IMPERIAL

Simulation-Based Inference of the Wilson Coefficient C_9 from $B^0 \to K^{0*}\mu^+\mu^-$ Decays

Project: PART-Smith-2

Student: Nils Coutant

Supervisor: Dr Mark Smith

Assessor: Professor Mitesh Patel

Word Count: 8 679

Imperial College London

HEP Group

April 2026

Abstract

The Standard Model of particle physics is extremely successful, yet incomplete. Several measurements performed by the LHCb at CERN have revealed tensions with its predictions referred to as B anomalies. The $B^0 \to K^{0*}\mu^+\mu^-$ decay occurs only at loop level, i.e. through rare higher-order processes, making it sensitive to new physics contributions. The value of the Wilson coefficient C_9 can be determined from the kinematic distributions measured by the LHCb. This is traditionally done using likelihood-based fits that rely on a large number of parameters with complex correlations potentially leading to an underestimated uncertainty and biased estimation. This work explores Simulation-Based Inference as a new approach to estimate the value and uncertainty of C_9. The ideal simulator at theory level fails the misspecification test. Experimental effects are then implemented, partially correcting this issue. The expected scaling relation between the uncertainty and the number of events per sample is confirmed. Several configurations are compared to identify the most suitable one for an accurate inference. Estimates of C_9 with relative deviations of order 15% with respect to the Standard Model prediction and existing fit measurements are obtained using LHCb toy data. However, the simulator is too simplified and the neural networks are insufficiently robust for these estimates to be reliable, as they may be biased and their uncertainties underestimated. These results highlight both the potential and the limitations of Simulation-Based Inference for the estimation of the Wilson coefficients.

Contents

1 Introduction

The Standard Model of particle physics is one of the most successful theories in modern physics. It provides a precise description of the fundamental particles and their interactions. Over the past decades, its predictions have been confirmed by a wide range of experiments with remarkable accuracy. However, the Standard Model is known to be incomplete. It does not explain observations like dark matter, dark energy, the neutrino masses and the matter–antimatter asymmetry. These limitations motivate the search for new physics beyond the Standard Model. Measurements performed by the LHCb (Large Hadron Collider beauty) experiment at CERN have revealed several tensions with Standard Model predictions, referred to as the B anomalies.

In particular, the $B^0 \to K^{0*}\mu^+\mu^-$ decay occurs only at loop level, i.e. through rare higher order processes, making it sensitive to potential contributions from new particles. The distributions of the kinematic observables depend on Wilson coefficients such as C_9. Over the past decade, precise measurements of this decay have been performed by the LHCb. The Wilson coefficient C_9 has been estimated using likelihood-based fits, revealing a discrepancy of around 4σ with the Standard Model prediction [1]. However these fits rely on simplifying assumptions and a large number of free parameters with complex correlations, which could lead to underestimated uncertainties and biased estimations.

Simulation-Based Inference offers an alternative approach to infer parameter values when a simulator capable of reproducing the data is available. With this approach, neural networks are trained on simulated datasets to learn the relationship between the parameters and the observables. This work explores the use of Simulation-Based Inference to estimate the value and uncertainty of the Wilson coefficient C_9 from $B^0 \to K^{0*}\mu^+\mu^-$ decay measurements.

A simulator modelling the theoretical decay distributions and the experimental effects is implemented in order to generate a training dataset. Diagnostics of misspecification are applied to assess the agreement between the simulated data and the LHCb measurements. Several neural networks with different configurations are then trained and compared to determine the most suitable configuration. The calibration and robustness of these models are assessed through implemented diagnostics. The impact of the number of events per sample, the training dataset size and the training duration will be investigated. For experimentation purposes, inferences on LHCb toy data will be attempted and compared with the Standard Model predictions and fit measurements.

Section 2 introduces the physical context and the Simulation-Based Inference framework. Section 3 explains the implementation of the simulator and the experimental effects. Section 4 treats the technical realization of Simulation-Based Inference. Finally, Section 5 presents the results which are discussed in Section 6.

2 Theory

2.1 Beyond the Standard Model

The Standard Model (SM) of particle physics is a theory that describes the behaviour of the fundamental particles through the electromagnetic, the strong and the weak forces. It provides extremely successful predictions that match almost all experimental results obtained over the past decades. For example, the measurement of the electron's anomalous magnetic dipole moment gives $g/2 = 1.00115965218062(12)$ where the parentheses indicate the uncertainty in the

last digits. This measurement agrees with the SM prediction to more than 12 significant figures, making it the most successful prediction in the history of physics [2].

However, the Standard Model is known to be incomplete. It doesn't explain phenomena such as neutrino mixing, dark matter, dark energy, and the matter-antimatter asymmetry. Therefore, it is important to test this theory in order to identify its limitations and to find clues for new physics. This is investigated experimentally in two main ways. The first approach is to try to produce new particles with a particle accelerator like the Large Hadron Collider (LHC) at CERN. This strategy led the ATLAS and CMS experiments to discover the Higgs boson in 2012. However, this method is limited by the energy that the accelerator can reach and no new particle has been found since. The second method is to precisely measure well-understood processes and search for deviations from Standard Model predictions. This approach is used by the LHCb (Large Hadron Collider beauty) experiment, focusing on measurements of particles containing b and c quarks. New particles can affect these predictions through virtual contributions. They can therefore contribute without being directly produced at their physical energy scale. Deviations mainly in quark transitions $b \to s\ell^+\ell^-$ and $b \to c\tau\nu$ have been measured and grouped under the name B anomalies. Figure 1 shows the discrepancy between the Standard Model prediction and the values of the Wilson coefficients C_9 and C_{10} favoured by fits. The origin, corresponding to the Standard Model prediction, lies outside the 2σ preferred regions. Figure 2 illustrates the deviation between the Standard Model predictions and the measurements of the angular observable P_5'. The points, representing LHCb data, deviate from the coloured regions corresponding to the Standard Model predictions. Together with other observable deviations, this amounts to a deviation of 4.1σ [1].

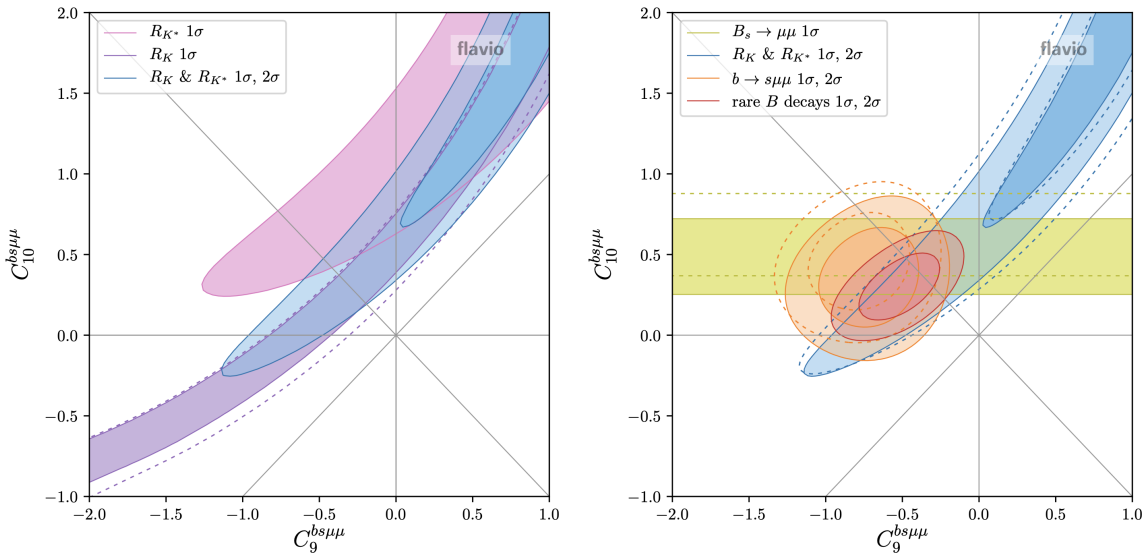


Figure 1: Global fit to $b \to s\mu^+\mu^-$ data in the (C_9, C_{10}) plane. The origin corresponds to the SM prediction and the coloured regions to parameters preferred by experimental data [3].

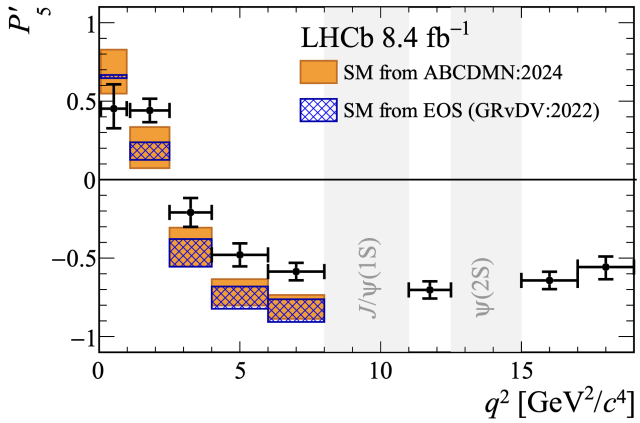


Figure 2: $B^0 \to K^{0*}\mu^+\mu^-$ angular observable P'_5 as a function of the $\mu^+\mu^-$ invariant mass squared q^2. Points are LHCb data, coloured regions are SM predictions [1] [2].

The observed deviations could be explained by new physics like a new boson Z' or new leptoquark particles directly coupling leptons and quarks. However, these deviations could also be explained by a misestimation of Standard Model contributions, such as hadronic uncertainties [2]. These come from non perturbative effects, non local contributions, and limited ability to calculate strong interaction effects inside hadrons. Finally, the deviations could be explained by biased measurements and underestimated uncertainties.

2.2 $B \to K^*\ell^+\ell^-$ decays

The $B \to K^*\ell^+\ell^-$ decays involve a flavour changing neutral current $b \to s\ell^+\ell^-$, which is forbidden at tree level and happens mainly through the loop Feynman diagrams shown in Figure 3. The weak interaction loop causes the decay rate to be suppressed, making it sensitive to potential new contributions and an interesting probe of physics beyond the Standard Model. Figure 3 also illustrates the Feynman diagrams of possible contributions from new physics, such as a new boson Z' or a leptoquark $\mathcal{LQ}$.

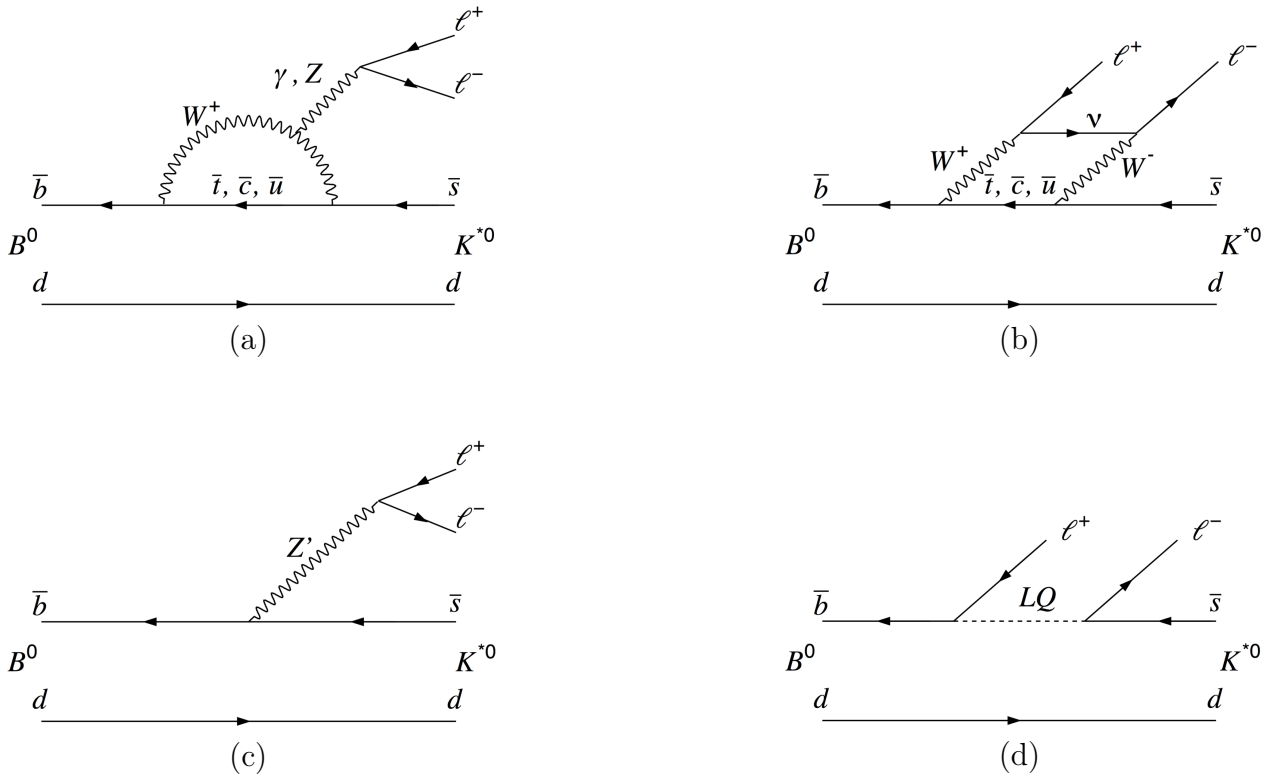


Figure 3: Feynman diagrams contributing to the $B^0 \to K^{0*}\ell^+\ell^-$ decay [4]: (a) penguin diagram (b) box diagram (c) possible contribution from a new heavy boson Z' (d) possible contribution from a leptoquark $\mathcal{LQ}$.

Among all the decays grouped under the label $B \to K^*\ell^+\ell^-$, this study focuses on $B^0 \to K^{0*}\mu^+\mu^-$. At low energies, electrons ($\ell = e$) tend to radiate a large fraction of their energy through interactions with atomic nuclei known as Bremsstrahlung, making them harder to detect. The tau lepton ($\ell = \tau$) is heavier, needs more energy, suppresses the decay rate, and decays before reaching the detectors, also making its identification difficult. Therefore, choosing muons ($\ell = \mu$) makes the decay easier to study experimentally. A similar reasoning motivates the choice of $B = B^0$ and $K = K^{0*}$. The particle K^{0*} is an excited resonance that decays rapidly via $K^{0*} \to K\pi$. The full decay chain can therefore be written as $B^0 \to K^{0*}(\to K\pi)\mu^+\mu^-$.

2.3 Kinematics of the decay

An instance of this decay, called event, is described by four observables: q^2 corresponds to the invariant mass of the dilepton pair $\mu^+\mu^-$, $\theta_K \in (-\frac{\pi}{2}, \frac{\pi}{2})$ is the angle between the B and K momenta in the K^* rest frame, $\theta_\ell \in (-\frac{\pi}{2}, \frac{\pi}{2})$ is the angle between the momenta of B and μ^- in the $\mu^+\mu^-$ rest frame. Finally, $\phi \in (-\pi, \pi)$ is the angle between the planes $K\pi$ and $\mu^+\mu^-$ in the B frame. It is worth noting that the definitions of these angles depend on the convention. The LHCb also measures an additional observable m_B describing the mass of the reconstructed B^0 candidate using the final particles momenta. In ideal conditions, this mass should be equal to the real B^0 mass. But due to measurement uncertainties and energy losses, it differs and follows a distribution that peaks around the true value. Including this observable is important to distinguish real events from background contributions discussed in Section 3.2.2.

The decay $B^0 \to K^{0*}\mu^+\mu^-$ is modelled by the Weak Effective Theory (WET). This framework describes low-energy weak interactions by integrating out the contributions of heavy particles like W and Z bosons in a sum of effective operators O_i. The effective Hamiltonian is expressed as $\mathcal{H}_{\text{eff}} = \sum_i C_i\mathcal{O}_i$ where C_i are the Wilson coefficients. These operators and coefficients determine the distributions of the kinematic observables $(q^2, \theta_K, \theta_\ell, \phi)$. The coefficient C_9 is particularly important for this decay. The Standard Model provides a prediction of $C_9^{\text{SM}} \simeq 4.27$. Measuring a large number of events with the LHCb allows to extract the effective value of C_9 and to compare it with the prediction and potentially identify new physics. This value extraction is traditionally performed using likelihood-based fits giving a deviation of approximately 4σ [1]. However, these fits rely on limited statistics and many correlated parameters. This may lead to biased estimates or underestimated uncertainties. This study investigates Simulation-Based Inference (SBI) as an alternative approach to determine the value of C_9 directly from the data. This method could validate or reject the estimator and uncertainty determined by the traditional fits and could also decrease the uncertainty of the inference to decide between a contribution of new physics or a SM agreement. The goal of this study is not to implement this final inference but to experiment with SBI to illustrate its potential for the inference of C_9. Toy data reproducing very closely the measurements of the LHCb are used as observed data instead of actual LHCb events for data confidentiality reasons.

The value of C_9 depends on the conventions used for the effective Hamiltonian, the renormalization scale, the hadronic contributions included, etc. Several fits using different methods get slightly different results. For these reasons, it is not appropriate to directly compare the inferred value of C_9 to single-point values derived either from the Standard Model or from fit measurements. To simplify the investigation and to focus on the inference itself rather than the physical details, the theoretical value predicted by the Standard Model will be considered to be around $C_9^{\text{SM}} \simeq 4.27$ and the value measured by fits to be $C_9^{\text{fit}} \simeq C_9^{\text{SM}} - 0.93^{+0.18}_{-0.16} \simeq 3.34^{+0.18}_{-0.16}$ [1]. This value should not be interpreted as the definitive measurement made by fits, but as an indication of the values obtained.

2.4 Simulation-Based Inference Framework

Simulation-Based Inference (SBI) is a method used to infer the posterior distribution of model parameters given observed data. In contrast to traditional fitting methods, which require an explicit likelihood, SBI only requires the ability to generate data from a simulator. This makes it especially well suited for this problem, which involves complex distributions and detector effects that are difficult to model analytically. SBI is more convenient than a regular neural network single value inference, as inferring a posterior allows to evaluate the uncertainty of the final estimator. The uncertainty includes the intrinsic uncertainty of the observed sample and the stochasticity of the neural network's inference. However, the simulator model is assumed to be correct and a large amount of data has to be simulated. Many diagnostics also need to be implemented to ensure that the estimator and its uncertainty are correctly determined. These tests can only be applied on simulated data because they require to know the parameters values.

The simulator allows to generate events $x_i = (q^2, \cos\theta_K, \cos\theta_\ell, \phi, m_B)$ following the distribution $p(x \mid \Theta)$, where $\Theta = (C_9)$ is a parameter that can be modified. The notation (C_9) is used to emphasize the fact that the parameter could be multidimensional. N_s values of Θ are drawn from a prior distribution covering the expected values of the parameters. In this study, C_9 is sampled uniformly between 3 and 5. For each value of Θ, N_e events are simulated following the distribution $p(x \mid \Theta)$ and grouped into a sample $X_j = \{x_1, ..., x_{N_e}\}$. All samples form the dataset $\Omega = \{X_1, ..., X_{N_s}\} \in \mathbb{R}^{N_s \times N_e \times 5}$. A neural network is trained on this dataset Ω to infer the posterior $p(\Theta \mid X)$ representing the probability that each value of Θ generates the observed sample X. Once the neural network is correctly trained, it can be used to infer the effective posterior of Θ using the data measured by the LHCb experiment $X^\star$.

In this study, assuming the posterior to be symmetrical and unimodal, i.e. peaking only around one value, is a reasonable simplification. Under these assumptions, the mean, the median and the maximum a posteriori are equivalent. Hence, the estimator is chosen to be the mean as it is easier to compute. Its uncertainty is defined as half the width of the central 68% interval of the posterior $(q_{84} - q_{16})/2$, where q_{16} and q_{84} are the 16th and 84th posterior quantiles. These statistics are computed by sampling a large number of parameters from the inferred posterior.

2.5 Requirements for reliable inference

An inference is accurate when the uncertainty is small. But with SBI, the inference also needs to be reliable. There are mainly three reasons that can cause a prediction to be wrong: a misspecification, a wrong calibration and a lack of robustness. These are detailed in the following subsections.

2.5.1 Misspecification

An SBI model is said to be misspecified if the simulator cannot reproduce the real observed data for any parameter value. The simulated data is too different from the observed sample and this causes the neural network to learn an incorrect relationship between the parameter Θ and the samples X. Given the effective LHCb data $X^\star$, different from the training dataset, the neural network will output a biased inference.
A misspecification can be diagnosed using the Maximum Mean Discrepancy (MMD) metric which measures the distance between two distributions. The MMD is computed between multiple datasets generated by the simulator in order to estimate the natural variability of MMD values. The MMD between the LHCb toy data and simulated data can then be computed. If the distance lies within the intrinsic MMD variation of the simulator, the simulator is reliable, otherwise it is misspecified [5]. If a simulator is misspecified, it should be made more complex

to resemble the real data.

2.5.2 Calibration

A neural network is considered well calibrated if the predictions are not biased and the uncertainties are neither underestimated nor overestimated. Many diagnostics allow to assess the calibration of a neural network. These include: Simulation-Based Calibration (SBC), Expected Coverage Test (ECT), Posterior Predictive Check (PPC) and Test of Accuracy with Random Points (TARP). All of these diagnostics have been implemented for this study. However to not go too deep into the details, only the Expected Coverage Test is described here.
A value of $\tilde{\Theta}$ is drawn from the prior (see section 2.4) and a sample $\tilde{X}$ is simulated using the simulator $\tilde{X} \sim p(X \mid \tilde{\Theta})$. The neural network outputs the posterior $p(\Theta \mid \tilde{X})$. The credible intervals of level $\alpha \in [0, 1]$ are computed. This process is repeated many times for different $\tilde{\Theta}$ and the frequency $f(\alpha)$ of the true value $\tilde{\Theta}$ being in the credible interval of level α is computed. If the neural network is well calibrated, the plot of $f(\alpha)$ should follow a diagonal line [5]. Otherwise, the uncertainty would be over- or underestimated or the inference biased. The diagnostics SBC and TARP are very similar but use different criteria than the frequency of the true value being in the credible interval. Poor calibration can be caused by an undertraining, an overfitting or by a limited training dataset.

2.5.3 Robustness

Finally, it is important to make sure that the neural network is robust, i.e. that its inferences are stable under small deviations between the training dataset and the observed sample. Indeed, even if the simulator is not misspecified, small deviations between the simulated data and the real observed sample are unavoidable. This is due to the approximations made while creating the simulator. For example, the case where the particles $K\pi$ are created with orbital angular momentum $L = 0$, i.e. not via K^{0*}, is called an S-wave. It has a small contribution that has been ignored for the simulator, as no simple model is yet available. For the inference to be reliable, the neural network should not learn information from these discrepancies. Otherwise giving the neural network a slightly different observed sample will make the inference biased.

The robustness can be evaluated by assessing the behaviour of the average uncertainty and estimator shift under small deviations between the simulated training data and observed samples, for example as a function of the amplitude of a small Gaussian noise added to a simulated observed sample, or as a function of a reduced number of events in the observed sample. The estimator should stay stable or drift slowly. The uncertainty should stay consistent with the estimator shift and increase slowly. If instead the estimator shifts very quickly away from the first prediction, or if the uncertainty decreases, the neural network is not robust and the final inference will be biased. The estimator shift as a function of an additional Gaussian noise magnitude δ is calculated as

$$\langle |\Theta(\delta) - \Theta_{\text{ref}}(0)| \rangle$$

where $\Theta(\delta)$ is an estimator for an observed sample with an additional noise of amplitude δ, and $\Theta_{\text{ref}}(0)$ is an estimator when no noise is added. The average is computed over multiple observed samples, multiple random noise generations of amplitude δ and multiple inferred estimators for each of the noisy observed samples. Even when there is no noise, i.e. $\delta = 0$, the average is not zero because of the small random variations when calculating an estimator by sampling the posterior. This uncertainty is called the baseline, it can be reduced by increasing the number of sampled parameters when calculating an estimator from a posterior. The average uncertainty $\langle \sigma(\delta) \rangle$ is computed with the same conditions. Finally, the quantity

$$\left\langle \frac{|\Theta(\delta) - \Theta_{\text{ref}(0)}|}{\sigma(0)} \right\rangle$$

is computed to compare the scale of the estimator shift with the initial uncertainty. A neural network is considered to be robust if the value is $\ll 1$ for physically reasonable noise amplitude δ. Similar quantities for robustness under a reduced number of observed events are computed:

$$\langle |\Theta(N) - \Theta_{\text{ref}}(N_{\text{max}})| \rangle \,, \qquad \langle \sigma(N) \rangle, \qquad \left\langle \frac{|\Theta(N) - \Theta_{\text{ref}}(N_{\text{max}})|}{\sigma(0)} \right\rangle ,$$

with $\Theta(N)$ an estimator for an observed sample containing N events and $\sigma(N)$ its uncertainty.

Poor robustness can be caused by an undertraining, an overfitting, a limited training dataset or a too complex architecture. It can be improved by adding neural network regularizations during the training. These can for example be an early stopping, a weight decay or a dropout. In this study, only an early stopping has been implemented as most regularisations are not supported by the Python library used, as discussed later.

3 Data Generation

3.1 Ideal physics simulator

The Python library EOS [6], designed for precision calculations in flavour physics, has been used to generate the observables q^2, $\cos\theta_K$, $\cos\theta_\ell$, ϕ of the events at theory level. In other words, the data simulated do not correspond exactly to the measurements made by the LHCb. This is discussed in the Section 3.2. The Table 1 gives the EOS configuration used.

Category	**Parameter**	**Value**
Global	EOS Decay	`B->K*ll::d^4Gamma@LargeRecoil`
	Parameter	`b->smumu::Re{c9}`
	Model	`WET` (Weak Effective Theory)
Flavours	ℓ	μ
	q	d
Observables ranges	q^2 [GeV^2]	[1.1, 6.0]
	$\cos\theta_\ell$	$[-1, +1]$
	$\cos\theta_K$	$[-1, +1]$
	ϕ	$[-\pi, +\pi]$
MCMC sampling	`stride`	100
	`pre_N`	1000
	`preruns`	10

Table 1: EOS configuration used to simulate ideal data.

EOS uses Markov Chain Monte Carlo (MCMC) sampling that can lead to correlation between consecutive points or with the initial starting point. This issue is fixed using the following method. MCMC begins at the starting point, generates `preruns` times `pre_N` events. Because of a random walk diffusion, the correlation with the initial conditions decreases to nearly zero. Then, events are selected every `stride` points to prevent any correlation between consecutive events. The values of `preruns`, `pre_N` and `stride` are indicated in Table 1. The Figure 4 shows that the correlation drops rapidly to zero for the next selected point, indicating that the MCMC parameter values are satisfying. These values could likely be reduced to accelerate data generation without affecting the quality.

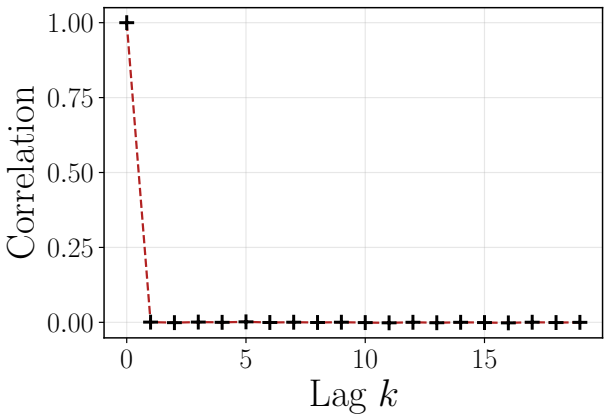


Figure 4: Correlation between selected events separated by a lag k, correlation for $k = 1$: 7.3×10^{-4}

Figure 5 illustrates the distributions obtained for the expected value $C_9^{\text{fit}} \simeq 3.34$ and compares them with the LHCb toy data. Indeed, the distributions do not exactly match because of the detector effects discussed in Section 3.2. The observed discrepancies occur for low q^2, high q^2, low $\cos\theta_\ell$, high $\cos\theta_K$, and intermediate values of m_B.

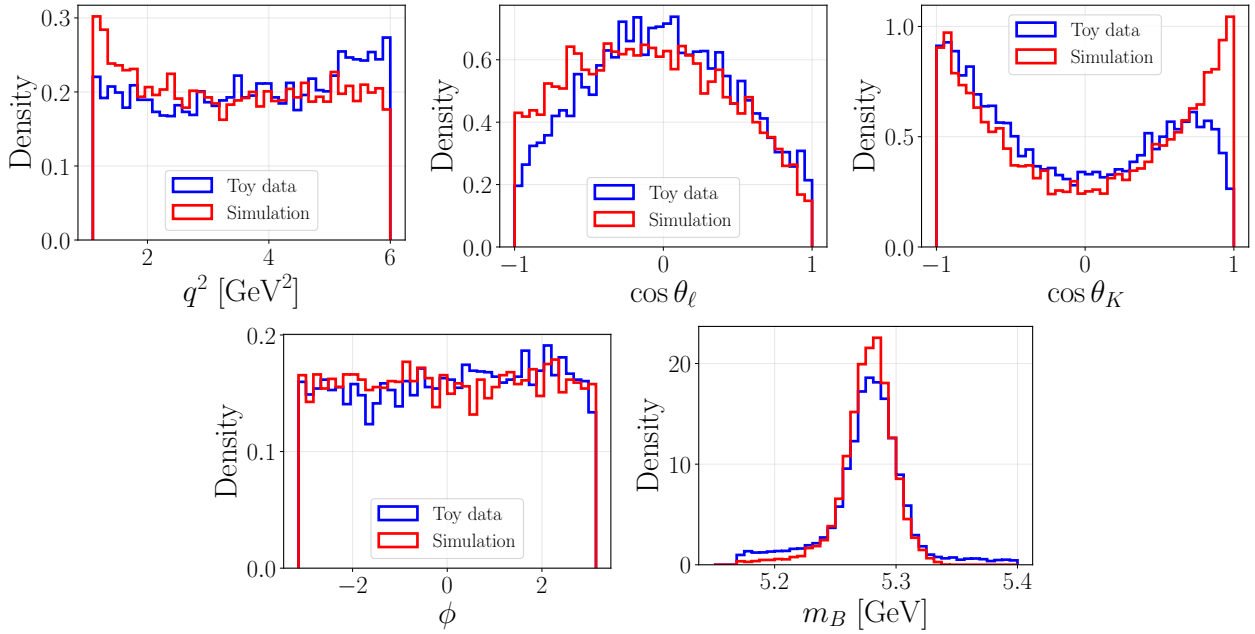


Figure 5: Distributions of the observables q^2, $\cos\theta_\ell$, $\cos\theta_K$, ϕ, and m_B generated by the ideal simulator for $C_9^{\text{fit}} \simeq 3.34$.

The last observable m_B is drawn independently of the first four observables using a custom probability density function determined by a past study [1]. This function is a sum of two crystal ball functions [7] weighted by the coefficient f_{core}. The mathematical definition of a crystal ball is presented in Equation (2) in the Appendices. They share the same mean μ but have different widths σ_i, transition points α_i and tail powers n_i. The values of m_B are generated within $m_{B,\text{min}}$ and $m_{B,\text{max}}$. Table 2 lists the parameter values used. The resulting distribution is illustrated in Figure 5.

Parameter	**Value**
f_{core}	0.7995
μ [GeV]	5.2800
σ_1 [GeV]	0.0157
α_1	1.8470
n_1	1.1296
σ_2 [GeV]	0.0255
α_2	-2.1940
n_2	2.3460
$m_{B,\text{min}}$ [GeV]	5.17
$m_{B,\text{max}}$ [GeV]	5.70

Table 2: Parameters of the double Crystal Ball function used for the m_B distribution [1]

It is useful to generate this observable because its distribution is different for signal events and for background contributions, as explained in Section 3.2.2. This allows the neural network to identify background contributions and limit their impact.

3.2 Experimental effects

The neural network will use simulated data as a training dataset but the final inference is performed on LHCb toy data. For this inference to be relevant, the simulated data should be very close to the LHCb data. Thus, it is necessary to implement the impact of the detectors on the observable distributions. The detector effects can be modelled by mainly three sources: the acceptance, the background and the resolution. These are discussed in the following subsections.

3.2.1 Acceptance

The LHCb has an efficiency $\varepsilon(x)$ to detect an event x depending on the values of the observables $x = (q^2, \cos\theta_K, \cos\theta_\ell, \phi, m_B)$. For example, some angles might be easier to detect than others. This deforms the ideal distributions generated by EOS. The acceptance is modelled by accepting simulated events with a probability $\varepsilon(x)$. This function is expressed as a sum of polynomials with coefficients determined by a previous study [1]. The probability is often very low, and has been boosted artificially in this study by multiplying the output by 40 in order to make the data generation faster. This appears to have no impact on the final distributions as the acceptance probabilities are still low, of order 0.1. Furthermore, the coefficients of $\varepsilon(x)$ are slightly different depending on the year of the LHCb measurements, on the scale of q^2 and for B^0 or $\bar{B}^0$. To simplify the implementation, this has been ignored and only the coefficients from the year 2017, for high q^2 and B^0 have been used. Finally, the coefficients depend on the invariant mass $m_{K\pi}$ of the system $K\pi$, which for simplicity has been assumed to be constant at $m_{K\pi} = 892$ MeV. The shape of the efficiency function $\varepsilon(x)$ is shown in Figure 6 and is independent of the value of m_B.

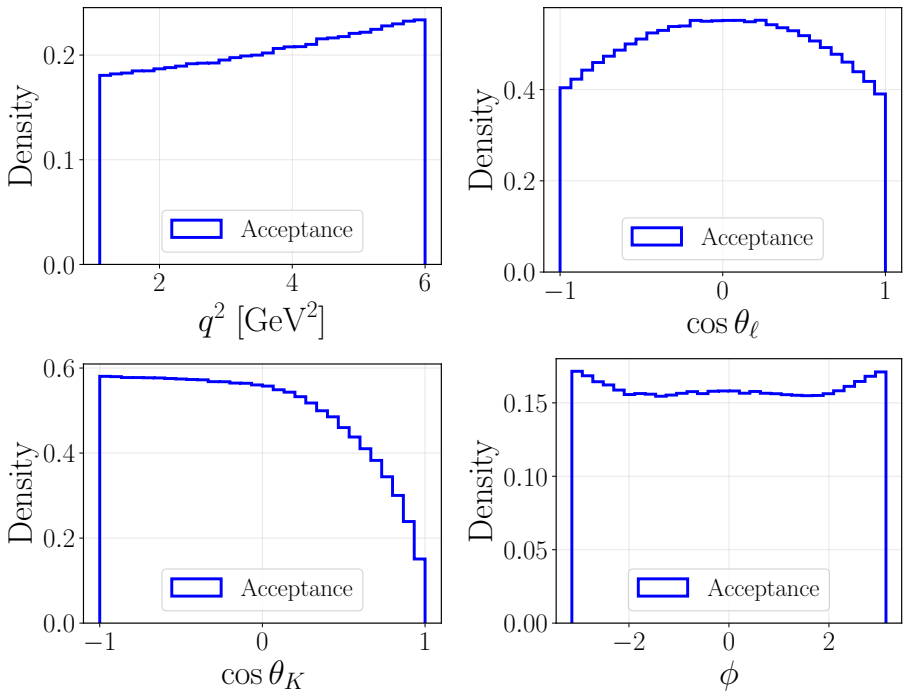


Figure 6: Efficiency function $\varepsilon(x)$ modelling the detector acceptance

3.2.2 Background contributions

The background contributions correspond to events that come from other decays that have been misidentified or to random noise combinations in the detectors. The background is modelled by replacing a fraction f_b of simulated events by new events drawn from another distribution. The invariant mass squared q^2 is sampled uniformly between q^2_{min} and q^2_{max}, the angles $\cos\theta_K, \cos\theta_\ell$ and ϕ are drawn from distributions expressed as second order Chebyshev polynomials $1+p_1 t+p_2(2t^2-1)$. The reconstructed B mass m_B is drawn from a truncated exponential $e^{-\tau m_B}$. These observables are sampled in their usual ranges indicated in Table 1 and Table 2. The Table 3 presents the values of the background parameters used. The shape and coefficients have been determined by a previous study [1]. The distributions are illustrated in Figure 7.

Parameter	Value
$p_1^{\cos\theta_\ell}$	0.4773
$p_2^{\cos\theta_\ell}$	0.2071
$p_1^{\cos\theta_K}$	0.0834
$p_2^{\cos\theta_K}$	0.3354
p_1^{ϕ}	0.2216
p_2^{ϕ}	0.0674
τ [GeV^{-1}]	5.745
$1-f_b$	0.7708

Table 3: Values used to model the background [1]

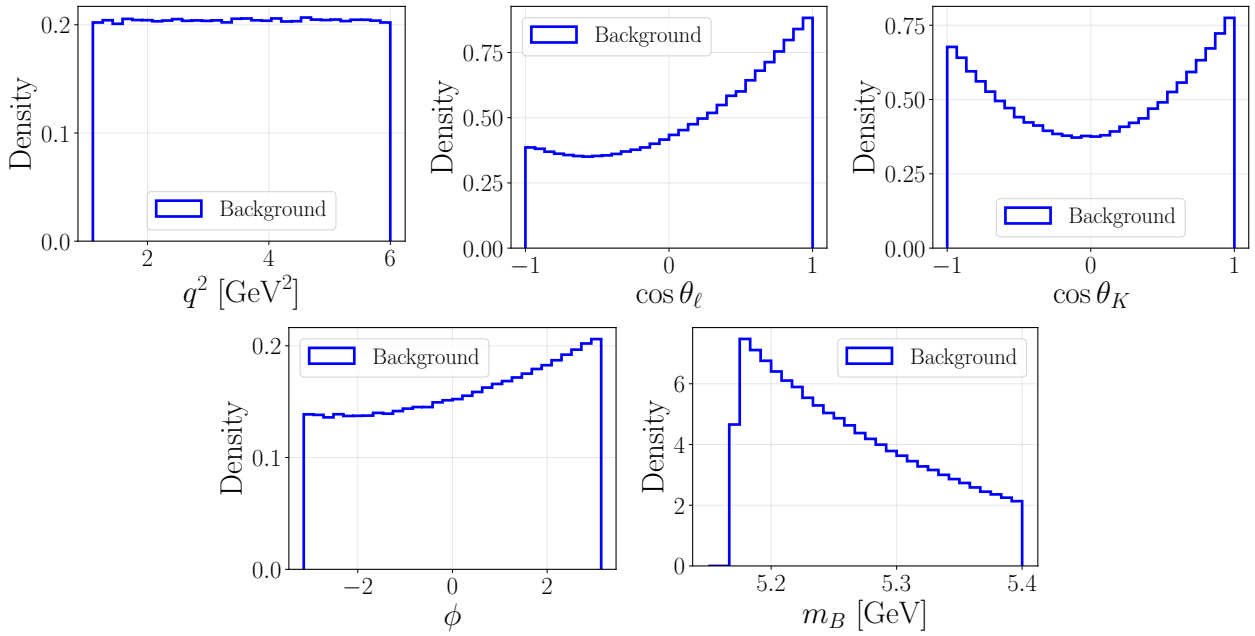


Figure 7: Probability density functions of the observables for background events

3.2.3 Resolution

The resolution models the finite precision of the LHCb detectors by smearing the EOS distributions very slightly. This effect is not dominant and could be neglected. However, it is also useful to add a small noise on the simulated data to improve the robustness of the neural network as discussed in Section 2.5.3. The resolution is modelled by adding a small Gaussian noise of amplitude σ to the simulated data and clipping values outside the observable ranges indicated in Table 1. No noise is added to m_B as its distribution already corresponds to a stochastic variation.

Parameter	Value
$\sigma_{q^2}^{\text{core}}$ [GeV^2]	0.05
$\sigma_{q^2}^{\text{tail}}$ [GeV^2]	0.10
f_{tail}	0.10
$\sigma_{\cos\theta_\ell}$	0.02
$\sigma_{\cos\theta_K}$	0.02
σ_ϕ	0.02

Table 4: Parameters of the detector resolution model

For the invariant mass squared q^2, a fraction f_{tail} of the points uses a higher noise amplitude σ_{tail} to describe the behaviour of a tail. Table 4 presents the values of the parameters used. These values have not been determined rigorously by a previous study, but have been chosen to minimally impact the distributions while improving robustness.

3.3 Data–simulation agreement

Including the newly implemented detector effects, the distributions of the observables are now closer to the LHCb toy data as shown in Figure 8.

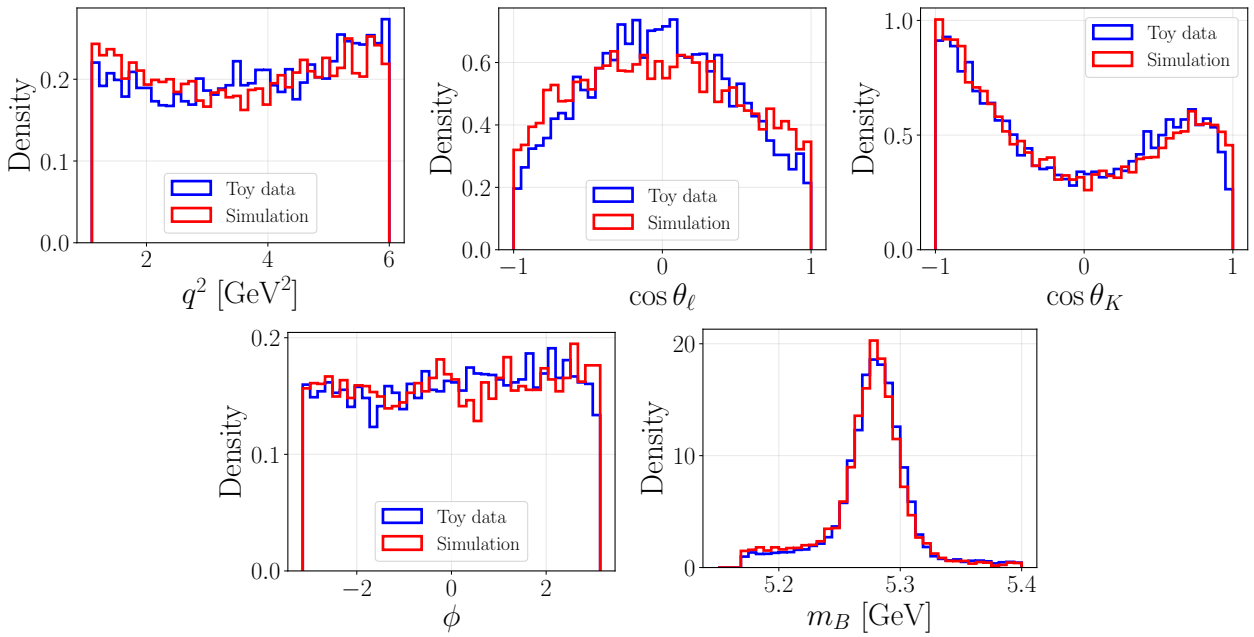
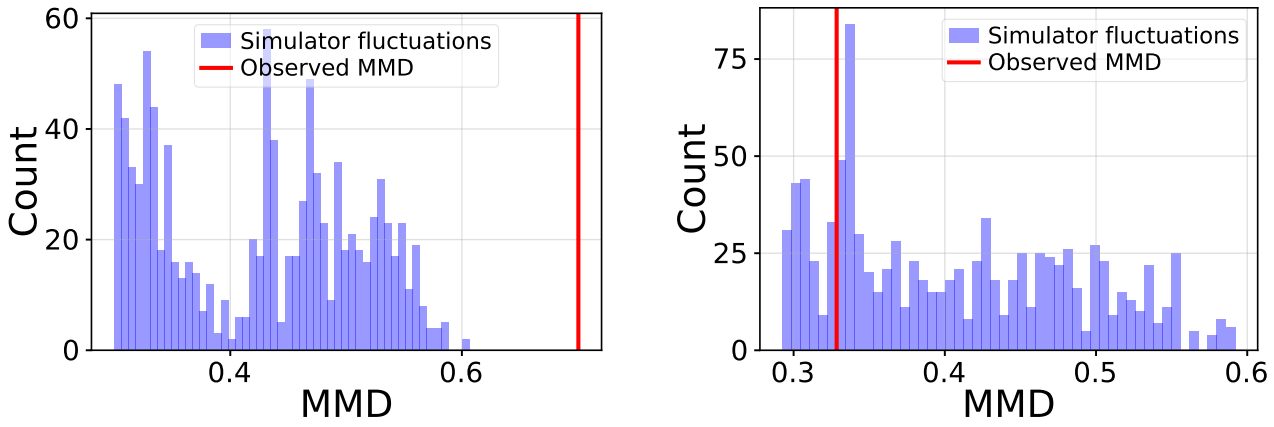


Figure 8: Distributions of the observables q^2, $\cos\theta_\ell$, $\cos\theta_K$, ϕ, and m_B generated by the simulator including detector effects for $C_9^{\text{fit}} \simeq 3.34$

To ensure that no significant discrepancy remains, it is important to test the misspecification, as described in Section 2.5.1. The results are shown in Figure 9. With the ideal simulator, the diagnostic fails as the red line is outside the histogram mass. However, when using the detector effects, the red line now lies within the distribution and the p-value of $0.82 \gg 0.05$ is now acceptable. It is important to note that the red line does not need to be in the middle of the blue histogram, it only needs to lie within it.



(a) (b)

Figure 9: Misspecification diagnostic (a) ideal simulator, p-value: $0 \ll 0.05$ (b) simulator including detector effects, p-value: $0.82 \gg 0.05$

3.4 Data preprocessing

Once the simulated data is satisfying, it needs to be formatted to improve the stability and efficiency of the neural network training. The periodicity of ϕ is enforced by mapping it to $\cos\phi$, $\sin\phi$ and the data is normalized using the following method. For a dataset of N_s samples, containing each N_e events of dimension d, the mean $\mu \in \mathbb{R}^d$ and standard deviation $\sigma \in \mathbb{R}^d$ are computed keeping the observables independent, but mixing the events from different samples. Parameter normalisation is also implemented but not used for simplicity. This has no significant impact, as only a single parameter is inferred and its range $[3, 5]$ is of order 1. Enabling this feature would improve the training stability if several parameters were inferred simultaneously. With this preprocessing, the neural network does not need to learn the different scale of the observables and the periodicity of ϕ.

4 Simulation-Based Inference

4.1 Neural Network Architecture

To perform a SBI, the neural network needs to have a specific architecture. In the context studied here, a sample X is composed of events x_i independently drawn from the same distribution. The first layers of the neural network must enforce the permutation invariance of the events in a sample. This is achieved by using a DeepSet encoder. It calculates a summary z of a sample using the formula

$$z(X) = \rho\left(\frac{1}{N_e}\sum_{i=1}^{N_e}\varphi(x_i)\right)$$

The function φ is a small neural network applied to events. The results are grouped in a permutation invariant mean and passed to a second neural network ρ to output the final summary. This summary is then transmitted to the rest of the neural network.

The second part of the network is a Neural Spline Flow (NSF). It allows the modelling of the complex distribution $p(\Theta \,|\, z)$ through a sequence of spline transformations $f = f_K \circ ... \circ f_1$ applied to a base standard Gaussian distribution $y \sim p_y(y) = \mathcal{N}(0, 1)$. The posterior is then given by

$$p(\Theta \,|\, z) = p_y(f^{-1}(\Theta; z))\left|\det \frac{\partial f^{-1}}{\partial \Theta}\right|$$

Here monotonic piecewise rational quadratic splines are used. They each correspond to a smooth function f_i defined on subintervals $[y_{i,j}; y_{i,j+1}]$ by

$$f_i(y) = \frac{a_{ij}y^2 + b_{ij}y + c_{ij}}{d_{ij}y + e_{ij}}$$

where the coefficients of the polynomials and the bounds of the intervals are determined by the neural network. These splines are used because they are flexible, analytically invertible and numerically stable.

To simplify the implementation, the Python library SBI [5] has been used. The Table 5 gives the SBI objects and values of the parameters used. SBI uses the learning optimizer Adam to ensure a stable and efficient convergence. The encoder activation function is fixed by SBI to ReLU. The NSF activation function is ReLU but can be set to SiLU or GeLU. These functions are plotted in Figure 10 and defined in Equation (1) [8]. The event encoder φ and the set embedding ρ have each 2 layers of 64 neurons. The Neural Spline Flow consists of 10 transformations, each defined on 8 subintervals. Each transformation has a layer of 128 neurons. The choice of that architecture will be discussed in Section 6.3.

Component	Class / Type	Parameters
Prior	BoxUniform	[3, 5]
Event encoder φ	FCEmbedding	layers: 2, hidden neurons: 64, output dimension: 64
Set embedding ρ	PermutationInvariant-Embedding	layers: 2, hidden neurons: 64, output dimension: 128
Aggregation	Mean	–
Density estimator	Neural Spline Flow (NSF)	10 transforms f_i, 8 bins $[x_{i,j}, x_{i,j+1}]$
Flow hidden layers	Fully connected	hidden neurons: 128
Flow activation function	ReLU (or SiLU, GeLU)	–
Encoder activation function	ReLU	–
Inference method	Neural Posterior Estimation (NPE)	–
Learning optimizer	Adam	–

Table 5: Neural network architecture and hyper-parameters used for SBI

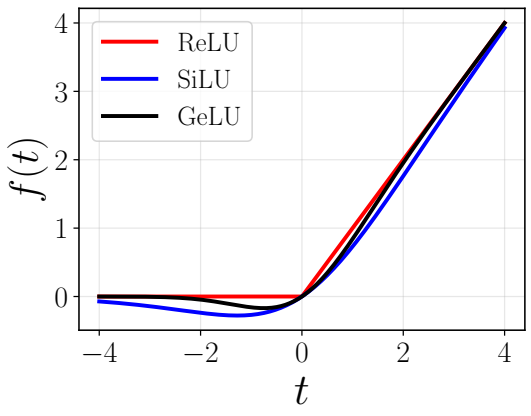


Figure 10: Comparison of ReLU, SiLU and GeLU

$$\begin{aligned}
\mathrm{ReLU}(x) &= \max(0, x) \\
\mathrm{SiLU}(x) &= \frac{x}{1 + e^{-x}} \\
\mathrm{GeLU}(x) &= \frac{1}{2}x\left(1 + \mathrm{erf}\left(\frac{x}{\sqrt{2}}\right)\right) \\
\mathrm{erf}(z) &= \frac{2}{\sqrt{\pi}}\int_0^z e^{-t^2}\mathrm{d}t
\end{aligned} \tag{1}$$

ReLU is the fastest to compute. However, its zero derivative for negative entries can lead to dead neurons which are no longer updated, and decrease the capacity of the neural network. Using SiLU reduces that issue but makes the computations slower. Finally, GeLU is even slower but allows a better expressivity.

4.2 Performance evaluation and improvement

It is important to be able to ensure that the inference uncertainty is dominated by the intrinsic uncertainty of the observed sample rather than by a limited training dataset, an undertraining or an architecture not complex enough. Diagnostics have been implemented to assess the impact of the number of events N_e in a sample, of the number of samples N_s in the training dataset, of the duration of the training and of the architecture. These tests correspond to plots of the average uncertainty of many inferences as a function of the relevant variables. The duration of the training is characterized by the number of epochs, i.e. the number of times the neural network iterates over the entire dataset during the training.

4.3 Sequential Neural Posterior Estimation

In this study, the final goal is to perform a precise inference near a single true value $\Theta^\star$ associated with the observed LHCb sample $X^\star$. Thus, it is relevant to focus the capacity of the neural network near the value $\Theta^\star$ rather than on the entire prior. This is done by using a Sequential Neural Posterior Estimation (SNPE). The network is trained using values of Θ drawn from the prior as described in Section 2.4. An intermediate inference of $p(\Theta \mid X^\star)$ is done. New values of Θ are sampled from this posterior. New samples are generated for each value of the parameters using the simulator in order to form a new training dataset. The same neural network is then further trained on this new dataset to refine the performance around the true value $\Theta^\star$. This process can be repeated multiple times, referred to as "rounds". This method is relevant to improve the calibration and the predictions themselves around the true value.

4.4 Technical details

The code developed for this project is available on its GitHub page [9]. A large effort has been put in order to make the code modular and easy to improve.
One of the key features is the ability to save and load data files and neural network instances. The SBI library saves the dataset along with each instance of the neural network, taking a huge amount of storage space. To resolve this issue, a more convenient method was implemented. It allows training to be paused and resumed by saving the data files paths instead of the dataset itself. Models can also be loaded without the training dataset when only an inference is needed.

Many diagnostics were added to ensure the proper functioning of the simulator, the detector effects model and the neural networks. Furthermore, because of the EOS MCMC sampling described in Section 3.1, and the acceptance described in Section 3.2.1, generating data takes a very long time. Simulating a sample of $N_e = 10\,000$ events takes around 8 hours. It has been necessary to use powerful Imperial HEP batch machines managed by the workload management software HTCondor and to generate multiple data files in parallel. A final dataset of $N_s = 1\,500$ samples of $N_e = 10\,000$ events each was generated in approximately 2 weeks because of the waiting time for batch jobs. This dataset is satisfactory to explore the potential of SBI. Since $N_s = 1\,500$ is relatively small, the training of a neural network is much faster and takes around 1 hour. The slowness of the data generation also makes the implementation of model diagnostics more challenging. When possible, it is beneficial to generate the data beforehand and store it. But some diagnostics need simulated samples for specific values of the parameters, forcing the generation of new data. This can make diagnostics like Posterior Predictive Check (PPC) very

slow. Finally, using all the dataset at the same time would take too much memory. Thus, it is necessary to split and manage the data in smaller batches.

5 Results

This section presents the configurations used and the results obtained. Their outcomes will be discussed in Section 6.

5.1 Models with the ideal and complete simulators

The first neural network is trained with a dataset generated by the ideal simulator ignoring the detector effects. The training and validation losses are shown in Figure 11. The epoch with the lowest validation loss is selected for the following analysis. The Figure 12 presents an example of a posterior prediction given a simulated observed sample with a known true parameter. To ensure the proper functioning, this process is repeated many times to produce the Figure 27 in the Appendices. The second neural network is trained using the complete simulator. The two datasets for the two models each consist of $N_s = 1\,500$ samples containing $N_e = 10\,000$ events. Figure 13 illustrates the calibration of the neural networks on simulated samples as explained in Section 2.5.2. The robustness of the neural networks, explained in Section 2.5.3, is presented in Figure 14 and Figure 15. The Gaussian noise is added to the normalized data, so a noise amplitude of $\delta = 1$ corresponds to variations of the same order as the data itself.

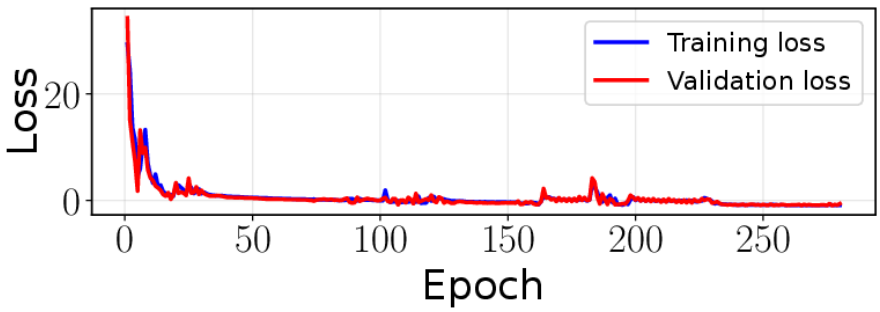


Figure 11: Training and validation losses during the training of the neural network with the ideal simulator

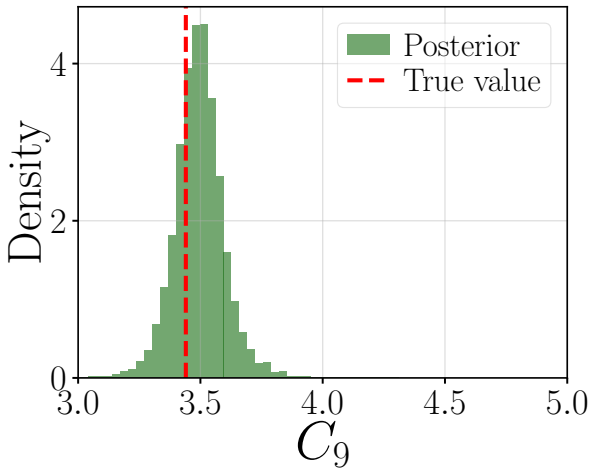
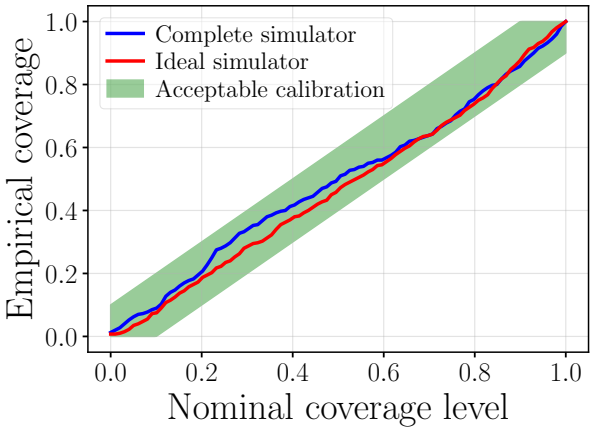
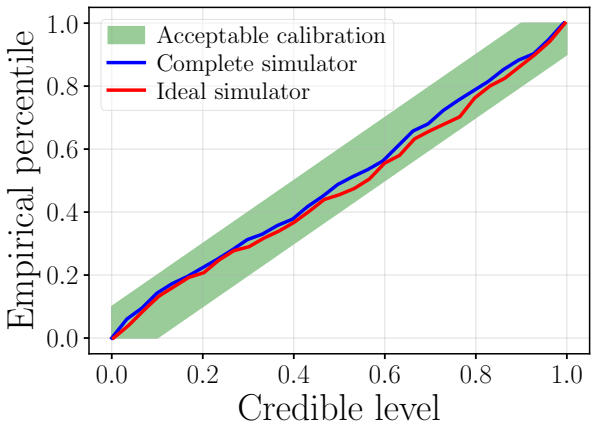


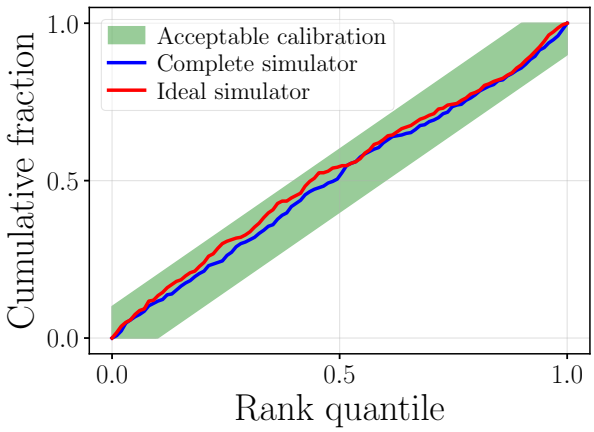
Figure 12: Example of a posterior inference given an observed simulated sample by the neural network using the ideal simulator. Estimator: $\hat{C}_9 = 3.49 \pm 0.09$, True value $C_9^\star = 3.4404$, Error: 1.4%



(a)

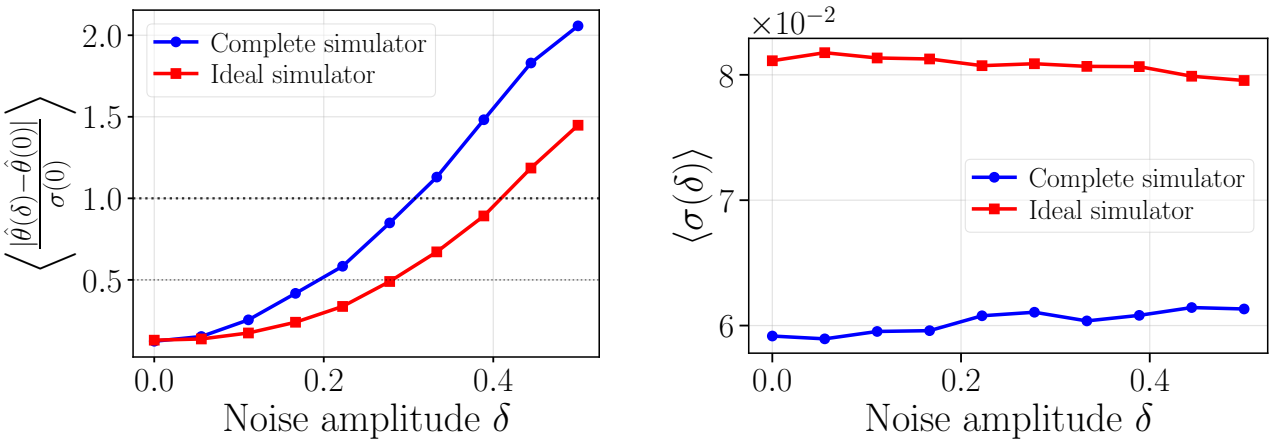


(b)



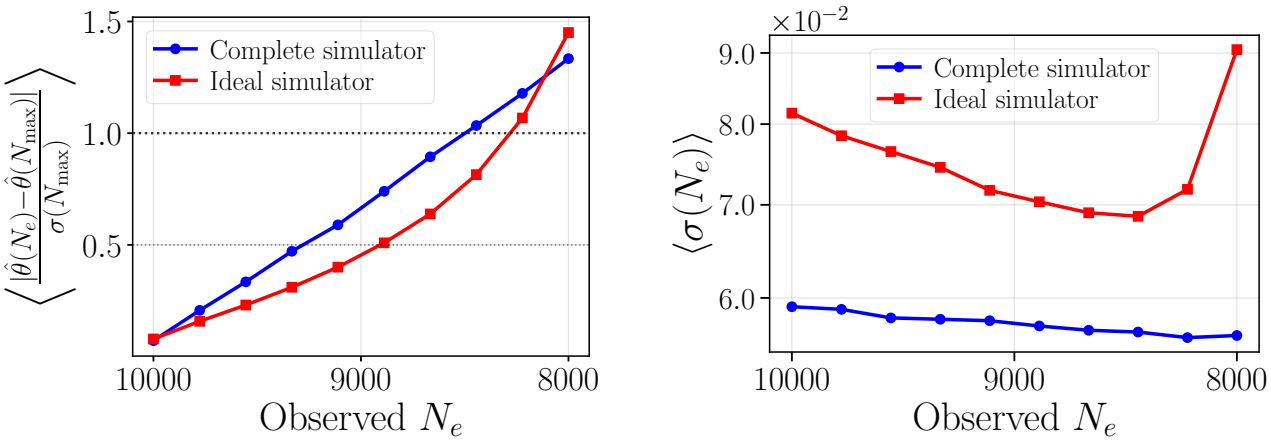
(c)

Figure 13: Calibration diagnostics of the neural networks trained with and without the detector effects (a) Expected Coverage Test (b) Test of Accuracy with Random Points (c) Simulation-Based Calibration



(a) (b)

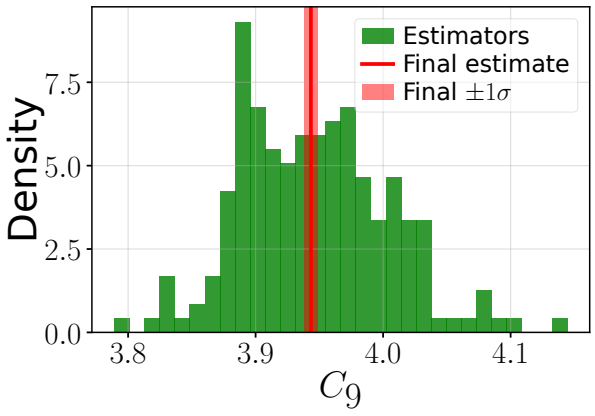
Figure 14: Robustness to noise of the neural networks trained with the simulators including and excluding detector effects (a) Estimator shift (b) Uncertainty evolution



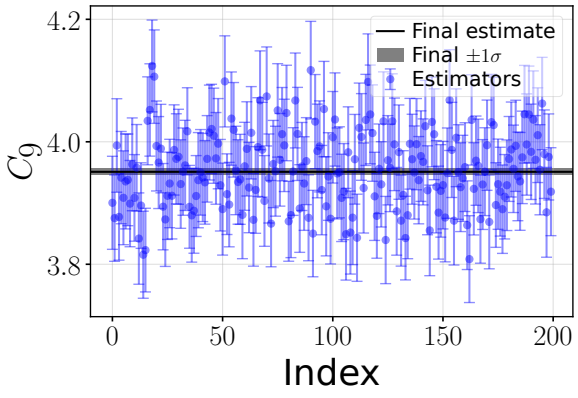
(a) (b)

Figure 15: Robustness to reduced number of events in the observed samples for the neural networks trained with the simulator including and excluding the detector effects (a) Estimator shift (b) Uncertainty evolution

A total of 200 inferences with different observed LHCb toy samples allow to estimate the value of C_9. The resulting estimators are shown in Figure 16. The same inferences are done for the neural network trained with the complete simulator and the results are presented in Figure 17.

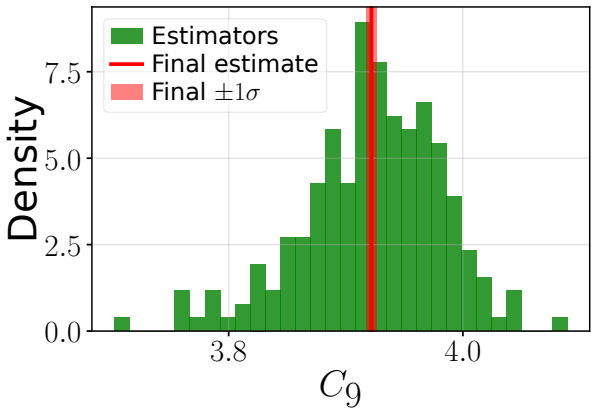


(a)

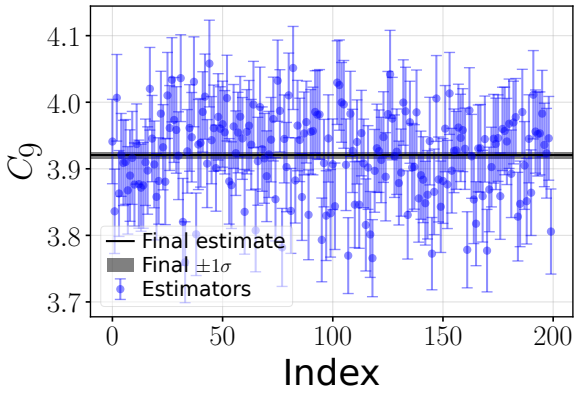


(b)

Figure 16: Final inference given observed LHCb toy samples for the neural network trained on ideal data (a) Estimators distribution (b) Estimators error-bars. Final estimate: $\hat{C}_9 = 3.950 \pm 0.005$



(a)



(b)

Figure 17: Final inference given observed LHCb toy samples for the neural network trained with the complete simulator (a) Estimators distribution (b) Estimators error-bars. Final estimate: $\hat{C}_9 = 3.920 \pm 0.005$

5.2 Impact of the number of events per sample

This subsection assesses the impact of the number of events per sample N_e on the average inference uncertainty. Neural networks using the complete simulator are trained with datasets of $N_s = 1\,000$ samples and number of events N_e varying between 100 and 10 000. The networks are trained until reaching their convergence, i.e. their validation loss minimum. These models are given observed samples with the same number of events per sample as what they were trained on. The average inference uncertainty is plotted on Figure 18.

5.3 Impact of the training dataset size

This subsection focuses on the impact of the number of samples N_s in the training dataset. The complete simulator including detector effects is used. The number of events per sample is fixed to $N_e = 10\,000$ and the number of samples N_s varies from 30 to 1 500. The average inference uncertainty is shown in Figure 19.

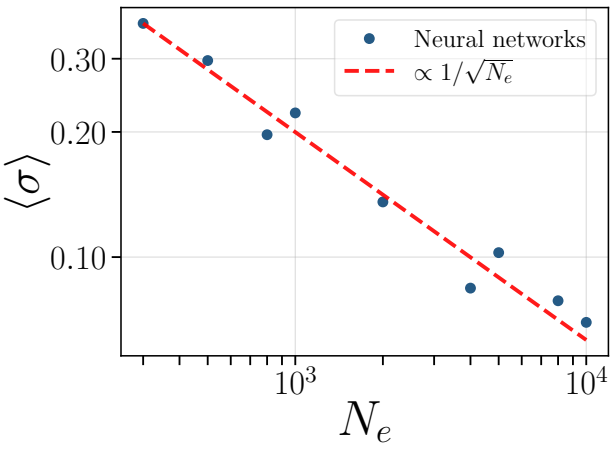


Figure 18: Average inference uncertainty as a function of the number of events per sample N_e in the training dataset and observed samples

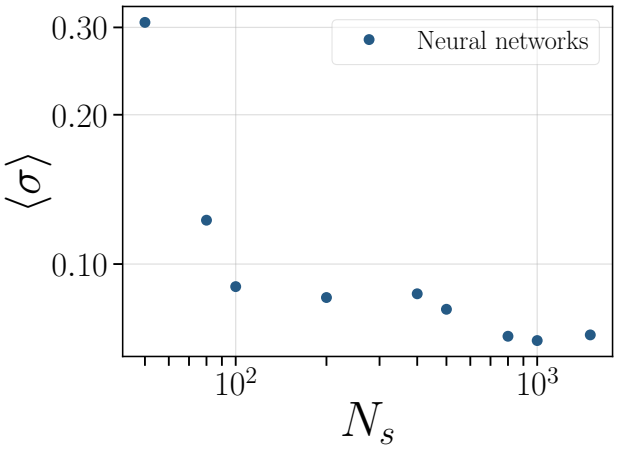


Figure 19: Average inference uncertainty depending on the number of samples N_s in the training dataset

5.4 Impact of the training duration

This subsection investigates the impact of under-training and overfitting on the inference. A neural network is trained with the complete simulator, $N_s = 1\,000$ and $N_e = 1\,500$. The training and validation losses as a function of the number of epochs are drawn in Figure 20. An epoch is an iteration over the entire dataset during the training. Figure 21 shows the progression of the average inference uncertainty for different number of epochs. It is also relevant to compare the neural network at three stages: at epochs number 10, 240 and 1 800. The Figure 22 compares their calibration and the Figure 23 their robustness.

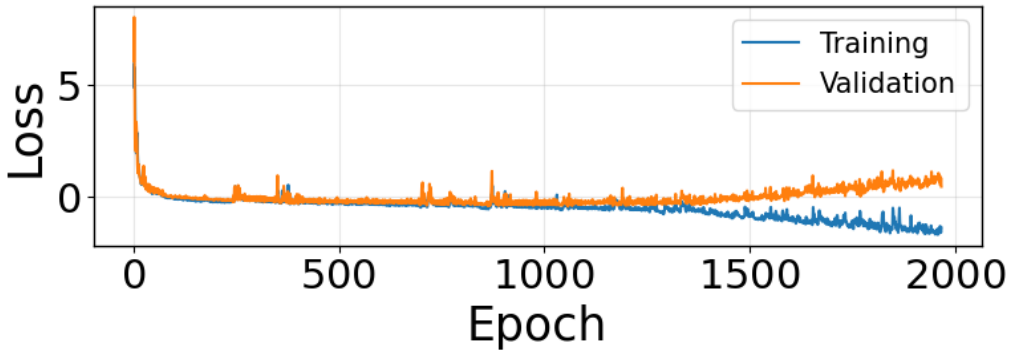


Figure 20: Training and validation losses during the training

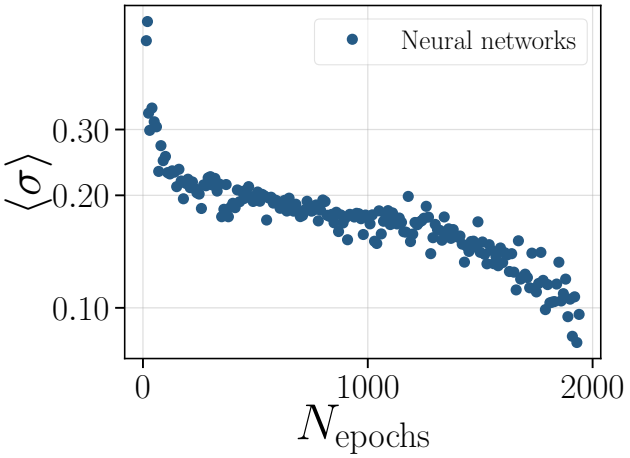


Figure 21: Average inference uncertainty of the neural network at different epochs

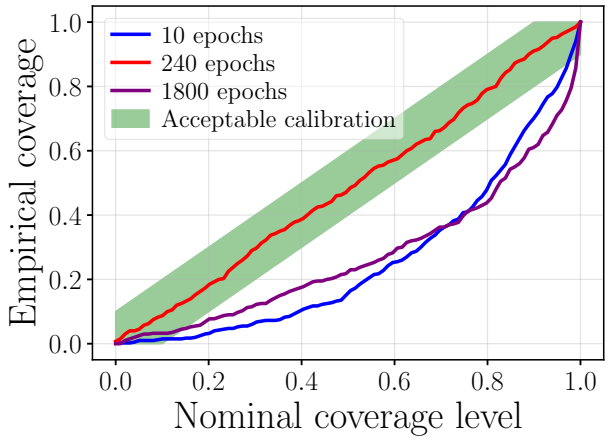
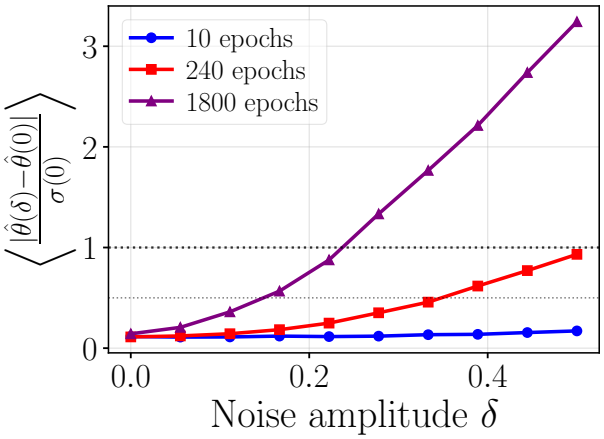
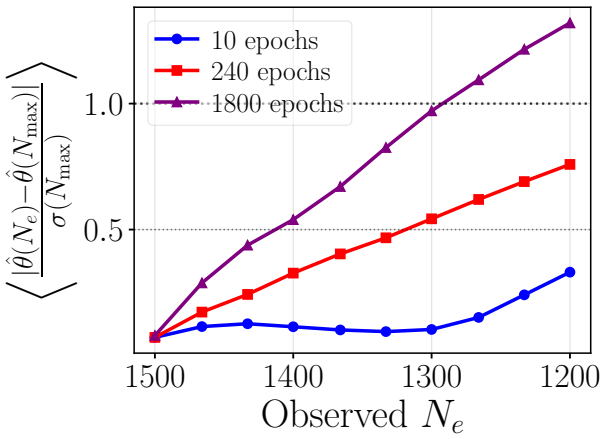


Figure 22: Expected Coverage Test of the neural network at epochs 10, 240 and 1800



(a)



(b)

Figure 23: Robustness of the neural network at different epochs (a) Robustness to noise (b) Robustness to a reduced number of observed events

5.5 Impact of the architecture

Finally, the impact of the neural network architecture is investigated. Multiple neural networks are trained with the complete simulator, $N_s = 1\,000$ and $N_e = 10\,000$. However this time, the configuration indicated in Table 5 is modified. Three neural networks use ReLU, SiLU and GeLU as their NSF activation functions. Another neural network is trained with the activation function ReLU and with 20 NSF layers instead of 10. Finally a fifth neural network is trained with only 5 NSF layers and ReLU. The average inference uncertainty of each model is indicated in Figure 24. The calibration and robustness of these models is compared in Figure 25 and Figure 26 respectively.

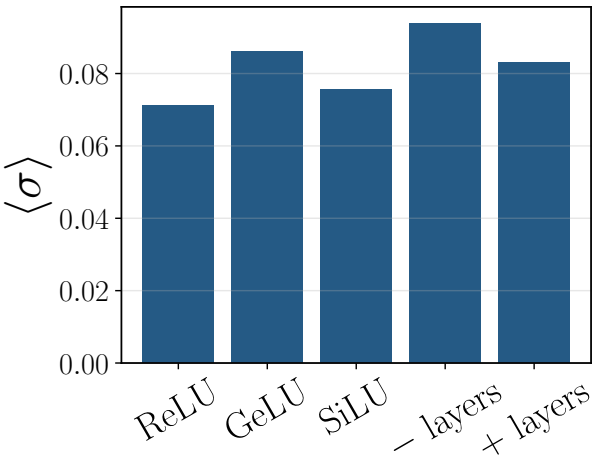


Figure 24: Average inference uncertainty for different neural network architectures. ReLU, GeLU, SiLU have 10 NSF layers, − layers has 5 NSF layers and uses ReLU, + layers has 20 NSF layers and also uses ReLU

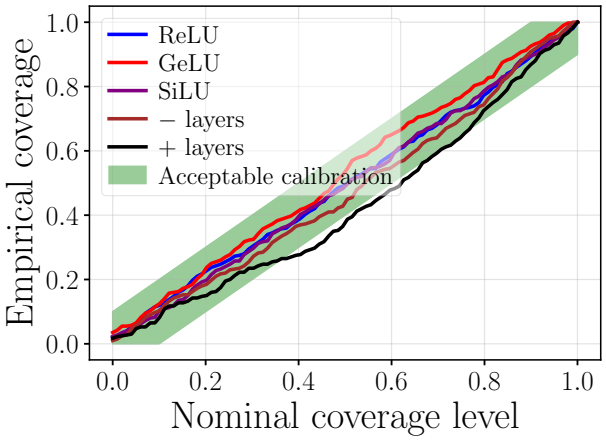
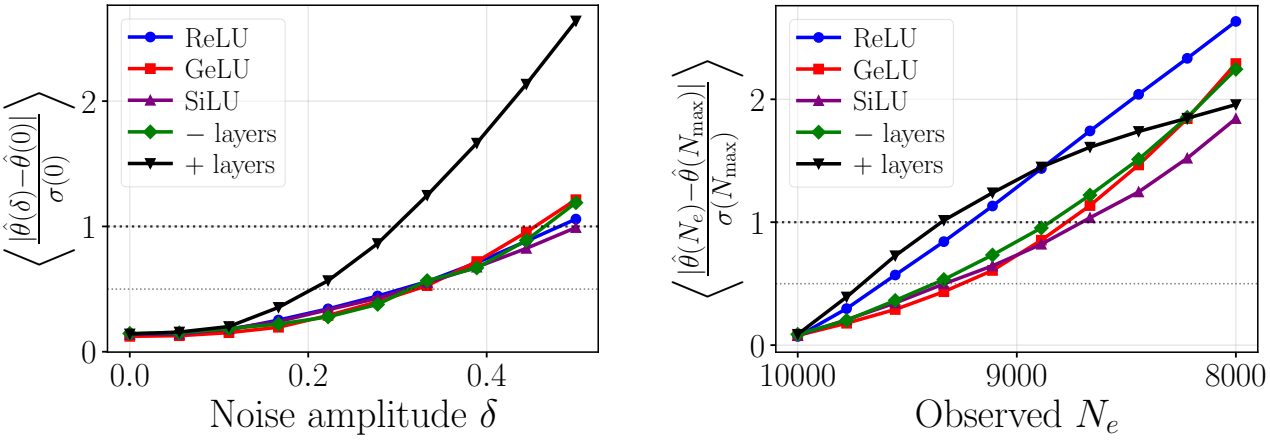


Figure 25: Expected Coverage Test of different neural network architectures. ReLU, GeLU, SiLU have 10 NSF layers, − layers has 5 NSF layers and uses ReLU, + layers has 20 NSF layers and also uses ReLU



(a) (b)

Figure 26: Robustness of different neural network architectures (a) Robustness to noise (b) Robustness to a reduced number of observed events. ReLU, GeLU, SiLU have 10 NSF layers, − layers has 5 NSF Layers and uses ReLU, + layers has 20 NSF layers and also uses ReLU

6 Discussion

6.1 Model with the ideal simulator

This first neural network was trained with the ideal simulator, ignoring detector effects. Figure 11 shows that the neural network learns most of the relationship between the observed sample and the parameter within approximately 40 epochs. The learning process then becomes significantly slower. The impact of the number of epochs will be discussed in Section 6.3. As illustrated by Figure 12, the inferred posterior is generally close to the true parameter that generated the observed sample. Here, the final estimator $\hat{C}_9 = 3.49 \pm 0.09$ differs by 1.4% from the true value $C_9^\star = 3.4404$. The narrow posterior width lead to a satisfactory estimator uncertainty that correctly includes the true value. The elements determining the width will be

discussed in Section 6.3. The proper behaviour of the neural network is confirmed over a larger number of inferences in Figure 27 in the Appendices.

The diagnostics in Figure 13 reveal a satisfactory calibration as the curves are close to the diagonals. Hence, the neural network can accurately infer a posterior from a simulated observed sample. The predictions are unbiased and the uncertainties are correctly estimated. Figure 15 indicates that the estimator shifts when the neural network is given a reduced number of events in the observed sample. For 9 000 observed events instead of 10 000, a bias of half the initial uncertainty is introduced. Moreover, the uncertainty decreases, leading to an overconfident and biased inference. Hence, the neural network is particularly sensitive to the number of observed events. Figure 14 shows that the estimator slowly shifts from its original value when a Gaussian noise is added to the observed sample. The shift can be neglected for noise amplitudes $\delta < 0.1$. However, the uncertainty stays approximately constant leading to an underestimation of the estimator uncertainty for higher noise amplitudes. If the real LHCb data deviates more from the simulations than a noisy observed sample with $\delta = 0.1$, the estimates will be biased and the uncertainty will be underestimated. This is expected, as Figure 9 in Section 2.5.1 demonstrated that the ideal simulator was misspecified.

Figure 16 shows the estimator distribution obtained for the inferences with LHCb toy data. The final estimate is found to be $\hat{C}_9 \simeq 3.950 \pm 0.005$. As explained in Section 2.3, reference values for the Standard Model prediction and fit results are $C_9^{\text{SM}} \simeq 4.27$ and $C_9^{\text{fit}} = 3.34^{+0.18}_{-0.16}$ respectively. Hence, the estimator has a relative error of approximately 7% with respect to the SM prediction and 18% with respect to the fit measurements. This inference suggests an overestimated disagreement of the fit measurements with the theoretical predictions, but it should not be trusted, since the misspecification diagnostic has failed.

6.2 Model with the complete simulator

This second model was trained using a simulator that includes the detector effects. A good calibration is depicted in Figure 13. Figure 15 shows that this neural network is also sensitive to the number of events in the observed samples as the uncertainty decreases and the estimator quickly shifts away when the number of events is reduced. A bias of around half the initial uncertainty is introduced when the number of observed events is decreased from 10 000 to 9 300. Figure 14 reveals that the estimator shift for a noisy observed sample can be neglected for a noise amplitude $\delta < 0.1$. For higher amplitudes, the shift starts becoming comparable to the uncertainty that remained roughly constant leading to a bias and underestimation of the uncertainty. Furthermore, the robustness is slightly worse than that of the ideal simulator model as the estimator shift increases faster. The addition of the resolution noise is expected to improve the robustness, however the acceptance and background contributions increase the complexity of the data, making the learning harder and reducing the robustness of the neural network.

Figure 17 presents the estimators obtained from the LHCb toy data inferences. The final estimate is $\hat{C}_9 = 3.920 \pm 0.005$ corresponding to an 8% error relative to the Standard Model prediction and a 17% error relative to the fit measurements. This time, the complete simulator is not misspecified as demonstrated by Figure 9 and the inference should be reliable. However, the final estimate is very close to the one calculated with the ideal simulator with a relative difference of only 0.8%. This can be explained by three main factors. First, the misspecification of the ideal simulator may introduce a random bias that happens to be small for the particular LHCb toy dataset considered. In this case, the complete simulator inference might be reliable. Secondly, it is possible that the information learned by the neural networks is not sensitive to

the inclusion of detector effects. However, this would suggest that the networks have only captured key features of the data and no finer details. Finally, this behaviour may indicate that the simulator remains insufficiently realistic. When given LHCb toy data, the neural networks may interpret small discrepancies with respect to their training data as meaningful features, leading to an incorrect posterior with a similar bias for the two models. The third explanation appears to be the most plausible, suggesting that the inferred estimators are not reliable. Indeed, the correct calibration of the neural networks indicates that they have learned non trivial features and a randomly small bias for the ideal simulator case is unlikely. Even though the complete simulator successfully passes the misspecification diagnostic, more stringent tests may reveal remaining discrepancies between the simulated data and the LHCb toy data that cannot be neglected.

The aim of this study is to explore the use of Simulation-Based Inference rather than to implement the final estimation of C_9. Inferring a value with errors of order 10-20% with the Standard Model prediction and the fit measurements using a simplified model of the detector effects already demonstrates the potential of this approach. However, it also illustrates its limitations as a very accurate simulator needs to be implemented to prevent any bias in the inference, and the quality of the simulator must be assessed using very precise diagnostics.

6.3 Comparison of neural network configurations

This section examines the impact of several factors on the average uncertainty, the calibration and the robustness of the neural networks.

Impact of the number of events per sample

Figure 18 shows that the average inference uncertainty follows the expected $1/N_e$ scaling with N_e the number of events per sample. Hence, the posterior width is mainly determined by the intrinsic uncertainty of the observed sample. Indeed, for well calibrated neural networks, the posterior widths only reflect the uncertainty in the observed samples.

Impact of the number of samples in the training dataset

The effect of the training dataset size N_s on the average uncertainty is shown in Figure 19. At low numbers of samples in the dataset, the uncertainty decreases quickly. The neural network is limited by the information present in the training dataset. The uncertainty reaches a plateau and starts decreasing much slower at approximately $N_s = 100$. Most of the relevant information has been learned by the network and new samples only help refine the information already learned. To obtain a reliable inference, it is important to reach and go beyond the beginning of the plateau to ensure that most of the relevant information has been learned by the neural network.

Impact of the duration of the training

The impact of the training duration is now investigated. Figure 20 shows the training and validation losses of a neural network. The small losses oscillations can be explained by two factors: the learning rate being set too high by the optimizer Adam and the model being trained and evaluated on small data batches. These oscillations play no role and can be ignored. Between the epochs 1 000 and 1 200, the validation loss starts to increase while the training loss continues to decrease past the previous plateau. This phenomenon is called overfitting, the neural network learns features specific to the training dataset, causing a deterioration of the inferences on the validation dataset. To obtain the best predictions, it is important to select the neural network version with the lowest validation loss rather than the training loss.

Figure 21 presents the average inference uncertainty as a function of the epoch. The uncertainty decreases rapidly during the first epochs. The neural network learns most of the relevant information and becomes more confident in its predictions. The average uncertainty then reaches a plateau, as it becomes limited by the intrinsic uncertainty in the observed samples. After epoch 1 000, the uncertainty starts to decrease again as the neural network begins to overfit the training dataset. The observed samples used to calculate the average uncertainty are not part of the training dataset. Thus, this lower uncertainty is associated with an underestimation of the intrinsic uncertainty and a deterioration of the calibration rather than with a better accuracy.

The calibration and robustness of the neural network at epochs 10, 240 and 1800 are tested in Figures 22 and 23. At epoch 10, the network is still learning important information and is not yet well calibrated. The Expected Coverage Test shows that the empirical coverage is lower than the confidence level. This is due to the neural network not being able to infer the correct value of the parameter. Indeed, under additional noise, the inference is unaffected, the estimator does not shift and the uncertainty stays constant. The shift is also very slow when the number of observed events is decreased. At epoch 240, the neural network has learned most of the relevant information and the calibration is satisfying. The estimator shifts very slowly under an added noise, this deviation can be neglected for noise amplitudes $\delta < 0.3$. As before, the uncertainty remains incoherently constant leading to an underestimated error, non negligible for higher amplitudes. The estimators shift on average by half the original uncertainty when the number of observed events is decreased only from 1 500 to 1 300. At epoch 1 800, the network loses its calibration and robustness because of overfitting. The sub-diagonal curve in the Expected Coverage Test is consistent with a global underestimation of the uncertainties. Under added noise or a reduced number of observed events, the estimators shift much faster. This loss of robustness is explained by the neural network learning fine details about the training dataset. Hence, both the calibration and robustness are better and more realistic when the validation loss is minimized.

Impact of the architecture

Figure 24 shows that the different architectures studied give roughly the same average inference uncertainty. The small variations of order 10% might be caused by the stochasticity of the training. Multiple neural networks with each architecture should be trained, and their results averaged to really be able to interpret such differences. Therefore, the uncertainty is mainly determined by the intrinsic uncertainty in the observed sample and not by the neural network architecture. This is because these models are all correctly calibrated and expressive enough. Hence, the configuration shown in Table 5 could be simplified without affecting the inferences. The model using ReLU as its NSF activation function seems to have a lower average uncertainty, but this is likely due to better training conditions. Indeed, multiple neural networks using ReLU were trained, and the best performing model was selected.
All the neural networks, independently of their architecture, are well calibrated as shown in Figure 25. Figure 26 reveals that all the neural networks have approximately the same robustness under an additional noise or a reduce number of observed events except for the neural network with more NSF layers under an additional noise. Indeed, adding more layers allows to learn a finer relation between the simulated data and the parameter values making the neural network more sensitive to small deviations and less robust.

Conditions for an optimal inference

These results allow to conclude the conditions needed for an optimal LHCb inference. The number of events in the observed samples needs to be maximised and match the quantity measured by the LHCb. The neural networks should not be given smaller observed samples than the samples in the training dataset. Furthermore, the training dataset needs to be large enough to capture most of the relevant information, with additional refinements. It does not need to be excessively large. The optimal epoch is the one minimizing the validation loss as it offers the best calibration and robustness. The architecture of the neural network does not play a major role as long as it is not overcomplicated. The one given in Table 5 can be simplified by reducing the number of layers and number of neurons per layer without affecting the performance. SiLU is a good compromise to prevent the dead neurons problem explained in Section 4.1 without slowing the calculations too much.

6.4 Possible improvements and further work

This section describes elements that could be improved or studied further.

Implementing the actual inference

To implement the real inference of C_9 using SBI, the robustness of the neural networks and the accuracy of the simulator need to be improved to prevent any bias and underestimated uncertainty. Under an additional noise or reduced number of observed events, the uncertainty should increase slowly and the estimator should remain stable or shift in a consistent way. None of the neural networks trained in this study follows this ideal behaviour. Several elements could potentially improve the robustness: increasing the amplitude of the resolution noise but keeping it physically plausible, introducing nuisance parameters making the noise and detector effects intensities vary for different samples, implementing more neural network regularizations like a weight decay or a dropout and simplifying the model architecture. This project uses Neural Posterior Estimation (NPE) as it is more intuitive, faster and better suited for diagnostics. However, other approaches like Neural Likelihood Estimation (NLE) and Neural Ratio Estimation (NRE) could also improve considerably the robustness. These methods do not directly infer the posterior, but other quantities from which the posterior can be deduced [5].
Likewise, it is important to improve the simulator by making the implementation of the detector effects more realistic. As mentioned in Section 3.2.1, only the acceptance coefficients from 2017, for high q^2 and B^0 have been used. Varying the coefficients corresponding with the ratio of events measured by the LHCb for each type could improve the simulation of the acceptance. Another element would be to implement correctly the S-wave contributions, explained in Section 2.5.3.

Improving the comparisons

This study is an experimentation of SBI as a tool to infer C_9 from the LHCb measurements. Before making the final inference, the comparisons made in this study can be verified with a more rigorous method. The results like the inference uncertainty should be averaged using multiple neural networks trained with the same configuration. This would make the results independent of stochastic variations, making them more credible. This might also reveal a difference between the different architectures studied. The impact of the number of layers and number of neurons should also be studied in more depth. This would allow to determine a better compromise between expressivity and simplicity. The number of layers described in Table 5 has probably been set too high as decreasing it did not make the predictions worse.

Potential further work

Despite the simulator and robustness issue, this study illustrates the potential of SBI as a new approach to infer the value of C_9 from LHCb data. The neural network is able to correctly infer the value of C_9 and its associated uncertainty for observed samples generated by the same simulator. Improving the simulator and the robustness of the neural network could extend this result to LHCb data and make a real inference of C_9 possible. The prediction could be refined using sequential methods described in Section 4.3. This approach has been implemented but could not be used due to time constraints [9].
As described before, it would also be interesting to improve the comparison of the model configurations and to explore deeper the impact of the architecture. Finally, this work could be extended by allowing the inference of multiple Wilson coefficients simultaneously. For example, the coefficient C_{10} also affects the distributions of the observables and could be inferred with C_9.

7 Conclusion

The Standard Model of particle physics, although extremely successful, is known to be incomplete. The B anomalies are an example of a discrepancy between its predictions and the LHCb measurements. The decay $B^0 \to K^{0*}\mu^+\mu^-$ is particularly interesting because of its sensitivity to potential contributions from new physics. The value of the Wilson coefficient C_9 is traditionally estimated using likelihood-based fits, but these methods are limited by required approximations and complex correlations between a large number of parameters. This work explored the use of Simulation-Based Inference as an alternative approach to estimate the value of C_9 directly from LHCb measurements.

A model of the detector effects was implemented to partially fix the misspecification of the ideal simulator. Calibration and robustness diagnostics were implemented to compare several neural network models and identify a suitable configuration. Estimates of C_9 obtained from LHCb toy data show relative deviations of order 15% with respect to the Standard Model prediction and fit measurements. However, the accuracy of the simulator and the robustness of the neural networks need to be improved for these estimates to be reliable, in order to prevent any bias and uncertainty underestimation.

Despite this issue, this project illustrates the potential of Simulation-Based Inference to improve the current estimations of C_9. With a more realistic simulator, improved robustness, sequential methods, and alternative SBI approaches such as Neural Ratio Estimation, an accurate inference of C_9 could be achieved. In the long term, such developments may help determine whether the observed tensions with the Standard Model predictions are due to underestimated uncertainties or signs of new physics.

Acknowledgements

I would like to express my sincere gratitude to Dr Mark Smith for the supervision of this project and his clear explanations of the underlying physics. His quick and detailed responses were very helpful to resolve any issues. I am particularly grateful for the balance between the independence and the support he provided.
This project gave me the opportunity to develop my understanding of particle physics and neural networks, which were largely new to me at the beginning.

I would also like to thank EPFL for allowing me to undertake an exchange year, and Imperial College London for hosting me and giving me the opportunity to undertake this project.

References

[1] LHCb Collaboration. A comprehensive analysis of the $B^0 \to K^{*0}\mu^+\mu^-$ decay, 2025. URL: https://arxiv.org/abs/2512.18053, arXiv:2512.18053.

[2] Michael McCann. Lecture 25: Beyond the Standard Model, 2026. Advanced Particle Physics course notes, Imperial College London.

[3] Wolfgang Altmannshofer and David M. Straub. New physics in $b \to s$ transitions after 2021. *European Physical Journal C*, 81(10):1–73, 2021. arXiv:2103.13370.

[4] LHCb Collaboration. Test of lepton universality using $B^0 \to K^{*0}\ell^+\ell^-$ decays — plots, 2017. CERN Document Server, accessed 12 April 2026. URL: https://cds.cern.ch/record/2264957/plots.

[5] SBI Developers. SBI Documentation, 2026. Accessed: 12 April 2026. URL: https://sbi.readthedocs.io/en/stable/.

[6] EOS Collaboration. EOS — flavor physics phenomenology software, 2024. Accessed: 11 April 2026. URL: https://eoshep.org/.

[7] Wikipedia Contributors. Crystal Ball function, 2026. Accessed: 11 April 2026. URL: https://en.wikipedia.org/wiki/Crystal_Ball_function.

[8] Wikipedia Contributors. Activation function, 2026. Accessed: 16 April 2026. URL: https://en.wikipedia.org/wiki/Activation_function.

[9] Nils Coutant. Simulation-Based Inference for Particle Physics, GitHub Repository, 2026. Accessed: 2026. URL: https://github.com/NilsCt/sbi-particle-physics.

Appendices

Definition of the crystal ball function [7]:

$$f(x;\alpha,n,\bar{x},\sigma) = N \cdot \begin{cases} \exp\left(-\dfrac{(x-\bar{x})^2}{2\sigma^2}\right), & \text{for } \dfrac{x-\bar{x}}{\sigma} > -\alpha, \\ A \cdot \left(B - \dfrac{x-\bar{x}}{\sigma}\right)^{-n}, & \text{for } \dfrac{x-\bar{x}}{\sigma} \leq -\alpha, \end{cases} \tag{2}$$

$$A = \left(\frac{n}{|\alpha|}\right)^n \cdot \exp\left(-\frac{|\alpha|^2}{2}\right),$$

$$B = \frac{n}{|\alpha|} - |\alpha|,$$

$$N = \frac{1}{\sigma(C+D)},$$

$$C = \frac{n}{|\alpha|} \cdot \frac{1}{n-1} \cdot \exp\left(-\frac{|\alpha|^2}{2}\right),$$

$$D = \sqrt{\frac{\pi}{2}}\left(1 + \operatorname{erf}\left(\frac{|\alpha|}{\sqrt{2}}\right)\right),$$

$$\operatorname{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} \mathrm{d}t$$

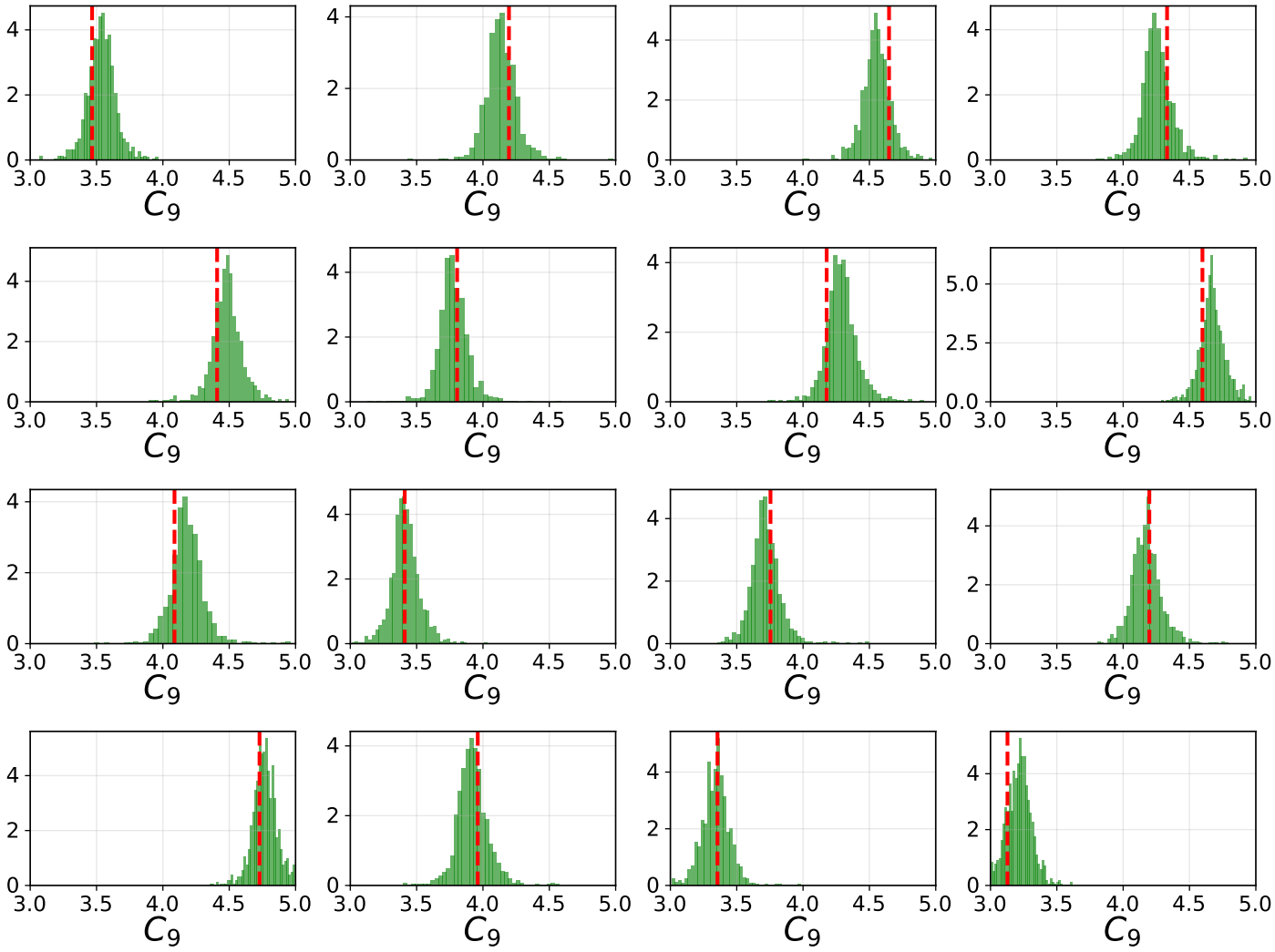


Figure 27: Many posterior inferences of a model using the ideal simulator. The green regions are the inferred posteriors, the red dashed lines indicate the true parameter values. The Y-axis gives the posterior probability density.