Estimating the area of Mandelbrot set using Monte Carlo Method

Meifang Li
13043390
meifang.li@student.uva.nl
16/11/2020

Yuhao Qian
13011456
yuhao.qian@student.uva.nl
16/11/2020

ABSTRACT

In this report, we implemented *Monte Carlo Method* to estimate the area of *Mandelbrot set*. To give a more accurate estimation of the area, we evaluated the impacts of iteration number and sample size respectively. The iteration number did not influence the area much when it exceeded a certain number. However, the results of using four different sampling methods(pure random sampling, Latin Hypercube sampling, Orthogonal sampling, and Antithetic variables) showed that the area converged significantly as sample size increased.

1 INTRODUCTION

The Mandelbrot set has been viewed as one of the most complex objects in mathematics and widely studied for a very long time. One of the possible ways to estimate the area of the Mandelbrot set is the Monte Carlo method, which is based on the hit-and-miss technique. The algorithm to generate random numbers, which is the initial number C before iterations, is critical throughout this report. The randomness of random numbers would affect the resulting quality.

Ewing and Schober [1] tried to find the upper bound for Mandelbrot set and pointed out the estimation using pixel counting is quite poor. The pixel counting scheme is limited because it needs a very fine pixel size. Because we are estimating the area of the Mandelbrot set, we would run the simulation multiple times and calculate the sample mean θ as our estimated data. Hence, we could implement some variance reduction techniques in the simulation process.

The accuracy of this estimated area relies heavily on the sample size and iteration number. In intuition, the larger sample size and iteration number are, the estimation is more accurate. However, the computing cost would be extremely expensive when these two parameters are very large. Referred to our limited computing power, we balanced sample size and iteration number and evaluated the impact of these two parameters using the control variable method. To improve the accuracy of the estimated area, we implemented four different sampling methods during the Monte Carlo method: random uniform sampling, Latin Hypercube sampling, Orthogonal sampling, and Antithetic variables. In general, we would like to evaluate the convergence behavior $A_{i,s} \rightarrow A_M$. Firstly we investigated convergence rate $|A_{j,s} - A_{i,s}|$ with fixed sample size s. For fixed iteration number i, we used three different sampling techniques to compare the resulting quality with pure random sampling.

2 THEORY AND METHOD

2.1 Mandelbrot Set

The Mandelbrot set is known as a famous fractal and is defined as a set of complex numbers C such that equation1 does not diverge to infinity after iterations. The initial Z_0 equals 0 and the absolute value of Z_n remains bounded. To be specific, the absolute value can not exceed 2 for C to be included in the Mandelbrot set M.

$$Z_{n+1} = Z_n^2 + C \tag{1}$$

$$c \in M \Longleftrightarrow |Z_n| \leq 2 \tag{2}$$

As mentioned before, pixel counting is widely-used in plotting Mandelbrot set. We used the same trick to plot a figure for the Mandelbrot set. Based on the estimated bound in Mitchell's article[3], we generated 1200*800 pixels in $[-2, 1] \times [-1, 1]$, each pixel representing an initial point in complex plane. Based on equation1, we built *mandelbrot-test* to compute the iteration number for each point before it escape from absolute value 2. Then we colored each pixel referring to the iteration number: the points which did not escape away after the maximum iterations were colored yellow.

2.2 Calculating area using Monte Carlo Method

Monte Carlo method is a widely-used method for the simulation of stochastic process and it could solve deterministic problems using repeated random sampling. The general process of Monte Carlo method is taking U as a uniform distribution above (0,1) and generating n independent u_i from this distribution at first; then evaluating the transformed integral function $h(y)$ at these u_i ; finally calculating the average of these samples: $\bar{h} = \frac{1}{n}\sum_i^n h(u_i)$. Due to the Strong Law of Large Numbers, the expected value is the limit of the average: $\sum_i^n \frac{1}{n} g(U_i) \longrightarrow E(g(U))$ as $n \longrightarrow \infty$.

We could calculate the area of the Mandelbrot Set using this Monte Carlo method. First, we should generate a relatively large number of initial points as C using certain sampling methods. Then, we should take these C into equation1 to decide whether they are still in the Mandelbrot Set after a certain number of iterations. Finally, we could obtain the estimated area by calculating the ratio of those being within the Mandelbrot Set out of the total number and multiplying the ratio by the whole area of sample space.

2.3 Convergence rate

The estimated area of the Mandelbrot set is dependent on the number of iteration and sample size. If we generate infinitely many sample points and infinitely large iteration numbers, we would obtain a more accurate estimation of the Mandelbrot set. However, in practice it is impossible; we could only generate finite sample points and limited iterations. We defined $|A_{j,s} - A_{i,s}|$ as our convergence rate, where sample size s is fixed and i is the maximum iteration number according to our computing limit. In our experiment, we set i as 1000, sample size s as 1000 and j as $[20, 40, 60, ..., 1000]$. For each pair (s, j), we ran the simulation 100 times and then calculated the sample mean. The convergence rate can be calculated afterward.

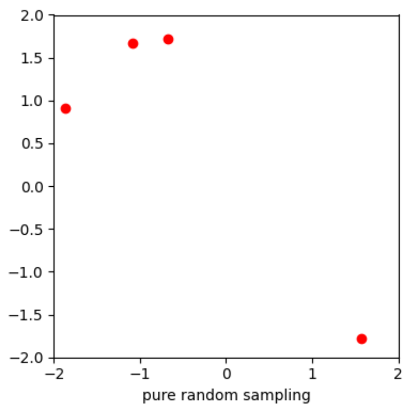

Figure 1: Random uniform sampling with four points.

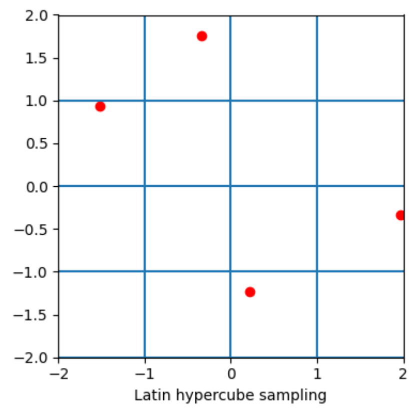

Figure 2: Latin Hypercube sampling with four points.

2.4 Sampling methods

In the stochastic simulation, the quality of the estimated Mandelbrot set area relies on how we generate random initial points, which can be drawn from different sampling methods as below.

2.4.1 Random uniform sampling. The probability density function of the uniform distribution is given by:

$$f(x) = \frac{1}{b-a} \tag{3}$$

anywhere within the interval(a, b), $a < b$ and zero elsewhere. In other words, x is uniformly distributed over (a,b) if it puts all its mass on that interval and it is equally likely to be "near" any point on that interval[4]. The example plot of random uniform sampling is shown in Figure1.

2.4.2 Latin Hypercube sampling. Latin hypercube sampling can be viewed as a compromise procedure that incorporates many of the desirable features of random sampling and stratified sampling[5]. When generating a sample of size n, the range of each variable is divided into n disjoint intervals of equal probability, and one value is selected at random from each interval. In two dimensions, the row and column where each sample point occupies are different from any other sample points, which means that all sample points spread more evenly than pure random sampling in the sample space. The example plot of Latin Hypercube sampling is shown in Figure2, where the sample space is divided by blue lines.

2.4.3 Orthogonal sampling. To a certain extent, orthogonal sampling is an extension and innovation of Latin Hypercube sampling. When generating a sample of size n,the sample space is divided into $M \times M (M = \sqrt{n})$ subspaces. All sample points should be sampled with Latin Hypercube sampling method but further restricted in a way that the density of sample points in each subspace is the same. Thus, orthogonal sampling is simply better space-filling designs, on average, than general Latin Hypercube sampling[2]. The example plot of Latin Hypercube sampling is shown in Figure3, where the sample space is divided by blue lines and subspaces are distinguished by dark blue lines.

2.4.4 Antithetic variables. The Antithetic variables could be viewed as a variance reduction technique rather than a new sampling

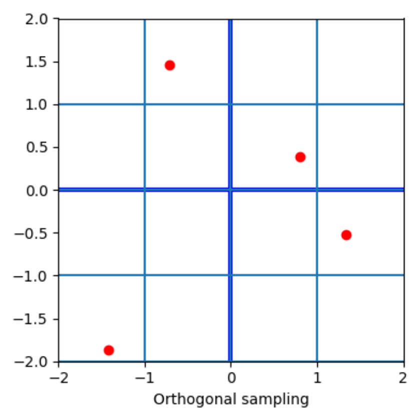

Figure 3: Orthogonal sampling with four points.

method. When simulating the function h that depends on m uniformly distributed random numbers from $U(0,1) : X = h(U_1, \cdots, U_m)$, if U_i is from $U(0,1)$, so is $1 - U_i$. We could generate a single set of m random numbers and compute the two samples as:

$$X_1 = h(U_1, \cdots, U_m) \tag{4}$$

$$X_2 = h(1 - U_1, \cdots, 1 - U_m) \tag{5}$$

where both X_1 and X_2 will follow the same distribution. Note that:

$$E(\frac{X_1 + X_2}{2}) = \frac{1}{2}[]E(X_1) + E(X_2)] = E(X) \tag{6}$$

$$Var(\frac{X_1 + X_2}{2}) = \frac{1}{4}[Var(X_1) + Var(X_2) + 2Cov(X_1, X_2)] \tag{7}$$

$$= \frac{1}{2}Var(X) + \frac{1}{2}Cov(X_1, X_2)$$

It is obvious that a negative co-variance can reduce variance. We also know that $Corr(X,Y) = \frac{Cov(X,Y)}{\sqrt{Var(X)Var(Y)}}$ and if correlation is negative, the co-variance must be negative. Since U_i is clearly negatively correlated to $1 - U_i$, X_1 and X_2 could also be negatively correlated and finally the variance is reduced.

When it comes to the Mandelbrot set, the real part of C (denoted as X_1) is randomly distributed in [-2,1] and the imaginary part of C(denoted as Y_1) is randomly distributed in [-1,1]. If we generate

X_1 using *numpy.random.uniform(-2,1)*, then $X_2 = -1 - X_1$ is also in $[-2, 1]$. Besides, $Cov(X_1, X_2) < 0$. Likewise we generate Y_1 using *numpy.random.uniform(-1,1)* and have $Y_2 = -Y_1$, again we have $Cov(Y_1, Y_2) < 0$. Then we calculate the mean area of $C = (X_1, Y_1)$ and $C = (X_2, Y_2)$ as the estimated area, which reduced the variance in the meantime as mentioned before.

2.5 Confidence interval

When we use simulation to approximate a value, it is always accompanied by confidence interval. If $X_1, \cdots, X_n$ are from the same distribution with mean θ and variance σ, the sample mean $\bar{X} = \sum_{i=1}^{n} X_i/n$ is an effective estimator of θ, but $\bar{X}$ does not exactly equal θ. Note that $E(\bar{X}) = \theta$ and $Var(\bar{X}) = \frac{\sigma^2}{n}$ and thus, from *Central Limit Theorem*, we know that for large n : $\frac{\bar{X}-\theta}{\frac{\sigma}{\sqrt{n}}}$ $N(0, 1)$. We also know that if we approximate $\sigma \longrightarrow S$, then it remains the case and we can get $\frac{\bar{X}-\theta}{\frac{S}{\sqrt{n}}}$ $N(0, 1)$. For any α, $0 < \alpha < 1$, let Z_α be such that $P\{Z > Z_\alpha\} = \alpha$, where Z is a standard normal random variable. We could obtain:

$$P\{\bar{X} - Z_{\alpha/2}\frac{S}{\sqrt{n}} < \theta < \bar{X} + Z_{\alpha/2}\frac{S}{\sqrt{n}}\} \approx 1 - \alpha \tag{8}$$

In other words, with probability $1 - \alpha$ the population mean θ will lie within the region $\bar{X} \pm Z_{\alpha/2}\frac{S}{\sqrt{n}}$. In most cases, a 95% confidence level is used[6]. Thus for $\alpha = 0.5$, we could get $Z_{.025} = 1.96$, then $\bar{X} \pm \frac{1.96S}{\sqrt{n}}$ given 95% confidence level.

When we take multiple simulations to calculate the area of the Mandelbrot set, we could get the sample mean and sample variance to calculate the confidence interval with 95% confidence level as mentioned above.

3 RESULTS AND DISCUSSION

3.1 Image of Mandelbrot Set

Figure4 and Figure5 showed two estimations of Mandelbrot set with different maximum iteration number. Figure5 revealed more details than Figure 4 because its maximum iteration number was much larger. That meant it was stricter to include a candidate point in the Mandelbrot set. It is intuitively clear that if we generate finer pixels and increase the maximum iteration number, we would obtain a more accurate Mandelbrot set.

3.2 Convergence rate when $A_{i,s} \longrightarrow A_M$

Firstly we used pure random sampling to investigate the convergence rate for increasing iteration number. As shown in Figure6, the convergence rate dropped significantly when the iteration number j was smaller than 180. The iteration number did not reduce the absolute error afterwards, as the convergence rate fluctuated when the sample size increased from 180 to 1000. We concluded that the accuracy could be improved by increasing the iteration number but the convergence rate is not significant enough when j is relatively large. We used this result in the following experiments and set the maximum iteration number as 600.

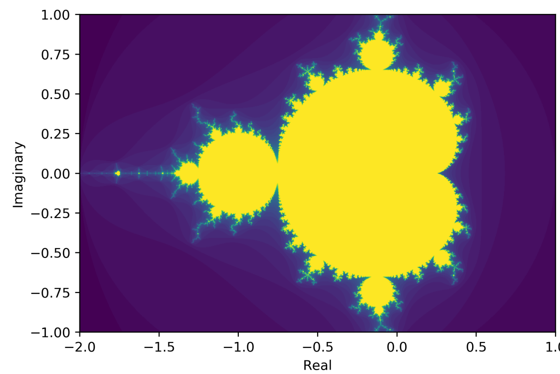

Figure 4: The Mandelbrot set in which iteration number is 50.

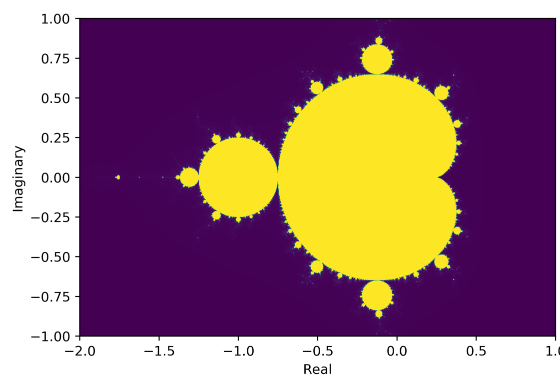

Figure 5: The Mandelbrot set in which iteration number is 1000.

3.3 Accuracy for different methods

In the following scenarios we have fixed maximum iteration number $i = 600$, we would like to observe the convergence behavior when increasing sample size s. Figure7,8,9 and 10 showed that all the methods have similar convergence behaviour: the estimated mean area fluctuated when sample size was small but reached a stable value when sample size further increased. We also noticed that the range of confidence interval shrank as the sample size grew. In all four methods, Orthogonal sampling converged fastest, followed by Latin Hypercube sampling, antithetic variables and the slowest one is pure random sampling. Detailed difference was shown in Table1. As we can see, all the methods could have a smaller standard deviation and confidence interval as sample size $s = 900, 3600, 8100$. Given the sample size $s = 8100$ iteration number $i = 600$, we had 95% of confidence to say that the estimated area of Mandelbrot set is $[1.5084963 \pm 0.00055591]$,$[1.51124444 \pm 0.00039544]$,$[1.50948148 \pm 0.00010819]$, $[1.5109963 \pm 0.00046684]$ using pure random sampling,

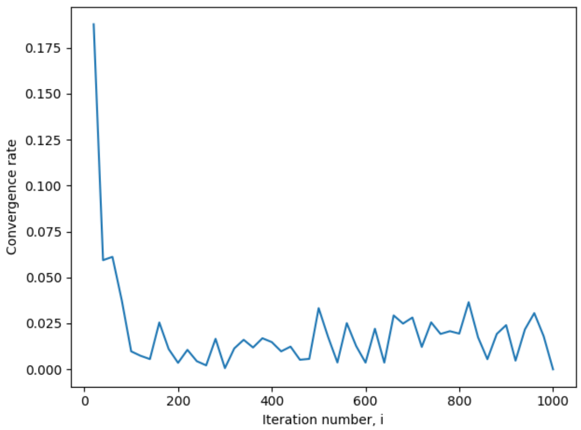

Figure 6: Convergence rate for $A_{j,s} \rightarrow A_{i,s}$. $s = 1000$,$i = 1000$,$j = [20, 40, 60, ..., 1000]$. The used sampling method is pure random sampling.

Latin Hypercube sampling, Orthogonal sampling and Antithetic variables respectively.

Figure11 compared the standard deviation between four methods. It revealed that Orthogonal sampling had the smallest standard deviation throughout the whole experiment, meanwhile pure random sampling had the largest, which also proved that antithetic variables could further improve the convergence when compared with pure random sampling. Still, all the methods had a similar trend: the standard deviation decreased as the sample size s grew, but its decreasing rate fell during the process. Given the fact that Orthogonal sampling had the smallest confidence interval and standard deviation, we could conclude that Orthogonal sampling has the best accuracy in estimating the area of the Mandelbrot set. Its sampling quality is better than the other three methods.

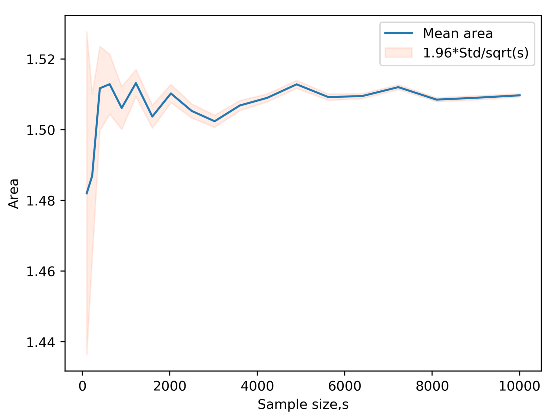

Figure 7: Accuracy for pure random sampling. $i = 600$, **simulation=100,** $s = [10^2, 15^2, 20^2, ..., 100^2]$**, the shadowed area is** 95% **confidence interval.**

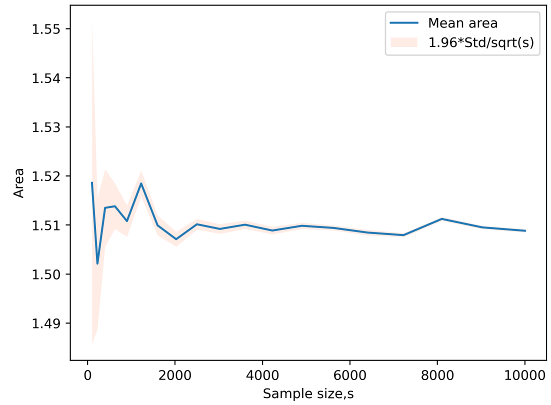

Figure 8: Accuracy for Latin Hypercube sampling. $i = 600$, **simulation=100,** $s = [10^2, 15^2, 20^2, ..., 100^2]$**, the shadowed area is** 95% **confidence interval.**

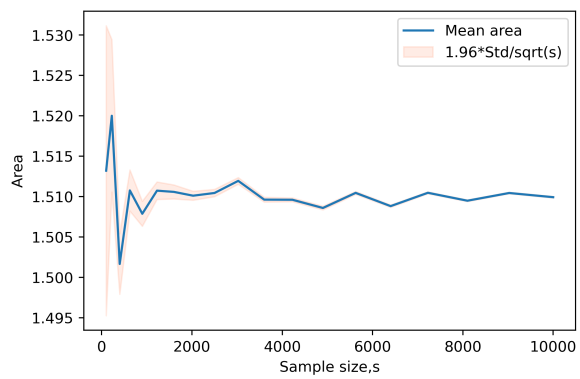

Figure 9: Accuracy for Orthogonal sampling. $i = 600$, **simulation=100,** $s = [10^2, 15^2, 20^2, ..., 100^2]$**, the shadowed area is** 95% **confidence interval.**

4 CONCLUSION

We used the Monte Carlo Method to estimate the area of the Mandelbrot set. For random number generating, we operated four sampling methods, namely pure random sampling, Latin Hypercube sampling, Orthogonal sampling and Antithetic variables. Our experiment results have shown that the iteration number i did not affect the convergence rate significantly after a certain number, which is 180 in our experiment. However, the sample size s had a great effect on the estimation. According to our results, Orthogonal sampling had the best quality and the estimated area is $[1.50948148 \pm 0.00010819]$ with 95% confidence level, given sample size $s = 8100$ and iteration number $i = 600$.

REFERENCES

[1] John H Ewing and Glenn Schober. 1992. The area of the Mandelbrot set. *Numer. Math.* 61, 1 (1992), 59–72.
[2] FJ Davis JC Helton. 2003. Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems. *Reliability Engineering & System*

Table 1: Comparison between different sampling methods.

Sample size	item	Random	LHS	Orthogonal	Antithetic
900	Mean	1.50613333	1.5108	1.50786667	1.5072
900	std	0.09257636	0.04878779	0.02367145	0.07071024
900	CI	0.00604832	0.00318747	0.00154654	0.00461974
3600	Mean	1.50683333	1.51006667	1.50961667	1.513925
3600	std	0.04356253	0.02619258	0.00916471	0.03662352
3600	CI	0.00142304	0.00085562	0.00029938	0.00119637
8100	Mean	1.5084963	1.51124444	1.50948148	1.5109963
8100	std	0.02552654	0.01815809	0.00496794	0.02143659
8100	CI	0.00055591	0.00039544	0.00010819	0.00046684

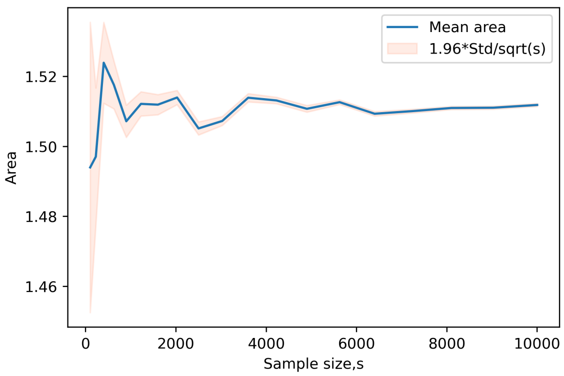

Figure 10: Accuracy for Antithetic variables. $i = 600$, **simulation=100,** $s = [10^2, 15^2, 20^2, ..., 100^2]$**, the shadowed area is** 95% **confidence interval.**

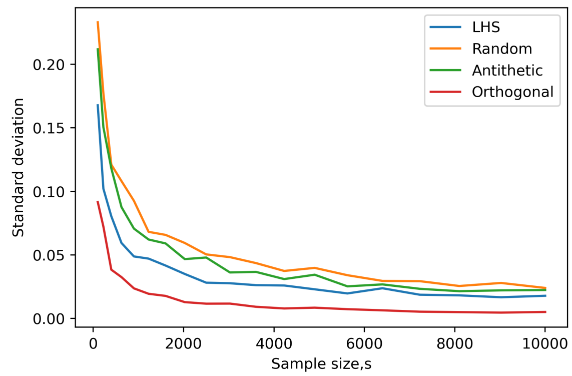

Figure 11: Comparison of standard deviation between four sampling methods. $i = 600$, **simulation=100,** $s = [10^2, 15^2, 20^2, ..., 100^2]$**.**

Safety 30 (2003), 23–69.

[3] Kerry Mitchell. 2001. A statistical investigation of the area of the mandelbrot set.

[4] Sheldon Ross. 2013. *Simulation.* Academic Press, Elsevier.

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

[6] Jerrold H Zar. 1999. *Biostatistical analysis.* Upper Saddle River, N.J. : Prentice Hall.