Stochastic Simulation Project 1
Integrating the Mandelbrot set

Jeroen Hofman
10194754

November 13, 2011

1 Introduction

In this report we use stochastic simulations to integrate the Mandelbrot set. We first describe the definition of the Mandelbrot set. Next we describe which methods we used to calculate the integral. In the results section the results from this method are shown. The discussion at the end gives some points of improvement.

2 Theory

The Mandelbrot set is a so called *fractal*, which is a geometric shape with a high amount of self-similarity, i.e. by zooming in on a part of the fractal a partial copy of the fractal is obtained. Fractals can be defined by recurrence formulas, and the Mandelbrot set is an interesting fractal because its exact area is not known. There are however many computational studies which have tried to determine the area of the set. A study by reference [1] has computed the area of a Mandelbrot set by using a technique called pixel counting. Pixel counting is a technique where the Mandelbrot set is generated on a grid and the area of the set can be estimated by counting the number of pixels that are in the set after a large number of iterations (defined below). The best known value for the Mandelbrot set is 1.50659177±0.00000008.

The Mandelbrot set is generated using the following (complex-valued) equation:

$$P_c : z \mapsto z^2 + c \tag{1}$$

where a point c in the complex plane is in the Mandelbrot set if and only if:

$$|P_c^n(0)| \leq 2 \text{ for all } n > 0 \tag{2}$$

Since this condition has to hold for all n and has to be computed for the whole complex plane, the area of the Mandelbrot set cannot be solved exactly by a computer. Below is a simple graph showing the Mandelbrot set for the first 100 iterations, where the points that are computed are on an equally spaced grid of size 1000x1000.

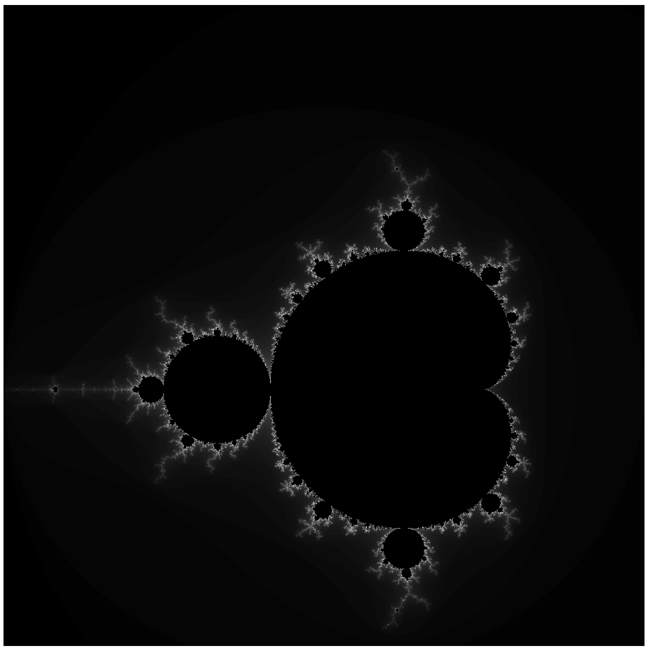

Figure 1: Visualization of the Mandelbrot set on the complex plane obtained by computing the condition in equation 2 for $n \leq 100$ for points in the complex plane equally spaced on a grid of size 1000x1000.

3 Methods

Apart from the method of pixel counting one can also use sampling to estimate the area of the Mandelbrot set. By sampling one considers a random point in the complex plane and computes whether or not this point is in the Mandelbrot set by using the condition in equation 2 for some maximum iteration depending on the computer power. By repeating this and taking many random sample points the area of the set can be estimated by dividing the amount of samples in the set by the total amount of samples N_{samples}, and multiplying this by the total area on which the samples are taken. The following three sample methods are used:

- Random sampling: The samples are taken randomly in the domain on which the sampling is applied.
- Latin Hypercube sampling: The domain is divided in N_{samples} rows and N_{samples} columns. One sample is taken in every column such that no sample is in the same row.
- Orthogonal sampling: The domain is divided in N_{samples} equal subspaces, and latin hypercube sampling is applied but now taking into account that every subspace has to contain one sample.

Though the orthogonal sampling gives the most uniformly distributed samples over the domain, this method is much slower than random sampling or latin hypercube sampling.
By keeping track at which iteration a sample point 'jumps' out of the Mandelbrot set, one obtains an estimate for the area A_i as a function of the iteration i up to a maximum iteration I_{max} depending on the computing power. By repeating this procedure N_{runs} times one can obtain an area estimate with an error for every iteration up to the maximum iteration I_{max}. The area estimate is equal to the mean area of iteration i, $\bar{A}_i$ given by:

$$\bar{A}_i = \frac{1}{N_{\text{runs}}} \sum_{j=0}^{N_{\text{runs}}} A_i \tag{3}$$

with a standard deviation, $\sigma(\bar{A}_i)$, of the mean area given by:

$$\sigma(\bar{A}_i) = \sqrt{\frac{N_{\text{runs}}}{N_{\text{runs}} - 1}} \sqrt{\overline{A^2}_i - \bar{A}_i^2} \tag{4}$$

The error in $\bar{A}_i$ is then, using a confidence interval of 99%, given by $\frac{2.58}{\sqrt{N_{\text{runs}}}}\sigma(\bar{A}_i)$ [2]. By assuming the mean area follows a power law as a function of the iteration an exponential function can be fitted to this series of iterations and corresponding $\bar{A}_i$. By letting the iteration go to infinity in the resulting fitted function one can estimate the area of the Mandelbrot set A_∞ as defined in the previous section.

4 Results

We use the method described in the previous section to estimate the area of the Mandelbrot set. The number of sample points used is $10^4, 10^5$ and 10^6 and the samples are generated using the Mersenne Twister. We have used all three sample methods as described in the previous section to calculate A_i by counting the number of samples inside the Mandelbrot set of iteration i and dividing this by the total amount of samples used. We have taken $I_{\text{max}} = 10000$. By taking $N_{\text{runs}} = 100$ we obtain a mean area $\bar{A}_i$ and an error for the mean area given by $\frac{2.58}{\sqrt{N_{\text{runs}}}}\sigma(\bar{A}_i)$. Figure 2 shown below gives this average area with error bars as a function of the iteration using the random sampling method with 10^6 samples. The shape of the data indeed seems to follow a power law. The thick black line is obtained by fitting the exponential function A + Bexp(Ci) to the data.

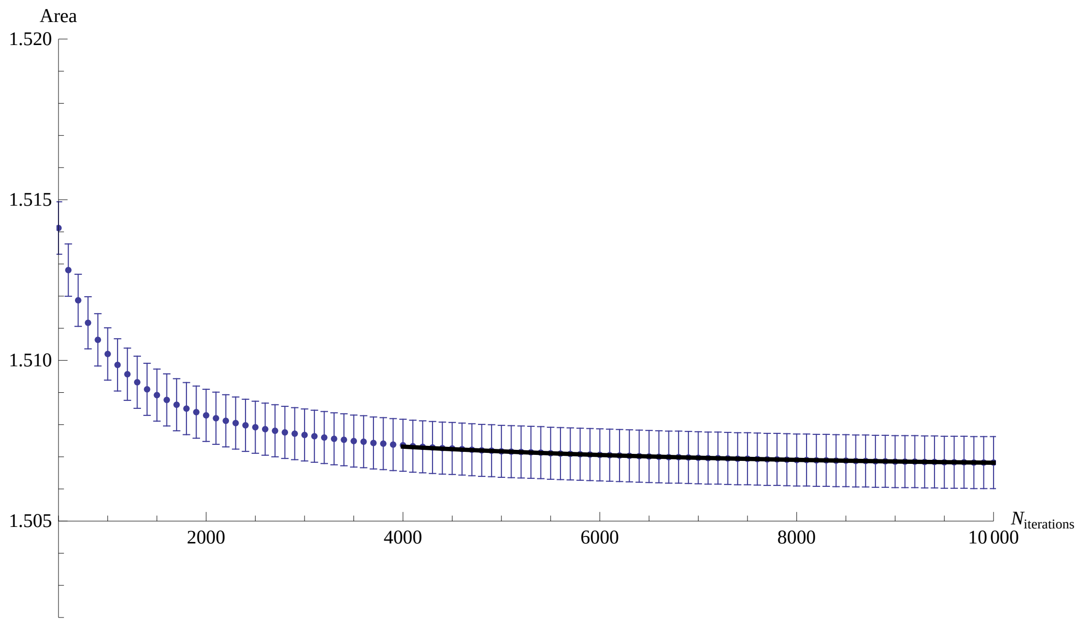

Figure 2: The mean area with error estimates as a function of the iteration i obtained by using a random sampling method and 10^6 samples. The thick black line is obtained by fitting the exponential function A + Bexp(Ci) to the data.

The function is fitted to the data using Mathematica, specifically the function NonLinearModelFit. One of the advantages of this function is that it takes the errors in the data into account when estimating the parameters and the error in the parameters of the function. If we let $i \to \infty$ the exponential term drops out (provided C is negative) and we obtain a value for the asymptote, equal to A. We can repeat this procedure for different sample sizes and different sample methods to obtain in every case a value for the parameter A and a corresponding value of the error of that parameter, taking into account the data errors.
Figure 3 shown below gives the asymptote values for the different sample methods with uncertainty (again taking a confidence interval of 99%) as a function of sample size. The black points correspond to random sampling, the red points to latin hypercube sampling and the blue points to orthogonal sampling. The horizontal line corresponds to the most accurately known values at the moment, 1.50659177. To make the graph more clear the points are not placed at the exact same sample size, however the calculation itself was performed on the exact sample sizes mentioned above. One of the differences that is immediately seen is that the error is much smaller when the orthogonal method is used than in the other two cases. Also, the value of the orthogonal method is already accurate for small sample sizes while for orthogonal or latinhypercube sampling this is not the case.

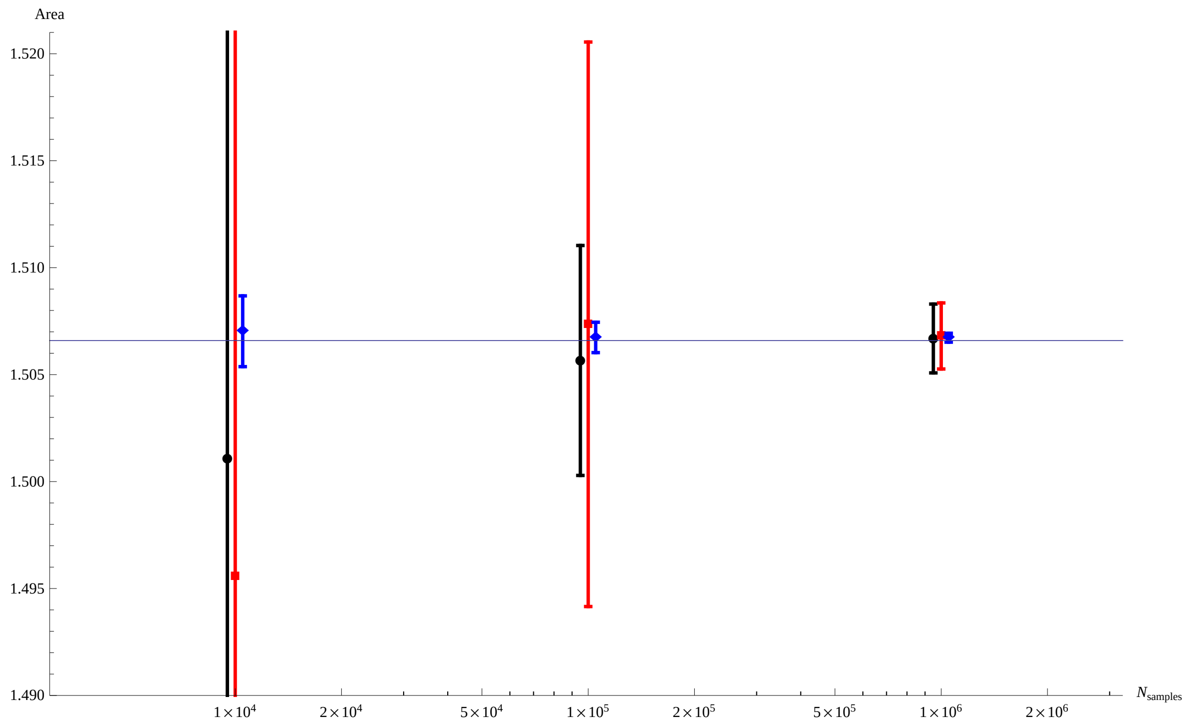

Figure 3: The asymptote values for different sample sizes and sampling methods. The black points correspond to random sampling, the red points to latinhypercube sampling and the blue points to orthogonal sampling.

It is important to remark that the outcome of the asymptote values themselves are a stochastic variable, so by repeating the whole procedure one could obtain very different values for the asymptotes. Especially since we take $N_{\text{runs}} = 100$, repeating the whole procedure could give a very different outcome. However regardless of the specific asymptote values that are obtained, the error in the orthogonal method is much smaller than the error in the random sampling or latin hypercube sampling (this has been tested). Also using the orthogonal method the asymptote value approaces the best know value best. By the arguments above the orthogonal method will be used to estimate the area of the Mandelbrot set. The numerical values of the figure above for the orthogonal method are:

Sample Size	Asymptote Value	Error
10^4	1.50703	0.00165
10^5	1.50674	0.00071
10^6	1.50673	0.00021

Table 1: The asymptote values from figure 3 for orthogonal sampling for different sample sizes including the error with a 99% confidence interval.

So our final estimate for the area of the Mandelbrot set is $A_\infty = 1.50673 \pm 0.00021$. The best known value is within the confidence interval of 99% of this value.

5 Discussion

There are a few remarks to be made about this project. We used the method described in the section 'Methods' to calculate the asymptotic values. From experimenting with other values of I_{max} it turns

out that the accuracy with which one can get close to the best known value increases as the maximum number of iterations increases, which is expected, because the fit becomes more accurate. However, due to time limitations, we were only able to test this up to 10000 iterations, since we also need a significant number of samples and runs to get reasonable good statistics.
Secondly it is not clear from the documentation provided by Mathematica, what exactly the NonLinear-ModelFit does with the errors in the data to get an error in the parameter and hence in my case, the error for the asymptote. The error scales however with the square root of the number of points that are taken into account for the fit.
Thirdly, since I didn't use that many runs, calculating the asymptote values can differ greatly from one calculation to the next, since the asymptote value is a stochastic variable itself. For instance, it has happened that the asymptote value for 10^4 samples and using random sampling gave almost exactly the same value as the best known value. However, as mentioned above, even though the asymptote was almost exactly right, the error remained very large, so there is a large spreading in area from one run to the next, but the mean ended up 'on accident' on the best known value.
Another remarkable issue I have found is that the difference between random sampling and latin hypercube sampling is not very large in terms of uncertainty in the final asymptote values. The method that is clearly much more precise is the orthogonal sampling. This error analysis was consistent en does not vary between experiments like the asymptote values themselves can.

In the model that I have used there is room for improvement. Firstly I was constrained by time and only having access to my own laptop for the computation. I had to balance between enough samples, enough runs for the statistics and large enough iterations to get a good final estimate. The combined running time of my program for all three sampling methods using 10^6 samples, 100 runs and 10000 iterations was slightly less than an hour.
A possible improvement on the model is decreasing the area in which the samples are taken. Figure 1 shows that there are large areas for which it is clear, for sufficiently high iteration level, that the area is inside or outside the Mandelbrot set. The sampling could be focused on the border area, using the same procedure as used above and then adding the area of the geometric shape that is not considered for sampling but is in the Mandelbrot set.

References

[1] http://www.mrob.com/pub/muency/pixelcounting.html

[2] Ross, S.,*Simulation*, 4th edition, 2006