CentraleSupélec

Parametric Inference for Granulometer Data

Authors

Rayen BEN HASSEN
Alaa BOUATTOUR
Amadou DIALLO

Supervised by

Julien COLIN
Hervé FREZZA-BUET

May 2025

Contents

Chapter 1

Introduction

1.1 Background and Motivation

In many industrial contexts, ranging from pharmaceuticals to materials processing, a granulometer provides two-dimensional measurements (e.g., Feret diameters, projected area) of powder or grain samples via digital imaging. However, the true particles are three-dimensional. Accurately inferring 3D shape characteristics from 2D observations is crucial for predicting properties like flowability, packing density, and downstream processing performance. This project addresses the fundamental challenge of recovering the underlying 3D particle shape distributions from readily available 2D granulometer measurements.

1.2 Problem Statement

The core problem lies in the inherent ambiguity of projecting a 3D object onto a 2D plane: different 3D shapes and orientations can lead to similar 2D projections. This project investigates whether it is possible to infer the parameters of a statistical distribution of 3D particle shapes from observed distributions of 2D projections. We employ parametric inference methods, specifically Approximate Bayesian Computation (ABC) and Markov Chain Monte Carlo (MCMC), to tackle this inverse problem.

The overall workflow of this project is conceptually illustrated in Figure 1.1. The **Forward Model** simulates the measurement process of the granulometer (detailed in Chapter 2): starting from a parameterized distribution of 3D particle shapes (defined by their three semi-axes), each particle undergoes a random rotation before being projected onto a 2D plane, yielding observable 2D characteristics such as the major and minor axes of the projected ellipse (corresponding to maximum and minimum Feret diameters for ellipsoidal particles). The **Inverse Problem** aims to use these 2D observations to infer the parameters of the original 3D shape distribution. This is achieved through Bayesian inference methods (MCMC and ABC, discussed in Chapter 3), which combine a prior belief about the parameters with the information contained in the 2D data to produce a posterior distribution over the parameters.

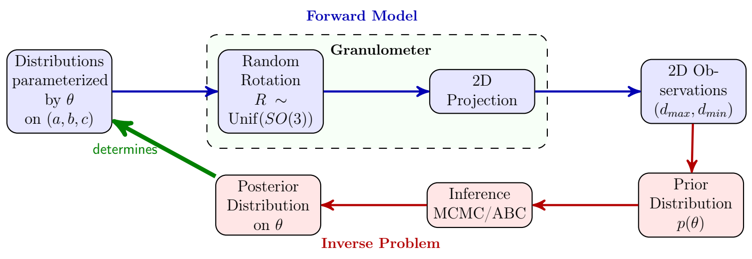

Forward Model
Granulometer
Distributions parameterized by θ on (a, b, c)
Random Rotation $R \sim$ Unif$(SO(3))$
2D Projection
2D Observations (d_{max}, d_{min})
Prior Distribution $p(\theta)$
Inference MCMC/ABC
Posterior Distribution on θ
determines
Inverse Problem

Figure 1.1: Workflow of the project

Chapter 2

Modeling

2.1 Particle Modeling: Ellipsoidal Particles

As a simplifying assumption to make the problem tractable, each particle is modeled as a 3D ellipsoid. An ellipsoid centered at the origin is defined by the equation $\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}=1$, where a, b, c are the lengths of the semi-axes. The shape of an ellipsoid is uniquely determined by the set of its semi-axis lengths $\{a, b, c\}$. However, without imposing an order, different permutations of the same three lengths (e.g., (a, b, c), (b, a, c), (c, b, a), etc.) would all describe the same physical ellipsoid shape. To ensure a unique representation for each distinct ellipsoid shape in our model and avoid such ambiguities in the parameter space, we impose the constraint $a \geq b \geq c > 0$. This ordered triplet (a, b, c) then uniquely defines a specific ellipsoid shape with positive semi-axes.

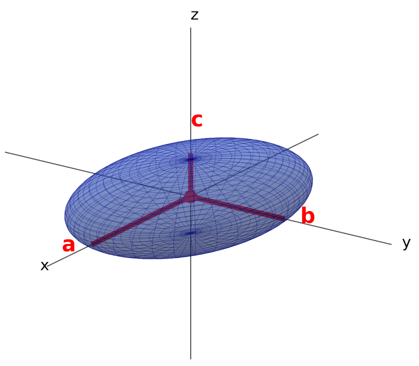


Figure 2.1: Notation for the ellipsoid semi-axes (a, b, c).

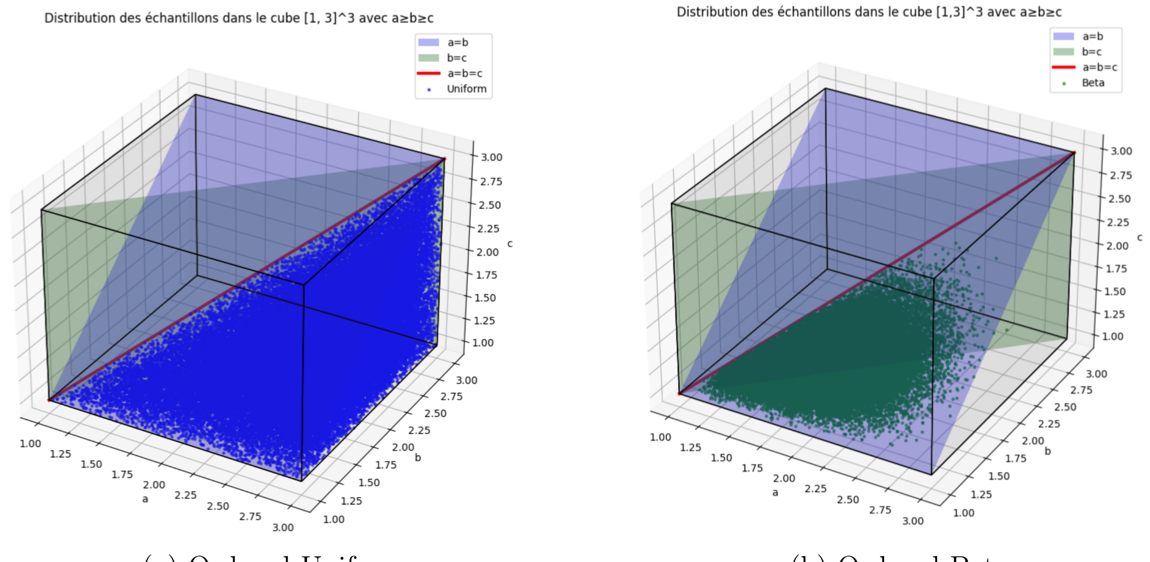
2.2 Generation of Particles

To simulate a collection of particles with varying shapes, the 3D axes (a, b, c) for each particle are sampled from specific probability distributions, while respecting the constraint $a \geq b \geq c$. We explore three different parametric models for the distribution of (a, b, c):

2.2.1 Ordered Uniform and Ordered Beta

In the Ordered Uniform model, three values are drawn independently from a uniform distribution over a specified interval $[u, u + l]$ and then sorted in descending order to obtain (a, b, c). While the parameters u and l control the scale and range of the particle sizes, the uniform distribution itself imposes a fixed distribution shape within that range, limiting the variety of particle elongations and flattenings that can be modeled.

In contrast, the Ordered Beta model offers greater flexibility. Three variates are drawn from a Beta distribution, rescaled to the interval $[u, u + l]$, and then ordered. The Beta distribution is characterized by two shape parameters, α and β. By adjusting α and β, we can model a wide variety of distribution shapes (e.g., skewed, U-shaped, symmetric), allowing for a richer representation of the underlying distribution of particle elongations and flattenings compared to the fixed shape of the uniform distribution.



(a) Ordered Uniform (b) Ordered Beta

Figure 2.2: Joint distributions of axes (a, b, c) generated under the Ordered Uniform and Ordered Beta sampling schemes.

2.2.2 Log Norm-Beta Construction

The LogNorm-Beta construction provides an alternative and flexible way to generate correlated axes, particularly suited for distributions where the largest axis may follow a skewed distribution. The process is conceptually outlined in Figure 2.3. The largest axis a is sampled from a log-normal distribution, i.e., $\log(a) \sim \mathcal{N}(\mu, \sigma^2)$. The ratios of successive axes, $r_1 = b/a$ and $r_2 = c/b$, are sampled independently from Beta distributions, $r_1 \sim \text{Beta}(\alpha_1, \beta_1)$ and $r_2 \sim \text{Beta}(\alpha_2, \beta_2)$. The axes b and c are then determined sequentially as $b = a \cdot r_1$ and $c = b \cdot r_2$. This construction inherently satisfies the ordering $a \geq b \geq c$ and allows for control over the overall size (via the log-normal parameters) and the distribution of axis ratios (via the Beta parameters).

A key advantage of this construction is its modularity and richness. The distribution for the largest axis a is not restricted to the log-normal distribution; it can be replaced by any distribution on $\mathbb{R}^+$ (e.g., Gamma, Exponential, Weibull) to better match the characteristics of the particles being studied. This provides a powerful framework for modeling a wide variety of 3D particle shape distributions.

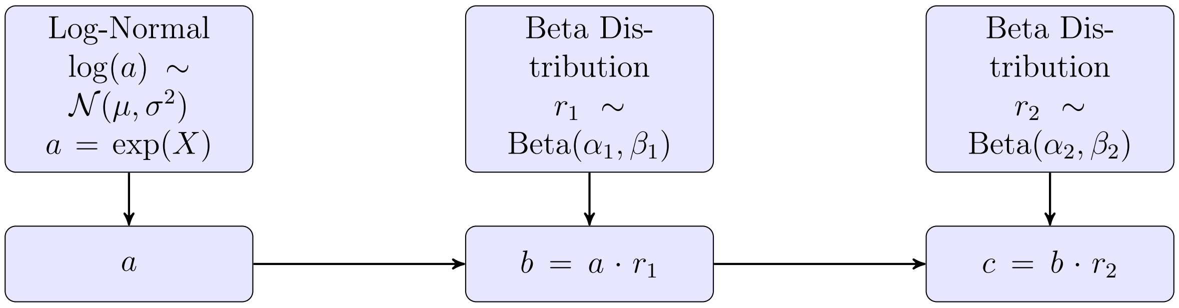

Log-Normal $\log(a) \sim \mathcal{N}(\mu, \sigma^2)$ $a = \exp(X)$
Beta Distribution $r_1 \sim \text{Beta}(\alpha_1, \beta_1)$
Beta Distribution $r_2 \sim \text{Beta}(\alpha_2, \beta_2)$
a
$b = a \cdot r_1$
$c = b \cdot r_2$

Figure 2.3: Flowchart of the LogNorm-Beta sampling process. This construction guarantees the ordering of the semi-axes and is generalizable by choosing different distributions for the largest axis a.

2.3 Granulometer Measurement Simulation

The process of measuring particles with a granulometer is simulated by applying a random rotation to each generated 3D ellipsoid and then projecting it onto a 2D plane.

2.3.1 Random Rotation

A critical aspect of simulating the granulometer measurement is accurately modeling the random orientation of particles as they are observed. This requires generating random rotation matrices that are uniformly distributed on the Special Orthogonal group SO(3). While this might seem straightforward, several distinct methods exist, each with different properties regarding statistical uniformity and computational efficiency. In this project, we implemented and evaluated three common approaches:

- **QR Decomposition:** This approach generates a 3x3 matrix with independent entries drawn from a standard Gaussian distribution. A QR decomposition is then performed, followed by adjustments to the resulting orthogonal matrix to ensure it belongs to SO(3). For more detailed information on this method (and more generally on random rotation matrix generation), see this paper by Maris Ozols (2009).

- **Axis-Angle:** This method relies on sampling a random rotation axis uniformly from the unit sphere and a corresponding rotation angle, typically sampled in $[0, \pi]$. The rotation matrix is subsequently constructed using the well-known Rodrigues' rotation formula.

- **Quaternion:** This technique involves drawing four independent standard Gaussian variables. These are then normalized to form a unit quaternion, which lies on the 3-sphere S^3 and uniquely represents a 3D rotation. The final 3x3 rotation matrix is derived from this unit quaternion.

We compared these three methods to determine which best suited our simulation needs in terms of generating truly uniform rotations and minimizing computational cost. The assessment of uniformity involved rotating a fixed point multiple times and analyzing the distribution of the resulting points on the unit sphere; a uniform distribution on SO(3) implies, for instance, that the z-coordinates of these rotated points should be uniformly distributed in the interval $[-1, 1]$. Computational performance was evaluated by timing the generation of a large ensemble of rotation matrices.

The key findings from this evaluation are presented in Figure 2.4. The empirical distribution of rotated points, visualized alongside a theoretical uniform distribution and assessed via the Kolmogorov-Smirnov test, indicated that both the QR Decomposition and Quaternion methods produced rotations closely approximating uniformity on SO(3). The Axis-Angle implementation, based on the statistical test, appeared less uniform in practice. From a computational perspective, the Quaternion method demonstrated a clear advantage, being significantly faster than the other two approaches for bulk generation of rotations.

Consequently, the Quaternion method was selected for implementing random particle rotations in our granulometer simulation due to its favorable balance of statistical uniformity and computational efficiency.

2.3.2 2D Projection

Following the random rotation, the 3D ellipsoid is projected onto the observation plane, which we consider to be the XY plane without loss of generality. This projection of a 3D ellipsoid results in a 2D ellipse. The lengths of the semi-axes of this projected ellipse, denoted as A and B, along with its orientation angle in the XY plane, can be computed

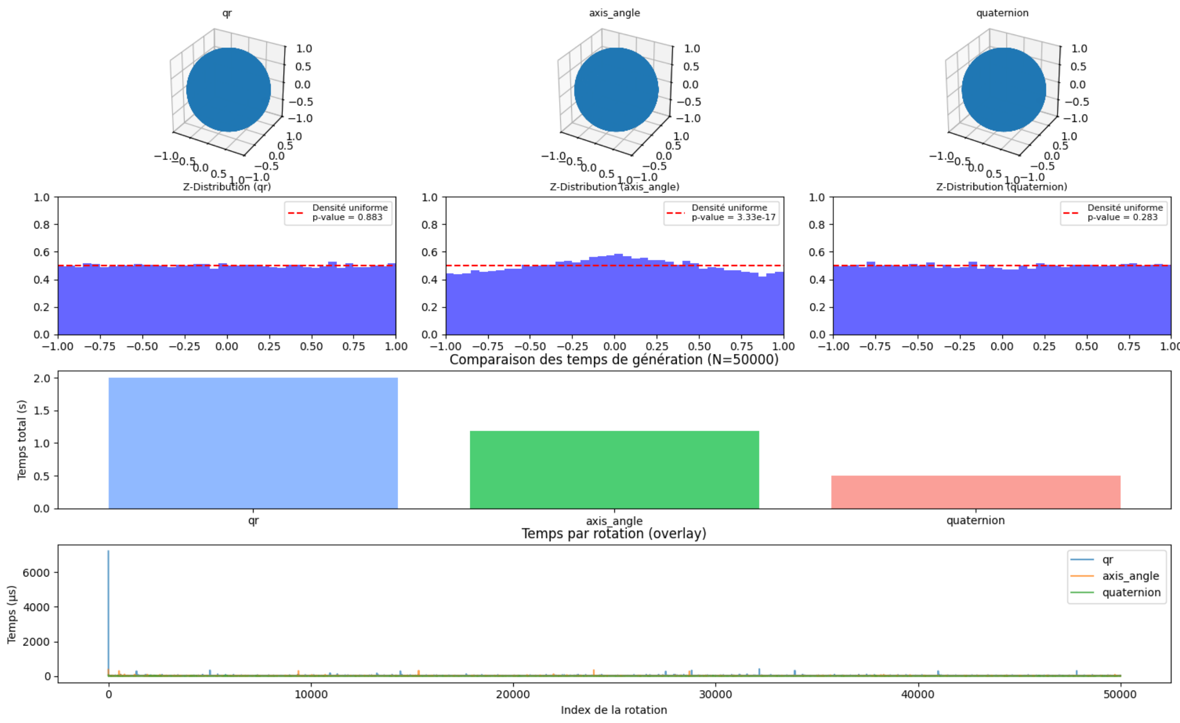


Figure 2.4: Comparison of different methods for generating random rotations in SO(3), showing uniformity assessment and performance benchmarks.

analytically. These (A, B) values represent the observed data from the granulometer for a single particle.

The analytical derivation begins with the equation of the ellipsoid in 3D space after rotation, given by $\mathbf{X}^T Q \mathbf{X} = 1$, where $\mathbf{X} = (X, Y, Z)^T$ and $Q = R^T D R$ is the quadratic form matrix, with $D = \text{diag}(1/a^2, 1/b^2, 1/c^2)$ containing the inverse squares of the original semi-axes and R being the rotation matrix.

To find the boundary of the projected ellipse on the XY plane, we consider the points on the ellipsoid where the tangent plane is vertical (i.e., parallel to the Z-axis). This condition is met by setting the discriminant of the quadratic equation of the ellipsoid in Z to zero. This process yields a 2D quadratic form in X and Y of the form $EX^2 + FXY + GY^2 = 1$. The coefficients E, F, and G are functions of the entries of the matrix Q.

The semi-axes of the projected ellipse (A, B) and its orientation are then determined from the eigenvalues and eigenvectors of the matrix $M = \begin{pmatrix} E & F/2 \\ F/2 & G \end{pmatrix}$. Specifically, the squared lengths of the semi-axes of the projected ellipse (A^2 and B^2) are the reciprocals of the eigenvalues of M. The direction of the eigenvectors corresponds to the orientation of the projected ellipse's axes. This analytical approach allows us to directly compute the observed 2D characteristics (A, B) from the 3D ellipsoid's shape (a, b, c) and its orientation R. A detailed derivation of the coefficients E, F, G and the computation of the projected axes can be found in Appendix A. The overall projection process is visualized in Figure 2.5.

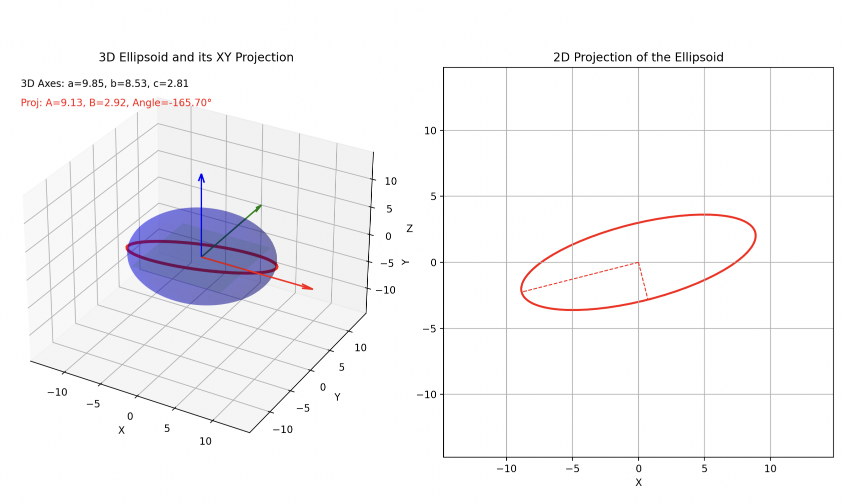


Figure 2.5: Rotation of a 3D ellipsoid and its projection onto the XY-plane, yielding observed axes (A, B).

It is important to note that this projection from 3D to 2D is not a one-to-one mapping, as different 3D ellipsoids and orientations can result in the same 2D projection. This non-bijectivity is the core challenge of the inverse problem we aim to solve.

Chapter 3

Bayesian Inference: MCMC

3.1 Introduction

In modern scientific and engineering applications, it is common to encounter problems where one must infer unknown parameters from observed data. When uncertainty and prior knowledge are essential considerations, the Bayesian framework provides a principled approach to parameter estimation. In the Bayesian paradigm, we represent our beliefs about unknown parameters through a prior distribution, and update these beliefs using the data via Bayes' theorem:

$$p(\theta \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \theta)\, p(\theta)}{p(\mathcal{D})},$$

where θ denotes the parameters of interest, $\mathcal{D}$ the observed data, $p(\theta)$ the prior, and $p(\mathcal{D} \mid \theta)$ the likelihood. The denominator $p(\mathcal{D}) = \int p(\mathcal{D} \mid \theta)p(\theta)d\theta$ is called the marginal likelihood or evidence.

3.1.1 The Challenge of Posterior Computation

In general, computing the posterior distribution analytically is infeasible because the normalizing constant $p(\mathcal{D})$ involves a high-dimensional integral. This renders many exact Bayesian computations (e.g., posterior means, variances, credible intervals) intractable. For example, the posterior mean is given by:

$$\mathbb{E}[\theta \mid \mathcal{D}] = \int \theta\, p(\theta \mid \mathcal{D})\, d\theta,$$

which again requires evaluating the full posterior. In simple models (e.g., conjugate priors), this is possible. But in most real-world applications—including the one studied in this report—numerical approximation is required.

3.1.2 Motivation for Monte Carlo Methods

Monte Carlo methods offer a general strategy for approximating expectations using random samples. If we can obtain samples $\theta^{(1)}, \dots, \theta^{(N)} \sim p(\theta \mid \mathcal{D})$, then:

$$\mathbb{E}[f(\theta)] \approx \frac{1}{N} \sum_{i=1}^{N} f(\theta^{(i)}).$$

This idea is powerful and underlies the use of simulation-based inference techniques in Bayesian analysis.

3.1.3 The Role of MCMC

When the posterior distribution cannot be sampled from directly, Markov Chain Monte Carlo (MCMC) methods are used to generate samples approximately distributed according to $p(\theta \mid \mathcal{D})$. These algorithms build a Markov chain whose stationary distribution is the desired posterior. Once the chain has mixed, subsequent samples can be used as approximate draws from the posterior.

MCMC becomes particularly valuable in scenarios where the parameter space is high-dimensional, the likelihood function is analytically intractable or computationally expensive to evaluate, or the posterior distribution does not admit a closed-form expression—as is often the case when working with simulation-based models.

In this project, we apply Bayesian simulation-based inference techniques to a shape inference problem. We primarily focus on Markov Chain Monte Carlo (MCMC) methods due to their flexibility and robustness. As an alternative likelihood-free approach, Approximate Bayesian Computation (ABC) rejection sampling was also explored; details on the ABC method can be found in Appendix B. The parameters θ in this problem describe the distribution of ellipsoidal axes that generate 2D projection data. The complexity of the forward model and the lack of an analytical likelihood make this problem a strong candidate for these simulation-based inference techniques.

3.2 Theoretical Foundations of MCMC

3.2.1 Markov Chains Review

MCMC relies on the theory of Markov chains to generate samples from a complex posterior. We briefly review key concepts.

Definition 1 (Markov Chain)**.** *A sequence of random variables $\{X_t\}_{t=0}^{\infty}$ is a **Markov chain** if:*

$$\mathbb{P}(X_{t+1} \mid X_t, X_{t-1}, \dots, X_0) = \mathbb{P}(X_{t+1} \mid X_t),$$

i.e., the future state depends only on the current state.

Definition 2 (Transition Kernel)**.** *The transition kernel $P(x, A)$ gives the probability of moving from state x to a measurable set $A \subset \mathcal{X}$:*

$$P(x, A) = \mathbb{P}(X_{t+1} \in A \mid X_t = x).$$

Definition 3 (Stationary Distribution)**.** *A probability distribution π is **stationary** for P if:*

$$\pi(A) = \int P(x, A)\, d\pi(x), \quad \forall A \subset \mathcal{X}.$$

A core idea of MCMC is to construct a transition kernel P such that the posterior $p(\theta \mid D)$ is its stationary distribution.

Irreducibility and Aperiodicity

For a Markov chain to converge to its stationary distribution, it must satisfy two fundamental properties: irreducibility and aperiodicity. Irreducibility means that every state in the parameter space can be reached from any other state with positive probability, possibly in multiple steps. This ensures that the chain is capable of exploring the entire space over time. Aperiodicity, on the other hand, guarantees that the chain does not become trapped in deterministic cycles; in other words, it can return to a given state at irregular intervals rather than only at fixed multiples. Together, these properties are essential for ensuring that long-run averages over the chain reflect expectations under the target distribution.

Ergodic Theorem

Let $f\colon \mathcal{X} \to \mathbb{R}$ be integrable. Then:

$$\frac{1}{T}\sum_{t=1}^{T} f(X_t) \xrightarrow{a.s.} \mathbb{E}_\pi[f(X)].$$

This theorm justifies using MCMC samples to estimate posterior expectations.

3.2.2 The Metropolis-Hastings Algorithm

The most widely used MCMC method is Metropolis-Hastings (MH). Given a target distribution $\pi(\theta) \propto p(D \mid \theta)p(\theta)$, the algorithm proceeds as follows:

1. Start at $\theta^{(0)}$.

2. At iteration t, propose $\theta^* \sim q(\cdot \mid \theta^{(t)})$.

3. Compute the acceptance probability:

$$\alpha = \min\left(1, \frac{p(D \mid \theta^*)p(\theta^*)q(\theta^{(t)} \mid \theta^*)}{p(D \mid \theta^{(t)})p(\theta^{(t)})q(\theta^* \mid \theta^{(t)})}\right).$$

4. Accept θ^* with probability α; otherwise, keep $\theta^{(t+1)} = \theta^{(t)}$.

Special Case: Symmetric Proposal

If $q(\theta^* \mid \theta) = q(\theta \mid \theta^*)$, the acceptance simplifies to:

$$\alpha = \min\left(1, \frac{p(D \mid \theta^*)p(\theta^*)}{p(D \mid \theta^{(t)})p(\theta^{(t)})}\right).$$

Pseudocode

```
for t in range(N):
    theta_star = propose(theta[t])  # From q
    log_accept_ratio = (
        log_likelihood(theta_star) + log_prior(theta_star)
        - log_likelihood(theta[t]) - log_prior(theta[t])
    )
    if log(rand()) < log_accept_ratio:
        theta[t+1] = theta_star
    else:
        theta[t+1] = theta[t]
```

Listing 3.1: Metropolis-Hastings pseudocode

3.2.3 Burn-in and Trace Plots

Before analyzing the posterior samples produced by MCMC, it is important to ensure that the Markov chain has reached its equilibrium distribution. Two common techniques used to assess this early-stage convergence are the application of a **burn-in period** and the use of **trace plots**.

Burn-in. The burn-in phase refers to the initial portion of the MCMC chain that is discarded before computing any statistics or visualizations. These early samples are often influenced by the initial state and may not reflect the true posterior distribution. By discarding the first $T_{\text{burn-in}}$ samples, we reduce the bias introduced by poor initialization. In practice, the choice of burn-in length is guided by visual and statistical diagnostics.

Trace plots. A trace plot shows the evolution of a single parameter across MCMC iterations and serves as a key visual diagnostic. It allows one to assess whether the chain has stabilized around a specific region, which is indicative of convergence. It also helps evaluate whether the chain adequately explores the parameter space, which reflects good mixing. Additionally, trace plots can reveal problematic behavior such as periodicity, slow drift, or multimodality, all of which may signal issues with the sampler or model specification.
An ideal trace plot should appear as a "fuzzy caterpillar" — stationary, fluctuating randomly around a fixed mean, with no visible trends or stickiness. Trace plots are typically generated for each parameter separately and are among the first diagnostics to inspect after a run.

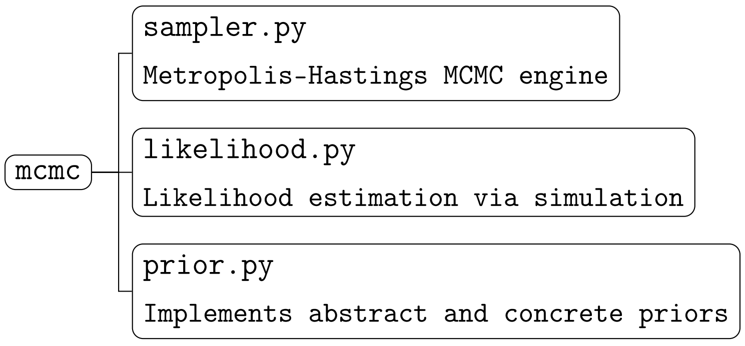
Examples of these diagnostics will be presented in the following section, once we introduce specific likelihood models and experimental setups.

3.3 Implementation in Practice

We now describe the software architecture and practical implementation of the MCMC sampler used throughout this study. The design emphasizes modularity, extensibility, and separation of concerns. All components are implemented in Python and grouped under a lightweight `mcmc` package.

3.3.1 Package Architecture

The core `mcmc` package is organized into three main components, as shown below:


mcmc
sampler.py
Metropolis-Hastings MCMC engine
likelihood.py
Likelihood estimation via simulation
prior.py
Implements abstract and concrete priors

Each component adheres to a common interface via abstract base classes (`Prior` and `Likelihood`), enabling plug-and-play flexibility.

3.3.2 Prior Class

The `Prior` abstract base class defines a general interface for evaluating the log-prior density of any parameter vector θ. One concrete implementation provided in the package is the `Uniform` prior, which supports arbitrary dimensionality by specifying independent lower and upper bounds for each parameter.

This class can optionally enforce ordering constraints of the form $\theta_1 < \theta_2 < \cdots < \theta_d$, which is useful when the model assumes a natural ordering of parameters (e.g., axis lengths). If a proposed parameter vector violates the bounds or the ordering constraint, the prior returns a log-probability of $-\infty$, ensuring that the sample is automatically rejected during the Metropolis-Hastings step.

```
prior = Uniform(bounds=[(0, 10), (0, 10)], enforce_order=True)
log_p = prior.log_prior(theta)
```

3.3.3 Likelihood Class

The `Likelihood` base class defines a general interface for evaluating the approximate log-likelihood $\log p(\mathcal{D} \mid \theta)$. Since the true likelihood is intractable, all subclasses rely on simulation-based approximations using a forward generator. This generator produces synthetic 2D projection data from a proposed 3D ellipsoid distribution parameterized by θ. Two concrete methods are implemented:

KDE-based Likelihood

The `KDE` subclass estimates the likelihood by:

1. Generating synthetic 2D data using the forward model.
2. Fitting a multivariate Gaussian KDE on the synthetic data.
3. Evaluating the observed data likelihood under this KDE.

This corresponds to a nonparametric likelihood estimate $\hat{p}_\theta^{\text{KDE}}(x)$ for each datapoint $x \in \mathcal{D}$, with the total log-likelihood:

$$\log p_{\text{KDE}}(\mathcal{D} \mid \theta) = \sum_{i=1}^{N} \log \hat{p}_\theta^{\text{KDE}}(x_i).$$

```
kde_lik = KDE(simulator=generator, param_names=["u", "v"])
log_lik = kde_lik.log_likelihood(theta, observed_data)
```

KL-based Likelihood

The `KL` subclass takes a different route: instead of fitting a density to the simulated data, it computes the reverse Kullback-Leibler divergence:

$$\log p_{\text{KL}}(\mathcal{D} \mid \theta) = -D_{\text{KL}}(\hat{p}_{\text{obs}} \,\|\, \hat{p}_\theta),$$

where:

- $\hat{p}_{\text{obs}}$: KDE fit on real observed data.
- $\hat{p}_\theta$: KDE fit on simulated data given θ.

This can be interpreted as a model selection criterion: parameters θ are preferred if the synthetic distribution closely matches the real one.

```
kl_lik = KL(simulator=generator, param_names=["u", "v"])
log_lik = kl_lik.log_likelihood(theta, observed_data)
```

Unlike the standard KDE-based method, the KL approach compares full distributions rather than pointwise likelihoods, which can make it more robust when samples are sparse or noisy.

Comparison and Trade-offs

The KDE-based likelihood is sensitive to local variations in the data and tends to perform well in well-sampled settings, where accurate pointwise density estimation is feasible. In contrast, the KL-based approach compares global distributional shapes, making it more robust to sampling noise but somewhat less interpretable as a traditional likelihood function.

Both methods are used interchangeably within the MCMC loop by simply swapping the likelihood object.

Example:

To test how the choice of likelihood approximation influences the inferred posterior distribution, we construct a simplified experiment using a mock simulator.

Synthetic data generation. The simulator samples from a uniform distribution parameterized by $\theta = (u, l)$, where values are drawn from $\mathcal{U}(u, u+l)$. Two random values are generated for each sample, and we retain their minimum and maximum as (Min, Max). This provides a simple 2D feature vector for inference.

The observed dataset is generated using the ground-truth parameters:

$$\theta^\star = (u = 3.0,\ l = 4.0).$$

MCMC inference. We infer the posterior distribution over $\theta = (u, l)$ using two different likelihood approximations: KDE and KL likelihoods.

Both samplers use the same prior, proposal, and runtime configuration.

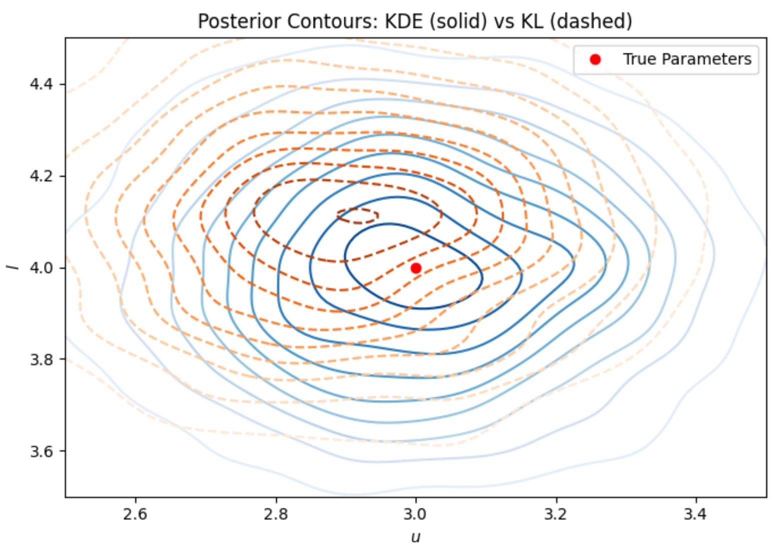


Figure 3.1: Posterior contours in parameter space (u, l). Solid lines represent the KDE-based posterior; dashed lines represent the KL-based posterior. The red dot indicates the ground truth parameters used to generate the observed dataset.

Interpretation. As shown in Figure 3.1, the KDE-based posterior (solid contours) is well-centered around the true parameter values, suggesting that this method yields an approximately unbiased estimate in this setting. In contrast, the KL-based posterior (dashed contours) exhibits a clear bias: its density peaks are noticeably shifted leftward, underestimating the value of u and misaligning with the ground truth. This discrepancy may be attributed to instability in the KL divergence when comparing empirical distributions in low-sample regimes or due to asymmetry in histogram-based approximations. Overall, while both methods produce smooth and unimodal posteriors, only the KDE formulation recovers the correct region with high confidence.

Note. The simulator used here is purely synthetic and does not represent a physical model. Its purpose is to validate the relative behavior of different inference strategies under controlled conditions.

3.3.4 Sampler Class

The `Sampler` class provides a general-purpose implementation of the Metropolis-Hastings algorithm, suitable for any parameter vector $\theta \in \mathbb{R}^d$. Internally, all computations are carried out in log-space to ensure numerical stability, especially when evaluating likelihoods and priors with potentially very small values. The proposal mechanism is a simple isotropic Gaussian random walk, whose scale can be tuned to achieve an appropriate acceptance rate.

In addition to generating samples, the sampler tracks the full chain history, the acceptance decision at each step, and summary diagnostics such as acceptance rate. The main entry point is the `metropolis-hastings` method which performs the MCMC algorithm.

```
sampler = Sampler(prior, kde_lik, data=data)
samples = sampler.metropolis_hastings(theta_init=np.array([3.0,
   6.0]), n_iter=n_iter, proposal_scale=0.1)
```

3.3.5 Modularity and Reusability

The implementation is intentionally modular. Likelihood functions can be swapped—between KDE, KL divergence, or even neural estimators—without modifying the core MCMC logic. Priors can encode custom constraints on the parameter space, such as monotonicity or bounded support, directly through their log-density. Because the interface between components is clearly defined, the sampler can be reused across a variety of inference problems, simply by providing new prior and likelihood objects that conform to the expected API.

3.3.6 Diagnostics and Outputs

To assess convergence and sampling behavior, we generate trace plots for each parameter, showing their evolution over iterations, and histograms summarizing the marginal posteriors. Quantitatively, we report the posterior mean and the acceptance rate, which reflects how efficiently the sampler is exploring the parameter space. These diagnostics are sufficient to detect convergence issues and assess sampling quality in most low-dimensional problems.

3.4 First Inference Example

Having established the architecture of our MCMC framework, we now turn to a first concrete inference example that connects directly with our target application: recovering the distribution of 3D ellipsoid axes from 2D projection data. This inverse problem arises in contexts such as powder studies using granulometry, where direct 3D observations are unavailable and only 2D measurements—here modeled as the maximal and minimal diameters of projected ellipsoids—are accessible.

3.4.1 Experimental Setup

We generate a synthetic dataset using a known uniform distribution over ellipsoid axes, with true parameters $(u, l) = (3.0, 7.0)$. This corresponds to axis lengths sampled uniformly from the interval $[u, u+l] = [3.0, 10.0]$, followed by random rotations in SO(3) and projection onto the 2D plane. The resulting dataset consists of $N = 150$ observations of projected diameters $(D_{\text{max}}, D_{\text{min}})$, forming our observed data.

The inference goal is to recover the values of u and l based solely on this 2D projection data. To do so, we define a uniform prior over the rectangle $[0, 10]^2$. Using (u, l) instead of allows us to avoid using the ordering constraint enforced between parameters.

We then perform two separate inference runs:

1. Using a **KDE-based likelihood**, where the observed and simulated data are compared via a kernel density estimate.

2. Using a **KL-divergence-based likelihood**, where binned histograms of observed and simulated data are compared.

Both inference pipelines use the same Metropolis-Hastings sampler, initialized at a random point.

3.4.2 Posterior Analysis

The trace plots in Figures 3.2 and 3.4 highlight a stark contrast between the behavior of the MCMC chains under the two likelihood formulations. In the KDE-based setting (Figure 3.2), the chains exhibit clear signs of convergence: after a brief initial phase, both parameters stabilize and fluctuate within a narrow range around their respective ground truth values. The trace is characterized by stair-like dynamics, indicating that proposals are occasionally rejected and plateaus form—yet the overall mixing remains adequate. This behavior reflects a well-behaved posterior surface with sufficient local gradient to guide the sampler.

In contrast, the KL-based trace plots (Figure 3.4) reveal erratic behavior. The chain drifts significantly, with no visible convergence phase and wide excursions across the parameter space. This is especially evident for the u parameter, which behaves like a near-random walk, crossing well beyond the region of interest. Such dynamics suggest poor identifiability under the KL-based likelihood or potentially an overly diffuse posterior induced by low sample counts or misaligned binning.

The histograms (Figures 3.3 and 3.5) corroborate this diagnosis. The KDE-based posterior distributions are unimodal and concentrated near the true values $(u, l) = (3.0, 7.0)$, with tight spread and consistent modes. On the other hand, the KL-based posteriors appear flat or even multimodal, indicating a lack of concentration and noisy posterior mass allocation.

Overall, while both approaches theoretically target the same posterior, the KDE-based likelihood yields a more reliable and interpretable posterior landscape under the current simulation settings.

3.4.3 Numerical Summary

Likelihood	**Posterior Mean** (u, l)	**RMSE**
KDE	(3.067, 6.684)	0.323
KL	(3.262, 8.108)	1.139

Table 3.1: Posterior means and RMSE relative to ground truth $(3.0, 7.0)$.

3.4.4 Figures

KDE

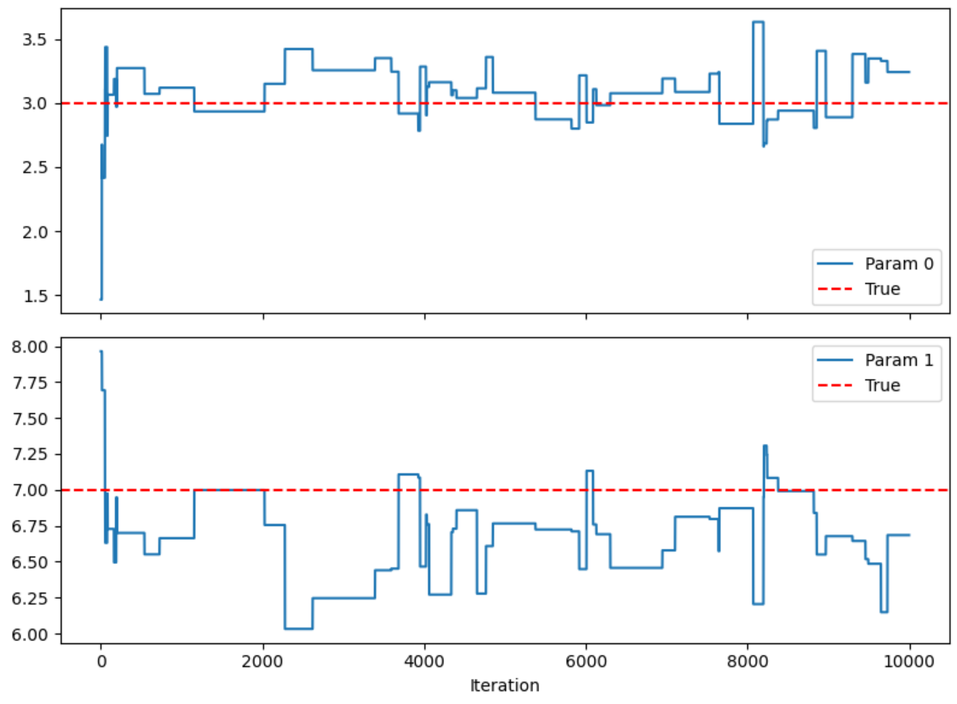


Figure 3.2: Trace plot for (u, l) parameters using KDE likelihood.

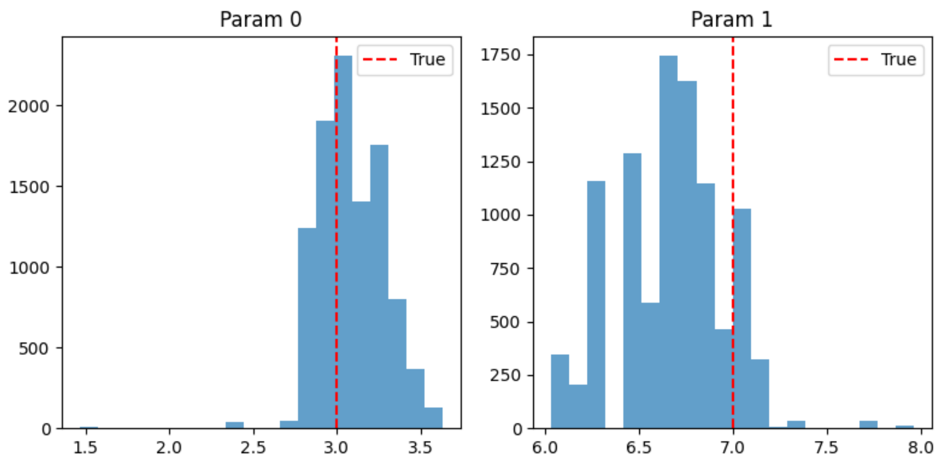


Figure 3.3: Posterior histograms using KDE-based likelihood.

KL

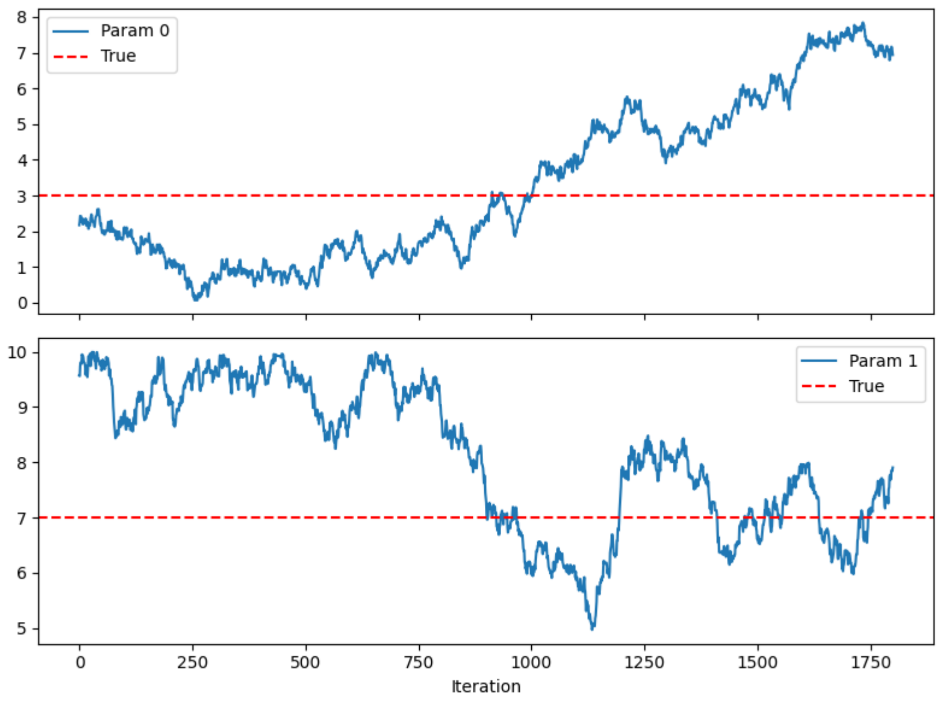


Figure 3.4: Trace plot for (u, l) parameters using KL divergence.

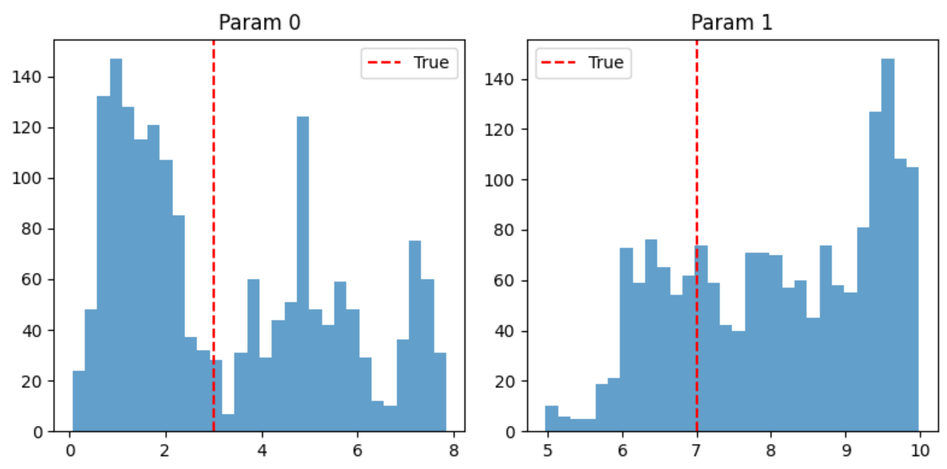


Figure 3.5: Posterior histograms using KL-divergence-based likelihood.

3.5 Second Inference Example and hyperparameters

Having validated our framework on the recovery of parameters (u, l) for a single ellipsoidal population, we now turn to a more realistic and challenging scenario: inference in the presence of a mixture of species. This situation mimics industrial or biological settings where a powder sample contains particles from multiple distinct size distributions, each governed by its own uniform law in 3D.

3.5.1 Problem Description

We assume the data originates from a mixture of two independent uniform distributions in 3D space. The first species is governed by parameters (u, l), and the second by (u_2, l_2). At each sample, an ellipsoid is drawn from the first species with probability α, and from the second with probability $1 - \alpha$. Our goal is to infer the full parameter vector:

$$\theta = (u, l, u_2, l_2, \alpha)$$

from a dataset of 2D projected diameters $(D_{\max}, D_{\min})$, generated by projecting randomly oriented ellipsoids.

3.5.2 Inference Setup

We reuse the modular sampler and likelihood classes developed earlier, adapting them to a new custom distribution `MixtureOfUniforms`. The prior is taken to be uniform over each parameter's admissible range: both u and u_2 lie in the interval $[0, 10]$, as do the corresponding lengths l and l_2, while the mixture proportion α is constrained to the interval $[0, 1]$.

For likelihood evaluation, we again adopt a KDE-based simulation likelihood, shown to be more stable than KL-divergence in previous sections.

The inference process runs the Metropolis-Hastings algorithm over the 5-dimensional parameter space. At each iteration, simulated projection data is generated from the current parameters, and compared against the observed mixture using kernel density estimation.

3.5.3 Hyperparameter Roles

We will study how the following hyperparameters affect convergence quality and global error:

- `bandwidth`: the smoothing parameter of the KDE likelihood; small values may overfit noise, while large values blur relevant structures.
- `n_sim`: the number of forward simulations per likelihood evaluation; more samples yield smoother likelihoods but increase cost.
- `proposal_scale`: the standard deviation of the Gaussian proposal; controls the exploration–acceptance tradeoff in MCMC.
- `n_iter`: the number of total MCMC iterations; longer chains improve posterior exploration but add computational burden.
- `N`: the number of observed data points; impacts likelihood sharpness and the signal-to-noise ratio.

3.5.4 Baseline Run and Trace Interpretation

In our baseline configuration, the true parameters are set to:

$$(u, l, u_2, l_2, \alpha) = (1.0, 1.5, 5.0, 3.0, 0.4)$$

We run the sampler for 3,000 iterations, using a Gaussian proposal with scale 0.1, a KDE bandwidth of 0.3, and $n_{\text{sim}} = 2000$ simulated samples per likelihood call. After discarding the initial 40% as burn-in, we compute posterior means and errors for each parameter.

Figures 3.6 and 3.7 present the trace plots for each of the five parameters (`u, l, u2, l2, alpha`) and the random-walk evolution of parameter pairs (u, l) and $(u2, l2)$ respectively.

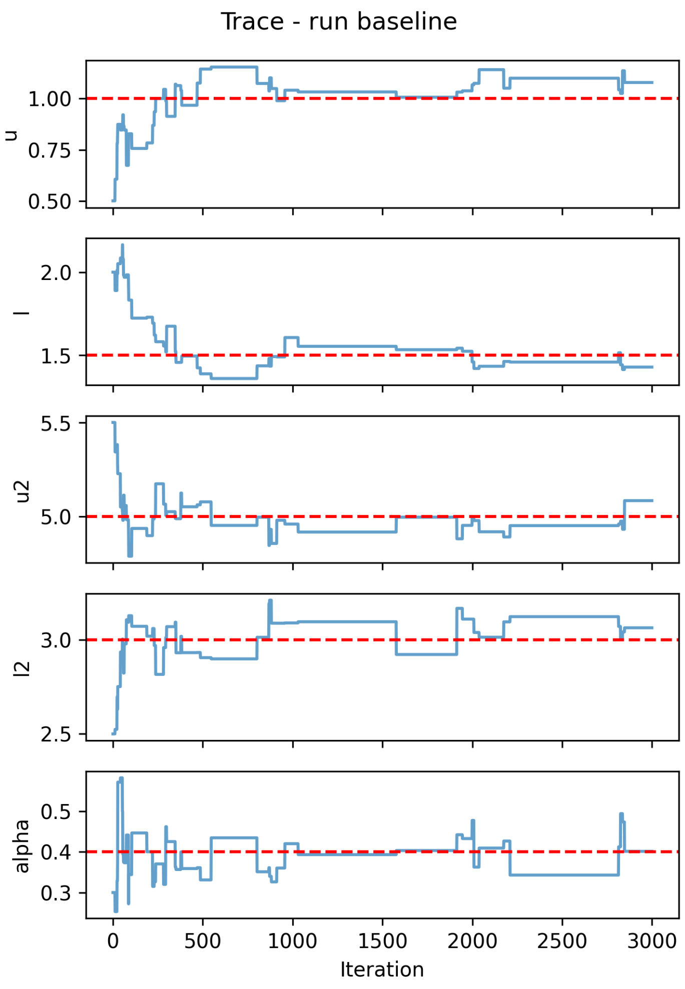


Figure 3.6: Trace plots for the `baseline` run. The red dashed line indicates the true parameter values.

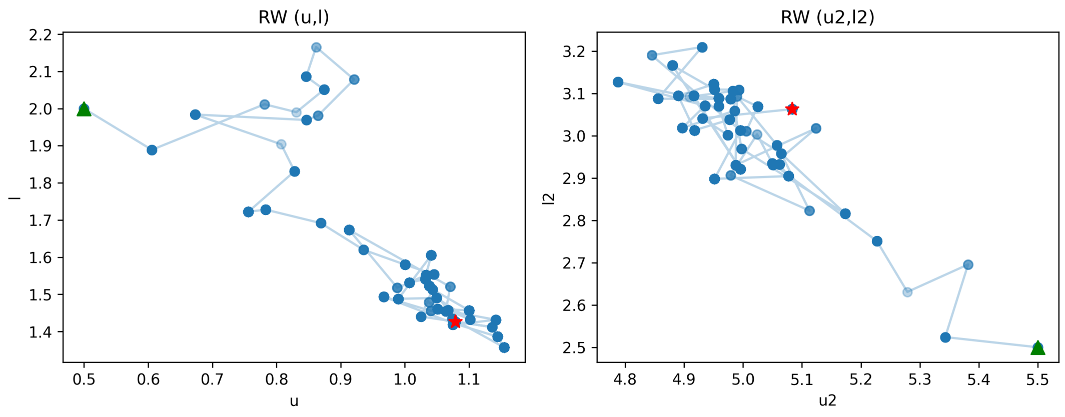


Figure 3.7: Random-walk behavior in (u, l) and $(u2, l2)$ subspaces for the `baseline` run.

The baseline run demonstrates satisfactory convergence towards the true parameters, with low estimation error. It thus serves as the reference point for our subsequent hyperparameter sensitivity analysis, where we vary one parameter at a time while keeping the others fixed.

3.5.5 Impact of Hyperparameters

We now assess the effect of individual hyperparameters by varying them independently from the baseline, and tracking the resulting `err_global` (overall error).

Effect of `n_sim`

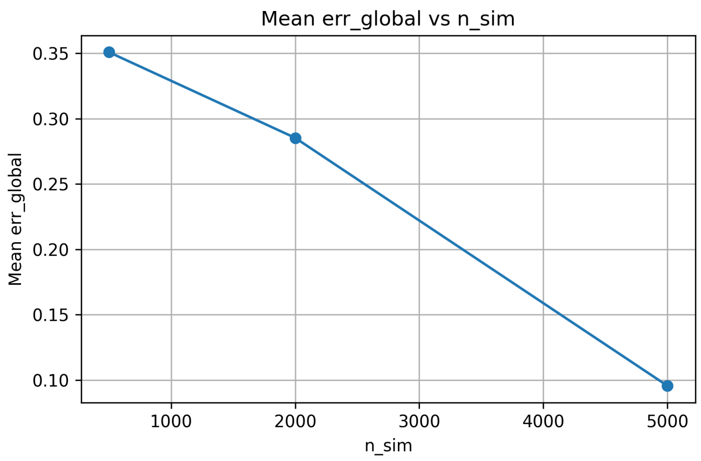


Figure 3.8: Mean global error vs. number of simulations (`n_sim`).

Increasing `n_sim` leads to a consistent decrease in error. This confirms that the KDE-based likelihood becomes more accurate when built from a larger number of simulated observations.

Effect of `n_iter`

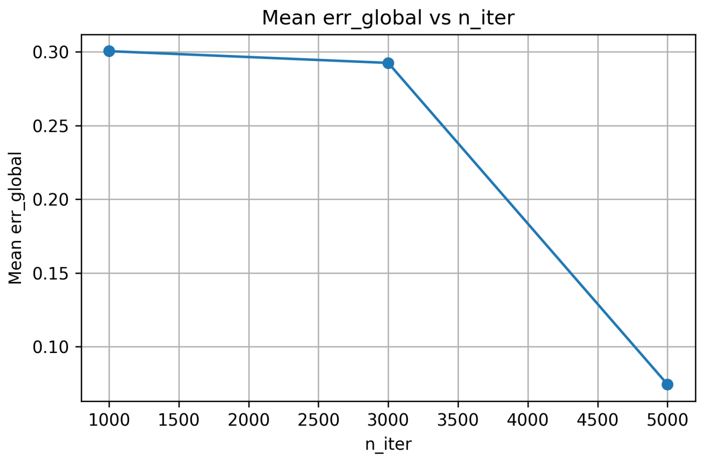


Figure 3.9: Mean global error vs. number of MCMC iterations (`n_iter`).

A higher number of MCMC iterations allows the chain to better explore the parameter space, which results in improved estimates and reduced global error.

Effect of `bandwidth`

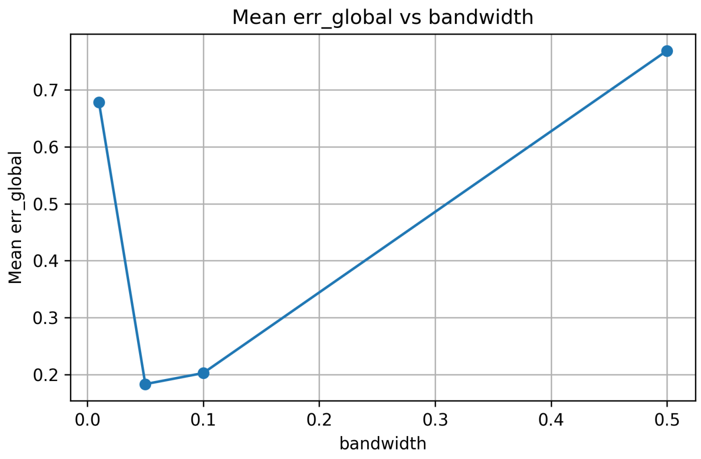


Figure 3.10: Mean global error vs. KDE bandwidth.

The bandwidth parameter is critical for kernel density estimation. Both overly small and overly large values degrade the inference quality. The error exhibits a U-shaped pattern, suggesting an optimal range around 0.05–0.1.

Effect of `N`

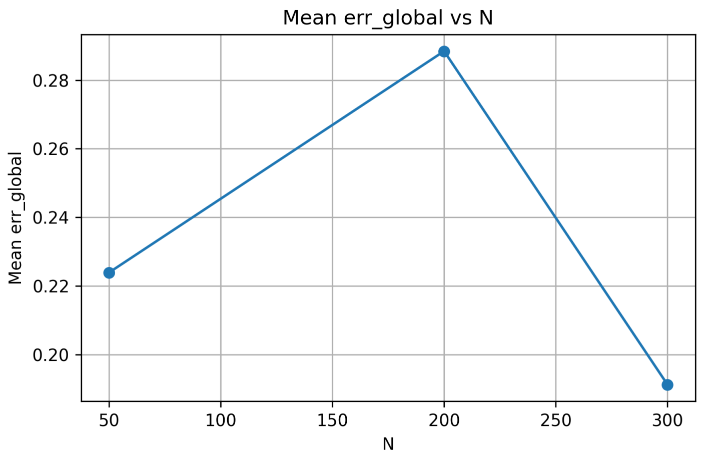


Figure 3.11: Mean global error vs. number of observed projections (`N`).

The impact of `N` is non-monotonic. While increasing N generally improves statistical robustness, too small or too large values can respectively lead to overfitting or poor estimation due to noisy likelihoods.

Effect of `proposal_scale`

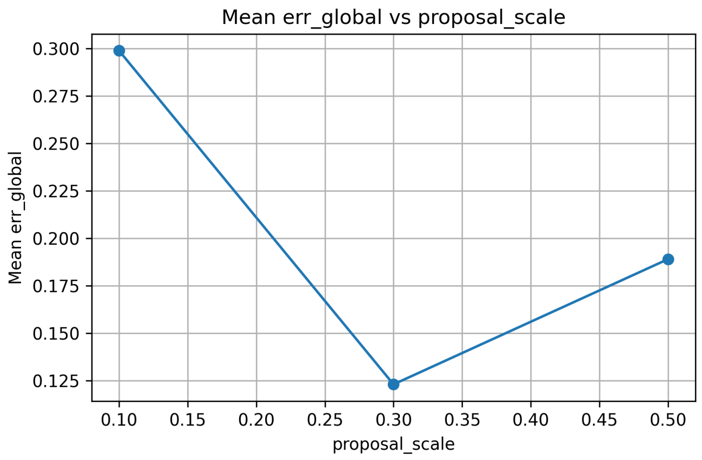


Figure 3.12: Mean global error vs. Metropolis-Hastings proposal scale.

The proposal scale determines how far proposals move in parameter space. A scale that is too small results in slow exploration, while one that is too large leads to poor acceptance. A value around 0.3 appears to yield the best balance in this experiment.

This sensitivity analysis highlights optimal ranges for each hyperparameter, providing guidance for future experiments and efficient tuning of the inference pipeline.

3.5.6 Conclusion on MCMC Inference

This section has demonstrated the capacity of MCMC methods to recover multiple parameters of a mixture distribution from 2D projection data, even in the presence of latent mixing proportions. By combining a well-calibrated KDE-based likelihood with modular prior and sampling structures, we achieved accurate and stable inference. Additionally, our hyperparameter study confirmed the importance of tuning simulation and proposal parameters to optimize convergence and accuracy.

While MCMC provides a flexible and principled Bayesian framework, it can be computationally expensive and sensitive to tuning. In the next section, we explore alternative approaches rooted in frequentist inference, where parameter estimation is performed via direct optimization of discrepancy measures.

Chapter 4

Frequentist Approach

While Bayesian inference aims to obtain a complete characterization of the uncertainty surrounding parameters through the posterior distribution, the frequentist approach typically focuses on estimating a single point value for the parameters, considered the "best" explanation for the observed data. This section details the formulation of this approach within our context and discusses its advantages and disadvantages compared to the Bayesian approach using MCMC.

4.1 General Idea and Formulation

Let us first recall the Bayesian framework. Given a model for generating 3D particles parameterized by θ (e.g., the parameters of a LogNormBetaBeta distribution for the axes (a, b, c)), and 2D observations $y = \{(d_{\max,i}, d_{\min,i})\}_{i=1}^{N}$ resulting from random projections, the Bayesian objective is to infer the posterior distribution of the parameters $p(\theta|y)$. According to Bayes' theorem, this is proportional to the product of the likelihood $p(y|\theta)$ and the prior distribution $p(\theta)$:

$$p(\theta|y) = \frac{p(y|\theta)\, p(\theta)}{p(y)} \propto p(y|\theta)\, p(\theta).$$

MCMC methods, presented earlier, are designed to sample from this posterior distribution, thus providing a complete description of the plausibility of different values of θ after observing the data y.

The **frequentist approach**, on the other hand, adopts a different perspective. Rather than seeking the entire distribution $p(\theta|y)$, it often aims to find the single value $\hat{\theta}$ that maximizes this posterior density. This estimator is known as the **Maximum A Posteriori (MAP)** estimator:

$$\hat{\theta}_{\text{MAP}} = \arg\max_{\theta} p(\theta|y) = \arg\max_{\theta} \{p(y|\theta)\, p(\theta)\}\,.$$

The term $p(y) = \int p(y|\theta')p(\theta')d\theta'$ in the denominator of Bayes' law is a normalization constant with respect to θ and therefore does not affect the location of the maximum.

In many practical situations, particularly when prior information about the parameters θ is limited or when one wishes to let the data "speak for themselves," a **non-informative or uniform prior** is used over a plausible domain, Θ_{allowed}. For example, one might assume that $p(\theta)$ is constant for all values of θ within this interval and zero outside:

$$p(\theta) \propto \mathbb{I}(\theta \in \Theta_{\text{allowed}}) = \begin{cases} 1 & \text{if } \theta \in \Theta_{\text{allowed}} \\ 0 & \text{otherwise} \end{cases}$$

Under this assumption of a uniform prior, maximizing the posterior density $p(y|\theta)p(\theta)$ becomes equivalent to maximizing the **likelihood function** $p(y|\theta)$ alone, subject to the constraint that θ belongs to the allowed domain Θ_{allowed}:

$$\hat{\theta}_{\text{MLE}} = \underset{\theta \in \Theta_{\text{allowed}}}{\arg\max}\, p(y|\theta).$$

This estimator is called the **Maximum Likelihood Estimator (MLE)**. The goal then becomes to find the parameters $\hat{\theta}$ that make the observations y most probable, according to the model. It is this MLE approach that we primarily adopt in our frequentist framework.

However, a major challenge remains, identical to the one encountered with MCMC methods: the likelihood function $p(y|\theta)$ is **analytically intractable**. The complex data generation process – 3D sampling according to $p(a, b, c|\theta)$, random uniform rotation in $SO(3)$, projection onto a 2D plane, and measurement of characteristics (d_{max}, d_{min}) – does not allow for an explicit mathematical expression for $p(y|\theta)$.

To overcome this difficulty, we use a simulation-based approach. We leverage our `granulometer` module as a "forward model". For a candidate parameter value θ, we can generate a large number M of simulated samples $y_{\text{sim}}(\theta) = \{(d'_{\text{max},j}, d'_{\text{min},j})\}_{j=1}^{M}$. These simulated data allow us to obtain an **empirical estimate of the likelihood**. A common method for this is using Kernel Density Estimation (KDE). By applying KDE to the simulated data $y_{\text{sim}}(\theta)$, we construct an approximate probability density $\widehat{p}_\theta(y)$ which represents our best estimate of the true (but unknown) distribution $p(y|\theta)$. The likelihood (or more practically, the log-likelihood) of our actual observed data $y_{\text{obs}} = \{(d_{\text{max},i}, d_{\text{min},i})\}_{i=1}^{N}$ can then be approximated by:

$$\log \widehat{\mathcal{L}}(\theta|y_{\text{obs}}) \approx \sum_{i=1}^{N} \log\big(\widehat{p}_\theta(d_{\text{max},i}, d_{\text{min},i})\big).$$

The frequentist inference problem thus transforms into a **numerical optimization** problem: finding the value $\hat{\theta}$ that maximizes this estimated log-likelihood (or, equivalently, minimizes its negative, the Negative Log-Likelihood, NLL):

$$\hat{\theta} = \underset{\theta \in \Theta_{\text{allowed}}}{\arg\max} \left[\sum_{i=1}^{N} \log\big(\widehat{p}_\theta(d_{\max,i}, d_{\min,i})\big) \right].$$

A conceptual advantage of this optimization-based approach is that it **explicitly** aims for convergence towards a single optimal value $\hat{\theta}$ (the estimated maximum likelihood). Optimization algorithms are designed to actively search for this optimal point in the parameter space. In comparison, MCMC methods explore the entire posterior distribution $p(\theta|y)$. Although the most densely sampled region by a converged MCMC chain generally corresponds to the mode of the distribution (which often coincides with $\hat{\theta}_{\text{MAP}}$ or $\hat{\theta}_{\text{MLE}}$ under a uniform prior), the primary goal of MCMC is not to precisely locate this maximum, but rather to provide a representative sample of the entire distribution, thereby capturing the uncertainty associated with θ. The frequentist approach via optimization thus offers a more direct path to a point estimate, potentially at the cost of losing information about the overall parameter uncertainty.

4.2 Likelihood Approximation via Kernel Density Estimation (KDE)

As established in the previous section, the core of the frequentist approach involves maximizing the likelihood function $p(y|\theta)$, where $y = \{y_1, ..., y_N\}$ represents the set of N observed 2D data points $y_i = (d_{\max,i}, d_{\min,i})$. However, the analytical form of $p(y|\theta)$, the probability density of observing a 2D projection y given the 3D parameters θ, is intractable due to the complex simulation process (sampling 3D shape, random rotation, projection).

To overcome this, we approximate the true likelihood density $p(y|\theta)$ using simulation. For a given set of parameters θ, we use the `granulometer` simulator to generate a large number M of independent simulated 2D data points, $y'_{\text{sim}}(\theta) = \{y'_1, ..., y'_M\}$, where each $y'_j = (d'_{\max,j}, d'_{\min,j})$. We then use these simulated points to construct a non-parametric estimate of the underlying density using **Kernel Density Estimation (KDE)**.

The KDE $\widehat{p}_\theta(y)$ at a point y in the 2D space ($d = 2$) is defined as:

$$\widehat{p}_\theta(y) = \frac{1}{Mh^d} \sum_{j=1}^{M} K\left(\frac{y - y'_j}{h}\right)$$

where:

- M is the number of simulated data points used for the estimation for a given θ.

- y'_j are the individual simulated 2D data points from $y'_{\text{sim}}(\theta)$.
- y is the 2D point (e.g., an observed data point y_i) at which we want to estimate the density.
- h is the **bandwidth**, a smoothing parameter that controls the width of the kernels.
- $d = 2$ is the dimensionality of the data points y and y'_j.
- $K(\cdot)$ is the **kernel function**, typically a symmetric probability density function itself (e.g., Gaussian kernel), which assigns weights to simulated points based on their distance to y.

Intuitively, the KDE places a smoothed "bump" (the kernel K, scaled by h) centered at each simulated data point y'_j and then averages these bumps across all M points to form the density estimate $\widehat{p}_\theta(y)$.

For this KDE $\widehat{p}_\theta(y)$ to be a reliable approximation that converges to the true underlying density $p(y|\theta)$ as the number of simulations M increases, certain conditions must generally be met [Silverman, 1986]:

Bandwidth (h): The bandwidth h (which often depends on M, denoted h_M) must satisfy two crucial conditions as $M \to \infty$:

1. $h \to 0$: The bandwidth must shrink to zero. *Intuition:* As more simulation data becomes available ($M \to \infty$), less smoothing is needed. Shrinking the bandwidth allows the estimate to capture finer details of the true density $p(y|\theta)$ and reduces bias. If h remained constant, the KDE would be overly smooth and biased.
2. $Mh^d \to \infty$ (for dimension $d = 2$, $Mh^2 \to \infty$): The product of the sample size and the d-th power of the bandwidth must tend to infinity. *Intuition:* The bandwidth cannot shrink too quickly relative to the sample size. Mh^d is related to the effective number of simulated points contributing to the estimate at any given point y. This condition ensures that even as the kernel window shrinks ($h \to 0$), enough simulated points fall within the relevant neighborhood to stabilize the estimate and reduce its variance. If $Mh^d \to 0$, the estimate would become excessively noisy.

Kernel Function (K): The kernel function $K(u)$ must typically satisfy:

- Normalization: $\int K(u)du = 1$. (Ensures $\widehat{p}_\theta(y)$ integrates to 1, i.e., is a valid density).
- Non-negativity: $K(u) \geq 0$. (Ensures the density estimate is non-negative).

- Symmetry (usually): Often assumed symmetric around 0 ($K(u) = K(-u)$), implying $\int uK(u)du = 0$. (Simplifies bias analysis).
- Finite Variance (often): $\int u^2K(u)du < \infty$. (Typically needed for analyzing convergence rates, particularly the bias term). Boundedness may also be required for some results.

True Density Function ($p(y|\theta)$): The underlying true density $p(y|\theta)$ that we are trying to estimate needs to be sufficiently smooth. Typically, it is required to be at least continuous, and for results concerning convergence rates (bias analysis), it often needs to be differentiable (e.g., twice differentiable with bounded derivatives) at the points of interest.

Under these conditions, the KDE $\widehat{p}_\theta(y)$ converges to the true density $p(y|\theta)$ in various senses. Standard results in non-parametric statistics establish several types of convergence:

- **Pointwise Consistency:** $\widehat{p}_\theta(y) \to p(y|\theta)$ (in probability or almost surely) for specific points y where $p(y|\theta)$ is continuous.
- **Mean Squared Error (MSE) Convergence:** $E[(\widehat{p}_\theta(y) - p(y|\theta))^2] \to 0$. The MSE decomposes into squared bias and variance. The bandwidth conditions $h \to 0$ and $Mh^d \to \infty$ are precisely what's needed to ensure both bias and variance tend to zero.
- **Integrated Mean Squared Error (MISE) Convergence:** $\int E[(\widehat{p}_\theta(y)-p(y|\theta))^2]dy \to 0$. This is a global measure of convergence, indicating that the average squared error across the entire domain approaches zero. This is a key result demonstrating the overall fidelity of the KDE [Silverman, 1986].
- **Uniform Consistency:** $\sup_y |\widehat{p}_\theta(y) - p(y|\theta)| \to 0$ (in probability or almost surely). This stronger form guarantees that the maximum deviation between the estimate and the true density vanishes. It often requires slightly stronger conditions, potentially on the rate at which $Mh^d \to \infty$ (e.g., $Mh^d/\log(M) \to \infty$) and uniform continuity/boundedness of $p(y|\theta)$ and its derivatives.

Therefore, by using a sufficiently large number of simulations M and choosing the bandwidth h appropriately (often done automatically by KDE implementations, e.g., using rules like Scott's rule or Silverman's rule-of-thumb, or via cross-validation), the KDE $\widehat{p}_\theta(y)$ provides a statistically sound approximation to the true likelihood density $p(y|\theta)$. Consequently, maximizing the log-likelihood based on this KDE approximation,

$$\log \widehat{\mathcal{L}}(\theta|y_{\text{obs}}) = \sum_{i=1}^{N} \log\big(\widehat{p}_\theta(y_i)\big)$$

becomes a valid and practical approach to approximate the true Maximum Likelihood Estimator $\hat{\theta}_{\text{MLE}}$. The optimization algorithms discussed next will operate on this approximated (negative) log-likelihood function.

4.3 A Note on Model Identifiability

Before proceeding to the optimization algorithms and results, it is crucial to address the concept of **model identifiability**. Our inferential goal relies on the premise that the parameters θ governing the 3D ellipsoid distribution $p_{3D}(a, b, c|\theta)$ can be uniquely determined from the distribution of the observed 2D projections $y = (d_{\max}, d_{\min})$.

Specifically, the simulation process (sampling from $p_{3D}(a, b, c|\theta)$, applying a random rotation $R \sim \text{Unif}(SO(3))$, and projecting) induces a probability distribution for the 2D observations, which we have denoted as $p_{2D}(y|\theta)$. This is the likelihood function central to both Bayesian and frequentist inference.

The question of identifiability is whether this mapping from the parameter space Θ to the space of possible 2D distributions is *injective*. That is, if we have two different sets of parameters $\theta_1 \in \Theta$ and $\theta_2 \in \Theta$ such that $\theta_1 \neq \theta_2$, does it necessarily follow that the resulting 2D distributions are also different, i.e., $p_{2D}(y|\theta_1) \neq p_{2D}(y|\theta_2)$?

If the model lacks identifiability, it means that there exist distinct parameter sets θ_1 and θ_2 which generate the exact same statistical distribution of 2D projections. This would imply that, even with an infinite amount of 2D observation data y, we could not fundamentally distinguish between θ_1 and θ_2. The likelihood function $p_{2D}(y|\theta)$ would be identical for both parameter sets, leading to ambiguity in any point estimate $\hat{\theta}$ or posterior distribution $p(\theta|y)$. The non-bijective nature of the 3D $\rightarrow$ 2D projection for individual shapes, as noted earlier, raises concerns about potential non-identifiability at the distributional level as well.

Establishing identifiability (or lack thereof) formally for this kind of complex simulation-based model is often extremely challenging. It typically requires rigorous mathematical analysis that depends heavily on the specific chosen parametric family for the 3D distribution (e.g., Uniform, LogNormal, LogNormBetaBeta, TriLogNormal) and the nature of the observed characteristics $(d_{\max}, d_{\min})$. A general result covering all possible 3D distributions is unlikely. Ideally, identifiability should be studied on a case-by-case basis for each specific parametric model considered.

Given the complexity of such an analysis, for the remainder of this work, we will proceed under the **working assumption** that the parametric models we investigate are identifiable, at least within the parameter ranges of practical interest. This means we assume that distinct parameter values θ do lead to distinct observable distributions $p_{2D}(y|\theta)$, allowing us, in principle, to infer a unique θ from sufficient data y. Acknowledging this assumption is important when interpreting the inference results. Further investigation into the identifiability of specific models could be a subject for future research.

4.4 Optimization Strategy: Differential Evolution

The frequentist inference goal is to find the parameter vector $\hat{\theta}$ that minimizes the approximated negative log-likelihood (NLL) or distance function $D(\theta)$, derived from the KDE likelihood:

$$\hat{\theta} = \underset{\theta \in \Theta_{\text{allowed}}}{\arg\min}\, D(\theta) = \underset{\theta \in \Theta_{\text{allowed}}}{\arg\min} \left[-\sum_{i=1}^{N} \log\big(\widehat{p}_\theta(y_i)\big) \right].$$

However, due to the complexities of the underlying simulation process and KDE approximation, the objective function landscape $D(\theta)$ can be rugged and potentially contain multiple local minima (see Figure 4.1), where standard local optimization methods might fail.

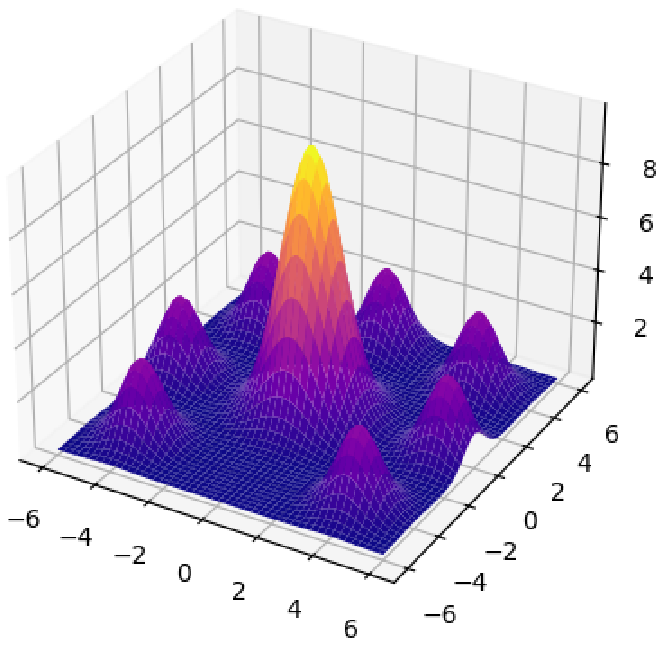


Figure 4.1: Schematic illustration of a complex objective function landscape (e.g., negative log-likelihood) with multiple local minima and a single global minimum, necessitating a global optimization approach.

Therefore, a robust **global optimization algorithm** is essential. We chose **Differential Evolution (DE)**[Storn and Price, 1997], a population-based stochastic search algorithm. DE is well-suited for potentially multi-modal, non-differentiable, and noisy continuous optimization problems like ours. Its effectiveness and the availability of a reliable implementation in ‘scipy.optimize.differential_evolution‘ motivated its selection.

Differential Evolution Algorithm Outline

DE operates by evolving a population $\mathcal{P}^{(g)} = \{\theta_1^{(g)}, ..., \theta_{N_p}^{(g)}\}$ of N_p candidate solutions over generations g. Key steps in each generation include:

1. **Mutation:** For each individual θ_i, a mutant vector v_i is created by adding a scaled difference between two other randomly chosen population members to a third random member. A common strategy is:

   $$v_i^{(g)} = \theta_{r1}^{(g-1)} + F \cdot (\theta_{r2}^{(g-1)} - \theta_{r3}^{(g-1)})$$

   where $r1, r2, r3$ are distinct random indices $\neq i$, and F is a scaling factor. This uses the population's current diversity to explore new areas.

2. **Crossover:** A trial vector u_i is formed by mixing components from the original vector θ_i and the mutant vector v_i, controlled by a crossover probability CR.

3. **Selection:** The trial vector u_i replaces the original vector θ_i in the next generation's population ($\mathcal{P}^{(g)}$) only if it yields a better (lower) objective function value $D(u_i) \leq D(\theta_i)$. Otherwise, θ_i persists.

This process repeats until a termination criterion is met (e.g., maximum generations, convergence). The best individual found throughout the process is taken as the final estimate $\hat{\theta}$. DE's mechanism allows it to effectively navigate complex landscapes and identify promising regions corresponding to low NLL values.

4.5 Numerical Experiments with Synthetic Data

To validate the frequentist inference approach using KDE-approximated likelihood and Differential Evolution optimization, we conducted experiments on synthetic data. We generated "observed" 2D projection data using known 3D ellipsoid distribution parameters (θ^*) and then attempted to recover these parameters using our method. This allows us to assess the accuracy and behavior of the estimation procedure under different modeling assumptions.

4.5.1 Experiment 1: LogNormBetaBeta Distribution

Model Description and Motivation

The first model considered is the **LogNormBetaBeta** distribution. In this model, the semi-axes (a, b, c) of the ellipsoid are generated sequentially, ensuring $a \geq b \geq c$ by construction:

1. The largest semi-axis a follows a LogNormal distribution: $\log(a) \sim \mathcal{N}(\mu, \sigma^2)$.
2. The ratio $r_1 = b/a$ follows a Beta distribution: $r_1 \sim \text{Beta}(\alpha_1, \beta_1)$, with $r_1 \in [0, 1]$. Then $b = a \cdot r_1$.
3. The ratio $r_2 = c/b$ follows another Beta distribution: $r_2 \sim \text{Beta}(\alpha_2, \beta_2)$, with $r_2 \in [0, 1]$. Then $c = b \cdot r_2$.

The parameter vector for this model is $\theta = (\mu, \sigma, \alpha_1, \beta_1, \alpha_2, \beta_2)$.

This model structure, inspired by similar approaches for modeling dependent ellipsoid axes [Wu et al., 2018], naturally incorporates a dependency between the dimensions. A larger value of a tends to lead to potentially larger values for b and c, which aligns with the intuition that particles elongated in one dimension might also tend to be larger in others, rather than being completely independent. This structure might also lead to a simpler optimization landscape compared to models involving sorting of independent variables.

Experimental Setup and Results

We performed a simulation study detailed in the accompanying Jupyter notebook `frequentist_testing.ipynb`.

- **Ground Truth Parameters (θ^*):** We generated synthetic 2D projection data using the following parameters:

$$\theta^* = (\mu = 0.5, \sigma = 0.3, \alpha_1 = 2.0, \beta_1 = 5.0, \alpha_2 = 2.0, \beta_2 = 5.0)$$

- **Optimization:** We applied the Differential Evolution algorithm to minimize the negative log-likelihood $D(\theta)$ based on the KDE approximation. An initial guess for the optimization was set to:

$$\theta_{\text{init}} = (0.0, 0.2, 5.0, 5.0, 5.0, 9.0)$$

  (Note: DE is less sensitive to the initial guess than local optimizers, but it defines the initial population spread).

- **Estimated Parameters ($\hat{\theta}$):** The optimization process converged to the following parameter estimates:

$$\hat{\theta} \approx (0.5201, 0.3119, 1.9059, 4.7139, 4.6810, 8.6652)$$

Interpretation

Comparing $\hat{\theta}$ with θ^*, we observe that the estimated parameters are remarkably close to the true values used to generate the data. This provides strong evidence that the frequentist approach, combining KDE likelihood approximation with Differential Evolution, can successfully recover the parameters of the LogNormBetaBeta distribution from 2D projection data. The good performance might be partly attributed to the model structure potentially leading to a well-behaved objective function with a dominant global minimum.

4.5.2 Experiment 2: TriLogNormal Distribution

Model Description and Identifiability Challenge

The second model involves sampling three independent values (a', b', c') from potentially different LogNormal distributions and then sorting them to obtain the final ellipsoid semi-axes (a, b, c) such that $a \geq b \geq c$.

$$\log(a') \sim \mathcal{N}(\mu_1, \sigma_1^2), \quad \log(b') \sim \mathcal{N}(\mu_2, \sigma_2^2), \quad \log(c') \sim \mathcal{N}(\mu_3, \sigma_3^2)$$

The parameter vector is $\theta = (\mu_1, \sigma_1, \mu_2, \sigma_2, \mu_3, \sigma_3)$.

A key characteristic of this model is the **lack of identifiability** arising from the sorting step. If we permute the pairs (μ_i, σ_i), the distribution of the initial (a', b', c') changes, but the distribution of the final sorted axes (a, b, c) remains exactly the same. For example, the parameter set $(\mu_1, \sigma_1, \mu_2, \sigma_2, \mu_3, \sigma_3)$ produces the same distribution of (a, b, c) as $(\mu_2, \sigma_2, \mu_1, \sigma_1, \mu_3, \sigma_3)$.

Consequently, the likelihood function $p(y|\theta)$ (and our approximation $D(\theta)$) will have multiple $(3! = 6)$ equivalent global optima in the parameter space, corresponding to the different permutations of the underlying (μ_i, σ_i) pairs. This multi-modal structure is schematically illustrated in Figure 4.2.

However, this lack of identifiability is not catastrophic for inference in the sense that all these global optima correspond to the same underlying physical model (the same set of three LogNormal distributions). Finding any one of these optimal parameter vectors using our optimization algorithm is sufficient, as it correctly identifies the set of underlying marginal distributions, albeit possibly in a different order than specified in the ground

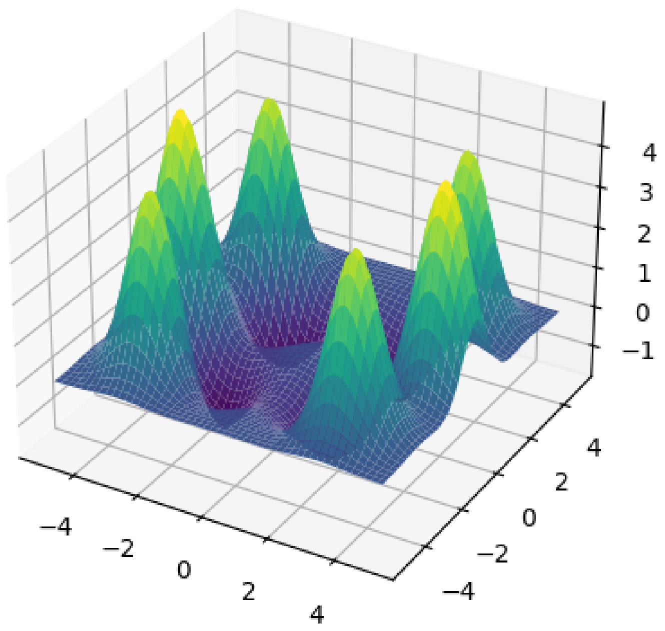


Figure 4.2: Schematic illustration of a multi-modal objective function landscape expected for the TriLogNormal model due to permutations yielding equivalent distributions for the sorted axes. Finding any one of the global minima is sufficient.

truth θ^*.

Experimental Setup and Results

We conducted a similar experiment using the TriLogNormal model:

- **Ground Truth Parameters (θ^*):** We generated synthetic data using:

  $$\theta^* = (\mu_1 = 1.0, \sigma_1 = 0.5, \mu_2 = 2.0, \sigma_2 = 0.8, \mu_3 = 0.0, \sigma_3 = 0.9)$$

  The underlying LogNormal distributions have parameters $(1.0, 0.5)$, $(2.0, 0.8)$, and $(0.0, 0.9)$.

- **Optimization:** Differential Evolution was used again, with the same number of simulated data points and maximum iterations as in the previous experiment.

- **Estimated Parameters ($\hat{\theta}$):** The optimization yielded the following result:

  $$\hat{\theta} \approx (0.7275, 0.8560, 0.6774, 0.5181, 1.9336, 0.7939)$$

Interpretation

To interpret $\hat{\theta}$, we must consider the permutations. The estimated parameter vector contains the pairs $(0.7275, 0.8560)$, $(0.6774, 0.5181)$, and $(1.9336, 0.7939)$. Comparing these to the true underlying pairs $(1.0, 0.5)$, $(2.0, 0.8)$, and $(0.0, 0.9)$:

- The estimated pair $(1.9336, 0.7939)$ is quite close to the true pair $(2.0, 0.8)$, which corresponds to the largest mean component in the ground truth.
- The other two estimated pairs do not closely match the remaining true pairs $(1.0, 0.5)$ and $(0.0, 0.9)$.

This suggests that, with the current computational budget, the optimization successfully located the region of one of the global optima and recovered the parameters associated with the most dominant component (largest mean) reasonably well, up to permutation. However, the parameters for the other two components were estimated less accurately.

This run took approximately two hours. It is plausible that increasing the computational effort – either by using a larger initial synthetic dataset (N) or by generating more simulations (M) for the KDE estimation at each step of the DE algorithm – could lead to more accurate estimates for all parameter components. Nonetheless, the fact that the method converged to a region near a global optimum, even in this more challenging multi-modal scenario, demonstrates the capability of the approach. The relative ease of parameter recovery in the LogNormBetaBeta case likely stems from its potentially simpler, uni-modal optimization landscape compared to the TriLogNormal case.

4.6 Model Selection via Minimum Distance

In practical applications involving real-world data, the true underlying probability distribution governing the 3D particle shapes is unknown. We might observe the empirical distribution of the 2D projections $(d_{\text{max}}, d_{\text{min}})$ and hypothesize several plausible parametric families for the 3D axes (a, b, c) – for example, Uniform, LogNormal-based, or Gamma-based models. A crucial question then arises: how can we determine which of these candidate distribution families provides the best representation of the observed data?

The frequentist inference framework developed in the previous sections offers a natural approach to model selection. The core idea is to leverage the minimized distance (approximated negative log-likelihood) achieved for each candidate model. The procedure is as follows:

1. For each candidate parametric distribution family $\mathcal{F}_k$ (e.g., $\mathcal{F}_{\text{uniform}}$, $\mathcal{F}_{\text{lognormal}}$, $\mathcal{F}_{\text{gamma}}$), use the optimization procedure (e.g., Differential Evolution) to find the

parameter vector $\hat{\theta}_k$ that minimizes the approximated negative log-likelihood $D_k(\theta)$:

$$\hat{\theta}_k = \arg\min_{\theta \in \Theta_k} D_k(\theta) = \arg\min_{\theta \in \Theta_k} \left[-\sum_{i=1}^{N} \log\big(\widehat{p}_\theta(y_i|\mathcal{F}_k)\big) \right]$$

where Θ_k is the parameter space for family $\mathcal{F}_k$, and $\widehat{p}_\theta(y|\mathcal{F}_k)$ is the KDE density estimated from simulations using model $\mathcal{F}_k$ with parameters θ.

2. Record the minimum distance value achieved for each family: $D_{\min,k} = D_k(\hat{\theta}_k)$.

3. Select the model family $\mathcal{F}_k$ that yields the **smallest** minimum distance $D_{\min,k}$.

The rationale behind this approach is that $D_{\min,k}$ represents the negative of the maximum achievable log-likelihood for the observed data $y_{\text{obs}} = \{y_1, ..., y_N\}$ under the constraints imposed by the model family $\mathcal{F}_k$. The model family achieving the highest maximum log-likelihood (i.e., the lowest $D_{\min,k}$) is considered the one that best explains the observed data, among the candidates tested.

It is important to note that the absolute value of the distance $D_{\min,k}$ depends on several factors, including the true underlying data generating process $p_{\text{true}}(y)$, the chosen model family $\mathcal{F}_k$, and the number of observed data points N. Conceptually, minimizing the NLL is related to minimizing the Kullback-Leibler (KL) divergence between the true distribution $p_{\text{true}}(y)$ and the model distribution $p(y|\theta, \mathcal{F}_k)$. The minimum achievable NLL for a correctly specified model (where p_{true} belongs to $\mathcal{F}_k$) is expected to be lower than for a misspecified model, reflecting a smaller KL divergence (ideally zero for the true model). The dependence on N is roughly linear; comparing $D_{\min,k}$ values is meaningful when they are calculated using the same observed dataset y_{obs}.

4.6.1 Experimental Validation

We performed experiments, detailed in the notebook `frequentist_testing2.ipynb`, to verify this model selection capability using synthetic data where the true generating distribution is known. In these experiments, the "LogNormal" and "Gamma" candidate models refer to the case where the three axes (a', b', c') are sampled independently from the same LogNormal or Gamma distribution, respectively, before being sorted to ensure $a \geq b \geq c$.

Experiment 1: Data Generated from Uniform Distribution

Synthetic 2D data was generated using a true Uniform distribution for the 3D axes. We then attempted to fit this data using three candidate models: Uniform, LogNormal (identical parameters), and Gamma (identical parameters). The results were:

Table 4.1: Model fitting results when true generation is Uniform.

Fitted Model	Estimated Parameters $\hat{\theta}$	Min. Distance $D_{\min}$
Uniform	$[0.9773, 3.4403]$	**3537.93**
LogNormal	$[0.9416, 0.3954]$	4037.59
Gamma	$[4.7387, 0.5638]$	3995.95

As expected, the Uniform model achieved the lowest minimum distance (Table 4.1), correctly identifying the true generating distribution family as the best fit among the candidates.

Experiment 2: Data Generated from LogNormal Distribution

Synthetic 2D data was generated using a true LogNormal distribution (with identical parameters for the three axes before sorting). We again fitted the Uniform, LogNormal, and Gamma models:

Table 4.2: Model fitting results when true generation is LogNormal.

Fitted Model	Estimated Parameters $\hat{\theta}$	Min. Distance $D_{\min}$
Uniform	$[0.0704, 8.8976]$	8472.32
LogNormal	$[1.0545, 0.5224]$	**5347.73**
Gamma	$[3.0462, 1.1052]$	5595.93

In this case (Table 4.2), the LogNormal model yielded the significantly lowest minimum distance, again correctly identifying the generating family. The estimated parameters for the LogNormal fit ($\hat{\mu} = 1.0545, \hat{\sigma} = 0.5224$) are also close to the likely ground truth values used for generation (which were likely $\mu = 1.0, \sigma = 0.5$ based on typical experimental setups).

4.7 Application to Real Experimental Data

Having validated the frequentist inference and model selection methodology on synthetic data, we now apply it to real experimental data obtained from the granulometer measurements provided by Mr. Colin. The dataset used consists of a random sample of 5000 pairs of $(d_{\min}, d_{\max})$ measurements.

4.7.1 Data Description and Initial Analysis

A first step involves visualizing the empirical distribution of the observed 2D characteristics. Figure 4.3 shows the estimated marginal probability densities for $d_{\min}$ and $d_{\max}$

based on the 5000 data points.

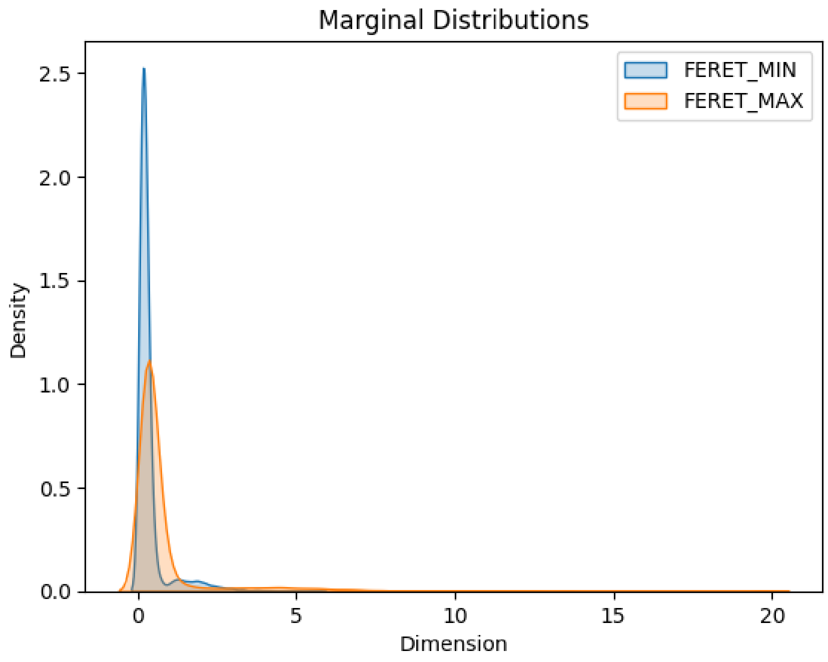


Figure 4.3: Estimated marginal probability densities for the real observed d_{min} and d_{max} data (N=5000).

Visually, both densities exhibit a peak relatively close to zero, followed by long, heavy tails ("fat tails"), particularly noticeable for d_{max}. This suggests that distributions like the LogNormal or Gamma, which can model such skewness and heavy tails, might be more appropriate than a simple Uniform distribution.

Basic descriptive statistics for the real data sample are:

- Minimum Feret (d_{min}): Mean ≈ 0.368, Minimum ≈ 0.096.
- Maximum Feret (d_{max}): Mean ≈ 0.8216, Maximum ≈ 19.72.

These statistics, especially the observed range (minimum and maximum values), are valuable for setting reasonable bounds for the parameter search space Θ_{allowed} during the optimization process. For instance, the parameters (μ, σ) of a LogNormal distribution should be constrained such that the resulting distribution covers the observed range plausibly.

4.7.2 Model Fitting and Selection

We applied the model selection procedure described in Section 4.6 to this real dataset. We fitted the same three simplified candidate models tested previously on synthetic data:

1. **Uniform:** Ellipsoid axes (a, b, c) are sampled independently from $U[u, u + l]$ and sorted. Parameters $\theta = (u, l)$.

2. **LogNormal-Identical:** Axes (a', b', c') are sampled independently from the same LogNormal(μ, σ) and sorted. Parameters $\theta = (\mu, \sigma)$.

3. **Gamma-Identical:** Axes (a', b', c') are sampled independently from the same Gamma(k, θ') (shape k, scale θ') and sorted. Parameters $\theta = (k, \theta')$.

The optimization was performed using Differential Evolution to minimize the KDE-approximated negative log-likelihood (D_{min}), as detailed in the notebook `frequentist_testing_realDa` The results are summarized in Table 4.3.

Table 4.3: Model fitting results for the real data sample (N=5000).

Fitted Model	Estimated Parameters $\hat{\theta}$	Min. Distance D_{min}
Uniform	$[0.0226, 7.2491]$	9899.6
LogNormal	$[-0.9715, 1.3051]$	**1321.55**
Gamma	$[0.475, 1.915]$	1796.88

Consistent with the visual impression from the data densities (Figure 4.3) and the model selection experiments on synthetic data, the **LogNormal-Identical model achieved a significantly lower minimum distance ($D_{\text{min}} \approx 1321.55$)** compared to the Uniform and Gamma models. This indicates that, among these three simple candidate families, the LogNormal distribution provides the best fit to the observed 2D projection data. The best-fit parameters found were $\hat{\mu} \approx -0.9715$ and $\hat{\sigma} \approx 1.3051$.

4.7.3 Visual Comparison of Best Fit Model

To assess the quality of the fit provided by the best-performing model (LogNormal-Identical with $\hat{\theta} = [-0.9715, 1.3051]$), we generated a large synthetic dataset of $(d_{\text{min}}, d_{\text{max}})$ pairs using this estimated model and parameters. We then compared the marginal densities of this synthetic data to the densities of the original real data.

Figures 4.4 and 4.5 show these comparisons for d_{max} and d_{min}, respectively.

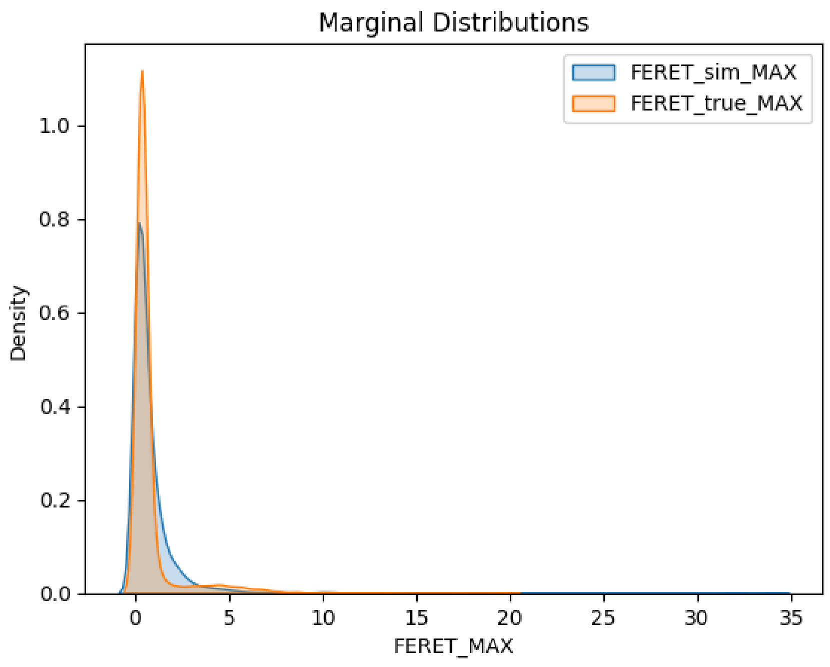


Figure 4.4: Comparison of marginal probability densities for $d_{\max}$: Real data vs. Synthetic data generated from the best-fit LogNormal model.

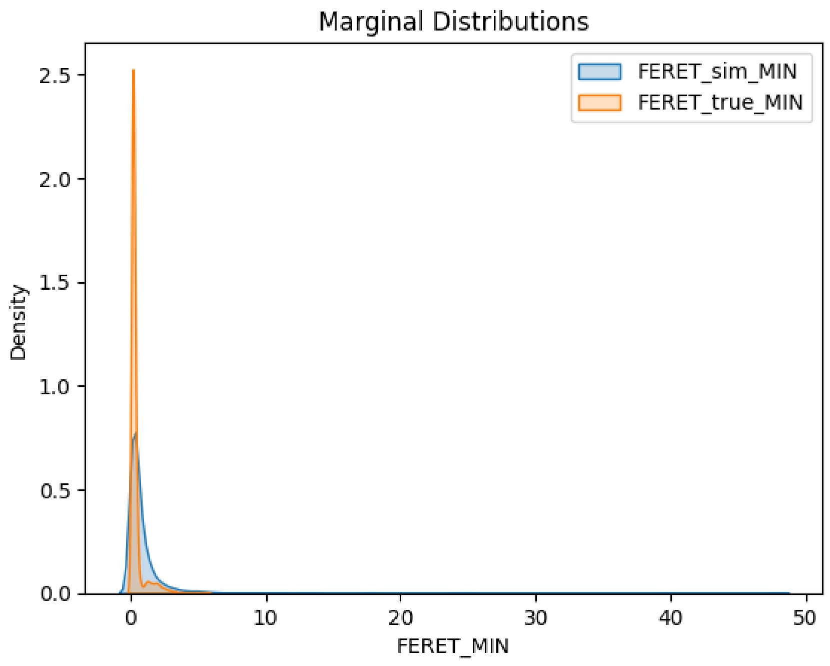


Figure 4.5: Comparison of marginal probability densities for $d_{\min}$: Real data vs. Synthetic data generated from the best-fit LogNormal model.

From Figure 4.4, we observe a reasonably good match between the real and synthetic densities for the maximum Feret diameter ($d_{\max}$). The general shape, peak location, and tail behavior are captured quite well by the fitted LogNormal model.

However, the comparison for the minimum Feret diameter ($d_{\min}$) in Figure 4.5 reveals a noticeable discrepancy. While the peak location is similar, the density generated from the synthetic LogNormal data appears significantly wider and possesses a heavier tail than the density estimated from the real $d_{\min}$ data. The real data seems more concentrated around its peak.

This mismatch suggests that while the simple LogNormal-Identical model captures some key features of the data (particularly $d_{\max}$), it may not fully represent the complexities of the real particle size and shape distribution, especially concerning the minimum dimension or the relationship between $d_{\min}$ and $d_{\max}$. More sophisticated 3D models, perhaps incorporating correlations between axes (like LogNormBetaBeta or multivariate distributions) or different marginal distributions, might be required to achieve a better simultaneous fit to both $d_{\min}$ and $d_{\max}$ distributions. Nonetheless, the LogNormal model provides a significantly better baseline fit than the Uniform or Gamma models tested here.

Chapter 5

Conclusion and Perspectives

In this report, we tackled the challenging inverse problem of characterizing the 3D shape distribution of particles using only 2D projection data acquired through granulometry. Our work was motivated by the need to better understand granular materials, such as those encountered in biomass processing. To address this, we developed a simulation-based inference framework, built upon a simplified yet expressive model where particles are represented as 3D ellipsoids whose random orientations are accounted for by uniform rotations prior to 2D projection.

We investigated two primary statistical paradigms for inferring the parameters of the 3D shape distribution: Bayesian inference and a frequentist approach based on optimization. Within the Bayesian framework, we focused on Markov Chain Monte Carlo (MCMC) methods, particularly the Metropolis-Hastings algorithm, employing simulation-based approximations of the intractable likelihood using Kernel Density Estimation (KDE) and Kullback-Leibler divergence. We also explored Approximate Bayesian Computation (ABC) rejection sampling as a likelihood-free alternative, detailing its theoretical basis and implementation in the appendix. For the frequentist approach, we formulated the inference as an optimization problem aimed at maximizing an approximated likelihood derived from KDE, and we utilized Differential Evolution as a robust global optimization strategy capable of navigating potentially complex objective function landscapes.

Our numerical experiments on synthetic data provided valuable validation for the developed methods. We demonstrated the ability of both the KDE-based MCMC and the frequentist optimization approach using Differential Evolution to successfully recover the known parameters of a LogNormBetaBeta distribution, indicating the potential for accurate inference under this model. Challenges emerged when working with the TriLog-Normal distribution, which highlighted issues of model non-identifiability due to permutations of underlying distributions. Nonetheless, the optimization method proved capable of locating regions of high likelihood corresponding to valid parameter sets, illustrating its robustness even in multi-modal scenarios. A sensitivity analysis on key MCMC hyperparameters offered crucial insights into tuning the algorithm for improved convergence behavior and estimation accuracy. When applying the frequentist approach and its associated model selection criterion to real experimental granulometer data, we found that the LogNormal distribution provided the best fit among the simple Uniform, LogNormal, and Gamma models considered. However, a visual comparison of the resulting synthetic data with the real observations revealed that even this best-fitting simple model did not fully capture the nuances of the real particle shape distribution, particularly for the minimum projected dimension.

This project successfully established and validated a simulation-based pipeline for inferring 3D particle shape parameters from 2D measurements. By implementing and comparing different inference methods on both synthetic and real data, we gained significant insights into the feasibility and challenges of this inverse problem, providing a foundation for future research in this area.

Looking ahead, several promising avenues exist to build upon this work and enhance the capabilities of the inference framework. One crucial direction involves exploring more sophisticated and flexible parametric models for the 3D particle shapes. This could include investigating models beyond simple ellipsoids, such as super-ellipsoids, or employing multivariate probability distributions that can explicitly account for complex correlations between the particle's dimensions. Furthermore, efforts could be directed towards improving the computational efficiency and robustness of the inference techniques themselves. This might involve exploring more advanced MCMC samplers that can navigate high-dimensional spaces more effectively, or investigating alternative likelihood approximation methods and distance metrics for ABC that offer better statistical properties or reduced computational cost. Deeper theoretical investigations into the identifiability of the chosen parametric models are also warranted to fully understand the inherent ambiguity in inferring 3D shapes from 2D data. Finally, exploring entirely different mathematical frameworks, such as those based on optimal transport theory or diffusion models, could potentially offer novel and powerful approaches to compare and infer the parameters of complex probability distributions in this context. By pursuing these directions, the tools and insights gained from this project can contribute to more accurate and versatile methods for the 3D characterization of particles from readily available 2D measurements, benefiting various fields in materials science and engineering.

Appendix A

Detailed Projection Computation

In this section, we provide a detailed derivation of the analytical method used to compute the semi-axes of the 2D ellipse resulting from the projection of a 3D ellipsoid onto the XY plane, following a random rotation.

We begin with the equation of a 3D ellipsoid centered at the origin with semi-axes a, b, c aligned with the coordinate axes:

$$\frac{x'^2}{a^2} + \frac{y'^2}{b^2} + \frac{z'^2}{c^2} = 1 \tag{A.1}$$

This can be written in matrix form as $\mathbf{X}'^T D \mathbf{X}' = 1$, where $\mathbf{X}' = (x', y', z')^T$ and D is a diagonal matrix containing the inverse squares of the semi-axes:

$$D = \begin{pmatrix} 1/a^2 & 0 & 0 \\ 0 & 1/b^2 & 0 \\ 0 & 0 & 1/c^2 \end{pmatrix} \tag{A.2}$$

When the ellipsoid is rotated by a rotation matrix $R \in SO(3)$, a point $\mathbf{X}'$ in the original frame is related to a point $\mathbf{X} = (X, Y, Z)^T$ in the rotated frame by $\mathbf{X}' = R\mathbf{X}$. Substituting this into the equation of the ellipsoid, we get:

$$(R\mathbf{X})^T D (R\mathbf{X}) = 1 \tag{A.3}$$

Using the property of matrix transposition $(AB)^T = B^T A^T$, this becomes:

$$\mathbf{X}^T R^T D R \mathbf{X} = 1 \tag{A.4}$$

Let $Q = R^T D R$. The equation of the rotated ellipsoid in the new coordinate system is:

$$\mathbf{X}^T Q \mathbf{X} = 1 \tag{A.5}$$

Expanding this quadratic form in terms of the coordinates X, Y, Z and the entries of the symmetric matrix Q ($Q_{ij} = Q_{ji}$), we get:

$$Q_{11}X^2 + Q_{22}Y^2 + Q_{33}Z^2 + 2Q_{12}XY + 2Q_{13}XZ + 2Q_{23}YZ = 1 \tag{A.6}$$

To find the projection of the ellipsoid onto the XY plane, we seek the envelope of the ellipsoid with respect to the Z-direction. This corresponds to the set of points (X, Y) in the XY plane for which there is a unique solution for Z on the ellipsoid. We can rewrite the equation as a quadratic in Z:

$$Q_{33}Z^2 + (2Q_{13}X + 2Q_{23}Y)Z + (Q_{11}X^2 + 2Q_{12}XY + Q_{22}Y^2 - 1) = 0 \tag{A.7}$$

A unique solution for Z exists when the discriminant of this quadratic equation is zero. The discriminant Δ is given by $b^2 - 4ac$ for a quadratic $aZ^2 + bZ + c = 0$. In our case, $a = Q_{33}$, $b = (2Q_{13}X + 2Q_{23}Y)$, and $c = (Q_{11}X^2 + 2Q_{12}XY + Q_{22}Y^2 - 1)$. Setting the discriminant to zero gives:

$$(2Q_{13}X + 2Q_{23}Y)^2 - 4Q_{33}(Q_{11}X^2 + 2Q_{12}XY + Q_{22}Y^2 - 1) = 0 \tag{A.8}$$

Expanding and rearranging this equation gives the equation of the projected ellipse in the XY plane:

$$\begin{aligned} &4Q_{13}^2X^2 + 8Q_{13}Q_{23}XY + 4Q_{23}^2Y^2 \\ &- 4Q_{33}Q_{11}X^2 - 8Q_{33}Q_{12}XY - 4Q_{33}Q_{22}Y^2 + 4Q_{33} = 0 \end{aligned} \tag{A.9}$$

$$(4Q_{13}^2 - 4Q_{33}Q_{11})X^2 + (8Q_{13}Q_{23} - 8Q_{33}Q_{12})XY + (4Q_{23}^2 - 4Q_{33}Q_{22})Y^2 + 4Q_{33} = 0 \tag{A.10}$$

Dividing by 4 and rearranging to the form $EX^2+FXY+GY^2 = 1$, we get the coefficients:

$$E = -\frac{Q_{13}^2 - Q_{33}Q_{11}}{Q_{33}} = Q_{11} - \frac{Q_{13}^2}{Q_{33}} \tag{A.11}$$

$$F = -\frac{2Q_{13}Q_{23} - 2Q_{33}Q_{12}}{Q_{33}} = 2Q_{12} - 2\frac{Q_{13}Q_{23}}{Q_{33}} \tag{A.12}$$

$$G = -\frac{Q_{23}^2 - Q_{33}Q_{22}}{Q_{33}} = Q_{22} - \frac{Q_{23}^2}{Q_{33}} \tag{A.13}$$

(This assumes $Q_{33} \neq 0$, which is true unless the ellipsoid is infinitely thin and oriented perfectly within the XY plane).

The equation of the projected ellipse is thus $EX^2 + FXY + GY^2 = 1$. The lengths of the semi-axes of this ellipse are related to the eigenvalues of the matrix representing this quadratic form in 2D. The matrix is given by:

$$M = \begin{pmatrix} E & F/2 \\ F/2 & G \end{pmatrix} \tag{A.14}$$

The eigenvalues λ_1 and λ_2 of this matrix M are related to the squared lengths of the semi-axes of the projected ellipse (A^2 and B^2) by the relationship:

$$A^2 = \frac{1}{\lambda_{\text{min}}}, \quad B^2 = \frac{1}{\lambda_{\text{max}}} \tag{A.15}$$

where λ_{min} and λ_{max} are the minimum and maximum eigenvalues of M, respectively. The semi-axes lengths are therefore $A = 1/\sqrt{\lambda_{\text{min}}}$ and $B = 1/\sqrt{\lambda_{\text{max}}}$. The eigenvectors associated with these eigenvalues give the direction of the major and minor axes of the projected ellipse.

Appendix B

Approximate Bayesian Computation (ABC) Rejection Sampling

This appendix provides a detailed description of the Approximate Bayesian Computation (ABC) rejection sampling method implemented and used in this project as an alternative Bayesian inference technique. ABC is particularly useful in scenarios where the likelihood function is computationally intractable, but it is possible to simulate data from the forward model given a set of parameters.

B.1 ABC Rejection Sampling Algorithm

The core idea behind ABC rejection sampling is to directly sample candidate parameter values from the prior distribution and accept them if the synthetic data they generate is "close enough" to the observed data in terms of their summary statistics. Similarity is assessed by comparing a set of summary statistics computed from both the synthetic and observed datasets.

The algorithm proceeds as follows:

1. Initialize an empty set of accepted parameter samples $\mathcal{S} = \emptyset$.
2. For $i = 1$ to $N_{\text{iterations}}$:
    (a) Draw a parameter vector θ^* from the prior distribution $p(\theta)$.
    (b) Simulate a dataset $\mathcal{D}^*$ from the forward model parameterized by θ^*, i.e., $\mathcal{D}^* \sim p(\mathcal{D} \mid \theta^*)$. The size of $\mathcal{D}^*$ is typically the same as the observed dataset $\mathcal{D}_{\text{obs}}$.
    (c) Compute a set of relevant summary statistics $s(\mathcal{D})$ for the observed data $\mathcal{D}_{\text{obs}}$ and s^* for the simulated data $\mathcal{D}^*$.
    (d) Choose a distance metric $\rho(\cdot,\cdot)$ and compute the distance between the observed and simulated summary statistics, $d = \rho(s_{\text{obs}}, s^*)$.

(e) Define a tolerance threshold ϵ. If $d \leq \epsilon$, accept the parameter vector θ^* and add it to the set of accepted samples: $\mathcal{S} = \mathcal{S} \cup \{\theta^*\}$.

3. After $N_{\text{iterations}}$ simulations, the set $\mathcal{S}$ constitutes an approximate sample from the posterior distribution $p(\theta \mid \mathcal{D}_{\text{obs}})$.

The quality of the ABC approximation depends crucially on the choice of summary statistics (ideally sufficient statistics), the distance metric, and the tolerance ϵ.

B.2 Theoretical Justification

While in practice a non-zero tolerance $\epsilon > 0$ is always used, the theoretical validity of ABC can be most clearly understood by considering the ideal case where $\epsilon = 0$. In this scenario, a candidate parameter θ^* drawn from the prior is accepted only if the simulated dataset $\mathcal{D}^*$ generated from θ^* *exactly matches* the observed dataset $\mathcal{D}_{\text{obs}}$. That is, acceptance occurs if $\rho(s(\mathcal{D}_{\text{obs}}), s(\mathcal{D}^*)) = 0$, which for perfect summary statistics and distance, implies $\mathcal{D}^* = \mathcal{D}_{\text{obs}}$.

Let $\pi(\theta) = p(\theta)$ denote the prior distribution. The process of drawing θ^* from the prior and then simulating $\mathcal{D}^*$ from the model given θ^* means that the joint distribution of $(\theta^*, \mathcal{D}^*)$ before any acceptance step is $p(\mathcal{D}^* \mid \theta^*)\pi(\theta^*)$.

Now, consider the distribution of the pairs $(\theta^*, \mathcal{D}^*)$ that are *accepted* by the algorithm with $\epsilon = 0$. Acceptance occurs if and only if $\mathcal{D}^* = \mathcal{D}_{\text{obs}}$. Let $f(\theta^*, \mathcal{D}^*)$ denote the joint distribution of such accepted pairs. This distribution is proportional to the original joint distribution, but restricted to the case where $\mathcal{D}^* = \mathcal{D}_{\text{obs}}$:

$$f(\theta^*, \mathcal{D}^*) \propto p(\mathcal{D}^* \mid \theta^*)\pi(\theta^*)\mathbb{I}(\mathcal{D}^* = \mathcal{D}_{\text{obs}}) \tag{B.1}$$

where $\mathbb{I}(\mathcal{D}^* = \mathcal{D}_{\text{obs}})$ is an indicator function that is 1 if $\mathcal{D}^* = \mathcal{D}_{\text{obs}}$ and 0 otherwise.

To find the marginal distribution of the accepted parameter values θ^*, we marginalize the joint distribution $f(\theta^*, \mathcal{D}^*)$ over $\mathcal{D}^*$:

$$\begin{aligned} p_{\text{accepted}}(\theta^*) &\propto \int p(\mathcal{D}^* \mid \theta^*)\pi(\theta^*)\mathbb{I}(\mathcal{D}^* = \mathcal{D}_{\text{obs}})\, d\mathcal{D}^* && \text{(B.2)} \\ &\propto p(\mathcal{D}_{\text{obs}} \mid \theta^*)\pi(\theta^*) \int \mathbb{I}(\mathcal{D}^* = \mathcal{D}_{\text{obs}})\, d\mathcal{D}^* && \text{(B.3)} \end{aligned}$$

The integral $\int \mathbb{I}(\mathcal{D}^* = \mathcal{D}_{\text{obs}})\, d\mathcal{D}^*$ is a constant with respect to θ^* (it's essentially evaluating the "mass" at the single point $\mathcal{D}_{\text{obs}}$ in the space of $\mathcal{D}^*$). Therefore, the marginal distribution of the accepted θ^* is proportional to $p(\mathcal{D}_{\text{obs}} \mid \theta^*)\pi(\theta^*)$:

$$p_{\text{accepted}}(\theta^*) \propto p(\mathcal{D}_{\text{obs}} \mid \theta^*)\pi(\theta^*) \tag{B.4}$$

By Bayes' theorem, $p(\mathcal{D}_{\text{obs}} \mid \theta^*)\pi(\theta^*) \propto p(\theta^* \mid \mathcal{D}_{\text{obs}})$. Thus, in the limit where $\epsilon = 0$ and using sufficient statistics (or comparing the full datasets), the accepted parameter samples θ^* are indeed drawn exactly from the true posterior distribution $p(\theta \mid \mathcal{D}_{\text{obs}})$.

While this demonstrates the theoretical exactness of ABC at $\epsilon = 0$, this scenario is practically impossible for continuous data or high-dimensional discrete data due to the curse of dimensionality; the probability of simulating a dataset $\mathcal{D}^*$ that exactly matches $\mathcal{D}_{\text{obs}}$ is typically zero. Hence, in practice, a positive tolerance $\epsilon > 0$ is used, leading to an approximation of the true posterior.

B.3 Implementation Details: The `RejectionSampling` Class

Our implementation of the ABC rejection sampling algorithm is encapsulated within the `RejectionSampling` Python class of our `abc_sampling` package.

B.3.1 Class Attributes

The `RejectionSampling` class is initialized with key components required for the ABC process:

- `simulator`: An instance of the forward model simulator (`GranulometreSimulator`) that can generate synthetic 2D projection data given 3D parameters. This maintains a clear separation between the simulation model and the inference algorithm.
- `dataAB`: Contains the observed 2D projection data, against which simulated data will be compared.
- `prior_sampler`: An object representing the prior distribution. By providing this as a separate object with a `.sample()` method, the `RejectionSampling` class is agnostic to the specific type of prior used.
- `distance_func`: A function that computes the distance between simulated and observed data. This abstraction allows for easy swapping of different distance metrics or summary statistic comparisons.
- `accepted_params`: A list to store the parameter vectors that satisfy the acceptance criterion ($d \leq \epsilon$).
- `history`: A list to store the outcome of every simulation attempt (the proposed parameter vector and the computed distance), which is useful for post-run analysis and diagnostics.

- `best_dist`: Tracks the minimum distance observed throughout the sampling, providing an indication of how well the model could fit the data within the explored parameter space.

B.3.2 The Sampling Process (`run` Method)

The main logic of the ABC rejection sampling algorithm is executed within the `run(N_iter, epsilon)` method. This method iterates a specified number of times (N_{iter}), performing the core steps of the algorithm:

1. Sampling a new parameter vector θ^* from the `prior_sampler`.
2. Updating the `simulator` with θ^* and generating a synthetic dataset $\mathcal{D}^*$ of the same size as the observed data.
3. Computing the distance between $\mathcal{D}^*$ and the observed data using the provided `distance_func`.
4. Storing the proposed θ^* and its distance in the `history`.
5. Checking if the distance is less than the tolerance ϵ. If so, θ^* is added to the `accepted_params` list.

This method encapsulates the entire sampling loop, providing a clean interface to initiate and control the ABC process.

B.3.3 Summary Statistics and Distance Metric

An important step in ABC is the selection of a distance metric $\rho(\mathcal{D}_{\text{obs}}, \mathcal{D}^*)$ that quantifies the discrepancy between the observed data $\mathcal{D}_{\text{obs}}$ and the simulated data $\mathcal{D}^*$. This distance implicitly relies on comparing informative characteristics of the datasets, which can be viewed as summary statistics.

For our problem, where the data consists of empirical distributions of 2D projected semi-axes (A, B), comparing entire distributions is more robust than comparing simple low-dimensional summary statistics (like means or variances) which might not capture the full shape of the distribution. We chose to use the Wasserstein distance to compare the empirical distributions of the observed and simulated projected semi-axes.

The Wasserstein distance provides a measure of the "cost" of transforming one probability distribution into another. For two one-dimensional empirical distributions based on samples $\{u_1, \dots, u_m\}$ and $\{v_1, \dots, v_n\}$, the L_1 Wasserstein distance (W_1) is given by:

$$W_1(U, V) = \int_{-\infty}^{\infty} |F_U(x) - F_V(x)| dx \tag{B.5}$$

where $F_U(x)$ and $F_V(x)$ are the empirical cumulative distribution functions (CDFs) of the samples $U = \{u_i\}$ and $V = \{v_j\}$, respectively. For comparing our 2D data (A, B) to (A^*, B^*), we computed the sum of the 1D Wasserstein distances for the marginal distributions of A and B separately, i.e., $\rho(\mathcal{D}_{\text{obs}}, \mathcal{D}^*) = W_1(A_{\text{obs}}, A^*) + W_1(B_{\text{obs}}, B^*)$. This distance quantifies how much "effort" is required to move the mass of one empirical distribution to match the other.

This choice of distance allows us to compare the overall shapes of the simulated and observed distributions of projected dimensions, providing a more comprehensive comparison than simple moment matching.

B.3.4 Analyzing ABC Results: Auxiliary Classes

Beyond the core `RejectionSampling` class, which performs the parameter sampling and acceptance, auxiliary classes were developed for post-sampling analysis and visualization of the results. The primary output of the `RejectionSampling` algorithm is the history of all attempted parameter proposals and their corresponding computed distances to the observed data.

These auxiliary classes include:

- `Interpolator`: This class takes the sampling history (parameter-distance pairs) and trains a regression model (e.g., using scikit-learn) to approximate the distance function across the parameter space. This allows for estimating the distance for parameter values that were not explicitly sampled during the rejection sampling process.

- `HistoryPlotter`: This class provides visualization tools to help understand the results stored in the sampling history and the landscape approximated by the `Interpolator`.

When the parameter space is low-dimensional (e.g., 2D, as in some of our simplified models), the output of the `Interpolator` can be visualized as an interpolated distance surface. Figure B.1 shows an example of such a surface fitted to the history of sampled parameters and their calculated distances. The scatter points represent the individual proposals evaluated by the ABC sampler, colored by their distance. The surface represents the smoothed approximation of the distance function across the parameter space.

Intuitively, regions in the parameter space that result in lower distances to the observed data correspond to parameter values that are more compatible with the data under the model. In the context of Bayesian inference, these low-distance regions intuitively align with areas of higher posterior probability density. Thus, the interpolated distance surface can serve as a visual proxy for the shape of the posterior distribution, with minima on the surface suggesting likely locations for the posterior mode.

However, while the accepted samples from the ABC rejection algorithm with a sufficiently small tolerance ϵ have theoretical guarantees to approximate the true posterior distribution, we have not formally explored or established similar theoretical guarantees for the fidelity of this interpolated distance surface as a precise representation of the posterior density landscape. Its primary role in this study is as a diagnostic tool to visualize the relationship between parameters and model fit as measured by the chosen distance metric, and to gain intuition about the likely location of high-density regions explored by the sampler.

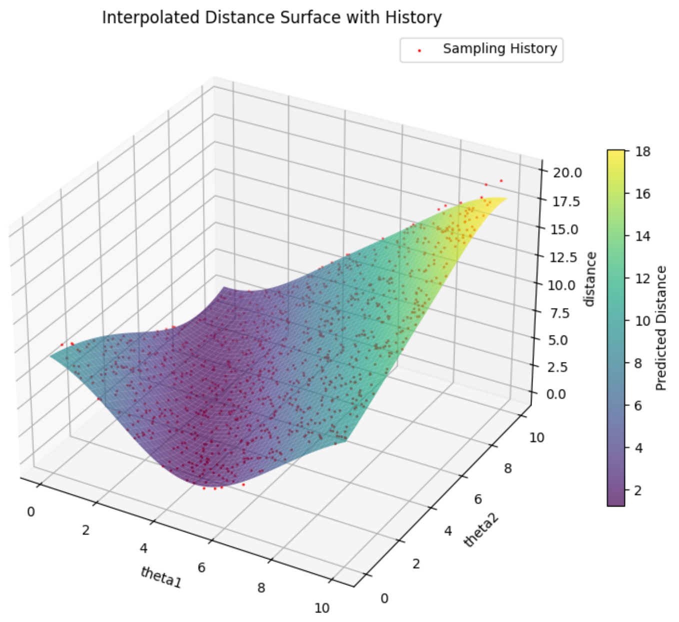


Figure B.1: Interpolated distance surface fitted to the history of parameter proposals and their corresponding distances. Low distance regions intuitively indicate higher posterior probability.

Bibliography

Bernard W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall/CRC, 1986.

Rainer Storn and Kenneth Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4): 341–359, 1997. doi: 10.1023/A:1008202821328.

Jiongyi Wu, Yahong Yuan, Hui Gong, and T. L. (Bill) Tseng. Inferring 3d ellipsoids based on cross-sectional images with applications to porosity control of additive manufacturing. *IISE Transactions*, 50(7):570–583, 2018. doi: 10.1080/24725854.2017.1419316.