Stochastic Simulation Assignment 1

Gijs van Langen - 12768510
Daniël Vink - 10675140

November 2020

1 Abstract

In this report we will briefly take a look at the Mandelbrot set. We will use stochastic simulation with different sampling methods and different sampling parameters to give an estimate of the area of the figure produced by this set. Furthermore, we try to improve on our accept/reject method by sampling from a more efficient range of random numbers.

2 Introduction

The Mandelbrot set was defined by Adrien Douady, who named it in honour of Benoit Mandelbrot (Douady and Hubbard, 1985). The set contains complex numbers which do not converge to infinity when recursively called in a certain function (which will be described in more detail in the next section). The best way to get a feeling for the set is via visualization, for example figure 1 below. The Mandelbrot set itself is visualized by darker shades of red (or black even), with the 'coastlines' being increasingly more light in colour:

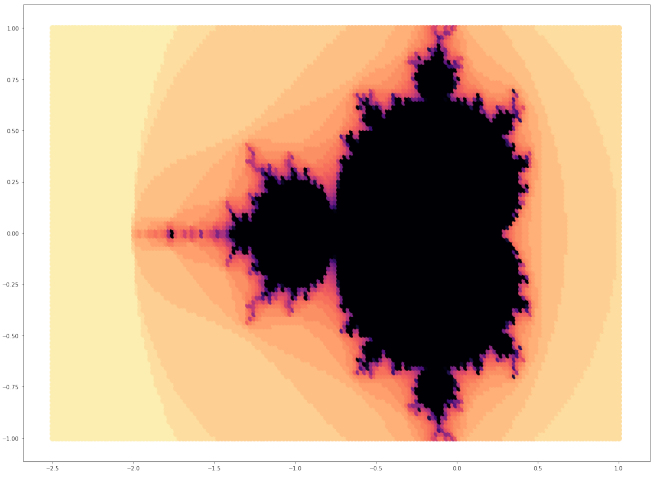

Figure 1: The Mandelbrot set visualized

This set has certain properties which complicate matters (Mandelbrot, n.d.). First and foremost, the figure can be infinitely detailed. Zooming in on the 'coastline' reveals more and more details and new patterns. This also means that in theory, the circumference of the Mandelbrot set tends to infinity. However, the area of this figure can be approximated, which will be shown later on. The rest of this report is structured as follows: First, we will show more theoretical background on both simulation and the Mandelbrot set. Next, we publish our approximations with also a brief discussion of improvements applied to our simulations. Lastly, we will discuss our results and conclude our findings.

3 Theoretical Background

As mentioned, the Mandelbrot set contains complex numbers c, for which a certain function, f_c, does not tend to infinity if it is called recursively. The following function is used (Mandelbrot, n.d.):

$$F_c(z_n) = z_n^2 + c = z_{n+1} \tag{1}$$

where c is a certain complex number of the form $a + bi$ (where a and b are real numbers and i represents the imaginary component) and z is a recursively changing parameter, with $z_0 = 0$. So if this function does not diverge for a certain complex number c, then that c belongs to the Mandelbrot set. A popular example to illustrate this is -1, which belongs to the Mandelbrot set. -1 can also be written as $-1 + 0i$, i.e. -1 does not have an imaginary part. With starting value $z_0 = 0$, filling in $c = -1$ yields the following results:

$$F_c(0) = 0^2 - 1 = -1$$

$$F_c(-1) = -1^2 - 1 = 1 - 1 = 0$$

$$F_c(0) = 0^2 - 1 = -1$$

One can already see the pattern that has established itself here: z alternates between 0 and -1. For $c = -1$, the function will not tend to infinity (or, it will not diverge), meaning that $-1 + 0i$ is a complex number that belongs to the Mandelbrot set. In figure 1 above, $c = -1$ is included. The x-axis displays the real part (a), while the y-axis displays the imaginary part (b). We can indeed see that the point $(-1, 0)$ is found in the figure.

For our simulations we make use of the Accept/Reject technique (Zavodszky, 2020a). The main idea is that we sample random coordinates in a certain area and for each coordinate check if it is included in the Mandelbrot set. To assess the area of the set, look at the proportion between rejections and acceptances. Since we know the area of the space from which we sample, we can approximate the area of the Mandelbrot set using this proportion.

4 Results

4.1 Pure Random

The first attempt at estimating the area of the Mandelbrot set was an implementation of a 'Pure Random Sampling' method. From the visualisation, a range was determined to sample points in.

This range, a rectangular area encompassing the Mandelbrot set, spans from [-2, 1] on the real (X) axis, and from [-1, 1] on the imaginary (Y) axis. Then, points were sampled from uniform random distributions covering these two ranges. The two sampled points combined to form a single point on the complex plane. This process was repeated for a set amount of samples.

For a given amount of maximum iterations, every sampled point is then translated to the corresponding complex number and passed into equation (1). A number is included in the Mandelbrot set if the algorithm reaches the maximum amount of iterations before the complex number attains a value higher than 2, which is when the equation will rapidly tend toward infinity.

After all samples are tested for their inclusion, the area of the Mandelbrot set is approximated by taking the product of the sampled area and the ratio of included samples.
An average taken over $n = 30$ runs, with a sample size of $S = 100.000$, and a maximum amount of $i = 100$ iterations resulted in an approximated area of:

$$Area_M(n = 20, S = 100000, i = 100) = 1.5804$$

4.2 Parameter convergence

The sample size and maximum amount of iterations from the initial approximation were arbitrarily picked. As both parameters directly translate to an increased amount of calculations, the computational time rises with them. Therefore, it is reasonable to investigate the effect of the parameters on the approximations.

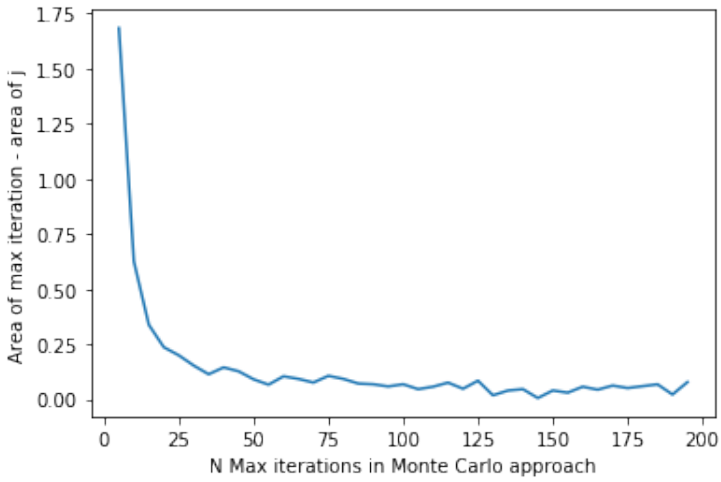

Figure 2: Convergence of incremental steps over parameter i

To test the direct effect of the size of the maximum amount of iterations, the Mandelbrot approximation algorithm was run multiple times and averaged over a range of iteration sizes. This list spanned from [5, 200] with incremental steps of 5. To compare the results, the difference in area between each iteration size and a baseline approximation was calculated and plotted in figure 2. This 'most accurate' baseline is the final calculation with an $i = 200$. From the resulting graph it was determined that a parameter value of $i > 50$ was generally not worth the computing time.

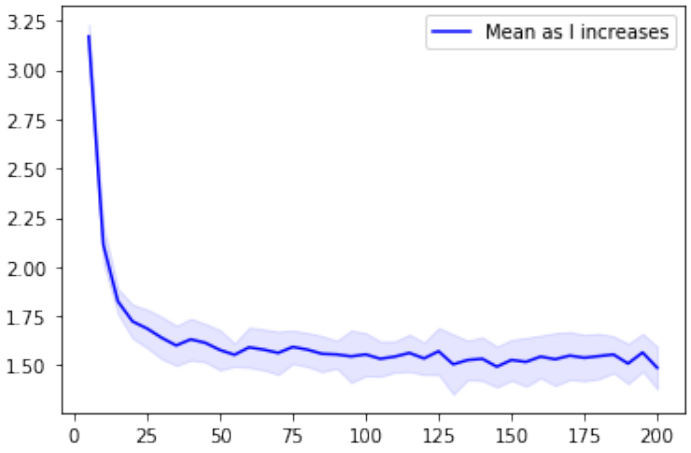

Figure 3: Confidence interval of iteration convergence

A similar approach was used to determine the optimal sample size. An exponential sample range over $[10^2, 10^7]$ was considered, but the implementation of a sample size larger than 10^6 proved to be too computationally expensive. As a result, the largest amount of samples was capped at 10^5. This corresponding 'most accurate' result was again used as a baseline of comparison for the other results. The resulting difference, visible in figure 4 in the area estimation proved minimal when comparing sample sizes, but figure 5 shows that the variance greatly decreases with a higher sample size, leading to a better confidence interval (Zavodszky, 2020b). Considering this upper bound sample size was able to be computed adequately, the optimal sample size was determined as $S = 10^5$.

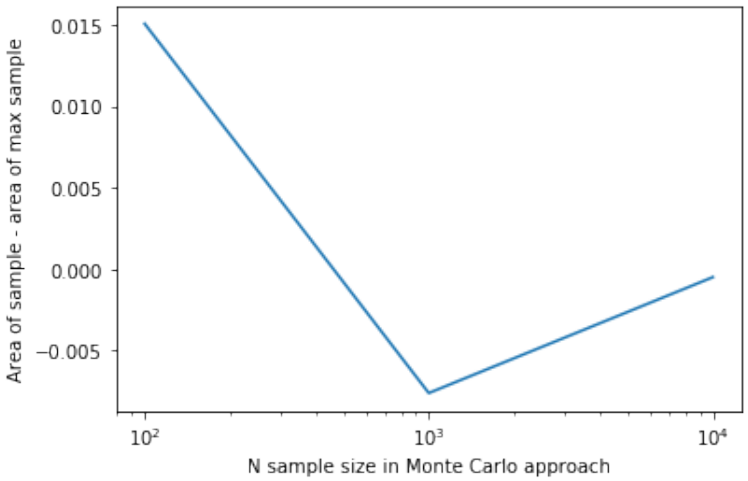

Figure 4: Caption

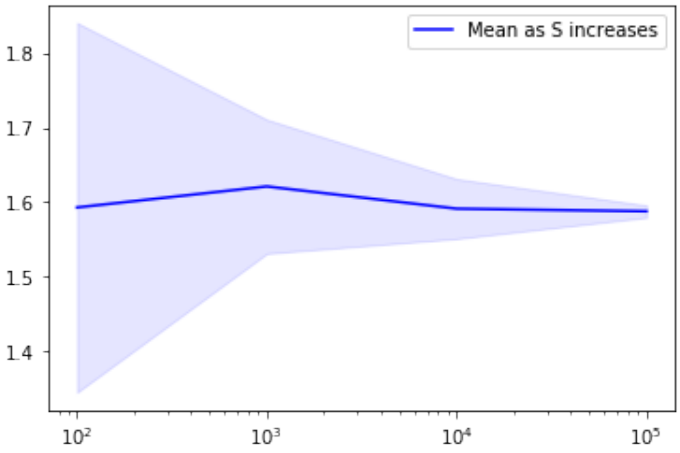

Figure 5: Caption

4.3 Latin Hypercube Sampling

An alternative way of estimating the area of the Mandelbrot set would be by using the Latin Hypercube Sampling method (Zavodszky, 2020c). This method semi-enforces uniformity among samples draw. With pure random sampling, the probability exists that all samples drawn lie very close to each other in a grid. Latin Hypercube Sampling seeks to eliminate this chance by not allowing samples from the same row or column twice. This means that if we have drawn a random coordinate (5,3) from our sample space, we can no longer draw other coordinates from either row 5 or column 3. So coordinates (5,1), (3,3), (10,3), (5,5), etc. are all no longer possible. This means you eliminate the probability of all random sampling being in the same corner of the sampling space. In our simulations, this meant that we would create 2 lists of $S = 100000$ samples each, corresponding to our possible x and y coordinates (so ranging from $x = -2$ to $x = 1$ and $y = -1$ to $y = 1$. These samples are not picked randomly, but at a fixed distance from one another. Then, we shuffled both lists such that the coordinates were no longer ordered. By matching these shuffled lists, we got $S = 100000$ coordinates with no possibility of a certain x-value or y-value ever repeating. So with this new LHS-sampling method, we again created estimates for the area of the Mandelbrot set with iterations $i = 50$, number of tests $n = 30$ and sample size $S = 100000$ as constants. Figure 6 shows the approximation retrieved from our simulations as well as the confidence interval around our sample mean for both conventional pure random sampling and LHS sampling:

We can see that both methods estimate the area to be around 1.59, so the actual resulting sample mean does not differ as much. However, the accompanying confidence intervals drawn around these sample means differ greatly. For pure random sampling, we are 95% certain that the sample mean lies between 1.58 and 1.60. For the Latin Hypercube sampling method, our confidence interval is much smaller. From the graph it is even hard to tell what the interval exactly is. A possible explanation for this smaller confidence interval is that samples are more uniformly drawn with Latin Hypercube sampling. This means that area approximations do not differ as much between simulation runs compared to pure random sampling. So the variance between means of different simulation runs is a lot smaller.

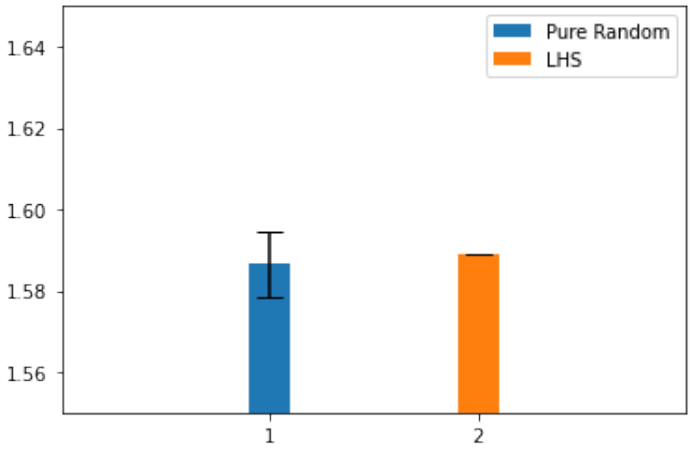

Figure 6: Pure Random sampling vs Latin Hypercube sampling

4.4 Orthogonal Sampling

As orthogonal sampling is an extension of Latin Hypercube Sampling, the calculation steps are as described in 4.3. For the sampled points, however, it is of importance to ensure there is no variance in between sample points. Intuitively this can be explained by points laying on the diagonal in a roster. Every column and row of the roster are sampled just once, but there is a clear pattern in the data and entire regions of the set remain unsampled. To enforce this lack of variance the imagined roster is divided in subregions where every region contains the same amount of samples, meanwhile still enforcing the 'rules' from LHS (Zavodszky, 2020c).

In practice, this was done by reshaping one of the sample lists into multiple equal sized parts. To obtain the targeted sample size of S, the size of the subregions was defined as the square root of the sample size, resulting in an S amount of subregions each sampled once. Because the sample size of 10^5 does not have a rounded square root, the sample size was set to the nearby 99225 which has a rounded square root of 315.

After dividing one of the lists in its subregions, both the elements in these regions and the regions themselves are shuffled. This resulted in a list of shuffled samples, albeit groups of related values are visible. Iterating over the size of the regions, a new list is created by picking the index of every shuffled subregion corresponding to the modulo of the iteration divided by the region size. Pairing this with an unshuffled axis results in a blockwise selected coordinate pair, or an orthogonal sample. The workings of the algorithm can be seen in figure 7.

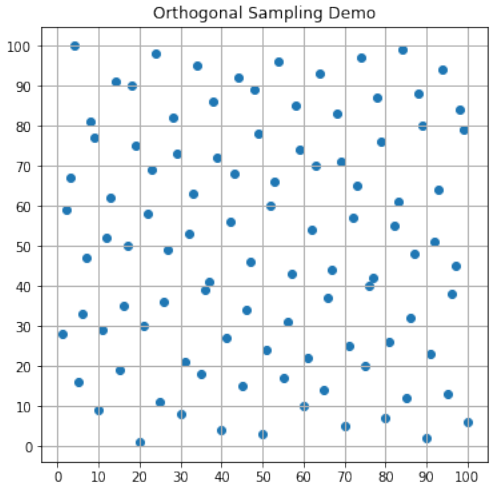

Figure 7: Demo of orthogonal sampling on a 100x100 grid

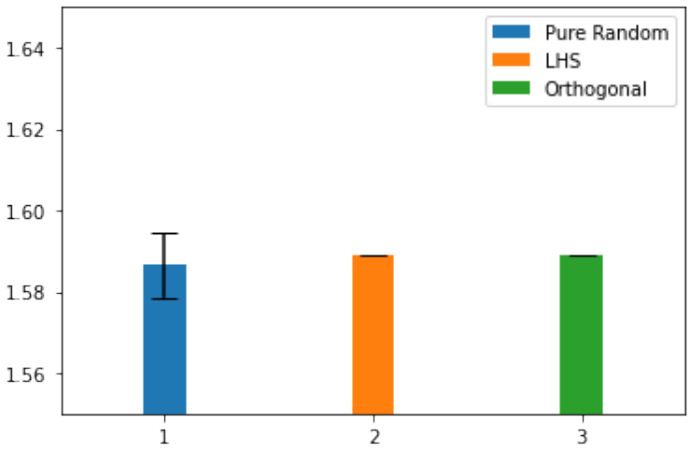

Figure 8: Pure Random vs LHS vs Orthogonal

As the implementation and inner working of orthogonal sampling is nearly identical to LHS, similar results were expected. From figure 8, however, results are indistinguishable. The average of the approximated was again calculated to be about 1.59 with an incredibly small confidence interval. A possible explanation is given in section 4.3.

4.5 Improved Monte Carlo convergence

From section 4.2 we concluded that increasing the sample size S affected how confident we were in our approximation of the area found, as seen in figure 5. If we want to improve on our confidence, we could sample more 'informative' points. The accept/reject method's accuracy depends not only on how many points you sample, but also on how many rejects you get. If you sample $S = 10000$ coordinates but only 1 of those coordinates is included in the Mandelbrot set, then your information received is low. Sampling $S = 100$ points with 10 of them being in the Mandelbrot set gives a more information already, even though sample size S might be smaller. So ideally, you want your sample space to be as close to your Mandelbrot set as possible. In our previous Monte Carlo method, we sampled from a square ranging from $x = (-2, 1$ and $y = (-1, 1)$. As figure 1 shows, we sample quite some redundant points from which we know in advance that they will be rejected. For example, points samples in the squares ranging from $x = (-2, -075)$ and $y = (-1, -0.5) \vee y = (0.5, 1)$ will always be rejected. So if we take a smarter sample space, we can increase the number of accepts we have without increasing sample size S. Hence, we decided to divide the sample space in 2 rectangles:

1. A larger rectangle ranging from $x = (-0.75, 0.5)$ and $y = (-1, 1)$ can be drawn around the larger 'island' or 'peninsula' on the right side of the set of figure 1

2. A smaller rectangle ranging from $x = (-2, -0.75)$ and $y = (-0.5, 0.5)$ can be drawn around the smaller 'peninsula' on the left side of the set of figure 1

Dividing total sample size proportionally over the size of these 2 individual rectangles yields the same total amount of samples drawn as before, namely $S = 100000$. However, since our sample space is now drawn more tightly around the Mandelbrot figure, we will on average have more coordinates sampled that lie within the set. The total area of our sample space is now smaller, namely $(1.25 * 1) + (1.25 * 2) = 3.75$ (where the area used to be $3 * 2 = 6$).

Then, we conducted the same analysis as the sample size convergence of section 4.2: analyse the confidence interval around our approximation while increasing S exponentially. We took iterations $i = 50$ and number of tests $n = 30$ as constants.

So overall there will be a higher change of accepting a coordinate pair with the same amount of samples S drawn. So after running that simulation $n = 30$ times we computed the sample mean and standard deviation, which we can use to compute the confidence interval of the approximations using this new sampling method, similar to figure 5. Figure 9 illustrates this confidence interval as well as our original confidence interval used from pure random sampling as described in section 4.2.

Figure 9 shows that the confidence interval using our smarter sampling method is closer around the mean compared to conventional pure random sampling. This means that we have a higher confidence in our sample mean: with the same amount of confidence (95%) we can specify a smaller interval in which we think the true value will be. It even seems that for $S = 10000 = 10^4$, our confidence interval with smarter sampling is half the size compared to conventional sampling. So that could even mean that we doubled the accuracy of our estimation! It thus seems that sampling in a closer space around the Mandelbrot set gives more informative coordinates (more accepts compared to rejects), and thus our estimates become more informative or robust as well.

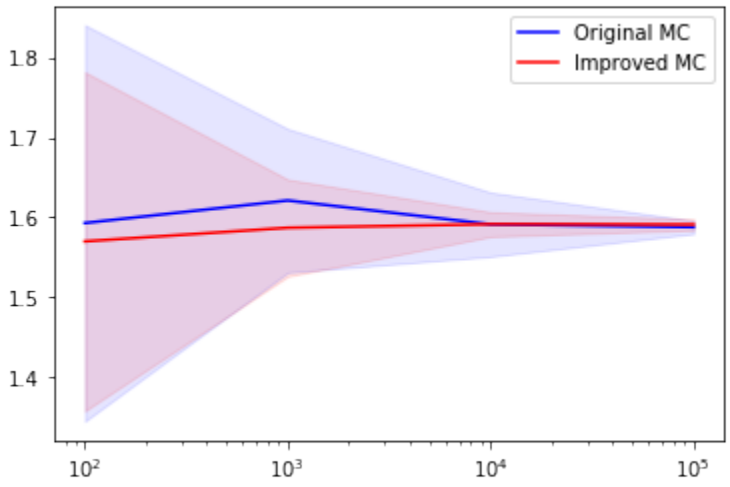

Figure 9: Improving Convergence rate with smarter sampling

5 Discussion and Conclusion

In this lab report we described different methods of approximating the area of the Mandelbrot set using simulations. We found that increasing the sample size would decrease the confidence interval substantially at the cost of computational power. Increasing the number of iterations seemed to converge to a constant confidence interval after around 50 iterations. Furthermore, we found that alternatives to the pure random sampling method decreased the confidence interval as well for simulations with iteration $i = 50$ and sample size $S = 100000$ ran 30 times. This seemed to indicate that using either LHS or orthogonal sampling resulted in samples being distributed more uniformly between different simulation runs, thus decreasing the variance between different approximations of the area.

For the orthogonal sampling, a fast and intuitive Python implementation was written. From a logical standpoint this made sense because the algorithm itself is complicated to understand, and this would prevent the necessity importing non-standard Python modules for potential reproductions of this experiment. At first glance, the results showed a good spread with the correct amount of samples per subregion. However, after closer inspection, it turned out this implementation is not completely free of variance. All samples in a y-axis subregion have the same spread over the x-axis, which is equal to the size of the subregions. This is clearly visible in the [0, 10] y-range as all samples lay on the grid lines. Although the algorithm is sub-optimal, its not expected to have a big impact on the approximation, but a better implementation would have been preferred.

Lastly, we tried to improve on the original Monte Carlo pure random sampling method by selecting a smarter sampling space. We found that if you sample from a sampling space that more closely envelops the Mandelbrot set that the confidence interval decreases as well, meaning that variance between simulation runs decreased. This also resulted in a faster convergence to a lower confidence interval as the number of samples S increased.

6 References

1. Douady, A. Hubbard, J.H. (1985): Etude dynamique des polynômes complexes. Prépublications mathémathiques d'Orsay 2/4

2. Mandelbrot, B. (n.d): Fractal aspects of the iterations of $z \mapsto \lambda z(1-z)$ for complex λ, z, Annals of the New York Academy of Sciences 357, 249/259

3. Zavodszky, G. (2020a): Random Numbers. Lecture Slides, Stochastic Simulations: University of Amsterdam, Lecture 2

4. Zavodszky, G. (2020b): Statistical Analysis of Data and Error Estimation. Lecture Slides, Stochastic Simulations: University of Amsterdam, Lecture 3

5. Zavodszky, G. (2020c): Variance Reduction and Sampling Techniques. Lecture Slides, Stochastic Simulations: University of Amsterdam, Lecture 5