Bar-Ilan University

Estimating the Decay Rate of Small Deviations of Gaussian Stationary Processes

Yuval Lavie

Submitted in partial fulfilment of the requirements for the Master's Degree in the Department of Mathematics, Bar-Ilan University.

Ramat Gan, Israel 2023

This work was carried out under the supervision of Prof. Simi Haber and Doctor Naomi Feldheim at the Department of Mathematics, Bar-Ilan University.

Acknowledgements

I would like to express my deepest gratitude to Professor Simi Haber and Doctor Naomi Feldheim for their invaluable guidance and support throughout my thesis research. Their expertise, patience, and encouragement have been instrumental in shaping my academic journey and achieving my research goals.

Contents

Abstract

A Gaussian process is a stochastic process such that every finite collection of random variables derived from the process has a multivariate normal distribution. These processes are extremely useful in practice and are used as basis for many novel machine learning techniques and time series prediction models. We focus our attention on Centered Stationary Gaussian Processes with continuous paths (CSGP) and look for algorithms which estimate small and large deviations within a valid and practical precision. In this thesis we tackle the challenge of estimating rare events by presenting contemporary methods of estimation and proposing new estimation and simulation methods.

1 Introduction

The focus of this work is on the numerical estimation of the **persistence** and **ball** probabilities of a real Gaussian stationary stochastic process. The former is the event where a process stays above a level ℓ and the latter is the event where a process stays inside a symmetric set $[-\ell, \ell]$ over a long time interval $[0, T]$. These quantities appear in many theoretical and practical applications. In economy, the ball event is used to describe the event that a certain stock price remains between two barriers in the given time interval. The persistence event, also known as the survival event, describes the event that a system does not fail by going below a critical threshold in the given time interval. Our interest lies in the decay rate of these probabilities as the time interval grows to infinity. We begin this section by introducing the necessary definitions and continue to show that under mild conditions, the behavior of both the persistence and the ball probabilities decays exponentially in T.

1.1 Definitions

Let $\mathcal{T}$ be a topological space, and $(\Omega, S, \mathbb{P})$ a probability space. The triplet (X, μ, K) *is called a Gaussian process* if:

1. $X : \Omega \times \mathcal{T} \to \mathbb{R}$ is a measurable functional that induces multi-normal marginal distributions at any finite times $(t_1, ..., t_d)$.
2. $\mu : \mathcal{T} \to \mathbb{R}$ is a mapping which represents the *mean values* of the process.
3. $K : \mathcal{T} \times \mathcal{T} \to \mathbb{R}$ is a positive semi-definite mapping representing the *covariance* between any two samples of X.

In other words, every finite dimensional random variable induced by X follows the normal law, as follows:

$$\forall d \in \mathbb{N} \setminus \{\infty\}, \forall t_1, ..., t_d \in \mathcal{T} : \left(X_{t_1}, ..., X_{t_d} \right) \sim \mathcal{N}_d \left(\mu|_{\{t_1,...,t_d\}}, K|_{s,t \in \{t_1,...,t_d\}} \right).$$

We restrict our attention to real-timed ($\mathcal{T} = \mathbb{R}$) Gaussian processes X with the following properties:

1. **Continuous**: $t \to X_t(\omega) \in C(\mathbb{R})$ a.s.
2. **Centered**: $\forall t \in \mathbb{R}$ we have that $\mu(t) = 0$.
3. **Stationary**: X is invariant to translations of elements in $\mathbb{R}$. That is:

$$\forall t_1, ..., t_d \in \mathbb{R}, h > 0 : \left(X_{t_1}, ..., X_{t_n} \right) \sim \left(X_{t_1+h}, ..., X_{t_n+h} \right).$$

Processes satisfying all of these properties shall be called *centered stationary Gaussian processes* (CSGP). These are widely used, in particular in applications for Machine learning [13] and statistical inference. The following properties make them particularly convenient to work with:

1. The covariance kernel is in fact a function of one real variable:

$$\forall s,t \in \mathbb{R}: \quad K(s,t) = \mathrm{cov}(X_s, X_t) = \mathrm{cov}(X_0, X_{t-s}) = r(t-s).$$

2. The multivariate central limit theorem states that a sum of independent and identically distributed random variables may be approximated by the multivariate normal distribution.

3. The projection of a Gaussian process of a linear subspace is a multivariate Gaussian vector.

4. Conditioning the Gaussian process on specific values of coordinates or a linear combination of them is a multivariate Gaussian vector.

Consider a real centered stationary Gaussian process, $X : \mathbb{R} \times \mathcal{P} \to \mathbb{R}$ admitting a covariance kernel $r(\tau)$. The persistence and ball probabilities are defined below.

Definition 1. The **ball** probability of a Gaussian process is defined as:

$$B_X(T,\ell) = \mathbb{P}\left[\sup_{t\in[0,T]} |X_t| \leq \ell\right] = \mathbb{P}\Big[|X_t| \leq \ell, \forall t \in [0,T]\Big].$$

Definition 2. The **persistence** probability of a Gaussian process is defined as:

$$P_X(T,\ell) = \mathbb{P}\left[\inf_{t\in[0,T]} X_t > \ell\right] = \mathbb{P}\Big[X_t > \ell, \forall t \in [0,T]\Big].$$

1.2 The exponential behavior of both quantities.

We are now equipped with all the fundamental knowledge needed to prove that both the persistence and ball probabilities decay exponentially with the length of the interval $[0,T]$.

1.2.1 Exponential behavior of the ball probability.

The proof of this section is rather simple and relies on the following two lemmas. The first is the Sub-additivity lemma proved by Fekete [7].

Lemma 3 (sub-additivity)**.** *Let a_n be a sub-additive measurable function or a sequence of non-negative real numbers such that*

$$a_{n+m} \leq a_n + a_m.$$

Then, the limit $\lim_{n\to\infty} \frac{a_n}{n}$ exists and is equal to $\inf_{n\geq 1} \frac{a_n}{n}$.

The second is a corollary of the Gaussian correlation inequality proved by Royen [14, 12]. This is an extension of the Khatri-Sidak lemma [11].

Lemma 4 (Gaussian correlation)**.** *Let X_t be a continuous stationary Gaussian process, Then for all $n, m > 0, \lambda \in \mathbb{R}$:*

$$\mathbb{P}\left(\sup_{t\in[0,n+m]} |X_t| \leq \lambda\right) \geq \mathbb{P}\left(\sup_{t\in[0,n]} |X_t| \leq \lambda\right) \cdot \mathbb{P}\left(\sup_{t\in[0,m]} |X_t| \leq \lambda\right).$$

The following claim states that the ball probability decays exponentially uniformly over all kernel functions $r(\tau)$ defining a Gaussian process.

Claim 5. The decay of the ball probability B_X is exponential in T with rate:

$$\theta_X^\ell = \lim_{T\to\infty} -\frac{1}{T}\log B_X(T,\ell).$$

That is, the ball probability can be represented as:

$$B_X(T,\ell) = e^{-\theta_X^\ell \cdot T + o(T)}.$$

Proof. Without loss of generality, fix $\ell \in \mathbb{R}$ and set $\alpha_T = B_X(T,\ell)$. Then, by Lemma 4

$$\forall T_1, T_2 : \alpha_{T_1+T_2} \geq \alpha_{T_1} \cdot \alpha_{T_2}.$$

Applying the monotone logarithm function and multiplying by minus one, we get that:

$$-\log \alpha_{T_1+T_2} \leq -\log \alpha_{T_1} + (-\log \alpha_{T_2}).$$

Notice that $\alpha_T \in [0,1]$ and set $\beta_T = -\log \alpha_T$ as the sequence of non-negative real numbers required by Fekete's lemma. Then, by applying Fekete's lemma (Lemma 3) we have that the rate function exists as a limit and is equal to:

$$\theta_X^\ell = \inf_{n\geq 1} \frac{\beta_T}{T}.$$

□

1.2.2 Exponential behavior of P_X

Unlike the case of the previous section, the behavior and existence of the persistence exponent is not uniformly explained for the whole class of Gaussian stationary processes. In his celebrated 1962 paper, Slepian conjectured that this exponent should exist under mild conditions and the validity of his conjecture has been taken for granted in the physics literature. While proofs of existence were readily available for non-negative correlated processes [5, 6], Markov processes [2] and auto-regressive processes [1] in the past, the general case has been long open. It was only quite recently that the existence of the exponent γ has been shown to be related to a specific measure defined by Bochner in the following theorem.

Theorem 6 (Bochner's theorem [3]). *Let $r : T \to \mathbb{R}$ be a continuous positive-definite function. Then, there exists a finite, symmetric and non-negative probability measure ρ, called the spectral measure, such that:*

$$r(t) = \mathcal{F}[\rho](t) = \int_T e^{-i\lambda t} d\rho(\lambda).$$

We focus our attention on the set of spectral measures $\mathcal{S}$ whose elements have finite $\log -(1+\beta)$ moment. That is:

$$\mathcal{L} = \left\{ \rho \in \mathcal{S} : \exists \beta > 0 : \int_0^\infty \max\left(\log^{1+\beta} \lambda, 1\right) d\rho(\lambda) < \infty \right\}.$$

Theorem 7 (Existence of the persistence exponent [8]). *Let X be a real centered stationary Gaussian process that admits a spectral measure $\rho \in \mathcal{L}$. if the limit*

$$\rho'(0) = \lim_{\epsilon \to 0} \frac{1}{2\epsilon} \rho([-\epsilon, \epsilon])$$

exists and $\rho'(0)$ is positive then the persistence exponent exists as the limit:

$$\gamma_X^\ell = \lim_{T \to \infty} -\frac{1}{T} \log P_X(T, \ell).$$

1.3 The goal of this Work

The primary objective of this thesis is to provide a comprehensive analytical and computational framework for estimating the ball probability exponent in stationary Gaussian processes. This work uniquely contributes to the field in the following ways:

1. **Bridging Theory and Practice**: A core ambition of this thesis is to seamlessly bridge the gap between theoretical insights and practical applications. By reviewing the current literature, leveraging analytical methods, computational algorithms, and statistical techniques, this work aims to create a robust toolkit that is not only grounded in rigorous mathematical theory but is also practically implementable for real-world applications.

2. **Analytical Estimations for Specific Models**: The thesis first tackles the ball probability estimation for Markov-Gauss processes and Gaussian Graph Models using analytical methods. This analytical treatment extends the current theoretical understanding of these models.

3. **Spectral Sampling Method for CSGPs**: A Python implementation of a Spectral Sampling method is introduced, designed specifically for Stationary Gaussian processes with band-limited covariance kernels. This method helps with Monte Carlo estimations but can be useful for other purposes such as digital signal processing and machine learning.

4. **Importance Sampling with Optimized Proposal Density**: Building on the established theory of Importance Sampling, an optimized proposal density is estimated via a proposed algorithm for sampling from the multivariate truncated normal distribution. We also propose a computationally efficient estimator for the ball probability when the ball is very small.

5. **Neural Density Model for Ball Probability**: As a pioneering approach, this thesis introduces the use of neural density models (Neural mixture models and Normalizing flows) for estimating the exponent of the ball probability in Gaussian processes. This technique aims to offer a flexible and scalable alternative to traditional methods, potentially capturing complex patterns in the data more effectively.

6. **Python implementation for researchers**: We include a Python implementation of an algorithm which either accepts a method that estimates the ball probability, or uses those proposed in this thesis, and returns the ball exponent.

By synthesising analytical theory with innovative computational algorithms, this thesis aims to advance both the theoretical and practical aspects of ball probability estimation in Gaussian processes.

Next, in Section 2 we present an overview of the relevant literature. Section 3, which is the main part of this thesis, describes and compares the algorithms we obtained during our research. In Section 4 we conclude our work and propose new research directions relevant to this thesis.

2 Literature review

The goal of this section is to briefly survey the knowledge available in the numerical computation literature, the next section will focus on aspects focused on our processes. A researcher observing a system described by a random variable X is interested in the probability that the system is in state D. That is, the calculation of the integral

$$p = \mathbb{P}(X \in D) = \int_D dF_X = \int_D f_X dx.$$

In most real world cases where closed-form solutions are not available, the researcher must rely on computational methods to calculate the integral. We begin by briefly describing some deterministic methods of integration and move towards random methods which require the knowledge of sampling from various distributions.

2.1 Deterministic integration

We briefly describe the three most common methods of integration. Assume that X is one dimensional and that $D = [a, b]$, the integral to be calculated is

$$p = \mathbb{P}(X \in [a, b]) = \int_a^b f_X dx.$$

To calculate p, we use a partition of the interval $[a, b]$ using $m + 1$ points: $x_0 = a < x_1 < ... < x_{m-1} < x_m = b$, and denote $\Delta_k = x_k - x_{k-1}$.

1. **Middle point method.** Using piecewise-constant interpolation:

$$p \approx \sum_{i=1}^{m} f_X \left(\frac{x_i - x_{i-1}}{2} \right) \cdot \Delta_i.$$

2. **Trapezoid rule.** Using linear interpolation:

$$p \approx \sum_{i=1}^{m} \frac{f_X(x_{i-1}) + f_X(x_i)}{2} \cdot \Delta_i.$$

3. **Simpson's rule.** Using quadratic interpolation:

$$p \approx \sum_{i=1}^{m} \frac{\Delta_i}{3} \left[f_X(x_i) + 4 f_X \left(\frac{x_{i+1} - x_i}{2} \right) + f_X(x_{i+1}) \right].$$

All three methods work well for one dimensional integrals with smooth integrands. When working with higher dimensions, the infamous **curse of dimensionality** states that the partitioning of a set D with evenly spaced points becomes exponentially more expensive and renders these methods inefficient.

2.2 Stochastic integration

Random integration methods rely on the ability of the researcher to sample psuedo-random variates, before describing these methods we remind the reader about the famous Inverse Transform method to sample from continuous random variables. Sampling from Gaussian stochastic processes will be handled later on.

2.2.1 Inverse transform sampling

This method relies on samples from the uniform distribution which can be done via the Mersenne-twister algorithm or any other (random enough) pseudorandom number generator. Let $U \sim \mathcal{U}(0,1)$ be the standard uniform variable. The following claim describes the necessary relation between standard uniforms and continuous random variables.

Claim 8. (Probability Integral Transform) Let X be a continuous random variable distributed by the law F_X. Then, the random variable $Y = F_X(X)$ is uniformly distributed on $(0,1)$.

Proof. Let $Y = F_X(X)$. Then, for all $y \in [0,1]$ we have that

$$\mathbb{P}(Y \leq y) = \mathbb{P}(F_X(X) \leq y) = \mathbb{P}\left(F_X^{-1}(F_X(X)) \leq F_X^{-1}(y)\right) = F_X\left(F_X^{-1}(y)\right) = y.$$

□

Notice that this claim implies that the random variable $F_X^{-1}(U)$ has the same distribution as X. That is:

$$F_X(X) = U \implies X = F_X^{-1}(U).$$

The following algorithm uses the probability integral transform to sample from any continuous random variable.

Algorithm 1 Inverse transform sampling (Python)

Input: m, F_X^{-1}

1. Generate S_U, a sized m sample of $U \sim \mathcal{U}(0,1)$.
2. Generate S_X, an empty list.
3. For each $u \in S_U$:
   Add $F_X^{-1}(u)$ to S_X.

Output: $S_X \sim F_X^m$.

2.2.2 Monte Carlo integration

Monte Carlo (MC) methods are a versatile class of computational algorithms that rely on random sampling to obtain numerical results. These methods are widely used in various fields such as physics, finance, statistics, and engineering. The Monte Carlo method can be seen as a loose generalisation of the Mean Value theorem for one dimensional functions which states that for an integration domain $D = [a, b]$, there exists a point $c \in D$ such that the integral of a continuous function f is:

$$I = \int_a^b f dx = f(c) \cdot (b - a).$$

The fundamental insight of the MC method is the ability to represent an integral as an expected value. Consider the problem of evaluating the following integral:

$$I = \int_D f(x) dx \tag{1}$$

where $f(x)$ is the function to be integrated over the domain D and set $p = \frac{1}{\text{Vol}(D)}$ to be the continuous uniform density over D. The integral can now be represented as an expected value:

$$I = \int_D f(x) dx = \int_D \frac{f(x)}{p(x)} \cdot p(x) dx = \mathbb{E}_p \frac{f}{p} = \text{Vol}(D) \cdot \mathbb{E}_p f. \tag{2}$$

The integral is then estimated by drawing n random samples from $p(x)$ and computing a re-weighted sample mean:

$$\hat{I}_n = \frac{\text{Vol}(D)}{n} \sum_{i=1}^{n} f(x_i) \tag{3}$$

2.3 Properties of Monte Carlo Methods

Monte Carlo estimators exhibit several key properties:

- **Unbiasedness**: The estimator is unbiased, meaning its expected value equals the true value.

$$\mathbb{E}[\hat{I}_n] = \mathbb{E}_p \left[\frac{1}{m} \sum_{i=1}^{m} \frac{g}{p} (u_i) \right] = \mathbb{E}_p \left[\frac{g}{p} \right] = I \tag{4}$$

- **Consistency** [1]: As the number of samples n approaches infinity, the Monte Carlo estimator converges in probability to the true value of the parameter. This is a result of the law of large numbers.

- **Efficiency**: The convergence rate of the Monte Carlo estimator is typically $O(n^{-1/2})$ where n is the number of samples. This convergence rate is independent of the dimensionality of the problem, making Monte Carlo methods particularly useful for high-dimensional problems.

- **Parallelism**: Monte Carlo simulations are inherently parallelizable, as each sample or simulation can be computed independently. This makes it well-suited for modern parallel computing environments.

These properties make Monte Carlo methods a powerful tool for numerical estimation and simulation of complex systems. We include a simple Python implementation (2) which demonstrates the abilities of this method.

Algorithm 2 Monte Carlo integration (Python)

Input: Real function g, Interval $[a, b]$

1. Sample $\{u_i\}_{i=1}^n$ variates from $U[a, b]$.
2. Calculate the estimator $T = \dfrac{b-a}{n} \sum_{i=1}^n g(u_i)$.

Output: estimator $T \approx \int_{[a,b]} g dx$.

2.3.1 Estimating probabilities

When the calculation of the cumulative distribution function is either infeasible or inefficient, we wish to estimate it using indirect methods like Monte Carlo. Recall that for a continuous random variable X, the probability that X belongs to a set D is simply an integral. That is:

$$p(D) = \mathbb{P}(X \in D) = \int_D dF_X = \int_D f_X dx.$$

The naive scheme to estimate a probability is to represent the integral as an expected value as:

[1] The law of large numbers ensures almost sure convergence of our estimator to the correct probability, and Glivenko and Cantelli's Fundamental Theorem of Statistics [9, 4] confirms that uniform convergence holds. The Dvoretzky-Kiefer-Wolfowitz inequality further quantifies the convergence rate of empirical distributions. (As an interesting aside, this theory also underpins machine learning, since Glivenko-Cantelli classes are synonymous with Vapnik-Chervonenkis [16] classes.)

$$p(D) = \int_{\mathrm{Supp}(X)} 1(X \in D) dF_X = \mathbb{E}_X 1_D.$$

We then approximate the probability by sampling from X and counting the fraction of samples that fall inside the region D. That is:

$$p(D) \approx \frac{|\{i : \ x_i \in D\}|}{n}$$

This method is demonstrated in Algorithm 3

Algorithm 3 Estimating probabilities (Python)

Input: Random variable X, Set D, max samples n

1. Sample $\{x_i\}_{i=1}^n$ independent variates from X.
2. Calculate the indicator function $\gamma_i = 1(x_i \in D)$.
3. Calculate the estimator $\hat{I}_n = \frac{1}{n} \sum_{i=1}^n \gamma_i$.

Output: estimator $\hat{I}_n \approx P(X \in D)$.

---

3 Research

This section focuses on ways to estimate the ball coefficient $\theta_K(\ell)$ either by estimating $B_K(T, \ell)$ or by introducing families of Gaussian processes for which we have have been able to find a solution analytically. To the best of our knowledge, this is the first time these families have been used in this context.

3.1 Analytical solutions

We now present our results for the analytical solution of the I.I.D and Markov(1) processes, we will use these results to test several other algorithms later on. We assume that the standard normal cumulative distribution function Φ and the 2 dimensional joint distribution $F_K(x_1, x_2)$, for any kernel K, are both readily available to the reader. For problems where a k-dimensional CDF is available through numerical methods we introduce Gaussian Markov models. In this context, these processes may have their ball exponents calculated efficiently.

3.1.1 Independent and Identically Distributed

The I.I.D assumption is so fundamental that it stands at the center of statistical inference and machine learning theories. These processes are easy to analyze and to simulate, but are usually an oversimplification of the real phenomena occuring in nature. To begin our analysis, let X be our process associated with the mean functional $\mu_t = \mu$ and a covariance kernel $K(s,t) = 1(s = t) \cdot \sigma^2$. We can conclude that $X_t \sim \mathcal{N}(\mu, \sigma^2)$ are independent Gaussian random variables and compute the ball[2] probability as:

$$B_K(T, \ell) = \mathbb{P}\left(X_1 \in [-\ell, \ell], ..., X_T \in [-\ell, \ell]\right) = \prod_{i=1}^{T} \mathbb{P}\left(X_i \in [-\ell, \ell]\right) = \left(\Phi\left(\frac{\ell - \mu}{\sigma}\right) - \Phi\left(\frac{-\ell - \mu}{\sigma}\right)\right)^T. \tag{5}$$

Taking the limit and the log we get that the decay coefficient can be directly calculated as:

$$\theta_K^\ell = \lim_{T \to \infty} -\frac{1}{T} \log B_X(T, \ell) = -\log\left(\Phi\left(\frac{\ell - \mu}{\sigma}\right) - \Phi\left(\frac{-\ell - \mu}{\sigma}\right)\right). \tag{6}$$

Notice that if the process is centred the calculation of the probability is simplified to $(2 \cdot \Phi(\frac{\ell}{\sigma}) - 1)$. The following table shows the values of the coefficient for some normal distributions.

[2]The persistence probability can be calculated in the same way.

Table 1: Summary of Calculated Results

μ	σ	$\ell = 0.25$	$\ell = 0.5$	$\ell = 1$
0	1	1.62246	0.959916	0.381715
0	2	2.30783	1.62246	0.959916
1	1	2.11215	1.41993	0.739715
1	2	2.43218	1.74488	1.07486
2	1	3.58147	2.8035	1.84957
2	2	2.80524	2.11215	1.41993

Table 2: This table presents the calculated results for various combinations of μ, σ, and ℓ, representing different IID Gaussian distributions and intervals. Python

3.1.2 First order Markov processes

The Markov assumption is a natural generalization of the independence assumption, recall that an independent process has the memory loss property:

$$\mathbb{P}(X_T|X_{T-1},...,X_1) = \mathbb{P}(X_T).$$

That is, all the information accumulated up to time $T-1$ is irrelevant to the distribution of X_T. A Markov process assumes that all the information that is relevant to the distribution of X_T is accumulated in X_{T-1}. That is:

$$\mathbb{P}(X_T|X_{T-1},...,X_1) = \mathbb{P}(X_T|X_{T-1}).$$

Now, let D be $[-\ell,\ell]$ and for simplicity we denote $X_{i,D} = X_i \in D$, then the ball probability becomes

$$p = \mathbb{P}\left(X_{1,D},...,X_{T,D}\right) = \mathbb{P}(X_{1,D}) \cdot \prod_{i=2}^{T} \mathbb{P}\left(X_{i,D}|X_{i-1,D}...,X_{1,D}\right).$$

Using both the Markov assumption and Stationarity we conclude that:

$$p = \mathbb{P}(X_{1,D}) \cdot \left(\frac{\mathbb{P}(X_{1,D},X_{2,D})}{\mathbb{P}(X_{1,D})}\right)^{T-1}.$$

The decay rate of the process corresponding to the set D is:

$$\theta_X^D = \lim_{T\to\infty} \frac{\log\frac{1}{\mathbb{P}(X_{1,D})} + (T-1)\log\frac{\mathbb{P}(X_{1,D})}{\mathbb{P}(X_{1,D},X_{2,D})}}{T} = \log\frac{\mathbb{P}(X_{1,D})}{\mathbb{P}(X_{1,D},X_{2,D})} \tag{7}$$

Since the calculation of the two dimensional CDF is available, we conclude that we can calculate the ball coefficient. To show the power of this method we introduce two stationary Markov processes.

1. The autoregressive process is widely used in econometrics and signal processing, it is mostly used to model interest rates and stock prices. We focus our attention on the $AR(1)$ model which assumes a linear dependence between each time step of the process using the following recurrence equation:

$$X_{t+1} = \alpha X_t + \epsilon_{t+1}, \text{ where } \epsilon_t \sim \mathcal{N}(0,\sigma_\epsilon^2).$$

This process is not naturally stationary, the condition for stationarity is that we restrict the recurrence coefficient (α) into the unit interval. To see it, we observe the first and second moment of the process.

$$\mathbb{E}X_{t+1} = \mathbb{E}\left[\alpha X_t + \epsilon_{t+1}\right] = \alpha \mathbb{E}X_t. \tag{8}$$

$$\text{Var}(X_{t+1}) = \text{Var}(\alpha X_t + \epsilon_{t+1}) = \alpha^2 \text{Var}(X_t) + \sigma_\epsilon^2. \tag{9}$$

Now, assuming that the process is stationary we get that:

$$\mathbb{E}X_t = 0, \text{Var}(X_t) = \frac{1}{1-\alpha^2}. \tag{10}$$

The stationary kernel function is given by:

$$\begin{aligned} K(t,t+h) &= \mathbb{E}\left[(X_{t+h} - \mathbb{E}X_{t+h})(X_t - \mathbb{E}X_t)\right] \\ &= \mathbb{E}\left[(\alpha^h X_t + \sum_{i=1}^{h} \epsilon_{t+i}) X_t\right] \\ &= \mathbb{E}\left[\alpha^h X_t^2 + \sum_{i=1}^{h} \epsilon_{t+i} X_t\right] \\ &= \alpha^h \mathbb{E}X_t^2 \\ &= \alpha^h \frac{\sigma_\epsilon^2}{1-\alpha^2}. \end{aligned} \tag{11}$$

2. The Ornstein-Uhlenbeck (OU) process is a continuous-time stochastic process that exhibits mean-reverting behavior. It is governed by the stochastic differential equation:

$$dX_t = \theta(\mu - X_t)dt + \sigma dW_t \tag{12}$$

where θ is the rate of mean reversion, μ is the long-term mean, σ is the volatility, and W_t is a standard Brownian motion. The OU process is Markovian, characterized by an exponential covariance function:

$$K(t,t+h) = \frac{\sigma^2}{2\theta} e^{-\theta|h|}, \quad \forall t,h \in \mathbb{R}. \tag{13}$$

It is often used to model various phenomena in finance, physics, and other fields, particularly when a tendency to revert to a long-term mean is present.

The following tables shows the results of calculating the coefficient directly for both processes.

α	σ	$\ell = 0.25$	$\ell = 0.5$
0.2	0.5	1.286959	0.655601
0.2	1	1.622864	0.961399
0.2	2	1.964062	1.286959
0.8	0.5	1.298818	0.696149
0.8	1	1.628951	0.983905
0.8	2	1.967146	1.298818

Table 3: Exact calculation for of the ball coefficient for various AR(1) processes.

θ	σ	$\ell = 0.25$	$\ell = 0.5$
0.2	0.5	1.536195	0.901636
0.2	1	1.872460	1.209758
0.2	2	2.213819	1.536195
0.8	0.5	1.289017	0.665375
0.8	1	1.623346	0.966348
0.8	2	1.963722	1.289017

Table 4: Exact calculation for of the ball coefficient for various OU processes.

3.1.3 Gaussian graph models

One generalisation of Gaussian-Markov processes is the Gaussian graphical model (GGM, [15]). In this model, there is an underlying graph $G = (V, E)$ where:

1. $V = (X_1, X_2, X_3, X_4, \ldots)$ are Gaussian variables.
2. E is a set of edges which represent the conditional dependence structure of each node.

First, consider the case of a Markov chain as visualized in Figure 1. In this example, every node X_t is connected to its predecessor X_{t-1}, representing the conditional independence of X_t on $\{X_{t-2}, ..., X_1\}$ when conditioned on X_{t-1}.

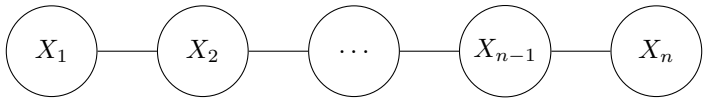


Figure 1: Illustration of a Markov(1) Gaussian graph model.

A Gaussian Markov chain can be represented as a GGM by presenting its structure as following:

1. $V = (X_1, X_2, X_3, X_4, \ldots, X_n)$.
2. $E = (X_1, X_2), (X_2, X_3), (X_3, X_4), \ldots, (X_{n-1}, X_n)$.

This chain-like structure is determined solely by $P = \Sigma^{-1}$, the precision matrix of the Gaussian, which reflects the conditional dependence of each dimension. The tri-diagonal[3] structure of the precision matrix of a Gaussian

[3]A Markov(k) process always induces an (k+1)-diagonal precision matrix and has its pdf decomposed accordingly.

Markov process allows us to decompose the quadratic form $X^T PX$ and the joint pdf into the product of marginals. That is:

$$P = \begin{bmatrix} \alpha & \beta & 0 & \dots & 0 \\ \beta & \alpha & \beta & \dots & 0 \\ 0 & \beta & \alpha & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \alpha \end{bmatrix} \implies f(x_1, x_2, x_3, \dots, x_n) = f(x_1) \cdot \prod_{i=2}^{n} f(x_i \mid x_{i-1}).$$

The fact that the precision matrix controls the conditional dependence between the variables is not limited to Markovian processes, we can create a different dependence structure by building the precision matrix directly. For example, we construct a 5-dimensional GMM with the following conditional independence structure:

- X_1 is independent of the other variables.
- X_2 is dependent on X_1 but independent of the rest given X_1.
- X_3 is dependent on X_1 but independent of the rest given X_1.
- X_4 is dependent on X_2 and X_3 but independent of the rest given X_2 and X_3.
- X_5 is dependent on X_4 but independent of the rest given X_4.

The corresponding graph is defined by:

1. $V = (X_1, X_2, X_3, X_4, X_5)$.
2. $(X_1, X_2), (X_1, X_3), (X_2, X_4), (X_3, X_4), (X_4, X_5)$

The precision matrix P corresponding to the above graph is:

$$P = \begin{bmatrix} a & b & c & 0 & 0 \\ b & d & 0 & e & 0 \\ c & 0 & f & g & 0 \\ 0 & e & g & h & i \\ 0 & 0 & 0 & i & j \end{bmatrix}$$

and the joint probability density function can be factorized as:

$$f(x_1) \cdot f(x_2 \mid x_1) \cdot f(x_3 \mid x_1) \cdot f(x_4 \mid x_3, x_2) \cdot f(x_5 \mid x_4).$$

We conclude that any Gaussian Graphical Model with dependence structure that allows for the efficient calculation of its marginal distributions can have its ball coefficient calculated efficiently. Two questions we leave open are:

1. How can one construct a valid precision matrix that will induce a stationary covariance matrix?
2. Which kernel functions induce sparse precision matrices other than Markov structures?

3.2 Numerical approximations

We move our focus to general families of Gaussians of which a special structure is unknown and the coefficients must be approximated via numerical methods. We present two algorithms which efficiently sample from Gaussian processes and and two justifications for such discretization. We continue to analyse the results of several approximation methods for the ball coefficient.

3.2.1 Sampling from a Gaussian process

Let X be a continuous time Gaussian process over the interval $[0, T]$. Since it is impossible for a computer with finite memory to simulate every point $X_{t\in[0,T]}$ we wish to partition the interval to $\lceil \frac{T}{\epsilon} \rceil + 1$ equidistant points where ϵ is the discretization rate and simulate X only on the discrete set

$$T_\epsilon = \{0, \epsilon, 2\epsilon, ..., T\}.$$

If X is a discrete time process then no discretization is needed and the simulation can be applied directly on its support $\{t_0, t_1, ..., T\}$. The following two methods allow us to sample Gaussian processes on discrete time sets.

3.2.2 Kernel method

The kernel method uses the linear combination representation of Gaussian vectors as described in the proposition below.

Proposition 9. *Linear combination representation*
Let X be a Gaussian process admitting a mean function μ and a covariance kernel K. Then, by definition any discretization $X^\epsilon = (X_0, X_\epsilon, X_{2\epsilon}, ...X_T) \sim \mathcal{N}(\mu_\epsilon, \Sigma_\epsilon)$ where μ_ϵ and Σ_ϵ are the corresponding mean vector and covariance matrix induced by the discretization. Since every multivariate normal vector can be represented as a linear combination of a matrix $A : \Sigma = AA^T$ and a multivariate normal vector $Z = (z_0, z_1, ..., z_T)$ we get that:

$$f^\epsilon = A_\epsilon Z + \mu_\epsilon.$$

and we simulate discrete Gaussian paths with the following algorithm.

Algorithm 4 Sampling a Gaussian process (Python)

Input: Kernel K, mean function μ, sampling rate ϵ, interval length T.

1. Create the equidistant partition T_ϵ of $[0, T]$ with sampling rate ϵ.
2. Calculate the induced covariance matrix Σ_ϵ.
3. Calculate the lower triangular root of the matrix $\sqrt{\Sigma_\epsilon} = A_\epsilon$ using Cholesky's decomposition.
4. Generate a vector of independent identically distributed standard normal random variables Z.

Output: Gaussian vectors $X^\epsilon = A_\epsilon Z + \mu_\epsilon$.

3.2.3 Spectral method

Next we present an efficient simulation method for band-limited processes, that is, processes whose spectral measure is compactly supported. Let $r(h)$ be such kernel and $\rho(\lambda)$ be the spectral measure supported on $[-D, D]$, then the kernel can be reconstructed from its spectrum using Fourier transform. That is:

$$r(h) = \int_{-D}^{D} e^{-i\lambda h} d\rho(\lambda), \tag{14}$$

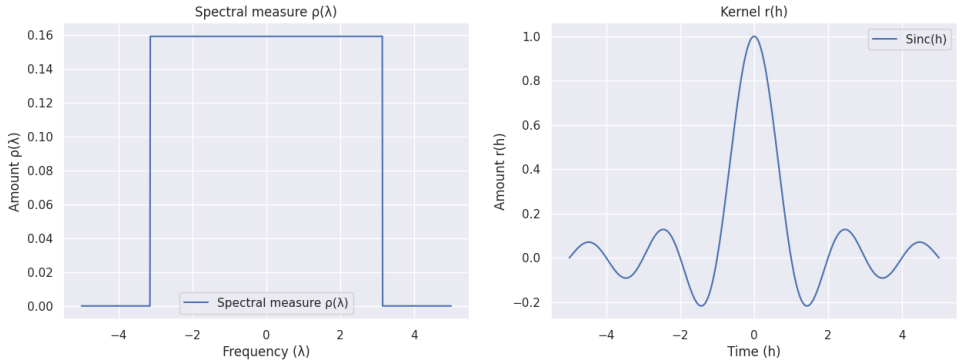


Figure 2: The SINC kernel alongside its spectral measure. Simulating this process using the kernel method (Algorithm 4) is rather slow and inaccurate due to sign-changes and slow decay of the covariance. In contrast, applying the spectral method (Algorithm 5) is simple and efficient, as the spectral measure is simply a constant over its support.

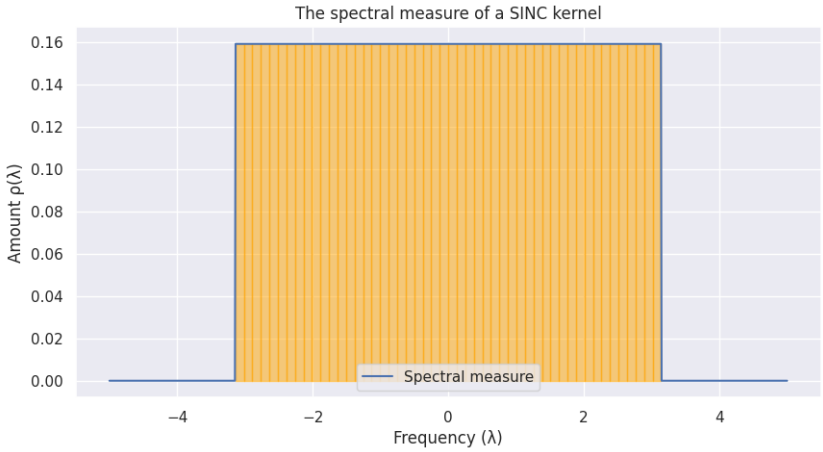
Following the approximation method introduced by [8, 10], we partition $[-D, D]$ into $2n$ sub-intervals denoted $\{I_{\pm j}\}_{j=1}^{n}$. Then we write

$$X_\rho(t) \overset{d}{=} \sum_{i=1}^{n} \sqrt{\rho(I_j \cup I_{-j})}\,(\alpha_j C_j(t) \oplus \beta_j S_j(t)) \oplus R_n(t),$$

where $\{\alpha_j\}_{j=1}^n$ and $\{\beta_j\}_{j=1}^n$ are i.i.d $\mathcal{N}(0,1)$-distributed random variables,

$$C_j(t) = \frac{1}{\rho(I_j)} \int_{I_j} \cos(\lambda t) d\rho(\lambda), \quad S_j(t) = \frac{1}{\rho(I_j)} \int_{I_j} \sin(\lambda t) d\rho(\lambda),$$

and $R_n(t)$ is a Gaussian process independent of $\{\alpha_j, \beta_j\}$ whose maximal variance on $[0, T]$ tends to zero, as n tends to infinity.



This figure shows a possible partition of the spectral measure. In this case, $\rho(I_n)$ is simply a normalised interval length.

The number of partitions may be selected using the anti-aliasing theorem by Shannon and Nyquist or by the variance considerations in [8, 10]. This process may prove efficient for kernels which are not band-limited but are further constraints may apply. The spectral method has two main advantages over the covariance kernel method:

1. **Coverage of new processes:** It is common that a covariance kernel $r(h)$ induces a numerically singular covariance matrix. When that happens, pseudo-methods are used for the Cholesky decomposition which highly affect the amplitude of process. The spectral method is unaffected by this problem, and may therefore give good results where the covariance method fails. See Figure 2 for a classical example.

2. **Efficiency:** This method simulates m paths in $O(n \times m \times T)$ time, which is a big improvement over the preceding method's $O(n^3)$.

3. **Analysis:** This method proposes a dimension-reduced view of the Gaussian process. With this view, one may analyze the behavior of the process with respect to the $2n$ i.i.d standard normals instead of the $d >> 2n$ standard normals in the preceeding method.

We summarise the spectral method in Algorithm 5 below. In the Python implementation we demonstrate simulations from two kernels, the SINC and the squared SINC kernels.

Algorithm 5 Spectral sampling for Gaussian processes (Python)

Input: Stationary kernel R, sampling rate ϵ, interval length T, approximation rate Δ

1. Calculate the Fourier transform $\rho(\lambda) = \int_{-\infty}^{\infty} e^{-i\lambda t} dR(t)$.
2. Create the equidistant partition T_ϵ of $[0, T]$ with sampling rate ϵ.
3. Partition the domain of ρ to n sub-intervals each denoted I_n using Δ and set $\lambda_n \in I_n$
4. Generate $2n$ i.i.d standard normals $\{\alpha_i\}, \{\beta_i\}$.
5. for each $t \in T_\epsilon$:
   (a) $X(t) \approx \sum_{i=1}^{n} \sqrt{\rho(I_i)} \cdot (\alpha_i cos(\lambda_i t) + \beta_i \sin(\lambda_i t))$

Output: Gaussian vector X.

3.3 Justifying the discretization

Recall that when dealing with continuous time processes $[0, T]$ our goal is to estimate the continuous probability

$$B_X(T, \ell) = \mathbb{P}\left(X_t \in [-\ell, \ell], \forall t \in [0, T]\right).$$

Since no computer can simulate continuous points we have to use a discretization scheme and estimate

$$\overline{B_X(T, \ell)} = \mathbb{P}\left(X_t \in [-\ell, \ell], \forall t \in \{0, \epsilon, 2\epsilon, ..., T\}\right).$$

3.3.1 Partition into large intervals

To justify using this approach, we will present upper and lower bounds for our target probability. Since $\overline{B_X(T, \ell)}$ is discrete, it is bound to lose some information about variates which were not included and is therefore an upper bound to $B_X(T, \ell)$ That is:

$$B_X(T, \ell) \leq \overline{B_X(T, \ell)}.$$

A lower bound for this probability may be drawn from Lemma 4, we begin by splitting our domain $[0, T]$ to k parts and observe that

$$\mathbb{P}\left(\sup_{[0, A_1 + ... + A_{kA}]} |f_t| \leq \lambda\right) \geq \mathbb{P}\left(\sup_{[0, A_1]} |f_t| \leq \lambda\right) \cdot \ldots \cdot \mathbb{P}\left(\sup_{[(k-1)A, kA]} |f_t| \leq \lambda\right).$$

Since the process is stationary we get that:

$$B_X(T, \ell) \geq B_X(A_1, \ell)^k$$

We set A to be small and use an efficient dense sampler on it to assure that our estimator is tight. We now assume that we have:

$$\overline{B_X(A_1,\ell)}^k \leq B_X(T,\ell) \leq \overline{B_X(kA,\ell)}.$$

3.3.2 Direct discretization of the original interval

A different approach is to use discretization of the whole interval $[0,T]$. The following theorem from [8] justifies the use of this discretization scheme to calculate the exponent:

Theorem 10. *Let $\ell \in \mathbb{R}$ and $\rho \in \mathcal{L}$. For the spectral measure ρ define the ϵ-sampled exponents as:*

$$\gamma^{\ell}_{\rho,\epsilon} = -\lim_{T\to\infty} \log \mathbb{P}\left(\inf_{n\in\mathbb{Z}, n\epsilon\in[0,T]} X_{n\epsilon} > \ell\right),$$

$$\theta^{\ell}_{\rho,\epsilon} = -\lim_{T\to\infty} \log \mathbb{P}\left(\sup_{n\in\mathbb{Z}, n\epsilon\in[0,T]} |X_{n\epsilon}| \leq \ell\right).$$

Whenever these limits exists we have that:

$$\lim_{\epsilon\to 0} \gamma^{\ell}_{\rho,\epsilon} = \gamma^{\ell}_{\rho}.$$

$$\lim_{\epsilon\to 0} \theta^{\ell}_{\rho,\epsilon} = \theta^{\ell}_{\rho}.$$

3.4 Approximation methods

We now explore and analyze several approximation methods for the ball coefficient of a general Stationary Gaussian process. We will compare our results with those we have calculated analytically. Our goal is to find an efficient estimator $\overline{B_K(T,\ell)}$ of

$$B_K(T,\ell) = \mathbb{P}\left(X_1 \in [-\ell,\ell], X_2 \in [-\ell,\ell], ..., X_T \in [-\ell,\ell]\right).$$

We will use it to create an estimator of $\theta_K(\ell)$ by calculating

$$\overline{\theta_K(\ell)} = -\frac{1}{T}\overline{B_K(T,\ell)},$$

for growing times T and wish to observe the convergence of

$$\overline{\theta_K(\ell)} \to \ \theta^{\ell}_X.$$

3.4.1 Monte Carlo method

We begin with the Monte Carlo method to estimate probabilities as described in (2.3.1), this method assumes the availability of a sampling mechanism and proposes the estimator which counts the fraction of samples which fall inside the ball:

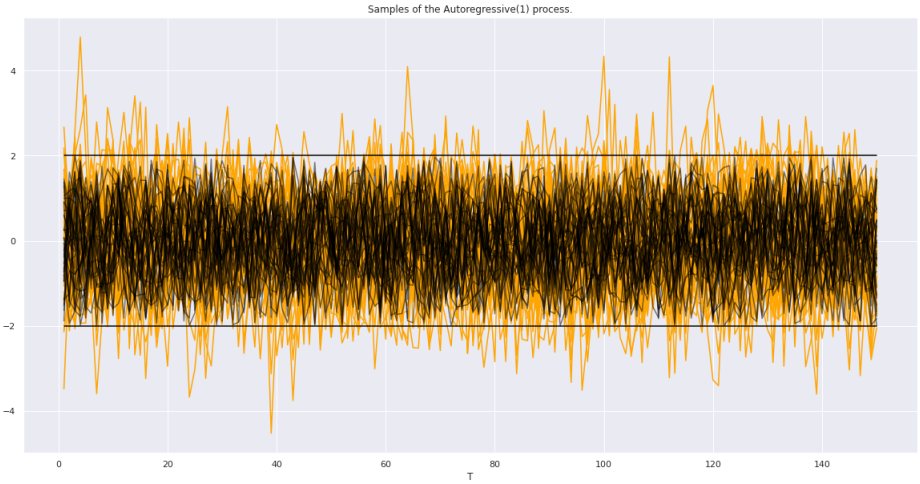
$$\overline{B_K(T,\ell)} = \frac{|\{x \in [-\ell,\ell]\}|}{m}. \tag{15}$$

The variance of this estimator is then given by:

$$\mathrm{Var}(\overline{B_K(T,\ell)}) = \frac{B_K(T,\ell)\,(1 - B_K(T,\ell))}{m}.$$

A visualization of this method is given below, m samples of an $AR(1)$ process were sampled, the black ones fall inside the ball and represent the fraction we are looking for.

Figure 3: Paths of the Autoregressive process



This figure depicts samples of the AR(1) process against the ball $D = [-2, 2]$ on the interval $[0, 150]$. The straight black lines present the ball, the orange samples do not stay inside the ball and the black ones do. The fraction of those is the Monte Carlo estimator.

To test this method, we set the autoregressive kernel

$$K(h) = \frac{0.5^{|h|}}{0.75}$$

and the ball coefficient to be twice the variance of each element, that is, $\ell = 2\frac{2}{3}$. While the results appearing in Figure 4 are impressive, three concerns are raised.

1. In order to use this estimator we must be able to create samples which agree with the event $x \in D$. This event can become to rare for even a

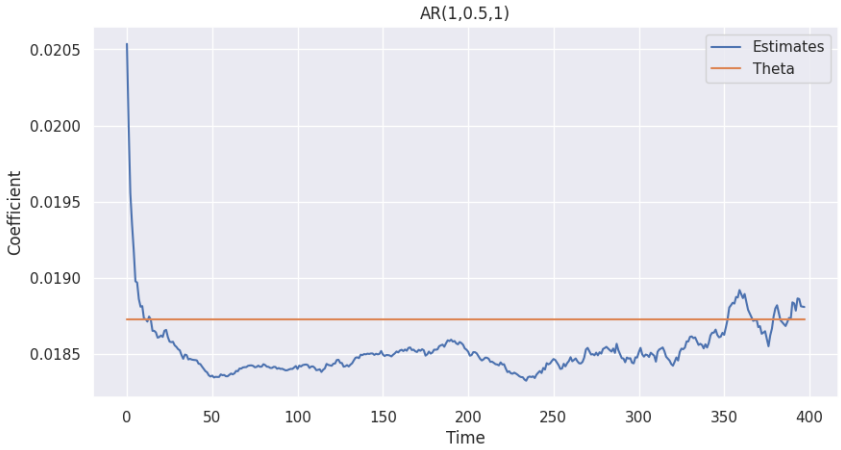
single sample to appear. In this case, this estimator fails terribly and always returns 0.

2. The time of convergence n_0 may happen at a very late time, even if the event is not rare, the memory complexity might make this method inefficient.

3. While its true that the estimator $\overline{B_K(T,\ell)}$ is unbiased, the same is not true for the estimator $\theta_K(T,\ell)$. Using Jensen's inequality we get that:

$$\mathbb{E}\left[-\frac{1}{T}\log\overline{B_K}\right] \leq -\frac{1}{T}\log(B_K).$$

Figure 4: Convergence of the estimates.



Convergence of the estimated decay coefficient to the real coefficient. It can be visually seen that this phenomena appears sometime around $t = 350$.

3.4.2 Monte Carlo integration.

Our goal in this section is to avoid the need to sample from the original distribution, thus ignoring the problem of rare events as described in the previous section. To do that, we use the same method as described in (2.2.2) and treat the integral induced by the probability directly. That is, We wish to estimate

$$B_K(T,\ell) = \int_{[-\ell,\ell]} \dots \int_{[-\ell,\ell]} f_X dx,$$

by sampling from the uniform distribution and calculating:

$$\overline{B_K(T,\ell)} \approx Vol([-\ell,\ell]^T) \cdot \frac{1}{m}\sum_{i=1}^{m} f_X(u_i).$$

its variance is now

$$\mathrm{Var}\left(\overline{B_K(T,\ell)}\right) = \frac{1}{m}\left(\mathrm{Vol}([-\ell,\ell]^T)\int_{[-\ell,\ell]^T} f(x)^2\,dx - \left(\int_{[-\ell,\ell]^T} f(x)\,dx\right)^2\right)$$

This estimator is a complement to the regular Monte Carlo method. A large hypercube will cause it to fail because it will have high volume and thus high variance, a small hypercube, where the other method fails will cause it to succeed with low variance. The following table shows the failure and success of this estimator on the OU kernel and two different hyper-cubes when trying to calculate the decay coefficient.

	$\ell = 3$	$\ell = 0.5$
θ	0.0024204300922572967	0.7741478154827315
θ_{est}	0.9472160764143754	0.7746949430343235

Estimation of the decay rate fails for large values of ℓ but shows significant success where the regular MC method fails miserably.

3.4.3 Importance Sampling

In the previous section we saw that an efficient estimator exists for rare events, the problem is that this estimator is highly affected by the variance of the uniform density, that is, the volume of the set. In order to control the variance, the Importance sampling method proposes to calculate

$$B_K(T,\ell) = p = \mathbb{P}\left(X_1 \in [-\ell,\ell], ..., X_T \in [-\ell,\ell]\right) = \int_{[-\ell,\ell]^T}\cdots\int dF_X$$

by a change of measure. That is, defining an alternative measure f_Y supported on $\mathcal{Y} \supseteq [-\ell,\ell]^T$ and using the equation:

$$p = \int_{\mathcal{Y}}\cdots\int \mathbb{I}(Y \in [-\ell,\ell]^T)\frac{f_X}{f_Y}dF_Y.$$

The importance estimator becomes:

$$\overline{B_K(T,\ell)} = \frac{1}{m}\sum_{i=1}^{m}\mathbb{I}(y_i \in [-\ell,\ell]^T)\cdot\frac{f_X}{f_Y}(y_i).$$

The variance of the estimator is:

$$\mathrm{Var}\left(\overline{B_K(T,\ell)}\right) = \frac{1}{m}Var_Y\left(\mathbb{I}(Y \in D)\frac{f_X}{f_Y}\right) = \frac{1}{m}\left(\mathbb{E}_Y\left[\left(\mathbb{I}(Y \in D)\frac{f_X}{f_Y}\right)^2\right] - p^2\right).$$

Choosing the correct alternative density is the art of importance sampling. The following are some rules which may help.

1. Ease of sampling - the proposal density must be easy to sample from.
2. Ease of weight calculation - The likelihood ratio $\left(\frac{f_X}{f_Y}\right)$ should be easy to calculate.
3. Proportionality - The proposal density should be proportional to the target density.

3.4.4 Cross entropy method

In this section we focus on selecting a good alternative distribution f_Y for the importance estimator. Since our estimator is uniformly unbiased we are looking to solve:

$$f_Y = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \left[\operatorname{Var}(\overline{B_K(T, \ell)}) \right],$$

where

$$\operatorname{Var}(T) = \frac{1}{m} Var_Y \left(\mathbb{I}(Y \in D) \frac{f_X}{f_Y} \right) = \frac{1}{m} \left(\mathbb{E}_Y \left[\left(\mathbb{I}(Y \in D) \frac{f_X}{f_Y} \right)^2 \right] - p^2 \right).$$

While this program is not as easy one, the optimal solution presents itself unto the viewer when looking at the definition of the estimator. Recall that

$$T = \frac{1}{m} \sum_{i=1}^{m} \mathbb{I}(y_i \in D) \frac{f_X}{f_Y}(y_i).$$

if we could choose the distribution to be $f_{opt} = \frac{\mathbb{I}(Y \in D) \cdot f_X}{p}$ we would get that

$$T = \frac{1}{m} \sum_{i=1}^{m} \cancel{\mathbb{I}(y_i \in D) f_X(y_i)} \cdot \frac{1}{\cancel{\frac{\mathbb{I}(y_i \in D) f_X(y_i)}{p}}} = p.$$

The Cross Entropy method is an iterative process that approximates f_{opt} by finding increasingly better candidate-distributions with respect to the Kullback-Leibler divergence. Formally, recall that

$$D_{KL}(f_{opt}, f) = \mathbb{E}_{f_{opt}} \left[\log \frac{f_{opt}}{f} \right] = \mathbb{E}_{f_{opt}} \log f_{opt} - \mathbb{E}_{f_{opt}} \log f,$$

and define the minimisation problem as:

$$f_Y = \underset{f \in \mathcal{F}}{\operatorname{argmin}} [D_{KL}(f_{opt}, f)] = \underset{f \in \mathcal{F}}{\operatorname{argmax}} [\mathbb{E}_{f_{opt}} \log f] \approx \underset{f \in \mathcal{F}}{\operatorname{argmax}} \frac{1}{m} \sum_{y_i \sim f_{opt}} \log f(y_i).$$

Algorithm 6 Iterative Cross-Entropy Method for Importance Sampling

Input: Random variable X, Set D, Prior $\mathcal{F}$

1. Initialise proposal distribution f_Y within the class $\mathcal{F}$.
2. **For** $t = 1, 2, \ldots, T$ **do**
   (a) Sample $S = \{x_i\}_{i=1}^m$ from current proposal distribution f_Y.
   (b) Calculate the likelihood ratio $w_i = \frac{f_X}{f_Y}(x_i)$.
   (c) Calculate the fitness function based on w_i or other criteria.
   (d) Select elite samples based on the fitness function.
   (e) Solve the stochastic program to fit a new f_Y to the elite samples:
      i. $f_Y = \underset{f \in \mathcal{F}}{\operatorname{argmax}} \frac{1}{|S|} \sum_{x \in S} \log f(x)$.
3. Calculate the indicator function $\gamma_i = \mathbb{I}(x_i \in D)$.
4. Calculate the estimator $T = \frac{1}{m} \sum_{i=1}^m \gamma_i w_i$.

Output: Final proposal distribution f_Y and estimator $T \approx P(X \in D)$.

We extend over the known literature by proposing two new methods which include the approximation of the truncated multivariate normal with the uniform distribution and a method to sample directly from the optimal distribution.

3.4.5 Sampling from the optimal density.

The current literature proposes the Metropolis-Hasting algorithm, or other Markov-Chain-Monte-Carlo methods to sample from densities of which the normalization constant p is unknown for. These methods do not work well when the area of acceptance is small and are not efficient in our context. We extend over the known algorithms by proposing an efficient algorithm which can sample from any multivariate normal density. We begin by proposing a method to sample directly from the optimal distribution

$$f_{opt} = \frac{\mathbb{I}(x \in D) f_X(x)}{p}.$$

To do that, we use the fact that conditioning the Gaussian process on specific values of coordinates or a linear combination of them is a multivariate Gaussian vector. That is, the variable X_t is normally distributed with conditional mean:

$$\mu_{X_t|X_{1:t-1}} = \mu_{X_t} + \Sigma_{X_t, X_{1:t-1}} \Sigma^{-1}_{X_{1:t-1}, X_{1:t-1}} (X_{1:t-1} - \mu_{X_{1:t-1}})$$

and the conditional variance:

$$\sigma^2_{X_t|X_{1:t-1}} = \Sigma_{X_t, X_t} - \Sigma_{X_t, X_{1:t-1}} \Sigma^{-1}_{X_{1:t-1}, X_{1:t-1}} \Sigma_{X_{1:t-1}, X_t}$$

Our algorithm samples each path using its conditional distribution, it can sample any truncated Gaussian trajectory efficiently. Algorithm 7 uses a non optimised approach but by using specialised matrix inversion and multiplication methods for symmetric Toeplitz matrices, and pre-calculating several elements, we can sample a process in $O(T^2 \log(T))$.

Algorithm 7 Sampling from the TMVN(Python)

Input: Gaussian random variable X, Set $[a, b]$

1. Initialise empty trajectory vector $\mathbf{x}$.
2. **For** $t = 1, 2, \ldots, T$ **do**
   (a) **If** $t = 1$ **then**
      i. Sample x_1 from truncated normal with bounds $[a, b]$, mean μ_{X_1}, and variance Σ_{X_1, X_1}.
   (b) **Else**
      i. Compute conditional mean and variance:
         A. $\mu_{X_t|X_{1:t-1}} = \mu_{X_t} + \Sigma_{X_t, X_{1:t-1}} \Sigma^{-1}_{X_{1:t-1}, X_{1:t-1}} (X_{1:t-1} - \mu_{X_{1:t-1}})$.
         B. $\sigma^2_{X_t|X_{1:t-1}} = \Sigma_{X_t, X_t} - \Sigma_{X_t, X_{1:t-1}} \Sigma^{-1}_{X_{1:t-1}, X_{1:t-1}} \Sigma_{X_{1:t-1}, X_t}$.
      ii. Sample x_t from truncated normal with bounds $[a, b]$, mean $\mu_{X_t|X_{1:t-1}}$, and variance $\sigma^2_{X_t|X_{1:t-1}}$.
   (c) Append x_t to trajectory vector $\mathbf{x}$.
3. Output the trajectory $\mathbf{x}$.

Now that we have efficiently gathered samples from the optimal distribution we can remove the iterative process from the Cross-Entropy method and directly solve the optimization problem which leads to an efficient estimator.

Algorithm 8 Cross Entropy importance sampler (Python)

Input: Gaussian random variable X, truncation $[a, b]$,Prior $\mathcal{F}$

1. Set $f_{opt} = TMVN(\mu_X, \Sigma_X, a, b)$
2. Sample $S = \{x_i\}_{i=1}^m$ from f_{opt}.
3. Solve the stochastic program $f_Y = \underset{f \in \mathcal{F}}{\operatorname{argmax}} \frac{1}{|S|} \sum_{x \in S} \log f(x)$.
4. Calculate the likelihood ratio $w_i = \frac{f_X}{f_Y}(y_i)$.
5. Calculate the estimator $T = \dfrac{1}{m} \sum_{i=1}^m w_i$.

Output: estimator $T \approx P(X \in D)$.

The success of this estimator now solely depends on the complexity of the prior $\mathcal{F}$ we have selected. A different approach using these samples can be to approximate f_{opt} by introducing neural methods.

3.4.6 Approximating the TMVN

In this section we propose and analyze a method to directly calculate the ball probability by approximating the truncated multivariate PDF. Recall that f_{opt} is not generally available to us, but assume that there exists a single $y_j \in D$ such that $f_{opt}(y_j)$ is evaluated and we have:

$$c = f_{opt}(y_j) = \frac{\mathbb{I}(y_j \in D) f_X(y_j)}{p}.$$

Then a direct solution would be:

$$p = \frac{\mathbb{I}(y_j \in D) f_X(y_j)}{c} = \frac{f_X}{f_{opt}}(y_j).$$

If we could find any $y \in [-\ell, \ell]^n$ and any PDF f_Y such that $\ell \to 0 \implies d(f_Y(y), f_{opt}(y)) \to 0$ then we could approximate

$$p \approx \frac{f_X}{f_Y}(y).$$

We hypothesise that the uniform density f_U could be a fair approximate to the truncated normal distribution, and propose the estimator

$$p \approx \frac{f_X}{f_U}(y).$$

In the one-dimensional case this phenomena can be easily observed visually by plotting the two distributions on the same support 5.

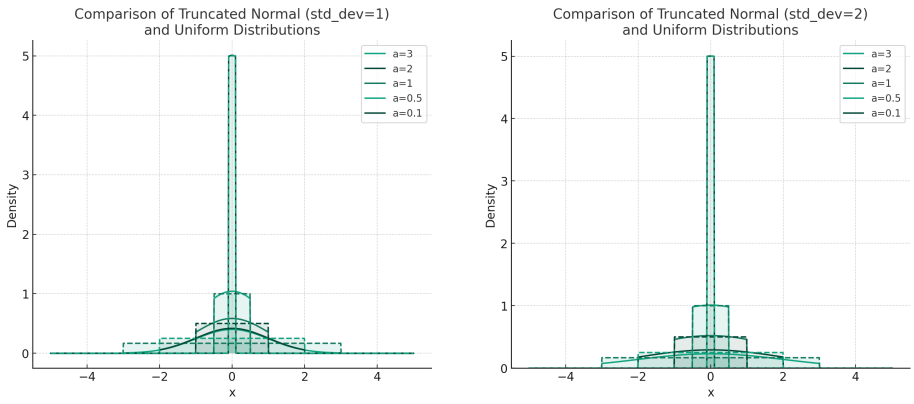


Figure 5: Comparison of truncated normal distribution (solid lines) and uniform distribution (dashed lines) for different values of a. The truncated normal distribution with a larger standard deviation appears flatter and more similar to the uniform distribution within the truncation bounds.

Another empirical evidence of this phenomena is that several metrics commonly used to estimate the similarity between distributions vanish when ℓ decreases 6.

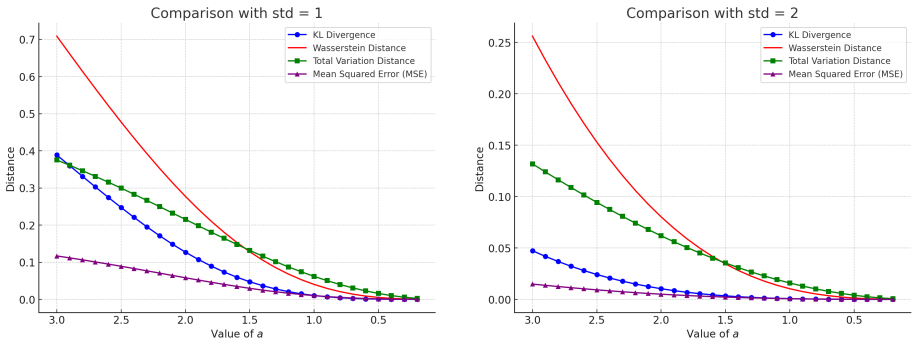


Figure 6: Comparison of the four different measures, KL, TD, Wasserstein and MSE on the centred one dimensional normal distributions with varying std.

To further increase our confidence of this hypothesis, we tried to understand how the ratio between these distributions behave in the limit, we got that for a normal centred Gaussian with variance σ^2:

$$\lim_{a \to 0} \frac{f_{opt}}{f_U} = e^{-\frac{x^2}{2\sigma^2}}.$$

From this behaviour we get that when $x = 0$

$$\sqrt{2\pi} \cdot p(\ell) \approx 2a.$$

In the n-dimensional case the results follow the same pattern, the plot below shows the two dimensional Gaussian becomes increasingly like the uniform distribution when the hyper-rectangle becomes smaller.

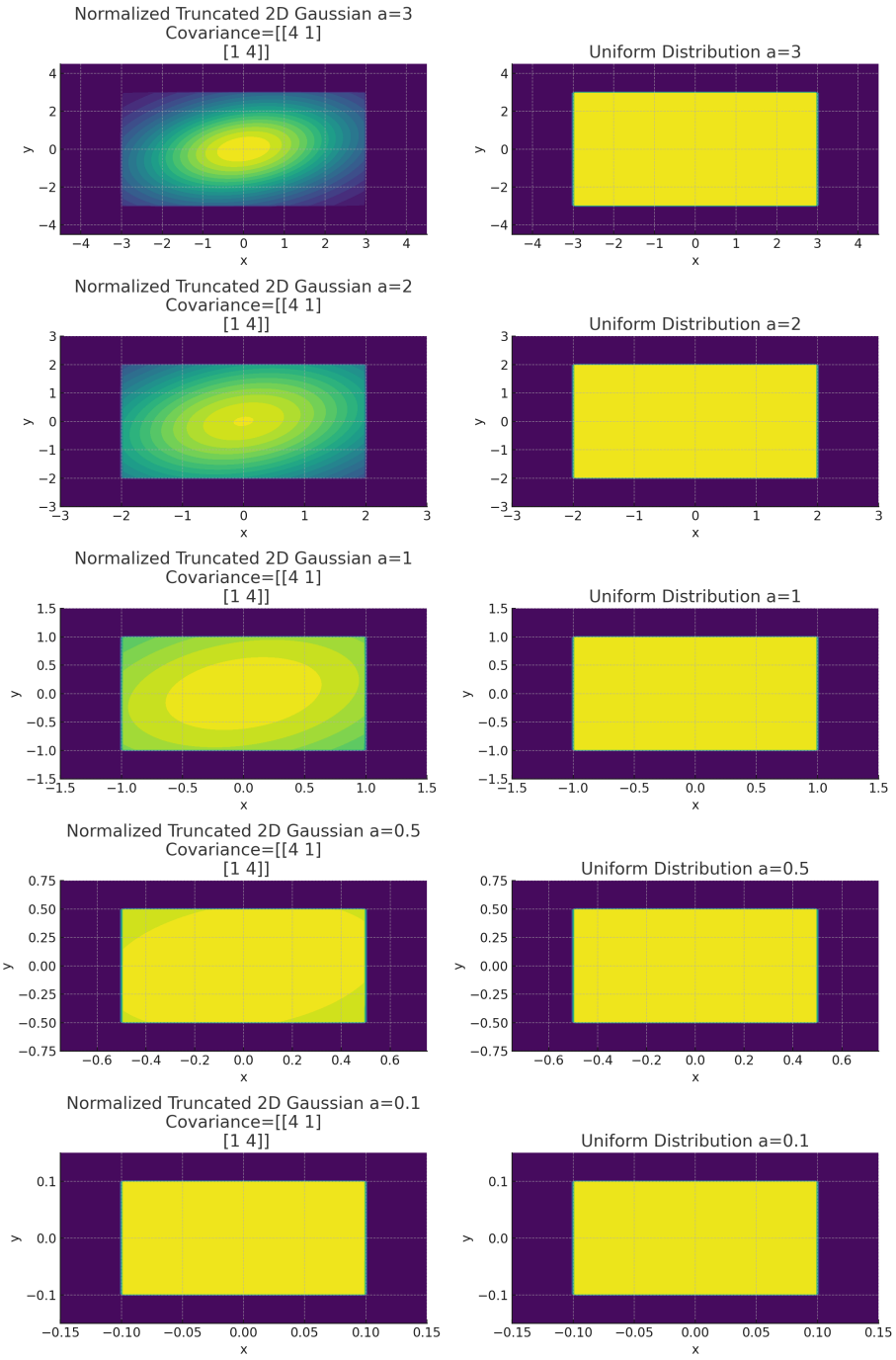


Figure 7: Contour plots for the normalised truncated 2D Gaussian distribution (left column) and uniform distribution (right column) for different values of ℓ. The covariance matrix introduces correlation between the two variables, affecting the shape of the truncated 2D Gaussian distribution.

We conclude that a good estimator for $p(\ell)$ when $\ell \to 0$ is a Riemann-like approximation to the integral at the center. That is:

$$p(\ell) \approx \frac{f_X}{f_U}(0) = f_X(0) \cdot Vol([-\ell, \ell]^d).$$

The approximation of the Truncated Multivariate Normal (TMVN) density is a critical task for efficient Importance Sampling. Traditional methods have often relied on Gaussian Mixture Models to approximate this density. However, we propose a forward-looking approach that utilizes advanced neural methodologies, specifically Neural Mixture Models and Normalizing Flows. These neural-based techniques offer the potential to capture the high-dimensional, complex nature of the TMVN distribution more accurately. Importantly, this initiative serves as a foundational step in bridging the rigorous mathematical framework of Gaussian processes with the state-of-the-art in neural network technology. Although investigating the full extent of these neural methods for TMVN approximation is beyond the scope of this thesis, it presents an exciting avenue for future research that could enrich both the theory and practice of statistical learning.

3.4.7 Separation of variables method

We focus on a variation of the SOV concept proposed by Rosenblatt, Geweke and Genz to calculate the multidimensional integral induced by the ball probability. That is, we wish to transform the integral

$$\begin{aligned} p = B_X(T, \ell) &= \mathbb{P}\left[X_t \in [-\ell, \ell), \forall t \in \{1, 2, ..., T\}\right] \\ &= \frac{1}{\sqrt{|\Sigma|(2\pi)^T}} \int_{-\ell}^{-\ell} \cdots \int_{-\ell}^{-\ell} e^{-\frac{1}{2}x^t \Sigma^{-1} x} dx, \end{aligned}$$

into to a form that is more efficient for numerical methods like the Monte Carlo integrator. As with many others, this method begins with the Cholesky algorithm to calculate $A : AA^T = \Sigma$ and the understanding that $X = AY$ where $Y \sim N(0, I)$. The following transformations are then used.

1. Change of measure - transform the integral to one over the independent measure.

$$x^t \Sigma^{-1} x = \left(y^t A^t\right)\left(A^{-t} A^{-1}\right)(Ay) = y^t y, dx = |\Sigma|^{\frac{1}{2}} dy.$$

2. Change of integration bounds - knowing that Σ is positive definite we can use A^{-1} and say that

$$\forall n : -\ell \leq x_n \leq \ell \implies a_n = \frac{-\ell - \sum_{i=1}^{n-1} A_{n,i} \cdot y_i}{A_{n,n}} \leq y_n \leq \frac{\ell - \sum_{i=1}^{n-1} A_{n,i} \cdot y_i}{A_{n,n}}$$

3. Change of measure - transform the integral to one over the uniform measure by using the inverse transform theorem and setting $y_i = \Phi^{-1}(z_i)$, at this point the integration limits become

$$l_i = \Phi(\frac{-\ell - \sum_{i=1}^{n-1} A_{n,i} \cdot \Phi^{-1}(z_i)}{A_{n,n}}), u_i = \Phi(\frac{\ell - \sum_{i=1}^{n-1} A_{n,i} \cdot \Phi^{-1}(z_i)}{A_{n,n}}).$$

4. Apply a min-max transform - use new uniforms denoted by w_i and set $z_i = \frac{w_i - l_i}{z_i - l_i}$ to transform the integral to

$$(u_1 - l_1) \int_0^1 (u_2 - l_2) \cdots \int_0^1 (u_T - l_T) dw.$$

The integral is now in a form that is efficient for numerical algorithms like Monte Carlo. This algorithm is the current state of art and is used in many software. It is the to go algorithm for the scipy Python package at scipy.stats.mvn.

Algorithm 9 Genz & Geweke MVN algorithm (Python)

Input: Covariance matrix Σ, lower bound ℓ, maximum iterations N_{max}.

1. Calculate the cholesky decomposition A of Σ.
2. Set $res = 0, N = 0, d_1 = \Phi(\ell / C_{1,1}), f_1 = 1 - d_1$.
3. While $N < N_{max}$:
   (a) Sample $\{w_i\}_{i=1}^m$ from the uniform distribution.
   (b) For $i \in \{2, 3, ..., m\}$:
      - Set $y_i = \Phi^{-1}(d_{i-1} + w_{i-1}(1 - d_{i-1}))$.
      - Set $d_i = \Phi(b_i)$ and $f_i = (1 - d_i) f_{i-1}$.
   (c) Set $N = N + 1, res = res + \frac{f_m - res}{N}$.

Output: estimator $p \approx res$.

3.4.8 Results

We ran several of the methods described above on the AR(1) and OU kernels, the ball coefficients are their respective point variances multiplied by various values. The results approve our hypothesis that for small hypercubes the riemann-like approximation is competitive with the state-of-art Genz and Monte-Carlo methods.

Radius	Real	MCE	MCI	Genz	Uniform
$2\sigma^2$	0.009488	0.008989	inf	0.008945	-1.028006
$1\sigma^2$	0.150837	inf	0.226391	0.145468	-0.334859
$0.5\sigma^2$	0.523902	inf	0.523345	0.522966	0.358288
$0.25\sigma^2$	1.096386	inf	1.097243	1.097246	1.051435
$0.1\sigma^2$	1.974198	inf	1.975299	1.975297	1.967726

4 Conclusions

This thesis focused on the estimation of the ball exponents

$$\gamma_X^\ell = \lim_{T\to\infty} -\frac{1}{T}\log P_X(T,\ell), \tag{16}$$

but most of its methods could be used to also estimate the persistence exponent:

$$\theta_X^\ell = \lim_{T\to\infty} -\frac{1}{T}\log B_X(T,\ell) \tag{17}$$

for stationary Gaussian processes. We have reviewed several sampling and estimation algorithms and showed their strengths and weaknesses. We tested the algorithms on the IID, AR(1) and OU kernels whose exponent we managed to calculate and visualized many different results. We conclude that the MCI, Genz and our Uniform-approximator are competitive and each have their own advantages and disadvantages.

4.1 Research directions

A major contributions of this thesis is the establishment of accessible software for researchers to simulate and estimate rare events for stationary Gaussian processes. It is of considerable interest to extend these tools to different kernels and setups, and use them to understand the conditional behaviour of the process under some rare event. We detail some of these directions below.

4.1.1 Graphical models

An active area of research could be the search of kernel functions which induce interesting structures on the precision matrix of the induced Gaussian vector. These kernels would allow for the efficient calculation of not only our exponents but many other hard to calculate probabilities.

4.1.2 Compact kernels

A compact kernel created a double-banded Toeplitz covariance matrix, these matrices have specialized algorithms which help greatly with their inversion and decomposition so algorithms like Genz's and even Cholesky's can benefit greatly.

4.1.3 Prior structure

If a prior is known about the a family of kernels, then specialized methods can be used. For example, the SINC kernel induces I.I.D variables when restricted to $\mathbb{Z}$. Knowing this, we can propose a bound on continuous probabilities in the same range. A research of the special properties of Kernel functions is required and will help these calculations greatly.

4.1.4 Spectral method

Our subsection 3.2.3 on the simulation of stationary Gaussian processes using the spectral method proposes that the process X is well-approximated by a a simple random series determined by the set $Z = \{\alpha_i, \beta_i\}_{i=1}^n$ of independent and identically distributed random variables. We propose either using the Monte Carlo estimators on these variables or to use machine learning methods to learn the features of Z which cause the ball or persistence events to occur.

4.1.5 Quasi Monte Carlo integration

In cases where the uniform distribution is chosen as the alternative density of the importance sampler, one can then use a Quasi-random sequence such as the Halton sequence to estimate the integral with better convergence rate. Quasi methods will usually converge in $O(1/m)$ instead of $O(m^{-0.5})$. More research in this area may increase the efficiency of such estimators.

4.1.6 Different domains of interest

It is possible to apply these methods to different areas of interest, to our knowledge, an interesting question is the varying ball problem, where ℓ is a function of t and may increase or decreases as time flows. Some of our methods are applicable even in that case and a direct calculation could still be made if the process has a nice precision structure.

Persistence probability To point through the direction we believe is possible for research, we shortly cover an extension of the Genz algorithm for unbounded probabilities. Recall that the Persistence probability of a discrete Gaussian process is:

$$\begin{aligned} p = P_X(T, \ell) &= \mathbb{P}\left[X_t \in [\ell, \infty), \forall t \in \{1, 2, ..., T\}\right] \\ &= \frac{1}{\sqrt{|\Sigma|(2\pi)^T}} \int_\ell^\infty \cdots \int_\ell^\infty e^{-\frac{1}{2}x^t \Sigma^{-1} x} dx, \end{aligned}$$

where $x = (x_1, x_2, ..., x_T)$ and Σ is the $T \times T$ covariance matrix induced by the kernel of the process. While the infinite integration limits of P_X causes most numerical algorithms to fail, a series of relatively straightforward transformations [proposed by Rosenblatt, Geweke and Genz] can be used in order to change the form of the integral into one that allows efficient computation with the Monte Carlo method. Recall that $x = Ay$ and by using the change of variable method we get that

$$x^t \Sigma^{-1} x = \left(y^t A^t\right)\left(A^{-t} A^{-1}\right)(Ay) = y^t y, dx = |\Sigma|^{\frac{1}{2}} dy.$$

Changing the variables requires that we change the integration limits, that is:

$$\ell \leq x < \infty \implies \forall n : a_n = \frac{\ell - \sum_{i=1}^{n-1} A_{n,i} \cdot y_i}{A_{n,n}} \leq y_n < \infty,$$

and the integral becomes:

$$\int_{a_1}^{\infty} \phi(y_1) \int_{a_2}^{\infty} \phi(y_2) \cdots \int_{a_T}^{\infty} \phi(y_T) dy.$$

Letting Φ be the standard normal cdf and setting $y_i = \Phi^{-1}(z_i)$, then:

$$\ell \leq x < \infty \implies \forall n : b_n = \Phi\left(\frac{\ell - \sum_{i=1}^{n-1} A_{n,i} \cdot \Phi^{-1}(y_i)}{A_{n,n}} \right) \leq z_n < 1.$$

and the integral becomes:

$$\int_{b_1}^{1} \int_{b_2}^{1} \phi \cdots \int_{b_T}^{1} dz.$$

The last transformation is the min-max transform moving z_i to $[0, 1]$ while setting $w_i = \frac{z_i - b_n}{1 - b_n}$, the integral becomes:

$$(1 - b_1) \int_0^1 (1 - b_2) \cdots \int_0^1 (1 - b_T) dw.$$

The integral is now in a convenient form and the following Monte Carlo algorithm estimates p.

Algorithm 10 Genz & Geweke MVN algorithm (Python)

Input: Covariance matrix Σ, lower bound ℓ, maximum iterations N_{max}.

1. Calculate the cholesky decomposition A of Σ.
2. Set $res = 0, N = 0, d_1 = \Phi(\ell / C_{1,1}), f_1 = 1 - d_1$.
3. While $N < N_{max}$:
   (a) Sample $\{w_i\}_{i=1}^{m}$ from the uniform distribution.
   (b) For $i \in \{2, 3, ..., m\}$:
      - Set $y_i = \Phi^{-1}\left(d_{i-1} + w_{i-1}(1 - d_{i-1})\right)$.
      - Set $d_i = \Phi\left(b_i\right)$ and $f_i = (1 - d_i) f_{i-1}$.
   (c) Set $N = N + 1, res = res + \frac{f_m - res}{N}$.

Output: estimator $p \approx res$.

References

[1] Frank Aurzada and Marvin Kettner. Persistence exponents via perturbation theory: Ar(1)-processes. *Journal of Statistical Physics*, 177:651–665, 2019.

[2] Frank Aurzada, Sumit Mukherjee, and Ofer Zeitouni. Persistence exponents in markov chains. *Annales de l'Institut Henri Poincaré, Probabilités et Statistiques*, 57(3):1411 – 1441, 2021.

[3] S. Bochner. Monotone funktionen, stieltjessche integrale und harmonische analyse. *Mathematische Annalen*, 108:378–410, 12 1933.

[4] F.P. Cantelli. Sulla determinazione empirica delle leggi di probabilità. *Giorn. Ist. Ital. Attuari (in Italian)*, 4:421–424, 1933.

[5] Amir Dembo and Sumit Mukherjee. No zero-crossings for random polynomials and the heat equation. *The Annals of Probability*, 43(1):85–118, 2015.

[6] Amir Dembo and Sumit Mukherjee. Persistence of gaussian processes: non-summable correlations. *Probability Theory and Related Fields*, 169(3-4):1007–1039, 2017.

[7] M. Fekete. Uber die verteilung der wurzeln bei gewissen algebraischen gleichungen mit ganzzahligen koeffizienten. *Mathematische Zeitschrift*, 17:228–249, 12 1923.

[8] N. Feldheim, O. Feldheim, and S. Mukherjee. Persistence and ball exponents for gaussian stationary processes. *Preprint*, 12 2021.

[9] V. Glivenko. Sulla determinazione empirica delle leggi di probabilità. *Giorn. Ist. Ital. Attuari (in Italian)*, 4:92–99, 1933.

[10] M. Grigoriu and S. Balopoulou. A simulation method for stationary gaussian random functions based on the sampling theorem, 1992.

[11] C. G. Khatri. On certain inequalities for normal distributions and their applications to simultaneous confidence bounds. *The Annals of Mathematical Statistics*, 38:1853–1867, 12 1967.

[12] Rafał Latała and Dariusz Matlak. Royen's proof of the gaussian correlation inequality. In *Geometric Aspects of Functional Analysis: Israel Seminar (GAFA) 2014–2016*, pages 265–275. Springer, 2017.

[13] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[14] Thomas Royen. A simple proof of the gaussian correlation conjecture extended to multivariate gamma distributions. *Far East J. Theor. Stat.*, 48:139–145, 2014.

[15] Caroline Uhler. Gaussian graphical models: An algebraic and geometric perspective. *Preprint*, 07 2017.

[16] V. N. Vapnik and A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications*, 16(2):264–280, 1971.

תקציר העבודה בעברית

תהליך גאוסי הוא תהליך סטוכסטי שבו כל אוסף סופי של משתנים אקראיים הנמצאים מתוך התהליך מצוייד בהתפלגות נורמלית רב-ממדית. תהליכים אלה שימושיים ביותר בפרקטיקה ומהווים את הבסיס לטכניקות למידת מכונה חדשניות רבות ולדגמים לחיזוי רצפים זמניים. אנו ממקדים את תשומת הלב שלנו על תהליכים גאוסיים סטציונריים ממורכזים עם נתיבים רציפים (CSGP) ומחפשים אלגוריתמים שמעריכים סטיות קטנות וגדולות ברמת דיוק והתאמה פרקטיות. במסגרת עבודה זו אנו מתמודדים עם האתגר של הערכת אירועים נדירים על ידי הצגת שיטות הערכה עכשוויות והצעת שיטות חדשות להערכה וסימולציה.

עבודה זו נעשתה בהדרכתם של פרופ׳ שימי הבר וד״ר נעמי פלדהיים מן המחלקה
למתמטיקה של אוניברסיטת בר-אילן.

אוניברסיטת בר-אילן

אומדן קצב הדעיכה של ההסתברות לסטיות קטנות עבור תהליכים גאוסיים סטציונאריים

יובל לביא

עבודה זו מוגשת כחלק מהדרישות לשם קבלת תואר מוסמך במחלקה למתמטיקה של
אוניברסיטת בר-אילן

רמת גן ה'תשפ"ד