STOCHASTIC SIMULATION

# ASSIGNMENT 1 - COMPUTING THE AREA OF THE MANDELBROT SET

November 16, 2018

Student ID: 12297127 & 11037466
Jordan Earle & Nathalie van Sterkenburg

# Contents

**1 Abstract** **3**

**2 Introduction** **4**

**3 Background and Theory** **6**

**4 Experimental Methods** **8**

**5 Results and Discussion** **11**

**6 Conclusion** **22**

# 1 Abstract

In this experiment, a hit and miss Monte Carlo simulation will be analyzed to explore the Mandelbrot set area and the properties of its convergence. Two important variables in determining the area are the number of iterations $i$ and the number of samples $s$. Both will be varied to examine their influence. It is expected that increasing the number of samples both increased the accuracy and decreased the variance, while increasing the number of iterations increased the accuracy. Increasing $i$ and $s$ is expected to push the convergence of the area to the literature value of the area.

The simulation was conducted with three different methods of sampling points: random sampling, latin hypercube sampling and orthogonal sampling. Each type of sampling was simulated with $s = (100, 500, 1000, 5000, 10000)$ and $i = (3000, 3500, 4000, 4500, 5000, 5500)$ and random sampling was conducted for $s = (20000, 30000)$. For $i = 5500$ and $s = 30000$, the mean value of random sampling was $1.507 \pm 0.005$. For $i = 5500$ and $s = 10000$, the mean value of latin hypercube sampling was $1.5060 \pm 0.0066$ and the mean value of orthogonal sampling was $1.5096 \pm 0.0044$. It became apparent that both the accuracy and the variance converged quicker for orthogonal sampling than for latin hypercube sampling and quicker for latin hypercube sampling than random sampling. It should be noted there was a trade off in computational time, as the hypercube and orthogonal sampling took significantly longer than the random sampling.

All simulations were run for a second time with antithetic variables, which led to significantly lower variance in all simulations. The mean value of random sampling was $1.50780 \pm 0.0027$,for latin hypercube sampling it was $1.50705 \pm 0.0036$ and the mean value of orthogonal sampling was $1.50513 \pm 0.0021$. The use of antithetic variables causes a decreased variance with little additional computational power.

# 2 Introduction

When modeling systems, a deterministic approach can become invalid due to randomness in the system. One method for dealing with the randomness is to incorporate a probability model, in order to examine the system as a function of its random variables. Due to this randomness, multiple runs of the system are required to determine the analytically solutions to the problem.

In this experiment, the area of the Mandelbrot set is analyzed using a stochastic method known as hit and miss Monte Carlo integration. The Mandelbrot set can be seen in Figure 1.

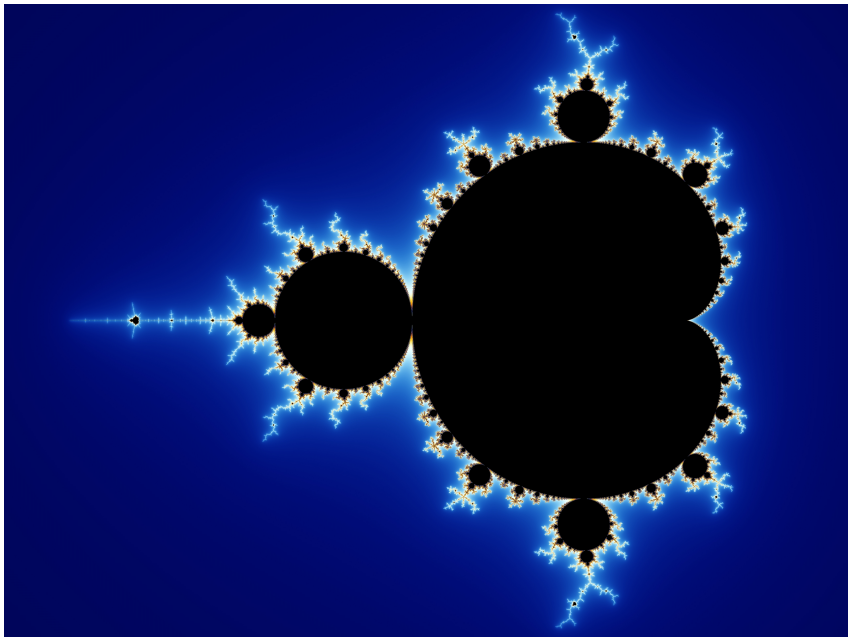


**Figure 1:** Mandelbrot Set . [1] [pp.4]

The method leverages the known search space area and that the probability that a random point will be in the set in that area is proportional to the area of the set, over the area of the search space. The experiments examines the importance of the number of points used during the integration and the number of iterations through the Mandelbrot equation. In order to speed up the evaluations on the system and reduce computational expense, certain properties of the Mandelbrot Set were utilized, including the symmetry of the set and known boundaries. The accepted value of the Mandelbrot set for this experiment comes from T. Forstemann [3], and is approximately 1.50659.

Once these were implemented a series of experiments were conducted to determine the convergence of the experimental area with an accepted area of the Mandelbrot set for increasing number of iterations and number of points/samples to the accepted area using

multiple sampling methods including pure random sampling, latin hypercube sampling and orthogonal sampling. In addition, The variance of the sampling methods was also examined. Once these experiments were conducted, In order to improve the convergence rate of the Monte Carlo approach, antithetic variables were applied to each of the sampling methods and the convergence of the area and the variance were examined and compared to those without antithetic variables.

# 3 Background and Theory

Before exploring the properties of the Mandelbrot set, it is useful to discuss what the Mandelbrot set is. The set consists of complex numbers $c = a + b\text{i}$ for which the function

$$f(z) = z^2 + c \tag{1}$$

does not diverge when $z = f(z-1)$ and $z$ starts out as 0. The Mandelbrot set was first thought of by Mandelbrot but first appears in Exploring the Mandelbrot set[2] in 1984. To determine whether a number belongs to the set or not, it is checked if $f(z)$ stays within a predetermined boundary for a predetermined number of steps. The predetermined boundary is located at

$$a^2 + b^2 = 4. \tag{2}$$

The Mandelbrot set can be visualized using the complex plane, where points that are in the set get a different color than points outside the set. This visualization leads to the well known Mandelbrot fractal, seen in Figure 1.

The predetermined number of steps, the number of iterations, can vary, but it has to be high enough so that most points that are not in the Mandelbrot set, cross the boundary within the number of iterations. Preferably, this would be the case for all points outside the set, but there's no way to guarantee if all the points not in the set have left by a certain step. The number of iterations should also be small enough to maintain an achievable run time of the simulation. In this experiment, the number of iterations was varied to determine an optimum amount for the simulation time available.

To determine the area of the Mandelbrot set in the complex plane, a hit and miss Monte Carlo simulation was utilized. In a hit and miss simulation, random points are generated within a certain interval and checked to determine if it belongs to the area of a function or not. The number of points that are created are called the number of samples $s$, where a higher $s$ generally results in a more accurate but also harder to compute system. So there are two variables that have to be high for good accuracy and low variance, but can't be too high because that's not computationally feasible. This means it is important too find a good balance between a high number of iterations and a high number of samples. If the number of iterations is higher, more points that don't belong to the set are excluded, making the result more accurate. But if the number of samples is higher, the variance is reduced. Both are desirable.

An important part of the Monte Carlo method is the generating random points, in this

case random complex numbers, to use in the simulation. There are multiple ways to go about generating these random numbers, but for the experiment an numpy library for random number generation was used. To sample the search space in this experiment, three methods of sampling were implemented and explored, random sampling, latin hypercube, and orthogonal sampling.

The first method is the purely random sampling. $a$ and $b$ are randomly generated within the ranges $[-2,1]$ and $[-1.5,1.5]$ respectively and such that they satisfy $a^2+b^2 \leq 4$. The second method is latin hypercube sampling. With this method, both the range of $a$ and the range of $b$ are divided into $s$ intervals of equal size. In each interval one random value is created and each value of $a$ is paired up with a random value of $b$ to generate $c$. This method guarantees that the points are spread out over the whole area in which you are sampling, which leads to a quicker convergence of the area when $s$ increases. The third method is orthogonal sampling. The ranges are divided into $s$ intervals, just like the latin hypercube sampling. Additionally, the sample area is divided into subareas, and instead of randomly pairing up an $a$ with a $b$, they are paired up so that there are an equal amount of points in each subarea. This method is an even better representation of the sample area as a whole than the latin hypercube sampling, making the area converge quicker.

In order to further increase the convergence rate and in order to decrease variance in the data, antithetic variables were implemented in each of the three systems, to see how effective they were in increasing the convergence and decreasing the variance. Antithetic variables take advantage of the variance relationship between variables having the same mean. The following equation shows the variance of the two variables:

$$Var(\frac{X_1+X_2}{2}) = \frac{1}{4}[Var(x_1)+Var(X_2)+2Cov(X_1,X_2)]. \tag{3}$$

From this equation, it can be seen that if the covariance between the first and second variable are negative, then there will be a reduction in variance for the system. As such it would be advantageous if $X_1$ and $X_2$ were negatively correlated. In order to do this, $X_1$ becomes a function of independent, random variables, $U$, and $X_2$ becomes a function of the random variable from $X_1$, $1-U$. This then causes the clear relationship that $X_1$ and $X_2$.

In order to determine how much the addition of the antithetic variable reduces the variance, the equation above were solved and compared to the variance if the system was unrelated. This was done in Simulation [4] and it was determined that the addition of antithetic variables should reduce the variance of the estimator by a factor of approximately 3.

# 4 Experimental Methods

In order to determine the area of the Mandelbrot set, a hit and miss Monte Carlo method was utilized to determine the probability of being in the set within the search space. Knowing this probability, and the size of the search space, and the relationship between the probability and the area:

$$\frac{P_{InSet}}{P_{Total}} = \frac{A_{InSet}}{A_{Total}} \tag{4}$$

the area of the Mandelbrot could be determined. From the simulation, the probability of the point being in a set could be determined, the total probability of the point being in the area is 1, and the area of the space being searched is known, and therefor the area can be calculated for each simulation. The simulation consisted of retrieving a random point and checking if the point was still in the Mandelbrot set by computing $f(z)$ for $i$ iterations.

In order to select the points to be searched, a random variable between 0-1 was created for the real and complex part of each point, and the distributed according to one of the sampling method being used, which are described below.

For random sampling, $a$ was taken randomly from the interval $[-2,1]$ and $b$ from $[-1.5,1.5]$ for each sample. It was then tested if these values met the requirement $a^2+b^2 \leq 4$. If the point was within that range, a new point was generated. Otherwise the point was put through the Mandelbrot simulation. Only these points were considered as samples, meaning that the sample area was the square in which the values were created minus the area that was excluded because of the requirement of being within a circle of radius 2. This area was 8.3765.

For latin hypercube sampling, two lists of random numbers were created, one for a-values and one for b-values. Each list contained a sample in each interval. A-values were created in the range $[-2,1]$ and b-values in $[-1.5,1.5]$, meaning the interval size was $3/s$. Because these lists were still in order from smallest to largest, both lists were shuffled randomly and then run through the Mandelbrot simulation. The Mandelbrot simulation then went through both lists to create random points, matching up the first entry in each list, the second entry in each list, et cetera. The sample area with this method was the a square of size 9.

For orthogonal sampling, $m$ lists were created for both $a$ and $b$, where $m^2$ is the number of sub spaces. Instead of putting all a-values in a single list, each list had $s/m$ entries, with the first list containing all the lowest values and the last list containing all the highest values. The same was done for the b-values. The range of values was the same as for latin hypercube sampling, meaning the sampling area was 9 as well. As a next step, the entries in each list were

shuffled. So the values were still divided by subspace, but the values within each subspace were shuffled. Then all a-lists were merged to a single a-list and all b-lists were merged to a single b-list. The b-lists were merged by first adding all values of the first list to the merge list, then adding all values of the second list, et cetera. The a-lists were merged by adding the first $1m$ part of the first list to the merge list, then adding the the first $1m$ part of the second list, et cetera. When part of each list had been added, the second $1m$ part of the first list was added, et cetera. This ensured that when pairing up the $a$'s and $b$'s, a part of each a-list was paired up with a part of each b-list, meaning each subarea had the same amount of points in it.

Then seeing how the system reacted, antithetic variables were added into further reduce the variance and speed up the convergence. Different approaches were needed in order to implement this into the different sampling systems seen above. I pure random sampling, for each sample, the random variable was generated between 0-1 ($U$) for both the real and complex part of the system, and then an antithetic variable was generated from $1-U$. These random and antithetic variables were then scaled to be in the range described previous for the complex and real planes respectively. These were then run through the simulation, and the outcome recorded. Since the search space was excluding values outside of the 4 radius, the sets of antithetic variable and random variable which contained a point outside this area were also excluded, reducing the area searched further. This was possible due to the specific form of the Mandelbrot set, and would possibly need to be modified if another system was examined. This reduced the search space to 7.753.

To implement antithetic variables into the latin hypercube and orthogonal sampling, a similar approach was implanted. Within the sample space of $a$ and $b$, $s$ intervals of equal size are created. In each interval one random value is created for both the real and the imaginary domains. A random variable is generated between 0-1 ($U$) for both the real and complex part of the system, and then an antithetic variable was generated from $1-U$. The random variables are then scaled to a value within the current interval for both the real and the imaginary. These values were then put back into the pipeline for the hypercube or orthogonal sampling as if they were both generated randomly. The search space for these sampling methods remained 9.

In this experiment, the search space that was used was $[-2,1]$ for $a$ and $[-1.5,1.5]$ for $b$. The number of iterations used was varied between 3000 and 5500, with steps of 500 in between. The number of samples was varied as well, for random sampling the list of sample sizes consisted of 1000, 5000, 10,000, 20,000, 30,000 and for latin hypercube and orthogonal sampling the list consisted of 100, 500, 1000, 5000, 10,000. Each of these combinations was run 100 times, in order to give a large sample size for the experiment.

Using these data sets the area of the Mandelbrot set was estimated, along with the confidence interval of that estimation. Furthermore, the properties simulation as the number of iteration and the number of sample points were increased were examined in addition to the properties of the hit and miss method.

# 5 Results and Discussion

The experiment was initially run with a random sampling method, using a sample size of 100000 and 2500 iterations to test the simulation. This was done a number of times to confirm the accuracy of the system before continuing. Below is an example of the set produced by the simulation.

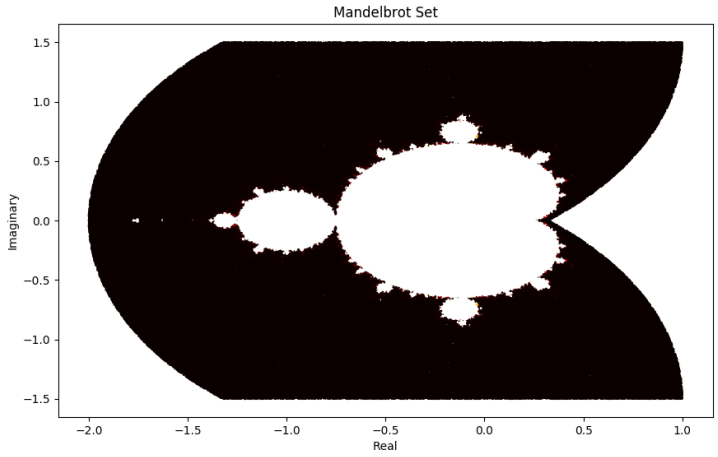



**Figure 2:** Mandelbrot Set generated from simulation at 100000 samples and 2500 iterations.

After the simulation was verified and the area was validated, a range of sample sizes(100, 500, 1000, 5000, 10000, 20000, 30000) and iteration sizes (3000 to 6000 with a step size of 500) were each simulated 100 times and the results analyzed. The number of runs of the simulations is fixed for each part of the experiment at 100 simulations.

It is important to note that when using small numbers of iterations, the number of iterations is very important to the set, as points that would normally be excluded are not. In order to counter this, the number of iterations were kept high in this experiment, but because of this, the effect of small numbers of iterations is not seen. At large numbers of iterations, fewer points are excluded per iteration. To ensure that the simulation followed the expected normal distribution, resultant areas were examined and can be seen in Figure 3.

The area of the Mandelbrot set was calculated from each of these runs, and compared to the actual value. The mean found with the 30000 samples and 55000 iterations was 1.50736 with a variance of 0.01692. Knowing this the actual value of the mean falls within the range $1.50736 \pm 0.005$ with a 99% confidence interval. A heatmap was utilized to show the variance and the difference in mean values of each combination of iteration and sample size. As seen in Figure 4 the simulation mean approaches the actual mean as sample size and the number

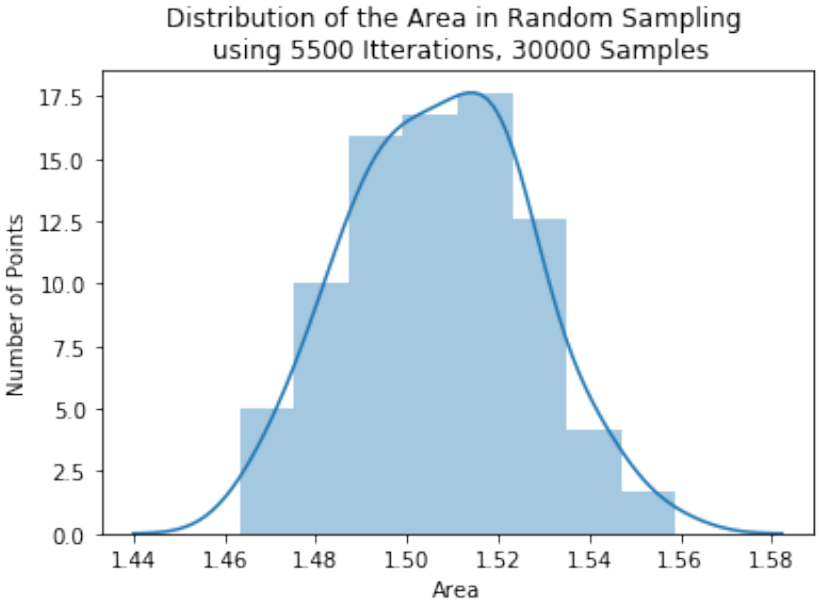
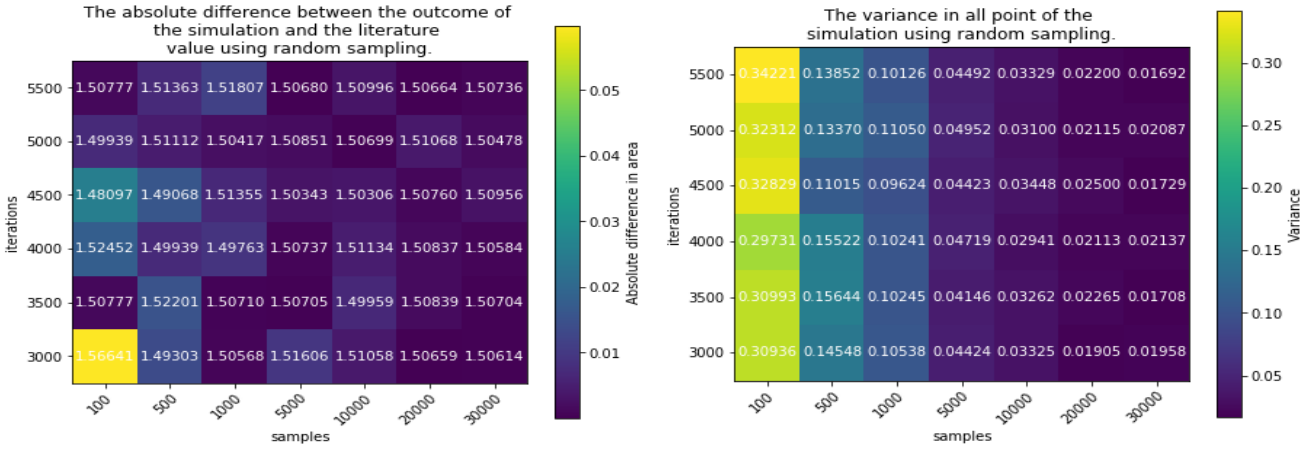



**Figure 3:** Distribution of samples from random sampling of the Mandelbrot simulation with 30000 samples and 5500 iterations.

of iterations increases, and the variance decreases as the number of samples increases.




**(a)** **(b)**

**Figure 4:** Heatmaps from random sampling of the Mandelbrot simulation with 30000 samples and 5500 iterations.

Once the random sampling was analyzed, latin hypercube sampling was implemented in the simulation over a similar range. Before the experiment on the samples was run, the process was implemented and checked with the same tests as the random sampling. The Mandelbrot set was observed, and the distribution of areas were graphed. These can be seen

in Figure 5.

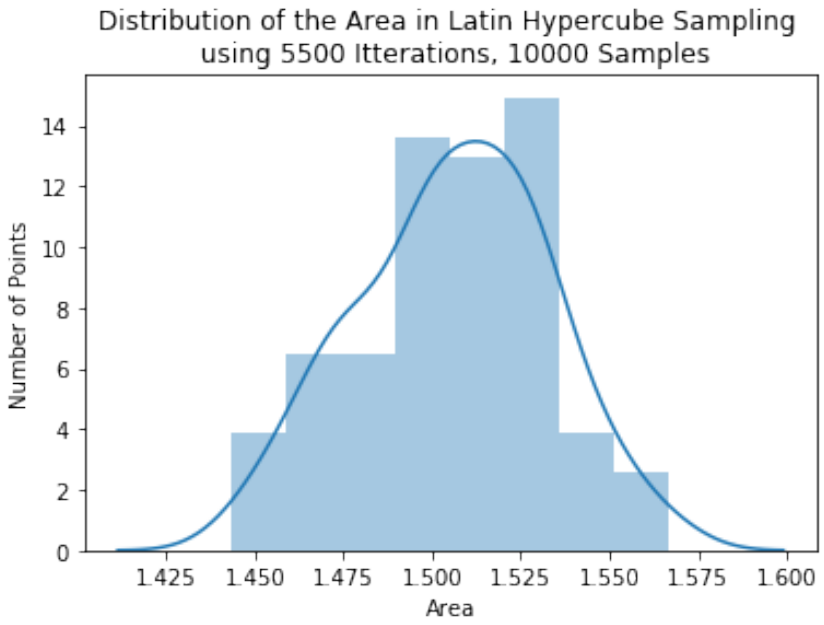
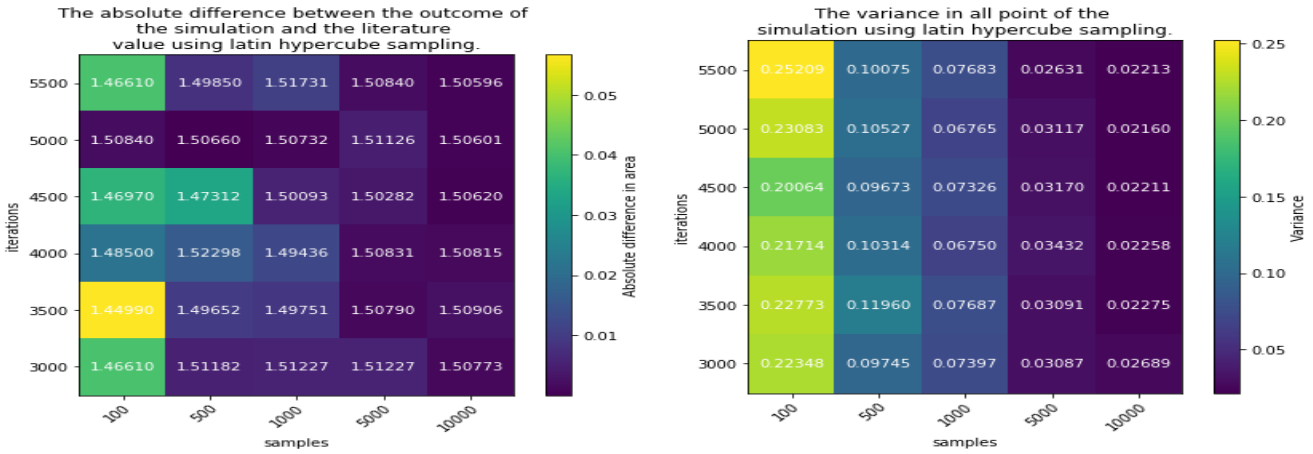



**Figure 5:** Distribution of samples from Latin Hypercube sampling of the Mandelbrot simulation with 10000 samples and 5500 iterations.

The hypercube performed as expected and the experiment was run on this simulation. Due to computational expense, the larger set sizes of 20000 and 30000 were not able to be completed due to the time it took to complete the simulation and problems with the computers they were being run on. With the data available from the samples sizes (100, 500, 1000, 5000, 10000) and the same number of iterations as the random simulation. The area of the Mandelbrot set was calculated from each of these runs, and compared to the actual value as with the random sampling. The mean found with 10000 samples and 55000 iterations was 1.50596 with a variance of 0.02213. Knowing this the actual value of the mean falls within the range $1.50596 \pm 0.0066$ with a 99% confidence interval. A heat map showing the variance and the difference in mean values of each combination of iteration and sample size can be seen in Figure 6. Once again it can be observed that the simulation mean approaches the actual mean as the number of iterations and sample size increases, and the variance decreases as the number of samples increases.

Once the system had been analyzed for latin hypercube, the orthoginal sampling conducted for the same range as and the same tests as the random and latin hypercube sampling. The Mandelbrot set was observed, and the distribution of areas were graphed. The distribution can be seen in Figure 7.




**(a)** **(b)**

**Figure 6:** heatmaps from Latin Hypercube sampling of the Mandelbrot simulation with 10000 samples and 5500 iterations.

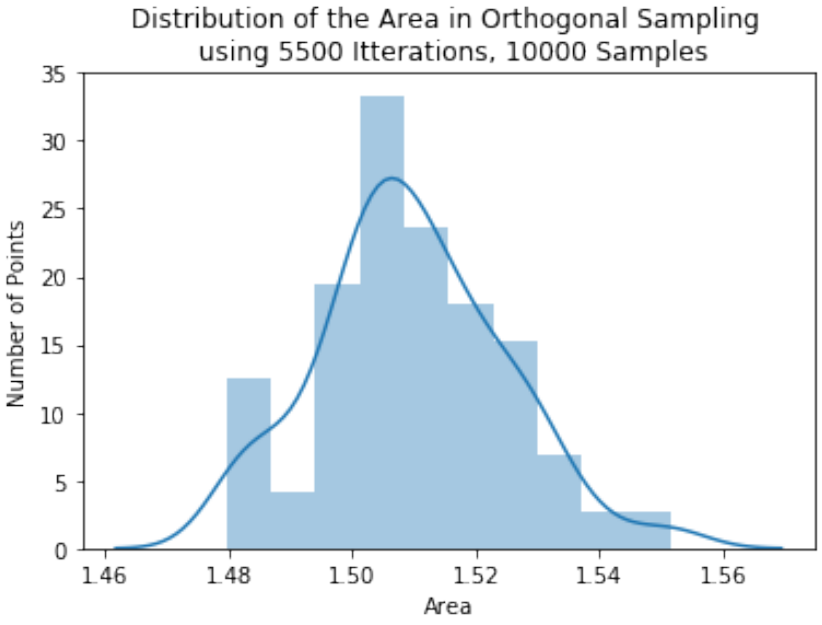
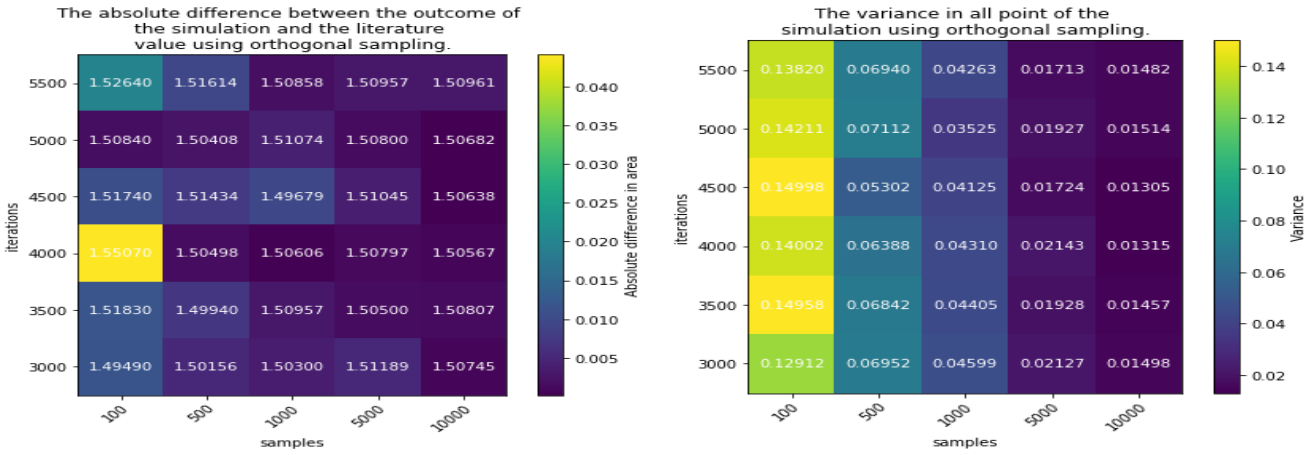



**Figure 7:** Distribution of samples from orthogonal sampling of the Mandelbrot simulation with 10000 samples and 5500 iterations.

Once this was implemented the same was conducted for orthogonal sampling method. Once again due to computational expense, the larger set sizes of 20000 and 30000 could not be computed. The area of the Mandelbrot set was calculated from each of these runs, and compared to the actual value as with the random sampling. The mean found with 10000 samples and 55000 iterations was 1.50961 with a variance of 0.01482. Knowing this the actual

value of the mean falls within the range $1.50961 \pm 0.0044$ with a 99% confidence interval. A heat map showing the variance and the difference in mean values of each combination of iteration and sample size can be seen in Figure 8. Once again it can be observed that the simulation mean approaches the actual mean as the number of iterations and sample size increases, and the variance decreases as the number of samples increases.



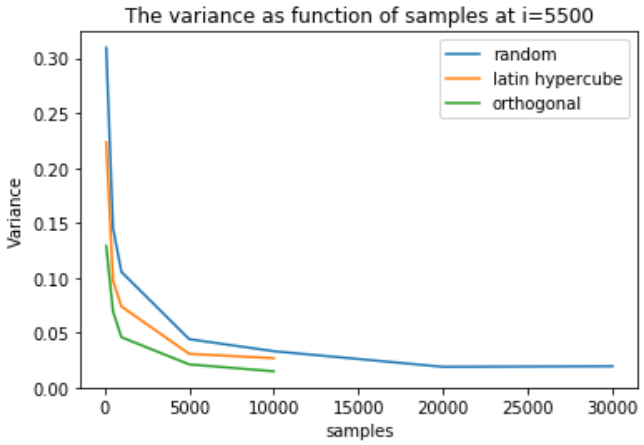

**(a)** **(b)**

**Figure 8:** Heatmaps from orthogonal sampling of the Mandelbrot simulation with 10000 samples and 5500 iterations.

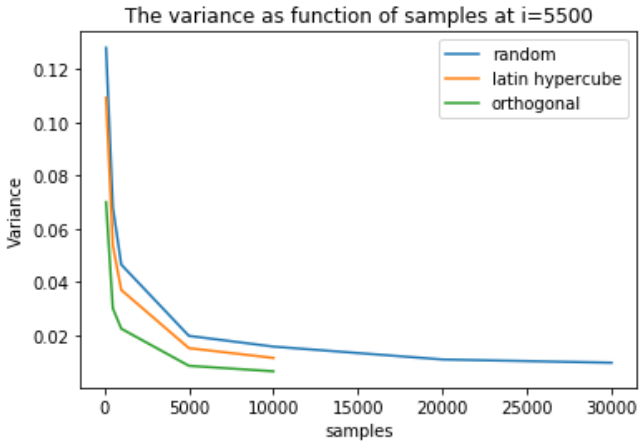
Once the data was collected, for all sampling methods without antithetic variables, the sampling methods were compared to each other. The effects on variance and the mean were examined with varying sample size and iterations. These comparisons can be seen in the Figures 9 and 10.

As can be seen from the figures, the number of points has a clear effect on the mean and the variance, as the number of points increases the variance decreases and mean approaches the accepted value for the experiment. From the heat plots shown previously, it can be seen that varying the number of iterations, will effect the mean, as the more iterations, the more points that are not in the set will be excluded. It can be seen in Figure 10 that there is not much correlation with the variance. This could be explained by the fact that the variance is independent of the number of iterations. At each iteration, the variance will be the same as the number of points excluded will be consistent.

It should be noted that while the mean value and variance do improve with the hypercube and orthogonal sampling, it should also be stated that the computational power required to run these sampling methods is much higher than the random sampling. The time to run hypercube is significantly larger than pure random, and orthogonal sampling is significantly

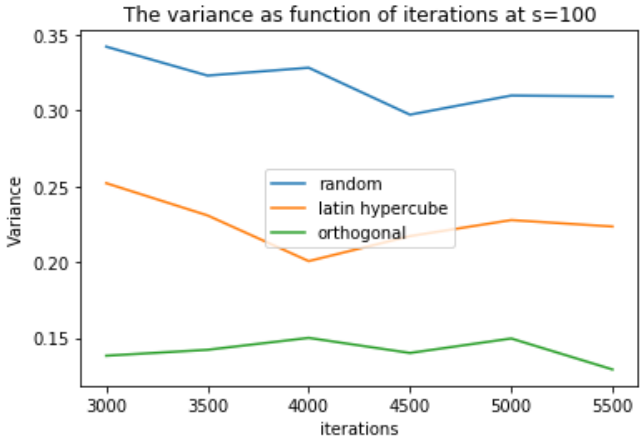



**(a)** Variance of sampling method without antithetic variables

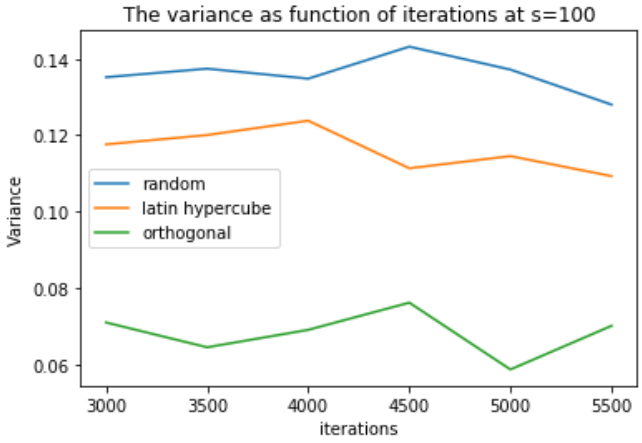



**(b)** Variance of sampling method with antithetic variables

**Figure 9:** Variance of the simulation as the number of samples increases.




**(a)** Variance of sampling method without antithetic variables




**(b)** Variance of sampling method with antithetic variables

**Figure 10:** Variance of the simulation as the number of iterations increases.

larger than hypercube. Depending on the computational power available this should be taken into consideration. In this experiment, hypercube and orthogonal sampling took too long to run at the higher point values, and had to be excluded from the experiment as the computer crashed while running them.

In order to determine how the sampling methods compared to each other, a boxplot was created at the highest conditions that all 3 shared. The comparison can be seen in Figure 11 which clearly shows that the orthogonal method has less variance than the hypercube method, which has less variance than the random method. This is due to the intelligent

sampling of the system allowing the space to be better explored by the simulation. It can also be seen that the mean for the orthogonal and the hypercube are approximately in the same area, while the random is slightly higher. This could be due to the pure random sampling exploring a larger number of external points, due to the lack of intelligent sampling.

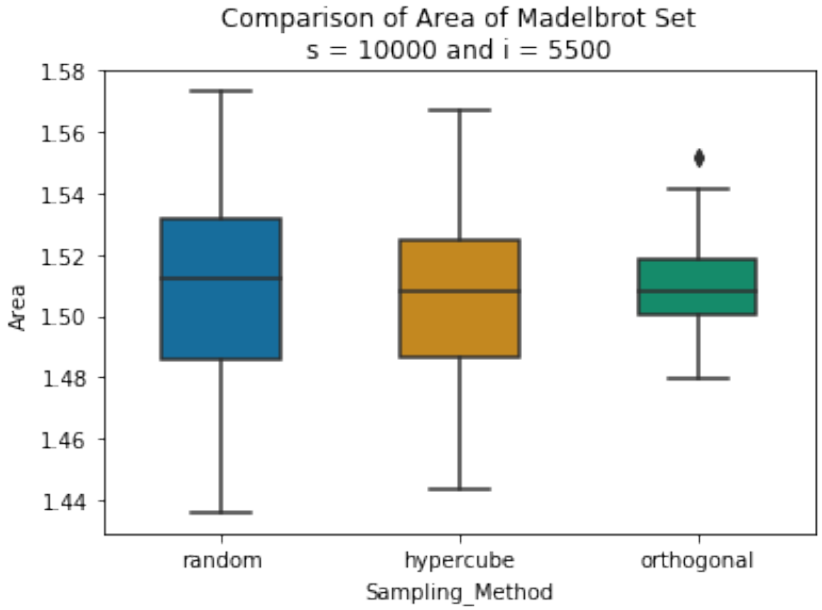



**Figure 11:** Boxplot of the three sampling methods (Random, Latin Hypercube, and Orthogonal) at 10000 samples and 5500 iterations.

Next, antithetic variables were added into the sampling methods in the process described in the Methods and the simulations were rerun, starting with random sampling. The following figures (Figures 12 - 13) show the heatmap and distribution of the random sampling with antithetic variables.

From Figure 12, it can be seen that the distribution is tighter around the mean this combined with the heatmap can show that the variance was indeed reduced in the system by the addition of antithetic variables. The mean found with 30000 samples and 55000 iterations was 1.50780 with a variance of 0.00920. Knowing this the actual value of the mean falls within the range $1.50780 \pm 0.0027$ with a 99% confidence interval.

Next the heatmap and the distribution of the latin hypercube sampling with antithetic variables was examined. This can be seen in Figures 14 - 15. These figures also show a tighter distribution around the mean, and combined with the heatmap show that the variance is decreased with the addition of antithetic variables. The mean found with 10000 samples and 55000 iterations was 1.50705 with a variance of 0.01220. Knowing this the actual value of the mean falls within the range $1.50705 \pm 0.0036$ with a 99% confidence interval.

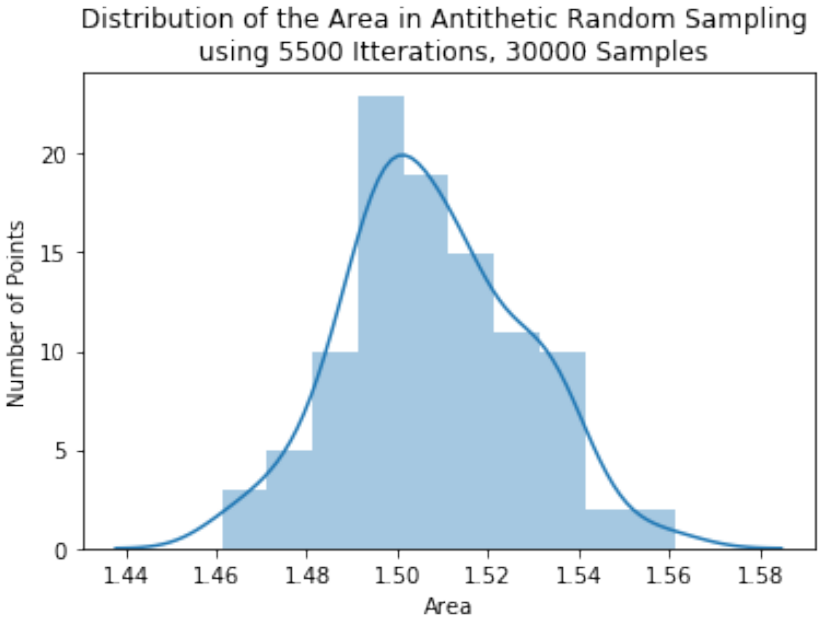
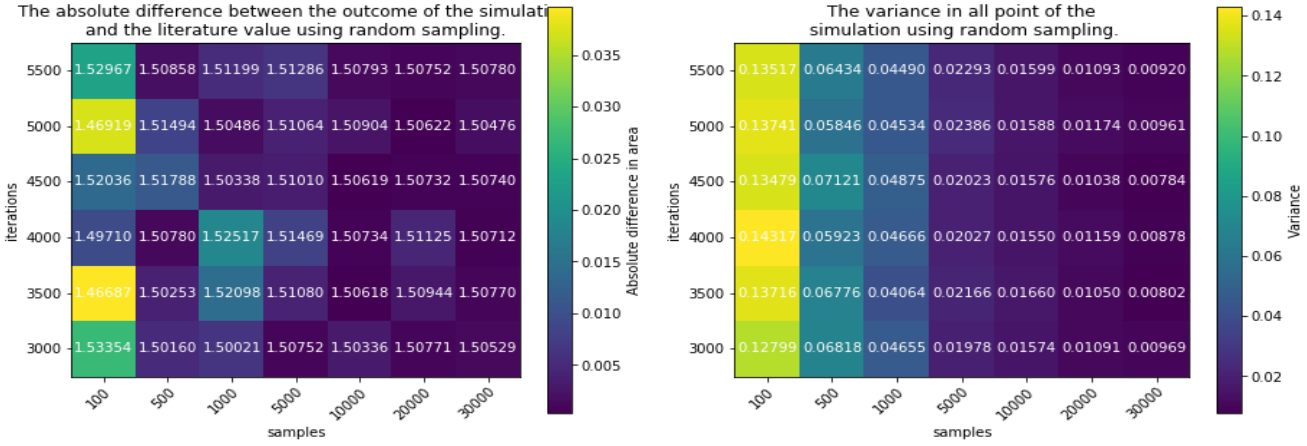



**Figure 12:** Distribution of samples from random sampling with antithetic variables of the Mandelbrot simulation with 30000 samples and 5500 iterations.




**(a)** **(b)**

**Figure 13:** Heatmaps from random sampling with antithetic variables of the Mandelbrot simulation with 30000 samples and 5500 iterations.

Finally the heatmap and the distribution of the orthogonal sampling with antithetic variables was examined. This can be seen in Figures 16 - 17. These figures also show a tighter distribution around the mean, and combined with the heatmap show that the variance is decreased with the addition of antithetic variables. The mean found with 10000 samples and 55000 iterations was 1.50513 with a variance of 0.00703. Knowing this the actual value of the

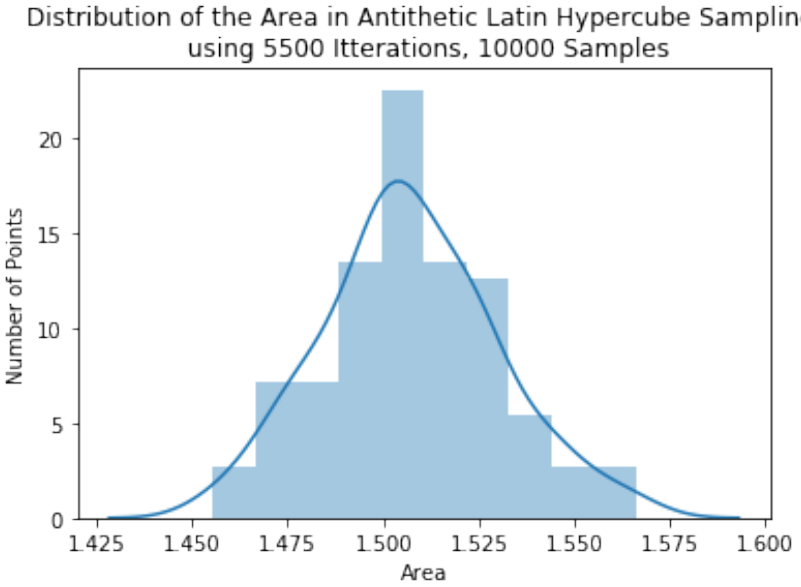
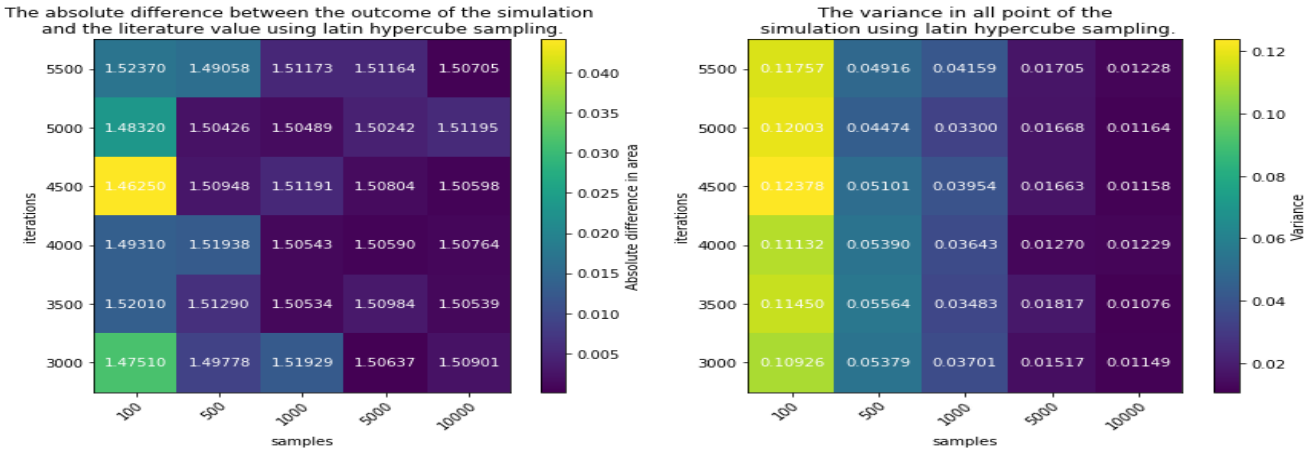



**Figure 14:** Distribution of samples from Latin Hypercube sampling with antithetic variables of the Mandelbrot simulation with 10000 samples and 5500 iterations.




**(a)** **(b)**

**Figure 15:** Heatmaps from Latin Hypercube sampling with antithetic variables of the Mandelbrot simulation with 10000 samples and 5500 iterations.

mean falls within the range $1.50513 \pm 0.0021$ with a 99% confidence interval.

The three sampling methods with antithetic variables were then comapred to each other through a boxplot at the highest conditions that all 3 shared. The comparison can be seen in Figure 18 which clearly shows that the orthogonal method has less variance than the hypercube method, which has less variance than the random method. This is once again due to the use of intelligent sampling. When comparing the boxplots of sampling with

and without antithetic variables it can be clearly seen that the antithetic variables have less variance and a righter distribution than the non-antithetic counterpart. This shows that the addition of antithetic variables gives a significant boost to the accurancy and decreased variance, with minimal additional comutational power.

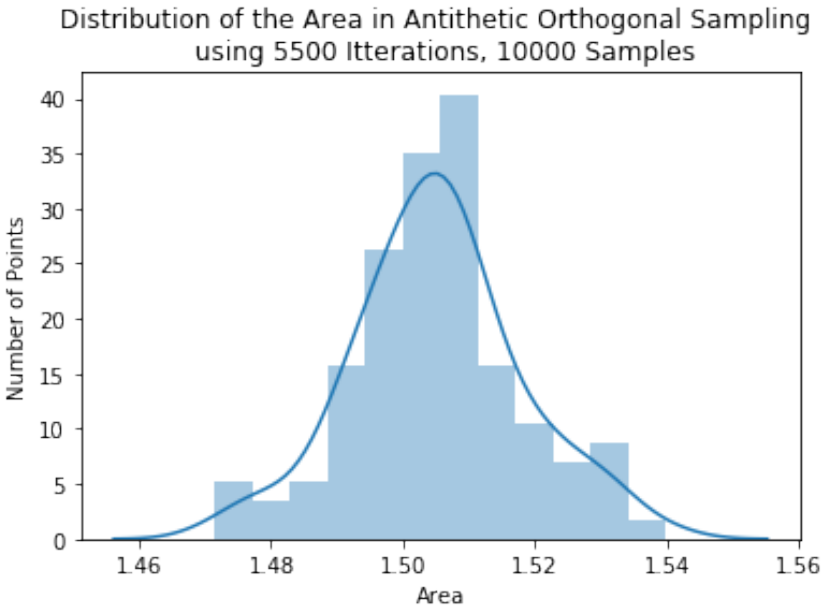
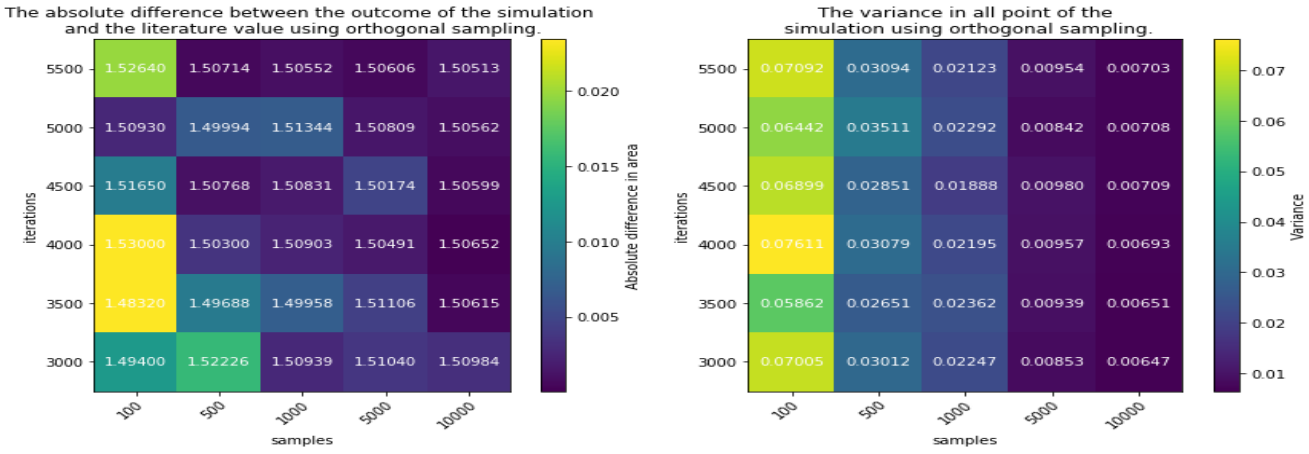



**Figure 16:** Distribution of samples from orthogonal sampling with antithetic variables of the Mandelbrot simulation with 10000 samples and 5500 iterations.




**(a)** **(b)**

**Figure 17:** Heatmaps from orthogonal sampling with antithetic variables of the Mandelbrot simulation with 10000 samples and 5500 iterations.

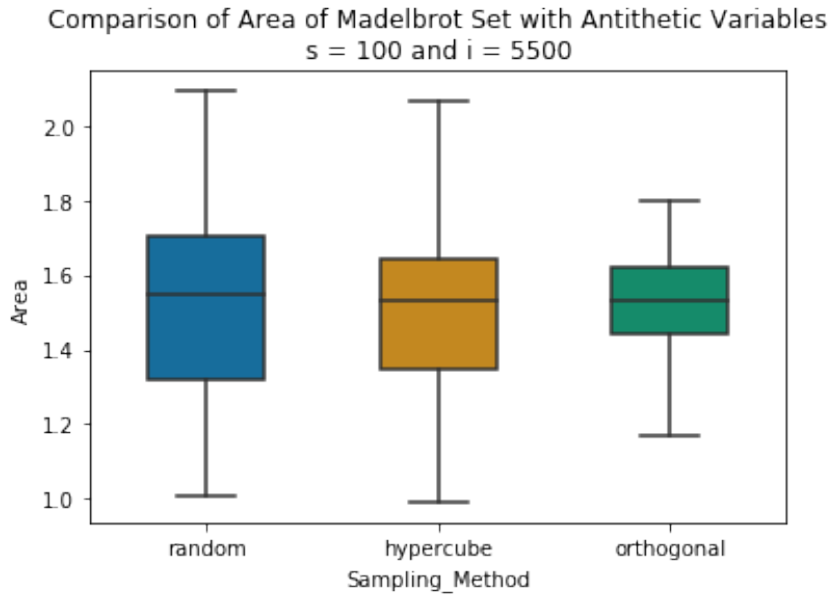



**Figure 18:** Boxplot of the three sampling methods with antithetic variables (Random, Latin Hypercube, and Orthogonal) at 10000 samples and 5500 iterations.

# 6 Conclusion

In this experiment, a hit and miss Monte Carlo simulation was used to explore the Mandelbrot set area and the properties of its convergence. Two important variables in determining the area are the number of iterations $i$ and the number of samples $s$. Both were varied to examine their influence. As expected, increasing the number of samples both increased the accuracy and decreased the variance, while increasing the number of iterations increased the accuracy. Increasing $i$ and $s$ let to the convergence of the area to the literature value of the area.

The simulation was done with three different methods of sampling points: random sampling, latin hypercube sampling and orthogonal sampling. Each type of sampling was simulated with $s = (100, 500, 1000, 5000, 10000)$ and $i = (3000, 3500, 4000, 4500, 5000, 5500)$ and random sampling was also done for $s = (20000, 30000)$. For $i = 5500$ and $s = 30000$, the mean value of random sampling was $1.507 \pm 0.005$. For $i = 5500$ and $s = 10000$, the mean value of latin hypercube sampling was $1.5060 \pm 0.0066$ and the mean value of orthogonal sampling was $1.5096 \pm 0.0044$. It became apparent that both the accuracy and the variance converged quicker for orthogonal sampling than for latin hypercube sampling and quicker for latin hypercube sampling than random sampling. It should be noted there was a trade off in computational time, as the hypercube and orthogonal sampling took significantly longer than the random sampling.

All simulations were run for a second time with antithetic variables, which led to significantly lower variance in all simulations. The mean value of random sampling was $1.50780 \pm 0.0027$,for latin hypercube sampling it was $1.50705 \pm 0.0036$ and the mean value of orthogonal sampling was $1.50513 \pm 0.0021$. The use of antithetic variables causes a decreased variance with little additional computational power.

# References

[1] Wolfgang Beyers. Mandelbrot set: Zoom Sequence 00. `https://wolfgangbeyer.de/`, 2001. [Online; accessed 15-Nov-2018].

[2] Adrien Douady and John H Hubbard. Exploring the Mandelbrot set. The Orsay notes. *Publ. Math. Orsay*, 1984.

[3] T. Forstemann. Numerical estimation of the area of the Mandelbrot set. `http://www.foerstemann.name/labor/area/Msetarea.pdf`, 2012. [Online; accessed 15-Nov-2018].

[4] Sheldon M. Ross. *Simulation.* 2013.