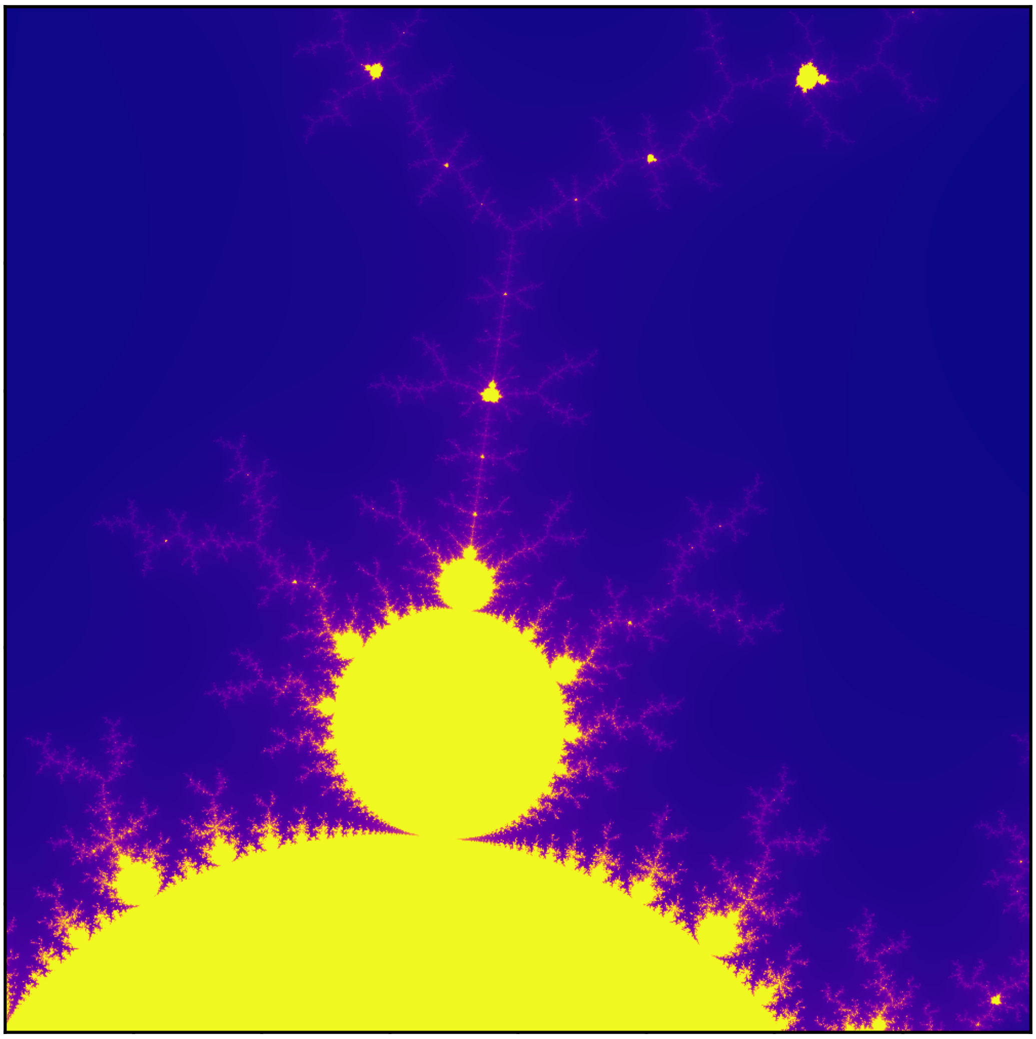
Determining the surface area of the Mandelbrot set using Monte Carlo integration

Kaat Brinksma
13919938
karin.brinksma@student.uva.nl

Nina Wagenaar
11326336
nina.wagenaar@student.uva.nl



November 21, 2022

University of Amsterdam
Faculty of Science

Abstract

This report explores three different sampling methods for Monte Carlo integration applied on calculating the surface area of the Mandelbrot set. The sampling methods are Pure Random Sampling, Latin Hypercube sampling and Orthogonal sampling which all three aim to sample evenly within the sample space. A fourth method is introduced, called Recursive Boundary Zoom, which specifically aims to sample densely in the most varying area's of the function and sample sparsely in the less varying area's. Overall, all sampling methods show similar results considering convergence towards a mean area based on the number of samples as well as a decreasing confidence interval of 95%. Future research is needed to further explore the possibilities of Boundary Zoom sampling.

1 Introduction

In mathematics, there is a set of shapes called fractals. These shapes have the property of being infinitely complex and never-ending. In other words, fractals are iterative, repeating the same pattern over and over again. The term fractal was first introduced by mathematician Benoit Mandelbrot (Mandelbrot, 1975), although several fractals have been studied before the introduction of this term. Mandelbrot also was the first to visualize the fractal in high quality that later became well-known as the Mandelbrot set.

The Mandelbrot set is a set of numbers in the complex plane. For certain complex numbers, the result of applying a function $f_c = z^2 + c$ iteratively remains bounded while for others it will diverge to infinity. Those numbers for which the result remains bounded are in the Mandelbrot set. The Mandelbrot set itself is also bounded, as it is known that for all $c > 2$, f_c will go to infinity. However, the Mandelbrot does have an infinite perimeter (Shishikura, 1991). Therefore, it is extremely hard to determine the exact area of the Mandelbrot set. Indeed, it has never been found. Yet, there are methods to get an estimate of the area of the Mandelbrot set.

In this report we will focus on estimating the area of the Mandelbrot set using Monte Carlo integration, though other methods of estimation also exist (Bittner et al., 2017); (Andreadis and Karakasidis, 2015). Monte Carlo integration is a numerical integration method. Computationally integrating a function is not as easy as it seems, it usually involves both discretizing and analytically approximating the integrand. Some discretizing methods, like the trapezoidal rule, becomes too computationally costly in higher dimensions. In these cases Monte Carlo integration can be used to get an estimate of the integrand. Monte Carlo integration is non-deterministic, meaning that each time it is performed the outcome will be be slightly different. It works by repeatedly choosing random point within a certain interval and evaluating the integral at that point.

In the case of the Mandelbrot set, the area can be estimated using Monte Carlo integration by repeating randomly sampling a point in the complex plane and calculating for that point whether it is in the Mandelbrot set. There are two parameters of importance for the accuracy of the area: the amount of samples that are taken and how many iterations of f_c are calculated. The main goal of the first part of this report is to investigate the effect of these parameters on the area estimate of the Mandelbrot set. Specifically, our question is whether the estimate converges as the amount of samples and iterations increases. We expect to indeed see convergence.

One problem with randomly sampling points for evaluation is that the sample space might not be properly explored. Since every sample is taken independently of all past samples, it could be that we only sample one area of the sample space. As a result, our estimations might not be an accurate reflection of the true area. Of course, it is possible to diminish the changes of this happening by taking a lot of samples. However, this is computationally expensive. Therefore, to ensure that the sampling space is properly explored without wasting computational resources, sampling methods such as latin hypercube sampling and orthogonal sampling have been designed. In this report we will experiment with these methods. Our main question is whether these methods yield similar estimates of the area as well as whether they improve the convergence rate. We expect to see no difference between estimates.

Lastly, in this report we will propose a method to improve the convergence rate of the Monte Carlo approach. This method, which we call recursive boundary zoom, works by focusing the computational power on the areas that are of the most interest, namely the boundaries. We expect to see that this proposed method increases the convergence rate compared to the rate found in orthogonal sampling.

2 Methods

The accuracy of the approximation of the integrand depends on the number of iterations done when finding if a number is in the Mandelbrot set or not.

Firstly we will describe how the Mandelbrot set is found, and briefly touch on how we made the pretty pictures. Then we will discuss the hard stuff, finding the surface area and comparing some random sampling methods.

2.1 Creating and visualising the Mandelbrot set

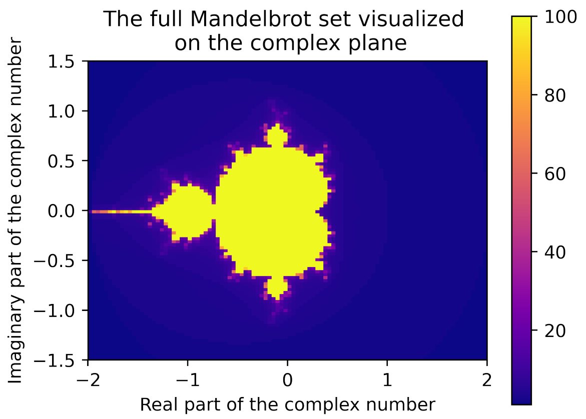
The Mandelbrot set is the collection of complex numbers (z, c) for which the repeated call of the function $f_c(z_{n+1}) = z_n^2 + c$ does not cause the number to diverge to infinity. For $z < -2$ and $z > 2$ the number always diverges. In order to find the surface area of the Mandelbrot set, it is important to first visualise the Mandelbrot set as to get an idea why finding this area is so hard.

To determine whether a point lies within the Mandelbrot set, a recursive algorithm is implemented, see Algorithm 1. If this function returns the value equal to the maximum number of iterations, the point c is seen as being in the Mandelbrot set.

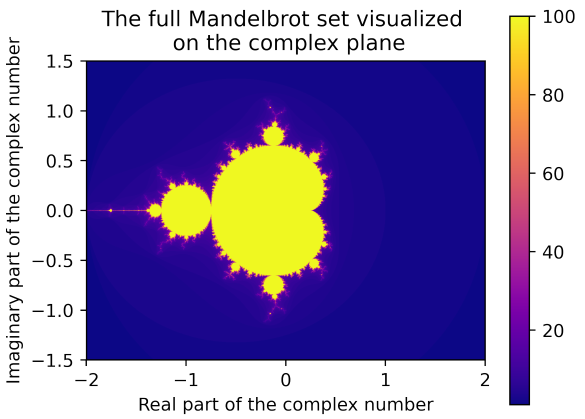
Algorithm 1 Finding the Mandelbrot set

```
Require: $c$                      ▷ Complex number (z, c) where $z = 0$
Require: $MaxIt$                  ▷ Maximum number of iterations
Require: $Iteration = 0$          ▷ The current number of iterations, initialized at 0
  if $Iteration < MaxIt$ and $abs(z) \leq 2$ then:
      $z = z^2 + c$
      $Iteration+ = 1$
      Recursive function call
  else
      return $Iteration$
  end if
```

To visualize the Mandelbrot set, each point is given is certain colour corresponding to how many iterations it takes before the the result of applying the function $z = z^2 + c$ diverges. Some points will not diverge within the maximum number of iterations, these points are given the colour yellow. The results of these visualizations, where the maximum number of iterations is 100, can be see in Figure 1. Here the right picture is less granular than the left picture, i,e the right visualization takes more points in account and in return gives a more accurate description of what the Mandelbrot set looks like.



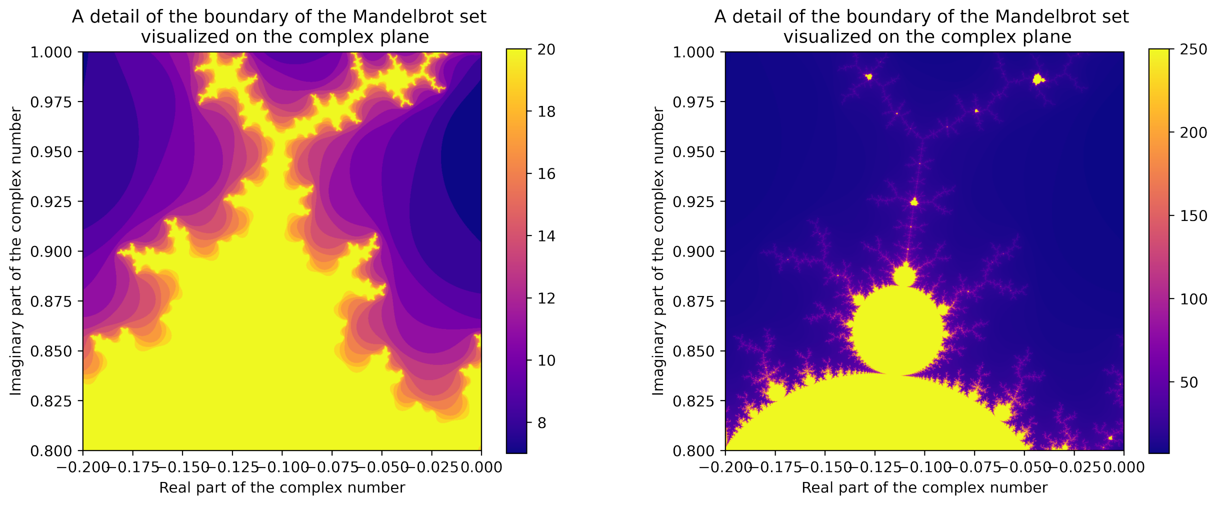
(a) The total Mandelbrot set, with $1e^4$ points and 100 iterations.



(b) The total Mandelbrot set, with $1e^6$ points and 100 iterations

Figure 1: The effect of the grain size iterations on the surface area of the Mandelbrot set, where a point lies within the Mandelbrot set if its value is equal to the maximum number of iterations. The colours represent the amount of iterations until divergence.

The most interesting parts of the Mandelbrot set can be found at the boundaries, as it is here we can find the iterative structure typical of fractals. We made some visualizations that zoom in on these boundaries, see Figure 2. The left picture was made using maximum iterations = 20 while the right picture was made using maximum iterations = 250. Clearly, there is a stark difference between these two pictures with the right being much more precise, indicating how important it is to use enough iterations to get a precise visualization of the Mandelbrot set.



(a) A detailed view of an edge of the Mandelbrot set, with $1e^6$ points and 20 iterations.

(b) A detailed view of an edge of the Mandelbrot set, with $1e^6$ points and 250 iterations.

Figure 2: The effect of the number of iterations on the surface area of the Mandelbrot set. The colour represents the number of iterations after which the Mandelbrot iteration on that complex number causes divergence to infinity, where it is assumed that a point lies within the Mandelbrot set if its value is equal to the maximum number of iterations.

2.2 Estimating the Area with Monte Carlo Integration

For each sample we considered the sample to be in the Mandelbrot set when Algorithm 1 returned the maximum amount of iterations, otherwise it was not considered to be in the Mandelbrot set. The area of the Mandelbrot set was then estimated using the following formula:

$$A_{is} = A_{sampled} * \frac{S_{Mandelbrot}}{S} \tag{1}$$

Here $A_{sampled}$ is the area that is sampled, $S_{Mandelbrot}$ is the number of samples that are classified to be in the Mandelbrot set and S is the amount of samples taken in total.

2.2.1 Parameters

For all simulations the following parameters are of importance; the area that is sampled, the number of runs and the number of samples and iterations.

For each experiment we sampled between -2 and 2 for the real part of the complex plane and between -1.5 and 1.5 for the imaginary part of the complex plane. This was chosen based on the visualizations in Figure 1, as these visualizations showed this chosen area captures the entire Mandelbrot set.

As discussed in the introduction, Monte Carlo simulation is non-deterministic. Thus every time a simulation is run, a slightly different outcome is obtained and therefore we need to look at both the mean and variance calculated on multiple replications to get an accurate picture of the behavior of the simulation. However, more runs also mean more computing power, meaning that the amount of replications is a trade-off between accuracy and computational resourced. Based on some experimentation we decided to run each experiment 25 times, as this was still within the limits of our computational resources.

Lastly, we did some some additional research to investigate the effect of increasing the maximum amount of iterations and samples. First, we want to determine how many iterations we need to get a reasonable level of precision. Therefore, we investigate the difference between A_{is} and A_{js}, j goes from 0 to i. We also look at how A_{is} converges as s increases. For both the amount of iterations and samples we have to make a trade-off between precision and computational resources. Therefore, our purpose is to find the amount of samples and iterations that yield consistent results while also being within our computational resources. For the experiments into the influence of parameters as described here holds that they were done using pure random sampling. The results can be found in Section 3.1.

2.2.2 Pure Random Sampling

The pure random samples are obtained using a uniform distribution between -2 and 2 for the real part and -1.5 and 1.5 for the imaginary part of the complex number of interest. the samples are independent en identically distributed, however, they are not truly random, as we used the function from the *numpy* library to generate these samples. Rather the samples are pseudo-random.

2.2.3 Latin Hypercube Sampling

With Pure Random Sampling, the samples are independent en identically distributed, but they can be spread across the sample space very unevenly. To improve this spread of samples, Latin Hypercube Sampling (LHS) is introduced (Loh, 1996). These samples are obtained by evenly dividing the sample space over each axis into the required number of samples, and taking one sample per row- and column index respectively.

Algorithm 2 Latin Hypercube Sampling

```
Require: $no\_samples$                                  ▷ Required number of samples
Require: $MaxIt$                     ▷ Maximum number of iterations until divergence
  Divide real and complex axis into $no\_samples$ equally spaced points
  Randomly choose a real and complex part          ▷ Each value can only be chosen once
  Check if number converges to infinity under Mandelbrot iteration using algorithm 1
```

2.2.4 Orthogonal Sampling

Latin Hypercube sampling can ensure a more even spread across the axes, but since the real and complex parts of the numbers are sampled independently, it can still happen that some sub-spaces are sampled more densely than others. An example is when the samples are all taken on the diagonal of the sample-space. This is still a Latin Hypercube, but can very well provide a heavily skewed estimate. To prevent this uneven spread of samples, an orthogonal sampling method is implemented (Garcia, 2000). This improvement on Latin Hypercube sampling first divides the sample-space into M^2 subspaces (by dividing the axes into M intervals) and then takes Latin Hypercube samples for each subspace, as described in algorithm 3.

Algorithm 3 Orthogonal Sampling

```
Require: $no\_samples$                                  ▷ Required number of samples
Require: $MaxIt$                     ▷ Maximum number of iterations until divergence
Require: $M$      ▷ When the sample space needs to be divided into $M^2$ equally sized subspaces
  Divide real and complex axis into $M$ intervals
  For each subspace, find the area using Latin Hypercube Sampling (see algorithm 2)
```

2.2.5 Recursive Boundary Zoom

When looking at the visualizations of the Mandelbrot set, the most interesting behaviour takes place around the boundaries, as visualized in figure 2. This raises the belief that dense sampling in these boundary area's is preferred, while more sparse sampling can be done in area's where either all points diverge to infinity, or none of the points do. This inspired us to explore a method where more samples are taken in area's around the boundary, while simultaneously sample sparsely in the other area's. This method is named Recursive Boundary Zoom, after the recursive implementation of this idea. The implementation of Recursive Boundary Zoom is described in algoritm 4.

A small number of samples is taken over the sample(sub)-space. The samples are taken using Pure Random Sampling (see section 2.2.2) with a uniform distribution. These samples determine whether further investigation of this (sub)space is needed. If all samples are either in or out, or the maximum number of recursive function calls is reached, the function returns the scaled ratio of samples which are in the Mandelbrot set. If not, the (sub)space is split into 4 smaller sub-spaces and *no_samples* are taken per subspace. When done, the ratio can be multiplied with the total sampling area after running the algorithm to determine the area of the Mandelbrot set.

Algorithm 4 Recursive Boundary Zoom

```
Require: $no\_samples$        ▷ Required number of samples per function call
Require: $MaxIt$        ▷ Maximum number of iterations until divergence
Require: $MaxIterZoom$        ▷ Maximum number of recursive function call depth
Require: $re_{max}$        ▷ Maximum value of the real part of the complex number
Require: $re_{min}$        ▷ Minimum value of the real part of the complex number
Require: $im_{max}$        ▷ Maximum value of the imaginary part of the complex number
Require: $im_{min}$        ▷ Minimum value of the imaginary part of the complex number
Require: $CurrentIter = 0$        ▷ To track the recursive function calls
  Get $no\_samples$        ▷ Using pure random sampling
  if All samples in Mandelbrot set then
    return $\frac{1}{4^{CurrentIter}}$
  else if No samples in Mandelbrot set then
    return 0
  else if $CurrentIter == MaxIterZoom$ then
    return $\frac{1}{4^{CurrentIter}} * \frac{samples\_in}{no\_samples}$
  else
    $CurrentIter += 1$
    $re_{mid} = re_{min} + \frac{re_{max} - re_{min}}{2}$
    $im_{mid} = im_{min} + \frac{im_{max} - im_{min}}{2}$
    $TopLeft = BoundaryZoom(CurrentIter, re_{max} = re_{mid}, im_{min} = im_{mid})$
    $TopRight = BoundaryZoom(CurrentIter, re_{min} = re_{mid}, im_{min} = im_{mid})$
    $BottomLeft = BoundaryZoom(CurrentIter, re_{max} = re_{mid}, im_{max} = im_{mid})$
    $BottomRight = BoundaryZoom(CurrentIter, re_{min} = re_{mid}, im_{max} = im_{mid})$
    return $TopLeft + TopRight + BottomLeft + BottomRight$
  end if
```

The main difference between boundary zoom and the other previously described sampling methods, is that the actual number of samples is not predetermined when running this algorithm, however, there is an analytical maximum number of both total and leaf samples. The leaf number of samples is the number of samples taken to determine the ratio of samples which lie in the Mandelbrot set, and the total number of samples is the sum of the leaf samples and the samples taken to determine if more detailed sampling is needed within a specific area. These maxima depend on the maximum number of recursive function calls ($MaxIterZoom$) and the number of total samples per function call ($no\_samples$). These values can be calculated using the following equations:

$$no\_samples^{max}_{total} = \sum_{i=0}^{MaxIterZoom} 4^i * no\_samples \tag{2}$$

$$no\_samples^{leaf}_{total} = 4^{MaxIterZoom} * no\_samples \tag{3}$$

See table 1 for the calculated maximum numbers of samples for a maximum recursive depth up to 5.

3 Results

The sampling methods as introduced in section 2 are compared by studying the effect of the number of samples on the confidence interval around the mean of the area. First, appropriate constant values for the number of iterations and samples are found using pure random sampling, after which the sampling methods are compared for these constants using the mean, variance and confidence interval around the mean for 30 model runs.

3.1 Effect of Samples and Iterations on Estimate

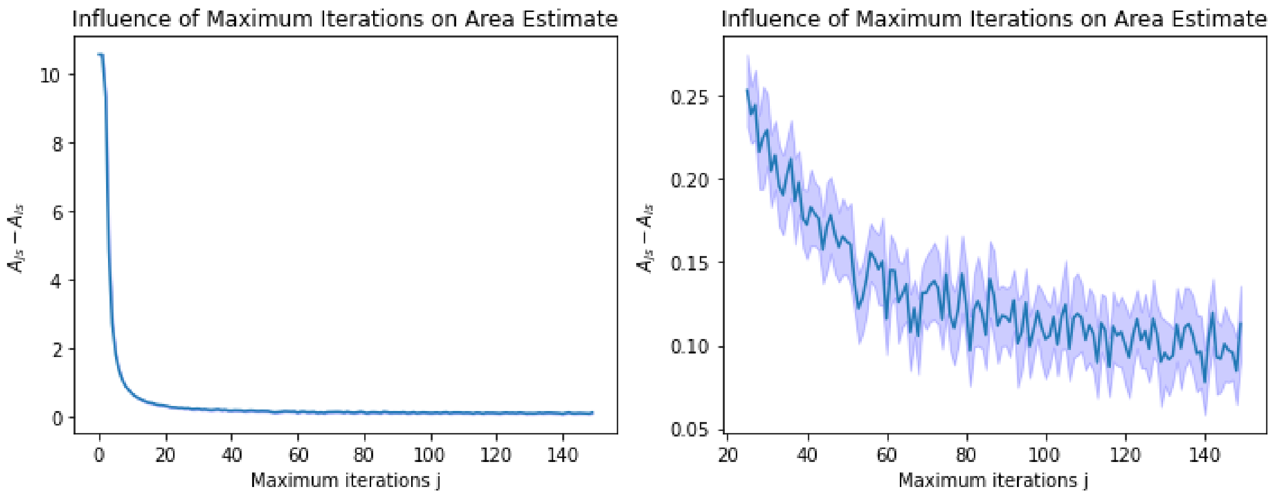


Figure 3: Influence of maximum iterations on the accuracy of the area estimate of the Mandelbrot set when taking 5000 purely random samples. Dark blue represents the mean estimate, light blue represents the 95% confidence interval.

The aim of our first experiment is to investigate the influence of the amount of maximum iterations on the estimate on the area, by seeing how $A_{is} - A_{js}$ changes as a function of the amount of iterations j. Here i is 200 and j increases from 0 to 200. The results of this experiment can be seen in figure 3. In the left part of this figure we see that as the the maximum iteration j increases, $A_{is} - A_{js}$ drops sharply and seems to stabilize quickly. However, zooming in on only the higher values of j, i.e $j > 25$ shows that even for these higher values $A_{is} - A_{js}$ still decreases, see the right part of Figure 3. Yet, more iterations also require more computational resources. Therefore, we use for all further experiments maximum iterations $i = 100$, as this still fits with the computational budget.

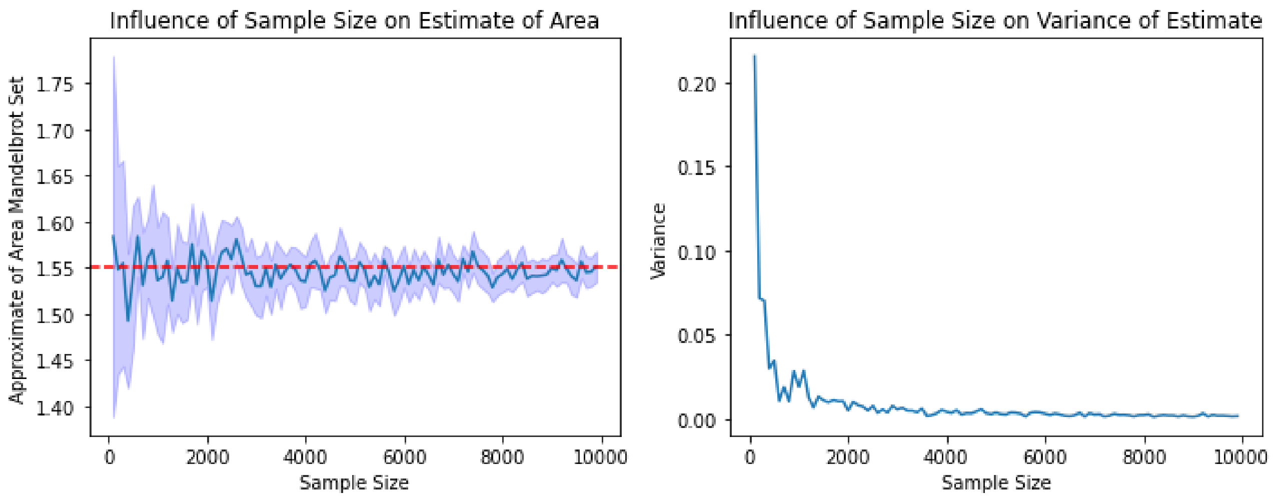


Figure 4: Influence of number of samples on the accuracy of the area estimate of the Mandelbrot set when allowing for 100 Mandelbrot iterations. Method is pure random sampling. Dark blue represents the mean estimate, light blue represents the 95% confidence interval The red line represents the mean when using 10.000 samples

Our second experiment, also using pure random sampling, investigates the influence of the number of samples on the area estimate. Specifically, we are interested in whether the estimate converges. The results of this experiment can be seen in Figure 4. The left part of the figure shows the area estimate of the Mandelbrot set as the sample size increases. Here we can see that the 95% confidence interval becomes smaller as more samples are used, indicating a more confident estimate. The right part of the figure shows the effect of the sample size on the variance. As can be seen in this figure, as the sample size increases, the variance decreases indicating that the estimate converges. After around 4000 samples the variance seems to stabilize more, although there is still a gradual decrease. Based on these figures and the computational resources needed, we will use a maximum sample size of 5000 for any further experiments.

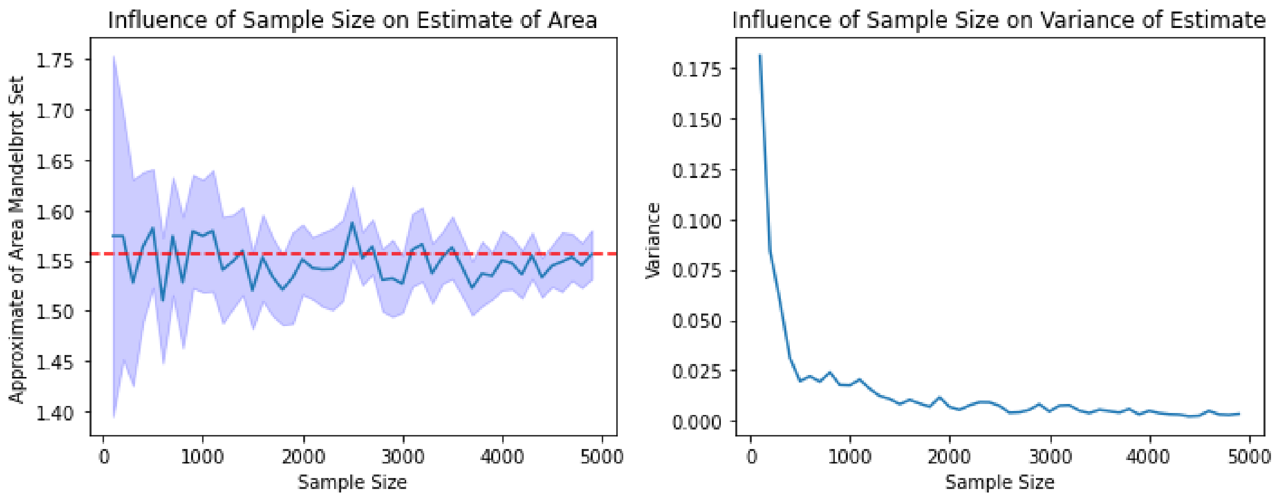
3.2 Performance comparison of Sampling Methods

After determining the appropriate values for the number of iterations and maximum number of samples, the Monte Carlo integration is done using the three different sampling methods.

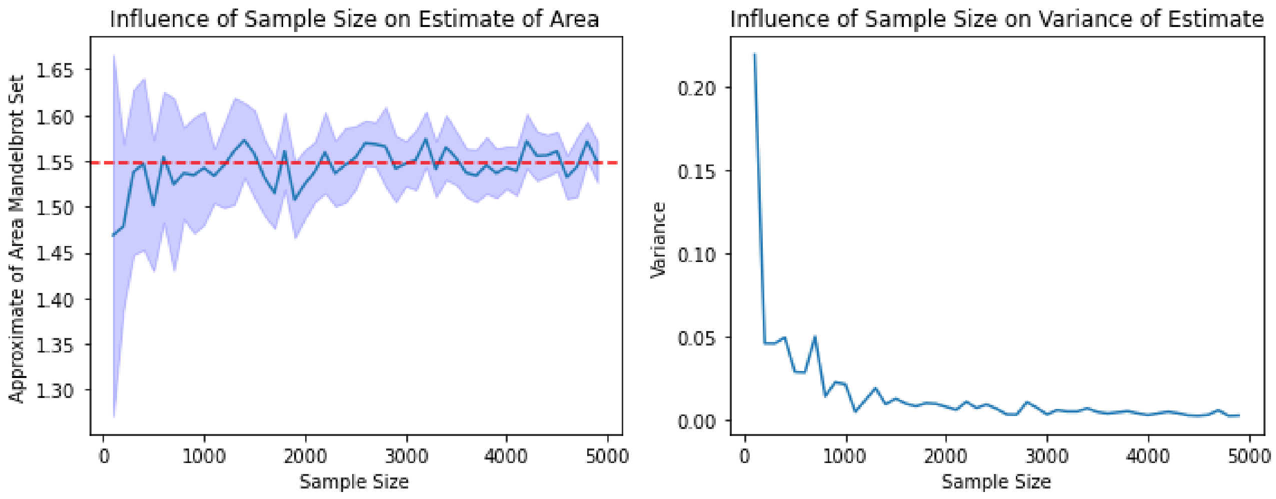
The sampling methods are tested by using increasing numbers of samples, starting with 100, increasing by 100 each until the maximum number of 5000. For every number of samples, the model is run 30 times. Figure 5a shows the confidence interval and variance around the mean area of the Mandelbrot set using Pure Random Sampling, figure 5b shows the confidence interval and variance using Latin Hypercube sampling and figure 5c shows the same statistics using Orthogonal sampling.

The confidence interval when using Pure Random Sampling is slightly wider than when using both the Latin Hypercube and the Orthogonal Sampling, and the greatest decrease in the confidence interval can be seen when using Orthogonal sampling.

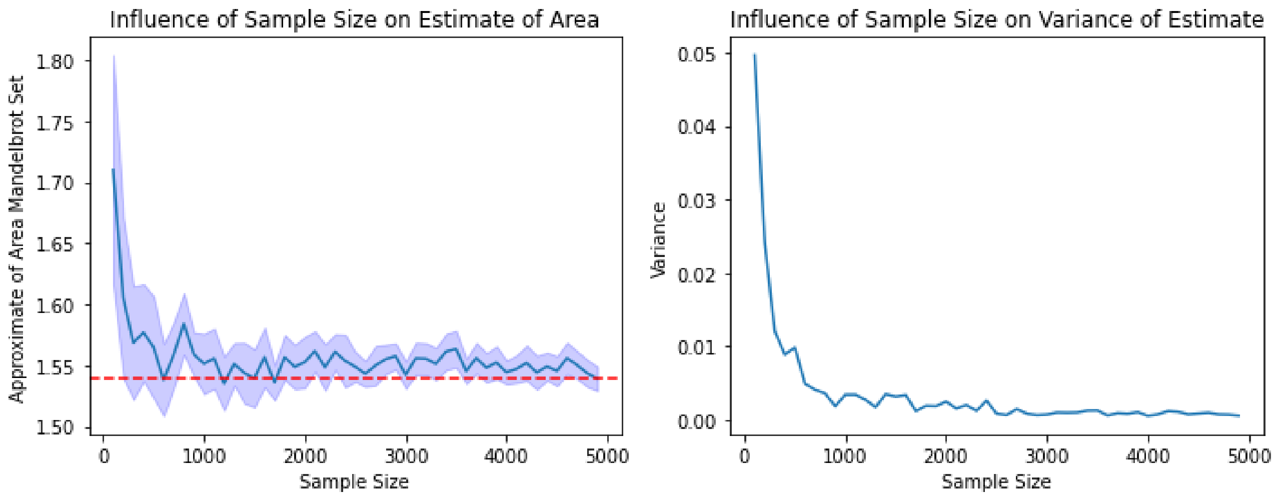
The variance of the mean when using Pure Random and Latin Hypercube Sampling is similar, whereas the Orthogonal Sampling shows a much smaller variance around the mean. This corresponds with the smaller confidence interval.



(a) The confidence interval and variance around the estimated area of the Mandelbrot set, estimated using Pure Random sampling with a maximum of 100 iterations and 5000 samples.



(b) The confidence interval and variance around the estimated area of the Mandelbrot set, estimated using Latin Hypercube Sampling with a maximum of 100 iterations and 5000 samples.



(c) The confidence interval and variance around the estimated area of the Mandelbrot set, estimated using Orthogonal Sampling with a maximum of 100 iterations and 5000 samples.

Figure 5: The comparison of the sampling methods. Dark blue represents the mean estimate, light blue represents the 95% confidence interval. Both are taken using 30 model runs. The red line represents the mean of 30 runs when using 5000 samples

Maximum depth recursive calls	**Maximum leaf samples**	**Maximum total samples**
0	100	100
1	400	500
2	1600	2100
3	6400	8500
4	25.600	34.100
5	102.400	136.500

Table 1: The theoretical maximum leaf and total samples when using Recursive Boundary Zoom

3.3 Performance Improvement using Boundary Zoom

The number of samples per function call *no_samples* is kept constant to 100. The resulting maximum numbers of samples can be found in table 1. These maxima are based on equations 2 and 3.

Figure 6 shows a much wider confidence interval compared to the other sampling methods when comparing the theoretical maximum number of samples to the samples taken in the other sampling methods. However, the number of taken samples roughly doubles for every extra maximum iteration allowed, as shown in figure 7, instead of the quadrupling which can be seen in table 1. This means that the larger the maximum recursive depth, the less reliable the theoretical maximum number of samples becomes as a metric for comparison.

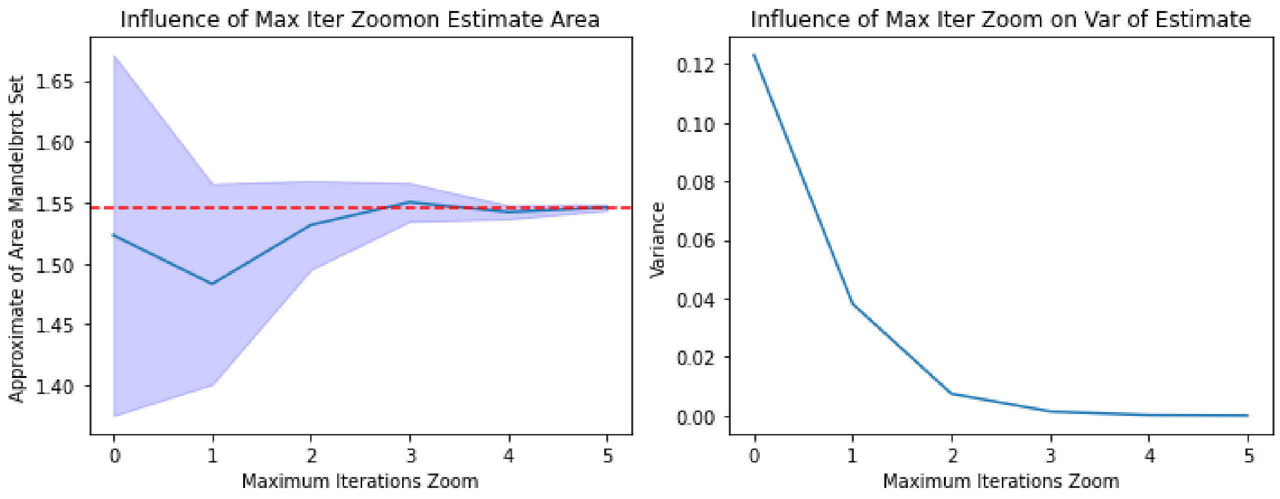


Figure 6: The confidence interval and variance around the estimated area of the Mandelbrot set, estimated using recursive boundary zoom with a maximum depth of 5 recursive calls. Dark blue represents the mean estimate, light blue represents the 95% confidence interval. Both are taken using 30 model runs. The red line represents the mean of 30 runs when using 5000 samples

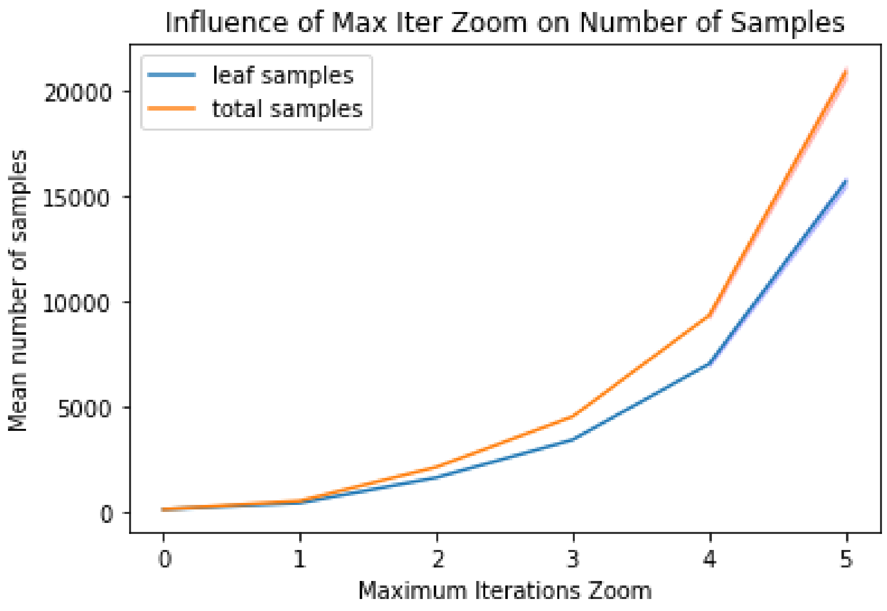


Figure 7: The ratio between the leaf samples and total samples when using boundary zoom, estimated using a maximum depth of 5 recursive calls. Orange represents the total number of samples, blue represents the number of leaf samples.

4 Discussion

For a maximum recursion depth of 4 and 5, the confidence interval around the mean is very narrow when using Boundary Zoom, but this may be caused by the fact that these calculations had more samples available than the calculations using the established sampling methods.

The confidence interval and the variance do decrease if both the number of samples and the number of iterations increase. The confidence interval around the mean is the most narrow when using Orthogonal sampling when comparing similar numbers of samples drawn. This includes the comparison to boundary zoom, since the maximum recursive depth of 3 has the most representable number of samples drawn.

The Boundary Zoom sampling method needs further research before any decisive statements on performance increase can be made. Considering these results only, Boundary Zoom sampling is not an improvement of the established sampling methods. Future research, especially when applying the theory behind Boundary Zoom on other classic integration problems, is needed.

When a function behaves highly irregular, and has many small area's instead of one big area, the Boundary Zoom might not improve on the other sampling methods. Therefor it would be best to implement the Boundary Zoom, or a similar method, only on functions where the surface is concentrated, and the most interesting behaviour is around the edges of the figure when plotting the function.

Another point which needs further investigating is the improvement of run-time when using Boundary Zoom compared to Orthogonal Sampling. Even when the theoretical maximum of Boundary Zoom samples exceeds the number of samples, Boundary Zoom runs faster than Orthogonal sampling. This is caused by the fact that the actual number of samples drawn, both the total and the leaf samples, is much lower than the theoretical maximum.

All tested sampling methods show a similar trend, where an increase in the number of samples shows a decrease in the 95% confidence interval. This is what we expected. All sampling methods converge to a mean area of approximately 1.55.

References

[1] Ioannis Andreadis and Theodoros E Karakasidis. “On a numerical approximation of the boundary structure and of the area of the Mandelbrot set”. In: *Nonlinear Dynamics* 80.1 (2015), pp. 929–935.

[2] Daniel Bittner et al. “New approximations for the area of the Mandelbrot set”. In: *Involve, a Journal of Mathematics* 10.4 (2017), pp. 555–572.

[3] Antonio G Garcia. “Orthogonal sampling formulas: a unified approach”. In: *SIAM review* 42.3 (2000), pp. 499–512.

[4] Wei-Liem Loh. “On Latin hypercube sampling”. In: *The annals of statistics* 24.5 (1996), pp. 2058–2080.

[5] B.B. Mandelbrot. *Les objets fractals: forme, hasard et dimension.* Nouvelle bibliothèque scientifique. Flammarion, 1975. ISBN: 9782082106474. URL: `https://books.google.nl/books?id=dj8ZAQAAIAAJ`.

[6] Mitsuhiro Shishikura. “The Hausdorff dimension of the boundary of the Mandelbrot set and Julia sets”. In: *arXiv preprint math/9201282* (1991).