UNIVERSITY OF AMSTERDAM

STOCHASTIC SIMULATION ASSIGNMENT 1

Approximating the Area of the Mandelbrot Set Through Monte Carlo Methods

November 22, 2021

Students:
Steven Oud
(steven.oud2@student.uva.nl)
13688650

Malou Bastiaanse
(malou.bastiaanse2@student.uva.nl)
11426934

Lecturer:
Gábor Závodszky

Course:
Stochastic Simulation

Course code:
5284STSI6Y

Abstract

The Mandelbrot set is famous for its complex structures and dynamics, most prominently at its boundary. Hence, computing the area within the Mandelbrot set makes for an interesting challenge. Although different approaches have been used, the Monte Carlo integration method is an easily realizable method with relatively low computational cost. This work investigates various Monte Carlo methods for approximating the area of the Mandelbrot set. This is done by assessing the convergence and quality of the approximated area for different parameter values and sampling techniques. The investigated sampling techniques are pure random sampling (PRS), Latin hypercube sampling (LHS), orthogonal sampling (OS), and the randomized Quasi-Monte Carlo (RQMC) method. We found that increasing the sample size greatly decreased the variance of the area estimates, ultimately influencing the quality of the result. Although all sampling techniques had no statistically significant differences in the mean, statistical differences in variance were found between each combination of sampling techniques. The best performance, in terms of variance reduction and convergence rate, were the OS and RQMC methods, with the RQMC method performing slightly better.

Contents

1 Introduction

The Mandelbrot set is well known for producing aesthetically pleasing visuals consisting of complex structures by iterating over a seemingly simple function. It was first defined by Brooks and Matelski (1981), who also made the first visualization of the Mandelbrot set. Unfortunately, at the time, the computational resources required to fully explore the Mandelbrot set's fractal nature were not available. Nowadays, due to the consistent increase in computational power according to Moore's law, we can do much more advanced analyses and visualizations of the Mandelbrot set. One such analysis is the calculation of the area of the Mandelbrot set.

The exact value of the area of the Mandelbrot set remains an open question. Ewing and Schober (1992) give a solution for the Mandelbrot area that can be computed recursively, but no closed-form expression is known. Recursively calculating the coefficients for the solution described by Ewing and Schober (1992) converges very slowly — an order of 10^{1181} terms are required to get the first three digits. Hence, statistical methods have been suggested for numerically estimating the area of the Mandelbrot set. The most straightforward method is the pixel counting method where the Mandelbrot set is rendered on a high-resolution grid and the number of pixels that do not diverge to infinity is counted. The best-known estimate of the Mandelbrot so far was given by Thorsten (2012), who estimated the area of the Mandelbrot set to be approximately $1.5065918849 \pm 2.8 \cdot 10^{-9}$ with 95% confidence using the pixel counting method.

Another set of statistical methods used for approximating the area of the Mandelbrot set are Monte Carlo methods. Monte Carlo methods use random sampling and statistical principles to obtain useful numerical results (Kroese et al., 2014; Metropolis and Ulam, 1949). They are often used for approximating solutions for problems that are otherwise difficult or impossible to solve, thus making them an attractive candidate for estimating the area of the Mandelbrot set. Monte Carlo methods are used in fields such as engineering (Odriozola et al., 2019), computational biology (Ojeda et al., 2009), and artificial intelligence (Silver et al., 2016). While the computational cost of Monte Carlo methods is often less than exact methods, getting useful results can still be computationally expensive. In general, Monte Carlo methods require many samples in order to get a good approximation (Gentle, 2003), but due to the embarrassingly parallel nature of these methods, this is usually not a problem in practice (Cunha Jr et al., 2014).

This work investigates Monte Carlo methods for approximating the area of the Mandelbrot set. This investigation will be done with the following research objectives: (1) assess the effects of varying parameters on the convergence of the Monte Carlo integration method, (2) assess the performance of different sampling methods in terms of the convergence and quality of the area approximation, and finally, (3) introduce an additional Monte Carlo method and assess its performance against the best performing sampling technique (as established from objective 2). For the second research objective, we will be comparing the sampling techniques of pure random sampling, Latin hypercube sampling, and orthogonal sampling. For the third research objective, we will be considering the randomized Quasi-Monte Carlo method.

This report will be structured as follows: Firstly, the methodology including background information is provided for the Mandelbrot set and its visualization (Section 2.1), the Monte Carlo method and its application to the approximation of the Mandelbrot set area (Section 2.2), the three sampling techniques (Section 2.3), the additional Quasi-Monte Carlo method (Section 2.4) and, the statistical tools used for meaningful analysis of the results (Section 2.5). In the results section, a visualization and description of the Mandelbrot set are provided (Section 3.1), after which the

results are discussed in the respective order of the research objectives as discussed above (Section 3.2-3.4). Finally, a discussion is provided (Section 4), discussing the three main research objectives and possible future research.

2 Methods

2.1 Mandelbrot Set

The Mandelbrot set is the set of complex numbers c for which the absolute value of the function z_n remains bounded for all $n \geq 0$ (Brooks and Matelski, 1981), where z_n is:

$$z_0 = 0, \tag{1}$$

$$z_{n+1} = z_n^2 + c. \tag{2}$$

For any complex number c, one of two things will happen:

- The sequence blows up to infinity ($|z_n| > 2$)
- The sequence if bounded ($|z_n| \leq 2$)

Then, c belongs to the Mandelbrot set M if the second case ($|z_n| \leq 2$) holds:

$$M = \{c \in \mathbb{C} \mid |z_n| \leq 2\}. \tag{3}$$

We implement the calculation of the sequence z_n as shown in Algorithm 1. This algorithm iteratively calculates z_n as follows

$$z_n = z_0 + \underbrace{\left(z_0^2 + c\right) + \left(\left[z_0^2 + c\right]^2 + c\right) + \left(\left[\left(z_0^2 + c\right)^2 + c\right]^2 + c\right) + \cdots}_{n \text{ iterations}}. \tag{4}$$

The algorithm terminates as soon as $|z_j| > 2$, or if the maximum number of iterations n is reached. To improve the efficiency of the algorithm, we simulate complex number operations by using two real numbers:

$$c = x + yi, \tag{5}$$

where $x, y \in \mathbb{R}$. Then, $c^2 = x^2 - y^2 + 2xyi$, and we can rewrite Equation 2 as

$$x_{n+1} = \text{Re}(z_n^2 + c) = x_n^2 - y_n^2 + \text{Re}(c), \tag{6}$$

$$y_{n+1} = \text{Im}(z_n^2 + c) = 2x_n y_n + \text{Im}(c). \tag{7}$$

The calculation for y_{n+1} requires two multiplications in this form (Equation 7). Given that we need to calculate x_n^2 and y_n^2 for the calculation of x_{n+1}, and that $(x_n + y_n)^2 = x_n^2 + 2x_n y_n + y_n^2$, we can rewrite Equation 7 as

$$y_{n+1} = (x_n + y_n)^2 - x_n^2 - y_n^2 + \text{Im}(c), \tag{8}$$

reducing the total number of multiplications performed per iteration from five to three.

2.1.1 Visualizing the Mandelbrot Set

The Mandelbrot set can be visualized to explore its famously interesting and aesthetic properties. In this report, we use the escape-time algorithm to visualize the Mandelbrot set. This algorithm is based on the number of iterations necessary to determine if the Mandelbrot sequence blows up or stays bounded for a complex number c (Garcia et al., 2009; Sisson, 2007). A grid with width w and height h is created, and every (x, y) coordinate is scaled and converted to a complex number c_{xy} such that $\text{Re}(c_{\text{start}}) \leq \text{Re}(c_{xy}) \leq \text{Re}(c_{\text{end}})$ and $\text{Im}(c_{\text{start}}) \leq \text{Im}(c_{xy}) \leq \text{Im}(c_{\text{end}})$:

$$\begin{aligned} c_{xy} = \text{Re}(c_{\text{start}}) + \left(\frac{x}{w}\left[\text{Re}(c_{\text{end}}) - \text{Re}(c_{\text{start}})\right]\right) + \\ \text{Im}(c_{\text{start}}) + \left(\frac{y}{h}\left[\text{Im}(c_{\text{end}}) - \text{Im}(c_{\text{start}})\right]\right) i. \end{aligned} \tag{9}$$

Algorithm 1: Mandelbrot Sequence

Input: $c \in \mathbb{C}$, Max number of iterations $n \in \mathbb{N}$
Output: z_n, number of iterations j
$x \leftarrow \text{Re}(c)$;
$y \leftarrow \text{Im}(c)$;
$x_2 \leftarrow 0$;
$y_2 \leftarrow 0$;
$w \leftarrow 0$;
$j \leftarrow 0$;
while $x_2 + y_2 \leq 4$ ***and*** $j < n$ **do**
  $x \leftarrow x_2 - y_2 + \text{Re}(c)$;
  $y \leftarrow w - x_2 - y_2 + \text{Im}(c)$;
  $x_2 \leftarrow x^2$;
  $y_2 \leftarrow y^2$;
  $w \leftarrow (x + y)^2$;
  $j \leftarrow j + 1$;
end
$z_n \leftarrow x + yi$;
return (z_n, j);

Then, for every scaled complex number c_{xy}, we count the number of iterations that our algorithm described in Algorithm 1 takes to finish with a maximum number of iterations of n. A color is then assigned based on the number of iterations with a high number of iterations having a cold (blue) color, and a low number of iterations having a warm (red) color.

2.2 Monte Carlo Method

For approximating the area of the Mandelbrot set, this study makes use of a Monte Carlo integration approach. Under the term Monte Carlo falls a wide array of approaches, with the underlying concept of repetitive generation of random samples, to which information can be incurred using the law of large numbers or other processes of statistical inference (Kroese et al., 2014). Monte Carlo methods can have various applications such as incurring information about the data distribution, optimization of complicated functions, or, estimation of integrals (Kroese et al., 2014). This latter approach of estimating the integral of a function, and thus the area under the function, applies to this report.

We will be illustrating the Monte Carlo Integration Method with an example of calculating the integral of a function $f(x)$, as described in Ross (2013). To compute the area A in the interval $(0, 1)$ under $f(x)$, we find that:

$$A = \int_0^1 f(x)\, dx. \tag{10}$$

An alternative approach that approximates A can be done by using the set $U, U_1 ..., U_n$ of n independent uniformly distributed $(0, 1)$ random variables, such that:

$$A = E[f(U)], \tag{11}$$

with $f(U_1), \ldots, f(U_n)$ being independent and identically distributed random variables with mean A. Using the law of large numbers, which states that the mean of independent samples will convergence to the expected value with increasing sample size, it can be derived that:

$$\sum_{i=1}^{k} \frac{f(U_i)}{k} \rightarrow E[f(U)] = A \text{ as } k \rightarrow \infty. \tag{12}$$

Here lies the essence of the Monte Carlo integration method, where with an increasing number of simulations, the estimator of a specific statistic will converge towards its true value.

2.2.1 Monte Carlo Integration of the Mandelbrot Set Area

To apply the previously described Monte Carlo integration method for approximating the area of the Mandelbrot set, we first define a function $h(x)$ for whether a complex number c is in the Mandelbrot set:

$$h(c) = \begin{cases} 1, & \text{for } c \text{ where } |z_n| \leq 2 \\ 0, & \text{for } c \text{ where } |z_n| > 2 \end{cases} \tag{13}$$

where $|z_n|$ is calculated according to Equation 4, which depends on the complex number $c = x+yi$. To obtain a set of random samples for c, which will be referred to as set C, random variables are generated for both the real (x) and imaginary (y) component of c.

The random variables for x and y, $X_i, \ldots, X_n \in [-2, 1]$ and $Y_i, \ldots, Y_n \in [-1.12, 1.12]$ with sample size n, are independent and uniformly distributed within the area A_{rect}. Here A_{rect} is a rectangular area large enough to fully encompass the Mandelbrot set, but small enough to avoid generating unnecessary samples to increase computational efficiency. The boundaries of A_{rect}, and thus the intervals of x and y, are chosen such that the rectangular area fully contains the Mandelbrot set. It is common for uniformly distributed random variables (U_i) to be drawn from the interval $[0, 1)$, hence the samples can be scaled as follows:

$$(b - a)U_i + a, \tag{14}$$

where a and b are the lower and upper bounds of the intervals for the respective random variables. We use this method to transform uniform random variables from $[0, 1)$ to the intervals described above.

Although the samples within the set C are complex numbers that consist of two independent random variables for the real and imaginary part, within the following equations the random variables are denoted as C for the sake of simplicity. Thus, the area of the Mandelbrot set A_M can be defined as:

$$A_M = A_{\text{rect}} E[h(C)] \tag{15}$$

For a sample size of s, based on the law of large numbers, the expectation can be estimated as follows:

$$\sum_{i=1}^{s} \frac{h(C_i)}{s} \to E[h(C)] \text{ as } s \to \infty. \tag{16}$$

Thus the area of the Mandelbrot set can be approximated as:

$$A_M \approx A_{\text{rect}} \sum_{i=1}^{s} \frac{h(C_i)}{s} = A_{n,s}, \tag{17}$$

where n is the maximum numbers of iterations used for calculating z_n. From this derivation, it can be seen that the Monte Carlo approximation of A_M can easily be implemented by following a few easy steps as described in Algorithm 2.

As the Monte Carlo method relies on random variables, it is considered a stochastic process and thus the approximation of A_M will differ between simulations. Hence we need to run multiple simulations to have significant results within our studies. All experiments are at least repeated $r = 50$ times or more when specified. The appropriate statistical tools for evaluating stochastic simulations are specified in Section 2.5.

Algorithm 2: Monte Carlo approximation Mandelbrot Set Area

Input: Sample size $s \in \mathbb{N}$, Max number of iterations $n \in \mathbb{N}$
Output: The approximated Mandelbrot set area $A_{n,s}$
$X_{\min} \leftarrow -2$;
$X_{\max} \leftarrow 1$;
$Y_{\min} \leftarrow -1.12$;
$Y_{\min} \leftarrow 1.12$;
$A_{\text{rect}} \leftarrow (X_{\max} - X_{\min}) \cdot (Y_{\max} - Y_{\min})$; ▷ Area of rectangle around the Mandelbrot set
$k \leftarrow 0$; ▷ Number of points inside Mandelbrot set
for $j \leftarrow 0$ **to** s **do**
  $x \leftarrow U(X_{\min}, X_{\max})$;
  $y \leftarrow U(Y_{\min}, Y_{\max})$;
  $c \leftarrow x + yi$;
  if $|z_n| < 2$ **then** ▷ Calculate z_n for c using Algorithm 1
    $k \leftarrow k + 1$;
  end
end
$f \leftarrow \frac{k}{s}$; ▷ Fraction within Mandelbrot Set
$A_{n,s} \leftarrow A_{\text{rect}} \cdot f$;
return $A_{n,s}$;

2.3 Sampling Techniques

2.3.1 Pseudorandom Number Generation

Random numbers are at the heart of many statistical methods, including Monte Carlo methods (Gentle, 2003). This random sampling is usually done on deterministic machines: computers. As there is no straightforward method for generating true random numbers on a deterministic device[1], we often use pseudorandom number generators (PRNGs) that deterministically generate sequences of numbers whose properties approximate the properties of sequences of true random numbers.

For different use-cases, different qualities are desired from PRNGs. For Monte Carlo methods, Sawilowsky (2003) defines the following characteristics that a PRNG should attend to for ensuring a high-quality Monte Carlo simulation: (1) the PRNG has a long period, and (2) the PRNG produces values that pass tests for randomness. Characteristic (1) is about the length of the period at which values repeat in a pseudorandom sequence. MacLaren (1992) shows that using more random numbers than the 2/3 power of the period of a PRNG leads to excessive uniformity rather than a random sequence. Because Monte Carlo simulations often use a large number of random numbers (Gentle, 2003), one should choose a PRNG with a large enough period to ensure high-quality results. Characteristic (2) is a bit more ambiguous: what is a good test for randomness? This is a rather involved question and is beyond the scope of this report, but a well-known battery of tests called the Diehard tests is a popular measure for the quality of PRNGs (Marsaglia, 1995). In the case of Monte Carlo methods, an additional important characteristic of a PRNG is efficiency — if we are generating a large number of random numbers, a fast PRNG is highly desirable.

For our Monte Carlo simulations, we use the Mersenne Twister PRNG as described by Matsumoto and Nishimura (1998). This PRNG has attractive features for Monte Carlo applications: it has a very long period of $2^{19937} - 1$ ($\approx 4.3 \cdot 10^{6001}$), passes the Diehard tests, generates random numbers faster than most other PRNGs (Route, 2017, Section 5.2), and is freely and easily available. One disadvantage of the Mersenne Twister PRNG is that using multiple instances that differ only in seed value is not appropriate for Monte Carlo simulations that require independent random number generators (Matsumoto and Nishimura, 2000). As we use parallelized sampling to increase the efficiency of our Monte Carlo simulations, we use the method described by Haramoto,

[1] There exist methods for generating random numbers based on quantum phenomena which are inherently unpredictable (Brakerski et al., 2018).

Matsumoto, and L'Ecuyer (2008) and Haramoto, Matsumoto, Nishimura, et al. (2008) to advance the state of each subsequent PRNG as if 2^{128} random numbers have been generated. This splits the original random sequence so that each worker process can use distinct segments of the pseudorandom sequence.

2.3.2 Pure Random Sampling

As explained within Section 2.2, the Monte Carlo method relies on generating a large number of random variables, done for multiple simulation runs. The most trivial approach is pure random sampling (PRS), for which successively vectors of independent and uniformly distributed random variables $U_{i,j}(0,1)$ are created for each of the runs (Ross, 2013):

$$\begin{aligned} U_1 &= (U_{1,1}, U_{1,2} \ldots, U_{1,j}, \ldots, U_{1,s}) \\ U_2 &= (U_{2,1}, U_{2,2} \ldots, U_{2,j}, \ldots, U_{2,s}) \\ &\vdots \\ U_r &= (U_{r,1}, U_{r,2} \ldots, U_{r,j}, \ldots, U_{r,s}), \end{aligned}$$

where r is the total amount of simulation runs and s is the number of random variables to be generated per simulation. The strength of PRS is in its easy implementation, yet it has the drawback that when the sample size s is relatively low, the random variables are not distributed sufficiently within the given interval of the desired distribution. In this case, more numbers can end up in one region of the interval, resulting in a set of random variables with a different distribution than desired (Menčík, 2016). This can be seen in Figure 1, with more numbers seemingly ending up on the left side of the sampling area. Hence, this distribution of the random variables, and thus the value of the approximated estimator, can show large variation between simulation runs. For this reason, better alternatives to PRS have long been proposed and implemented, such as LHS (McKay et al., 1979) as discussed in the next section.

Within this study we need to sample both the real and imaginary value (x and y) of our complex number, hence the above sampling approach is done twice. As mentioned in Section 2.3.1, we use the Mersenne Twister PRNG throughout this work to generate uniformly distributed pseudorandom numbers. We use the implementation from the `numpy` Python package (Harris et al., 2020), which uses the standard MT19937 implementation with a word length of 32 bits. The sampling methods described in the subsequent sections will use the same PRNG for generating uniform pseudorandom numbers. As the samples are taken from a uniform distribution with an interval different from the standard $[0, 1)$ interval, they will be scaled according to Equation 14.

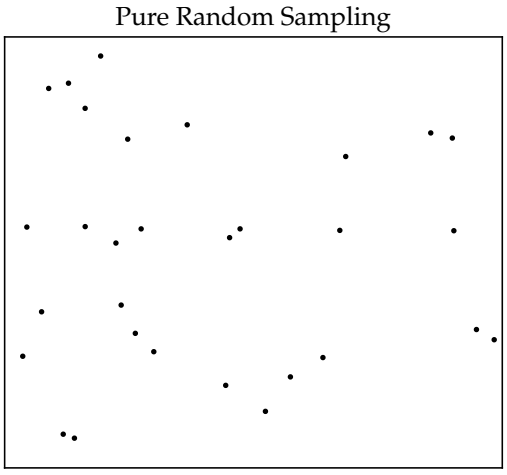


Figure 1: *Thirty points of a pure random sample in* $[0, 1)^2$.

2.3.3 Latin Hypercube Sampling

The Latin hypercube sampling (LHS) method was developed by McKay et al. (1979) to reduce the variance of the estimator values relatively to PRS. The LHS method relies on grid-based stratification of the generated random variables (Ross, 2013). Taking a single value U_j of each of the r runs, thus $U_{1,j} \dots U_{r,j}$, the aim is to stratify these values such that only a single value is in the interval $\left(\frac{k-1}{r}, \frac{k}{r}\right)$, where $k = 1, \dots, r$. If the order of k would be the same for each $U_{i,j}$, it would cause the unwanted result where each value within the s-vector of each run is within the same stratified interval. Hence, we generate s permutation vectors $p_1, \dots, p_s$ of the set $\{1, \dots, r\}$:

$$
\begin{aligned}
p_1 &= (p_{1,1}, p_{2,1}, \dots, p_{i,1}, \dots, p_{r,1}) \\
p_2 &= (p_{1,2}, p_{2,2}, \dots, p_{i,2}, \dots, p_{r,2}) \\
&\vdots \\
p_s &= (p_{1,s}, p_{2,s}, \dots, p_{i,s}, \dots, p_{r,s})
\end{aligned}
$$

Such that, the new random variables (U_i^*) can be described as:

$$
\begin{aligned}
U_1^* &= \left(\frac{U_{1,1} + p_{1,1} - 1}{r}, \frac{U_{1,2} + p_{1,2} - 1}{r}, \dots, \frac{U_{1,j} + p_{1,j} - 1}{r}, \dots, \frac{U_{1,s} + p_{1,s} - 1}{r} \right) \\
U_2^* &= \left(\frac{U_{2,1} + p_{2,1} - 1}{r}, \frac{U_{2,2} + p_{2,2} - 1}{r}, \dots, \frac{U_{2,j} + p_{2,j} - 1}{r}, \dots, \frac{U_{2,s} + p_{2,s} - 1}{r} \right) \\
&\vdots \\
U_r^* &= \left(\frac{U_{r,1} + p_{r,1} - 1}{r}, \frac{U_{r,2} + p_{r,2} - 1}{r}, \dots, \frac{U_{r,j} + p_{r,j} - 1}{r}, \dots, \frac{U_{r,s} + p_{r,s} - 1}{r} \right).
\end{aligned}
$$

This method of LHS results in less variance between the runs compared to pure random sampling, as the random values are less likely to be overabundant in certain regions when s is low, whilst still keeping the random variables independent and uniformly distributed (Ross, 2013). Figure 2 gives an example of the LHS method, in which the stratification of the random variables can be clearly seen. The subdivision of the interval would then be equal to the number of runs (r) that are performed. This spread of the random variables is often thought of as rooks on a chessboard, where no rook is allowed to be in each other's path.

As with the PRS method, the random variables for this study are sampled in parallel and scaled in accordance with Equation 14. For implementation of LHS the `LatinHypercube` class is used from the `scipy` Python library (Virtanen et al., 2020).

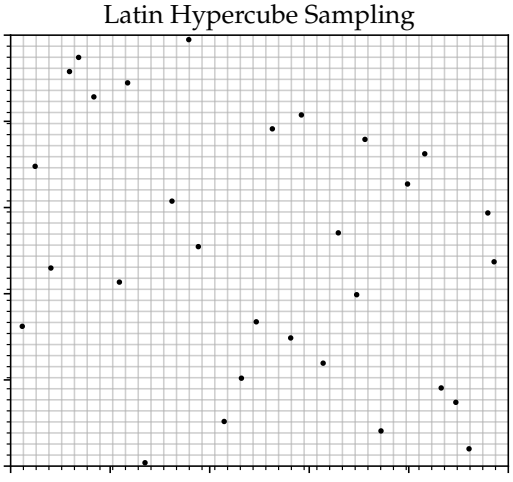


Figure 2: *Thirty points of a Latin Hypercube sample in* $[0, 1)^2$.

2.3.4 Orthogonal Sampling

While the LHS method is an improvement over pure random sampling for covering a region as uniformly as possible, an even better approach is by using orthogonal arrays to construct the Latin hypercubes as proposed by Tang (1993). This approach will be referred to as orthogonal sampling (OS). The main purpose of OS is to minimize the correlation between the sampling dimensions. Intuitively, this method can be thought of as dividing the sample space into equal subspaces and ensuring that the set of samples is a Latin hypercube whilst sampling from each subspace with the same frequency (Figure 4). Formally, this is implemented as follows. First, an orthogonal array $\text{OA}(n, m, s, r)$ is defined to be a $n \times m$ matrix with entries from the set $\{1, 2, \ldots, s\}$, of strength r, size n, with m constraints and s levels if each $n \times r$ submatrix contains all possible $1 \times r$ row vectors with the same frequency λ. In the context of our problem, n can be thought of as the number of samples, and m as the dimension of the sample. An orthogonal array can then be used to construct a Latin hypercube (Leary et al., 2003) by replacing every position with entry k for each column in $\text{OA}(n, m, s, r)$ with a random permutation of

$$\left[(k-1)\lambda s^{r-1} + 1, (k-1)\lambda s^{r-1} + 2, \ldots, (k-1)\lambda s^{r-1} + s^{r-1}\right] \tag{18}$$

for all $k \in \{1, 2, \ldots, s\}$. For our sampling purposes, we are interested in samples in a 2-dimensional space in which every subspace is sampled once, so we use $m = r = 2$, $\lambda = 1$, and $s = \lceil \sqrt{n} \rceil$. The permutation vector from Equation 18 can thus be simplified:

$$\left[(k-1)s + 1, (k-1)s + 2, \ldots, (k-1)s + s\right]. \tag{19}$$

For example, an orthogonal array $O = \text{OA}(9, 2, 3, 2)$ is transformed into a random Latin hypercube L as follows:

$$O = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 2 & 1 \\ 2 & 2 \\ 2 & 3 \\ 3 & 1 \\ 3 & 2 \\ 3 & 3 \end{bmatrix} \Rightarrow L = \begin{bmatrix} 1 & 1 \\ 3 & 6 \\ 2 & 9 \\ 6 & 3 \\ 4 & 4 \\ 5 & 7 \\ 8 & 2 \\ 7 & 5 \\ 9 & 8 \end{bmatrix} \tag{20}$$

We then subtract a uniformly distributed random number to each entry in L and normalize the matrix to create a $n \times m$ matrix S with random numbers in range $[0, 1)$:

$$s_{i,j} = \frac{1}{\max l_{i,j}} \left(l_{i,j} - U_{i,j}\right). \tag{21}$$

This range can then be scaled to any other range as described in Section 2.2.1. We implement our own orthogonal sampling method in Python using `numpy`.

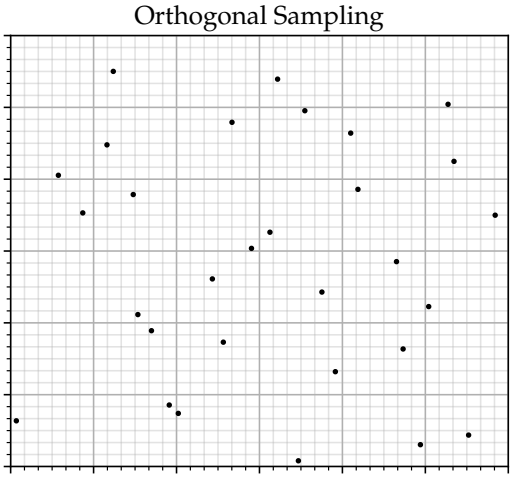


Figure 3: *Thirty points of an orthogonal sample in* $[0,1)^2$ *with orthogonal array* $\mathrm{OA}(30,2,6,2)$.

2.4 Quasi-Monte Carlo Method

The Quasi-Monte Carlo (QMC) method is a variation on the Monte Carlo method that uses low-discrepancy sequences instead of sequences of pseudorandom numbers (Niederreiter, 1992). These low-discrepancy sequences are also sometimes referred to as quasirandom numbers, as they are often used as a replacement for uniformly distributed random numbers. Low-discrepancy sequences are however fully deterministic, but they share certain properties of pseudorandom numbers. Using low-discrepancy sequences has shown to have an advantage over pseudorandom numbers in the rate of convergence (Asmussen and Glynn, 2007; Caflisch, 1998) as they ensure even coverage of the sample space. In the context of QMC, commonly used low-discrepancy sequences are the Sobol' sequence (Sobol', 1967), the Halton sequence (Halton, 1960), and the Faure sequence (Faure, 1982).

Given the deterministic nature of low-discrepancy sequences, it is difficult to estimate the accuracy of estimations made by QMC methods. In term, RQMC methods that add randomness to low-discrepancy sequences while retaining their low discrepancy have been developed. Randomizing the QMC points allows us to use the central limit theorem to estimate the error in RQMC methods. These RQMC methods can then be seen as a variance reduction technique like the other examples in this section (Dror et al., 2002). Methods for randomizing the Sobol and Halton sequence are described in Owen (1998) and Owen (2017) respectively. RQMC methods reduce the theoretical convergence rate compared to pure QMC methods. Furthermore, Tuffin (2004) showed that while the convergence rate of RQMC methods compared to standard Monte Carlo methods is not always better, the variance is smaller and the computation time is decreased when using RQMC methods. This makes RQMC methods preferable over QMC and standard Monte Carlo methods.

In our experiments, we use scrambled Sobol sequences provided by `scipy` in the `stats.qmc` submodule. We note that due to Sobol sequences' quadrature rule, they lose their balance properties if a number of samples that is not a power of 2 is used (Owen, 2020). Therefore, when using RQMC methods with the Sobol sequence, we set the number of samples used to a power of 2.

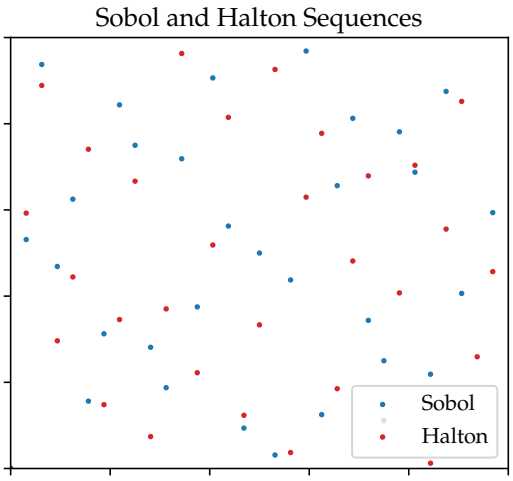


Figure 4: *32 points of the Sobol and Halton quasirandom sequences in* $[0, 1)^2$.

2.5 Statistical Analysis

Because of the stochastic processes underlying the Monte Carlo integration, multiple simulations are necessary to be run to generate statistically meaningful results. The appropriate statistics in the case of multiple simulations are the sample mean ($\overline{X}$) and the sample variance (S^2) (Ross, 2013). Although the Monte Carlo method also uses the term samples, meaning the number of random variables generated, the sample mean and variance refer in this report solely to the statistics over the collective of simulation runs. The sample mean and sample variance are defined as follows:

$$\overline{X} = \sum_{i=1}^{r} \frac{X_i}{r} \quad \text{and} \quad S^2 = \frac{\sum_{i=1}^{r}(X_i - \overline{X})^2}{r-1}, \tag{22}$$

where r is the number of runs. This is calculated using the `mean` and `var` functions of the `numpy` library, with parameter `ddof = 1` for the `var` function to ensure an unbiased estimator of the variance. For each of the sample means a confidence interval with a confidence level of $p = 95\%$ is provided. This confidence interval is calculated using the `stats.t.interval` function from `scipy`. We set the parameter values with `df = r - 1`, `loc` $= \overline{X}$ and `scale` is the standard error $\frac{\sqrt{S^2}}{\sqrt{r}}$, calculated with the function `stats.sem` from `scipy`.

This study will be comparing the different sampling techniques in terms of their convergence, as well as their difference in sample mean and sample variance. The comparison in the sample variance is performed using the F-test (Heumann et al., 2017), where the H_0-hypothesis is that the sample variance of two populations are equal ($S_x^2 = S_y^2$). For the F-test the p-value is calculated using the `stats.f.cdf` function of the `scipy` library, where $x = \max(S_x^2/S_y^2, S_y^2/S_x^2)$, `dfn` the sample size of the population with the largest variance - 1, and `dfd` the sample size of the population with the smallest variance - 1.

The comparison within the sample mean is done with the `stats.ttest_ind` function of the `scipy` library, where a two-sided test is performed for the H_0-hypothesis of equal means between two populations ($\overline{X} = \overline{Y}$), from which we will obtain a p-value. Following the results of the F-test, when the variances of the two populations are equal, a T-test is performed (Heumann et al., 2017), otherwise the Welch-test (Heumann et al., 2017) is performed by setting the parameter `equal_var = False`. In order to reject or accept the H_0-hypotheses, we have set an acceptance threshold of $\alpha = 0.05$, meaning the H_0-hypotheses will be rejected when $p < \alpha$. As the same test is performed numerous times (PRS vs LHS, PRS vs OS, LHS vs OS, best performing sampling technique vs RQMC), the p-values need to be adjusted accordingly, which will be done with a Bonferroni correction (Bell, 2018; Bonferroni, 1936). For this we make use of the `stats.multitest.multipletests` function of the `statsmodel` library.

For most of the visualization of the results the `matplotlib` library (Hunter, 2007) is used. Box plots and histograms are generated using the `boxplot` and `histplot` functions of the `seaborn` library (Waskom, 2021). The histograms will include a line of the kernel density estimate, by setting `kde = True`.

3 Results

3.1 Mandelbrot Set Visualization

We begin with exploring the Mandelbrot set by visualizing it using the escape-time algorithm described in Section 2.1.1 with a maximum number of iterations of $n = 100$. Figure 5 visualizes the Mandelbrot set at various levels of magnification. On the left, the full Mandelbrot set can be seen consisting of a large cardioid-shaped region in the middle with a circle to the left of the main cardioid. Most interestingly, the boundary of the Mandelbrot set is a fractal curve. That is, it retains the same general pattern no matter how far one magnifies it. We visualize two magnifications of a specific portion of the boundary of the Mandelbrot to highlight this fractal curve, but in principle this magnification can be continued indefinitely. On the right-most magnification, one can see the recursive nature of the fractal curve of the Mandelbrot set: it contains slightly different versions of itself. The Mandelbrot set is well known for creating visually pleasing images and has thus been extensively researched and visualized. We hereby fulfill our duty of adding another set of visualizations to this ever-growing collection of Mandelbrot visualizations.

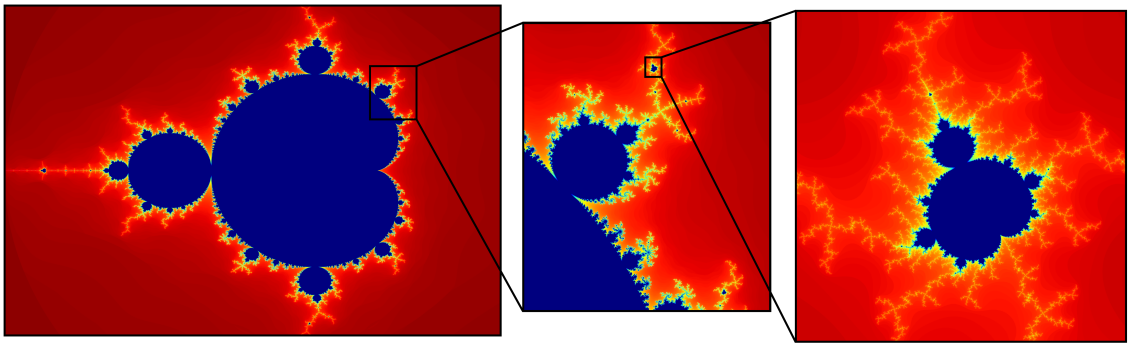


Figure 5: *Visualization of the Mandelbrot set using the escape-time algorithm. Left: Mandelbrot set from point $c_{start} = -2 - 1.12i$ to $c_{end} = 1 + 1.12i$. Middle: magnified point $c_{start} = -0.2 - 0.7i$ to $c_{end} = 0.5 - 0.35i$. Right: magnified point $c_{start} = -0.34775 - 0.655i$ to $c_{end} = 0.3675 - 0.635i$. These points are chosen to show the visually interesting part of the Mandelbrot set: the boundary between $|z_n| \leq 2$ and $|z_n| > 2$.*

3.2 Convergence of Mean Approximation Mandelbrot Set Area

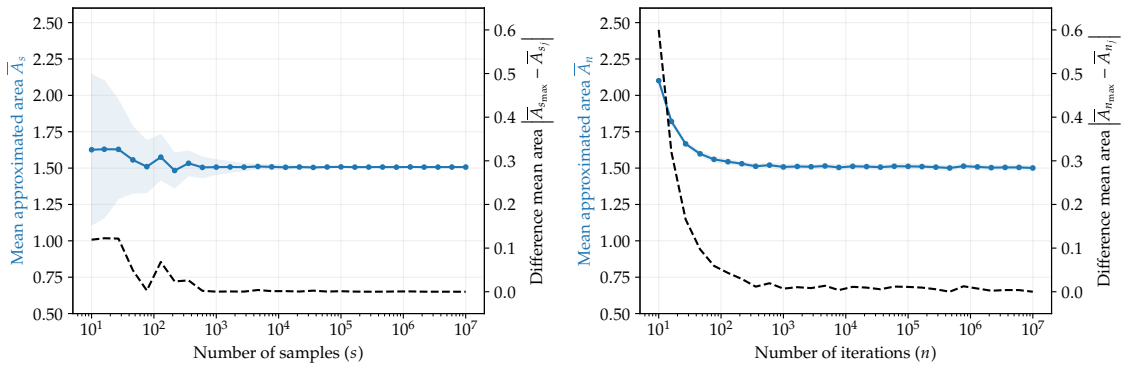
Within Figure 6 the convergence of the approximated area is investigated, for both varying values of the sample size s (fig. 6a) and the number of iterations n (fig. 6b). Whilst one parameter is varying, the other parameter is fixed. The data consists of 50 simulations runs, performed using the PRS method. The results are shown for the sample mean of the approximated area and the absolute difference in mean area, where the mean area with varying s or n ($\overline{A}_{s_j}$ or $\overline{A}_{n_j}$) is subtracted from the mean area with the largest sample size s or n of each experiment ($\overline{A}_{s_{\max}}$ or $\overline{A}_{n_{\max}}$).A confidence interval is given ($p = 95\%$) for the mean approximated area.

Firstly we will focus on Figure 6a, with a varying sample size and a fixed number of iterations ($n = 10^4$). The mean approximated area when s is small fluctuates towards the mean area of $\overline{A}_{s_{\max}}$, with approximations of the area falling both below and above the value of the value $\overline{A}_{s_{\max}}$. This can be seen back in the large confidence interval around the sample mean area, indicating large variations in the approximated area between simulations. Yet, this sizable confidence interval decreases together with an increasing s, converging towards $\overline{A}_{s_{\max}}$. The mean area seems to have

been mostly converged after s of 10^5. At the end of the experiment, the confidence interval has significantly reduced, with the sample mean area $\overline{A}_{s_{\max}}$ being $1.506873 \pm 4.19 \cdot 10^{-4}$.

Figure 6b has a fixed sample size ($s = 10^4$) with the number of iterations varying. Whereas the convergence within Figure 6a showed fluctuating behaviour, a steady decreasing trend can be observed within Figure 6a with only minor fluctuations towards the value of $\overline{A}_{n_{\max}}$. The confidence interval around the sample mean area shows little change and remains minimal during the entirety of the experiment, meaning little variation between the simulation runs. That the confidence interval is minimal for both small and large values of n, indicates that the confidence interval, and thus variation between simulation runs, is mostly a consequence of an ill-chosen sample size s. The mean approximated area seems to have been mostly converged after $n = 10^4$, showing only minor fluctuations around $\overline{A}_{n_{\max}}$. The mean approximated area of $\overline{A}_{n_{\max}}$ is $1.5121 \pm 1.55 \cdot 10^{-2}$, which has a larger confidence interval than $\overline{A}_{s_{\max}}$.

Using the results found within Figure 6, values for s and n can be chosen that are computationally efficient enough to fit within this study's time budget, whilst ensuring that convergence has been (mostly) reached. Hence all further results with different samplings methods will be performed with a fixed sample size of $s = 10^5$ and an iteration number of $n = 10^4$ unless noted otherwise.



(a) *Fixed number of iterations ($n = 10^4$), varying the sample size (s).*

(b) *Fixed sample size ($s = 10^4$), varying the number of iterations (n).*

Figure 6: *The convergence of both the mean approximated area for varying parameters of sample size s or iteration number n ($\overline{A}_{s_j}$ or $\overline{A}_{n_j}$) and the difference mean area are shown, where $\overline{A}_{s_{max}}$ or $\overline{A}_{n_{max}}$ are the mean values for the simulation run with the largest parameter value for s and n respectively. Results are shown with either the s or n value being fixed or varied. The sampling method is PRS, done for 50 simulation runs. The mean approximated area is shown with a confidence interval ($p = 95\%$).*

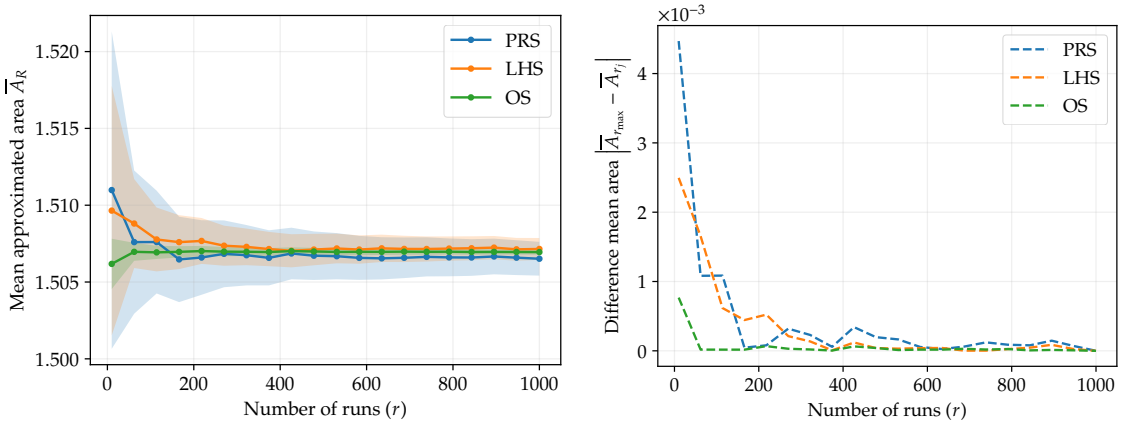
3.3 Comparison of Sampling Methods

In Figure 7 we compare the three main sampling techniques for varying numbers of simulation runs r. The sampling techniques in question are pure random sampling (PRS), Latin hypercube sampling (LHS), and orthogonal sampling (OS). The results are shown for both the mean approximated area and the absolute difference in the mean approximated area, where the mean approximated area for varying values of r ($\overline{A}_{r_j}$) is subtracted from the mean approximated area for the largest value of r ($\overline{A}_{r_{\max}}$). As mentioned within the previous section, a fixed sample size and iteration number were chosen with $s = 10^5$ and $n = 10^4$.

Firstly, an overall trend can be observed where with an increasing r, the confidence interval decreases, converging towards $\overline{A}_{r_{\max}}$, regardless of the sampling technique used. For all three sampling techniques, the mean approximated area seems to have been mostly converged when at minimum r is 500 runs. Focusing on the individual sampling techniques, PRS shows the largest confidence interval throughout the entirety of the experiment, with a relatively slow convergence towards $\overline{A}_{r_{\max}}$. The sampling technique of LHS shows similar behaviour, with slightly earlier convergence and with smaller confidence intervals. The most distinct sampling technique however

is OS with a noticeably quicker convergence and smaller confidence interval compared to LHS and PRS.

These differences between the sampling techniques are most prominent for smaller values of r, as both LHS and PRS take longer to convergence. This indicates that the choice for sampling method is the most important when the amount of simulation runs is relatively low, for example when only a few simulations can be performed due to computational constraints.



(a) *Mean approximated area.*

(b) *Absolute difference mean area.*

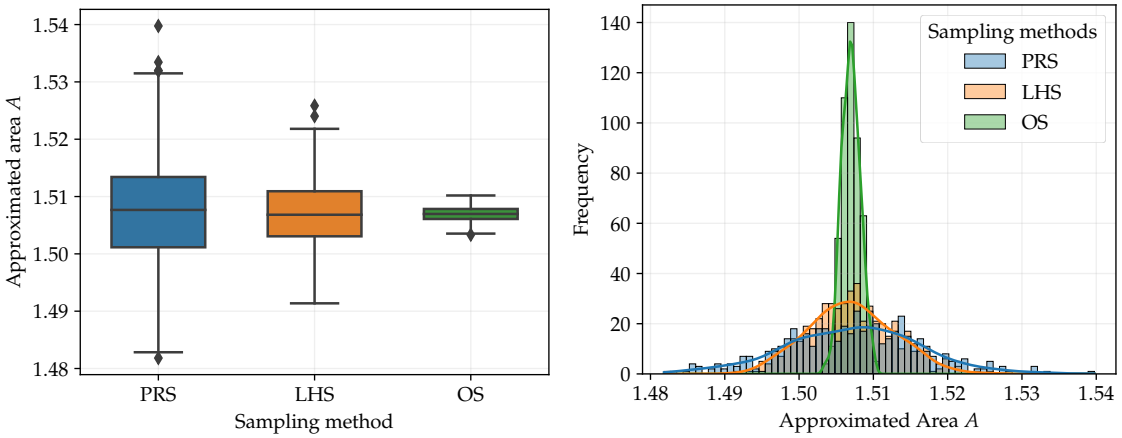
Figure 7: *The convergence of both the mean approximated area ($\overline{A}_{r_j}$) for varying number of runs r and the difference mean area are shown, where $\overline{A}_{r_{max}}$ is the mean area for the largest number of runs (r = 1000). The fixed parameters are a sample size of 10^5 and a iteration number of 10^4. The results are shown for three sampling techniques: PRS, LHS, and OS methods. The mean approximated area is shown with a confidence interval (p = 95%).*

Within this next section all parameters are fixed, with a sample size of $s = 10^5$, a number of iterations of $n = 10^4$, and a number of runs of $r = 500$. Similarly to s and n, the fixed value of r was chosen to be after the approximated area has mostly converged, whilst remaining within the computational budget of this study. Based on the results of these 500 simulations, Figures 8a and 8b show the differences between the three sampling techniques of PRS, LHS and OS via the means of a box plot and histogram plot respectively. These figures are accompanied with Table 1 in which the general statistical properties of the results are shown, together with the appropriate statistical test to identify significant differences between the sampling methods for both the sample mean (Welch's test) and sample variance (F-test). As these tests are both performed four times (this includes the tests within Section 3.4) all p-values were adjusted using the Bonferroni correction method.

Figure 8a shows the results for each sampling technique within a boxplot. For all sampling techniques, a few outliers were identified. All sampling techniques show similar median values of the approximated area. When observing the range of the data, meaning the difference between the minimum and maximum value, the largest range within the data can be observed with PRS, closely followed by the LHS method. Once again, the most distinct results can be seen for the OS method, with relatively small differences between the minimum and maximum value. Similar contrasts between the sampling techniques can be observed when analyzing the difference between the first and third quartile of the data.

Figure 8b provides a closer look at the data distribution of the approximated area values. The data distribution for each sampling technique seemingly follows a normal distribution with similar means. The largest differences between the samplings techniques can be observed in terms of variance. Once again, both PRS and LHS show large variation around the mean approximated area, with LHS only showing slightly lower variation. Similar to the results found in Figure 8a, the OS method stands out again, with very little variation within the data distribution and thus most data points being centered around the mean.

Table 1 displays the statistical properties of the data shown in Figure 8. The sample mean (confidence level of $p = 95\%$) of the approximated area for the sampling techniques of PRS, LHS and OS are $1.50721 \pm 1.59 \cdot 10^{-3}$, $1.50701 \pm 1.01 \cdot 10^{-3}$, and $1.506981 \pm 2.18 \cdot 10^{-4}$ respectively. The



(a) *Boxplots of approximated area values.*

(b) *Histograms of approximated area values.*

Figure 8: *Both plots represent the data distribution of the approximated area values for the three sampling techniques (PRS, LHS and OS). The data is based on 500 simulation runs. The fixed parameters are a sample size of* 10^5 *and a iteration number of* 10^4.

sample variances for the sampling techniques of PRS, LHS, and OS are $8.159 \cdot 10^{-5}$, $3.309 \cdot 10^{-5}$ and $1.54 \cdot 10^{-6}$ respectively, in decreasing order of variance size. Following the results of the F-test ($\alpha = 0.05$), all sampling techniques have statistically significant sample variances in comparison to each other. As the variances of the distributions differ, a Welch's test was performed ($\alpha = 0.05$), from which it could be clearly concluded there are are no statistically significant differences between the sample means of the sampling techniques. These findings are in accordance with the observations made from Figure 8.

Statistics			
	$\overline{A}$	Confidence radius (p = 95%)	S^2
PRS	1.50721	$1.59 \cdot 10^{-3}$	$8.16 \cdot 10^{-5}$
LHS	1.50701	$1.01 \cdot 10^{-3}$	$3.31 \cdot 10^{-5}$
OS	1.506981	$2.18 \cdot 10^{-4}$	$1.54 \cdot 10^{-6}$
F-Test			
	PRS	**LHS**	**OS**
PRS	x	$\ll 0.001$	$\ll 0.001$
LHS		x	$\ll 0.001$
OS			x
Welch's Test			
	PRS	**LHS**	**OS**
PRS	x	1	1
LHS		x	1
OS			x

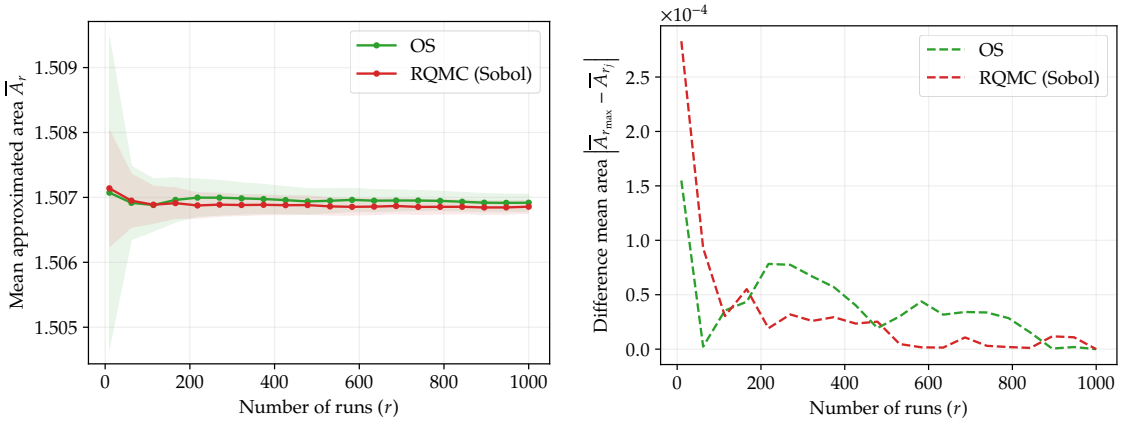
Table 1: *General statistical properties of the approximated area values for the three sampling techniques (PRS, LHS and OS). The data is based on 500 simulation runs. The fixed parameters are a sample size of* 10^5 *and a iteration number of* 10^4. *For hypothesis acceptance a threshold of* $\alpha = 0.05$ *was chosen.*

3.4 Randomized Quasi-Monte Carlo Method

As explained within the introduction, an additional research objective of this study is to compare the best performing method of the three main sampling techniques (PRS, LHS and OS) with the

randomized Quasi-Monte Carlo (RQMC). In this study, good performance is defined in terms of the convergence rate and in terms of variance in relation to the approximated area, where a quick convergence and low variance are optimal. Although a true value of the Mandelbrot set area is not known, these two measures of performance will ultimately impact the quality of the approximated area. From Section 3.3 it can be clearly established that the OS method has the best performance of the three methods evaluated, hence the following sections will compare the RQMC method with the OS method. Within the following results the fixed iteration number n will remain the same ($n = 10^4$), however, the fixed sample size s will be slightly changed. As mentioned within section 2.4, the RQMC method performs best using a sample size of a power 2 ($n = 2^m$), hence the value of s will be 2^{17} (= 131,072), which is closest to the previously used s value of 10^5.

Figure 9 shows a similar experiment to Figure 7, as in both the approximated mean area (fig. 9a) and the absolute difference mean area (fig. 9b) is shown for varying values of r. Here only the OS and RQMC method are compared and the plots are shown on a smaller scale. OS and RQMC show similar trajectories towards the value of $\overline{A}_{r_{\max}}$, with a slightly faster convergence for RQMC whilst both continue to show slight fluctuations of small magnitude ($< 10^{-4}$) around $\overline{A}_{r_{\max}}$ for the remaining values of r. Although the methods are quite similar, the RQMC method shows a slightly smaller confidence interval for all values of r. At the last data-point of the experiment ($r_{\max} = 1000$), the mean approximated area (confidence interval with $p = 95\%$) is $1.506918 \pm 1.31 \cdot 10^{-4}$ and $1.5068567 \pm 9.98 \cdot 10^{-5}$ for OS and RQMC respectively. Similarly to the results in Figure 7, generally speaking the confidence interval decreases with larger values of r.



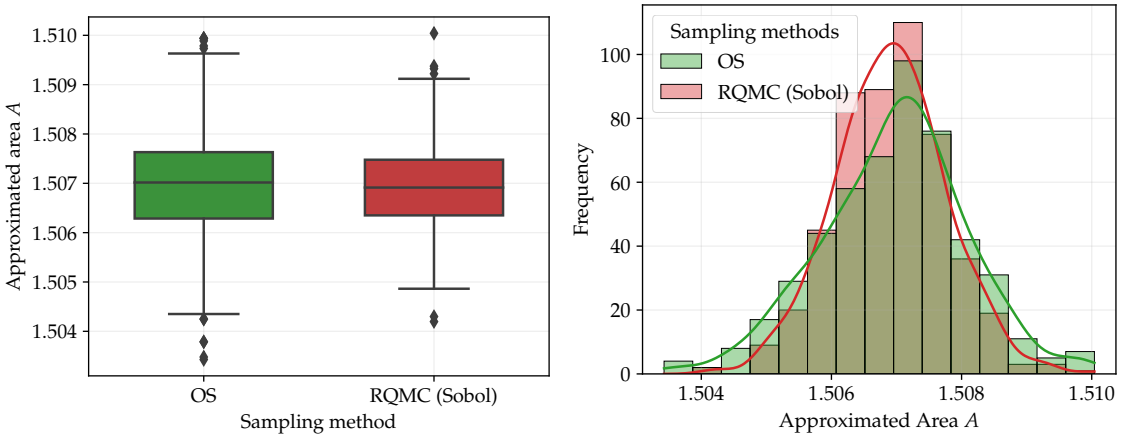
(a) *Mean approximated area.*

(b) *Absolute difference mean area.*

Figure 9: *The convergence of both the mean approximated area ($\overline{A}_{r_j}$) for varying number of runs r and the difference mean area are shown, where $\overline{A}_{r_{max}}$ is the mean area for the largest number of runs ($r_{max} = 1000$). The sample size is 2^{17} and the iteration number is 10^4. The results are shown for two sampling techniques: the OS and RQMC method. The mean approximated area is shown with a confidence interval ($p = 95\%$).*

To get a closer look at the data distribution of the approximated area values for OS and RQMC, the results for both sampling techniques are displayed again using box plots (Figure 10a) and histograms (Figure 10b). To remain consistent with Figure 8, 500 simulations were run. The fixed values for the number of iterations and sample size remain at 10^4 and 2^{17} respectively. Firstly focusing on Figure 10a, for both OS and RQMC a few outliers were identified. Again, similar medians between the two sampling techniques can be observed. Differences can be seen in terms of the range of the data points, with RQMC showing a smaller difference between the minimum and maximum value compared to OS. This contrast still holds in terms of the difference between the first and third quartiles of the dataset.

In Figure 10b histograms of the data distributions are shown. Both sampling techniques show similar means. However, the data distribution of the RQMC method has a higher peak around the mean value, meaning more data points are centered around the mean, indicating less variance within the data distribution. These findings are supported by the results shown in Table 2, in which general statistics of the two sampling techniques are shown. It should be noted that



(a) *Boxplots of approximated area values.*

(b) *Histograms of approximated area values.*

Figure 10: *Both plots represent the data distribution of the approximated area values for two sampling techniques (OS and RQMC). The data is based on 500 simulation runs. The fixed parameters are a sample size of 2^{17} and a iteration number of 10^4.*

similarly to before in Section 3.3, all p-values were adjusted using the Bonferroni correction. The sample means of OS and RQMC are $1.506963 \pm 1.95 \cdot 10^{-4}$ and $1.506909 \pm 1.50 \cdot 10^{-4}$ respectively. Using the RQMC method resulted in a relatively lower sample variance, with OS and RQMC having sample variances of $1.23 \cdot 10^{-6}$ and $7.28 \cdot 10^{-7}$ respectively. The results of the F-test show that this difference in sample variance is statistically significant. After performing Welch's test, no statistically significant difference between the sample means of OS and RQMC can be observed.

	Statistics		
	$\overline{A}$	Confidence radius (p = 95%)	S^2
OS	1.506963	$1.95 \cdot 10^{-4}$	$1.23 \cdot 10^{-6}$
RQMC	1.506909	$1.50 \cdot 10^{-4}$	$7.28 \cdot 10^{-7}$

	F-test	*Welch's Test*
OS vs RQMC	$\ll 0.001$	1

Table 2: *General statistical properties of the approximated area values for the two sampling techniques (OS and RQMC). The data is based on 500 simulation runs. The fixed parameters are a sample size of 2^{17} and a iteration number of 10^4. For hypothesis acceptance a threshold of $\alpha = 0.05$ was chosen.*

4 Discussion

This report investigated Monte Carlo methods for approximating the area of the Mandelbrot set. First, we assessed the effects of varying parameters on the converge of the Monte Carlo integration method. Second, we explored different sampling techniques and their impact on the convergence and quality of the area approximation. Finally, we consider an additional Monte Carlo method and compare its performance with the best performing sampling technique.

We first look at the convergence behaviour of our area approximation A towards A_M for different sample sizes s and number of iterations n. For both s and n, we find that as they increase, the mean approximated area converges to some value. We find that increasing the sample size decreases the variance, which in term is reflected in the confidence interval around the sample mean of A. Increasing the number of iterations was shown to not have a significant effect on the confidence interval. However, a sufficiently large number of iterations is still required to get a meaningful approximation. These results go to show that with a clever consideration of parameter values, meaning after the estimator has mostly converged, one can still have a moderately

accurate result with a limited computational budget. Hence, when using Monte Carlo methods an informed consideration between the balancing of computational time and desired accuracy is essential.

We then consider and compare three different sampling techniques: pure random sampling (PRS), Latin hypercube sampling (LHS), and orthogonal sampling (OS). For these comparisons, we fix the sample size and number of iterations in order to do fair comparisons. We find no significant difference in the sample mean between the different sampling methods. However, we find significant differences in the variance between PRS and LHS, PRS and OS, and LHS and OS. Most noticeably, OS has significantly less variance than both PRS and LHS. Additionally, OS converges significantly faster than PRS and LHS.

Finally, we compare the best sampling technique (OS) with the randomized Quasi-Monte Carlo (RQMC) method. The RQMC method makes use of randomized low-discrepancy sequences that ensure even coverage, which in theory improves convergence and reduces variance. We find no significant difference in sample mean between RQMC and OS, however the variance of RQMC is significantly lower than OS. RQMC and OS converge about equally fast, with RQMC converging slightly faster.

Our most accurate approximation of A_M is obtained through the RQMC method, giving $1.506909 \pm 1.50 \cdot 10^{-4}$ with a confidence of 95%. This approximation is still orders of magnitude less accurate than the estimate given by Thorsten (2012). However, the aim of this report is not to give the most accurate estimation of A_M. Rather, this report looks to assess and demonstrate various approaches towards improving the performance of Monte Carlo methods. More specifically, we look into different variance reduction techniques, which in term lead to an increase in convergence speed and quality of our results. This paper demonstrates how a seemingly daunting task such as approximating the area of the Mandelbrot set can be solved using Monte Carlo methods.

This work can be extended in different ways. First, more variance reduction techniques such as control variates can be implemented and compared. Second, the RQMC method can be further expanded by exploring different low-discrepancy sequences such as the Halton and the Faure sequences or by passing the generated points through a baker's transformation to potentially improve convergence, as described by Hickernell, 2002. Finally, experiments with larger sample sizes and number of iterations can be done to analyze the behaviour of the described methods at higher precision.

References

Asmussen, S., & Glynn, P. W. (2007). *Stochastic simulation: Algorithms and analysis* (Vol. 57). Springer Science & Business Media.

Bell, N. (2018). *Introduction to statistics*. Tritech Digital Media.

Bonferroni, C. (1936). Teoria statistica delle classi e calcolo delle probabilita. *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commericiali di Firenze*, *8*, 3–62. https://ci.nii.ac.jp/naid/20001561442/en/

Brakerski, Z., Christiano, P., Mahadev, U., Vazirani, U., & Vidick, T. (2018). A cryptographic test of quantumness and certifiable randomness from a single quantum device. *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, 320–331.

Brooks, R. M., & Matelski, J. P. (1981). The dynamics of 2-generator subgroups of psl(2, $\mathbb{C}$).

Caflisch, R. E. (1998). Monte carlo and quasi-monte carlo methods. *Acta numerica*, *7*, 1–49.

Cunha Jr, A., Nasser, R., Sampaio, R., Lopes, H., & Breitman, K. (2014). Uncertainty quantification through the monte carlo method in a cloud computing setting. *Computer Physics Communications*, *185*(5), 1355–1363.

Dror, M., L'Ecuyer, P., & Szidarovszky, F. (2002). *Modeling uncertainty: An examination of stochastic theory, methods, and applications* (Vol. 46). Springer Science & Business Media.

Ewing, J. H., & Schober, G. (1992). The area of the mandelbrot set. *Numerische Mathematik*, *61*(1), 59–72.

Faure, H. (1982). Discrépance de suites associées à un système de numération (en dimension s). *Acta arithmetica*, *41*(4), 337–351.

Garcia, F., Fernandez, A., Barrallo, J., & Martin, L. (2009). Coloring dynamical systems in the complex plane. *The University of the Basque Country, Plaza de Oñati*, *2*.

Gentle, J. E. (2003). *Random number generation and monte carlo methods* (Vol. 381). Springer.

Halton, J. H. (1960). On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numerische Mathematik*, *2*(1), 84–90.

Haramoto, H., Matsumoto, M., & L'Ecuyer, P. (2008). A fast jump ahead algorithm for linear recurrences in a polynomial space. *International Conference on Sequences and Their Applications*, 290–298.

Haramoto, H., Matsumoto, M., Nishimura, T., Panneton, F., & L'Ecuyer, P. (2008). Efficient jump ahead for $\mathbb{F}_2$-linear random number generators. *INFORMS Journal on Computing*, *20*(3), 385–390.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Heumann, C., Schomaker, M., & Shalabh. (2017). *Introduction to statistics and data analysis: With exercises, solutions and applications in r*. Springer International Publishing AG.

Hickernell, F. J. (2002). Obtaining o (n-2+$\in$) convergence for lattice quadrature rules. *Monte carlo and quasi-monte carlo methods 2000* (pp. 274–289). Springer.

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Kroese, D. P., Brereton, T. J., Taimre, T., & Botev, Z. I. (2014). Why the monte carlo method is so important today. *Wiley Interdisciplinary Reviews: Computational Statistics*, *6*, 386–392.

Leary, S., Bhaskar, A., & Keane, A. (2003). Optimal orthogonal-array-based latin hypercubes. *Journal of Applied Statistics*, *30*(5), 585–598.

MacLaren, N. (1992). A limit on the usable length of a pseudorandom sequence. *Journal of statistical computation and simulation*, *42*(1-2), 47–54.

Marsaglia, G. (1995). https://web.archive.org/web/20160125103112/http://stat.fsu.edu/pub/diehard/

Matsumoto, M., & Nishimura, T. (1998). Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, *8*(1), 3–30.

Matsumoto, M., & Nishimura, T. (2000). Dynamic creation of pseudorandom number generators. *Monte-carlo and quasi-monte carlo methods 1998* (pp. 56–69). Springer.

McKay, M. D., Beckman, R. J., & Conover, W. J. (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, *21*(2), 239–245. http://www.jstor.org/stable/1268522

Menčík, J. (2016). Latin hypercube sampling. In J. Mencik (Ed.), *Concise reliability for engineers*. IntechOpen. https://doi.org/10.5772/62370

Metropolis, N., & Ulam, S. (1949). The monte carlo method. *Journal of the American statistical association*, *44*(247), 335–341.

Niederreiter, H. (1992). *Random number generation and quasi-monte carlo methods*. SIAM.

Odriozola, M., Abraham, E., Lousada-Ferreira, M., Spanjers, H., & van Lier, J. B. (2019). Identification of the methanogenesis inhibition mechanism using comparative analysis of mathematical models. *Frontiers in bioengineering and biotechnology*, *7*, 93.

Ojeda, P., Garcia, M. E., Londoño, A., & Chen, N.-Y. (2009). Monte carlo simulations of proteins in cages: Influence of confinement on the stability of intermediate states. *Biophysical journal*, *96*(3), 1076–1082.

Owen, A. B. (1998). Scrambling sobol'and niederreiter–xing points. *Journal of complexity*, *14*(4), 466–489.

Owen, A. B. (2017). A randomized halton algorithm in r. *arXiv preprint arXiv:1706.02808*.

Owen, A. B. (2020). On dropping the first sobol'point. *arXiv preprint arXiv:2008.08051*.

Ross, S. M. (2013). *Simulation* (Fifth edition.). Academic Press.

Route, M. (2017). Radio-flaring ultracool dwarf population synthesis. *The Astrophysical Journal*, *845*(1), 66.

Sawilowsky, S. S. (2003). You think you've got trivials? *Journal of Modern Applied Statistical Methods*, *2*(1), 21.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *nature*, *529*(7587), 484–489.

Sisson, P. (2007). Fractal art using variations on escape time algorithms in the complex plane. *Journal of Mathematics and the Arts*, *1*(1), 41–45.

Sobol', I. M. (1967). On the distribution of points in a cube and the approximate evaluation of integrals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, *7*(4), 784–802.

Tang, B. (1993). Orthogonal array-based latin hypercubes. *Journal of the American statistical association*, *88*(424), 1392–1397.

Thorsten, F. (2012). Numerical estimation of the area of the mandelbrot set.

Tuffin, B. (2004). Randomization of quasi-monte carlo methods for error estimation: Survey and normal approximation.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2

Waskom, M. L. (2021). Seaborn: Statistical data visualization. *Journal of Open Source Software*, *6*(60), 3021. https://doi.org/10.21105/joss.03021