FFMassiveKinematics:IRCutoff` and `FFLightKinematics:IRCutoff`, respectively. Going forward, these will be treated as a single parameter, denoted as IRCutoff.

Another critical parameter of the parton shower is the strong coupling constant α_s. This parameter is particular as it affects both the hard scattering and shower process. In the generator, different instances of α_s are present for different parts of the event generation loop. For the purpose of tuning, all α_s implementations are treated the same and will be collectively referred to as AlphaQCD .

3.3 Hadronization

The next step in Monte Carlo event generation after the parton shower process is hadronization. Its goal is to combine the generated final state partons into hadrons,

representing a collision event's measurable final state. As the QCD coupling increases in these low-energy regimes, perturbative QCD calculations become unreliable. Currently, there is no established method to calculate or simulate these hadronization processes from first principles accurately. Therefore, hadronization simulations rely on heuristic models, which are inspired by QCD principles. There are two commonly used models in Monte Carlo event generators. One of these models is the *string model*, specifically implemented through the Lund model, which will be discussed in more detail in Section 3.3.2. The other approach is the *cluster model*, which will be described in more detail in Section 3.3.1.

3.3.1 Cluster model

The cluster model, the default hadronization used by the `Herwig7` package, is based on the concept of preconfinement, which states that the resulting color singlets following hadronization follow a universal mass distribution that is independent of the scale of the hard process Q. For more details, see Ref. [35]. An example of the hadronization process in the cluster model is shown in Figure 3.1. As a first step,

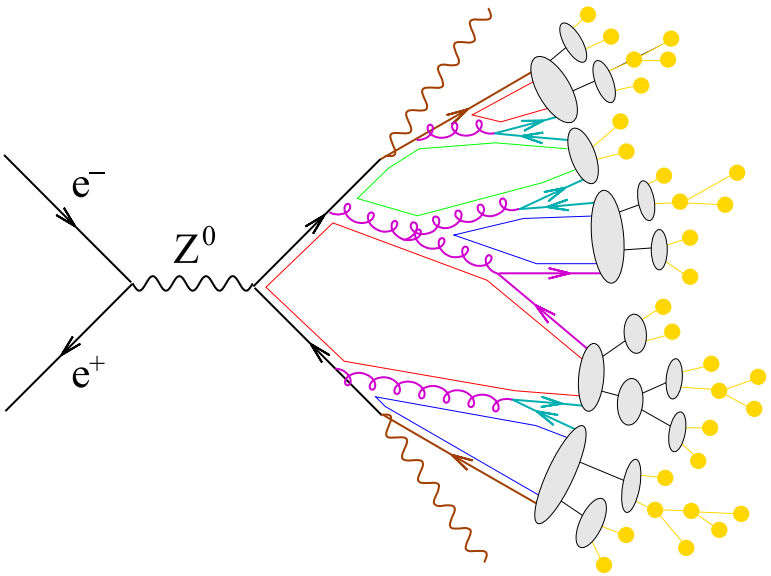


Figure 3.1: A schematic overview of the hadronization process for a $e^+e^- \to Z \to q\bar{q}$ event in the cluster hadronization model [36]. The grey areas represent clusters, while the yellow dots represent the hadrons.

all final state gluons are split isotropically into quark anti-quark pairs. This process is performed in a non-perturbative fashion, and as such, the quarks and gluons are assigned a mass quantity referred to as *constituent mass*. These constituent masses

are essential model parameters and will be the subject of tuning. After the decay of gluons, clusters are formed with the remaining quarks and anti-quarks. The momentum of these clusters follows the sum of the momentum of the constituent partons. These clusters are treated like mesonic resonances, further decaying into lighter resonances and, finally, stable hadrons. The decay of a heavy cluster with the mass M follows the condition

$$M^{\mathrm{Cl_{pow}}} \geq {\mathrm{Cl_{max}}}^{\mathrm{Cl_{pow}}} + (m_1 + m_2)^{\mathrm{Cl_{pow}}} \tag{3.1}$$

with the masses of the constituent partons $m_{1,2}$. The parameter $\mathrm{Cl_{max}}$ defines the maximum cluster mass while the parameter $\mathrm{Cl_{pow}}$ determines whether a cluster undergoes fission. In cases where a cluster splits, a new $q\bar{q}$ pair is taken out of the vacuum. These quarks can either be up, down, or strange quarks, with their respective probability being manually adjustable. The mass distribution of the resulting clusters, M_1 and M_2, is given by

$$M_1 = m_1 + (M - m_1 - m_q)\mathcal{R}_1^{1/P} \tag{3.2}$$

$$M_2 = m_2 + (M - m_2 - m_q)\mathcal{R}_2^{1/P} \tag{3.3}$$

with the mass of the parton popped from vacuum m_q and a random number $\mathcal{R}$. The parameter P, referred to as PSplit, regulates the mass distribution of the new clusters. Furthermore, a requirement is set on the sum of the clusters to be greater than their constituent parton and smaller than the mass of the original cluster. It is worth noting that clusters containing partons from the hard scattering process are treated differently. For these, a soft distribution is used to calculate the new mass of the cluster.

The next step in the cluster hadronization process pertains to the selection of the decay products for the clusters into hadrons. The main challenges for this step are determining appropriate rates for the flavor distribution and meson-to-baryon rates for the hadrons. Further details on this process are available in Ref. [37].

At last, after selecting the decay products, the clusters are decayed. Generally, these decays are carried out isotropically. However, when the hadrons contain partons from the perturbative scattering process, their general direction is preserved. In addition, Gaussian smearing is applied to the angle of the hadron, resulting in an angle θ_{smr} as given by

$$\cos(\theta_{\mathrm{smr}}) = 1 + \mathrm{Cl_{smr}} \log \mathcal{R}\,. \tag{3.4}$$

Here, $\mathcal{R}$ is a random number, and $\mathrm{Cl_{smr}}$ is a free parameter. Clusters with insufficient masses to decay into the selected hadrons are decayed into the lightest possible hadrons based on their flavor. Their energy and momentum are reshuffled to allow for such a decay. For more details, see Ref. [37].

All free parameters mentioned for the cluster hadronization model are listed in Table 3.1. The parameters $\mathrm{Cl}_{\mathrm{max}}$, $\mathrm{Cl}_{\mathrm{pow}}$, $\mathrm{Cl}_{\mathrm{smr}}$, and PSplit have different variations for light, charm, and bottom quarks. In the context of tuning, these variations will be treated equally, significantly reducing the number of tunable parameters. It is worth noting that the model has additional parameters, e.g., flag variables, which enable different settings, such as varying the cluster mass distributions. These parameters will be left to their default values. The complete list of parameters used for tuning can be found in Section 7.2.

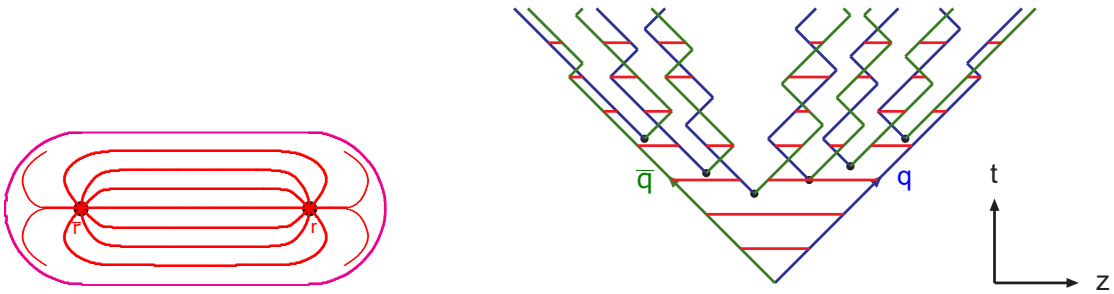
Table 3.1: Free parameters relevant to the cluster hadronization model in `Herwig7` including their flavor-specific variations and a short description. The parameter names listed in the variations reflect the internal parameter names in the `Herwig7` and the tuning framework.

Parameter	Parameter variations	Description
$\mathrm{Cl}_{\mathrm{max}}$	ClMaxLight, ClMaxCharm, ClMaxBottom	Maximum cluster mass
$\mathrm{Cl}_{\mathrm{pow}}$	ClPowLight, ClPowCharm, ClPowBottom	Cluster fission threshold
$\mathrm{Cl}_{\mathrm{smr}}$	ClSmrLight, ClSmrCharm, ClSmrBottom	Angle smearing in decays
PSplit	PSplitLight, PSplitCharm, PSplitBottom	Mass distribution in decays
m_g	g:ConstituentMass	Gluon constituent mass

3.3.2 Lund string model

The Lund string model is the default hadronization model of the `Pythia` event generator and is based on the Lund string fragmentation framework [38, 39]. It is based on the linear confinement principle present in QCD. One can imagine, for example, the production of a $q\bar{q}$ pair in a collider, moving along an axis z. The color charge of these particles creates a color flux tube connecting both quarks, as seen in Figure 3.2a. This connection tube can be imagined to be a massless string with a constant, linear energy density κ. Consequently, given the distance between the quarks r, the potential energy stored by the flux tube is given as $V(r) = \kappa r$. This linear potential is also present in other low-energy descriptions of QCD, such as lattice QCD and hadron mass spectroscopy [38].

As the quark and anti-quark move farther apart, the potential energy of the string increases. A schematic of this process is shown in Figure 3.2b. This allows for the creation of a new $q\bar{q}$ pair within the string, effectively splitting the string into two separate systems. This addition of quark anti-quark pairs, splitting the color connection flux tube, effectively screens the color charge from the outside, leaving color singlets in the final state. As the distance of the quarks increases further, additional quark anti-quark pairs are generated. For each breaking of the string,



(a) (b)

Figure 3.2: A sketch of the Lund string model [36]. Figure 3.2a shows a flux tube connecting a quark and anti-quark. Figure 3.2b illustrates the motion of a quark anti-quark pair and the corresponding string system. The horizontal lines represent the connecting string, while the diagonal lines represent (anti-)quarks.

only a fraction z of the energy and longitudinal momentum can be transferred to the newly created hadron. The choice for a probability distribution for this fraction $f(z)$ is generally arbitrary. However, due to symmetry requirements with respect to either beginning the splitting process with the q or $\bar{q}$, the choice is reduced, as shown in Ref. [40], to

$$f(z) \propto \frac{1}{z}(1-z)^{\mathrm{a_L}} \exp\left(-\frac{\mathrm{b_L} m_\perp^2}{z}\right), \tag{3.5}$$

with the transverse mass of the original system $m_\perp$ and the free parameters aLund ($\mathrm{a_L}$) and bLund ($\mathrm{b_L}$). Although bLund is universal, the parameter aLund can be modified using the $\mathrm{a_{ExS}}$ and $\mathrm{a_{ExDi}}$ parameters, which will increase the value of $\mathrm{a_L}$ in case of strange quark or diquark production during the fragmentation, respectively.

In addition to the fragmentation along the z axis, fragmentation in the perpendicular direction p_T has to be considered. The quark anti-quark pairs follow a Gaussian spectrum and have opposing p_T, as the string is not expected to have transverse momentum. As such, the distribution of p_x and p_y components follows

$$P(p_x, p_y) \propto \exp\left(-\frac{p_x^2 + p_y^2}{{\sigma_\mathrm{pt}}^2}\right). \tag{3.6}$$

The parameter σ_pt is a free parameter and determines the width of this distribution. As the hadron momentum will be impacted by the transverse momentum from two

separate quark pairs, the width of their transverse momentum is given by

$$\langle p_{\mathrm{T}}^2 \rangle_{\mathrm{Had}} = 2\sigma_{\mathrm{pt}} \ .$$

A table listing the parameters for the Lund string hadronization model, along with brief descriptions, can be found in Table 3.2. Further details of the Lund string hadronization model and its implementation in `Pythia` can be found in Ref. [41].

Table 3.2: List of free parameters relevant for the Lund string hadronization model implemented in `Pythia`, the internal parameter name, and a short description.

Parameter	Internal parameter	Description
a_{L}	aLund	Longitudinal momentum fragmentation parameter
b_{L}	bLund	Longitudinal momentum fragmentation parameter
a_{ExS}	aExtraSQuark	Additional aLund for strange quark produciton
a_{ExDi}	aExtraDiQuark	Additional aLund for diquark produciton
σ_{pt}	sigmaPT	Width of transverse momentum distribution

3.4 Event simulation with Herwig

The generation of hard processes in `Herwig7` is based on the `ThePEG` [42] tool. For the hard process calculations, `ThePEG` provides three main mechanisms. Some processes, such as lepton collisions, have hard-coded and pre-calculated matrix elements. Alternatively, a generic matrix element calculator can be employed for, for example, $2 \to 2$ processes. Lastly, `ThePEG` provides interfaces to read events produced from external sources. For this thesis, the hard process is computed at next-to-leading order precision. To accomplish this, `Herwig7` interfaces the generation of the hard process to the `MadGraph5` matrix element generator [43] in conjunction with the `OpenLoops` one-loop library [44]. Using this setup, the $e^+e^- \to Z/\gamma \to 2, 3, 4, 5$ partons process which will be studied in this thesis is produced. The generation of the process with two partons is performed at next-to-leading order, while processes with higher multiplicities are generated at leading order.

3.5 ThePEG - A Pythia8 interface for ThePEG MC event generator

For the scope of this thesis, two different hadronization models are used. By default, `Herwig7` provides the option to use the cluster hadronization model described in Section 3.3.1. The Lund string hadronization model, described in Section 3.3.2, is implemented in `Pythia8`. Therefore, an interface is needed to combine the hard process and shower from the default `Herwig7` workflow to the `Pythia8` hadronization model. This is provided by `ThePEG` package [33], a C++ framework specifically designed for this task. For more details on the application of this package, see Ref. [10].

3.6 Rivet - The event generator validation system

In order to perform Monte Carlo tuning on `Herwig7` and `Pythia8`, data for the observables of interest is needed. For that purpose, the framework `Rivet` is used. The `Rivet` (Robust Independent Validation of Experiment and Theory) toolkit serves as a validation system for Monte Carlo event generators [32, 45]. It provides a wide range of experimental analyses, containing and preserving data and analysis code.

The primary objective is to compare user-generated Monte Carlo predictions with actual experimental data. However, the complexity of measured observables at particle colliders can vary significantly, ranging from relatively simple event counts to sophisticated differential measurements of composite variables such as hadronic event shapes. Usually, these measurements are performed in a specific fine-tuned fiducial phase space. Additionally, detector effects such as efficiency, acceptance, and resolution have to be considered. To address this, one can either apply unfolding corrections to data or, alternatively, fold these detector effects into the Monte Carlo predictions. The calculation of variables and the handling of kinematic bounds of the fiducial phase space are accounted for by the analysis scripts provided by the experiments through `Rivet`.

In summary, `Rivet` provides a database of measurements and a straightforward and lightweight application of analysis code to directly compare Monte Carlo event generator output to particle collider experiments. As such, `Rivet` is widely used in Monte Carlo tuning efforts such as [3, 5–8]. It should also be noted that the reference data for `Rivet` is connected to the `HEPData` database [46], which will be useful when discussing correlations in Section 7.8 as it provides the statistical and systematic uncertainties separately.

4 Introduction of Bayesian statistics in data analysis

In this chapter, the concepts of Bayesian analysis, as well as the usage of numerical Monte Carlo methods in Bayesian inference, are discussed. Section 4.1 introduces the concept of probability in a Bayesian framework and the derivation of Bayes' theorem. The next Sections 4.2 and 4.3 cover the inference of information using a Bayesian ansatz, as well as its implications, interpretation, and possible difficulties. In the last Section 4.4, numerical methods for Bayesian inference and Monte Carlo methods such as Markov chain Monte Carlo and the Metropolis-Hastings algorithm are discussed.

4.1 Bayesian probability and Bayes theorem

There are several theories on the description and interpretation of probabilities. These theories generally employ axiomatic systems along with rules for calculating probabilities [47]. One such set of axioms, first formulated by Kolmogorov in 1933, remains fundamental in the field of probability theory to this day [48]. Given a set of elementary events E and a subset of this space A, a number $p(A)$ can be assigned, called the *probability of the event* A. The following three axioms apply to the probability measure p:

First Axiom - non-negativity

The probability p of the event A is a non-negative real number that is greater or equal to zero

$$p(A) \in \mathbb{R}, p(A) \geq 0\,. \tag{4.1}$$

Second Axiom - unity

The probability that at least one of the elements of E occurs is one

$$p(E) = 1\,. \tag{4.2}$$

Third Axiom - additivity

For any disjoint sets of E, for example, $A \in E$,$B \in E$ with $A \cap B = \emptyset$, the probability of either event occurring is equal to the sum of their probabilities:

$$p(A \cup B) = p(A) + p(B)\,. \tag{4.3}$$

From these axioms, some immediate properties of probabilities can be inferred. One of these is, for example, the rule for the complementary set of A, $\bar{A}$, of which the probability has to follow $p(\bar{A}) = 1 - p(A)$.

Bayes' Rule

Rules for the conditional probability of an event A occurring after the event B, given by $p(A|B)$, can be inferred from these axioms. Given $p(B) > 0$ the conditional probability is defined as

$$p(A|B) = \frac{p(AB)}{p(B)}\,, \tag{4.4}$$

where $p(AB) = p(A \cap B)$ is the probability of both events occurring. This implies

$$p(A \cap B) = p(A|B)p(B)\,. \tag{4.5}$$

The equality $p(A \cap B) = p(B \cap A)$ can be used to substitute the left side, which can then be divided by $p(B)$, resulting in the equation

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)} \tag{4.6}$$

which is known as *Bayes' theorem*. The first concepts of a Bayesian approach to probability were discussed by Thomas Bayes in Ref. [49]. In addition, the law of total probability

$$p(B) = \sum_{i=1}^{N} p(B|A_i)p(A_i), \tag{4.7}$$

where $A_i \cap A_j = \emptyset$ for $i \neq j$ and $\sum_{i=1}^{N} A_i = E$, can be used to derive:

$$p(A|B) = \frac{p(B|A)p(A)}{\sum_{i=1}^{N} p(B|A_i)p(A_i)}\,. \tag{4.8}$$

This modified version is often referred to as the *Bayes-Laplace theorem* and is generally used as the basis of knowledge inference in a Bayesian context. Additional information on the implications of Kolmogorov's axioms in the field of statistics can be found in Ref [48].

4.2 Bayesian inference

In Section 4, Bayes' theorem was derived as shown in Equation (4.6). While this equation serves as the foundational principle of Bayesian inference, some modifications are needed to apply this formalism to specific models and data. First, Bayes' theorem is rewritten in terms applicable to model-based analyses:

$$p(\vec{\theta}|\vec{D}, M) = \frac{p(\vec{D}|\vec{\theta}, M)p(\vec{\theta}|M)}{p(\vec{D}|M)} \tag{4.9}$$

given the model M, the parameters $\vec{\theta}$, and the data $\vec{D}$.

The resulting term $p(\vec{\theta}|\vec{D}, M)$ is the so-called *posterior probability distribution*, often referred to as posterior, and is the desired quantity. Given the model and data, it describes the distribution of the model parameters. Its result depends on the *prior probability density* $p(\vec{\theta}|M)$, which is also referred to as prior. The prior describes the distribution of the model parameters $\vec{\theta}$ before the data $\vec{D}$ is taken into account. The term $p(\vec{D}|\vec{\theta}, M)$ represents the probability of the data $\vec{D}$ being measured given the model M with the parameters $\vec{\theta}$. This term is commonly referred to as *likelihood*. Lastly, there is the denominator of the fraction $p(\vec{D}|M)$, also called marginal likelihood or *evidence*. This term is described by

$$Z = p(\vec{D}|M) = \int_{\vec{\theta}} p(\vec{D}|\vec{\theta}, M)p(\vec{\theta}|M)d\vec{\theta} \tag{4.10}$$

and represents the probability of measuring the data $\vec{D}$ under the assumption of the model M. In Bayesian inference, where the focus is the estimation of model parameters, the treatment of this integral is critical. The dimensionality of the parameters and the complexity of the likelihood can vary greatly depending on the model. This may lead to a high computational burden, so the direct calculation of the integral is often avoided. Typically, this is feasible when the desired quantities are independent of the normalization, such as the mode or mean from the posterior.

Model comparisons

Certain applications, such as limit setting, fitting, and phase space exploration, are independent of normalization. However, when comparing models in a Bayesian context, normalization serves as a measure itself. In such cases, proper evaluation of this evidence becomes essential. Different models M_1 and M_2 can be compared

using the so-called *Bayes factor* given by

$$K = \frac{p(\vec{D}|M_1)}{p(\vec{D}|M_2)} = \frac{\int_{\vec{\theta}_1} p(\vec{D}|\vec{\theta}_1, M_1) p(\vec{\theta}_1|M_1) d\vec{\theta}_1}{\int_{\vec{\theta}_2} p(\vec{D}|\vec{\theta}_2, M_2) p(\vec{\theta}_2|M_2) d\vec{\theta}_2} = \frac{Z_1}{Z_2} \,. \tag{4.11}$$

A value of $K > 1$ suggests that the first model is more compatible with the data. Although the scheme for classification based on the Bayes factor is clear-cut, the interpretation of its strength can vary, as discussed, for example, in Ref. [50].

Marginalization

Given the posterior distribution $p(\vec{\theta}|\vec{D}, M)$, one might be interested in the probability distribution of a single parameter θ_i. In order to calculate the corresponding distribution, the posterior is integrated over all other parameters. Consequently, the resulting probability distribution is described by

$$p(\theta_i|\vec{D}, M) = \int_{\theta_j} p(\vec{\theta}|\vec{D}, M) \prod_{j \neq i} d\theta_j \,. \tag{4.12}$$

This distribution is referred to as *marginalized posterior distribution*. The application of marginalization can be extended to the concept of nuisance parameters. In general, nuisance parameters $\vec{\nu}$ are additional model parameters that are not parameters of interest. As such, the posterior distribution can be integrated over the nuisance parameters with

$$p(\theta|\vec{D}, M) = \int p(\vec{\theta}|\vec{D}, \vec{\nu}, M) d\vec{\nu} \tag{4.13}$$

to obtain the posterior for the parameters of interest.

4.3 Applications and difficulties of Bayesian inference

The central aspect of Bayesian inference is the evaluation of the posterior probability in Equation (4.9). It should be noted that the analytical calculation of the posterior is only applicable in very few and, oftentimes, simple cases. As discussed in Section 4.2, the likelihood can be arbitrarily complex, and the dimensionality of the parameter space can be large. Historically, the computational challenge of Bayesian inference had an impact on Bayesian modeling for a long time. Certain prior types, so-called *conjugate priors* that lead to a posterior that itself is from a predicable family of functions, were heavily favored until numerical methods that allowed the sampling of arbitrary functions were adopted, see Refs. [51, 52]. As a result of using such numerical methods, the posterior is usually represented by a set of samples

rather than a functional expression. These samples can then be used to obtain the desired information. Values of interest often include the position of the mean and the maximum, also referred to as the mode, of the distributions. For these cases, the mode of the full distribution, the so-called *global mode*, and the modes of the marginalized distributions, the so-called *marginal modes*, can be of interest. Other quantities, such as estimates of the uncertainties, for example, using the variance or credible intervals, can also be inferred from the posterior distribution. Generally, deriving these properties using the sampled posterior distribution is straightforward. Therefore, the main challenge lies in the computational effort needed to sample the posterior.

There are several numerical approaches to explore the posterior distribution in Bayesian inference. The simplest form of numerical exploration is the *grid-based* evaluation of the posterior. In that approach, for each parameter θ_i, the corresponding axis is split into n points. The dimensionality of the parameter space d equals the number of parameters. Afterward, the product of the likelihood and prior is evaluated on each point. While this approach is straightforward, it quickly runs into the so-called *curse of dimensionality*. As the number of dimensions increases, the number of evaluation points will increase exponentially with $N = n^d$. Typically, this results in unfeasible grid-based sampling for dimensions above $d > 5$. This scalability issue of multidimensional spaces leads to other concepts of sampling, which will be discussed in Section 4.4.

4.4 Monte Carlo Methods as a numerical tool for Bayesian inference

As discussed in Section 4.3, numerical methods are essential in Bayesian inference in data analyses. Several techniques can be applied for sampling the posterior distribution and simulation-based inference. Many of them rely on the concept of *Monte Carlo* integration, which will be explained in the following. In general, Monte Carlo techniques use randomly drawn numbers to derive integrals or solve optimization problems. Usually, the foundation of these techniques relies on uniform random numbers that are independently identically distributed, in short, *IID* samples. The set of uniform IID random numbers between zero and one will be referred to as $\mathcal{U}_{[0,1]}$.

Transform Methods

Given a probability distribution f, the set of random numbers $\mathcal{U}_{[0,1]}$ can be directly transformed to follow f, if its cumulative distribution function (CDF) is invertible

(F^{-1}). This method effectively generates IID random numbers but is limited to distributions with invertible CDFs. Other transformation methods also apply in cases where a function f is linked to a function that can be sampled easily. As an example, sets of two numbers U_1 and U_2 that both follow $\mathcal{U}_{[0,1]}$ can be transformed into two numbers that follow the unit normal distribution $\mathcal{N}(0,1)$. This transformation method is known as the *Box-Muller transform*, as detailed in Ref. [53]. While these methods are beneficial for efficiently generating IID samples, their applicability is very limited. As such, more general methods are described in the following.

Accept-Reject Methods

While the transformation method is efficient yet restrictive, the *accept-reject method* is universally applicable but can suffer from low efficiency. Let $f(x)$ be the target distribution and $g(x)$ a distribution that is easy to sample and satisfies $f(x) \leq Mg(x)$ for a constant $M \geq 1$. Then, one can generate two random numbers X from g and U from $\mathcal{U}_{[0,1]}$. The point X is accepted under the condition that $U \leq f(X)/Mg(X)$. Otherwise, the point is rejected. One can imagine selecting random points from a distribution that covers the target distribution and accepting values within the target. This procedure is repeated and results in the set of X following the distribution $f(x)$. Here, the constant M scales the sample distribution $g(x)$ to cover the area under the function $f(x)$. As such, M and the efficiency depend on the similarity of the two functions. Hence, much optimization has been put into further methods such as *envelope accept-reject sampling* [52].

4.4.1 Markov Chain Monte Carlo

As discussed in the previous Section, several methods exist to sample from arbitrary functions or arbitrary posterior distributions in the case of Bayesian inference. The novelty of Markov chain Monte Carlo (MCMC), compared to simple accept-reject methods, is the inclusion of the already drawn samples to approximate the target distribution better. Samples X^t are drawn in an iterative procedure $(X^1, X^2, ...)$. The defining feature of a Markov chain is the so-called *Markov property*, which states that for any t, the sample X^t only depends on the previous sample X^{t-1}. It follows that the transition probability distribution is given as

$$p(X^t|X^{t-1}) = \pi \; \forall \; t \tag{4.14}$$

where π represents the equilibrium distribution. Defining π so that the Markov chain follows the desired distribution is the main challenge of constructing an MCMC algorithm. In Section 4.4.2, one such approach, the Metropolis-Hastings algorithm, is discussed. The Markov chain can be started once the appropriate

transition probability is constructed. However, it must be considered that the Markov chain only represents the desired distribution in the limit of many samples. The initial sequence in which the Markov chain is not yet converged is called *burn-in* phase. Based on specific criteria, samples generated before convergence are usually discarded. The length of the burn-in and the criterion used to check for the convergence of the Markov chain is a crucial point of the implementation of MCMC samplers and will be discussed further in Chapter 5.

4.4.2 Metropolis-Hastings algorithm

One of the first algorithms to generate Markov chain Monte Carlo samples was the so-called *Metropolis algorithm* [54], which adapts a random walk with an accept-reject criterion to converge to a target distribution. In the case of the Metropolis algorithm, the distribution to propose a new sample point must be symmetric. The *Metropolis-Hastings algorithm* generalized this aspect by also allowing asymmetric proposal functions and will be described in the following [55].

Algorithm 1 The Metropolis-Hastings algorithm

Require: Desired function $f(x)$
1: Draw a random starting point X^1 from the parameter space
2: **for** i in $1:N$ **do**
3: Propose a new point X^* from $g(X^*|X^i)$
4: Calculate $A(X^*, X^i) = \min\left(1, \frac{f(X^*)}{f(X)}\frac{g(X|X^*)}{g(X^*|X)}\right)$
5: Generate a random number u from $\mathcal{U}_{[0,1]}$
6: **if** $u \leq A(X^*, X^i)$ **then**
7: Set $X^{i+1} = X^*$ ▷ Accept the proposed sample
8: **else if** $u > A(X^*, X^i)$ **then**
9: Set $X^{i+1} = X^i$ ▷ Reject the proposed sample
10: **end if**
11: Increment $i = i + 1$
12: **end for**
13: Return the samples $[X^1, \ldots, X^{N+1}]$

The algorithm starts from a random point in the parameter space X and proposes a new point X^* according to a proposal function $g(X^*|X)$. This new point is either accepted or rejected with the probability given by

$$p_{\text{accept}}(X^*, X) = \min\left(1, \frac{f(X^*)}{f(X)}\frac{g(X|X^*)}{g(X^*|X)}\right) \tag{4.15}$$

where $f(x)$ is the target distribution. A pseudocode implementation of the algorithm can be found in Algorithm 1.

In order to show that there is a stationary distribution, that is, the distribution of interest, the so-called *detailed balance* criterion can be used. In simple words, detailed balance describes the property of an algorithm that the flow of probability mass from any point A to any point B is equal to the flow of point B to point A. In the case of the Metropolis-Hastings algorithm, this condition is satisfied as

$$f(X^i)\, p(X^j|X^i) = f(X^j)\, p(X^i|X^j)\ . \tag{4.16}$$

While the theoretical concepts of the Metropolis-Hastings algorithms are straightforward, the actual algorithmic implementation has additional nuisances. For example, to ensure an efficient convergence, the proposal distribution must be tuned dynamically for the algorithm. As part of the complete sampling algorithm, the burn-in and chain initialization processes and the check for convergence also need efficient implementation. The next chapter will discuss a software package for Bayesian inference and its implementation of these features.

5 BAT.jl - A Julia-based toolkit for Bayesian data analysis

Chapter 4 introduced the theoretical concepts of data analysis in a Bayesian context. However, the real-world application of these concepts requires a robust, modern, and effective implementation of these methods. This chapter presents the *Bayesian Analysis Toolkit* (BAT) as a possible software package for Bayesian inference. The main ideas and software considerations of the package are discussed in Section 5.1. The rewrite of the original toolkit to the modern BAT.jl package and the workflow of the toolkit are discussed in Section 5.2. Different sampling algorithms, as well as basic features of BAT.jl, will be elaborated in Section 5.3.These concepts provide the basis for designing and executing performance tests, which evaluate the package's functionalities and will be covered in Chapter 6.

5.1 The Bayesian Analysis Toolkit

In several scientific applications, such as Bayesian inference, complex models often necessitate computations beyond conventional approaches, resulting in a demand for robust numerical algorithms. However, these methods require fine-tuning several parameters to achieve high efficiency and reliability. Several tools provide such automated implementations in the field of Bayesian analysis [56–59]. Usually, these frameworks are specialized to fulfill the specific needs of analyses within a particular research field. The Bayesian Analysis toolkit (BAT) [60] has been designed for the field of high-energy particle physics. It is a C++ library that provides numerical algorithms for optimization, integration, hypothesis testing, and posterior sampling with a focus on the implementation of the Metropolis-Hastings algorithm.

The toolkit has found wide usage within the field of particle physics, including model fitting [61–64], kinematic fitting [65] and limit setting [66–70]. However, its usage extends towards other fields of research as well with applications in astrophysics [71], cosmology [72] and nuclear physics [73].

As the toolkit is designed for particle physics applications, it relies on specific software like the data analysis framework ROOT [74]. In order to remove these domain-specific dependencies and expand and modernize the toolkit, a software re-design was started, which will be discussed in the following section.

5.2 BAT.jl - The workflow of the modern rewrite in Julia

Since 2017, BAT.jl has been in development as a rewrite of the BAT software package in the *Julia* programming language [1]. One of the primary design considerations is portability, as there are no requirements on external, particle physics-specific software packages such as ROOT.

The Julia programming language was chosen as it allows for performant implementations while allowing a user-friendly script-like syntax similar to Python [75]. In addition, Julia offers automatic differentiation of code with few additional requirements [76]. This enables using, e.g., gradient-based sampling or optimization without needing user-defined gradients for likelihoods and prior densities. Furthermore, Julia supports parallelization and distributed computing and interfaces with C, C++, Python, FORTRAN, and many other languages. This allows users with pre-existing workflows in other languages to interface their models and likelihoods directly to BAT.jl with little to no performance loss. These features make Julia a great candidate language for numerical and scientific applications. A report on the performance of Julia in the context of high energy particle physics analyses can be found in Ref. [77].

BAT.jl is a registered package within the Julia package infrastructure that simplifies the installation procedure.

```
using Pkg
Pkg.add("BAT")
using BAT
```

The BAT.jl package is centered around probability densities that can be normalized or unnormalized. These densities are defined by the user, for example, by providing a likelihood of a model. The likelihood is then combined with a prior density in order to derive the posterior of interest.

```
posterior = BAT.PosteriorMeasure(likelihood, prior)
```

In these cases, the likelihood can be any model or function that returns a value for the (log)-likelihood for a given set of parameters. Priors, however, have to be a sub-type of `AbstractMeasureOrDensity` and are required to be IID sampleable. For commonly used distributions, the Julia package Distributions.jl [78] can be directly interfaced to the required type with BAT.jl. In all cases, the resulting posterior must provide boundaries for the parameters. These are relevant for sampling purposes, e.g., for initializing a grid of points to sample from. Usually, these bounds are set by

the bounds of the prior. However, if needed, the object containing the likelihood can also contain boundaries. In cases where both provide information on the bounds, the intersection of the sets of boundaries is used.

Following the posterior definition, BAT.jl allows users to integrate, sample, or optimize the posterior for mode finding.

```
integral = BAT.bat_integrate(posterior, integrator)
samples = BAT.bat_sample(posterior, sampler)
mode = BAT.bat_findmode(posteiror, estimator)
```

The sampling algorithms in BAT.jl, which will be described in Section 5.3, are implemented with a set of default settings, which should be sufficient for most use cases. However, all samplers, estimators, and integrator settings can be accessed and changed to fit user-specific needs.

5.3 Sampling Algorithms

The BAT.jl software package has implementations of several different sampling algorithms. These algorithms can be categorized by their approaches and serve different use cases. For instance, pseudo-random number samplers produce consistent results but are inefficient for problems in higher dimensions. In these use cases, MCMC-based samplers and nested samplers perform better. Other sampling algorithms, such as partitioned samplers, can perform very well in high-dimensional use cases with the trade-off of additional computation time due to integration. The following sections will describe the types of samplers and their implementations in BAT.jl, while integration and optimization will be discussed in Section 5.4.

5.3.1 Pseudo Monte Carlo samplers

As discussed in Section 4.3 it is reasonable to sample a function by evaluating the target distribution on a grid. However, this approach is only feasible in low numbers of dimensions d, typically $d < 5$. One of the main benefits of such an approach is the independence of the algorithm from the target distribution. Furthermore, such an algorithm is reliable and simplistic, as there is no need for convergence or tuning. These algorithms are called *pseudo Monte Carlo samplers* (PMC samplers).

There are three different PMC algorithms implemented in BAT.jl. The conceptually simplest is the `GridSampler`

```
sampling_algorithm = GridSampler(ppa=100)
```

which creates a point grid by splitting each axis into equidistant steps. The user can set the number of points per axis via the `ppa` arguments. However, one should consider the total number of samples N as it increases exponentially with the number of dimensions d to be $N = \texttt{ppa}^d$. The sampler returns the grid points as samples with the weight of the sample point set to the corresponding posterior value.

Another PMC sampling algorithm is the so-called `SobolSampler` algorithm.

```
sampling_algorithm = SobolSampler(nsamples=10^5)
```

It is based on the concept of Sobol sequences, which are constructed so that the estimation of the integral using the Sobol sample points converges as fast as possible[1] [79]. This is achieved by minimizing the gaps in the phase space in both the entire phase space and, most importantly, in lower-dimensional projections of the parameter space. This strongly contrasts with a point grid, where projections usually contain multiple points in the same space. In order to generate Sobol sequences, the Julia package Sobol.jl[2] is interfaced, which is an independent implementation of the algorithms for Sobol sequences found in Ref. [80, 81]. Contrary to the grid sampler, any arbitrary number of samples can be created using the `SobolSampler`.

The last PMC sampling algorithm is the `PriorImportanceSampler`.

```
sampling_algorithm = PriorImportanceSampler(nsamples=10^5)
```

In this case, IID samples are derived from the prior distribution using methods discussed in Section 4.4. As these samples follow the distribution of the prior rather than the posterior, these sample points are reweighted by multiplying the likelihood value of the sample by its corresponding weight. This sampling method can yield efficient results with fewer samples if the prior can be chosen to resemble the posterior closely.

5.3.2 Markov Chain Monte Carlo based samplers

The basic idea of Markov chain Monte Carlo was discussed in Section 4.4. However, details about the implementation, such as the burn-in process and the convergence criteria, will be discussed in the following.

[1] While integration with Sobol sequences outperforms classical Monte Carlo integration, the convergence of adaptive MC methods might be faster depending on the specific problem.
[2] `https://github.com/JuliaMath/Sobol.jl`

Burn-in process

Different MCMC samplers have different tuning parameters, so the specific burn-in processes differ but follow the same general procedure. The goal of the burn-in process is to create chains that follow the target distribution and are tuned to sample in an efficient manner. For that, multiple chains are initialized using a random sample drawn from the prior distribution. Each chain can be run independently in parallel on multiple threads in order to speed up the sampling process. After the initialization, each tuning step is performed in cycles. During each cycle, 10% of the total number of samples requested by the user are sampled. The sampling and algorithm-specific adjustment of the tuning parameters are performed iteratively for each chain. At the end of each cycle, a convergence check of each MCMC chain is performed. The burn-in process terminates when all chains have converged and the algorithm-specific tuning requirements are fulfilled. Per default, the tuning requires the acceptance ratio of samples to lie within 0.15 and 0.35. The convergence criteria will be discussed in the following section. If tuning and convergence are unsuccessful, the user can decide to either continue the sampling (displaying a warning) or interrupt the sampling with an error. If the sampling continues, points generated during the burn-in process are discarded, and the chains start the MC generation with the user-specified number of steps.

```
burnin = MCMCMultiCycleBurnin(
    nsteps_per_cycle = 10000
    max_ncycles = 30
)
```

The number of samples per chain per tuning cycle, as well as the total number of cycles, can be chosen by the user.

Convergence criteria

In order to ensure the chains of the MCMC sample the target distribution, a convergence test is employed. One convergence criterion implemented in BAT.jl is the Gelman-Rubin test [82]. The concept is based on analyzing the distances between the different Markov chains. Given M parallel chains with N samples per chain of a parameter θ then each chain i contains the sequence $\theta_{1i}, \ldots, \theta_{Ni}$. This allows the per-chain variance W and the between-chain variance B to be calculated as

$$W = \sum_{i=1}^{M} \sum_{j=1}^{N} \frac{(\theta_{ij} - \bar{\theta}_i)^2}{M(N-1)}, \quad \frac{B}{N} = \sum_{i=1}^{M} \frac{(\bar{\theta}_i - \bar{\theta})^2}{M-1} \tag{5.1}$$

where $\bar{\theta}_i$ is the mean of the i-th chain and $\bar{\theta}$ is the overall mean. These variances are used to estimate the target variance V with

$$V = \frac{(N-1)W}{N} + \frac{B}{N} \,. \tag{5.2}$$

Using Equation (5.1) and (5.2), the so-called *potential scale reduction factor* (PSRF) is constructed

$$\hat{R} = \frac{V}{W} \,. \tag{5.3}$$

Generally, given that the chains are initialized from an overdispersed distribution, V will overestimate the target variance, while W will underestimate the target variance as the individual chains do not yet cover the entire range of the target distribution. As such, during the initialization $\hat{R}$ will have a value greater than one while in the limit of $N \to \infty$ both values converge to the same target variance, making $\hat{R}$ approach one. Hence, the distance of $\hat{R}$ to one can be used to measure convergence. This approach was generalized to the multivariate case by Brooks and Gelman [83]. Per default, BAT.jl uses this criterion with a cut-off value $\hat{R} < 1.1$ to test for convergence.

```
convergence = BrooksGelmanConvergence(threshold=1.1)
```

Metropolis Hastings

One of the MCMC algorithms implemented in BAT.jl is the Metropolis-Hastings, which was introduced in Section 4.4.2.

```
sampling_algorithm = MCMCSampling(
    mcalg = MetropolisHastings(),
    nsteps = 10^6,
    nchains = 4
    )
```

The default proposal distribution is a multivariate Student's t distribution, which is tuned by modifying its scale matrix Σ. During the burn-in process, after each cycle, Σ is updated according to the correlation of the newly sampled points to represent the parameters' correlation more accurately. In addition, a scale factor c is applied to Σ, which modifies the range of the proposal distribution. The scale factor c is updated during each cycle according to the current acceptance rate of the chain α. Per default, c is limited to $c_{\text{min}} = 10^{-4} < c < c_{\text{max}} = 100$ while the desired acceptance rate is $\alpha_{\text{min}} = 0.15 < \alpha < \alpha_{\text{max}} = 0.35$.

Effective sample size

A drawback of samples obtained from MCMC methods is that they are correlated. This means that the samples are not independent, so the *effective sample size* (ESS) is smaller than the actual number of samples. BAT.jl provides an estimator for the ESS, which calculates which number of IID samples would provide equivalent information to the number of correlated samples [84]. The estimation is based on the autocorrelation of the samples.

```
ess = bat_eff_sample_size(samples).result
```

Hamilton Monte Carlo

The BAT.jl package includes *Hamilton Monte Carlo* (HMC) sampling, formerly referred to as *Hybrid Monte Carlo*, which, like Metropolis-Hastings, falls under the category of MCMC samplers [85–87]. As dimensionality increases, random-walk Monte Carlo samplers, such as Metropolis-Hastings, become more inefficient as fewer directions lead to the region of interest. HMC mitigates this problem by utilizing gradient information to guide the sampler's movement, keeping it within the relevant phase space. This results in HMC usually yielding higher acceptance rates, faster convergence, and a larger effective sample size. These benefits, however, come at the cost of additional computational effort in each sampling step.
The main concept of HMC is based on the application of Hamiltonian dynamics. To facilitate this, the d-dimensional parameter space is expanded to a $2d$ space by introducing additional parameters p, called momentum. As such, the phase space is changed to $\vec{q} \to (\vec{q}, \vec{p})$. Consequently, the target distribution $\pi(\vec{q})$ is lifted onto the phase space using conditional probabilities over $\vec{p}$

$$\pi(\vec{q}, \vec{p}) = \pi(\vec{p}|\vec{q})\pi(\vec{q}) \ = e^{H(\vec{q},\vec{p})} \tag{5.4}$$

where it can then be rewritten in terms of the Hamiltonian $H(\vec{q}, \vec{p})$. The Hamiltonian can then be decomposed

$$H(\vec{q}, \vec{p}) = -\log \pi(\vec{q}, \vec{p}) = -\log \pi(\vec{p}|\vec{q}) - \log \pi(\vec{q}) \tag{5.5}$$

and used in order to formulate the Hamilton equations of motions

$$\frac{\mathrm{d}q_i}{\mathrm{d}t} = \frac{\partial H}{\partial p_i}, \frac{\mathrm{d}p_i}{\mathrm{d}t} = -\frac{\partial H}{\partial q_i}. \tag{5.6}$$

These equations are then solved for a certain time T to calculate the new point $(\vec{q}^*, \vec{p}^*)$. In order to obtain the proposal $\vec{q}^*$, the solution is marginalized over the momenta. Analogously to the Metropolis-Hastings algorithm, this proposal is either

accepted or rejected by calculating an acceptance ratio. However, since HMC proposal points are calculated using the information on the target distribution, the acceptance rates are significantly higher compared to sampling with non-specific proposal functions.
While there are several benefits to HMC, its implementation is not straightforward, as gradient information is needed. In addition, multiple hyperparameters related to the momenta and times are introduced in the numerical integration of Equation (5.6). For single-sample steps, BAT.jl uses an interface to the AdvancedHMC.jl [88] package, which provides adaptive techniques like the No-U-Turn sampler (NUTS) [89]. Higher-level operations, such as the construction of the posterior and the burn-in process, are handled by BAT.jl. With the efficient capabilities of automatic differentiation in Julia, for example, through the ForwardDiff.jl [76] and Zygote.jl [90] packages, the derivation of the gradient needed for HMC can often be automated, which greatly benefits the usage of HMC in BAT.jl.

```
sampling_algorithm = MCMCSampling(
    mcalg = AdvancedHMC(),
    nsteps = 10^6,
    nchains = 4
    )
```

5.3.3 Partitioned sampling

Another approach to solving the problem of sampling in a high number of dimensions is called *partioned sampling* [91] and is performed by dividing the phase space into multiple sections and using Monte Carlo sampling on each space partition individually. At first, an exploration sampler with a large number of separate chains and a low number of steps is used. This step does not require tuning or convergence but is merely used to derive an initial image of the phase space, like discovering multiple modes or clusters. These exploration samples are then used to divide the space into separate partitions in which sampling is performed independently. As such, the sampling can easily be distributed across multiple threads or working nodes for parallel processing. Since samples from different sections have different normalizations, the samples must be reweighted with the value of the integral of the corresponding subspace. The integral estimation, per default, is calculated using Adaptive Harmonic Mean Integration (AHMI), which will be discussed in Section 5.4. After the reweighting, the samples are merged and returned. Besides the possibility of sampling the subspaces in parallel, partitioned sampling has the benefit of converging more quickly as the separate spaces can be chosen to be unimodal. In addition, due to the weighting using the integral, the relative heights

of modes in multimodal distributions can be estimated correctly in contrast to the ordinary Metropolis-Hastings algorithm. In order to use partitioned sampling, the PartitionedParallelSampling.jl package has to be loaded.

```
sampling_algorithm = PartitionedParallelSampling.PartitionedSampling(
    sampler = mcmc, npartitions=4,
    exploration_sampler=mcmc_exp, integrator = ahmi,
    nmax_resampling=5
    )
```

5.3.4 Nested sampling

An alternative to MCMC-based samplers implemented in BAT.jl is the so-called *nested sampling* [92]. First, a number of samples, so-called *live points*, are drawn from the prior and are assigned a volume. For N live point, each of these points represents $1/N$ of the total volume. In the next step, the live point with the lowest likelihood value, L_1, is removed. This results in the shrinkage of the total volume by $\delta V = 1/N$ to the reduced volume $V = 1 - \frac{1}{N}$. Next, a new live point is sampled from the prior with the requirement of the likelihood to be larger than L_1, which is referred to as likelihood-restricted prior sampling (LRPS). After this step, there is again a total of N live points, each representing $1/N$ of the remaining volume. The live point removal and LRPS steps are repeated with the volume shrinking by a constant factor while the likelihood threshold increases for each step. Sampling is stopped when the remaining volume $V_i = (1 - \frac{1}{N})^i$, for i iterations, becomes negligibly small. This process allows estimating the integral of the phase space Z as

$$Z \approx \sum_i \Delta V_i \cdot L_i \quad \text{with} \quad \Delta V_i = V_i - V_{i-1} = \left(1 - \frac{1}{N}\right)^i \cdot \frac{1}{N} \,. \tag{5.7}$$

In order to get samples that represent samples from the posterior distribution, the removed live points are weighted by $\Delta V_i \cdot L_i$ and returned.

BAT.jl provides two different implementations of nested sampling, which interface to different specialized packages. The first is the `Ultranest`[3] package [93], which is a python package that is accessible using the UltraNest.jl[4] Julia wrapper.

```
sampling_algorithm = ReactiveNestedSampling(min_num_live_points=400)
```

[3] `https://johannesbuchner.github.io/UltraNest/`
[4] `https://github.com/bat/UltraNest.jl`

It uses the *MLFriends* algorithm [94, 95] for the LRPS step, which comes with the benefit of being parameter-free. The other package is NestedSamplers.jl [96], which is natively implemented in Julia. It provides implementations for both single- and multi-ellipsoidal nested sampling.

```
sampling_algorithm = EllipsoidalNestedSampling(num_live_points=400)
```

Both packages generally provide efficient nested sampling implementations, allowing for simultaneous sampling and integration estimation.

5.4 Numerical integration and optimization

In Bayesian inference, the integral of the posterior as well as the point of the highest posterior value, the so-called global mode, are of particular interest. While Section 5.3 describes the sampling of the posterior, which can be used to estimate these quantities, BAT.jl provides numerical methods to derive these quantities directly, which will be discussed in the following.

Adaptive Harmonic Mean Integration

In order to calculate the integral of a posterior distribution, i.e., the evidence, in BAT.jl, the so-called *Adaptive Harmonic Mean Integration* (AHMI) algorithm can be used, which is provided by the AHMI.jl package [97]. AHMI uses already generated samples to estimate the integral using the harmonic mean estimator (HME) [98] in multiple subregions of the phase space. Given the full space E and some finite region $A \subset E$, the HME for the integral in that subspace is

$$I_A = \frac{N_A V_A}{\sum_{x_i \in A} \frac{1}{f(x_i)}} \,, \tag{5.8}$$

where N_A is the number of samples in the subregion, V_A is the volume of the subregion, x_i are the samples and f is the target distribution, i.e. the posterior. The total integral I can be estimated by dividing I_A by $r = N_A/N$ where N is the total number of samples. While the HME method converges to the correct values for an infinite amount of samples, further improvements are needed to provide a usable estimate using finite sample sizes in actual use cases. This is achieved by splitting the initial sample into two exclusive subsets, which are both used individually to create subregions in which HME is performed. However, these regions are then swapped so that the first sample subset is used in the HME for the regions defined by the second set and vice versa. This leads to two estimates for the total integral after combining the HME of the subregions, which are combined to give an overall

integral estimate with a smaller variance. More details and discussion on possible biases are found in Ref. [97].

```
integral = bat_integrate(posterior, AHMI.AHMIntegration()).result
```

In addition to AHMI, BAT.jl provides an interface to the CUBA library [99]. It implements several integration algorithms using both Monte Carlo and deterministic methods. One such example is the VEGAS algorithm [100, 101], which uses importance sampling in order to reduce the variance of the integral estimate.

```
integral = bat_integrate(posterior, VEGASIntegration()).result
```

Optimization algorithms

In addition to methods for evidence estimation, BAT.jl also offers algorithms for phase space exploration aimed at finding the global mode of the posterior distribution. For this purpose, the Optim.jl [102] package is interfaced, which provides different optimization algorithms. As an example of a gradient-free optimization, the Nelder-Mead [103] algorithm can be used.

```
mode = bat_findmode(sampleable, NelderMeadOpt()).result
```

In cases where the gradient of the posterior is known or can be derived automatically using Julia's autodifferentiation features, the LBFGS method [104] can be used for optimization.

```
mode = bat_findmode(sampleable, LBFGSOpt()).result
```

While the global mode of the posterior can be determined using optimization algorithms to find the maximum, the mode of the marginal distribution is also of interest in some instances. Since integrating the posterior, as shown in Equation (4.12), is computationally expensive, the marginal mode can be estimated by binning the samples and selecting the center of the bin with the highest weight. By default, the optimal bin number is chosen using the Freedman-Diaconis rule [105]. In addition to these quantities, BAT.jl allows the estimation of statistical measures like the mean, standard deviation, quantiles, and other custom intervals using samples from the posterior.

6 Implementing a numerical test suite for BAT.jl

In order to ensure the quality and efficiency of the numerical algorithms in BAT.jl, a robust testing framework is indispensable. This chapter elaborates on the design choices and the implementation of such a test suite. First, Section 6.1 will discuss the functions on which performance tests are conducted. Next, the figures of merit used to evaluate the sampling algorithms are introduced in Section 6.2. Last, in Section 6.3, the performance test results will be discussed for both the sampling and integration algorithms for low- and high-dimensional test functions. Beyond providing an ad hoc assessment of the quality of the numerical algorithms, the numerical test suite plays a crucial role in sustaining, or even enhancing, the quality and reliability of these methods during future developments. This is ensured by re-running the tests for every major BAT.jl release.

6.1 Defining test functions for benchmarking

Selecting test functions is a crucial step when establishing a testing framework for MC sampling. These test functions must meet specific criteria in order to serve as an adequate basis for testing. Firstly, IID sampling methods for these distributions are required as they allow the generation of sample sets that are guaranteed to reflect random points from the test functions. These can then be used as a baseline for evaluating the quality of the samples obtained from BAT.jl. Furthermore, test functions should be easily expandable into multiple dimensions. This, together with the presence of both uni-modal and multi-modal distributions, allows for enough complexity to represent the difficulties of real-world data analysis scenarios. Finally, it is helpful to normalize the distributions, as knowing the integral values helps to assess the sampling quality using integration methods.

6.1.1 Normal distribution

One baseline function used for testing is the normal distribution $\mathcal{N}$

$$\mathcal{N}(x|\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{6.1}$$

with the mean μ and standard deviation σ. In the multivariate case, for n dimensions, this extends to

$$f(\vec{x}) = \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\vec{x}-\vec{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\vec{x}-\vec{\mu})\right) \tag{6.2}$$

where $\boldsymbol{\Sigma}$ is the $n \times n$ covariance matrix and $\vec{\mu}$ the n-dimensional mean vector. In order to move the mode of the multivariate normal distribution and vary the variance and correlations, the mean vector and covariance matrix are dynamically changed for each dimension added to the distribution. The elements of the mean vector $\vec{\mu}$ for n dimensions are given by

$$\mu_i = 5 \cdot (1+i) \qquad \text{with} \qquad i \in \{1, ..., n\}$$

and the elements of the covariance matrix can be calculated with

$$\begin{aligned} \Sigma_{i,i} &= \left(4 \cdot \left(1 + \frac{i}{n}\right)\right)^2 \quad \text{with} \quad i \in \{1, ..., n\} \\ \Sigma_{k,i} = \Sigma_{i,k} &= 0.2 \left(1 + \frac{i}{10}\right) \sqrt{\Sigma_{i,i}\Sigma_{k,k}} \quad \text{with} \quad i,k \in \{1, ..., n\} \wedge i \neq k \,. \end{aligned}$$

However, these calculations are only performed in test cases for more than two dimensions. For the two-dimensional tests, the values are set manually and are found in the corresponding part in Section 6.3. An example of a four-dimensional multivariate Gaussian can be seen in Figure 6.1. The samples have been generated using 10^7 IID points and visualized using the plotting functions of BAT.jl, which show the one and two-dimensional marginalized distributions for each parameter.

6.1.2 Multi modal Cauchy distribution

A more complex function compared to the normal distribution is the Cauchy distribution, which is a notoriously difficult function for numerical applications due

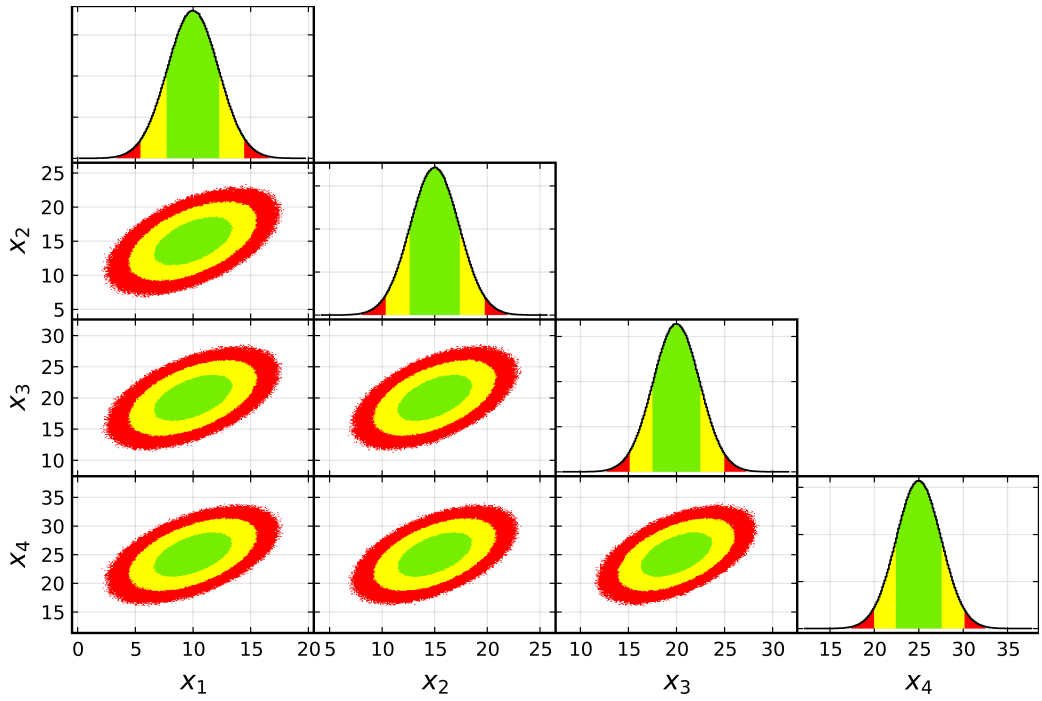


Figure 6.1: Four-dimensional multivariate normal distribution obtained using IID sampling. The diagonal shows the one-dimensional marginalized distributions, while the off-diagonal contains the two-dimensional marginal distributions. The green, yellow, and red areas represent the smallest intervals containing 68%, 95%, and 99% of the probability, respectively.

to its long tails. Its probability density function is given as

$$\text{Cauchy}(x|\mu,\sigma) = \frac{1}{\pi\sigma\left[1+\left(\frac{x-\mu}{\sigma}\right)^2\right]}\,. \tag{6.3}$$

In order to increase the complexity of the test function, four shifted Cauchy distributions are used in the first two dimensions, creating multiple modes. Any additional dimension is represented by a single Cauchy distribution at $\mu_i = 0$, leading to a functional form of the test function of

$$f(\vec{x}) = \prod_{i=1}^{2}[\text{Cauchy}(x_i|\mu_i,\sigma_i) + \text{Cauchy}(x_i|-\mu_i,\sigma_i)] \cdot \prod_{j=3}^{n}\text{Cauchy}(x_j|0,\sigma_j) \tag{6.4}$$

for n dimensions with the mode positions $\vec{\mu}$. The scale parameters $\vec{\sigma}$ determine the spread of the distribution, with larger values producing a wider outcome. Similar to the multivariate normal, the coefficients of the multi-modal Cauchy distribution are changed dynamically during testing. However, the individual elements within $\vec{\mu}$

and $\vec{\sigma}$ for n dimensions are identical and are set by

$$\mu_i = 5 \cdot n \quad \wedge \quad \sigma_i = 0.2 \cdot n \qquad \text{with} \qquad i \in \{1, ..., n\}. \tag{6.5}$$

The four-dimensional multi-modal Cauchy distribution is shown in Figure 6.2. The samples have been generated using 10^7 IID samples and visualized using the plotting functions of BAT.jl.

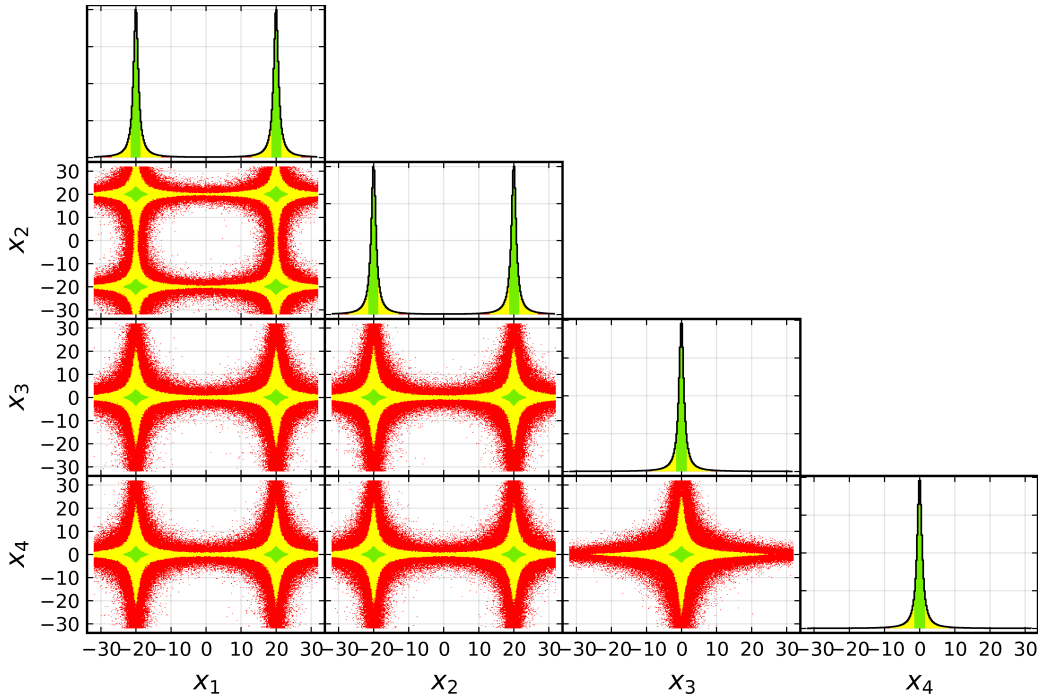


Figure 6.2: Four dimensional multi-modal Cauchy distribution obtained using IID sampling. The diagonal shows the one-dimensional marginalized distributions, while the off-diagonal contains the two-dimensional marginal distributions. The green, yellow, and red areas represent the smallest intervals containing 68%, 95%, and 99% of the probability, respectively.

6.1.3 Funnel distribution

The third test function is the so-called Funnel distribution. It is built using the product of one normal distribution with several additional normal distributions, which have a standard deviation dependent on the first parameter. Its functional form is given as

$$f(\vec{x}) = \mathcal{N}(x_1|0, a^2) \prod_{i=2}^{n} \mathcal{N}(x_i|0, \exp(2bx_1)) \tag{6.6}$$

where a and b are free coefficients that are set to a fixed value of $a = b = 0.5$. The a parameter sets the variance of the first normal distribution while b determines the increase in the spread of the distribution while moving along x_1, effectively setting the width along the parameters $x_{2,\ldots,n}$. The four-dimensional Funnel distribution is shown in Figure 6.3. The samples have been generated using 10^7 IID samples and visualized using the plotting functions of BAT.jl.

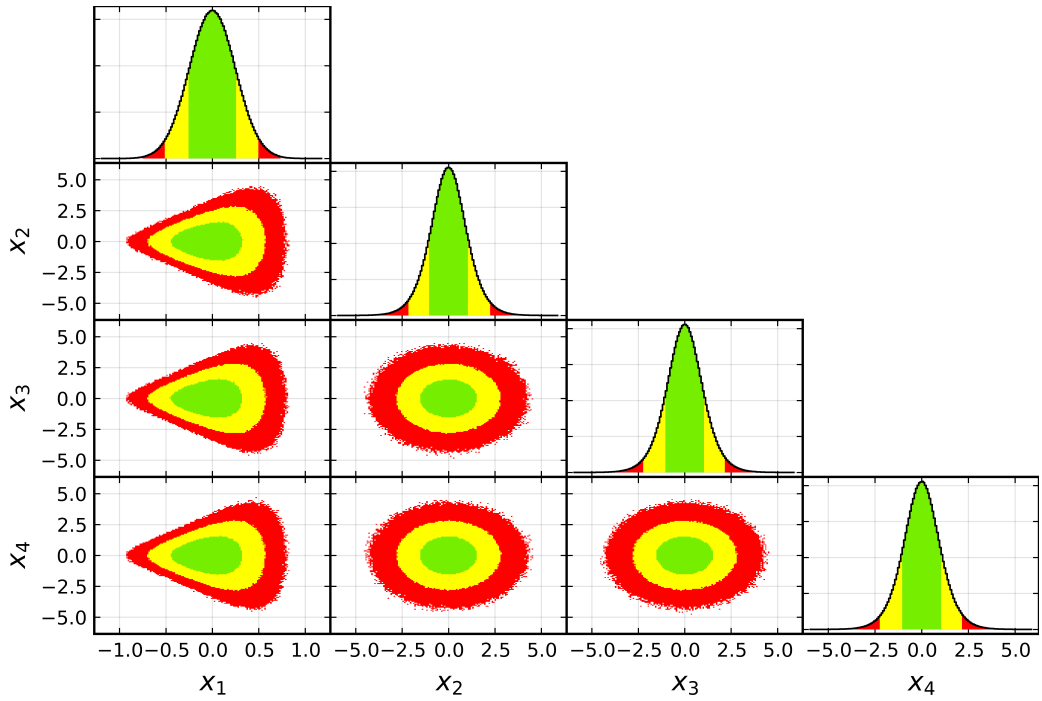


Figure 6.3: Four-dimensional Funnel distribution obtained using IID sampling. The diagonal shows the one-dimensional marginalized distributions, while the off-diagonal contains the two-dimensional marginal distributions. The green, yellow, and red areas represent the smallest intervals containing 68%, 95%, and 99% of the probability, respectively.

6.2 Figures of merit for evaluation of sampling performance

In order to determine the quality of the numerical algorithms of BAT.jl, some measurable criteria have to be defined that reflect the goodness of the desired quantities of the samples. Since the sampling is performed in many dimensions, an evaluation per eye becomes unfeasible. Hence, other metrics and statistical tests are used to compare to known quantities, i.e., mean and mode positions, or to compare

to the results of IID sampling. The following sections will introduce the different metrics and tests used in this work.

6.2.1 Characteristic metrics of sampling

Several metrics are derived from the sampled data that can be directly compared to the known quantities of the test functions. These are the position of the mode, i.e., the maximum of the distribution, the mean of the distribution, as well as the variance. In addition, the integral of the distribution is calculated using AHMI. See Section 5.4 for more details. As the estimation of the integral is performed using the sampled points, deviations of the integral from the expected value can be used as a metric for the sampling quality. In addition, for tests up to two dimensions, the pulls between the binned samples and the analytical values are calculated and visualized. The pulls are defined as the difference between the binned samples and the analytical values divided by the uncertainty of the binned samples. These uncertainties are calculated using the square root of the number of samples in each bin. Given a large enough number of samples, the pulls are expected to be distributed like a unit normal distribution. Lastly, a Kolmogorov-Smirnov test is performed using samples obtained from the MCMC and IID samples. The test is described in Section 6.2.3 and is used to determine if the samples are drawn from the same distribution.

6.2.2 Modification of metrics for tests in higher dimensions

While testing one and two dimensions can be visualized easily, evaluating higher dimensional distributions is more difficult. In order to still be able to compare the results of the sampling, the metrics are modified to be able to be used in higher dimensions. This is most obvious in the case of the KS test, which is only performed on one-dimensional distributions. As such, the test has to be performed on each one-dimensional marginalization of the n-dimensional multivariate distribution. This, however, leads to n different results for the KS test. Since it is expected for the p-values of the KS test to be distributed uniformly between zero and one, the test result can be evaluated using the distribution of these p-values.

6.2.3 The Kolmogorov-Smirnov test

The Kolmogorov-Smirnov (KS) is a statistical test that can be used to determine if two samples are drawn from the same distribution [106, 107]. It is based on the KS statistic D, which is defined as the maximum distance between the empirical

cumulative distribution functions (CDF) of the two samples. The CDF is the probability that a random variable X is less than or equal to x and is given by

$$F(x) = P(X \leq x) = \int_{-\infty}^{x} f(t)dt \tag{6.7}$$

where $f(t)$ is the probability density function of X. The KS statistic is then defined as the maximum distance D which is calculated as

$$D = \sup_{x} |F_1(x) - F_2(x)| \tag{6.8}$$

where F_1 and F_2 are the empirical CDFs of the two samples. Given two samples with size n and m and a significance level α the value of the test statistic D can be compared to the critical value c_α

$$c_\alpha = \sqrt{-\frac{1}{2}\ln\left(\frac{\alpha}{2}\right)} \cdot \sqrt{\frac{n_1 + n_2}{n_1 n_2}} \tag{6.9}$$

which results in the rejection of the hypothesis if $D > c_\alpha$. The test statistic D can also be used to determine the p-value of the test by either using pre-calculated tables or by using the asymptotic approximations. In this work, the p-value calculation is performed using the HypothesisTests.jl[1] package which uses a numerical approximation based on Ref. [108].

6.3 Evaluating the performance of BAT.jl

Using the metrics and tests introduced in the previous section, the performance of the different sampling algorithms implemented in BAT.jl is evaluated. The test focuses on the performance of the MCMC algorithms implemented in BAT.jl, i.e. the Metropolis-Hastings (MH) algorithm and the Hamilton Monte Carlo (HMC) algorithm. First, in Section 6.3.1, tests in lower dimensions are performed to verify the accurate sampling of the distributions. Then, in Section 6.3.2, the performance of the algorithms in higher dimensions is evaluated in order to compare the performance of the MH and HMC sampler.

[1] `https://github.com/JuliaStats/HypothesisTests.jl/`

6.3.1 Two dimensional tests

In order to test the performance in two dimensions, the test functions introduced in Section 6.1 are used. The parameter values for the test functions are summarized in Table 6.1. Testing is performed using the MH algorithm with eight chains and 10^6 sample steps. All other parameters are set to their default values. The resulting samples are visualized using the default BAT.jl plotting functionalities and are shown in Figures 6.4-6.6.

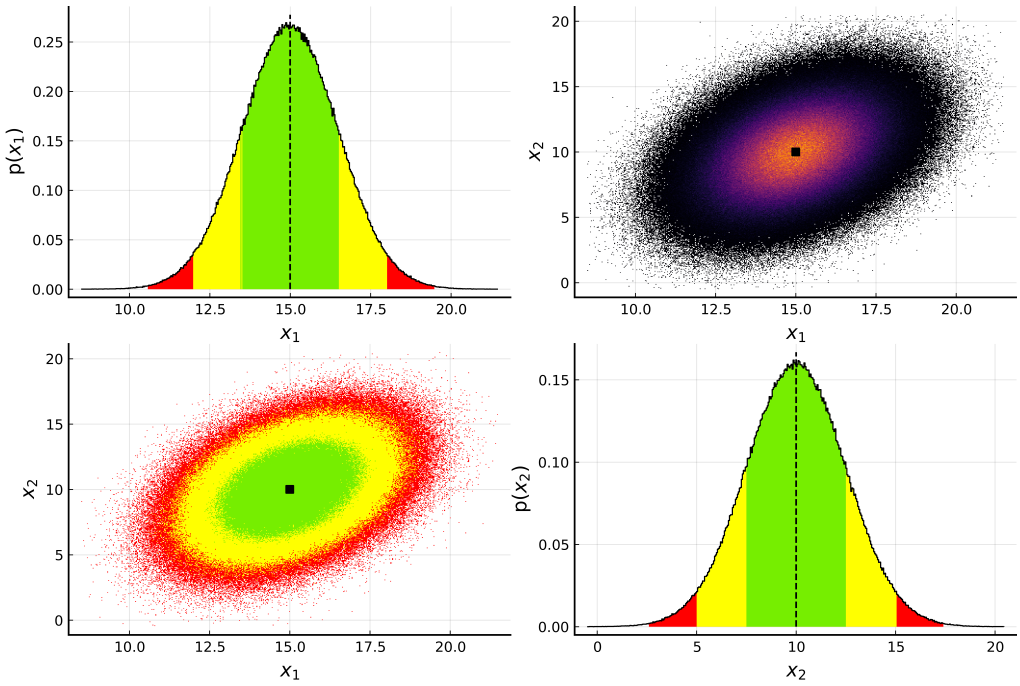


Figure 6.4: BAT.jl default plot for the two-dimensional normal distribution. The plots on the diagonal show the marginalized distributions for each dimension. The off-diagonal plots show the full two-dimensional distribution. The lower left plot shows the probability intervals, while the plot on the top right highlights the shape and contours of the distribution. The dashed lines and the dot represent the position of the global mode of the sample. The green, yellow and red areas represent the smallest intervals containing 68%, 95% and 99% of the probability, respectively. The distributions are normalized to unity.

The results of the mean, variance, KS test and integral are summarized in Table 6.2. For all test functions, the values for the mode position are in good agreement with the expected values. The largest deviation observed is 4.1% for the Cauchy distribution. The mean and variance values are also in good agreement with the analytical values for the normal and Funnel distribution showing at most a deviation of 1.1% for the variance of the Funnel distribution. Since the Cauchy distribution

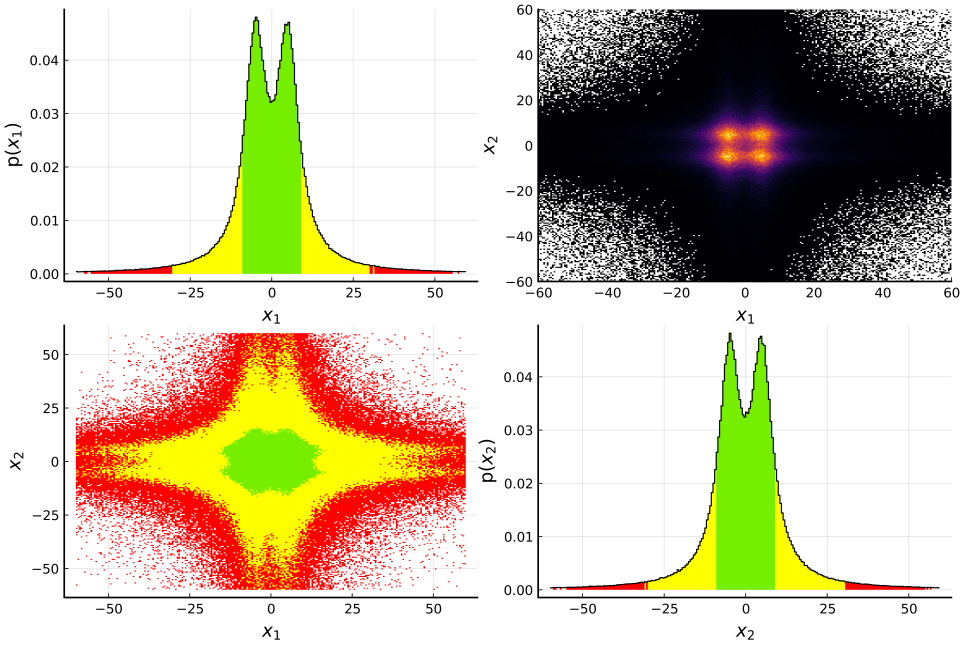


Figure 6.5: BAT.jl default plot for the two-dimensional Cauchy distribution. See the caption on Figure 6.4 for a detailed explanation.

does not have a defined mean and variance, these values are not used for this distribution. However, the KS test and AHMI integral values can be used. For the KS test, the p-values for the marginals are close to one in most cases, with one exception for one marginal of the normal distribution, which results in a p-value of 0.256. While this is significantly lower than the other p-values, it is still well above the typical threshold for the significance level of 0.05. As such, the null hypothesis that the samples are drawn from the same distribution cannot be rejected. Next, the AHMI integral values are compared to their expected values of one. Given the uncertainty provided by AHMI, all integral estimates are in agreement with unity for all distributions.

In order to calculate the pull distribution, an analytical value of a given bin is required. These are calculated by evaluating the analytical function at the center of the bin and scaling it by the bin area and the total number of samples of the function. Visual comparisons of the two-dimensional distributions to the analytical functions are shown in Figures 6.7-6.9. The pulls are calculated for each bin using the difference between the analytical and sampled values, in standard deviation units. The resulting pull distributions are shown in Figure 6.10. The pull distributions for the normal, Cauchy, and Funnel distributions are compatible with the expectation, which, for reference, is indicated in the plots by the orange line. In summary, given the results of the tests in this two-dimensional test case, the performance of the MH algorithm is considered to be satisfactory.

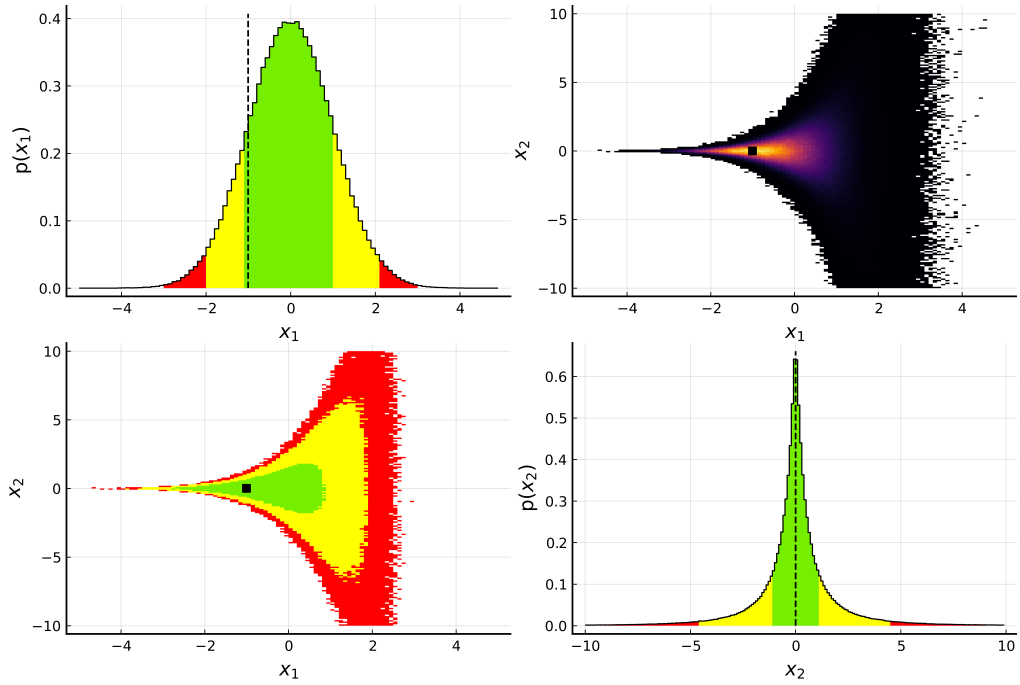


Figure 6.6: BAT.jl default plot for the two-dimensional Funnel distribution. See the caption on Figure 6.4 for a detailed explanation.

Table 6.1: Coefficients used for the testfunctions in the two dimensional tests.

Name	Parameters
Normal	$\mu = \begin{pmatrix} 15 \\ 10 \end{pmatrix}, \sigma = \begin{pmatrix} 2.25 & 1.5 \\ 1.5 & 6.25 \end{pmatrix}$
Cauchy	$\mu = \begin{pmatrix} 5 \\ 5 \end{pmatrix}, \sigma = \begin{pmatrix} 4 \\ 4 \end{pmatrix}$
Funnel	$a = 1, b = 0.5$

Table 6.2: Perfomance test results for the two-dimensional test functions.

Name	Normal		Multi Cauchy		Funnel	
	Target	**Test**	**Target**	**Test**	**Target**	**Test**
Mode	$\begin{pmatrix}15.0\\10.0\end{pmatrix}$	$\begin{pmatrix}15.0\\9.997\end{pmatrix}$	$\begin{pmatrix}5.0\\5.0\end{pmatrix}$	$\begin{pmatrix}4.806\\4.796\end{pmatrix}$	$\begin{pmatrix}-1.0\\0.0\end{pmatrix}$	$\begin{pmatrix}-0.998\\-0.0\end{pmatrix}$
Mean	$\begin{pmatrix}15.0\\10.0\end{pmatrix}$	$\begin{pmatrix}15.001\\9.996\end{pmatrix}$	$\begin{pmatrix}-\\-\end{pmatrix}$	$\begin{pmatrix}-5.687\\0.715\end{pmatrix}$	$\begin{pmatrix}0.0\\0.0\end{pmatrix}$	$\begin{pmatrix}0.001\\-0.001\end{pmatrix}$
Variance	$\begin{pmatrix}2.25\\6.25\end{pmatrix}$	$\begin{pmatrix}2.253\\6.257\end{pmatrix}$	$\begin{pmatrix}-\\-\end{pmatrix}$	$\begin{pmatrix}15765.2\\15050.4\end{pmatrix}$	$\begin{pmatrix}1.0\\7.407\end{pmatrix}$	$\begin{pmatrix}0.997\\7.326\end{pmatrix}$
KS test p-value	$\begin{pmatrix}0.988\\0.256\end{pmatrix}$		$\begin{pmatrix}1.0\\1.0\end{pmatrix}$		$\begin{pmatrix}0.914\\0.997\end{pmatrix}$	
AHMI integral	1.001 ± 0.002		1.0 ± 0.002		1.002 ± 0.002	

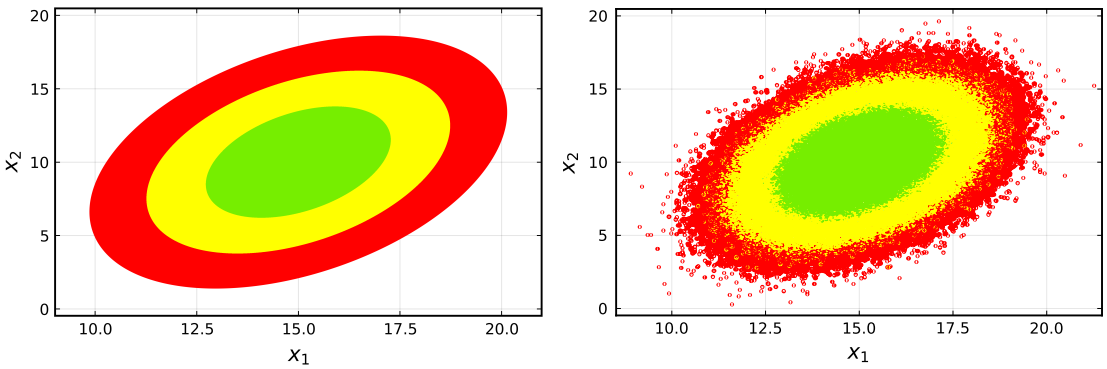


Figure 6.7: Two-dimensional normal distribution using the analytical function (left) and the sampled distribution using MH (right). The green, yellow and red areas represent the smallest intervals containing 68%, 95% and 99% of the probability, respectively.

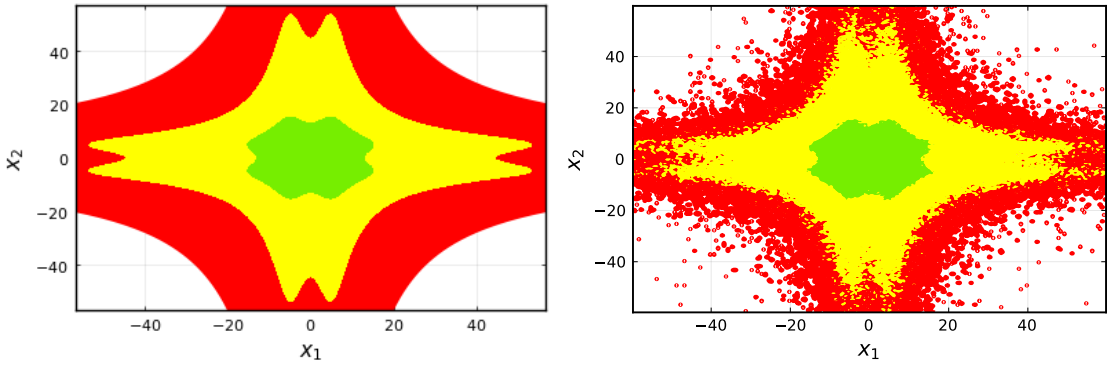


Figure 6.8: Two-dimensional Cauchy distribution using the analytical function (left) and the sampled distribution using MH (right). The green, yellow and red areas represent the smallest intervals containing 68%, 95% and 99% of the probability, respectively.

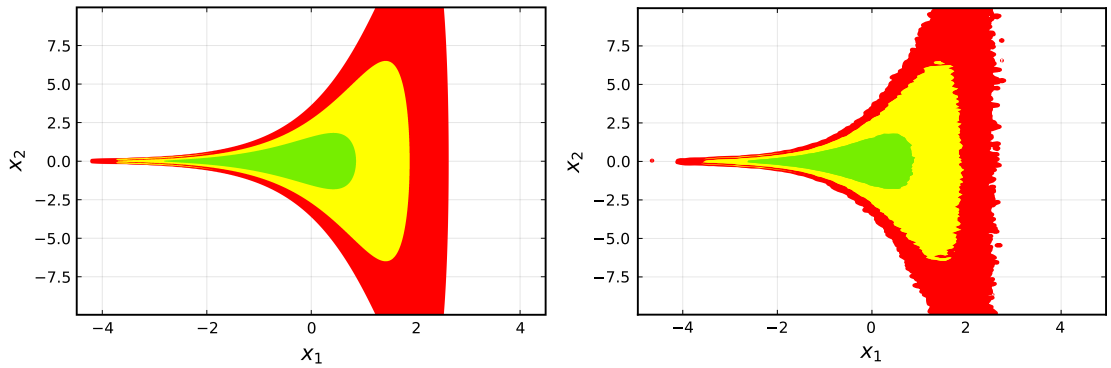
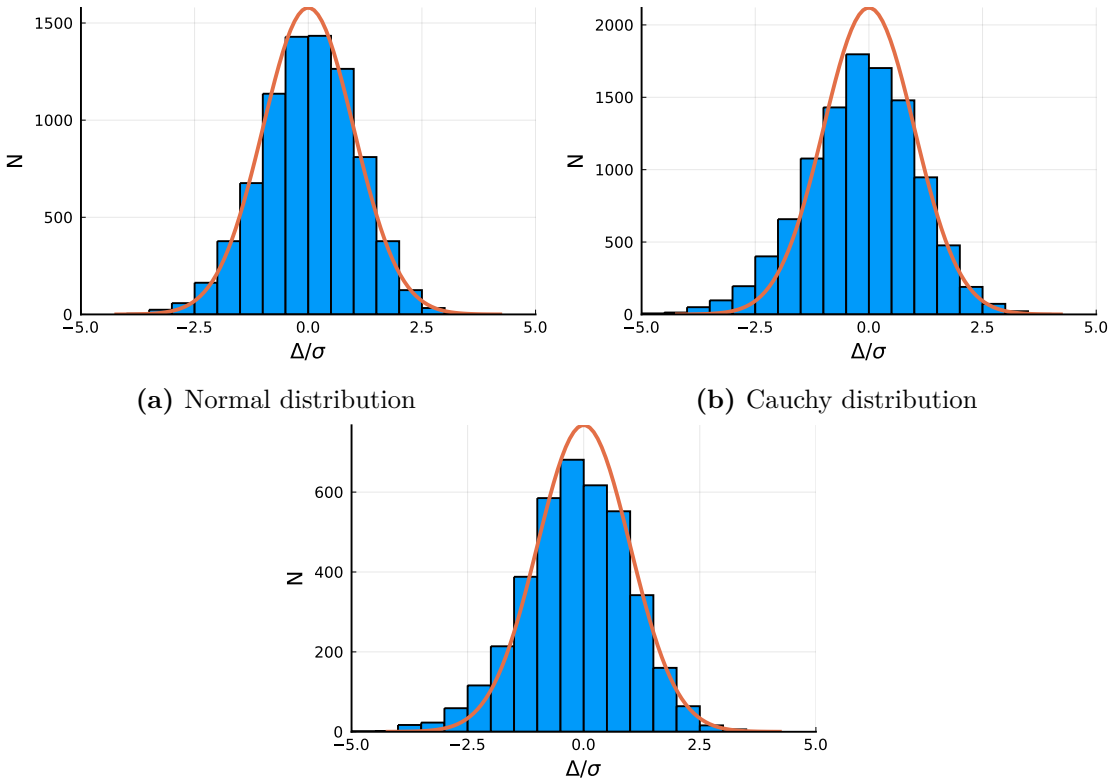


Figure 6.9: Two-dimensional Funnel distribution using the analytical function (left) and the sampled distribution using MH (right). The green, yellow and red areas represent the smallest intervals containing 68%, 95% and 99% of the probability, respectively.



(a) Normal distribution

(b) Cauchy distribution

(c) Funnel distribution

Figure 6.10: Pull distributions for the two-dimensional test functions. The pull distributions are calculated using the difference between the analytical and sampled values, in units of the standard deviation. The orange line represents the expected normal distribution with a mean of zero and a variance of one.

6.3.2 Higher dimensional tests

For the tests in higher dimensions, some settings are changed compared to the two-dimensional tests. These settings are shown in Table 6.3. Most notably, the threshold for the Brooks-Gelman criteria is relaxed in order to probe higher dimensional cases more easily. While this might lead to chains that are not fully converged, the following tests provide a higher sensitivity to verify correct sampling. The KS test and AHMI integration are performed for the MH algorithm from 2 up to 20 dimensions.

Table 6.3: Non-default settings used for the performance tests in n dimensions.

Parameter	Values
Brooks Gelman Convergence threshold	1.6
Brooks Gelman Convergence corrected	False
Init tries per chain	64..1024
Number of steps per chain	$10,000$
Number of steps per cycle	$100,000$
Maximum number of cycles	300

The results of the integration tests are shown in Figure 6.11. The multivariate normal distribution can be integrated up to 20 dimensions with the target value of one within the uncertainty provided by AHMI up to 18 dimensions. In contrast, the Funnel and Cauchy distributions can only be integrated up to 18 and 12 dimensions, respectively. This is due to the fact that the AHMI algorithm cannot create appropriate subvolumes for the integration. In these cases, the AHMI algorithm will report this error and not provide an estimate for the integral. Considering the colored areas in Figure 6.11, the uncertainty provided by AHMI increases with the dimensionality. This increase is especially noticeable in the case of the Cauchy distribution for dimensionalities close to the integration limit at 10 and 12 dimensions. Overall, in most cases where the AHMI algorithm can estimate the integral value, the result is compatible with the expected value of one when considering the uncertainty.

The distribution of the KS test p-values of the test functions between 2 and 20 dimensions are shown in Figure 6.12. As the test is performed for each marginal distribution, the number of tests, and hence p-values, increases with the dimensionality. It is expected that the p-values are uniformly distributed between zero and one. While the values spread throughout the whole range, the distribution tends to be distributed closer to one. This behavior is mostly likely explained by the

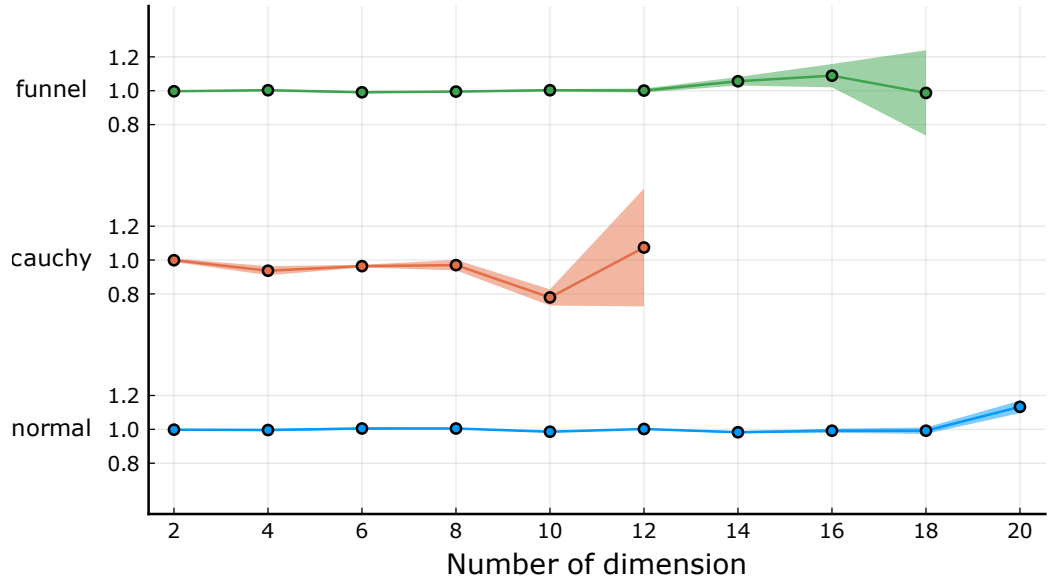


Figure 6.11: Estimaiton of the integral using AHMI for the normal, Cauchy, and funnel distributions from 2 to 20 dimensions. The colored areas represent the uncertainties provided by AHMI.

estimation of the effective sample size slightly underestimating the true value, which leads to larger p-values. However, the KS test p-values are all above the threshold of 0.05, supporting the conclusion that the MH algorithm is sampling the correct distribution.

In addition to the MH algorithm, the performance of the HMC algorithm is tested in higher dimensions in a side-by-side comparison. The KS tests are repeated for both algorithms using the Funnel distribution and are performed from 20 to 35 dimensions in order to compare the performance in a high dimensional use case. The results are shown in Figure 6.13. In general, both algorithms have p-values above the threshold of 0.05. However, the p-values of the HMC algorithm have a larger spread and more closely follow the expected uniform distribution. This is most likely because the estimation of the effective sample size performs better with the HMC algorithm, as the autocorrelation time for this algorithm is smaller.

In conclusion, following the consistent performance of the MH algorithm in the two-dimensional tests as well as the higher-dimensional tests, the algorithm is considered to be working correctly. In addition, as the HMC algorithm performs similarly to the MH algorithm, it is also considered to be able to sample distributions accurately.

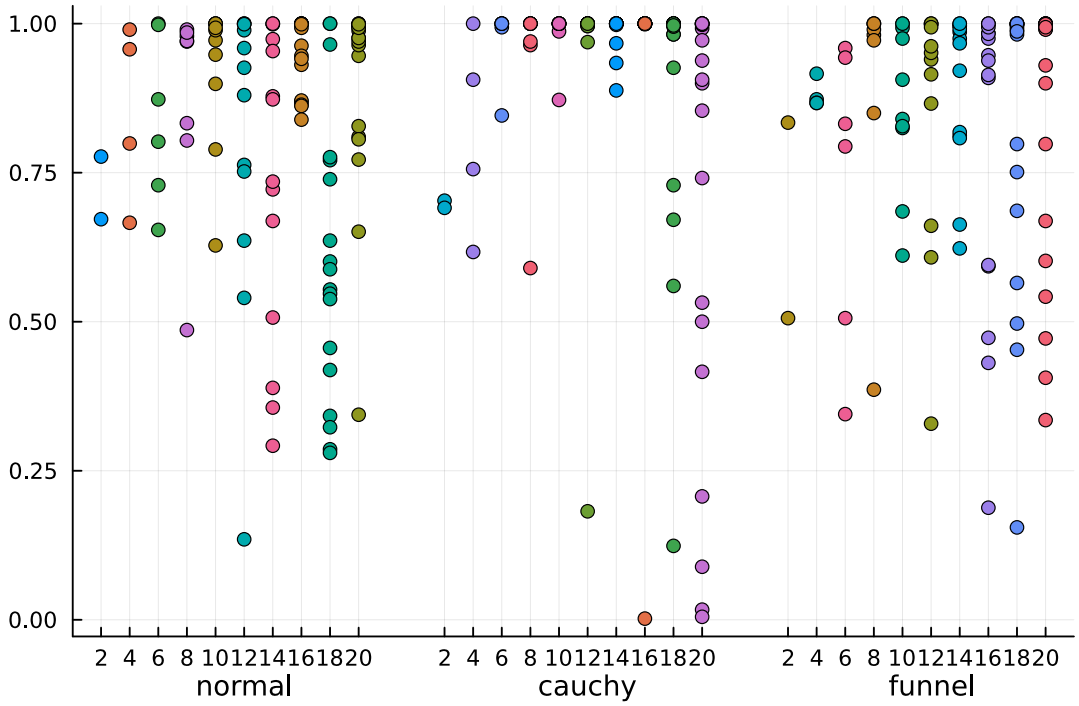


Figure 6.12: KS test p-values for each marginal distribution of the test functions between 2 and 20 dimensions. The horizontal axis shows the number of dimensions, split into the three different test functions, while the vertical axis indicates the p-value.

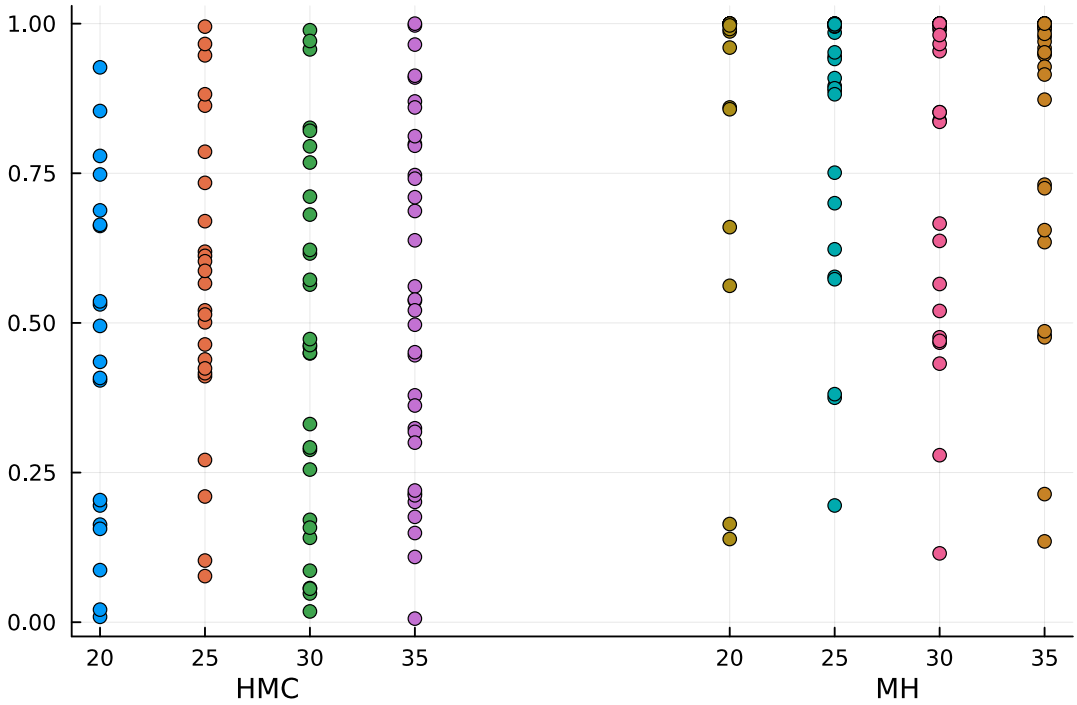


Figure 6.13: KS test p-values for each marginal distribution of the Funnel distribution between 20 and 35 dimensions. The horizontal axis shows the number of dimensions split between the MH and HMC algorithms, while the vertical axis indicates the p-value.

7 Monte Carlo tuning using EFTfitter.jl

In this chapter, the use of a Bayesian approach in the Monte Carlo tuning procedure will be explored. A workflow for tuning Monte Carlo generators in high-energy physics is developed using the numerical tools described in previous chapters. The `Herwig7` cluster hadronization model and the `Pythia8` Lund string model, which are described in Section 3.3, are tuned to LEP data using the BAT.jl toolkit. First, a general description of Monte Carlo tuning is given in Section 7.1. The Monte Carlo event generation is described in Section 7.2, where the parameters, their settings, the analyses used, and the observables are presented. The parametrization procedure of the observables is described in Section 7.3 followed by a comparison to the pseudoinverse method in Section 7.3.1. Then, the goodness of the parametrization is evaluated in Section 7.4. Multiple tests, such as the reduced χ^2 statistics and grid reevaluations, are performed and presented. The EFTfitter.jl library is described in Section 7.5 and is used with the BAT.jl toolkit to tune the Monte Carlo generators. The tuning results are presented in Section 7.6 followed by a discussion of the influence of the correlation of uncertainties in Section 7.8 and the effects of the weighting of observables to the tune in Section 7.9. Finally, the results of the tuning of the `Herwig7` cluster hadronization model and the `Pythia8` Lund string model are compared in Section 7.10.

7.1 Monte Carlo tuning

As described in Section 3, Monte Carlo generators are used to simulate particle collisions. The ability of the Monte Carlo generators to describe data is crucial for interpreting experimental results. While perturbative QCD can describe the interaction between partons, the hadronization process is non-perturbative and cannot be calculated from first principles. Therefore, the hadronization process is described by phenomenological models, which require sets of free parameters. These parameters are tuned to data to improve the description of the hadronization process. This procedure is known as Monte Carlo tuning.

There are several approaches to Monte Carlo tuning, which can be categorized into three types [2]. The first type is the *manual tuning*, performed by individually changing the parameters and comparing the results to data. However, this approach

requires a lot of expertise and is very time-consuming. Additionally, the complexity increases dramatically for larger parameter spaces. The second type is the *brute force tuning*. Here, the parameter space is scanned directly by generating Monte Carlo events and comparing them to data. The parameter values are updated when the proposed parameter set improves the data description. While this approach requires no additional assumptions and benefits from the automation of the tuning procedure, its computational cost is very high due to the event generation in each iteration. The third type is the *parametrization-based tuning*, which is the state-of-the-art approach using the Professor framework [2]. A schematic of this tuning procedure is shown in Figure 7.1. Parametrization-based tuning operates on the premise that the Monte Carlo generator response can be parametrized as a function of the tunable parameters. In order to obtain such a parametrization, the Monte Carlo generator is used to obtain samples for some, usually hundreds, of reference points in the parameter space. With the obtained data, a function is fitted to each bin of each observable. The resulting function is computationally cheap to evaluate and is used to tune the parameters to data.

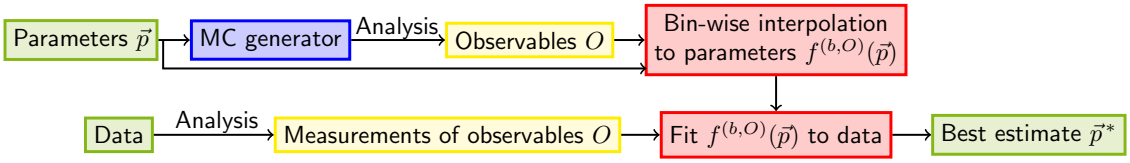


Figure 7.1: Schematic overview of parameter-based Monte Carlo generator tuning.

7.2 Data generation

As mentioned in Section 7.1, samples for some reference points in the parameter space are needed to obtain a parametrization of the Monte Carlo generator response. The samples are generated separately for the two different hadronization models using the `Herwig7` Monte Carlo generator. Per default, `Herwig7` uses the cluster hadronization model, referred to as the `Herwig7-H7` model in this work. Alternatively, `Herwig7` can be used with the `Pythia8` Lund string hadronization model using the `TheP8I` interface. This model will be referred to as `Herwig7-P8` model.

For both models, the tuning is performed using data from e^+e^- annihilation, which has the advantage of avoiding additional ambiguities from the choice of the parton distribution functions, which is inherent in proton-proton or proton-lepton collisions. Since the focus of this thesis is the tuning of the hadronization models of the Monte Carlo event generators, measurements that are sensitive to the hadronic final state modeling are chosen [109–113]. The `Rivet` [32, 45] framework provides

Table 7.1: Rivet codes of the analyses used for tuning with a short description of the contained observables and their references.

Rivet code	Type of observables	Reference
ALEPH_1996_S3486095	Event shapes, particle spectra	[109]
ALEPH_2001_S4656318	b quark fragmentation functions	[110]
DELPHI_1996_S3430090	Event shapes	[111]
JADE_OPAL_2000_S4300807	Event shapes, differential jet rates	[112]
PDG_HADRON_MULTIPLICITIES	Multiplicities	[113]

these measurements, and the corresponding `Rivet` analysis modules are listed in Table 7.1. The `Rivet` modules from ALEPH, DELPHI, JADE and OPAL provide event shape variables such as sphericity, thrust, aplanarity, etc. which are sensitive to AlphaQCD . This also applies to the differential jet rate from the analysis module by JADE and OPAL. Observables related to the multiplicities are obtained from the `Rivet` module by the PDG, while ALEPH provides the spectra of particles and the b quark fragmentation functions. Both these observables show sensitivities to AlphaQCD and fragmentation parameters, which will be discussed in more detail in Chapter 7.4.5.

In this thesis, the samples are generated using `Herwig` version 7.2.2 [19] with the `MENLOPS` method [114] using the `MadGraph5` matrix element generator [43] with the `OpenLoops` library [114]. This allows for the generation of $e^+e^- \to (Z/\gamma^*) \to 2, 3, 4, 5$ parton final states with the two-parton final states being generated at next-to-leading order (NLO) in perturbative QCD. For the usage of the Lund string hadronization model, the `ThePEG` interface is used with `Pythia 8.306`. The generation is performed at a center-of-mass energy of $\sqrt{s} = 91.2\,\text{GeV}$ using both the `Herwig7-H7` and `Herwig7-P8` hadronization models.

In order to choose the parameter values of the different hadronization models for each reference point, a parameter space is defined for each model. A lower and upper bound is defined, and the parameter values are chosen randomly from a uniform distribution within these bounds.

The parameter space for the `Herwig7-H7` model is shown in Table 7.2 together with the default values of the parameters, which are obtained from the default

tune settings for the cluster hadronization model provided in the source code of `Herwig7`. The ranges for the parameters are chosen to cover a wide range of physically meaningful parameters while fulfilling the constraints imposed by the models and Monte Carlo generator code. Hence, the ranges for most parameters are selected to be symmetric within a $\pm 50\%$ interval around the default value with some exceptions, which will be discussed in the following. The parameter AlphaQCD is chosen to be within a smaller range due to hardcoded minimum and maximum values in the Monte Carlo generator code of 0.1 and roughly 0.142, respectively. In a similar fashion, the parameters for the gluon constituent mass $m(g)$ and the strange quark constituent mass $m(s)$ are linked in the Monte Carlo generator, leading to constraints. In order to satisfy these constraints, the condition $m(g) > \frac{m(s)}{2}$ has to be fulfilled. A lower bound of $m(s) > m(u,d) = 0.35$ can be inferred as the strange quark constituent mass is expected to be larger than the constituent mass of the up and down quarks. Using these constraints, the parameter space for $m(s)$ is chosen to be expressed in terms of $m(g)$ with the lower bound of the fraction being set to the default ratio of $\frac{m(s)}{m(g)} = 0.47$ which imposes a lower bound of $m(g) > 0.74$.

Table 7.2: Parameters for the `Herwig7-H7` tune, their ranges and default values. See Table 3.1 for a description of the parameters.

Parameter	Range	Default
AlphaQCD	$[0.1000, 0.1417]$	0.1181
IRcutoff (GeV)	$[0.5004, 1.5012]$	1.0080
m(g) (GeV)	$[0.7445, 1.1400]$	0.9500
m(s) (GeV)	$[0.4734, 0.5000]m(g)$	0.4500
ClMax (GeV)	$[1.9334, 5.8000]$	3.8667
ClPow	$[0.8295, 2.4885]$	1.6590
ClSmr	$[0.1719, 0.8593]$	0.3437
PSplit	$[0.3450, 1.0348]$	0.6899

The parameter space for the `Herwig7-P8` model is shown in Table 7.3 with the default values for the parameters which are obtained from the Monash tune [3]. The ranges have been chosen in a similar way to the ranges of the `Herwig7-H7` model, with most ranges being determined by the allowed scope of the `Pythia8` framework. While the parameters aExtraDiQuark and aExtraSQuark have also been varied and fitted, a lack of sensitivity to these parameters has been observed. Hence, they have been fixed to their default values for the tuning.

In total, 500 randomly chosen parameter sets are selected for the `Herwig7-H7` model and 700 for the `Herwig7-P8` model. The additional samples for the `Pythia8` model

have been generated as a resampling due to the setting of the shower cutoff variable was necessary. The samples have been generated for each parameter set with 10^6 events.

Table 7.3: Parameters for the `Herwig7-P8` tune, their ranges and default values. The parameters marked as fixed are set to their default values for the tuning process. See Table 3.2 for a description of the parameters.

Parameter	Range	Fixed	Default
AlphaQCD	$[0.1000, 0.1417]$	x	0.1181
IRcutoff (GeV)	$[0.2002, 1.8014]$	x	1.0080
SigmaPT (GeV)	$[0.000, 1.000]$	x	0.335
aLund	$[0.20, 2.00]$	x	0.68
bLund (GeV^{-2})	$[0.00, 2.00]$	x	0.98
aExtraDiQuark	$[0.00, 2.00]$	✓	0.97
aExtraSQuark	$[0.00, 2.00]$	✓	0.00

YODA parsing

Using the settings from the previous section, samples can be generated for each reference point. The `Rivet` analysis modules are used to calculate the observables for each sample, and the results are stored in so-called YODA files[1]. These files contain the relevant information for each observable, such as the binning, the bin centers, the bin widths, the bin contents, and the bin errors in a standardized text format. In order to use the data from the YODA files, a parser has been implemented in the Julia programming language in the scope of this thesis, which is called YodaFiles.jl[2]. The YodaFiles.jl package reads the YODA files and stores the information in histograms provided by the StatsBase.jl[3] package, allowing these files to be used seamlessly in this framework.

7.3 Parametrization of observables

Using the samples generated with the parameters defined in the previous section, the parametrization of the observables can be performed. This parametrization is one of the crucial steps as it serves both as an interpolation of the Monte Carlo generator

[1] `https://gitlab.com/hepcedar/yoda/`
[2] `https://github.com/salvolc/YodaFiles.jl`
[3] `https://github.com/JuliaStats/StatsBase.jl`

response in the parameter space as well as a computationally cheaper alternative to the Monte Carlo generator. There are different approaches to solving such a parametrization problem. In this case, it is assumed that the change within a bin of an observable is a smooth function of the parameters. Hence, the parametrization is performed for each bin of each observable independently.

In this thesis, a multidimensional cubic polynomial is used to parametrize the observables:

$$f_b^{cubic}(\vec{\lambda}) = \alpha_0^{(b)} + \sum_i \beta_i^{(b)} \lambda_i + \sum_i \sum_{j \leq i} \gamma_{ij}^{(b)} \lambda_i \lambda_j + \sum_i \sum_{j \leq i} \sum_{k \leq j} \delta_{ijk}^{(b)} \lambda_i \lambda_j \lambda_k \tag{7.1}$$

Here, $\alpha_0^{(b)}$, $\beta_i^{(b)}$, $\gamma_{ij}^{(b)}$ and $\delta_{ijk}^{(b)}$ are the coefficients of the polynomial for the bin b of the observable with the parameters $\vec{\lambda}$. The total number of coefficients for a polynomial of order n in d dimensions is given by

$$N_d^{(n)} = 1 + \sum_{i=1}^{n} \frac{1}{i!} \prod_{j=0}^{i-1} (d + j)\,. \tag{7.2}$$

In order to determine the best values for the coefficients, the Julia package LsqFit.jl[4] is used. This package provides a least squares fitting routine that minimizes the χ^2 of the fit using the Marquardt–Levenberg algorithm [115, 116].

In addition to an estimate of the coefficients, the fitting routine also provides an estimate of the covariance matrix of the coefficients, which is used to estimate the uncertainty of the parametrization. The evaluation of the polynomial given the coefficients can be expressed as a linear equation system in the form of:

$$f_b^{cubic}(\vec{\lambda}) = \vec{c}^{\,T} \vec{\lambda} = \begin{pmatrix} 1 \\ \lambda_1 \\ \vdots \\ \lambda_1\lambda_2 \\ \vdots \\ \lambda_1^2 \\ \vdots \\ \lambda_1^2\lambda_2 \\ \vdots \\ \lambda_1^3 \\ \vdots \end{pmatrix}^T \cdot \begin{pmatrix} \alpha_0 \\ \beta_0 \\ \vdots \\ \gamma_0 \\ \vdots \\ \delta_0 \\ \vdots \end{pmatrix} \tag{7.3}$$

[4] `https://github.com/JuliaNLSolvers/LsqFit.jl`

Then, given the covariance matrix of the coefficients Σ, the uncertainty of the parametrization can be estimated by:

$$\sigma^2_{f_b^{cubic}}(\vec{\lambda}) = \vec{\lambda}^T \Sigma \vec{\lambda} = \begin{pmatrix} 1 \\ \lambda_1 \\ \vdots \\ \lambda_1\lambda_2 \\ \vdots \\ \lambda_1^2 \\ \vdots \\ \lambda_1^2\lambda_2 \\ \vdots \\ \lambda_1^3 \\ \vdots \end{pmatrix}^T \cdot \Sigma \cdot \begin{pmatrix} 1 \\ \lambda_1 \\ \vdots \\ \lambda_1\lambda_2 \\ \vdots \\ \lambda_1^2 \\ \vdots \\ \lambda_1^2\lambda_2 \\ \vdots \\ \lambda_1^3 \\ \vdots \end{pmatrix} \tag{7.4}$$

7.3.1 Comparison to Professors pseudoinverse method

An alternative approach to the parametrization using least squares is the *pseudoinverse method*, which has been implemented in the Professor package [2]. Using the notation of Equation 7.3, the fit problem for any individual bin can be written as:

$$\underbrace{\begin{pmatrix} f_0^b \\ f_1^b \\ f_2^b \\ \vdots \\ f_n^b \end{pmatrix}}_{\vec{y}} = \underbrace{\begin{pmatrix} 1 & \lambda_{1,1} & \cdots & \lambda_{1,1}\lambda_{2,1} & \cdots & \lambda_{1,1}^2 & \cdots & \lambda_{1,1}^2\lambda_{2,1} & \cdots & \lambda_{1,1}^3 \cdots \\ 1 & \lambda_{1,2} & \cdots & \lambda_{1,1}\lambda_{2,2} & \cdots & \lambda_{1,2}^2 & \cdots & \lambda_{1,2}^2\lambda_{2,2} & \cdots & \lambda_{1,2}^3 \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \lambda_{1,n} & \cdots & \lambda_{1,1}\lambda_{2,n} & \cdots & \lambda_{1,n}^2 & \cdots & \lambda_{1,n}^2\lambda_{2,n} & \cdots & \lambda_{1,n}^3 \cdots \end{pmatrix}}_{\Lambda} \cdot \underbrace{\begin{pmatrix} \alpha_0 \\ \beta_0 \\ \vdots \\ \gamma_0 \\ \vdots \\ \delta_0 \\ \vdots \end{pmatrix}}_{\vec{c}}$$

Here, $\vec{y}$ contains the bin contents for each MC sample while $\vec{c}$ contains the coefficients of the polynomial. The matrix Λ is constructed using the parameter sets $\vec{\lambda}_i$ of the corresponding MC sample and, as such, contains as many rows as there are MC samples which is equal to the dimensionality of $\vec{y}$. The first numerical subscript of Λ denotes the parameter index, while the second numerical subscript denotes the MC sample index. Given the degree of the polynomial n and the number of parameters d, the number of coefficients N is given by Equation (7.2). Hence, the matrix Λ has the dimensions $N \times N_{\mathrm{MC}}$. Using the equation above, the coefficients

can be obtained by solving the linear equation system:

$$\begin{pmatrix} \alpha_0 \\ \beta_0 \\ \vdots \\ \gamma_0 \\ \vdots \\ \delta_0 \\ \vdots \end{pmatrix} = \Lambda^{-1}\vec{y} \tag{7.5}$$

One way to solve this equation is to generate an equal number N of MC samples as there are coefficients and solve the equation using the inverse of the square $N \times N$ matrix Λ. While this approach is straightforward and produces an "exact" solution, it fails to reproduce the complexity of the Monte Carlo generator response, as additional samples would show deviations from this parametrization. Consequently, it is beneficial to use a larger number of MC samples and calculate coefficients that best describe the MC sample points in a least-squares sense. This can be achieved by using the pseudoinverse of the matrix Λ [117].

In order to compare the parametrization results of the least-squares method to the pseudoinverse method, both methods have been applied to the same set of MC samples. The resulting coefficients are then compared, calculating the norm of the relative difference of the coefficients:

$$\Delta c_i = \frac{\left|\vec{c}_i^{\,(\mathrm{lsq})} - \vec{c}_i^{\,(\mathrm{prof})}\right|}{\frac{1}{2}\left|\vec{c}_i^{\,(\mathrm{lsq})} + \vec{c}_i^{\,(\mathrm{prof})}\right|} \,. \tag{7.6}$$

Here, $\vec{c}_i^{\,(\mathrm{lsq})}$ and $\vec{c}_i^{\,(\mathrm{prof})}$ are the coefficients of the least-squares and the pseudoinverse method respectively for the i-th bin. As an example, this difference is calculated for each observable for the `Herwig7-H7` model. On average, the distance between the coefficients is $1.5 \cdot 10^{-5}$ with the highest difference being $1.1 \cdot 10^{-4}$. In order to translate the difference in the coefficients to a difference in the parametrization function, the parametrization is evaluated using the coefficients of both methods and the default parameter values. For the observable with the highest difference in the coefficients, the relative difference in the parametrization function is calculated to be 0.0049, which is negligible compared to the statistical uncertainty of the MC samples of around 3 to 4%.

In general, it can be concluded that both methods produce equivalent parametrization of the Monte Carlo generator response. However, for this thesis, the least-squares method is preferred over the pseudoinverse method as it allows for a more flexible choice of the parametrization function for future applications as well as a direct estimation of the uncertainty of the parametrization.

7.4 Evaluating the parametrization

In order to evaluate the goodness of the parametrization described in the previous sections, several tests are performed. During the parametrization, the χ^2_{red} of the fit is calculated for each parametrized bin. In addition, the pulls of the MC-generated points to the parametrization function are calculated. These two metrics can be used to determine bins that are difficult to parametrize. For the purpose of further evaluation, additional Monte Carlo samples were generated following a grid pattern in the parameter space. These samples are used to evaluate the parametrization for bins that suggest a poor performance following the other metrics while providing a visual representation of the parametrization function. Next, the parametrization is compared to the MC response and the data using the range of the parameter space. Lastly, the sensitivity of the parametrization to the parameters is evaluated using the *autodiff* functionality of the Julia programming language.

7.4.1 The reduced χ^2 statistics

One of the metrics used to evaluate the goodness of the parametrization is the reduced χ^2 statistics. It is calculated for each fitted bin using

$$\chi^2_{\text{red}} = \frac{1}{N_{\text{MC}} - N_{\text{coeff}}} \sum_{i=1}^{N_{\text{MC}}} \frac{(y_i - f_i)^2}{\sigma_i^2} \tag{7.7}$$

where N_{MC} is the number of MC samples, N_{coeff} is the number of coefficients of the polynomial. The sum runs over all MC samples i, where y_i is the bin value of the bin, f_i is the value of the parametrization function for that bin, and σ_i is the uncertainty of the corresponding MC sample i. It is expected for the χ^2_{red} statistics to be close to one for a good parametrization.

The distribution of the χ^2_{red} values for the `Herwig7-H7` model is shown in Figure 7.2. It shows a peak at $\chi^2_{\text{red}} \approx 1$ with a tail towards higher values. While this behavior suggests an acceptable parametrization for a large number of bins, the tail towards higher values indicates that there are bins that are difficult to parametrize. In order to further identify these bins and benchmark their performance, further metrics are applied in the following.

7.4.2 The pull distribution of the parametrization

In addition to the χ^2_{red} statistics, the pulls of the MC-generated points to the parametrization function are analyzed. The pulls are calculated for each MC sample

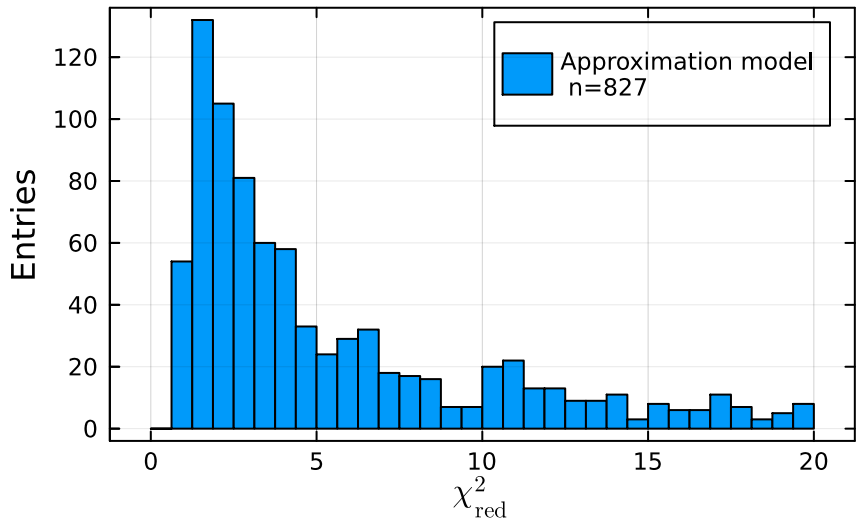


Figure 7.2: Distribution of the χ^2_{red} values between the parametrization and the MC samples of the `Herwig7-H7` model. Each fitted bin contributes one value to the histogram.

in each parameterized bin using

$$\mathrm{p} = \frac{f(\lambda) - y_{\text{MC}}}{\sigma_r} \tag{7.8}$$

where $f(\lambda)$ is the parametrization function for the bin, y_{MC} is the bin value of the MC sample. σ_r is the expected uncertainty of the residual which is calculated using

$$\sigma_r = \sqrt{\sigma^2_{\text{MC}} - \sigma^2_{\text{fit}}} \tag{7.9}$$

with the uncertainty of the MC prediction σ_{MC} and the uncertainty of the fit model σ_{fit}. It should be noted that the uncertainty of the fit model can be larger than the uncertainty of the MC prediction, resulting in a negative value for σ^2_r. This can be caused by multiple effects, such as the fit model not being able to fully describe the shape of the MC prediction and the uncertainties of the MC prediction being underestimated. For these cases, the pulls are discarded, however, these bins are considered for further evaluation as they might indicate areas that are difficult to parameterize. The distributions of the resulting pulls for the `Herwig7-H7` and `Herwig7-P8` models are shown in Figure 7.3. In general, the shape of the distribution follows a normal distribution, however, the width of the distribution is larger than expected. In an ideal case, the distribution would be centered around

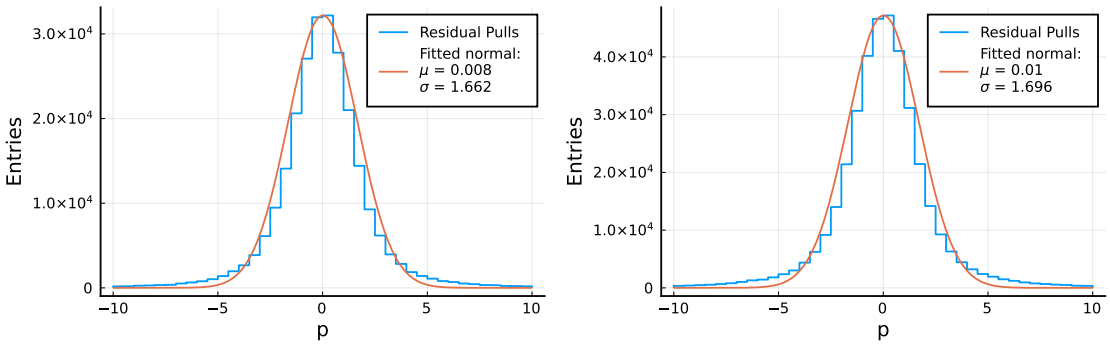


Figure 7.3: Distribution of the pulls between the parametrization and the MC samples of the `Herwig7-H7` model on the left and for the `Herwig7-P8` model on the right. Each fitted bin contributes one value to the histogram.

zero with a width of one. In order to determine the mean and standard deviation of the distribution, a Gaussian fit is performed. The fit is represented in Figure 7.3 by the orange line. The `Herwig7-H7` model shows a mean of -0.008 and a standard deviation of 1.662 while the `Herwig7-P8` model shows a mean of -0.01 and a standard deviation of 1.696. While both models show a mean close to zero, the standard deviation is larger in both cases. This indicates that the fitted models do not give a perfect description of the MC samples which leads to deviations that are larger than expected from statistical fluctuations.

7.4.3 Grid reevaluations

Since the parametrizations of the `Herwig7-H7` and `Herwig7-P8` models are based on a random sample of points in the parameter space, which is 8 and 5 dimensional, respectively, it is challenging to visualize the parametrization function. In order to obtain a visual representation of the parametrization function, additional samples are generated following a grid pattern in the parameter space. Hence, only one parameter is varied at a time in eleven equidistant steps while the other parameters are fixed to their default values according to Tables 7.2 and 7.3. The resulting samples are then used to evaluate the parametrization function for each bin. As an example, the first bin of the sphericity variable from the `DELPHI_1996_S3430090` analysis module is shown in Figure 7.4. The black markers indicate the MC samples and their statistical uncertainties, while the red line represents the parametrization function. The red band represents the uncertainty of the parametrization function, which is propagated from the uncertainty of the fitted coefficients following Equation 7.4. The blue band represents the range of values of the fitted model obtained by shifting the default values of the parameters by $\pm 5\%$. This is done to account for an

unfavorable choice of the default values of the parameters, which might lead to systematic deviations between the parametrization function and the MC samples. Generally, the parametrization function is able to describe the MC samples within the described uncertainties.

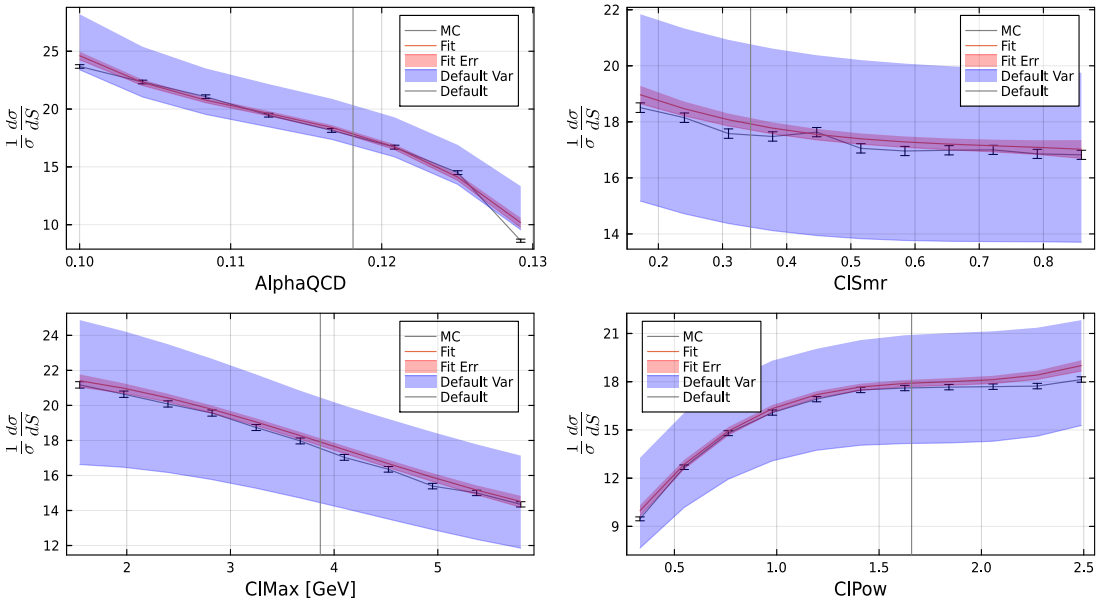


Figure 7.4: The content of the first bin of the sphericity distribution from the `Rivet` module DELPHI_1996_S3430090 as a function of `AlphaQCD`, `ClSmr`, `ClMax` and `ClPow` with parameter sets from the grid samples. The parametrized model is shown in red in comparison to the MC grid sample predictions. The red band represents the uncertainty on the fitted model obtained by propagating the uncertainty of the fitted coefficients. The blue band represents variations caused by shifting the default values for the evaluation of the parametrized model by $\pm 5\%$.

The grid samples are also used to evaluate the goodness of the fit by calculating the p-values of the χ^2 statistics for each fitted bin. These p-values are shown in Figure 7.5 for the `Herwig7-H7` and `Herwig7-P8` models. In both cases, the p-values are distributed mostly towards lower values, with most bins showing p-values below 0.25 and about 30% of the bins resulting in p-values below 0.05. It should be noted that the χ^2 test is performed without taking into account systematic biases due to the choice of the default values of the parameters. Furthermore, the number of degrees of freedom can only be approximated due to the number of free parameters being dependent on the correlation of the parameters. For these tests, the number of degrees of freedom is estimated by $N_{\mathrm{MC}} - 4$ as the projection of the polynomial into one dimension should result in four degrees of freedom for the model. However, as the fit is performed in multiple dimensions using a different set of MC samples,

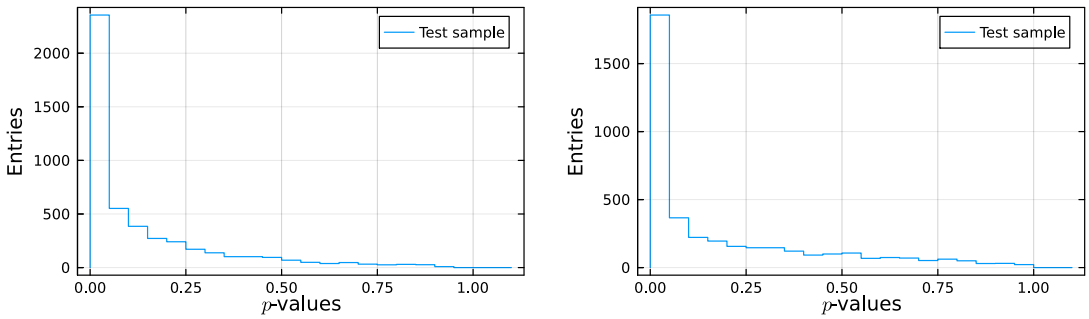


Figure 7.5: The p-values obtained from performing χ^2 test between the prediction of the parameterized model and the MC test samples, which were generated in a grid. The left plot shows the p-values for the `Herwig7-H7` model, while the right plot shows the p-values for the `Herwig7-P8` model. The test is performed independently for each bin of each observable and for each parameter of the model.

the number of degrees of freedom can vary. Nonetheless, bins with p-values below 0.05 are considered to be more difficult to parametrize and are hence considered for further evaluation.

In addition to χ^2 test and visualization purposes, the grid samples are used to evaluate the goodness of fit by calculating the pulls of the MC samples to the parametrization function in a similar fashion to the pulls described in Section 7.4.2. However, as the uncertainty of the grid samples is uncorrelated to the uncertainty of the parametrization function, as they were not used in the fitting process, the equation for the pulls is modified to:

$$\mathrm{p} = \frac{f(\lambda) - y_{\mathrm{MC}}}{\sqrt{\sigma^2_{\mathrm{MC}} + \sigma^2_{\mathrm{fit}}}}\,. \tag{7.10}$$

The distribution of the pulls for the `Herwig7-H7` and `Herwig7-P8` models are shown in Figure 7.6. Both distributions closely follow a normal distribution with means of -0.003 and -0.006 and standard deviations of 1.182 and 1.254 for the `Herwig7-H7` and `Herwig7-P8` models, respectively. Similar to the pulls of the MC samples used for the fitting, the standard deviations are larger than one, indicating a larger deviation of the model than expected from statistical fluctuations. However, the distribution is closer to a unit normal distribution, indicating an overall good performance of the parametrization function.

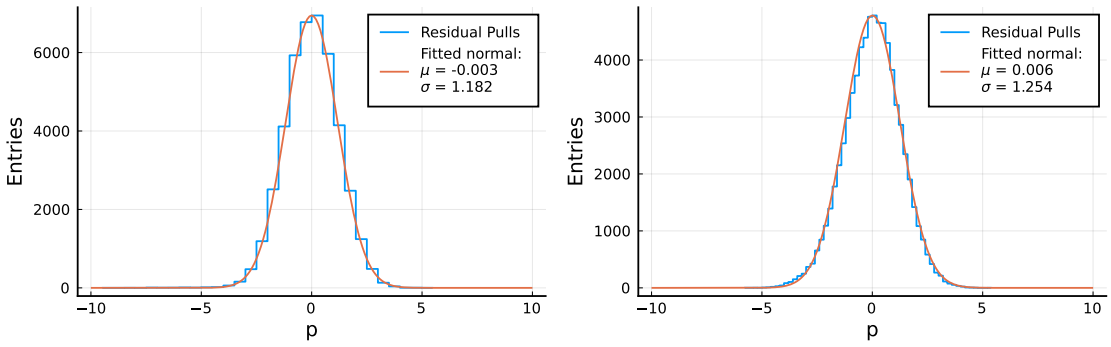


Figure 7.6: Distribution of the pulls between the parametrization and the MC grid samples of the `Herwig7-H7` model on the left and for the `Herwig7-P8` model on the right. Each fitted bin contributes one value to the histogram.

Combination of goodness of fit metrics

As described in the previous sections, various metrics are used to evaluate the goodness of the fit. The χ^2_{red} statistics and the pulls of the MC samples to the parametrization function are calculated during the fit. In addition, grid samples that are not used for the fit are used for χ^2 tests and calculating pulls. These tests vary in their sensitivity to different aspects of the fit, however, as a general statement, it can be said that a large amount of the bins show a good performance of the parametrization function while a significant amount of bins show a poor performance in at least one of the metrics.
In order to further quantify the performance of bins that show a poor performance in at least one of the metrics, these bins were collected and visualized using the grid samples similar to Figure 7.4. This allows for a visual inspection of the parametrization function on difficult-to-model bins. The check reveals that some of the bins are systematically under or overestimated by the parametrization model, while the shape of the fitted function represents the MC samples well. For a minority of bins, the parametrization function is not able to fully describe the shape of the MC samples. However, in all of these instances, the deviations are well within the ranges covered by the variation of the default values of the parameters. As such, the parametrization model is considered to describe the Monte Carlo generator response reasonably well for the purpose of this thesis. However, it should be noted that while the focus of this thesis is on the tuning process itself, a high-precision tune should address the parametrization in more detail and consider different options and techniques to improve the parametrization.

7.4.4 Further parametrization checks

While the tests described in the previous sections test the parametrization model against the MC samples, it is also important to test the parametrization model against the data. Hence, the ranges provided by the parametrization functions are compared to the data and the MC samples using the full range of the parameter space. This test allows to check if the parametrization is flexible enough to describe the data points given the bounds of the parameter space. For each bin of each observable b, the maximum and minimum values of the corresponding parametrization function f_b are calculated using

$$\max(f_b) = \max_{\vec{\lambda}_i} \left(f_b(\vec{\lambda}_i) \right) \quad \text{and} \quad \min(f_b) = \min_{\vec{\lambda}_i} \left(f_b(\vec{\lambda}_i) \right) \tag{7.11}$$

where $\vec{\lambda}_i$ are the parameter sets of each MC sample i. The ranges for the MC prediction are simply obtained by the minimum and maximum values reached by the MC samples for each bin. Since the goal of MC tuning is to obtain a parameter set that describes the data as well as possible, the data points should be contained within the ranges of the parametrization function. In addition, the overlap of

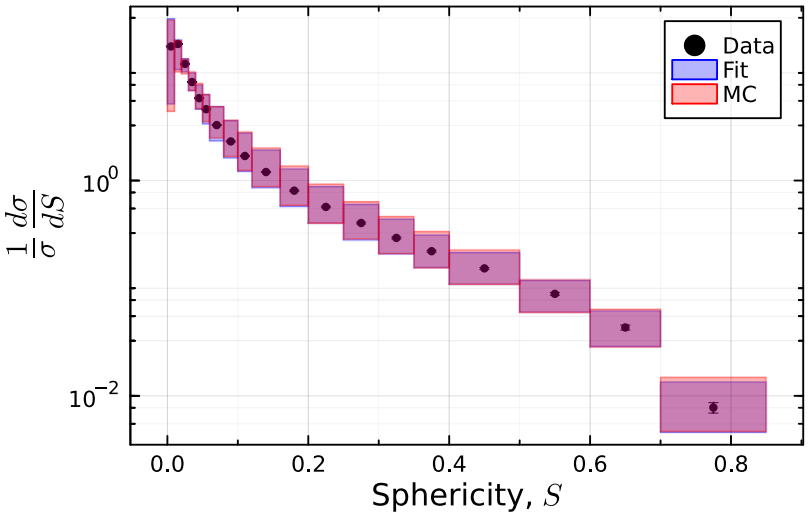


Figure 7.7: Distribution of the sphericity variable from DELPHI [111]. The red and blue bands represent the minimum and maximum ranges of the MC samples and the parametrization, respectively. The black data points include statistical and systematic uncertainties provided by DELPHI using the `Rivet` framework.

the ranges of the MC samples with the ranges of the parametrization function can be used as an indicator of the quality of the parametrization. As an example, the distribution of the sphericity variable is shown in Figure 7.7 for the `Herwig7-H7`

model. The red and blue bands represent the ranges of the MC samples and of the parametrization, respectively. Notably, these areas largely overlap, suggesting that the fitted model and the MC samples cover a similar range of values for the observable within the parameter space. In addition, it is shown that the data points are contained within the ranges of the parametrization function. This indicates that the parametrization function is sufficiently flexible to account for the data points, at least within individual bins, based on the confines of the parameter space.

7.4.5 Parameter sensitivity to observables

In addition to the goodness of the fit, the sensitivity of the parametrization to the parameters is evaluated. This is done by calculating the partial derivatives of the parametrization function with respect to the parameters. The partial derivatives are calculated using the autodiff functionality of the Julia programming language with the ForwardDiff.jl package [76]. This method is compared to the partial derivatives calculated using the finite differences with the available MC samples on the grid. These derivatives are used to create a sensitivity score according to Ref. [111] using

$$S_i^b = \frac{\delta MC_b(\vec{p})}{MC_b(\vec{p})}\bigg|_{p_i} \left(\frac{\delta p_i}{p_i}\right)^{-1} \approx \frac{\partial MC_b(p)}{\partial|\ln(p_i)|}\bigg|_{p_i} . \tag{7.12}$$

Here, MC_b is the parametrization or MC sample value for the bin b, and p_i is the parameter value. As an example, the sensitivity scores of the sphericity variable are shown in Figure 7.8 for the `Herwig7-H7` model. The top plot shows the sensitivity calculated using the parametrization function with automatic differentiation, while the bottom plot shows the sensitivity calculated using the MC samples generated on a grid with finite differences. In both cases, the sensitivity follows a similar pattern. However, due to the continuous nature of the parametrization function, the sensitivity is smoother compared to the one using the MC samples. Overall, the sensitivity is largest for the AlphaQCD parameter followed by the constituent mass parameters. The parameters related to the cluster hadronization as well as IRCutoff and PSplit show a smaller sensitivity. To better quantify the sensitivities of the parameters, the score is averaged over all bins of each observable. The full results are shown in Tables A.4-A.9 in Appendix A. Most observables show sensitivities of the parameters that are in agreement with the discussion in Section 7.2. Overall, it is observed that the sensitivity of the parameters is largest for the AlphaQCD parameter followed by the constituent mass parameters. While other parameters show a smaller sensitivity score overall, they can still show a large sensitivity for specific observables, such as the $\mathrm{Cl}_{\mathrm{max}}$ parameter for certain multiplicity and particle spectra observables. In addition, it can be seen that the sensitivity of the parameters $\mathrm{a}_{\mathrm{ExS}}$ and $\mathrm{a}_{\mathrm{ExDi}}$ is very small for most observables for the `Herwig7-P8`

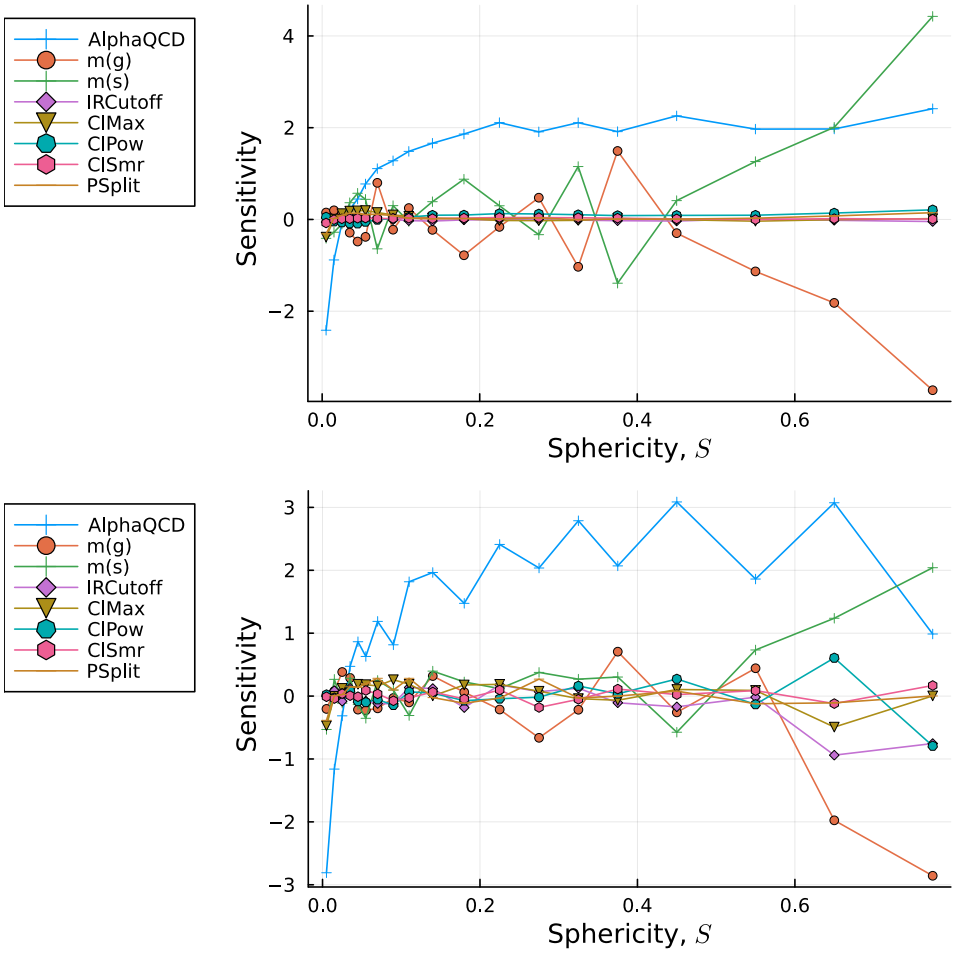


Figure 7.8: Sensitivity of the parameters for the `Herwig7-H7` model with respect to the sphericity observable. The top plot is calculated using the parametrization, while the bottom plot uses the MC samples generated on a grid.

model. This effect was also observed in tunes of preliminary studies, which led to the decision to fix these parameters to their default values. In summary, all of the parameters exhibit sensitivity to at least a few different observables. This suggests that the choice of observables for the given set of parameters is reasonable for performing a tune.

7.5 The EFTfitter.jl library

Using the parametrization of the Monte Carlo generator response described in the previous sections together with the data, the statistical model used for the tuning is

constructed. For this purpose, the EFTfitter.jl framework is used [15]. This software package is written in the Julia programming language and serves as a framework for constructing statistical models in a Bayesian context. It is designed for tasks like fitting effective field theory models and combining measurements, however, the likelihood function is generally applicable to all cases where the underlying uncertainties are described by a normal distribution. Thus, it is used in this thesis to construct the statistical model for the tuning of the MC generator parameters. The likelihood function is a multivariate Gaussian in the form of:

$$\ln L(\vec{D}|\vec{\lambda}) = -\frac{1}{2}[\vec{D} - \vec{f}(\vec{\lambda})]^T \cdot M^{-1} \cdot [\vec{D} - \vec{f}(\vec{\lambda})] \, . \tag{7.13}$$

The vector $\vec{D}$ contains the data points ordered by the bins of the observables while $\vec{f}(\vec{\lambda})$ contains the parametrization functions for each bin ordered corresponding to the data vector evaluated at the parameter set $\vec{\lambda}$. The matrix M is the covariance matrix of the data points, which is constructed using the statistical and systematic uncertainties of the data points provided by the `Rivet` framework. In order to construct the covariance matrix M, the correlation between the data points is needed. Since these values are not provided by the `Rivet` framework, the correlation is assumed to be zero. However, studies on the impact of the correlation on the tuning results were performed and are described in Section 7.8. Using the parametrization functions, the data, and the covariance matrix, the EFTfitter framework creates the likelihood function, which can be interfaced with the BAT.jl framework, where the posterior can be generated by adding a prior for the parameters.

7.6 MC tuning results using EFTfitter

Using the EFTfitter framework described in the previous section, the tuning of the `Herwig7-H7` and `Herwig7-P8` models is performed. The prior distribution of the parameters is chosen to be a uniform distribution within the range of the parameter space defined in Tables 7.2 and 7.3. However, in the case of the `Herwig7-H7` model, the parameters $m(g)$ and $m(s)$ are dependent on each other with the condition $m(g) > \frac{m(s)}{2}$. Hence, without any additional constraints, this could result in sample values that lead to an invalid parameter set. In order to avoid this, the BAT.jl framework allows for the usage of so-called *hierarchical priors* that guide the sampling process by defining priors as a function of the other parameters. In this case, the prior for $m(s)$ is defined to be a uniform distribution with a lower limit of 0.28, which results from the lowest possible value of $m(g)$ and an upper limit of $\frac{m(g)}{2}$. Using these priors and the likelihood function, the posterior distribution of the parameters is sampled using the BAT.jl framework. For both models, the

sampling is performed using the Metropolis-Hastings algorithm by running six chains with 10^6 steps. It should be noted that the distance R for the convergence of the Brooks-Gelman test is loosened to 1.3 from its default value of 1.1 to account for the larger number of chains. All other parameters for the sampling are set to their default values.

As a result, the global mode of the posterior distribution is used as the tuned parameter set. Furthermore, the marginal mode, as well as the 68% intervals of the one-dimensional marginalized posterior distributions, are shown as the individual parameter estimates. For the purpose of evaluating the goodness of the tune, an additional sample is created using the tuned parameter set. Then, for each observable, the p value of the χ^2 test between the data and the MC sample is calculated. This procedure is repeated for the so-called nominal MC sample, which uses the default parameter set of the model. The resulting p values can then be compared with higher values, indicating a better description of the data.

7.6.1 Tuning results for the `Herwig7-H7` model

The global mode, as well as the marginalized mode and the 68% intervals of the posterior distribution for the tuning of the `Herwig7-H7` model, are shown in Table 7.4. For most parameters, the global mode is within the 68% interval of the marginalized posterior distribution. The largest deviation is observed for the parameter $\mathrm{Cl}_{\mathrm{max}}$ where the global mode is outside of the 68% interval but within the 95% interval. This effect can be explained by the correlation of the parameters, as can be seen in the plot of the posterior distribution in the following. When comparing the default values of the parameters to the intervals of the tuned parameters, it can be seen that these values are in agreement with each other, considering the uncertainties for all parameters except $\mathrm{Cl}_{\mathrm{smr}}$. This parameter has a default value of 0.3437, which is within the 95% interval of the tuned parameter.

The posterior distribution for the tuning of the `Herwig7-H7` model is shown in Figure 7.11. The marginalized posterior distributions of the parameters are shown in the diagonal of the plot, while the two-dimensional posterior distributions are shown in the off-diagonal. The green, yellow, and red areas contain the smallest 68, 95, and 99% intervals of the marginalized probability distributions, respectively. At first glance, it is apparent that none of the one-dimensional distributions of the parameters follow a normal distribution. The distributions of the parameters AlphaQCD , IRCutoff , and PSplit have a slight asymmetry, while the parameters $\mathrm{Cl}_{\mathrm{max}}$ and $\mathrm{Cl}_{\mathrm{smr}}$ show a more pronounced asymmetry in their distributions. The asymmetry is even more pronounced in the distributions of the constituent masses $m(g)$ and $m(s)$ that are mostly one-sided, with a sharp increase at lower values and a slow drop-off towards higher values. Most parameter distributions display

Table 7.4: Tuning results for the `Herwig7-H7` model. Values of the global and marginalized mode of the posterior samples, along with the smallest intervals containing 68% of the probability, are presented.

Parameter	Global mode	Marginal mode	Smallest 68% interval
AlphaQCD	0.115	0.115	[0.112, 0.118]
IRCutoff (GeV)	0.879	0.755	[0.580, 1.020]
$m(g)$ (GeV)	0.709	0.738	[0.700, 0.955]
$m(s)$ (GeV)	0.353	0.375	[0.346, 0.470]
ClMax (GeV)	2.591	4.025	[3.200, 4.750]
ClPow	0.823	0.910	[0.740, 1.260], [1.540, 2.260]
ClSmr	0.675	0.725	[0.480, 0.885]
PSplit	0.868	0.728	[0.615, 0.865]

a single peak with the exception of $\mathrm{Cl}_{\mathrm{pow}}$, which shows a multimodal behavior. The first peak of the distribution corresponds with the global mode of the posterior, while the second peak is located closer to the default parameter set. The two-dimensional distributions show a weak correlation between most parameters. The most notable exceptions are the parameters $m(g)$ and $m(s)$, which show a very strong correlation, and the distribution between $\mathrm{Cl}_{\mathrm{max}}$ and $\mathrm{Cl}_{\mathrm{pow}}$, which has a highly non-gaussian shape with a strong correlation. This correlation, together with the multimodal $\mathrm{Cl}_{\mathrm{pow}}$ distribution, can explain the deviation of the global mode of the parameter $\mathrm{Cl}_{\mathrm{max}}$ from the 68% interval of its marginalized posterior distribution. As the first peak of $\mathrm{Cl}_{\mathrm{pow}}$ distribution corresponds to the global mode, it shifts the global mode of $\mathrm{Cl}_{\mathrm{max}}$ towards lower values, resulting in the observed deviation. The full correlation matrix of the parameters can be found in Figure A.1 in the Appendix A. Using the parameter set defined by the global mode of the posterior distribution, an additional MC sample is generated in order to compare the tuned model to the nominal model.

The resulting p-values of the χ^2 test between the data and the MC samples are shown in Figure 7.9. It should be noted that the p-values are generally lower than expected from a good description of the data. This could be caused by multiple effects, such as an underestimation of the uncertainties of the MC prediction or missing parameters of the model, which require tuning. However, since the p-values are used to compare two different models, the absolute value is less crucial than the difference between the p-values of the nominal and tuned model. In this case, the p-values of the tuned model are generally higher than the p-values of the nominal model, increasing from an average of 0.095 to an average value of 0.133. Since

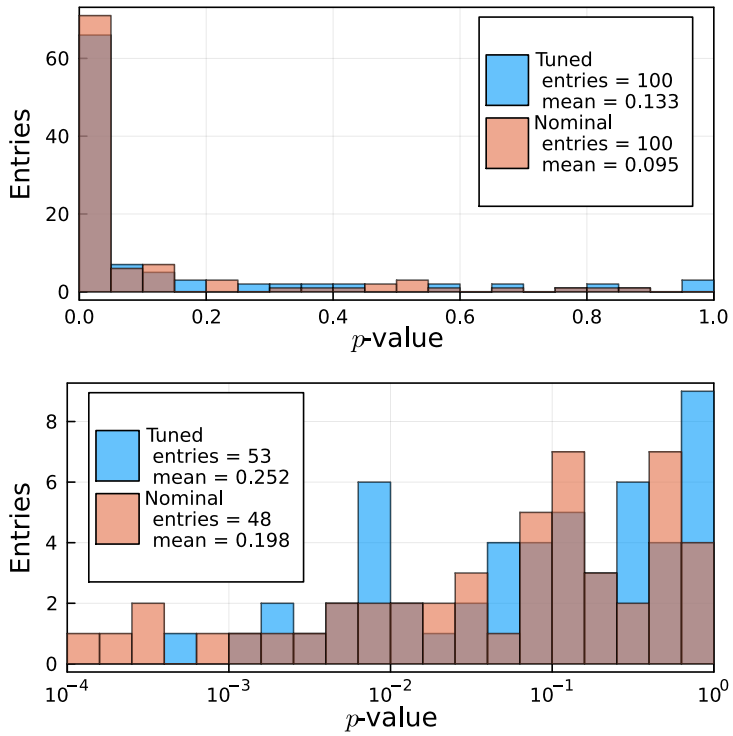


Figure 7.9: Histogram of the p-values for the χ^2 test between data and MC predictions for each observable using the nominal and tuned parameter sets with the `Herwig7-H7` model. The bottom histogram shows the p-values in a logarithmic scale for $p > 10^{-4}$.

the scale of the p-values is generally low, it is useful to compare the p-values on a logarithmic scale which is shown on the bottom of Figure 7.9. For better visualization p-values below 10^{-4} are cut from these distributions. Using the logarithmic plot, the improvement of the p-values can be seen more clearly, together with a reduction of the number of observables with p-values below 10^{-4} for the tuned MC sample.

To illustrate the effect of the tuning process, the distributions of two observables are shown in Figure 7.10 as an example. The left plot shows the distribution of the sphericity variable while the right plot shows the distribution of the B_u^+ multiplicity for both the tuned and nominal MC samples, including the data points. The uncertainty band for the nominal MC sample contains the statistical uncertainty while the uncertainty band for the tuned MC sample contains both the statistical uncertainties as well as a so-called *tuning uncertainty*. This tuning uncertainty is calculated by propagating the posterior distribution of the parameters through the parametrization model and is further explored in Section 7.7. These two sources are added in quadrature to obtain the total uncertainty of the tuned MC sample. Hence, the uncertainty bands of the tuned MC sample are generally larger than

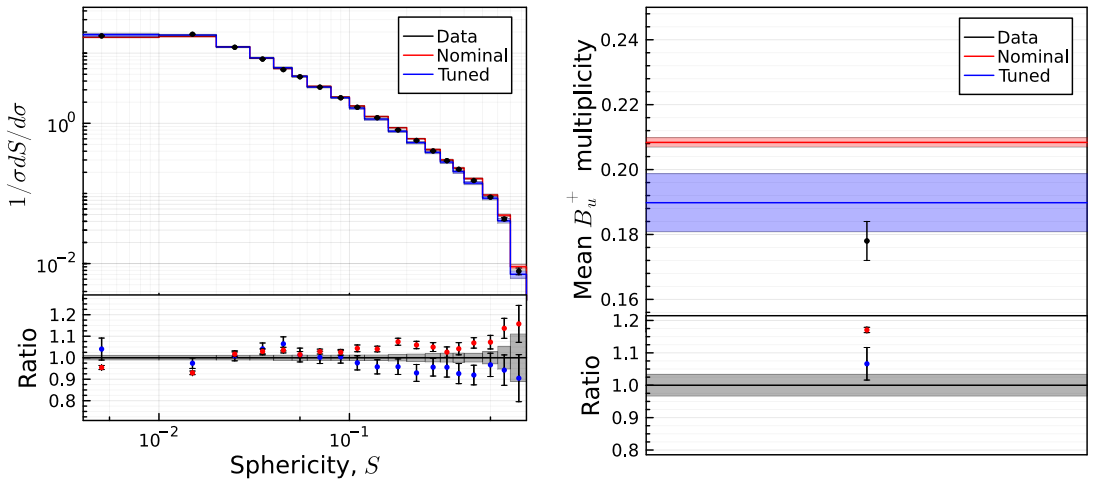


Figure 7.10: Distribution of the sphericity from DELPHI [111] on the left and the B_u^+ multiplicity [113] on the right. The black data points include statistical and systematic uncertainties provided by the `Rivet` framework. The uncertainty band for the nominal MC sample contains the statistical uncertainty while the uncertainty band for the tuned MC sample contains both the statistical uncertainties as well as a so-called *tuning uncertainty*. The bottom panels show the ratio of the data points to the MC samples. The samples were generated using the `Herwig7-H7` model.

the ones of the nominal MC sample. Overall, it is observed that the tuned MC sample is able to describe the data points equally or better than the nominal MC sample. This is a trend in the majority of the other observables, as can be inferred by the distribution of the p-values. It should also be noted that the fluctuations of the tuned MC sample seem to be described more accurately by the added tuning uncertainty, as most data points are contained within the uncertainty band of the tuned MC sample.

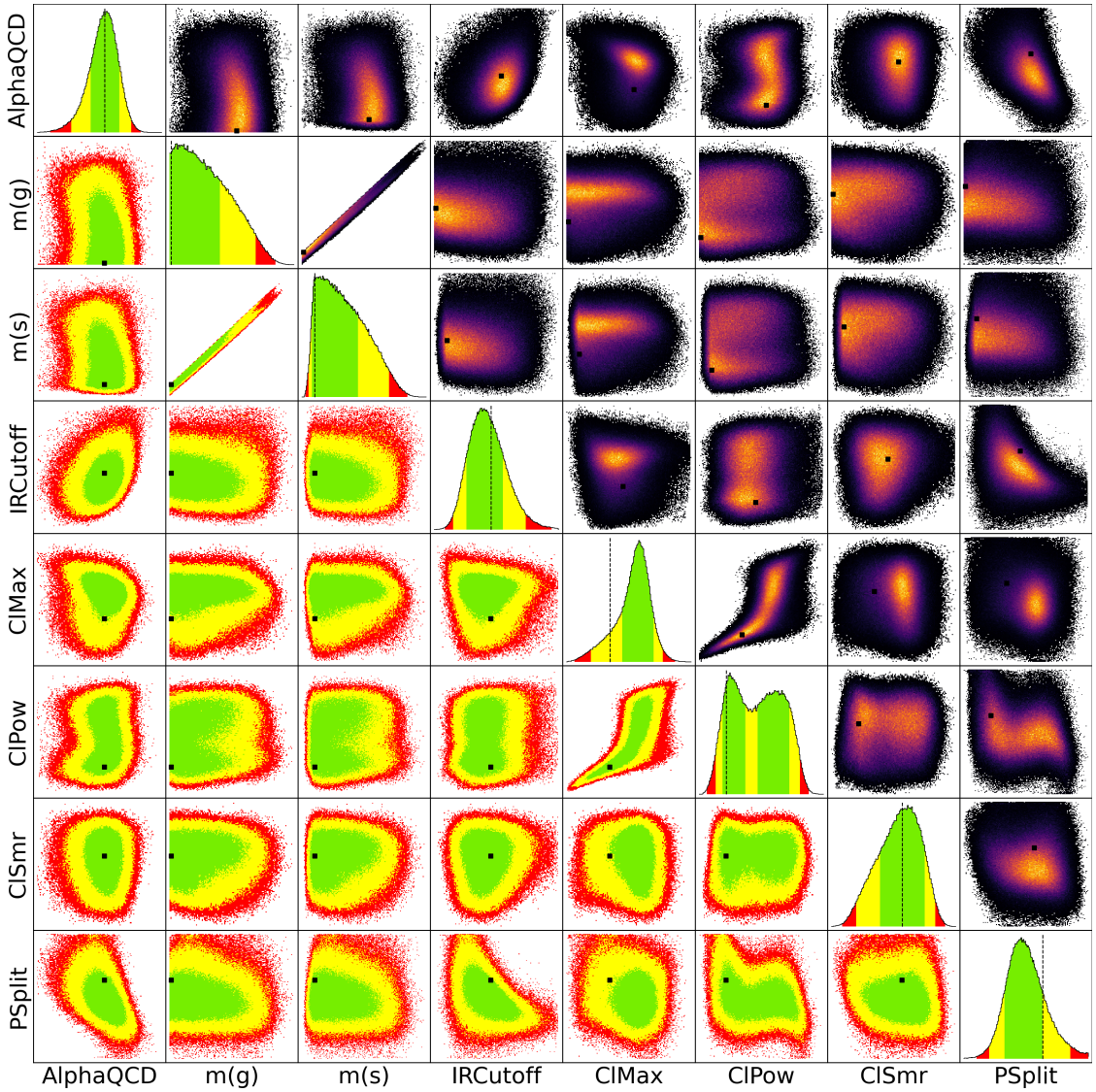


Figure 7.11: The one and two-dimensional marginalized posterior distributions of the parameters for the tune of the `Herwig7-H7` model. The one-dimensional marginals are shown in the diagonal plots, while the two-dimensional marginals are shown in the lower triangle plots. The green, yellow, and red areas contain the smallest 68, 95, and 99% intervals of the marginalized probability distributions, respectively. The heat maps of the two-dimensional marginals are shown in the upper triangle plots. The dashed lines and dots represent the position of the global mode, which is the point with the highest probability.

7.6.2 Tuning results for the Herwig7-P8 model

The tuning of the Herwig7-P8 model is performed in a similar fashion to the Herwig7-H7 model. The values for the global mode, the marginal modes, and the 68% intervals of the posterior distribution are listed in Table 7.5.

Table 7.5: Tuning results for the Herwig7-P8 model. Values of the global and marginalized mode of the posterior samples, along with the smallest intervals containing 68% of the probability, are presented. The fixed values are set to their default values for the tuning process.

Parameter	Global mode	Marginal mode	Smallest 68% interval	Fixed
AlphaQCD	0.120	0.120	[0.117, 0.122]	x
IRCutOff (GeV)	1.079	1.079	[0.730, 1.390]	x
SigmaPT (GeV)	0.303	0.311	[0.284, 0.336]	x
aLund	1.287	1.435	[0.950, 1.760]	x
bLund (GeV^{-2})	1.302	1.325	[0.940, 1.720]	x
aExtraDiquark	0.970	0.970	-	✓
aExtraSQuark	0.0	0.0	-	✓

For the AlphaQCD, IRCutoff, StringPT, and bLund parameters the default values are within the 68% interval of the posterior distribution. The only exception is the parameter aLund , which has a significantly lower default value of 0.68 compared to the marginal mode value of 1.435. However, the default value is still within the 95% interval of the posterior distribution.

The distributions of the one and two-dimensional marginalized posterior of the tuned parameters of the Herwig7-P8 model are shown in Figure 7.12. Similar to the results of the Herwig7-H7 model, the parameters are not distributed according to normal distributions. However, in contrast to the Herwig7-H7 model, the distributions show an overall unimodal behaviour. The parameters AlphaQCD, IRCutoff and StringPT seem to be well constrained by the data as the distributions are sufficiently narrow to be contained within the prior boundaries. Also, the correlation between these parameters is relatively weak, with only the parameters AlphaQCD and StringPT showing a medium correlation. By comparison, the parameters aLund and bLund are strongly correlated with each other. This is expected, as the two parameters both influence the probability distribution of the momentum transfer, see Equation 3.5 in Section 3.3.2, in a similar fashion. Furthermore, compared to the other parameters, aLund and bLund are constrained rather weakly as their distribution has a larger width, which results in a cut-off of the probability distribution at the prior boundary towards higher values.

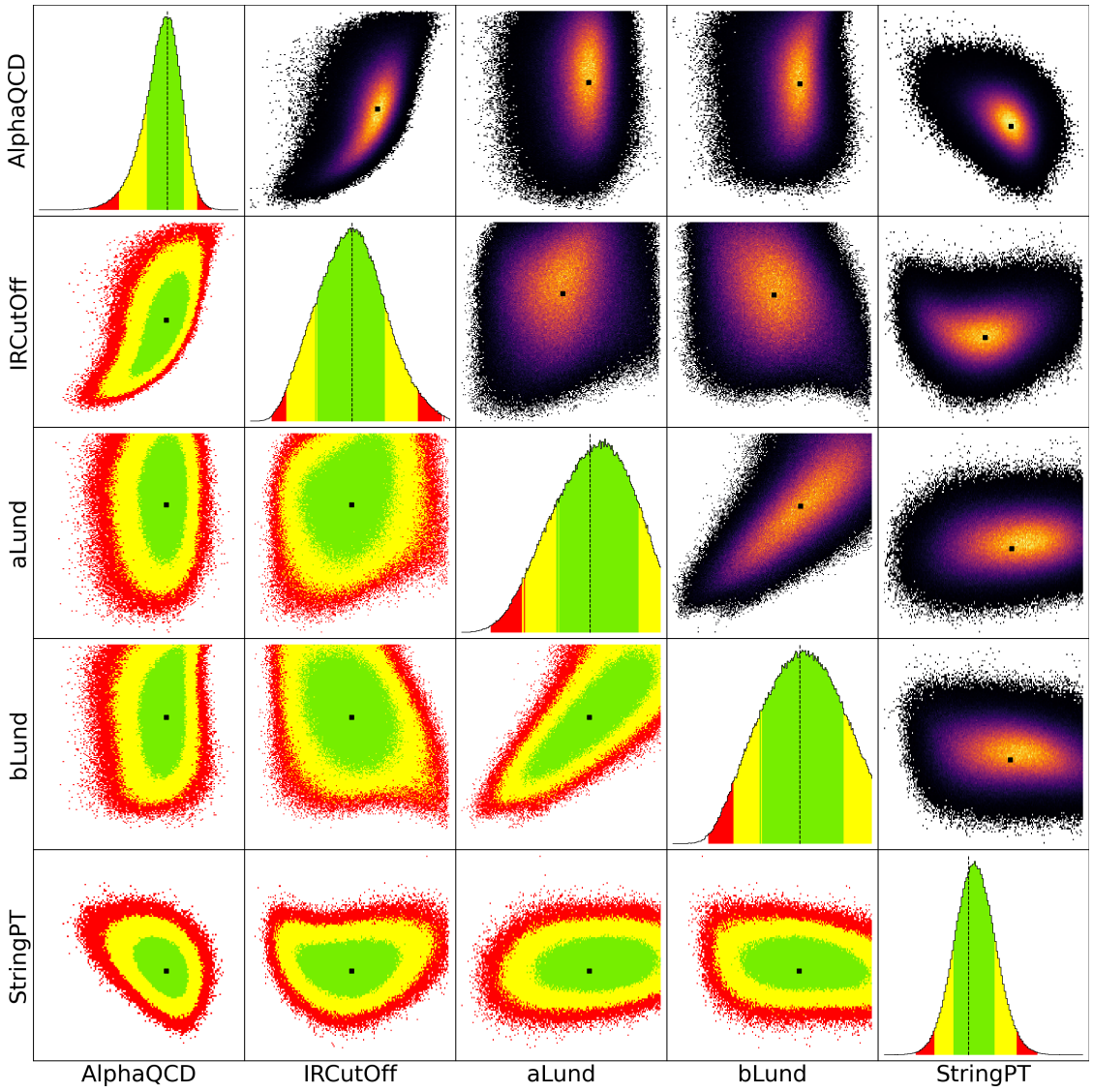


Figure 7.12: The one and two-dimensional marginalized posterior distributions of the parameters for the tune of the `Herwig7-P8` model. The green, yellow, and red areas contain the smallest 68, 95, and 99% intervals of the marginalized probability distributions, respectively. The dashed lines and dots represent the position of the global mode, which is the point with the highest probability.

While it is possible to increase the prior ranges to avoid such a cut-off, the points outside of the prior range would result in invalid parameter sets for the `Herwig7-P8` model. The full correlation matrix of the parameters can be found in Figure A.2 in the Appendix A.

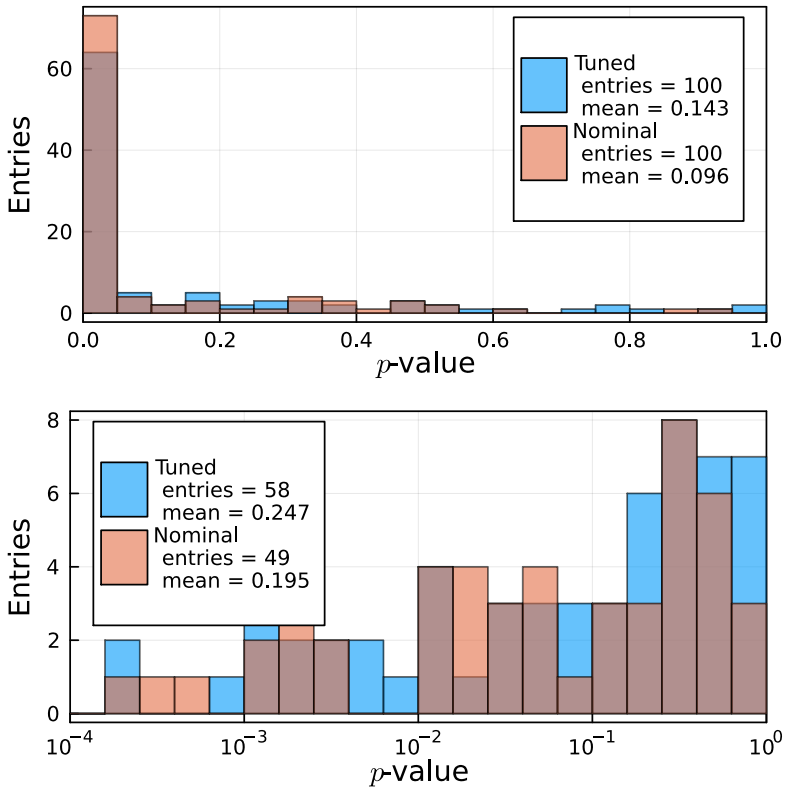


Figure 7.13: Histogram of the p-values for the χ^2 test between data and the MC predictions for each observable using the nominal and tuned parameter sets with the `Herwig7-P8` model. The bottom histogram shows the p-values in a logarithmic scale for $p > 10^{-4}$.

Using the parameter set defined by the global mode of the posterior distribution, an additional MC sample is generated in order to compare the tuned model to the nominal model analogously to the evaluation of the tune of the `Herwig7-H7` model. The p-values of the comparison between the data and the MC samples are shown in Figure 7.13. Again, an increase in the average p-value from 0.096 to 0.143 is observed together with a lower number of observables with p-values below 10^{-4} for the tuned MC sample. Both these observations indicate an improvement in the description of the data by the tuned MC sample.

The distributions of the sphericity and the B_u^+ multiplicity are shown in Figure 7.14 as an example of the effect of the tune. The uncertainties are treated in the same

way as for the `Herwig7-H7` model with the tuning uncertainty being added for the tuned MC sample. In this case, while the tuned MC sample shows an increased agreement with the data points for the sphericity observable, the agreement for the B_u^+ multiplicity is unchanged. However, overall, the agreement of the tuned MC sample with the data points is improved as seen by the increase of the average p-value.

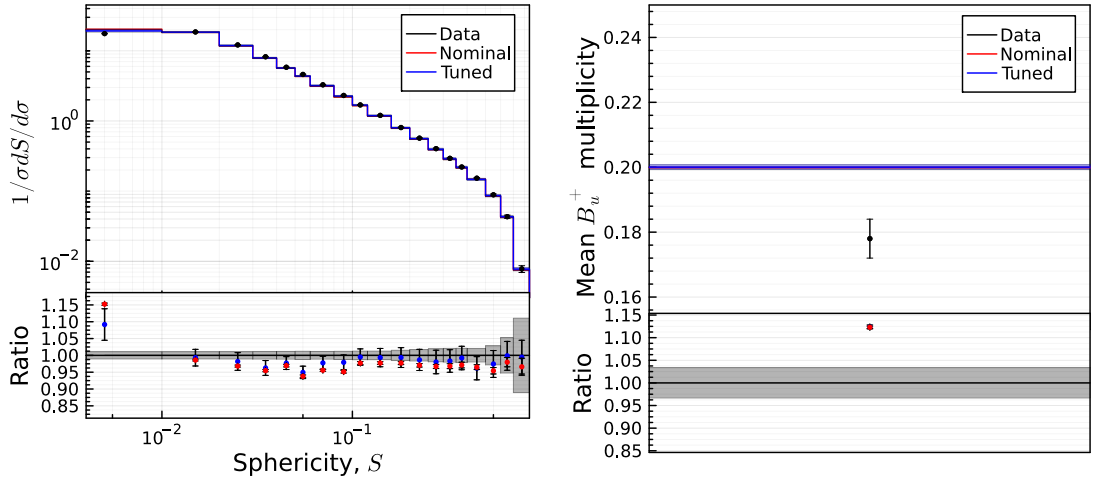


Figure 7.14: Distribution of the sphericity from DELPHI [111] on the left and the B_u^+ multiplicity [113] on the right. The black data points include statistical and systematic uncertainties provided by the `Rivet` framework. The uncertainty band for the nominal MC sample contains the statistical uncertainty while the uncertainty band for the tuned MC sample contains both the statistical uncertainties as well as a so-called *tuning uncertainty*. The bottom panels show the ratio of the data points to the MC samples. The samples were generated using the `Herwig7-P8` model.

7.7 Uncertainty propagation

A major advantage of the Bayesian approach in conjunction with a fast parametrization is the ability to propagate the uncertainty from the parameter space to the observable domain. This uncertainty estimate is reflective of the uncertainty of the tuning procedure due to the limited knowledge of the parameters. To calculate this estimate, 10^6 samples of parameters $\vec{p}$ are generated by resampling the posterior distribution. Using the parametrization function, the corresponding observable values $\vec{y}_{b,O}$ are calculated for each sample:

$$\vec{y}_{b,O} = f_{b,O}(\vec{p})\,. \tag{7.14}$$

The resulting values $\vec{y}_{b,O}$ represent the statistical distribution for the value of the bin content of that bin b of the observable O caused by the statistical uncertainty of the parameters from the tuning process. As such, the spread of the distribution reflects the uncertainty of the tuning process.

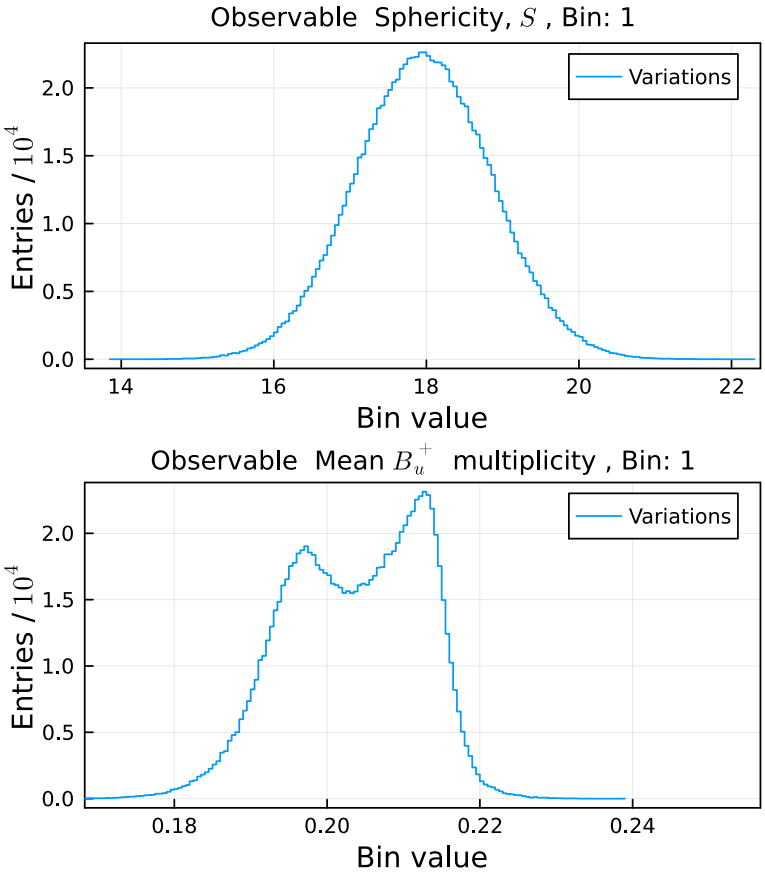


Figure 7.15: Distribution of the bin content of the first bin of the sphericity observable on the top and the B_u^+ multiplicity on the bottom. The distributions are obtained by propagating 10^6 samples of the posterior of the `Herwig7-H7` tune through the parametrization of the model.

As an example, using the tuned parameter set of the `Herwig7-H7` model, the distribution of bin values for the first bin of the sphericity observable and the B_u^+ multiplicity are shown in Figure 7.15. Usually, uncertainties are assumed to be distributed according to a normal distribution. In this case, the distribution of the first bin of the sphericity observable is reasonably well described by a normal distribution, as the distribution is unimodal and symmetric. However, the distribution of the B_u^+ multiplicity has two modes and is not symmetric and can therefore not be described by a normal distribution. In such a case, the uncertainty should be taken into account by quoting the full distribution instead of a single value.

Unfortunately, this greatly increases the complexity of treating and displaying the uncertainty. As such, in order to provide a single value, which reflects the tuning uncertainty, the standard deviation of the distribution is calculated and used as the measure of the uncertainty.

7.8 Influence of correlation of uncertainties to the tune

As mentioned in Section 7.5, the correlation between the data points is an important aspect when constructing the statistical model. For the tunes presented in the previous sections, the correlation between the data points is assumed to be zero. However, it is generally expected that especially systematic uncertainties are correlated between different bins and observables. When introducing a correlation between the data points, the covariance matrix M of the likelihood function in Equation (7.13) is modified to no longer be diagonal. Since the covariance matrix has a dimensionality of $N_{\text{bins}} \times N_{\text{bins}}$, in the case of the observables used in this thesis, the covariance matrix is roughly a 1000×1000 matrix. As manually setting the correlation between all bins results in a large number of parameters, the correlation is assumed to be the same for all bins of a single observable. This creates a correlation matrix that is block diagonal, which is greatly beneficial for runtime and memory usage while still allowing the study of the impact of the correlation on the tuning results.

For each block of the correlation matrix, the off-diagonal elements are set to the value $r \cdot \sigma_i \cdot \sigma_j$ where σ_i and σ_j are the systematic uncertainties of the data point of bin i, and j, and r is the correlation coefficient. Since the actual correlation between the data points is unknown, the correlation coefficient r is varied to account for different levels of correlation by using the values $r = [0.0, 0.4, 0.6, 0.8, 0.9]$. While these values are somewhat arbitrary, they cover a wide range of possible scenarios with low, medium, and high correlations.

Using these different correlation coefficients, the tuning of the `Herwig7-H7` and `Herwig7-P8` models is repeated. As an example of the effect of the correlation, the two-dimensional posterior distribution of the parameters Cl_{max} and Cl_{pow} for the `Herwig7-H7` model is shown in Figure 7.16 for different correlation coefficients. It is observed that the smallest area containing 68% of the probability shrinks with an increasing correlation coefficient. In addition to this effect, for high correlation coefficients, such as $r = 0.8$ and 0.9, the area splits into two distinct areas, indicating the presence of a second local mode within the parameter space. While the introduction of a correlation has a significant impact on the width of the posterior

and, as such, on the uncertainty estimation of the parameters, the effect on the global mode is relatively small. Table 7.6 shows the value for the global mode and the standard deviation of the parameters for the case without correlation and for the case with a correlation coefficient of 0.9. While the standard deviation decreases for all parameters with correlation, the global mode is stable within these uncertainties. In summary, it can be concluded that the tuning result in terms of its parameter values is unaffected by the correlation of the data points. However, it is imperative for the interpretation of their uncertainty to take the correlations into account.

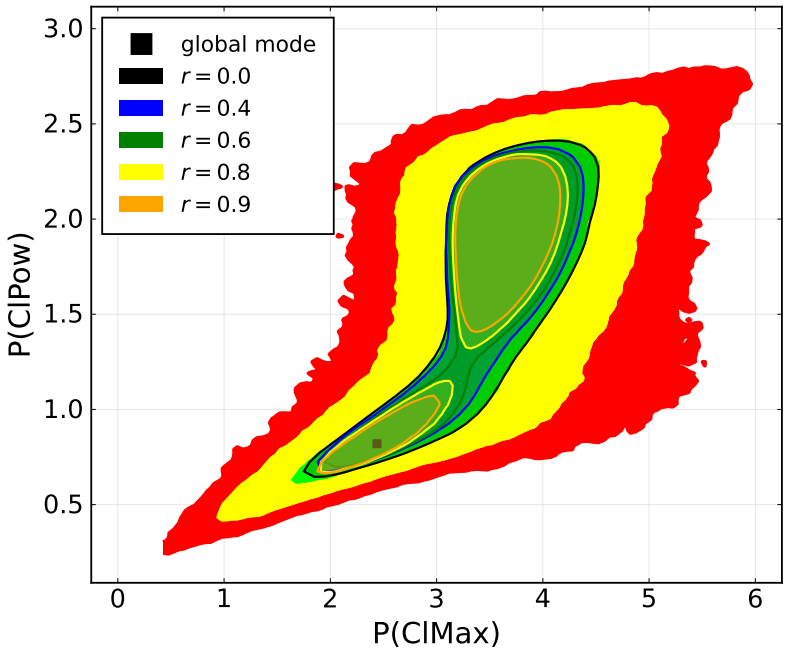


Figure 7.16: The two-dimensional marginalized posterior distribution of the $\mathrm{Cl}_{\mathrm{max}}$ and $\mathrm{Cl}_{\mathrm{pow}}$ parameter for the `Herwig7-H7` model. The green, yellow, and red areas represent the smallest intervals containing 68, 95, and 99% of the posterior distribution respectively. This is shown for the case without correlation. The other contoured areas represent the 68% intervals for the same distribution for different correlation coefficients. The dot represents the location of the global mode

Table 7.6: The position of the mode along with the standard deviation for the tuned parameters of the `Herwig7-H7` model. The results are shown for the scenario without correlation and a correlation of $r = 0.9$.

	Correlation			
	$r = 0.0$		$r = 0.9$	
Parameter	Mode	σ	Mode	σ
AlphaQCD	0.114	0.0032	0.115	0.0018
$m(g)$ (GeV)	0.716	0.129	0.780	0.106
$m(s)$ (GeV)	0.357	0.062	0.390	0.050
IRCutoff (GeV)	0.810	0.221	0.730	0.126
ClMax (GeV)	2.543	0.872	2.081	0.742
ClPow	0.800	0.560	0.667	0.545
ClSmr	0.662	0.194	0.461	0.148
PSplit	0.916	0.128	1.050	0.101

7.9 Influence of weighting of observables to the tune

In conventional MC tuning, weights are used to either stabilize the tuning process or give more importance to specific observables [2, 10]. However, the usage of weights affects the statistical interpretation of the resulting uncertainties and also may bias the outcome of the tune. The impact of weights on the tuning process is studied by repeating the tune of the `Herwig7-H7` and `Herwig7-P8` models using different, non-unity weighting schemes. In order to include weights in the Bayesian tuning framework, the statistical model is modified. The weights are applied to the likelihood function in Equation (7.13) by including an additional factor using a weight vector $\vec{w}_{i=1,\ldots,N_{\text{bins}}}$ which is normalized using the total sum of weights. As weights are defined on the level of observables, all bins of a single observable are weighted equally.

To explore the influence of weights on the tuning process, the weights are varied in two distinct schemes w_1 and w_2 which are listed in Tables A.1, A.2 and A.3 in the Appendix A. The results of the tune are then compared to each other as well as to the tune without weights. The first weighting scheme w_1 is designed to give more importance to the observables related to the multiplicities of particles while leaving most other observables unchanged. In contrast, the second weighting scheme w_2 sets the weights of the multiplicity-related observables to zero while increasing the weights of event shape variables and drastically increasing the weight of the mean charged multiplicity observables.

Table 7.7: The values for the mode and the standard deviation of the parameters for the different weighting schemes for the tune of the `Herwig7-H7` model.

	Weighting scheme					
	none		w_1		w_2	
Parameter	Mode	σ	Mode	σ	Mode	σ
AlphaQCD	0.115	0.003	0.113	0.004	0.115	0.003
$m(g)$ (GeV)	0.709	0.128	0.706	0.136	0.708	0.13
$m(s)$ (GeV)	0.353	0.062	0.352	0.066	0.346	0.063
IRCutoff (GeV)	0.879	0.223	0.859	0.245	0.837	0.214
ClMax (GeV)	2.591	0.871	3.187	0.911	3.761	0.832
ClPow	0.823	0.561	0.847	0.567	2.147	0.541
ClSmr	0.675	0.193	0.501	0.228	0.806	0.213
PSplit	0.868	0.130	0.867	0.141	0.776	0.137

The posterior distributions of the parameters for the different weighting schemes are shown in Figures 7.19 and 7.20 for the `Herwig7-H7` model. In contrast to the effects of different correlation scenarios, the shape of the posterior distribution is largely unaffected by the weighting scheme. However, the position of the global mode and, as such, the tuned parameter set can be affected by the weighting scheme. This effect can be seen most prominent in the one-dimensional marginalized distribution of the $\mathrm{Cl}_{\mathrm{pow}}$ parameter as the mode shifts from the lower peak to the higher peak when comparing the two weighting schemes. This shift is also visible in the two-dimensional distribution of $\mathrm{Cl}_{\mathrm{pow}}$ and $\mathrm{Cl}_{\mathrm{max}}$ where the global mode changes from the lower left region to the upper right region. The values for the global mode and the standard deviation of the parameters for the different weighting schemes are shown in Table 7.7. As indicated by the behavior of the posterior distribution, the AlphaQCD, $m(g)$, $m(s)$, and IRCutoff parameters are largely unaffected, while the $\mathrm{Cl}_{\mathrm{max}}$, $\mathrm{Cl}_{\mathrm{pow}}$, $\mathrm{Cl}_{\mathrm{smr}}$, and PSplit parameters show a shift in the global mode.

In order to study the impact of the weights on the quality of the tune, samples are generated using the tuned parameter sets for each weighting scheme and compared to data analogously to Section 7.6. The distribution of the resulting p-values are shown in Figure 7.17. The tune seems to benefit from the first weighting scheme as a slight improvement of the average p-value from 0.133 without any weights to 0.151 is observed. The second weighting scheme, however, decreases the quality of the tune as the average p-value is lowered to 0.099.

In a similar fashion, the tuning of the `Herwig7-P8` model is repeated using the

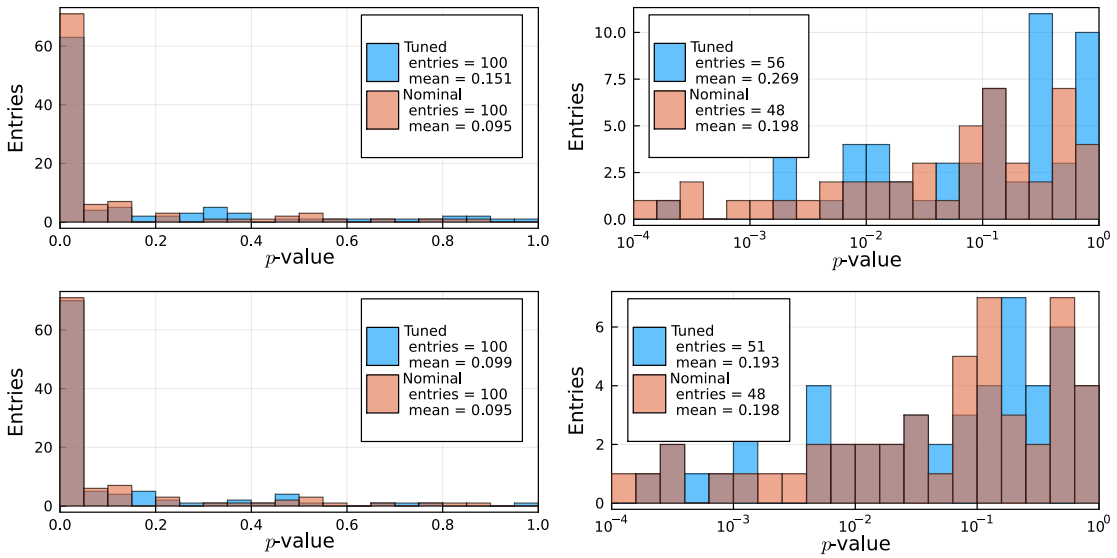


Figure 7.17: Histogram of the p-values for the χ^2 test between data and MC predictions for each observable using the nominal and tuned parameter sets with the `Herwig7-H7` model. The histograms on the right side show the p-values in a logarithmic scale for $p > 10^{-4}$. The top pair of histograms shows the p-values for the first weighting scheme, while the bottom pair shows the p-values for the second weighting scheme.

different weighting schemes. The posterior distributions of the parameters for the tunes using weights are shown in Figures 7.21 and 7.22. Similar to the `Herwig7-H7` model, the shape of the posterior distributions is largely unaffected by the weighting scheme. However, as the `Herwig7-P8` model is generally unimodal, the position of the global mode is only slightly affected by the weighting scheme. This can be confirmed by analyzing the values for the global mode and the standard deviation of the parameters which are listed in Table 7.8. Each mode value for the parameter only changes by a small fraction of the standard deviation. These small changes are also reflected in the p-values of the tuned MC samples, which are shown in Figure 7.18. The average p-value for the tuned MC sample without weights is 0.143 while the average p-values for the tuned MC samples using the weighting schemes w_1 and w_2 are 0.144 and 0.147, respectively. Hence, only a small improvement in the tune is observed for the second weighting scheme.

In conclusion, the effect of weights in the tuning process is highly dependent on the model. In cases where the model is multimodal, such as the `Herwig7-H7` model, the weights can have a significant impact on the position of the global mode and as such stabilize the convergence of the tuning process, which is one of the main reasons for using weights in conventional MC tuning. In other cases, such as the `Herwig7-P8` model, the effect of the weights is rather small, and the tuning process

Table 7.8: The values for the mode and the standard deviation of the parameters for the different weighting schemes for the tune of the `Herwig7-P8` model.

	Weighting scheme					
	none		w_1		w_2	
Parameter	Mode	σ	Mode	σ	Mode	σ
AlphaQCD	0.120	0.003	0.120	0.004	0.120	0.003
IRCutOff (GeV)	1.079	0.313	1.135	0.363	1.115	0.339
aLund	1.287	0.376	1.380	0.416	1.381	0.396
bLund (GeV^{-2})	1.302	0.359	1.369	0.379	1.393	0.363
aExtraDiquark	0.97	-	0.97	-	0.97	-
aExtraSQuark	0.0	-	0.0	-	0.0	-
SigmaPT (GeV)	0.303	0.026	0.304	0.031	0.302	0.027

is largely unaffected by the weights. However, as most full MC tunes are performed on a larger number of parameters and observables, it is more likely to encounter models with a multimodal behavior that would benefit from the usage of weights. It should be noted that in the Bayesian context of this work, the usage of weights to target specific areas of the parameter space to be prioritized could be circumvented by using all local modes of the posterior distribution as different tuned parameter sets, which can then be compared to each other. Although, further studies on the feasibility of this approach in the context of MC tuning are needed.

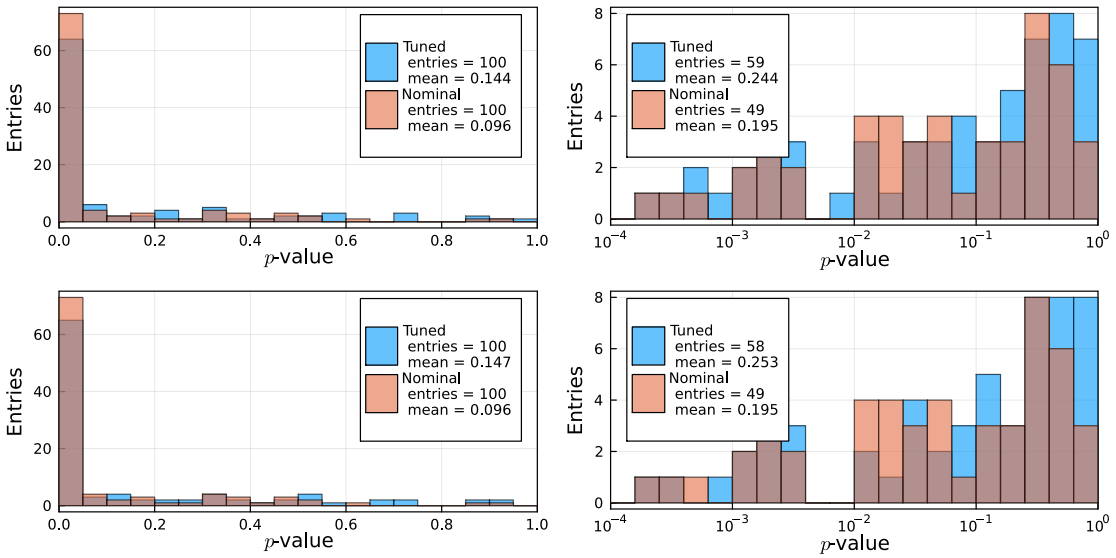


Figure 7.18: Histogram of the p-values for the χ^2 test between data and MC predictions for each observable using the nominal and tuned parameter sets with the `Herwig7-P8` model. The histograms on the right side show the p-values in a logarithmic scale for $p > 10^{-4}$. The top pair of histograms shows the p-values for the first weighting scheme, while the bottom pair shows the p-values for the second weighting scheme.

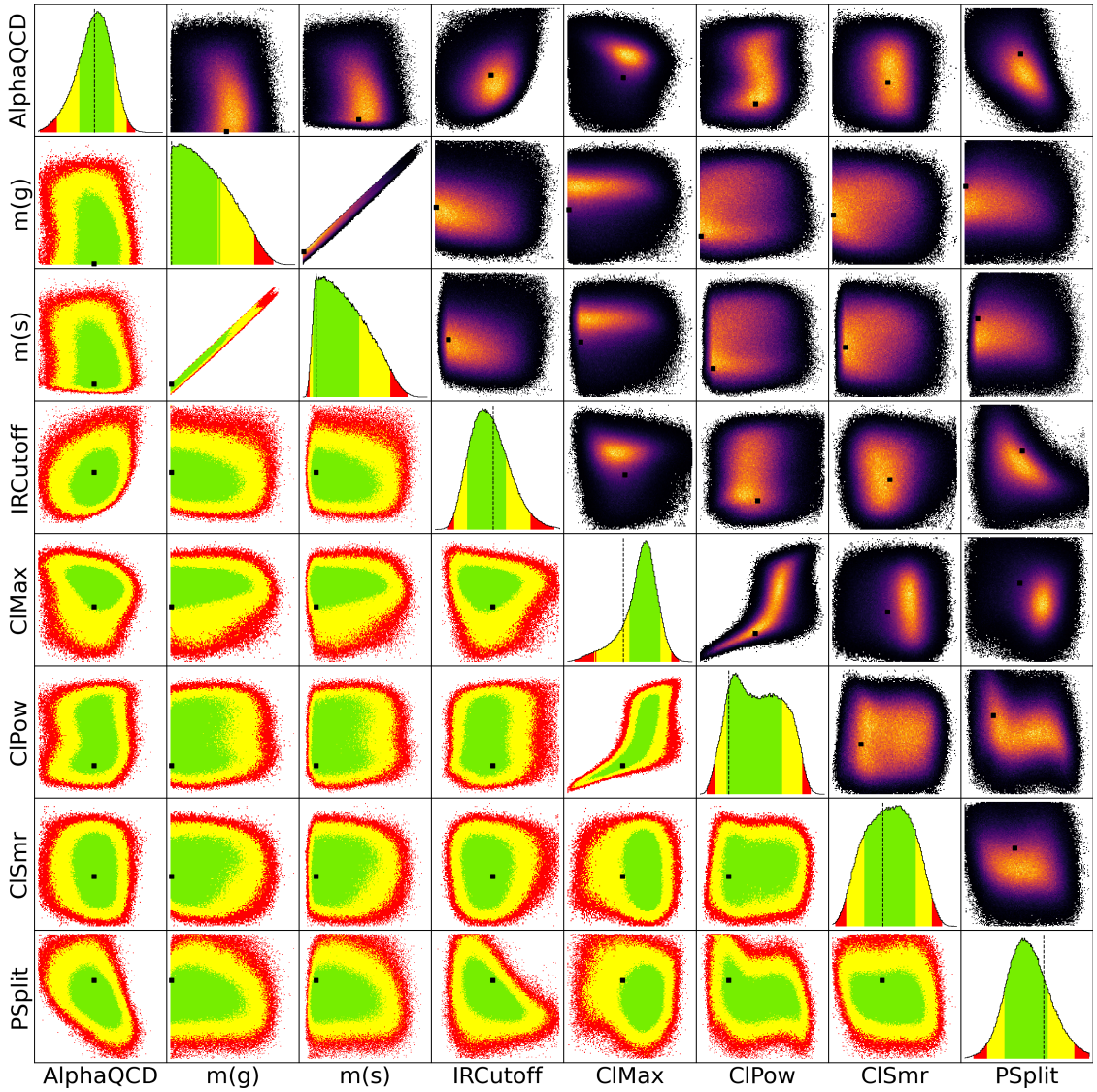


Figure 7.19: The one and two-dimensional marginalized posterior distributions of the parameters for the tune of the `Herwig7-H7` model for the first weighting scheme. The green, yellow, and red areas contain the smallest 68, 95, and 99% intervals of the marginalized probability distributions, respectively. The dashed lines and dots represent the position of the global mode, which is the point with the highest probability.

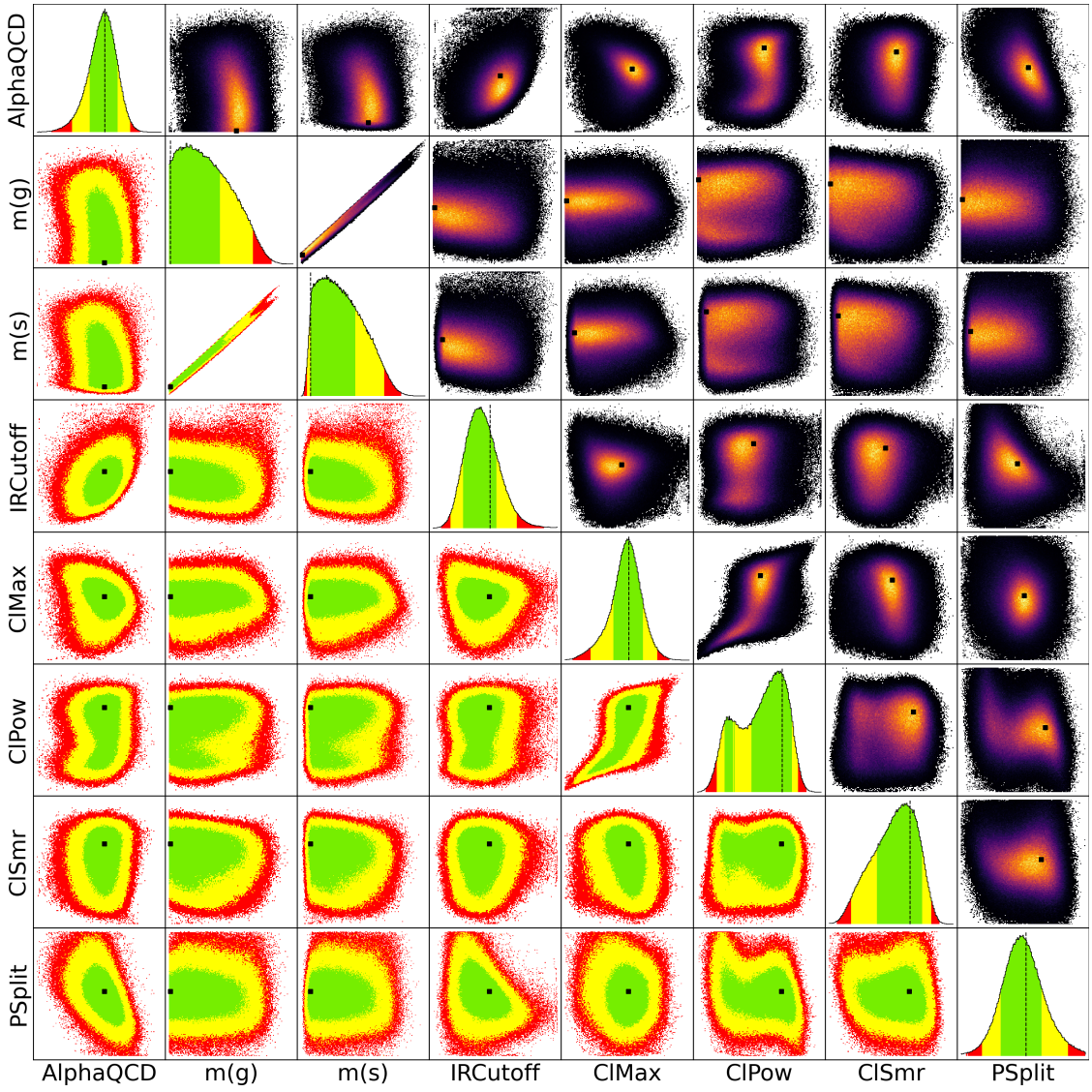


Figure 7.20: The one and two-dimensional marginalized posterior distributions of the parameters for the tune of the `Herwig7-H7` model for the second weighting scheme. The green, yellow, and red areas contain the smallest 68, 95, and 99% intervals of the marginalized probability distributions, respectively. The dashed lines and dots represent the position of the global mode, which is the point with the highest probability.

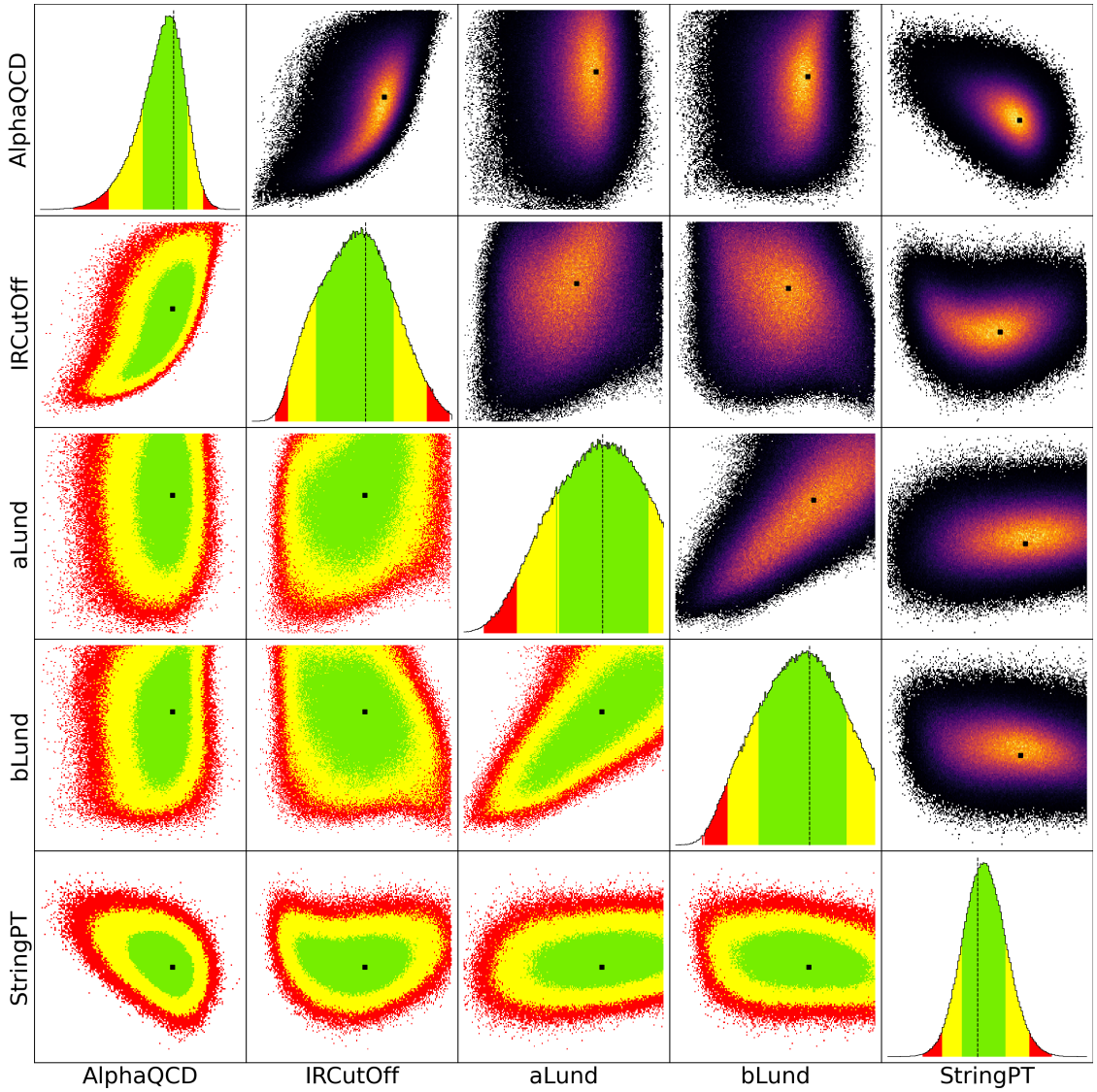


Figure 7.21: The one and two-dimensional marginalized posterior distributions of the parameters for the tune of the `Herwig7-P8` model for the first weighting scheme. The green, yellow, and red areas contain the smallest 68, 95, and 99% intervals of the marginalized probability distributions, respectively. The dashed lines and dots represent the position of the global mode, which is the point with the highest probability.

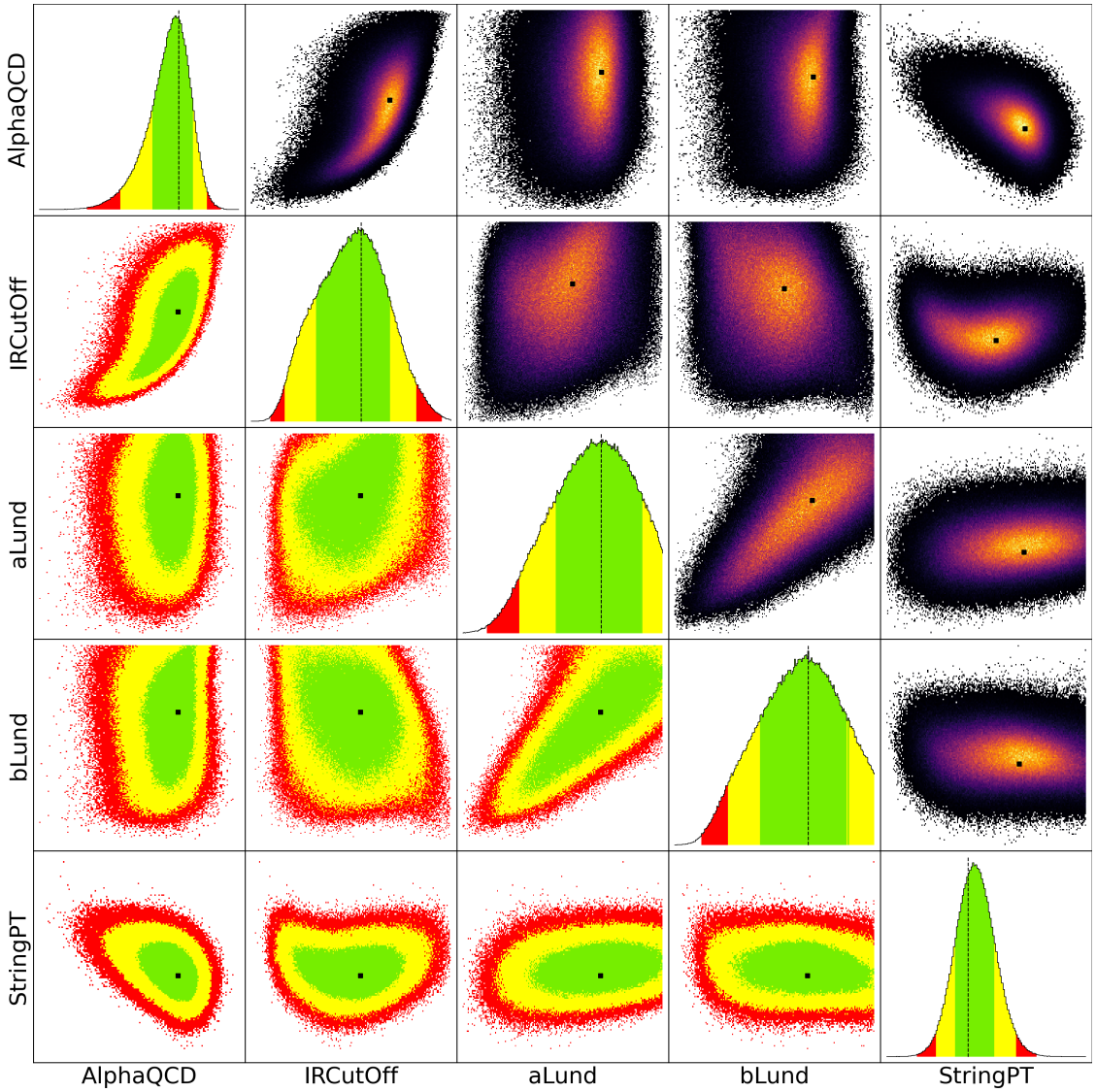


Figure 7.22: The one and two-dimensional marginalized posterior distributions of the parameters for the tune of the `Herwig7-P8` model for the second weighting scheme. The green, yellow, and red areas contain the smallest 68, 95, and 99% intervals of the marginalized probability distributions, respectively. The dashed lines and dots represent the position of the global mode, which is the point with the highest probability.

7.10 Comparison of different hadronization algorithms

In addition to the comparison of the tuned MC samples to the nominal MC samples, the results of the different hadronization models can also be compared to each other. As the `Herwig7-H7` and `Herwig7-P8` models rely on a different set of parameters, a direct comparison of the tuned parameter values is not possible. However, the quality of the tune can still be compared using the observables as well as the p-values of the χ^2 test to data. As an example, the distribution of the sphericity and the B_u^+ multiplicity are shown in Figure 7.23 for both models. In these examples, the

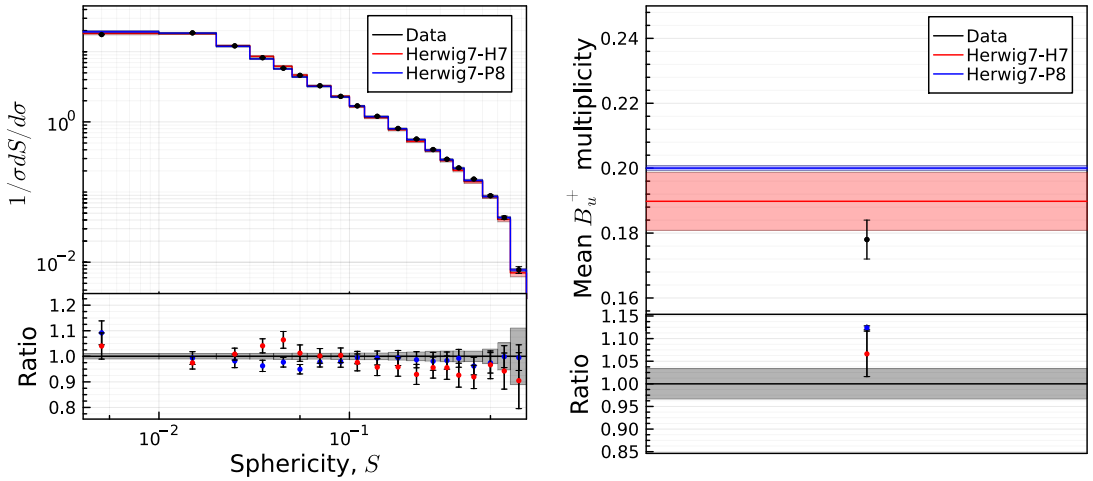


Figure 7.23: Distribution of the sphericity from DELPHI [111] on the left and the B_u^+ multiplicity [113] on the right. The black data points include statistical and systematic uncertainties provided by the `Rivet` framework. The uncertainty band for the MC samples contains the statistical uncertainty as well as the tuning uncertainty. The bottom panels show the ratio of the data points to the MC samples. The MC samples were generated using the `Herwig7-H7` and `Herwig7-P8` models.

`Herwig7-P8` model seems to describe the data points for the sphericity observable slightly better while the `Herwig7-H7` model seems to better describe the data points for the B_u^+ multiplicity. For further comparison, the p-values are shown in Figure 7.24. Overall, the p-values of the `Herwig7-P8` model are slightly higher than the p-values of the `Herwig7-H7` model, averaging at 0.143 and 0.133 respectively, while having 5 less observables with p-values below 10^{-4}.

In order to further quantify the difference between the two models, the observables are split into categories, i.e., event shape variables, multiplicity variables, and so on, and the average p-value for each category is calculated. The resulting average p-values for each category are shown in Table 7.9 together with the total number

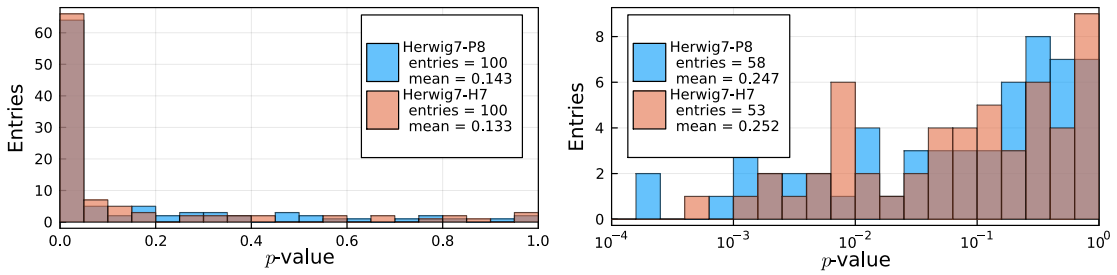


Figure 7.24: Distribution of the p-values of the χ^2 test between data and MC predictions for each observable using the tuned parameter sets with the `Herwig7-H7` and `Herwig7-P8` models. The right histogram shows the p-values in a logarithmic scale for $p > 10^{-4}$.

and the number of observables which are better described by the `Herwig7-P8` model in each category.

The first category of observables features the b quark fragmentation variables, which seem to be better described by the `Herwig7-P8` model according to the p-values. However, the number of observables in this category is rather small, with only 2 observables. The second category consists of the event shape variables, which show an average p-value of 0.0380 for the `Herwig7-H7` model and 0.0003 for the `Herwig7-P8` model. While the p values indicate a significantly better description of the data by the `Herwig7-H7` model, the number of observables that are described better by the `Herwig7-P8` model is higher with 15 observables compared to 13 observables for the `Herwig7-H7` model. This occurrence can be explained, as in this case, the `Herwig7-H7` model has an outlier with a p-value of 0.9794 for one of the observables, significantly increasing the average p-value. Without this outlier, the average p-value for the `Herwig7-H7` model is 0.0004, which is only slightly higher than the `Herwig7-P8` model. Hence, both models seem to describe the event shape variables equally well, with the `Herwig7-P8` model describing more observables better. In contrast, in cases where the `Herwig7-H7` model describes the data better, the difference is more pronounced. The third category is made up of the multiplicity variables, which are described equally well by both models with a similar average p-value and a similar number of observables that are described better by each model. The fourth category comprises the jet rates observables, which favor the description by the `Herwig7-P8` model. However, similar to the b quark fragmentation variables, the number of observables in this category is rather small, with only 4 observables. Lastly, the fifth category includes the particle spectra, which also favor the `Herwig7-P8` model with an average p-value of 0.1340 compared to 0.1042 for the `Herwig7-H7` model and a higher number of observables that are described better by the `Herwig7-P8` model.

Table 7.9: Mean p-values for the tune of the `Herwig7-H7` and `Herwig7-P8` model for the different categories of observables. The total number of observables in each category is shown along with the number of observables for which the p-value of the `Herwig7-P8` model is larger than the one of the `Herwig7-H7` model.

Category	`Herwig7-H7`	`Herwig7-P8`	Number of Observables	$p_{\texttt{P8}} > p_{\texttt{H7}}$
b quark fragmentation	$< 10^{-4}$	0.2224	2	2
Event shape	0.0380	0.0003	28	15
Multiplicity	0.1681	0.1755	54	27
Jet rates	0.4276	0.6484	4	4
Particle spectra	0.1042	0.1340	15	10

In summary, both models seem to describe the data equally well for most observables, especially in the case of the event shape variables and the multiplicity variables, which are the majority of the observables used in this thesis. The particle spectra seem to be better described by the `Herwig7-P8` model. While this is also the case for the b quark fragmentation and the jet rate observables, the lower number of observables in these categories makes it difficult to draw a qualitative conclusion. In conclusion, both models show very similar performance in describing the data, with the `Herwig7-P8` model having a slight advantage given the selection of observables used in this thesis.

8 Conclusions

In this thesis, the development and capabilities of the Bayesian Analysis Toolkit (BAT.jl) were presented. The underlying theory of Bayesian inference and the design principles behind the toolkit were discussed, alongside a detailed overview of state-of-the-art numerical algorithms for posterior exploration and integration included in the package. To ensure the validity and performance of these algorithms, a comprehensive numerical test suite was designed and implemented. This suite serves as a quality check, verifying that the package functions as intended across various scenarios and use cases. Tests were conducted for both low- and high-dimensional scenarios, with a particular focus on the Metropolis-Hastings (MH) algorithm. Various quality measures, including pulls, mode, mean, and variance, as well as Kolmogorov-Smirnov tests and integral estimates, were used to compare samples generated by BAT.jl with independent and identically distributed samples. Additionally, the performance of the Hamilton Monte Carlo sampler was evaluated against the MH algorithm in a high-dimensional test case. The results affirm that BAT.jl can generate samples that accurately represent the target distribution and is well-suited for high-dimensional problems.

Building on the foundation of BAT.jl, a novel Monte Carlo tuning procedure based on Bayesian reasoning was presented in this thesis. The procedure was applied to the `Herwig` Monte Carlo event generator (MCEG) using two different hadronization models, the cluster and Lund string model. Data from the LEP experiments served as the basis for the tuning process and included event-shape and jet-rate distributions, charged hadron momentum spectra, and multiplicities from the process $e^+e^- \to (Z/\gamma) \to$ hadrons. Using the analysis source code provided by the `Rivet` toolkit, the observables from the MCEG samples were derived and subsequently parameterized bin-wise using a third-order polynomial. To interface with the YODA file format generated by `Rivet`, a new package called Yodafiles.jl was developed and incorporated into the tuning procedure. The quality of the parametrization was evaluated using the reduced chi-squared statistic and pull distributions calculated from the reference points of the MCEG samples. Additionally, the robustness of the parametrization was verified through tests on grid-generated samples that were not part of the original fitting process. It was found that the majority of bins are accurately represented by the parametrization. However, some bins exhibited systematic deviations and low p-values, posing challenges for the fit. Despite these challenges, the parametrization was deemed sufficiently accurate for the purpose of

tuning. Further checks on the choice of parameters and observables were conducted. These confirmed that the observables are sensitive to different model parameters and that the parametrization is flexible enough to cover the data points within the chosen parameter ranges. The EFTfitter.jl package was employed to generate a posterior by combining the established parametrization and the data. This posterior was then sampled using the MH algorithm in BAT.jl, which had been previously tested. The global mode of the posterior was used to define the tuned parameter values. Their uncertainties were determined from the smallest interval of the marginalized posterior containing 68% of the probability. MC samples were generated using the tuned parameters and then compared to data through a χ^2 test. The resulting p-values from the tuned simulations significantly outperform those from the nominal MC samples, indicating a successful tune and an improved description of the data. Additionally, the posterior was used to propagate the parameter uncertainties to the realm of the observables, providing a measure for the tuning uncertainty. This uncertainty was found to be substantial in size. However, it should be noted that it is correlated to the uncertainties of the data. Furthermore, the impact of assigning weights to the observables was investigated. While these weights can alter the tuning result, their influence on the tune's quality was found to be minimal in this case. The impact of the correlation between measurements on the tuning was also studied. It was observed that, although the position of the global mode is stable, the associated parameter uncertainties estimates are affected substantially. Hence, it is strongly recommended to take the correlation between measurements into account when performing future tunes. Lastly, the two tuned hadronization models were compared to each other. For the chosen set of observables in this thesis, the Lund string model was found to describe the data slightly better than the cluster hadronization model.

In conclusion, this thesis has presented the development of a novel Monte Carlo tuning procedure rooted in Bayesian reasoning. The Bayesian Analysis Toolkit was introduced and validated through a numerical test suite. Its practical utility and effectiveness were demonstrated with its application in successfully tuning two hadronization models to LEP data. Investigations into aspects of the tuning process, such as the importance of weights, the impact of correlation between measurements, and the propagation of parameter uncertainties, demonstrate the feasibility and capabilities of Bayesian-based tuning. As Monte Carlo tuning remains at the forefront of research, with its methodologies being actively pursued in diverse fields, such as tuning for air shower simulations and forward physics experiments, the insights gained from the tuning procedure presented in this thesis provide valuable contributions to the field and can be used to improve future tunes.

A Appendix

Appendix to Tuning of Monte Carlo event generators using EFTfitter.jl

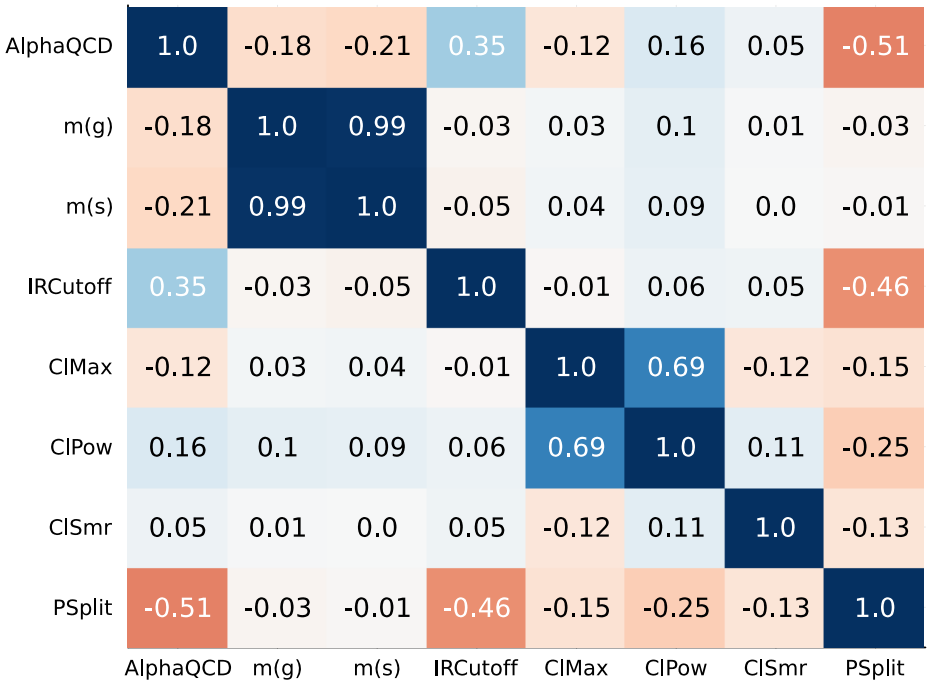


Figure A.1: Correlation matrix of the paramters for the tune of the `Herwig7-H7` model.

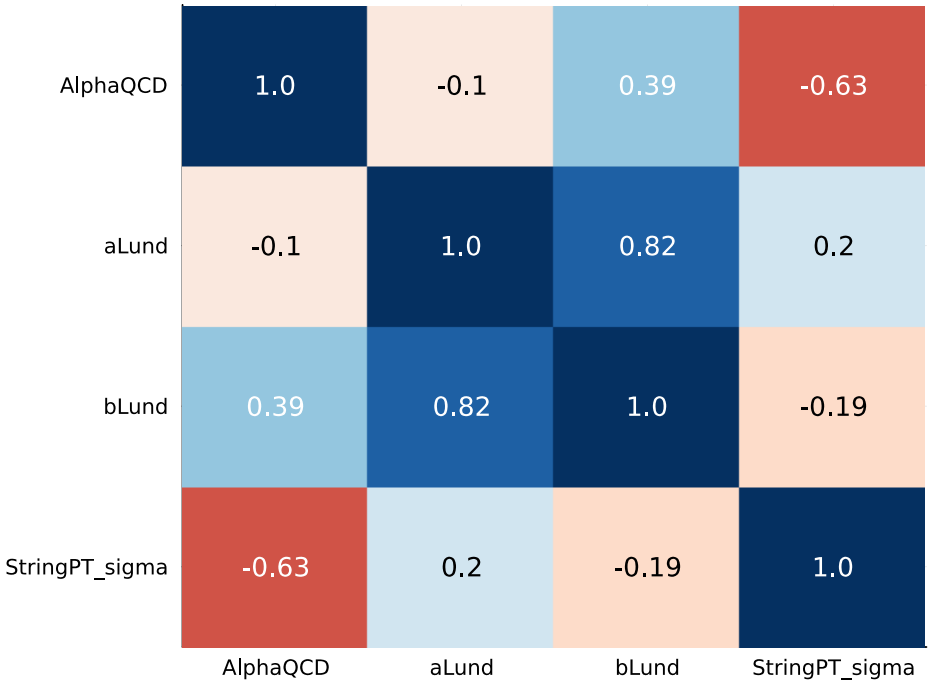


Figure A.2: Correlation matrix of the paramters for the tune of the `Herwig7-P8` model.

Table A.1: List of the `Rivet` analyses used in the tune of the `Herwig7-H7` and `Herwig7-P8` models including their code, description and weights for the two different weighting schemes, part one.

`Rivet` analysis and bin code	Description	Weight scheme	
		w_1	w_2
ALEPH_1996_S3486095 [109]			
d01-x01-y01	Sphericity, S (charged)	1	5
d02-x01-y01	Aplanarity, A (charged)	2	10
d03-x01-y01	1-Thrust, $1 - T$ (charged)	1	5
d04-x01-y01	Thrust minor, m (charged)	2	10
d07-x01-y01	C parameter (charged)	1	5
d08-x01-y01	Oblateness, $M - m$ (charged)	1	5
d09-x01-y01	Scaled momentum, $x_p = \|p\|/\|p_{\text{beam}}\|$ (charged)	1	5
d11-x01-y01	In-plane p_T w.r.t. sphericity axes (charged)	1	5
d12-x01-y01	Out-of-plane p_T w.r.t. sphericity axes (charged)	1	5
d17-x01-y01	Log of scaled momentum, $\log(1/x_p)$ (charged)	1	5
d18-x01-y01	Charged multiplicity	2	10
d19-x01-y01	Mean charged multiplicity	150	750
d25-x01-y01	$\pi^{\pm}$ spectrum	1	1
d26-x01-y01	$K^{\pm}$ spectrum	1	1
d29-x01-y01	π^0 spectrum	1	1
d30-x01-y01	η spectrum	1	1
d31-x01-y01	η' spectrum	1	1
d32-x01-y01	K^0 spectrum	1	1
d33-x01-y01	Λ^0 spectrum	1	1
d34-x01-y01	Ξ^- spectrum	1	1
d35-x01-y01	$\Sigma^{\pm}(1385)$ spectrum	1	1
d36-x01-y01	$\Xi^0(1530)$ spectrum	1	1
d37-x01-y01	ρ spectrum	1	1
d38-x01-y01	$\omega(782)$ spectrum	1	1
d39-x01-y01	$K^{*0}(892)$ spectrum	1	1
d40-x01-y01	ϕ spectrum	1	1
d43-x01-y01	$K^{*\pm}(892)$ spectrum	1	1
ALEPH_2001_S4656318 [110]			
d01-x01-y01	b quark fragmentation function $f(x_B^{\text{weak}})$	7	35
d07-x01-y01	Mean of b quark fragmentation function $f(x_B^{\text{weak}})$	3	15
JADE_OPAL_2000_S4300807 [112]			
d26-x01-y01	2-jet Durham diff. rate	2	10
d26-x01-y02	3-jet Durham diff. rate	2	10
d26-x01-y03	4-jet Durham diff. rate	2	10
d26-x01-y04	5-jet Durham diff. rate	2	10

Table A.2: List of the `Rivet` analyses used in the tune of the `Herwig7-H7` and `Herwig7-P8` models including their code, description and weights for the two different weighting schemes, part two.

`Rivet` analysis and bin code	Description	Weight scheme	
		w_1	w_2
PDG_HADRON_MULTIPLICITIES [113]			
d01-x01-y03	Multiplicity of π^+	10	0
d02-x01-y03	—"— π^0	10	0
d03-x01-y03	—"— K^+	10	0
d04-x01-y03	—"— K^0	10	0
d05-x01-y03	—"— η	10	0
d06-x01-y03	—"— $\eta'(958)$	10	0
d07-x01-y03	—"— D^+	10	0
d08-x01-y03	—"— D^0	10	0
d09-x01-y03	—"— D_s^+	10	0
d10-x01-y01	—"— B^+, B_d^0	10	0
d11-x01-y01	—"— B_u^+	10	0
d12-x01-y01	—"— B_s^0	10	0
d13-x01-y03	—"— $f_0(980)$	10	0
d14-x01-y01	—"— $a_0^+(980)$	10	0
d15-x01-y03	—"— $\rho^0(770)$	10	0
d16-x01-y01	—"— $\rho^+(770)$	10	0
d17-x01-y02	—"— $\omega(782)$	10	0
d18-x01-y03	—"— $K^{*+}(892)$	10	0
d19-x01-y03	—"— $K^{*0}(892)$	10	0
d20-x01-y03	—"— $\phi(1020)$	10	0
d21-x01-y03	—"— $D^{*+}(2010)$	10	0
d23-x01-y02	—"— $D_s^{*+}(2112)$	10	0
d24-x01-y01	—"— B^*	10	0
d25-x01-y02	—"— $J/\psi(1S)$	10	0
d26-x01-y01	—"— $\psi(2S)$	10	0
d27-x01-y01	—"— $\Upsilon(1S)$	10	0
d28-x01-y01	—"— $f_1(1285)$	10	0
d29-x01-y01	—"— $f_1(1420)$	10	0
d30-x01-y01	—"— $\chi_{c1}(3510)$	10	0
d31-x01-y03	—"— $f_2(1270)$	10	0
d32-x01-y01	—"— $f_2'(1525)$	10	0
d34-x01-y02	—"— $K_2^{*0}(1430)$	10	0
d35-x01-y01	—"— B^{**}	10	0
d36-x01-y01	—"— D_{s1}^+	10	0
d37-x01-y01	—"— D_{s2}^+	10	0
d38-x01-y03	—"— p	10	0
d39-x01-y03	—"— Λ	10	0

Table A.3: List of the `Rivet` analyses used in the tune of the `Herwig7-H7` and `Herwig7-P8` models including their code, description and weights for the two different weighting schemes, part three.

`Rivet` analysis and bin code	Description	Weight scheme	
		w_1	w_2
PDG_HADRON_MULTIPLICITIES [113]			
d40-x01-y02	—"— Σ^0	10	0
d41-x01-y01	—"— Σ^-	10	0
d42-x01-y01	—"— Σ^+	10	0
d43-x01-y01	—"— $\Sigma^\pm$	10	0
d44-x01-y03	—"— Ξ^-	10	0
d45-x01-y02	—"— $\Delta^{++}(1232)$	10	0
d46-x01-y03	—"— $\Sigma^-(1385)$	10	0
d47-x01-y03	—"— $\Sigma^+(1385)$	10	0
d48-x01-y03	—"— $\Sigma^\pm(1385)$	10	0
d49-x01-y02	—"— $\Xi^0(1530)$	10	0
d50-x01-y03	—"— Ω^-	10	0
d51-x01-y03	—"— Λ_c^+	10	0
d52-x01-y01	—"— Λ_b^0	10	0
d54-x01-y02	—"— $\Lambda(1520)$	10	0
DELPHI_1996_S3430090 [111]			
d01-x01-y01	In-plane $p_\perp$ w.r.t. thrust axes	1	5
d02-x01-y01	Out-of-plane $p_\perp$ w.r.t. thrust axes	1	5
d03-x01-y01	In-plane $p_\perp$ w.r.t. sphericity axes	1	5
d04-x01-y01	Out-of-plane $p_\perp$ w.r.t. sphericity axes	1	5
d07-x01-y01	Scaled momentum, $x_p = \lvert p\rvert/\lvert p_{\text{beam}}\rvert$	1	5
d08-x01-y01	Log of scaled momentum, $\log(1/x_p)$	1	5
d09-x01-y01	Mean out-of-plane $p_\perp$ w.r.t. thrust axes vs. x_p	1	5
d10-x01-y01	Mean $p_\perp$ vs. x_p	1	5
d11-x01-y01	$1 -$ Thrust	1	5
d12-x01-y01	Thrust major, M	1	5
d13-x01-y01	Thrust minor, m	2	10
d14-x01-y01	Oblateness $= M - m$	1	5
d15-x01-y01	Sphericity, S	1	5
d16-x01-y01	Aplanarity, A	2	10
d17-x01-y01	Planarity, P	1	5
d18-x01-y01	C parameter	1	5
d19-x01-y01	D parameter	1	5
d33-x01-y01	Energy-energy correlation, EEC	1	5
d35-x01-y01	Mean charged multiplicity	150	750

Table A.4: Sensitivity of the `Rivet` analyses used in the tune of the `Herwig7-H7` model to the parameters of the model.

`Rivet` analysis and bin code	AlphaQCD	m(g)	m(s)	IRCutoff	ClMax	ClPow	ClSmr	PSplit
ALEPH_1996_S3486095 [109]								
d01-x01-y01	81.0	31.0	40.0	0.0	2.0	4.0	1.0	2.0
d02-x01-y01	241.0	18.0	1.0	1.0	11.0	7.0	4.0	6.0
d03-x01-y01	42.0	6.0	8.0	1.0	1.0	2.0	0.0	8.0
d04-x01-y01	140.0	67.0	102.0	2.0	1.0	12.0	0.0	11.0
d07-x01-y01	136.0	4.0	3.0	2.0	3.0	4.0	1.0	1.0
d08-x01-y01	28.0	60.0	55.0	0.0	8.0	8.0	2.0	10.0
d09-x01-y01	44.0	10.0	13.0	5.0	7.0	3.0	5.0	13.0
d11-x01-y01	71.0	37.0	45.0	1.0	6.0	12.0	1.0	9.0
d12-x01-y01	205.0	77.0	127.0	7.0	2.0	39.0	5.0	13.0
d17-x01-y01	0.0	6.0	4.0	4.0	7.0	1.0	2.0	2.0
d18-x01-y01	32.0	75.0	123.0	72.0	12.0	6.0	15.0	32.0
d19-x01-y01	36.0	1.0	4.0	6.0	6.0	2.0	1.0	14.0
d25-x01-y01	6.0	13.0	9.0	6.0	10.0	0.0	1.0	8.0
d26-x01-y01	50.0	6.0	5.0	8.0	20.0	1.0	2.0	6.0
d29-x01-y01	49.0	11.0	15.0	2.0	13.0	4.0	2.0	17.0
d30-x01-y01	107.0	130.0	169.0	8.0	26.0	5.0	7.0	30.0
d32-x01-y01	44.0	20.0	14.0	8.0	21.0	2.0	0.0	1.0
d33-x01-y01	40.0	23.0	54.0	6.0	155.0	35.0	1.0	4.0
d34-x01-y01	40.0	449.0	398.0	22.0	281.0	94.0	6.0	3.0
d35-x01-y01	77.0	25.0	36.0	11.0	157.0	41.0	0.0	9.0
d36-x01-y01	13.0	243.0	298.0	0.0	345.0	111.0	5.0	1.0
d37-x01-y01	4.0	45.0	45.0	4.0	5.0	1.0	1.0	11.0
d38-x01-y01	3.0	1.0	0.0	1.0	4.0	1.0	1.0	5.0
d39-x01-y01	74.0	66.0	76.0	15.0	5.0	1.0	1.0	10.0
d40-x01-y01	79.0	45.0	57.0	14.0	8.0	3.0	2.0	6.0
d43-x01-y01	73.0	106.0	108.0	15.0	0.0	0.0	1.0	9.0
ALEPH_2001_S4656318 [110]								
d01-x01-y01	51.0	27.0	29.0	5.0	5.0	7.0	6.0	27.0
d07-x01-y01	81.0	28.0	12.0	6.0	16.0	9.0	3.0	47.0
JADE_OPAL_2000_S4300807 [112]								
d26-x01-y01	211.0	192.0	83.0	11.0	97.0	7.0	13.0	49.0
d26-x01-y02	91.0	85.0	72.0	10.0	25.0	11.0	7.0	44.0
d26-x01-y03	22.0	110.0	236.0	14.0	26.0	26.0	21.0	70.0
d26-x01-y04	115.0	163.0	87.0	21.0	16.0	40.0	1.0	49.0

Table A.5: Sensitivity of the `Rivet` analyses used in the tune of the `Herwig7-H7` model to the parameters of the model.

`Rivet` analysis and bin code	AlphaQCD	m(g)	m(s)	IRCutoff	ClMax	ClPow	ClSmr	PSplit
PDG_HADRON_MULTIPLICITIES [113]								
d01-x01-y03	29.0	6.0	1.0	8.0	9.0	1.0	1.0	14.0
d02-x01-y03	27.0	6.0	1.0	8.0	10.0	1.0	1.0	14.0
d03-x01-y03	85.0	5.0	13.0	11.0	14.0	1.0	1.0	9.0
d04-x01-y03	85.0	5.0	4.0	11.0	14.0	1.0	2.0	9.0
d05-x01-y03	12.0	13.0	10.0	13.0	28.0	1.0	1.0	16.0
d06-x01-y03	28.0	80.0	144.0	1.0	30.0	10.0	0.0	12.0
d07-x01-y03	60.0	20.0	10.0	9.0	17.0	10.0	2.0	2.0
d08-x01-y03	68.0	20.0	30.0	12.0	10.0	5.0	2.0	2.0
d09-x01-y03	18.0	66.0	90.0	2.0	21.0	17.0	2.0	0.0
d10-x01-y01	9.0	3.0	4.0	0.0	10.0	14.0	0.0	0.0
d11-x01-y01	11.0	30.0	21.0	0.0	11.0	15.0	0.0	0.0
d12-x01-y01	30.0	113.0	200.0	11.0	73.0	79.0	1.0	2.0
d13-x01-y03	41.0	101.0	63.0	7.0	51.0	7.0	0.0	27.0
d14-x01-y01	35.0	169.0	162.0	12.0	24.0	0.0	1.0	30.0
d15-x01-y03	28.0	63.0	63.0	11.0	8.0	2.0	1.0	24.0
d16-x01-y01	27.0	70.0	69.0	11.0	9.0	2.0	0.0	23.0
d17-x01-y02	28.0	64.0	63.0	11.0	12.0	1.0	0.0	24.0
d18-x01-y03	112.0	108.0	121.0	15.0	5.0	2.0	0.0	17.0
d19-x01-y03	110.0	80.0	94.0	14.0	4.0	1.0	0.0	17.0
d20-x01-y03	99.0	45.0	56.0	12.0	8.0	4.0	1.0	10.0
d21-x01-y03	75.0	101.0	108.0	14.0	3.0	2.0	1.0	5.0
d23-x01-y02	16.0	36.0	67.0	1.0	21.0	17.0	3.0	0.0
d24-x01-y01	12.0	2.0	3.0	1.0	7.0	11.0	0.0	1.0
d25-x01-y02	29.0	36.0	54.0	2.0	2.0	1.0	1.0	8.0
d26-x01-y01	16.0	54.0	66.0	1.0	1.0	1.0	9.0	10.0
d28-x01-y01	11.0	15.0	7.0	21.0	39.0	11.0	3.0	32.0
d29-x01-y01	83.0	95.0	88.0	4.0	141.0	31.0	2.0	13.0
d30-x01-y01	6.0	672.0	570.0	11.0	88.0	91.0	5.0	10.0
d31-x01-y03	10.0	29.0	28.0	21.0	122.0	29.0	3.0	25.0
d32-x01-y01	48.0	119.0	99.0	12.0	186.0	42.0	4.0	16.0
d34-x01-y02	75.0	33.0	15.0	3.0	117.0	28.0	3.0	16.0
d35-x01-y01	69.0	31.0	34.0	5.0	177.0	182.0	2.0	6.0
d36-x01-y01	64.0	137.0	120.0	6.0	175.0	82.0	2.0	4.0
d37-x01-y01	80.0	88.0	120.0	9.0	292.0	114.0	1.0	8.0
d38-x01-y03	36.0	100.0	108.0	18.0	76.0	28.0	0.0	21.0
d39-x01-y03	84.0	8.0	17.0	9.0	140.0	48.0	1.0	12.0

Table A.6: Sensitivity of the `Rivet` analyses used in the tune of the `Herwig7-H7` model to the parameters of the model.

`Rivet` analysis and bin code	AlphaQCD	m(g)	m(s)	IRCutoff	ClMax	ClPow	ClSmr	PSplit
PDG_HADRON_MULTIPLICITIES [113]								
d40-x01-y02	88.0	47.0	74.0	7.0	126.0	43.0	0.0	13.0
d41-x01-y01	88.0	45.0	68.0	8.0	125.0	34.0	0.0	15.0
d42-x01-y01	89.0	74.0	95.0	6.0	119.0	39.0	1.0	14.0
d43-x01-y01	88.0	60.0	82.0	7.0	122.0	37.0	1.0	15.0
d44-x01-y03	38.0	479.0	436.0	32.0	280.0	89.0	11.0	5.0
d45-x01-y02	5.0	24.0	33.0	30.0	112.0	25.0	1.0	21.0
d46-x01-y03	73.0	134.0	120.0	16.0	158.0	33.0	1.0	9.0
d47-x01-y03	73.0	49.0	62.0	10.0	148.0	44.0	2.0	8.0
d48-x01-y03	73.0	35.0	22.0	13.0	153.0	38.0	1.0	8.0
d49-x01-y02	15.0	404.0	445.0	20.0	319.0	105.0	6.0	3.0
d50-x01-y03	245.0	1099.0	752.0	73.0	508.0	142.0	19.0	29.0
d51-x01-y03	95.0	134.0	133.0	0.0	199.0	126.0	3.0	5.0
d52-x01-y01	36.0	209.0	198.0	9.0	410.0	339.0	7.0	9.0
d54-x01-y02	69.0	71.0	94.0	7.0	249.0	191.0	6.0	7.0
DELPHI_1996_S3430090 [111]								
d01-x01-y01	68.0	45.0	46.0	1.0	7.0	12.0	2.0	10.0
d02-x01-y01	184.0	4.0	22.0	6.0	6.0	20.0	2.0	8.0
d03-x01-y01	59.0	73.0	82.0	1.0	12.0	14.0	2.0	12.0
d04-x01-y01	197.0	6.0	43.0	7.0	5.0	30.0	4.0	10.0
d07-x01-y01	31.0	4.0	6.0	4.0	5.0	2.0	5.0	11.0
d08-x01-y01	15.0	1.0	2.0	6.0	7.0	1.0	4.0	4.0
d11-x01-y01	133.0	59.0	32.0	1.0	1.0	2.0	0.0	0.0
d12-x01-y01	29.0	43.0	31.0	2.0	12.0	6.0	3.0	20.0
d13-x01-y01	72.0	11.0	15.0	1.0	0.0	6.0	1.0	11.0
d14-x01-y01	49.0	48.0	47.0	1.0	9.0	9.0	1.0	7.0
d15-x01-y01	116.0	37.0	49.0	0.0	3.0	5.0	2.0	4.0
d16-x01-y01	288.0	3.0	26.0	2.0	13.0	10.0	4.0	8.0
d17-x01-y01	99.0	45.0	50.0	1.0	2.0	8.0	1.0	0.0
d18-x01-y01	117.0	15.0	22.0	2.0	2.0	3.0	1.0	1.0
d19-x01-y01	235.0	0.0	18.0	2.0	9.0	3.0	3.0	7.0
d33-x01-y01	101.0	0.0	10.0	0.0	3.0	2.0	1.0	5.0
d35-x01-y01	36.0	1.0	4.0	6.0	6.0	2.0	1.0	14.0

Table A.7: Sensitivity of the `Rivet` analyses used in the tune of the `Herwig7-P8` model to the parameters of the model.

`Rivet` analysis and bin code	AlphaQCD	IRCutoff	a	b	aExtraDiQ	aExtraSQ	StringPT
ALEPH_1996_S3486095 [109]							
d01-x01-y01	85.0	1.0	0.0	2.0	0.0	0.0	6.0
d02-x01-y01	148.0	8.0	1.0	4.0	1.0	1.0	21.0
d03-x01-y01	5.0	7.0	6.0	6.0	1.0	0.0	2.0
d04-x01-y01	11.0	21.0	14.0	11.0	0.0	1.0	1.0
d07-x01-y01	81.0	5.0	4.0	2.0	1.0	1.0	5.0
d08-x01-y01	18.0	10.0	9.0	14.0	0.0	0.0	7.0
d09-x01-y01	41.0	7.0	13.0	9.0	0.0	0.0	4.0
d11-x01-y01	46.0	7.0	14.0	15.0	0.0	0.0	16.0
d12-x01-y01	191.0	1.0	16.0	19.0	1.0	0.0	52.0
d17-x01-y01	25.0	1.0	1.0	3.0	1.0	0.0	10.0
d18-x01-y01	955.0	110.0	21.0	21.0	320.0	38.0	171.0
d19-x01-y01	87.0	8.0	10.0	15.0	1.0	0.0	22.0
d25-x01-y01	49.0	4.0	6.0	9.0	1.0	0.0	16.0
d26-x01-y01	58.0	7.0	6.0	10.0	4.0	0.0	26.0
d29-x01-y01	70.0	8.0	13.0	12.0	0.0	0.0	16.0
d30-x01-y01	138.0	17.0	27.0	26.0	1.0	0.0	18.0
d31-x01-y01	280.0	27.0	42.0	36.0	3.0	2.0	7.0
d32-x01-y01	44.0	4.0	2.0	5.0	3.0	0.0	19.0
d33-x01-y01	39.0	2.0	6.0	7.0	3.0	0.0	19.0
d34-x01-y01	96.0	8.0	11.0	21.0	2.0	0.0	18.0
d35-x01-y01	78.0	8.0	13.0	19.0	2.0	0.0	21.0
d36-x01-y01	75.0	8.0	10.0	21.0	0.0	0.0	17.0
d37-x01-y01	26.0	2.0	4.0	6.0	0.0	0.0	16.0
d38-x01-y01	11.0	1.0	0.0	1.0	1.0	0.0	7.0
d39-x01-y01	46.0	5.0	4.0	7.0	1.0	0.0	14.0
d40-x01-y01	44.0	5.0	3.0	6.0	2.0	0.0	12.0
d43-x01-y01	44.0	6.0	5.0	8.0	0.0	0.0	10.0
ALEPH_2001_S4656318 [110]							
d01-x01-y01	62.0	9.0	13.0	11.0	0.0	0.0	2.0
d07-x01-y01	22.0	12.0	15.0	10.0	4.0	1.0	8.0
JADE_OPAL_2000_S4300807 [112]							
d26-x01-y01	608.0	110.0	144.0	2.0357e8	18.0	1.0	100.0
d26-x01-y02	505.0	54.0	72.0	160.0	24.0	3.0	107.0
d26-x01-y03	2.7587e10	120.0	148.0	297.0	43.0	1.0	166.0
d26-x01-y04	633.0	86.0	1.23876e8	82.0	12.0	6.0	96.0

Table A.8: Sensitivity of the `Rivet` analyses used in the tune of the `Herwig7-P8` model to the parameters of the model.

`Rivet` analysis and bin code	AlphaQCD	IRCutoff	a	b	aExtraDiQ	aExtraSQ	StringPT
PDG_HADRON_MULTIPLICITIES [113]							
d01-x01-y03	88.0	8.0	11.0	15.0	1.0	0.0	22.0
d02-x01-y03	88.0	8.0	11.0	15.0	1.0	0.0	22.0
d03-x01-y03	80.0	9.0	8.0	12.0	1.0	0.0	16.0
d04-x01-y03	80.0	9.0	8.0	12.0	2.0	0.0	17.0
d05-x01-y03	97.0	9.0	13.0	18.0	1.0	0.0	27.0
d06-x01-y03	78.0	8.0	9.0	14.0	1.0	0.0	21.0
d07-x01-y03	49.0	9.0	3.0	0.0	3.0	1.0	0.0
d08-x01-y03	46.0	9.0	3.0	0.0	3.0	1.0	0.0
d09-x01-y03	22.0	3.0	2.0	0.0	3.0	1.0	1.0
d10-x01-y01	13.0	0.0	0.0	0.0	0.0	0.0	0.0
d11-x01-y01	13.0	1.0	0.0	0.0	0.0	0.0	0.0
d12-x01-y01	5.0	1.0	1.0	0.0	0.0	0.0	1.0
d13-x01-y03	26.0	1.0	2.0	1.0	1.0	1.0	6.0
d14-x01-y01	2.0	4.0	10.0	14.0	7.0	0.0	3.0
d15-x01-y03	97.0	8.0	13.0	17.0	1.0	0.0	27.0
d16-x01-y01	97.0	8.0	13.0	17.0	1.0	0.0	27.0
d17-x01-y02	96.0	8.0	13.0	17.0	1.0	0.0	26.0
d18-x01-y03	86.0	9.0	9.0	14.0	1.0	0.0	20.0
d19-x01-y03	86.0	9.0	9.0	13.0	2.0	0.0	19.0
d20-x01-y03	71.0	8.0	6.0	10.0	2.0	0.0	14.0
d21-x01-y03	41.0	9.0	2.0	0.0	3.0	0.0	0.0
d23-x01-y02	22.0	3.0	2.0	0.0	3.0	0.0	0.0
d24-x01-y01	12.0	0.0	0.0	0.0	0.0	0.0	0.0
d25-x01-y02	90.0	10.0	6.0	4.0	2.0	1.0	5.0
d26-x01-y01	2.0	3.0	0.0	4.0	5.0	1.0	5.0
d27-x01-y01	47.0	33.0	10.0	9.0	14.0	7.0	15.0
d28-x01-y01	72.0	21.0	8.0	5.0	16.0	1.0	2.0
d29-x01-y01	254.0	7.0	2.29543e8	17.0	0.0	5.0	25.0
d30-x01-y01	22.0	3.0	6.0	6.0	6.0	1.0	1.0
d31-x01-y03	88.0	9.0	5.0	0.0	6.0	0.0	4.0
d32-x01-y01	10.0	5.0	2.0	20.0	0.0	6.0	9.0
d34-x01-y02	43.0	5.0	2.0	4.0	2.0	0.0	4.0
d35-x01-y01	Inf	Inf	Inf	Inf	Inf	Inf	Inf
d36-x01-y01	38.0	3.0	1.0	3.0	1.0	0.0	0.0
d37-x01-y01	10.0	1.0	7.0	15.0	7.0	0.0	4.0
d38-x01-y03	95.0	7.0	14.0	19.0	1.0	0.0	27.0
d39-x01-y03	89.0	8.0	13.0	18.0	0.0	0.0	24.0

Table A.9: Sensitivity of the `Rivet` analyses used in the tune of the `Herwig7-P8` model to the parameters of the model.

`Rivet` analysis and bin code	AlphaQCD	IRCutoff	a	b	aExtraDiQ	aExtraSQ	StringPT
PDG_HADRON_MULTIPLICITIES [113]							
d40-x01-y02	93.0	7.0	14.0	18.0	1.0	0.0	26.0
d41-x01-y01	93.0	7.0	15.0	21.0	0.0	0.0	27.0
d42-x01-y01	85.0	8.0	15.0	19.0	1.0	0.0	26.0
d43-x01-y01	89.0	7.0	15.0	20.0	0.0	0.0	27.0
d44-x01-y03	89.0	7.0	11.0	18.0	0.0	0.0	21.0
d45-x01-y02	102.0	6.0	16.0	22.0	1.0	0.0	31.0
d46-x01-y03	91.0	8.0	15.0	21.0	0.0	0.0	28.0
d47-x01-y03	98.0	8.0	14.0	19.0	1.0	0.0	25.0
d48-x01-y03	94.0	8.0	14.0	20.0	0.0	0.0	26.0
d49-x01-y02	66.0	5.0	9.0	16.0	3.0	0.0	19.0
d50-x01-y03	77.0	4.0	7.0	16.0	4.0	1.0	17.0
d51-x01-y03	38.0	7.0	1.0	1.0	2.0	0.0	0.0
d52-x01-y01	12.0	0.0	1.0	1.0	2.0	0.0	1.0
d54-x01-y02	8.0	7.0	0.0	1.0	1.0	3.0	1.0
DELPHI_1996_S3430090 [111]							
d01-x01-y01	48.0	6.0	13.0	14.0	0.0	0.0	13.0
d02-x01-y01	186.0	2.0	10.0	12.0	1.0	0.0	39.0
d03-x01-y01	32.0	11.0	22.0	20.0	0.0	0.0	19.0
d04-x01-y01	193.0	1.0	12.0	15.0	1.0	0.0	48.0
d07-x01-y01	31.0	5.0	10.0	7.0	0.0	0.0	4.0
d08-x01-y01	5.0	1.0	5.0	0.0	1.0	0.0	7.0
d11-x01-y01	59.0	5.0	3.0	1.0	1.0	1.0	5.0
d12-x01-y01	218.0	26.0	28.0	37.0	4.0	1.0	36.0
d13-x01-y01	33.0	17.0	13.0	12.0	2.0	1.0	1.0
d14-x01-y01	14.0	6.0	7.0	10.0	0.0	0.0	5.0
d15-x01-y01	119.0	2.0	0.0	2.0	1.0	0.0	8.0
d16-x01-y01	174.0	13.0	2.0	4.0	1.0	2.0	24.0
d17-x01-y01	91.0	1.0	2.0	3.0	0.0	0.0	2.0
d18-x01-y01	83.0	3.0	3.0	2.0	1.0	0.0	5.0
d19-x01-y01	182.0	1.0	1.0	5.0	2.0	1.0	17.0
d33-x01-y01	123.0	5.0	2.0	4.0	1.0	0.0	7.0
d35-x01-y01	87.0	8.0	10.0	15.0	1.0	0.0	22.0

Bibliography

[1] O. Schulz et al., *BAT.jl – A Julia-based tool for Bayesian inference*, SN Computer Science **2** (2021) 210, arXiv: 2008.03132 [stat.CO].

[2] A. Buckley et al., *Systematic event generator tuning for the LHC*, Eur. Phys. J. C **65** (2010) 331, arXiv: 0907.2973 [hep-ph].

[3] P. Skands, S. Carrazza, and J. Rojo, *Tuning PYTHIA 8.1: the Monash 2013 Tune*, Eur. Phys. J. C **74** (2014) 3024, arXiv: 1404.5630 [hep-ph].

[4] P. Z. Skands, *Tuning Monte Carlo Generators: The Perugia Tunes*, Phys. Rev. D **82** (2010) 074018, arXiv: 1005.3457 [hep-ph].

[5] ATLAS Collaboration, *Summary of ATLAS Pythia 8 tunes*, ATL-PHYS-PUB-2012-003, CERN (2012), URL: https://cds.cern.ch/record/1474107.

[6] ATLAS Collaboration, *The Pythia 8 A3 tune description of ATLAS minimum bias and inelastic measurements incorporating the Donnachie-Landshoff diffractive model*, ATL-PHYS-PUB-2016-017, CERN (2016), URL: https://cds.cern.ch/record/2206965.

[7] CMS Collaboration, *Extraction and validation of a new set of CMS PYTHIA8 tunes from underlying-event measurements*, Eur. Phys. J. C **80** (2020) 4, arXiv: 1903.12179 [hep-ex].

[8] CMS Collaboration, *Event generator tunes obtained from underlying event and multiparton scattering measurements*, Eur. Phys. J. C **76** (2016) 155, arXiv: 1512.00815 [hep-ex].

[9] A. Buckley and H. Schulz, *Tuning of MC generator MPI models*, Adv. Ser. Direct. High Energy Phys. **29** (2018) 281, ed. by P. Bartalini and J. R. Gaunt, arXiv: 1806.11182 [hep-ph].

[10] J. Bellm and L. Gellersen, *High dimensional parameter tuning for event generators*, Eur. Phys. J. C **80** (2020) 54, arXiv: 1908.10811 [hep-ph].

[11] M. Krishnamoorthy et al., *Apprentice for Event Generator Tuning*, EPJ Web Conf. **251** (2021) 03060, arXiv: 2103.05748 [hep-ex].

[12] P. Ilten, M. Williams, and Y. Yang, *Event generator tuning using Bayesian optimization*, JINST **12** (2017) P04028, arXiv: 1610.08328 [physics.data-an].

[13] A. Andreassen and B. Nachman, *Neural Networks for Full Phase-space Reweighting and Parameter Tuning*, Phys. Rev. D **101** (2020) 091901, arXiv: 1907.08209 [hep-ph].

[14] M. Lazzarin, S. Alioli and S. Carrazza, *MCNNTUNES: Tuning Shower Monte Carlo generators with machine learning*, Comput. Phys. Commun. **263** (2021) 107908, arXiv: 2010.02213 [physics.comp-ph].

[15] N. Castro et al., *EFTfitter—A tool for interpreting measurements in the context of effective field theories*, Eur. Phys. J. C **76** (2016) 432, arXiv: 1605.05585 [hep-ex].

[16] OPAL Collaboration, *The OPAL detector at LEP*, Nucl. Instrum. Meth. A **305** (1991) 275.

[17] ALEPH Collaboration, *ALEPH: A detector for electron-positron annnihilations at LEP*, Nucl. Instrum. Meth. A **294** (1990) 121, [Erratum: Nucl. Instrum. Meth. A 303, 393 (1991)].

[18] DELPHI Collaboration, *The DELPHI detector at LEP*, Nucl. Instrum. Meth. A **303** (1991) 233.

[19] J. Bellm et al., *Herwig 7.0/Herwig++ 3.0 release note*, Eur. Phys. J. C **76** (2016) 196, arXiv: 1512.01178 [hep-ph].

[20] S. Weinberg, *A Model of Leptons*, Phys. Rev. Lett. **19** (1967) 1264.

[21] A. Salam, *Weak and Electromagnetic Interactions*, Conf. Proc. C **680519** (1968) 367.

[22] S. Weinberg, *Nonabelian Gauge Theories of the Strong Interactions*, Phys. Rev. Lett. **31** (1973) 494.

[23] S. L. Glashow, *Partial Symmetries of Weak Interactions*, Nucl. Phys. **22** (1961) 579.

[24] G. 't Hooft and M. J. G. Veltman, *Regularization and Renormalization of Gauge Fields*, Nucl. Phys. B **44** (1972) 189.

[25] D. Galbraith and C. Burgard, *Standard model of physics*, 2012, URL: https://texample.net/tikz/examples/model-physics/ (visited on 07/19/2022).

[26] Particle Data Group Collaboration, *Review of Particle Physics*, PTEP **2022** (2022) 083C01.

[27] P. W. Higgs, *Broken Symmetries and the Masses of Gauge Bosons*, Phys. Rev. Lett. **13** (1964) 508, ed. by J. C. Taylor.

[28] F. Englert and R. Brout, *Broken Symmetry and the Mass of Gauge Vector Mesons*, Phys. Rev. Lett. **13** (1964) 321, ed. by J. C. Taylor.

[29] *LHC Machine*, JINST **3** (2008) S08001, ed. by L. Evans and P. Bryant.

[30] J. Collins, *Foundations of perturbative QCD*, vol. 32, Cambridge University Press, 2013.

[31] T. Sjostrand, S. Mrenna, and P. Z. Skands, *A Brief Introduction to PYTHIA 8.1*, Comput. Phys. Commun. **178** (2008) 852, arXiv: 0710.3820 [hep-ph].

[32] C. Bierlich et al., *Robust Independent Validation of Experiment and Theory: Rivet version 3*, SciPost Phys. **8** (2020) 026, arXiv: 1912.05451 [hep-ph].

[33] L. Lönnblad et al., *TheP8I: an interface between the Pythia8 and the ThePEG toolkit*, 2023, URL: https://gitlab.cern.ch/ThePEG/ThePEG (visited on 01/26/2023).

[34] A. Buckley et al., *General-purpose event generators for LHC physics*, Physics Reports **504** (2011) 145, arXiv: 1101.2599 [hep-ph].

[35] D. Amati and G. Veneziano, *Preconfinement as a property of perturbative QCD*, Physics Letters B **83** (1979) 87.

[36] T. Sjöstrand, *Old Ideas in Hadronization: The Lund String — a string that works.* 2009, URL: http://home.thep.lu.se/~torbjorn/talks/durham09.pdf (visited on 04/15/2009).

[37] M. Bahr et al., *Herwig++ Physics and Manual*, Eur. Phys. J. C **58** (2008) 639, arXiv: 0803.0883 [hep-ph].

[38] B. Andersson, G. Gustafson, G. Ingelman, and T. Sjöstrand, *Parton fragmentation and string dynamics*, Physics Reports **97** (1983) 31.

[39] T. Sjöstrand, *Jet fragmentation of multiparton configurations in a string framework*, Nuclear Physics B **248** (1984) 469.

[40] B. Andersson, G. Gustafson, and B. Soderberg, *A General Model for Jet Fragmentation*, Z. Phys. C **20** (1983) 317.

[41] T. Sjostrand, S. Mrenna, and P. Z. Skands, *PYTHIA 6.4 Physics and Manual*, JHEP **05** (2006) 026, arXiv: hep-ph/0603175.

[42] L. Lönnblad, *ThePEG, Pythia7, herwig++ and Ariadne*, Nuclear Instruments and Methods in Physics Research Section A **559** (2006) 246.

[43] J. Alwall et al., *MadGraph 5 : Going Beyond*, JHEP **06** (2011) 128, arXiv: 1106.0522 [hep-ph].

[44] F. Cascioli, P. Maierhofer and S. Pozzorini, *Scattering Amplitudes with OpenLoops*, Phys. Rev. Lett. **108** (2012) 111601, arXiv: 1111.5206 [hep-ph].

[45] A. Buckley et al., *Rivet user manual*, Comput. Phys. Commun. **184** (2013) 2803, arXiv: 1003.0694 [hep-ph].

[46] E. Maguire, L. Heinrich, and G. Watt, *HEPData: a repository for high energy physics data*, J. Phys. Conf. Ser. **898** (2017) 102006, ed. by R. Mount and C. Tull, arXiv: 1704.05473 [hep-ex].

[47] A. Hald, *A History of Probability and Statistics and Their Applications before 1750*, Wiley Series in Probability and Statistics, Wiley, 2005.

[48] A. Kolmogorov, *Grundbegriffe der Wahrscheinlichkeitsrechnung*, Ergebnisse der Mathematik und ihrer Grenzgebiete, J. Springer, 1933.

[49] T. 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*, Phil. Trans. R. Soc. (1763) 370.

[50] H. Jeffreys, *The Theory of Probability*, Oxford Classic Texts in the Physical Sciences, 1939.

[51] C. Robert, *The Bayesian Choice: From Decision-Theoretic Foundations to Computational Implementation*, Springer Texts in Statistics, Springer New York, 2007.

[52] C. Robert and G. Casella, *Monte Carlo Statistical Methods*, Springer Texts in Statistics, Springer New York, 2013.

[53] G. E. Box and M. E. Muller, *A note on the generation of random normal deviates*, Ann. Math. Statist. **29** (1958) 610.

[54] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, *Equation of state calculations by fast computing machines*, J. Chem. Phys. **21** (1953) 1087.

[55] W. K. Hastings, *Monte Carlo Sampling Methods Using Markov Chains and Their Applications*, Biometrika **57** (1970) 97.

[56] B. Carpenter et al., *Stan: A Probabilistic Programming Language*, Journal of Statistical Software, Articles **76** (2017) 1.

[57] D. J. Lunn, A. Thomas, N. Best, and D. Spiegelhalter, *WinBUGS - A Bayesian modelling framework: Concepts, structure, and extensibility*, Statistics and Computing **10** (2000) 325, URL: https://doi.org/10.1023/A:1008929526011.

[58] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2017, URL: https://www.R-project.org/.

[59] J. Salvatier, T. Wiecki, and C. Fonnesbeck, *Probabilistic programming in Python using PyMC3*, (2016), arXiv: 1507.08050 [stat.CO].

[60] A. Caldwell, D. Kollar, and K. Kröninger, *BAT: The Bayesian Analysis Toolkit*, Comput. Phys. Commun. **180** (2009) 2197, arXiv: 0808.2552 [physics.data-an].

[61] A. Crivellin, M. Hoferichter, and C. A. Manzari, *Fermi Constant from Muon Decay Versus Electroweak Fits and Cabibbo-Kobayashi-Maskawa Unitarity*, Phys. Rev. Lett. **127** (2021) 071801, arXiv: 2102.02825 [hep-ph].

[62] A. Crivellin, F. Kirk, C. A. Manzari, and M. Montull, *Global Electroweak Fit and Vector-Like Leptons in Light of the Cabibbo Angle Anomaly*, JHEP **12** (2020) 166, arXiv: 2008.01113 [hep-ph].

[63] A. Caldwell, M. Ettengruber, A. Merle, O. Schulz, and M. Totzauer, *Global Bayesian analysis of neutrino mass data*, Phys. Rev. D **96** (2017) 073001, arXiv: 1705.01945 [hep-ph].

[64] M. Agostini, G. Benato, and J. Detwiler, *Discovery probability of next-generation neutrinoless double- β decay experiments*, Phys. Rev. D **96** (2017) 053001, arXiv: 1705.02996 [hep-ex].

[65] J. Erdmann et al., *A likelihood-based reconstruction algorithm for top-quark pairs and the KLFitter framework*, Nucl. Instrum. Meth. A **748** (2014) 18, arXiv: 1312.5595 [hep-ex].

[66] A. Collaboration, *Differential $t\bar{t}$ cross-section measurements using boosted top quarks in the all-hadronic final state with 139 fb^{-1} of ATLAS data*, JHEP **04** (2023) 080, arXiv: 2205.02817 [hep-ex].

[67] A. Collaboration, *Search for high-mass dilepton resonances in pp collisions at $\sqrt{s} = 8$ TeV with the ATLAS detector*, Phys. Rev. D **90** (2014) 052005, arXiv: 1405.4123 [hep-ex].

[68] A. Collaboration, *Search for new phenomena in the dijet mass distribution using $p - p$ collision data at $\sqrt{s} = 8$ TeV with the ATLAS detector*, Phys. Rev. D **91** (2015) 052007, arXiv: 1407.1376 [hep-ex].

[69] C. Collaboration, *First Results from CUORE: A Search for Lepton Number Violation via $0\nu\beta\beta$ Decay of ^{130}Te*, Phys. Rev. Lett. **120** (2018) 132501, arXiv: 1710.07988 [nucl-ex].

[70] G. Collaboration, *Background-free search for neutrinoless double-β decay of ^{76}Ge with GERDA*, Nature **544** (2017) 47, arXiv: 1703.00570 [nucl-ex].

[71] P. Ullio and M. Valli, *A critical reassessment of particle Dark Matter limits from dwarf satellites*, JCAP **07** (2016) 025, arXiv: 1603.07721 [astro-ph.GA].

[72] O. Luongo, G. B. Pisani, and A. Troisi, *Cosmological degeneracy versus cosmography: a cosmographic dark energy model*, Int. J. Mod. Phys. D **26** (2016) 1750015, arXiv: 1512.07076 [gr-qc].

[73] C. Rappold et al., *Hypernuclear production cross section in the reaction of ^{6}Li + ^{12}C at 2A GeV*, Phys. Lett. B **747** (2015) 129.

[74] R. Brun and F. Rademakers, *ROOT: An object oriented data analysis framework*, Nucl. Instrum. Meth. A **389** (1997) 81, ed. by M. Werlen and D. Perret-Gallix.

[75] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, *Julia: A Fresh Approach to Numerical Computing*, SIAM Review **59** (2017) 65, arXiv: 1411.1607 [cs.MS].

[76] J. Revels, M. Lubin, and T. Papamarkou, *Forward-Mode Automatic Differentiation in Julia*, (2016), arXiv: 1607.07892 [cs.MS].

[77] J. Eschle et al., *Potential of the Julia Programming Language for High Energy Physics Computing*, Comput. Softw. Big Sci. **7** (2023) 10, arXiv: 2306.03675 [hep-ph].

[78] M. Besançon et al., *Distributions.jl: Definition and Modeling of Probability Distributions in the JuliaStats Ecosystem*, Journal of Statistical Software **98** (2021).

[79] I. Sobol, *On the distribution of points in a cube and the approximate evaluation of integrals*, USSR Comput. Math. Math. Phys. **7** (1967) 86.

[80] P. Bratley and B. L. Fox, *Algorithm 659: Implementing Sobol's Quasirandom Sequence Generator*, ACM Trans. Math. Softw. **14** (1988) 88.

[81] S. Joe and F. Y. Kuo, *Remark on Algorithm 659: Implementing Sobol's Quasirandom Sequence Generator*, ACM Trans. Math. Softw. **29** (2003) 49.

[82] A. Gelman and D.B. Rubin, *Inference from Iterative Simulation Using Multiple Sequences*, Statist. Sci. **7** (1992) 457.

[83] S.P. Brooks and A. Gelman, *General Methods for Monitoring Convergence of Iterative Simulations*, J Comput. Graph. Stat. **7** (1998) 434.

[84] C. J. Geyer, *Practical Markov Chain Monte Carlo*, Statistical Science **7** (1992) 473.

[85] M. Betancourt, *A Conceptual Introduction to Hamiltonian Monte Carlo*, (2018), arXiv: 1701.02434 [stat.ME].

[86] S. Duane, A. Kennedy, B. J. Pendleton, and D. Roweth, *Hybrid Monte Carlo*, Physics Letters B **195** (1987) 216.

[87] R. Neal, *MCMC using Hamiltonian dynamics*, Handbook of Markov Chain Monte Carlo (2012).

[88] H. Ge, K. Xu, and Z. Ghahramani, "Turing: A Language for Flexible Probabilistic Inference," *International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain*, 2018 1682.

[89] M. D. Hoffman and A. Gelman, *The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo*, Journal of Machine Learning Research **15** (2014) 1593, arXiv: 1111.4246 [stat.CO].

[90] M. Innes, *Don't Unroll Adjoint: Differentiating SSA-Form Programs*, (2019), arXiv: 1810.07951 [cs.PL].

[91] V. Hafych, P. Eller, O. Schulz, and A. Caldwel, *Parallelizing MCMC sampling via space partitioning*, Statistics and Computing **32** (2022) 56, arXiv: 2008.03098 [stat.CO].

[92] J. Buchner, *Nested Sampling Methods*, Statistics Surveys **17** (2023) 169, arXiv: 2101.09675 [stat.CO].

[93] J. Buchner, *UltraNest - a robust, general purpose Bayesian inference engine*, Journal of Open Source Software **6** (2021) 3001, arXiv: 2101.09604 [stat.CO].

[94] J. Buchner, *A statistical test for Nested Sampling algorithms*, Statistics and Computing **26** (2016) 383, arXiv: 1407.5459 [stat.CO].

[95] J. Buchner, *Collaborative Nested Sampling: Big Data versus Complex Physical Models*, PASP **131** (2019) 108005, arXiv: 1707.04476 [stat.CO].

[96] M. Lucas et al., *TuringLang/NestedSamplers.jl: v0.8.3*, version v0.8.3, 2023, URL: https://doi.org/10.5281/zenodo.8009875.

[97] A. Caldwell et al., *Integration with an adaptive harmonic mean algorithm*, International Journal of Modern Physics A **35** (2020) 2050142.

[98] M. A. Newton and A. E. Raftery, *Approximate Bayesian Inference with the Weighted Likelihood Bootstrap*, Journal of the Royal Statistical Society. Series B (Methodological) **56** (1994) 3.

[99] T. Hahn, *Cuba - a library for multidimensional numerical integration*, Computer Physics Communications **168** (2005) 78, arXiv: hep-ph/0404043.

[100] G. P. Lepage, *A new algorithm for adaptive multidimensional integration*, Journal of Computational Physics **27** (1978) 192.

[101] G. P. Lepage, *VEGAS - an adaptive multi-dimensional integration program*, tech. rep., Cornell Univ. Lab. Nucl. Stud., 1980, URL: https://cds.cern.ch/record/123074.

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

[103] J. A. Nelder and R. Mead, *A Simplex Method for Function Minimization*, The Computer Journal **7** (1965) 308.

[104] D. C. Liu and J. Nocedal, *On the Limited Memory BFGS Method for Large Scale Optimization*, Math. Program. **45** (1989) 503.

[105] D. Freedman and P. Diaconis, *On the histogram as a density estimator:L2 theory*, Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete **57** (1981) 453.

[106] A. Kolmogorov, *Sulla determinazione empirica di una legge didistribuzione*, Giorn Dell'inst Ital Degli Att **4** (1933) 89.

[107] N. V. Smirnov, *On the estimation of the discrepancy between empirical curves of distribution for two independent samples*, Bull. Math. Univ. Moscou **2** (1939) 3.

[108] N. Smirnov, *Table for Estimating the Goodness of Fit of Empirical Distributions*, Ann. Math. Statist. **19** (1948) 279.

[109] A. collaboration, *Studies of quantum chromodynamics with the ALEPH detector*, Phys. Rept. **294** (1998) 1.

[110] A. collaboration, *Study of the fragmentation of b quarks into B mesons at the Z peak*, Phys. Lett. B **512** (2001) 30, arXiv: `hep-ex/0106051`.

[111] D. collaboration, *Tuning and test of fragmentation models based on identified particles and precision event shape data*, Z. Phys. C **73** (1996) 11.

[112] J. collaboration and O. collaboration, *QCD analyses and determinations of α_S in e^+e^- annihilation at energies between $35GeV$ and $189GeV$*, Eur. Phys. J. C **17** (2000) 19, arXiv: `hep-ex/0001055`.

[113] P. D. Group, *Review of Particle Physics*, Phys. Lett. B **667** (2008) 1.

[114] K. Hamilton and P. Nason, *Improving NLO-parton shower matched simulations with higher order matrix elements*, JHEP **06** (2010) 039, arXiv: `1004.1764 [hep-ph]`.

[115] D.W. Marquardt, *An algorithm for least-squares estimation of nonlinear parameters*, Journal of the society for Industrial and Applied Mathematics **11** (1963) 431.

[116] K. Levenberg, *A method for the solution of certain non-linear problems in least squares*, Quarterly of applied mathematics **2** (1944) 164.

[117] A.E. Albert, *Regression and the Moore-Penrose Pseudoinverse*, Mathematics in science and engineering : a series of monographs and textbooks, Academic Press, 1972.

Acknowledgements

Diese Arbeit wäre ohne die Unterstützung vieler Menschen nicht möglich gewesen. Ich möchte mich bei allen bedanken, die mich auf meinem Weg begleitet und unterstützt haben. Zunächst möchte ich mich bei Prof. Dr.Kevin Kröninger bedanken für die Möglichkeit, diese Arbeit in seiner Arbeitsgruppe anzufertigen, und für die Zusammenarbeit und Unterstützung während der gesamten Zeit. Ebenso möchte ich mich bei Priv.-Doz.Stefan Kluth bedanken als Zweitgutachter dieser Arbeit und für die Unterstützung und die vielen hilfreichen Diskussionen. Des Weiteren danke ich mich bei Andrii für die Zusammenarbeit und Expertise im MC-Tuning-Bereich. Zusätzlich möchte ich mich bei der BAT Crew in München für die Kollaboration bedanken, dabei insbesondere Olli, Lolian, Vasyl und Allen. Ein großer Dank geht ebenso an die Dortmunder BAT-Mans, an Lars Röhrig und Cornelius Grunwald für die tolle und spaßige Zusammenarbeit. Insbesondere möchte ich mich bei allen Korrekturlesern dieser Arbeit bedanken, vielen Dank an Chris, Donna, Cornelius, Lucas und Michi; Danke euch für das wertvolle Feedback!

Generell möchte ich an dieser Stelle meiner Arbeitsgruppe bzw. dem ganzen E4-Lehrstuhl für die Unterstützung danken. Dabei geht ein großes Dankeschön an Andrea Teichmann und Mike Muschak; ohne euch würde hier sicherlich Chaos ausbrechen. Und auch ein großes Dankeschön an unsere Admins, dabei insbesondere an Lucas und Christopher, dafür, dass ihr den Laden mit Tape und Gebet am Laufen haltet.

An dieser Stelle möchte ich mich bei den Leuten bedanken, die mich durch den täglichen Büroalltag im Büro 139 begleitet haben. Das Büro ist durch viele Iterationen gegangen, und ich möchte allen danken, die Teil waren, darunter die OG-Crew in meinem Debüt mit Stella und Marius. Die Chaos und Japan Edition mit Egor, Kärim, Marius, Falko und natürlich Benedikt. Man wird noch in Jahrzenten Nerf-Gun Pfeile und Tee finden aus dieser Zeit! Die Renaissance und Diva Edition mit Ramona und Donna. Bürokonfigurationen sind temporär, Whiteboard-(O)Art und Youtube-Looper sind für die Ewigkeit. Last but not least die BAT-enriched Version mit Michi, an den ich beim Schreiben viel denken musste, und natürlich Lars, der wahrscheinlich immer noch nach dem ominösen Z' sucht.

Ebenso möchte ich mich bei einigen für die alltäglichen Ablenkungen und Wahnsinn bedanken. Zum einen dabei an unsere alten Admins, Kevin und Björn, man könnte

wahnsinnig werden...aus so vielen Gründen! An Cornelius natürlich, E4 und BAT-O.G. aus erster Stunde, und man kann noch immer zusammen lachen und hat sich noch nicht erdrosselt, das muss was heißen. Ebenso an Donna für loopwürdige Musikvorschäge, das gemeinsame Wördeln, fitte Rücken und die tolle Zeit mit Agathe und Malte! Mit passender Überleitung zur Rückenfit-Gang aber auch Tichtennis-Truppe und Tanz-Trupp mit Benedikt, Donna, Lucas, Hendrik, Christopher, Willy, Michi, Tobi und Aaron. Zuletzt noch ein Danke an die Spielenachmittag Gruppe, an Michi, Lucas, Alina, Paulin und Jessica und auch generell an alle 16:16 Enthusiasten für den Kontrast zum Arbeitstag.

An dieser Stelle möchte ich mich bei allen außerhalb des Unilebens bedanken. Ein riesiges Dankeschön geht an Stella für den ganzen emotionalen Support, die unvergesslichen Zeiten, gemeinsamen Urlaube und auch für die noch kommenden Urlaube! Ich bin wahnsinnig dankbar für unsere Freundschaft und könnte dem hier gar nicht gerecht werden! Ein großer Dank geht auch an Marius und Philipp für den ganzen Support und die gemeinsame Zeit. Ein großes Danke geht an die LCK Gruppe und Wördler: Nicy, Kleine und Weller, ich bin sehr froh und dankbar für die tolle Zeit und hoffentlich kommen noch mehr dazu!

Ein gigantisches Dankeschön geht natürlich noch an meine Familie! An meine Eltern, für eure bedingungslose Unterstützung zu jeder Zeit in meinem Leben und dafür, dass ihr immer an mich geglaubt habt. Vielen Dank! Meinen beiden Schwestern Teresa und Rosa, meinem Schwager Salvatore, meinen Neffen Giovanni und meiner Nichte Valentina. Ich bin euch unendlich dankbar dafür eine so tolle Familie zu haben, die immer füreinander da ist!

In questa breve sezione, vorrei ringraziare tutti i miei parenti in Italia che mi hanno accompagnato in questo percorso. In particolare, le mie zie, gli zii, i cugini, le cugine e mia nonna. Grazie mille a tutti voi!