Exploring the Impact of Pseudo and Quasi Random Number Generators on Monte Carlo Integration of the Multivariate Normal Distribution

Marcus Stolz, Jonathan Ramani Christopher

6 januari 2024

Supervisor: Johan Larsson
Bachelor's Thesis in Statistics, 15 credits
The Department of Statistics
Lunds Universitet

Abstract

This thesis examines the effects of pseudo and quasi-random number generators on the accuracy and efficiency of Monte Carlo Integration in the case of the multivariate normal distribution. The study compares the performance of the Mersenne Twister (a pseudo-random number generator) with Sobol and Halton sequences (quasi-random number generators). By evaluating these generators across different dimensions and scenarios, the research aims to determine their impact on the behaviour of Monte Carlo Integration.

Our findings indicate notable differences in the variance of estimates. In cases involving integration of the distribution's "bump," the Sobol sequence exhibits higher variance, suggesting sensitivity to specific distributional characteristics. Additionally, in the bivariate case, extreme correlations result in increased variance, particularly for Sobol sequences. An interesting result is that from the fifth dimension onward, the Halton sequence demonstrates a notable increase in computational demand compared to the other generators. Due to computational constraints, we have been unable to go beyond 7 dimensions.

This thesis contributes to the field of computational statistics by providing insights into the optimal choice of random number generators for Monte Carlo methods in multivariate statistical analysis.

1 Foreword

We would like to begin by expressing our gratitude to Johan Larsson, PhD candidate at Lund University Department of Statistics, whose guidance and support has been invaluable throughout our research. Your mentorship has been instrumental in helping us to navigate throughout the jungle of our work, and ensuring that we could carry it through to completion.

As you continue to advance in your own academic journey, we wish you every success.

From Marcus and Jonathan, thank you!

Contents

2 Introduction

Monte Carlo integration represents a powerful technique for evaluating difficult integrals that are otherwise challenging to solve analytically. Central to this method is the concept of sampling, which necessitates the use of random numbers. The type and characteristics of these random numbers are crucial, as they influence the integration's accuracy and efficiency.

In the realm of random number generation, two primary types are considered: pseudo-random and quasi-random numbers. Pseudo-random numbers are algorithmically generated and exhibit properties of randomness, while quasi-random sequences are designed to cover the space more uniformly, appearing less random but often leading to more efficient integration in Monte Carlo methods.

The intuitive belief that a more random sample leads to better integration results is questioned by the effectiveness of less random, quasi-random sequences. This counter intuitive phenomenon forms the core of our investigation. We aim to explore the impact of choosing between pseudo and quasi-random number generators, particularly focusing on the Monte Carlo integration of the multivariate normal distribution.

3 Background

3.1 Random number generation

Random numbers have many uses in scientific computing and simulation, and specifically in monte carlo methods. The generation of random numbers has therefore been an important problem, with a long history of different approaches. Before modern computing, statisticians that needed random numbers had to use random numbers that were published in large tables. One early example of this being Tippets table of 41 600 random digits Tippett (1927) taken from a 1925 census report, supposedly on the suggestion of the famous statistician Karl Pearson L'Ecuyer & Montreal (2017). Later in the 40's the RAND corporation started the project to create the first fully automatically generated sequence of one million random decimal digits, using early computing, resulting in *A Million Random Digits with 100,000 Normal Deviates* (1955).

In the second half of the 20th century, with the development of computing, algorithmic random number generation became more prevalent. However these algorithms cannot create truly random numbers, because the algorithms follow deterministic rules and are therefore not stochastic and the digital logic of computing is not compatible with the concept of randomness. Therefore a deterministic computer algorithm cannot create stochastic sequences, only sequences that are approximately random. Johan von Neumann, one of the pioneers of algorithmic random number generation, jokingly wrote "Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin. For, as has been pointed out several times, there is no such thing as a random number there are only methods to produce random numbers, and a strict arithmetic procedure of course is not such a method." Von Neumann (1951).

The type of numerically generated random numbers or observed are typically categorized as listed below James (1988). We'll delve into the intricacies of some examples of these number generators later. Truly random numbers however will not be discussed further in the paper.

- *Truly random* numbers originate from unpredictable physical phenomena, such as atmospheric noise RANDOM.ORG (2024). Their limitations include a finite supply and the inability to reproduce them using seeds.
- *Pseudo-random* numbers are computationally generated and exhibit high discrepancy, appearing highly random and are generally faster and more flexible than truly random numbers.
- *Quasi-random* numbers, also computationally generated, have low discrepancy, which means they are more uniformly distributed across a range, therefore appearing less random, offering advantages in certain numerical applications.

3.2 Mersenne Twister, the standard pseudo random number generator

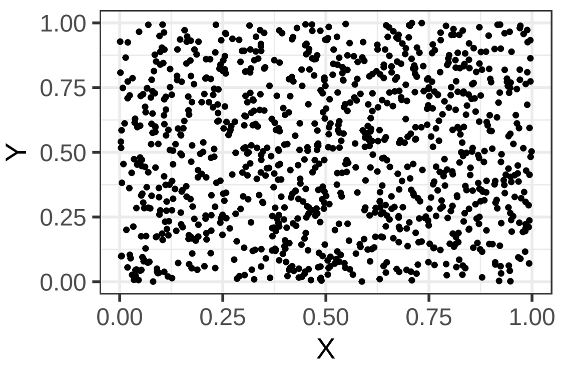


Figure 1: Two-dimensional Pseudo Sequence, $n = 1000$

There are countless algorithms for the creation of pseudo-random numbers; however, since its inception in 1997 by Japanese statisticians Makoto Matsumoto and Takuji Nishimura, the Mersenne Twister has become the standard algorithm for pseudo-random number generation.

The Mersenne Twister is a twisted generalized feedback shift register type of random number generator with a very long period without repetition. It is widely used because of its efficiency, good statistical properties, and stability across dimensions. The algorithm is the standard random generator in many programming languages; for example, the `runif()` function in base R uses it.

Originally, the generator was proposed to be named "Primitive Twisted Generalized Feedback Shift Register Sequence", but they thankfully settled on the less mouthful name *Mersenne Twister*.

The name Mersenne Twister stems from the fact that the period length of the random numbers is determined by a Mersenne prime. A Mersenne prime is in the form

$$M_p = 2^p - 1,$$

where p is a prime number. These primes are named after Marin Mersenne, a French priest and mathematician from the 17th century *Encyclopedia of Mathematics* (2023).

The Mersenne twister can be defined by the vector Y_k, consisting of 623 integers

$$Y_0 : \text{Seed}$$
$$Y_{k+1} = AY_k$$
$$Y_k = (X_{k-n+1} \cdots X_k)^T .$$

Here, Y_0 is the initial seed, and A is a matrix of dimension $19,937$ with elements having values 0 or 1 Noel (2016). This is simplified explenation of the Mersenne Twister, for a more indepth explenation go the orginal paper from Matsumoto & Nishimura (1998).

The Mersenne Twister is based on a linear recurrence, which is a sequence of terms where each term is a linear function of the preceding term *Encyclopedia of Mathematics* (2023). A famous example of such a sequence is the Fibonacci sequence, defined as

$$F_n = F_{n-1} + F_{n-2}.$$

The equation describing the linear recurrence for the Mersenne Twister is

$$x_{k+n} := x_{k+m} \oplus (x_k | x_{k+1})\, A, \quad (k = 0, 1, \ldots).$$

- n: Degree of the recurrence.
- m: Integer with $1 \leq m \leq n$, used in indexing.
- w: Word size in bits.
- r: Number of bits considered, with $0 \leq r \leq w - 1$.
- A: $w \times w$ matrix with binary entries.
- $x_0, \ldots, x_{n-1}$: Initial seeds.
- x_k^u: Upper $w - r$ bits of x_k.
- x_{k+1}^l: Lower r bits of x_{k+1}.
- $\oplus$: Bitwise XOR operation.
- $|$: Concatenation of vectors.

.

We will use the Mersenne twister in this paper as our representative for the pseudo random numbers and will refer to the Mersenne Twister generated numbers as Pseudo generated.

3.3 Quasi random numbers

3.3.1 Sobol

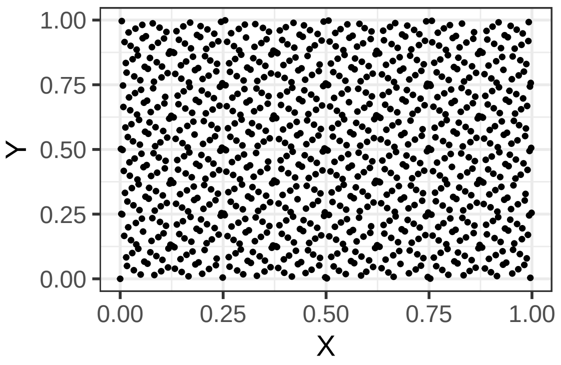


Figure 2: Two-dimensional Sobol Sequence, n = 1000

Sobol's sequence is a quasi-random number sequence, which means it consists of numbers generated to exhibit properties of randomness while maintaining a structured and systematic pattern. These sequences are widely used in Monte Carlo simulations for their lower discrepancy, leading to faster convergence and stable estimations. Discrepancy refers to how well the quasi-random numbers are distributed across the sample space. Lower discrepancy indicates a more even and uniform distribution of these numbers, which is crucial in Monte Carlo simulations. Convergence in Monte Carlo simulations occurs when the results approach greater accuracy and stability with an increasing number of iterations using quasi-random numbers. Introduced by I.M. Sobol' as a construction of (t,s)-sequences, where "t" represents dimensionality which defines how many dimensions each quasi-random number in the sequence will have. The "s" parameter is the number of different directions in which the sequence is generated within each dimension. That is, it defines how many different variations of the sequence exist within each dimension.

The Sobol sequence is a generalized Niederreiter sequence with the parameter characteristics $b = 2$, $P_1(x) = x$, and for $j = 2, \ldots, s$, $p_j(x)$ is the $(j-1)$-th primitive polynomial in a sequence of all primitive polynomials over F_2 — the finite field of two elements — arranged in nondecreasing order of degrees Leobacher & Pillichshammer (2015).

The Generalized Niederreiter Sequence is defined as

$$\frac{y_{j;i;k}(x)}{p_j(x^i)} = \sum_{r=1}^{\infty} a^{(j)}(i,k,r) \cdot x^{-r} \tag{1}$$

Here,

- $y_{j;i;k}(x)$ is the Sobol sequence element.
- $p_j(x)$ is the j-th primitive polynomial.
- $a^{(j)}(i,k,r)$ are the coefficients of the corresponding primitive polynomial.

The Sobol sequence construction involves using primitive polynomials to generate quasi-random numbers, providing improved distribution properties compared to standard pseudorandom sequences. The equation reveals a relationship between the Sobol sequence element, the primitive polynomial, and the coefficients of the polynomial. Let us elaborate upon this further by explaining how each component contributes to the Sobol sequence construction.

The $y_{j;i;k}(x)$ component represents a specific element in the Sobol sequence for dimension j, corresponding to the i-th component and k-th direction. The $p_j(x)$ term is responsible for generating Sobol sequences in each dimension. The choice of primitive polynomials is crucial in ensuring effective coverage of the sample space by the Sobol sequence. The $a^{(j)}(i,k,r)$ term determines the contribution of each term in the series expansion, with specificity to the dimension (j), component (i), and direction (k) of the sequence.

As the dimension (j) increases, the series expansion incorporates additional terms, creating a more complex mathematical relationship between the Sobol sequence element (the generated number) and the primitive polynomial. That is, for each dimension, the series contains more terms, making it harder to maintain a structured and uniform distribution of quasi-random numbers. Therefore, it becomes harder to maintain a uniform distribution and low discrepancy as the dimension (j) increases. Sobol sequences are designed to address this challenge by integrating primitive polynomials into their construction. These primitive polynomials play a vital role in shaping the distribution properties of the sequence, ensuring that it remains quasi-random and uniform across multiple dimensions.

3.3.2 Halton

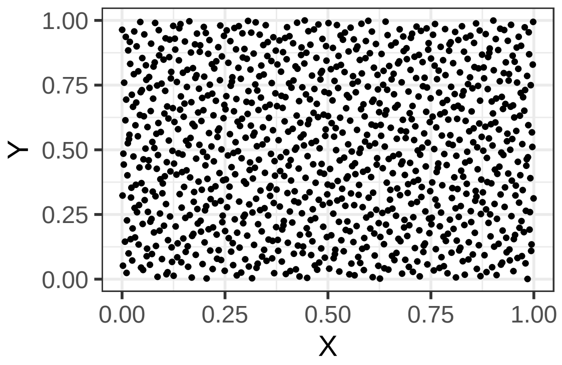


Figure 3: Two-dimensional Halton Sequence, n = 1000

The Classic Halton sequence is a low-discrepancy quasi-random number sequence, widely utilized in various numerical methods, including Monte Carlo simulations and numerical integration. Introduced by J. H. Halton, this sequence is part of the broader class of Halton sequences designed to provide more even coverage of the sample space compared to traditional pseudo-random sequences.

The construction of the Classic Halton sequence involves a set of radical-inverse base sequences. A base sequence is a series of values generated using the radical-inverse function with a chosen prime base for each dimension. This process generates a set of values, forming the base sequence for that dimension. For each dimension j, a prime number is chosen as the base, and the sequence is generated by applying the radical-inverse function to the corresponding base. Specifically, the i-th term in dimension j of the Halton sequence, denoted as $H_{j;i}$, is given by

$$H_{j;i} = \sum_{k=1}^{\infty} \frac{d_k}{p_j^k} \tag{2}$$

where $H_{j;i}$ is the i-th term in dimension j of the Classic Halton sequence, and p_j is the prime base chosen for dimension j.

The Classic Halton sequence is known for its deterministic nature and low-discrepancy properties. Its deterministic construction, in contrast to pseudo-random sequences, allows for reproducibility, essential in various scientific and computational applications.

One advantage of the Classic Halton sequence is its suitability for numerical problems involving low to moderate dimensions. As the dimensionality increases, challenges in maintaining quasi-random properties may arise. However, for many applications, particularly in low-dimensional scenarios, the Classic Halton sequence offers improved convergence rates and more uniform distribution of points compared to traditional pseudo-random sequences.

In summary, the Classic Halton sequence, rooted in the concept of radical-inverse functions, provides a deterministic and quasi-random approach to generating sequences of points. While its advantages are notable in lower dimensions Leobacher & Pillichshammer (2015), careful consideration is needed in higher dimensions to maintain its quasi-random properties.

3.4 Computation time of pseudo, Sobol and Halton

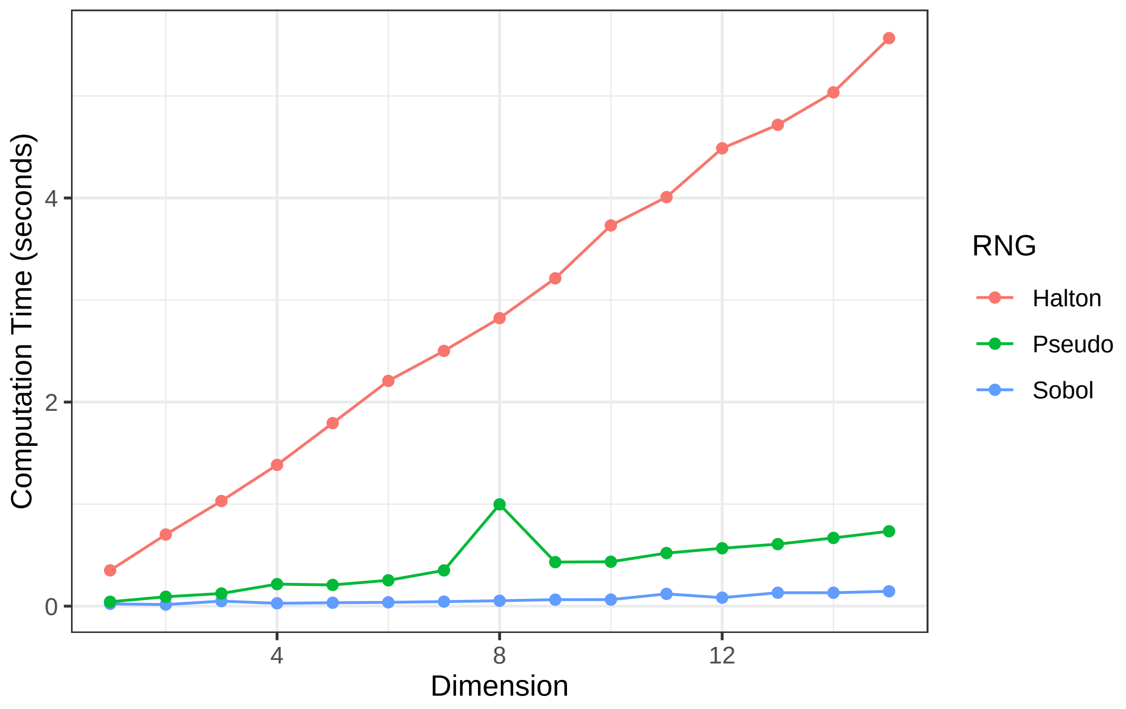


Figure 4: Computation time of 1 million numbers, by dimension

One important aspect to consider when choosing a random number generators for a Monte Carlo method is the computational cost. How long will it take to run the simulation? Especially in high dimensional cases, where the Curse of Dimensionality, see 3.6, dictates exponentially larger samples with increasing dimension. In these cases the computation time of the monte carlo experiment can become very long. Therefore the efficiency of the random numbers can become important.

In figure 4 we clearly see that the computational cost of Halton grows substantially faster, than the other number generators. Sobol is slightly faster than Pseudo (Mersenne Twister) in all dimensions.

In the dimensions 1-15 the complexity or computation time for all the number generator seem to grow linearly, where the cost of sobol and pseudo almost seem constant.

3.5 Monte Carlo Integration

Monte Carlo integration is a numerical technique used to estimate definite integrals, especially in high-dimensional spaces, through the utilization of random sampling. Named after the Monte Carlo casino known for chance and randomness, this method leverages random sampling to approximate integrals Kalos & Whitlock (2008). Let us look at an example of a Monte Carlo integration:

The basic idea behind Monte Carlo integration is the use of random sampling to generate points within the domain of the integral. By evaluating the function at these randomly chosen points and taking the average of the results, we get an approximation of the integral. This process leverages the law of large numbers, that is, when the number of samples increases, the approximation converges towards the true value. To illustrate this, let us consider an example where we want to calculate the definite integral of a function $f(x)$ over the interval $[a, b]$. We can express this as

$$\int_a^b f(x)\,dx \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

where N is the number of random samples, and x_i are points uniformly sampled from the interval $[a, b]$. As N increases, the accuracy of the approximation improves.

3.5.1 Derivation of simple Multidimensional Monte Carlo Integration

Consider a multidimensional integral I over a domain defined by the intervals $[a_X, b_X]$, $[a_Y, b_Y]$, and $[a_Z, b_Z]$. The integral I can be expressed as

$$I = \int_{a_Z}^{b_Z}\int_{a_Y}^{b_Y}\int_{a_X}^{b_X} f(x, y, z)\,dx\,dy\,dz.$$

Let w_i be an independent and identically distributed (i.i.d.) random variable uniformly distributed over $[a_i, b_i]$ with probability density function defined as $p(w_i) = \frac{1}{b_i - a_i}$. The joint pdf for the random vector (w_X, w_Y, w_Z) is then given by the product of individual pdfs:

$$p(w_X, w_Y, w_Z) = \frac{1}{(b_X - a_X)(b_Y - a_Y)(b_Z - a_Z)}.$$

Using the property of expectation over this joint distribution, the integral I can be rewritten as

$$\begin{aligned}
I &= \int_{a_Z}^{b_Z}\int_{a_Y}^{b_Y}\int_{a_X}^{b_X} \frac{f(x, y, z)}{p(w_X, w_Y, w_Z)} p(w_X, w_Y, w_Z)\,dx\,dy\,dz \\
&= \frac{1}{p(w_X, w_Y, w_Z)}\int_{a_Z}^{b_Z}\int_{a_Y}^{b_Y}\int_{a_X}^{b_X} f(x, y, z)p(w_X, w_Y, w_Z)\,dx\,dy\,dz \\
&= (b_X - a_X)(b_Y - a_Y)(b_Z - a_Z)\mathbb{E}_{w_X, w_Y, w_Z}[f(x, y, z)].
\end{aligned}$$

Applying the Law of Large Numbers, we can estimate the integral I by the sample mean $\overline{f(x, y, z)}$.

$$\hat{I} = \prod_{i=1}^{3}(b_i - a_i)\overline{f(x, y, z)}.$$

In the general case for a D-dimensional integral, the estimate $\hat{I}$ is given by

$$\hat{I} = \prod_{i=1}^{D}(b_i - a_i)\overline{f(\underline{\mathbf{x}})} \tag{3}$$

3.5.2 Properties of Monte Carlo integral estimate

We have shown in the previous section that the estimator is unbiased. The estimate is unbiased, because the sample mean is an unbiased estimator of the expected value by the law of Large numbers.

We can derive the variance of the estimator as follows.

$$\begin{aligned}
V[\hat{I}] &= V[\prod_{i=1}^{D}(b_i - a_i)\frac{\sum_{i=1}^{n} f(\underline{\mathbf{x}})}{n}] \\
&= [\prod_{i=1}^{D}(b_i - a_i)]^2 \frac{\sum_{i=1}^{n} V[f(\underline{\mathbf{x}})]}{n^2} \\
&= [\prod_{i=1}^{D}(b_i - a_i)]^2 \frac{nV[f(\underline{\mathbf{x}})]}{n^2} \\
&= \frac{1}{n}\prod_{i=1}^{D}(b_i - a_i)^2 \sigma_{\underline{\mathbf{x}}}
\end{aligned}$$

The resulting expression only has an n in the denominator, which means that

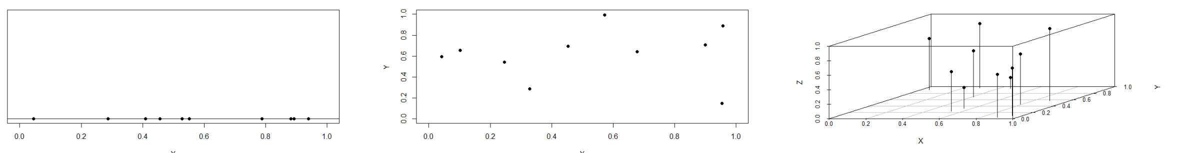
$$\lim_{n\to\infty} V[\hat{I}] = 0$$

or in other words that the estimator is consistent. We have now shown that the estimator is consistent and unbiased, the estimator is therefor efficient.

3.6 The Curse of Dimensionality

The Curse of Dimensionality is one of the central problems of probability and data analysis in higher dimensions. Curse of Dimensionality can be seen as the exponential growth in algorithmic complexity related to the increase in dimension Vershynin (2018). In terms of MC integration, the Curse of Dimensionality creates difficulties related to sampling due to the increase in dimension, which demands an equal increase in sample size in order to maintain the efficiency of the estimate.

3.7 Example of the Curse of Dimensionality, related to sample size



(a) Sample of a one-dimensional space

(b) Sample of a two-dimensional space

(c) Sample of a three-dimensional space

Figure 5: The curse of dimension, related to sample size

Consider a geometric space S with each side measuring l. The volume of this space grows exponentially with the number of dimensions Dim. This relationship can be expressed mathematically as $S = l^{\text{Dim}}$.

To maintain consistent coverage of this space with a sample, the sample size n must increase in proportion to the expanding space. This ensures that the coverage level c remains constant across different dimensions. Mathematically, this relationship is given by $c = \frac{n}{l^{\text{Dim}}} \Leftrightarrow n = c \cdot l^{\text{Dim}}$.

The implication is that as the dimensionality increases, the sample size required to maintain the same level of coverage also increases exponentially.

3.8 Multivariate normal distribution

The multivariate normal distribution is a generalized version of the normal distribution to higher dimensions. In our case, we will use the standard normal distribution and the bivariate case where we will check different correlation values. As such, it deals with a vector of random variables X, instead of a single variable. This distribution is characterized by two key components: a mean vector μ and a covariance matrix Λ.

Definition A random vector X with $\mathrm{E}X = \mu$ and $\mathrm{Cov}X = \Lambda$, such that $\det \Lambda > 0$ is $\mathcal{N}(\mu, \Lambda)$-distributed if and only if the density equals

$$f_X(x) = \frac{1}{(2\pi)^{n/2}} \frac{1}{\sqrt{\det \Lambda}} \exp\left\{ -\frac{1}{2}(x-\mu)'\Lambda^{-1}(x-\mu) \right\}, \quad x \in \mathbb{R}^n,$$

where the covariance matrix $\mathrm{Cov}X = \Lambda$, mean vector $\mathrm{E}X = \mu$ and the x vector can be represented as follows:

$$\Lambda = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1n} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1} & \sigma_{n2} & \cdots & \sigma_n^2 \end{bmatrix}, \quad \mu = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}.$$

Gut (1995)

4 Methodology

To test the effects of the choice of pseudo or quasi random number generators on the monte carlo integration of the multivariate normal distribution we explore three main cases. In these three cases we observe the effect of increasing dimension and sample size on the integral estimates. We also look at the effect of correlation in the bivariate case.

In the first case we keep multidimensional probability space constant, by adjusting the integration limits with increasing dimension. For the second case we keep the integration limits constant with increasing dimension. For the last case we integrate over the entire distribution. In all the experiments we increase the sample size exponentially to the dimension, to keep the estimates consistent in between dimensions. This however limits our experiment to seven dimension due to computational constraints.

4.1 The four cases explored

4.1.1 1. Constant Probability Space

We integrate over a constant probability space of 0.05. We adjust the lower integration limits for each increasing dimension to keep the space constant. As we increase the dimension, we need to correspondingly increase $P(x_1|x_2,\dots,x_n)$, the probability mass of one dimension.

$$\begin{aligned} P(x_1|x_2,\dots,x_n)^D &= 0.05 \\ \Leftrightarrow P(x_1|x_2,\dots,x_n) &= 0.05^{\frac{1}{D}} \\ &\Rightarrow \text{lower} = -qnorm\left(P(x_1|x_2,\dots,x_n)\right) \end{aligned}$$

When this adjustment is made to the integration limits the the distance between the integration limits become slightly larger for each increase in dimension, so the estimate will be come weaker for each increase in dimension. To adjust for this we introduced a compensation factor to keep the sample to integration space ratio constant.

The compensation factor is simply the relative difference in the integration limits from one dimension to the next. Let $\text{Comp\_factor} = \frac{\text{Length}_D}{\text{Length}_{D-1}}$, where Length_D is the length of the integration limits at dimension D.

Consequently, the sample size increases as $n = (\text{Comp\_factor} \cdot n_0)^D$, where n_0 is the initial sample size for the one-dimensional case (i.e., $n = 10$).

4.1.2 2. Constant Integration Limits

In the second case, we integrate with constant integration limits. This means that the probability space being integrated decreases with increasing dimension. In our case, we integrate over a probability of 0.95 in the first dimension, which decreases as a function of the dimension. Specifically, the integrated probability space I diminishes with increasing dimension D according to the formula $I = 0.95^D$. To control for the curse of dimensionality, the sample size is increased exponentially, represented as n^D, where n is the base sample size.

4.1.3 3. Full Probability Space

For the third case, we integrate over approximately the entire distribution, aiming for the integral to equal one for all dimensions. This approach is based on the premise that the total probability space should be fully represented. However, as the dimension increases, the integral slightly decreases because it is finite and doesn't extend from $-\infty$ to $+\infty$. Despite this, the difference is negligible and does not significantly impact the overall integration for practical purposes. To compensate for the diminishing integral and to control for the curse of dimensionality, we also increase the sample size exponentially with the dimension in this case.

4.1.4 4. Correlated bivariate case

For the last case we look the effects of correlation on the Monte Carlo estimates. In this case we use the integration limits that cover the entire distribution and a 0.05 probability, and we cycle through a sequence of the bivariate correlation between (-0.999,0.999, by = 0.001) and we observe the variance of the estimates. We have decided to not look at the estimates or bias, because they would be different for each single iteration and therefore difficult to compare.

5 Results

5.1 Case 1.

5.1.1 Estimates

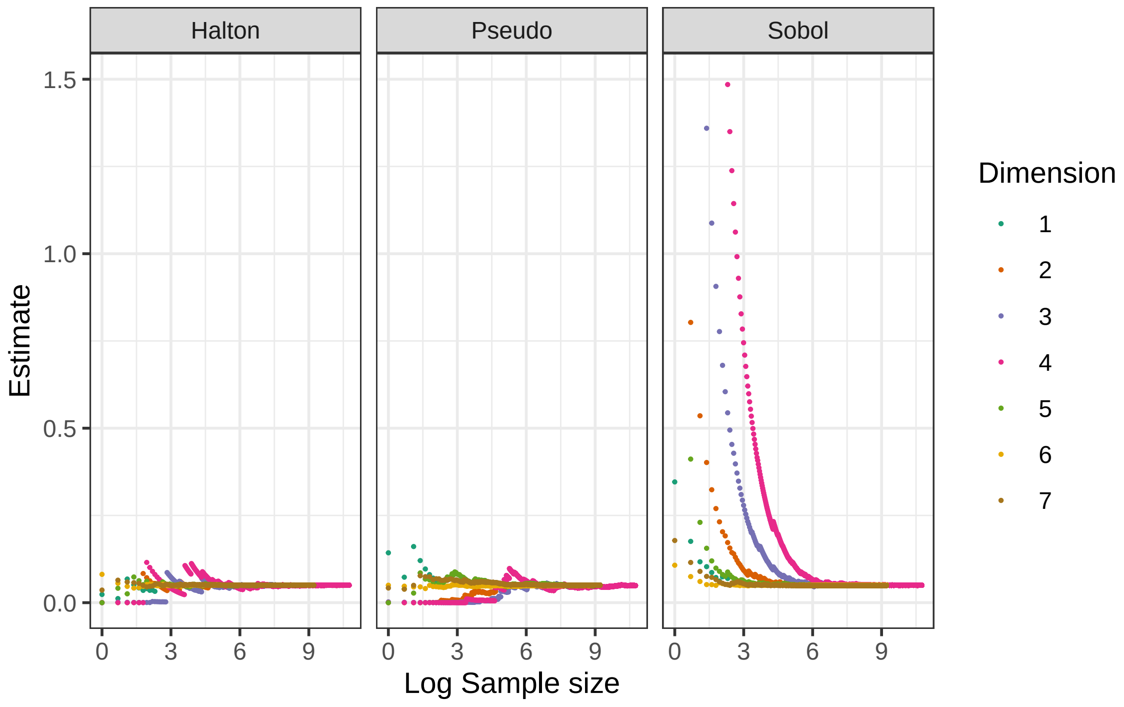


Figure 6: Sequential estimate by sample size, case 1. constant probability space.

We can see that across all the dimensions, the Halton estimation consistently has low variance. This can also be found in table 1, where we can see that the mean squared error (MSE) for Halton remains small for all dimensions. Similar to Halton, we see that Pseudo estimation also gives us low variance. The Sobol estimation exhibits higher variance compared to the Halton and Pseudo estimates.

5.1.2 Variance of estimates

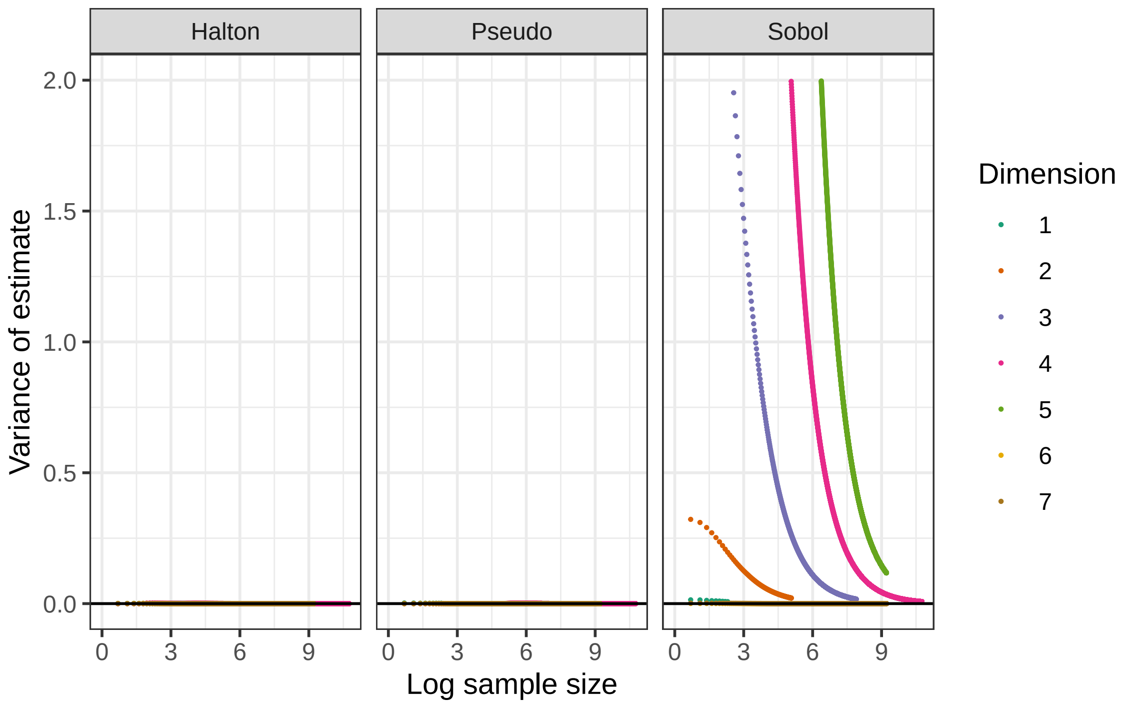


Figure 7: Sequential variance by sample size, case 1. constant probability space.

In 7 we can see that Halton and Pseudo have low variance near zero. For Sobol we see that the variance of estimate is substantially higher for dimensions 1, 2, 3, and 4. Therefore the convergence of the variance of estimate is faster for Halton and Pseudo.

5.2 Case 2.

5.2.1 Estimates

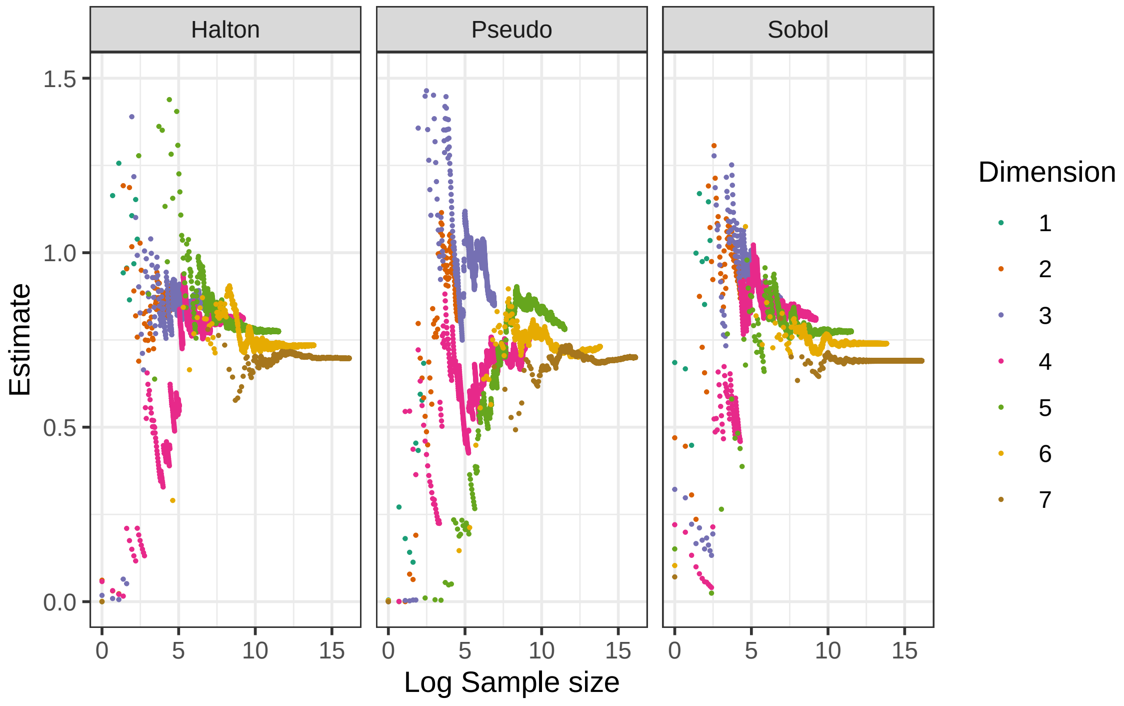


Figure 8: Sequential estimate by sample size, case 2. constant integration limits.

When we do estimation with constant integration limits we see that the estimate for the integral converges to a lower level with increasing dimensions, this is due to the second case having fixed integration limits as explained in 4.1.2. We see that across the sequences there is substantial under- and overestimation in the beginning, that slowly converges at very different paces as we increase sample size. We can see that the variance also varies significantly between the different dimensions. We see in table 4 that at dimension 1 the mean squared error (MSE) is relatively high and that the precision increases significantly as we increase dimensions. This is the true for all the sequences.

5.2.2 Variance of estimates

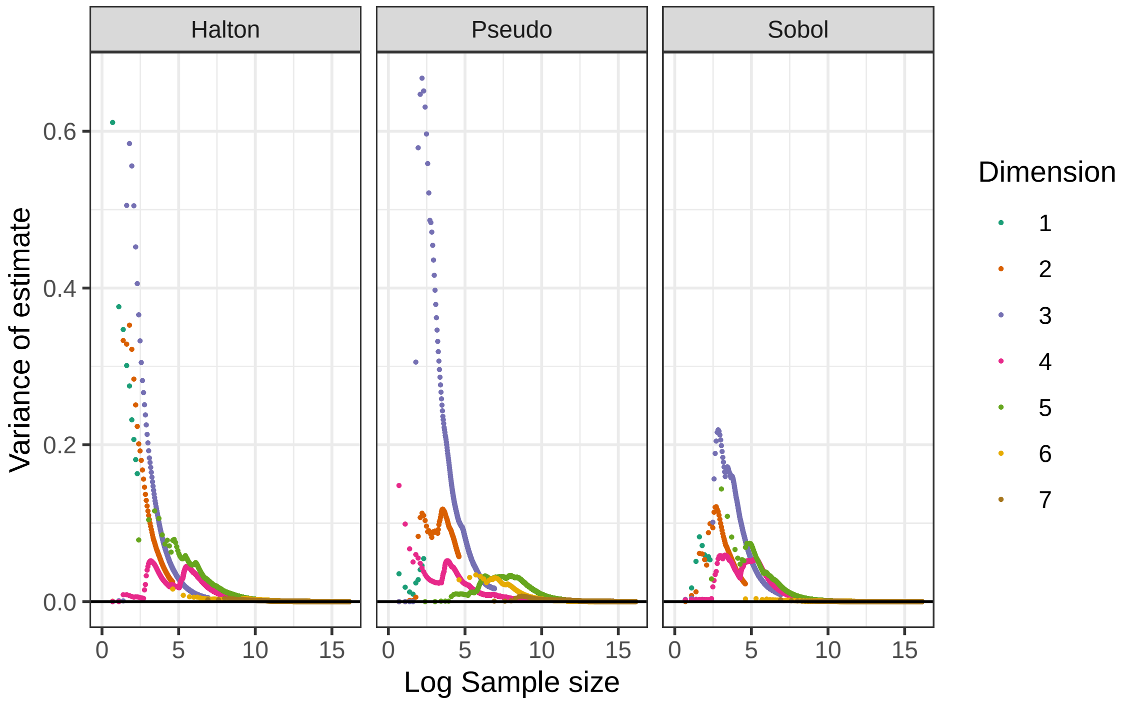


Figure 9: Sequential variance by sample size, case 2. constant integration limits.

The variance for the Halton and Psuedo sequences are substantial in the beginning and fall as sample size increases. For the Sobol sequence the variance estimates are lower than the aforementioned sequences but converge around the same sample size. This reflects the increase in sample size, we introduce in order to counter the Curse of Dimensionality.

5.3 Case 3.

5.3.1 Estimates

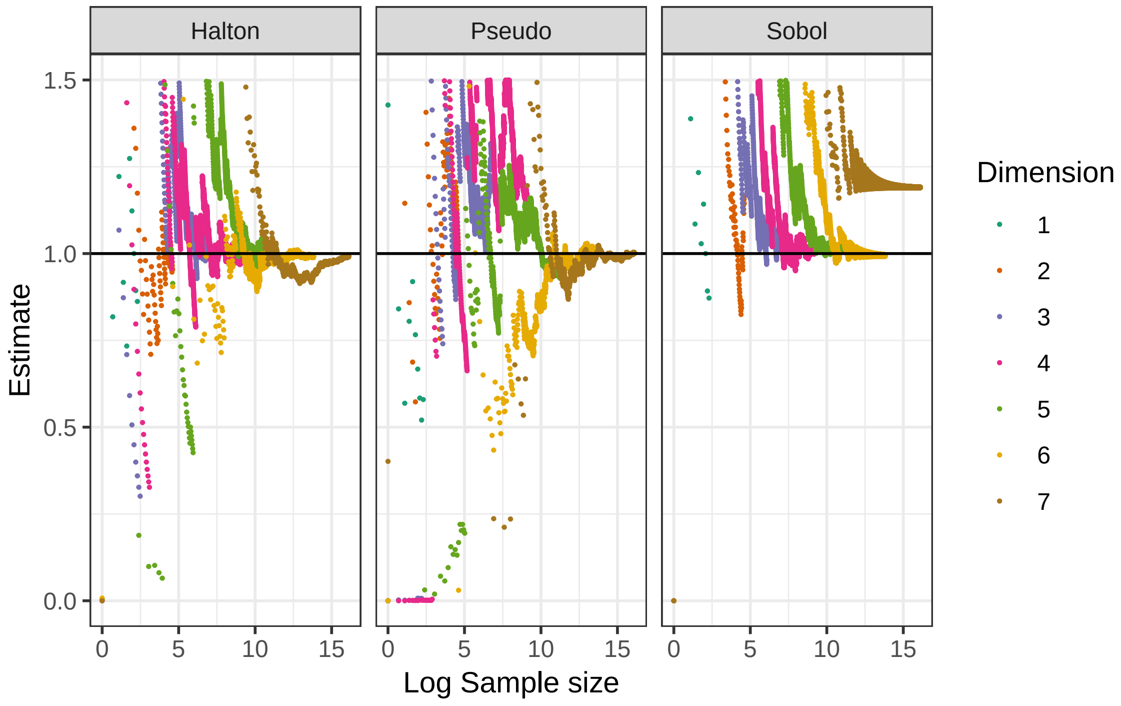


Figure 10: Sequential estimate by sample size, case 3. full distribution.

We see that the variance is high for all the sequences in the beginning and converge at different sample sizes. The Halton and Pseudo sequences seem to exhibit some level of underestimation for some dimensions. For the Sobol sequence we see that the estimations consistently leans towards overestimation. In table 7 we can see the Halton sequence mean squared error (MSE) ranged from 0.0074 to 0.7103 from dimension 1 to 7. The MSE for the Pseudo sequence shows varying results with less consistent performance compared to the Halton sequence. For the Sobol sequence we can see that the performance is mixed, with MSE ranging from 0.2251 and 16.7034. For the Sobol sequence, it should be noted that the estimates for dimension 7 appear to be anomalous. This observation suggests a possible discrepancy in the data or the method of calculation specific to this dimension is faulty.

5.3.2 Variance of estimates

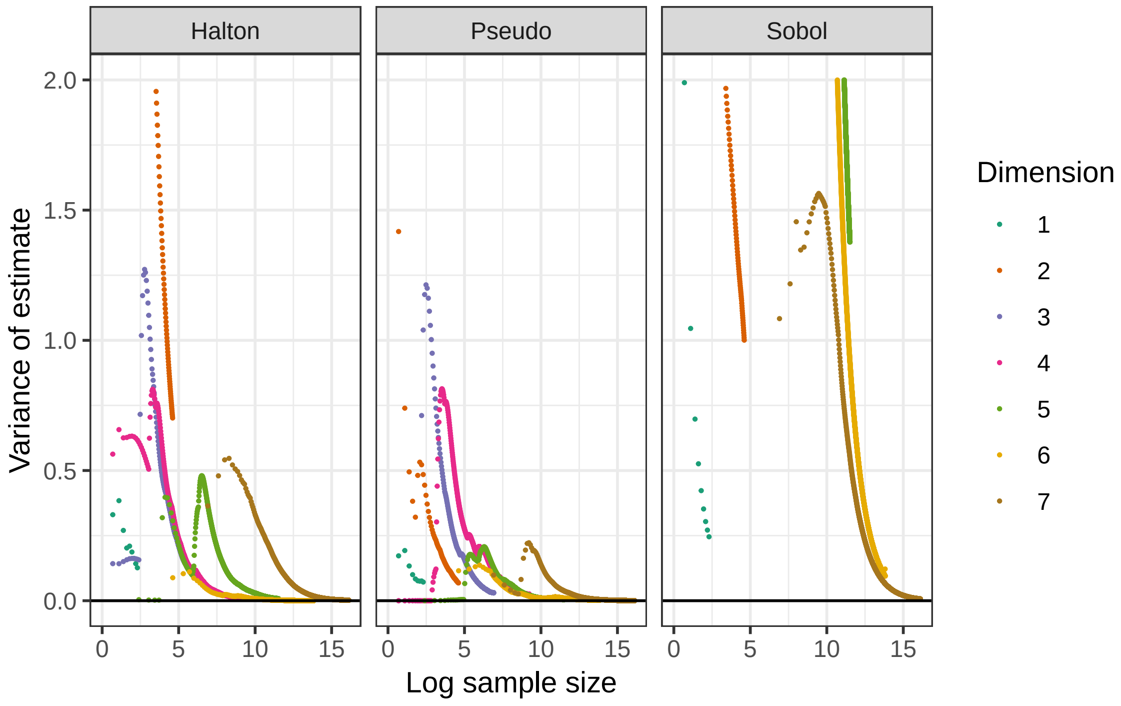


Figure 11: Sequential variance by sample size, case 4. Full distribution.

Figure 12: Sequential variance by sample size for different cases.

The variance of estimates is low for the Halton and Pseudo sequences, but substantially high for Sobol at the dimensions 5, 4 and 3. For the Halton sequence the variance of estimates remains relatively low and provides stable and consistent estimations. For the Sobol sequence the variance fluctuates significantly across dimensions and has less accuracy compared to the Halton and Pseudo sequences.

5.4 Case 4. Correlated bivariate case.

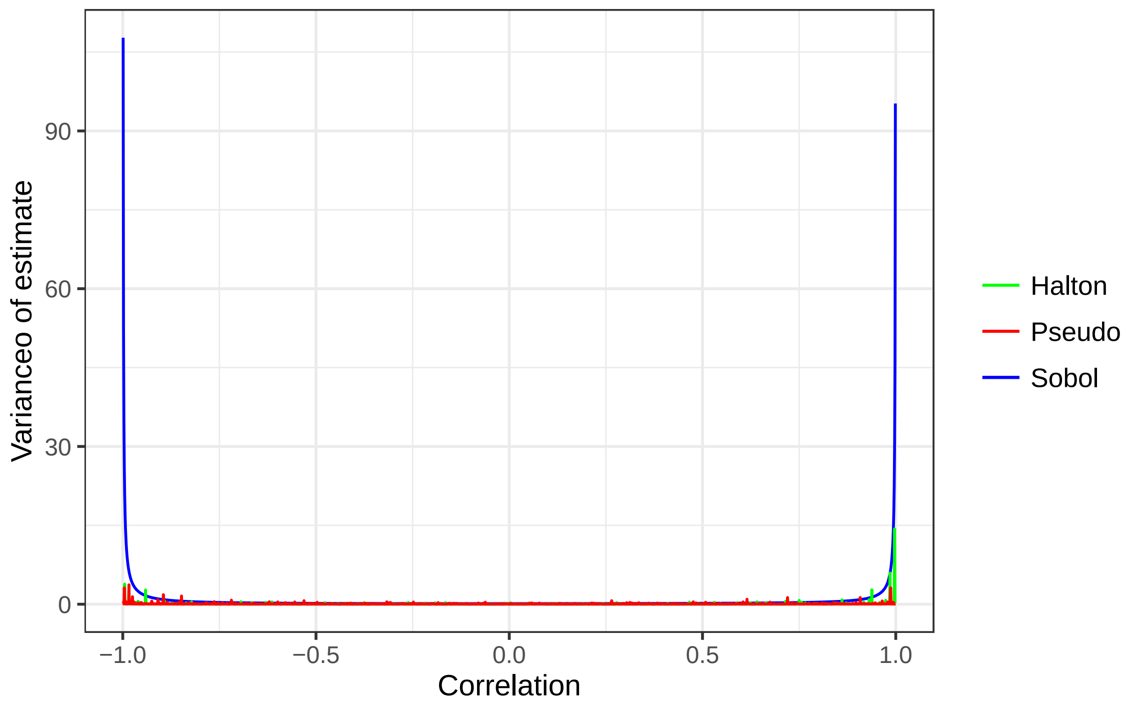


Figure 13: Variance of integral estimate, by correlation

The correlated bivariate cases show us that the correlation is high for Sobol at the extremes. In contrast, the Halton and Pseudo sequences consistently display low correlation in the variance of their estimates across the correlated bivariate cases.

5.5 Computation time

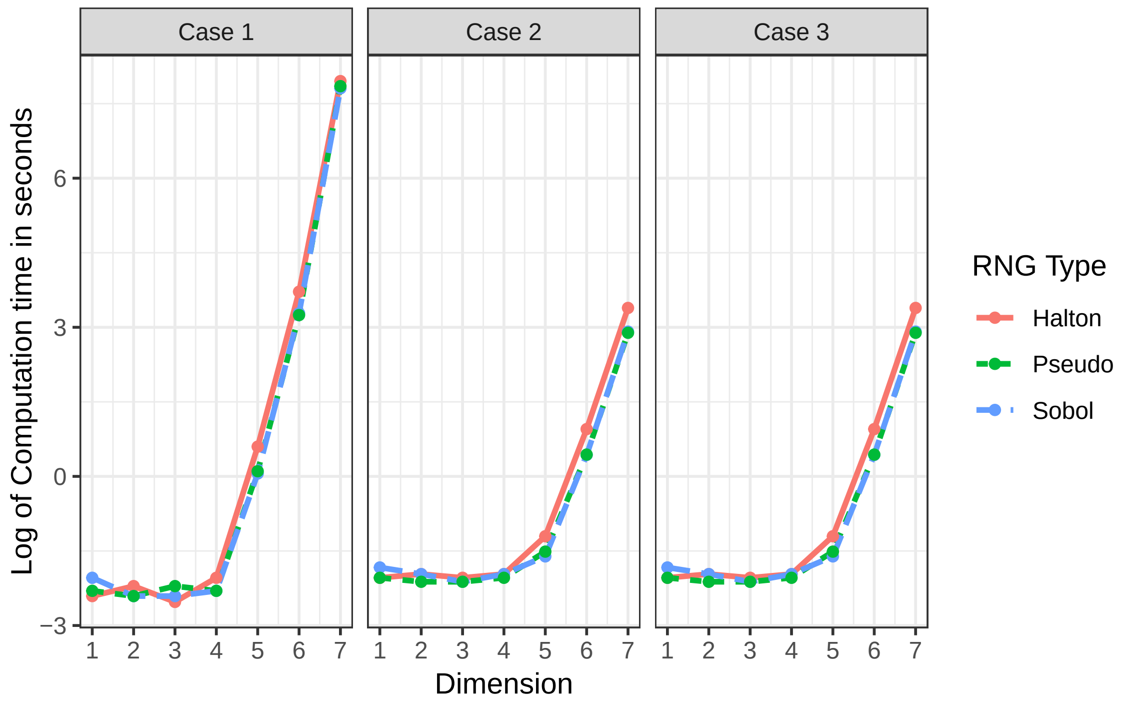


Figure 14: Computation time by dimension and case

For the first case with constant probability space, we can see a significantly larger increase in the computation time compared to the other cases. The second and third case with fixed limits and the full distribution have are not differentiable form each other. When we look at table 10 in the appendix, we see that the computation time of the second and third case are identical. The difference in the computation time between the cases can be explained by the increased sample size due to the compensation factor, see 4.1.1.

However in all three cases we see that the computation time of the Halton estimate is longer in the dimension 5-15. In the first 4 dimensions there are no real discernible differences between all generators and between Pseudo and Sobol there are no noticeable difference in all dimensions.

6 Discussion

The purpose of this paper was to explore the effect of the choice of quasi or pseudo random number generators for simple Monte Carlo integration of the multivariate normal distribution. In our limited experiment, due to computational constraints that limited us to the first seven dimensions, we found significant differences in behaviours between the random number generators Mersenne Twister (Pseudo), Sobol and Halton in Monte Carlo integration of the multivariate normal distribution.

6.1 Differences in variance of the estimates

Particularly in the first and third case the variance of the Sobol estimate was significantly higher than the other estimates. For example the variance of the Sobol estimate was close to 16 in the 7th dimension in the with the full distribution, see table 9. A possible explanation of why the variance of the sobol estimate, was similair in these two cases. Could be due to that the left integration limit of the first case, with a constant probability space, moves further to the left for each dimension. This will mean that, with increasing dimension the first case will approximate the third case, with the full distribution. For example the the left integration limit of the first case at the 7th dimension is at approximately -0.4, which means that a large part of the bump in the distribution is included. This means that there is a big difference in the value of possible observations that can occur. This is most likely more prevalent in the Sobol Estimates, particularly with small sample sizes, because of the Sobol sequence low discrepency. It has a larger and a more uniform, spread of observations that lead to big differences in observations. These big differences in observations lead to a bigger variance of the estimates.

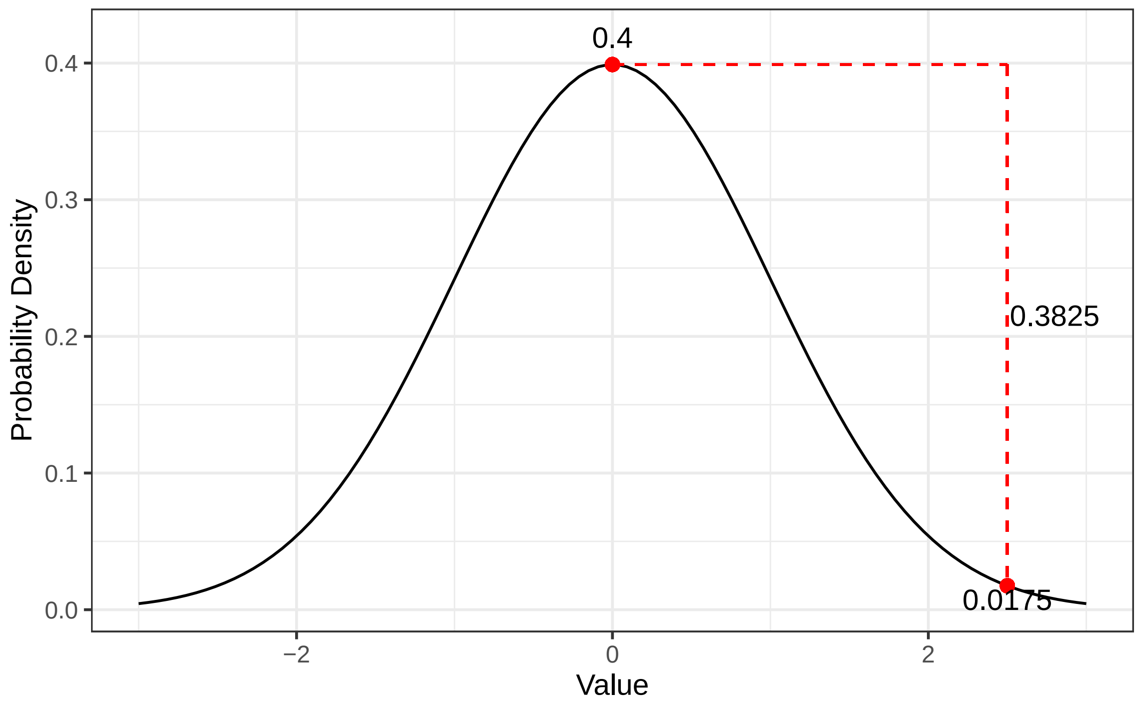


Figure 15: Standard normal distribution

In figure 15, we see an example of this effect.

This effect probably exists in other function or distributions with large differences in function values in the codomain. This issue however is insignificant with larger sample sizes. There are commonly used sampling methods used, that could elevate this issue that are not touched upon in this paper. However it could be interesting to explore this phenomena in other functions and in higher dimensions.

We also found that variance of the Sobol estimate increased sharply at extreme correlations close to 1 or -1 in the bivariate case. This is a very special and theoretical case, so the practical implications of this result are not clear. However the cause of the issue is likely the same as the other cases. That the correlated bivariate normal has a big range of function values in the codomain.

It should be noted that the Sobol estimate improves quite a bit with the increase of the sample size, so to skip a certain amount of the first values would greatly improve the sobol estimate. This is likely appropirate for most Monte Carlo integrations using Sobol numbers.

6.2 Differences in computation time

For the first four dimension we did not observe any noticeable differences in the computation time of the integral estimates. For dimension 5 and beyond we can see that the Halton estimate is significantly slower than the other estimates. No significant differences were found between Pseudo and the Sobol estimate. Most likely a difference in speed will appear between the Sobol and Pseudo estimate, if the sample size and dimensions are increased. Specifically the computational efficiency of the number generators in higher dimensions is an interesting topic to dive into further, however this task will need substantially more computing power, than was available for our study.

6.3 Contradictory result related to the Curse of Dimensionality

A small and slightly strange effect we found, was that the estimates of the first three dimensions were slightly more biased than the subsequent dimensions. Our understanding of the Curse of Dimensionality is that when we increase the sample size of the estimates exponentially with the increase in dimension, the efficiency of the estimators should be constant across dimensions. In our experiment, however we observe an increase in efficiency with the four first dimensions. This might just be because the sample size is quite small for the lower dimensions, or it might be because of some special characteristics of our experiment design with our different cases.

6.4 Future research

Further investigations in the effect of the choice of random number generators in Monte Carlo Integration is definitely warranted. Especially in higher dimensions, were the differences and limits of the random number generators are more visible. This research will likely need a lot more computational power, than was available to us in our study. The behaviour of the variance of the sobol estimate is particularly interesting, so further study into the causes of the increased variance of the Sobol Monte Carlo estimates are also justified. For example by integrating certain function with different special characteristics, dimensions and applied cases in for example financal Monte Carlo methods.

References

A Million Random Digits with 100,000 Normal Deviates (1955), RAND Corporation, Santa Monica, CA.

Encyclopedia of Mathematics (2023). Accessed: 2024-01-04.

Gut, A. (1995), *An intermediate course in probability*, Springer.

James, F. (1988), A review of pseudo-random number generators, Technical report, CERN-Data Handling Division. CERN-DD-78-20.

Kalos, M. H. & Whitlock, P. A. (2008), *Monte Carlo Methods*, Wiley-Blackwell.

Leobacher, G. & Pillichshammer, F. (2015), *Introduction to quasi-monte carlo: Integration and applications*, Springer.

L'Ecuyer, P. & Montreal, U. (2017), History of uniform random number generation.

Matsumoto, M. & Nishimura, T. (1998), 'Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator', *ACM Trans. Model. Comput. Simul.* **8**(1), 3–30.

Noel, K. (2016), 'Analysis of random generators in monte carlo simulation: Mersenne twister and sobol'.

RANDOM.ORG (2024), 'The history of random.org'. Accessed: 2024-01-03.

Tippett, L. H. C. (1927), *Random Sampling Numbers*, Cambridge University Press.

Vershynin, R. (2018), *High-dimensional probability: An introduction with applications in Data Science*, Cambridge University Press.

Von Neumann, J. (1951), 'Various techniques used in connection with random digits', *National Bureau of Standards Applied Mathematics Series* **12**, 36–38.

7 Appendix

7.1 Tables of metrics of the Monte carlo estimates

7.1.1 Case 1. constant probability space

Table 1: Halton Estimate, case 1. constant probability space.

Dimension	Estimate	Variance	MSE	Calculation_time
1	0.0324756	0.0002255	0.0003563	0.09
2	0.0505412	0.0001035	0.0001105	0.11
3	0.0497815	0.0000214	0.0000223	0.08
4	0.0500642	0.0000036	0.0000036	0.13
5	0.0500564	0.0000003	0.0000003	1.82
6	0.0499794	0.0000006	0.0000006	41.09
7	0.0499806	0.0000000	0.0000000	2845.72

Table 2: Pseudo Estimate, case 1. constant probability space.

Dimension	Estimate	Variance	MSE	Calculation_time
1	0.0504635	0.0014469	0.0030167	0.10
2	0.0570300	0.0002368	0.0005839	0.09
3	0.0445961	0.0001008	0.0001106	0.11
4	0.0490551	0.0000138	0.0000180	0.10
5	0.0502878	0.0000034	0.0000037	1.11
6	0.0499580	0.0000003	0.0000006	25.69
7	0.0500910	0.0000001	0.0000001	2569.55

Table 3: Sobol Estimate, case 1. constant probability space.

Dimension	Estimate	Variance	MSE	Calculation_time
1	0.0694342	0.0075263	0.0114449	0.13
2	0.0526678	0.0213091	0.0227606	0.09
3	0.0505727	0.0170197	0.0171434	0.09
4	0.0498776	0.0075833	0.0075875	0.10
5	0.0501241	0.0023098	0.0023100	1.06
6	0.0503179	0.0005236	0.0005237	26.59
7	0.0489664	0.0000925	0.0000936	2461.33

7.1.2 Case 2. constant integration limits

Table 4: Halton Estimate, case 2. constant integration limits.

Dimension	Estimate	Variance	MSE	Calculation_time
1	1.0388178	0.1631744	0.1960539	0.11
2	0.8660199	0.0249977	0.0294098	0.10
3	0.8636356	0.0052713	0.0052660	0.09
4	0.8117951	0.0022684	0.0023532	0.11
5	0.7745710	0.0007415	0.0007845	0.25
6	0.7345650	0.0001565	0.0001569	2.35
7	0.6973159	0.0000342	0.0000342	29.20

Table 5: Pseudo Estimate, case 2. constant integration limits.

Dimension	Estimate	Variance	MSE	Calculation time
1	0.6831348	0.0548461	0.4149422	0.19
2	0.8518203	0.0573332	0.0608644	0.11
3	0.8484347	0.0172693	0.0256645	0.10
4	0.7580878	0.0021198	0.0133734	0.11
5	0.7816118	0.0023076	0.0034614	0.17
6	0.7309600	0.0002375	0.0004035	1.36
7	0.7001593	0.0000522	0.0000544	16.29

Table 6: Sobol Estimate, case 2. constant integration limits.

Dimension	Estimate	Variance	MSE	Calculation_time
1	1.0349517	0.0533434	0.0509382	0.14
2	0.9344067	0.0226621	0.0239092	0.11
3	0.8618369	0.0093423	0.0101202	0.10
4	0.8099536	0.0014797	0.0015623	0.11
5	0.7739196	0.0002353	0.0002367	0.17
6	0.7396254	0.0000605	0.0000831	1.34
7	0.6901762	0.0000087	0.0000757	16.92

7.1.3 Case 3. full probability space

Table 7: Halton Estimate, case 3. Full probability space.

Dimension	Estimate	Variance	MSE	Calculation_time
1	0.8621009	0.1264516	0.1270950	0.13
2	0.9538129	0.7016322	0.7103188	0.14
3	1.0250009	0.0549266	0.0642448	0.13
4	0.9833606	0.0073344	0.0073542	0.14
5	0.9870817	0.0078179	0.0080508	0.30
6	0.9910669	0.0012247	0.0012948	2.59
7	0.9919506	0.0029864	0.0034505	29.65

Table 8: Pseudo Estimate, case 3. Full probability space.

Dimension	Estimate	Variance	MSE	Calculation_time
1	0.5795533	0.0717012	0.1184130	0.13
2	1.0255518	0.0683105	0.0900552	0.12
3	1.3044161	0.0314951	0.0626570	0.12
4	1.0974348	0.0253678	0.0922129	0.13
5	0.9538058	0.0049348	0.0053563	0.22
6	1.0115369	0.0020103	0.0020275	1.55
7	1.0016698	0.0006793	0.0007355	17.98

Table 9: Sobol Estimate, case 3. Full probability space.

Dimension	Estimate	Variance	MSE	Calculation_time
1	0.8718123	0.2456085	0.2250931	0.16
2	1.1443714	1.0006487	1.2125205	0.14
3	1.0189003	2.2137617	2.2912483	0.12
4	1.0018518	3.8848758	3.9039861	0.14
5	1.0067525	6.4675593	6.4744519	0.20
6	0.9929964	10.4380298	10.4393300	1.54
7	1.1898820	16.6599348	16.7033612	18.41

7.2 Computation time by case

Table 10: Computation time by case

Dimension	Halton			Sobol			Pseudo		
	Case 1	Case 2	Case 3	Case 1	Case 2	Case 3	Case 1	Case 2	Case 3
1	0.09	0.13	0.13	0.13	0.16	0.16	0.10	0.13	0.13
2	0.11	0.14	0.14	0.09	0.14	0.14	0.09	0.12	0.12
3	0.08	0.13	0.13	0.09	0.12	0.12	0.11	0.12	0.12
4	0.13	0.14	0.14	0.10	0.14	0.14	0.10	0.13	0.13
5	1.82	0.30	0.30	1.06	0.20	0.20	1.11	0.22	0.22
6	41.09	2.59	2.59	26.59	1.54	1.54	25.69	1.55	1.55
7	2845.72	29.65	29.65	2461.33	18.41	18.41	2569.55	17.98	17.98

7.3 R Code

7.3.1 MC-integration function in R

Listing 1: Simple MC-integration function in R

```
library(qrng)

mcIntNDim <- function(f, lower, upper, muVector, covMatrix,
                      RNG, nValues) {
  nDim <- length(upper)

  # Generation of random number
  if (RNG == "Sobol") {
    sobolMatrix <- sobol(nValues, nDim)
    randomVariables <- t(apply(sobolMatrix, 1,
        function(row) lower + (upper - lower) * row))
  }

  else if (RNG == "Halton") {
    haltonMatrix <- ghalton(nValues, nDim)
    randomVariables <- t(apply(haltonMatrix, 1,
        function(row) lower + (upper - lower) * row))
  }

  else{
    randomVariables <- t(apply(matrix(0, nValues, nDim), 1,
        function(row) runif(nDim, lower, upper))) #Pseudo
  }

  # Calculating expected value estimate
  sum <- apply(randomVariables, 1,
    function(row) f(row, mean = muVector, sigma = covMatrix))
  meanEstimate <- mean(sum)

  # Final estimate
  estimate <- prod(upper - lower) * meanEstimate
  return(estimate)
}
```