Area Of The Mandelbrot Set By Monte Carlo Method

Final Project Computational Physics II

William Jones

April 2021

One of the most beautiful and fascinating fractal in complex dynamics. The Mandelbrot set has been produced from students to professionals since its first being seen by Beniot Mandelbrot. It is apart of the field of chaos theory, which is how in dynamics states of disorder and chaos can follow patterns and laws that have volatile conditions. One of the disorder systems that falls under chaos theory is fractals. These complex patterns are never ending images of dynamic complex systems. They are ever present in daily life from leaves to mountains all can be made by fractals. As stated above we are going to be looking at the most famous fractal, the Mandelbrot set, and apply the Monte Carlo method to find the area of the Mandelbrot. Furthermore, checking how changing the iteration limit and the number of points affects our calculation of it.

Fractal comes from the word fractus which in Latin means 'broken'/'fractured'. This coins the term perfectly as all fractals are made of fractional parts. While they very widely in size and shapes all are governed by few laws which are

- Self similarity: contains the original fractals at different scales within the fractal

- Irregular at any scale and not easily described by Euclidean space
- Iterative formula
- Hausdorff dimension greater than topological dimension

The Mandelbrot set is a is a very complex fractal that is never ending coming from a simplistic formula, zooming in on itself at any point will bring out smaller and smaller images of fractals within the Mandelbrot. Some prominent features is the main carotid that lies at the center with infinitely many bulbs attached along the outside. From the fractal it makes up the Julia and Fatou sets within itself. These can be seen while magnifying a point on the Mandelbrot, though the images will not always look the same as Julia and Fatou sets differ depending on the complex number. The equation for the Mandelbrot set is

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

where c is the complex number a + bi, i = $\sqrt{-1}$. In the set there are several complex numbers that when iterated from z = 0 remains bounded and doesn't diverge. The bounds of the Mandelbrot are located inside [-2, 2] and [-2i, 2i] in the complex plane.

To perform the calculation of finding the area of the Mandelbrot it was decided that the best approach was the Monte Carlo Method. Due to the Mandelbrot's nonlinear and rough shape its difficult to calculate the area by a formula alone, this is why we have chosen the Monte Carlo Method. It is a collection of samplings that use random numbers to find the desired outcome.Since the Mandelbrot is vast and continuous at smaller and smaller scales the Monte Carlo can give a approximate result using less number of points but to a higher accuracy than going point by point. Specific for this project we are using the hit or miss integration of the Monte Carlo. It can be told as

$$f(x) >= 0, x \subseteq [a, b] \tag{2}$$

essentially being

$$I = \int_a^b f(x)dx \tag{3}$$

the integration is telling us that I, the area of interest, is bounded on the left by a and right by b. The bottom is bounded by the x-axis thus leaving us with the top bound, where we have σ being the max(f(x)). We can then take the found variables to obtain the probability of point being inside the area of interest I

$$R = \frac{I}{(b-a)\sigma} \tag{4}$$

This can be further simplified to

$$R \approx \frac{N_I}{N} \tag{5}$$

where N is the number of data points ran through the Monte Carlo Method and N_I is the number of data points that fall inside of the area of interest.Taking this we can multiply it by the total area of space A_s to get the area of f(x)

$$A = A_s \frac{N_I}{N}, A_s = (b-a)\sigma \tag{6}$$

Taking the Monte Carlo Method and applying it to the Mandelbrot Set we can get a approximate area of it. using the known equation from the Mandelbrot

$$f_c(c) = z^2 + c, c \subseteq [-2, 2] \tag{7}$$

for both imaginary and real parts of c. Using the previous predefined algorithm for Monte Carlo we choose a set number of points P, in each point we choose a random point in the subset [-2,2] for the real component a in c = a + bi, like wise we do the same for b in the imaginary plane. from there we run the normal Mandelbrot. Given our random points x,y we insert them into c, x+yi. Then run the function for set max iterations,N, which says that if the point doesn't diverge for the whole iteration scheme this point will reach infinity thus being a point lying in the Mandelbrot,

$$f_c^N(c) = z_N^2 + c, f_c^N(c) \approx z_{n+1} \tag{8}$$

We start of with a $z_0 = 0$ and each calculation of z_{n+1} becomes the z_n for the next iteration. If the point is in set we store its value into a array and if its not out of the set we discard the data point and move onto the next. Now, as stated equation(7) the Mandelbrot lies in the complex plane [-2,2] for both axis thus helping us solve the height σ = (c, d) making the Area of interest

$$A = (b-a)(d-c)\frac{P_m}{P} = 16\frac{P_m}{P} \tag{9}$$

where P_m is the number of points that hit the max iteration. In the Code we explore how changing the max iteration limit with fixed number of points affects the area approximation $N = [15, 30, 45, 60, 85, 90, 105, 120, 135]$ with $P = 100,000$. Secondly, we see how changing the number of data points $P = [4, 16, 64, 1024, 4096, 16384, 65536, 262144]$ for the fixed max iteration limit $N = 75$. Once the change in variables is complete we can compare it to the true area of the mandelbrot set $\approx$ 1.507 founded in 2000.

Code:

```
%monte carlo
```

```
trueArea = 1.507;%True Approximate Area of the Mandelbrot
points = 100000; %number of data points
areas = zeros(9, 1);%setting array for changing iteration limit mc
iterc = zeros(9, 1);%setting array for iteration limit
%getting different iteration limits
for i = 1:9
    nIter = 15 *i;%iteration limit
    iterc(i) = nIter;%storing limit
    areas(i) = mc(points, nIter);%calculating the Area from MC
end
%plot for Mandelbrot
mandelbrott(points, 100);
%plotting MC for changing iteration limit
figure(9)
plot(iterc, areas);
title('MC Area from Limit Change')
xlabel('Iteration Max')
ylabel('Area')
areapc = zeros(9, 1);%setting array for point change monte carlo area
pointpc = zeros(9, 1);%setting array for points
for i =1:9
    pchange = 4^i;%points
```

```
    pointpc(i) = pchange;%storing value of poinnts into array
    areapc(i) = mc(pchange, 100);%storing MC area result into array
end
%plotting the MC from Point Change
figure(10)
plot(pointpc, areapc)
title('MC from Changing Data Points')
xlabel('# points')
ylabel('Area')
errorN = abs((areas - trueArea)./trueArea)*100;%percent error calc for MC lim
errorP = abs((areapc - trueArea)./trueArea)*100;%percent error calc for MC po
%plot for error of limit change
figure(11)
plot(iterc, errorN)
title('Error of Iteration Limit Change Monte Carlo')
xlabel('Iteration Limit')
ylabel('% error')
%plot for error of point change
figure(12)
plot(pointpc, errorP)
title('Error of Point Change Monte Carlo')
xlabel('# of Points')
```

```
ylabel('% error');

%function for mandelbrot plotting
function mandelbrott(points, N)
    step = 4/(points./2);%step size
    r = zeros(points/2,1);% set array for real points
    im = zeros(points/2, 1);%set array for imaginary points
    manr = NaN(points, 1);%set array for real points in Set
    mani = NaN(points,1);%set array for imaginary points in Set
    indexm = 1;%index value for in Set
    outr = NaN(points, 1);%set array for real points outside set
    outi = NaN(points, 1);%set array for imaginary points outside set
    indexo = 1;%index value for out set
    %getting points for graph in [-2, 2]
    for i = 1:(points/2)
        im(i) = -2 + i*step;%imaginary
        r(i) = -2 + i*step;%real
    end
    %Mandelbrot iterative formula
    for j = 1:(points/2)%real movement
        for k = 1:(points/2)%imag movement
            z = 0;%starting z
```

```
            i = 1;% iterative starting value
            c = complex(r(j), im(k));%complex number
            %iterative method
            while (abs(z) < 2 && i < N)%as long as z is in [-2, 2] and under
            z = z^2 + c;%formula
            i = i + 1;%increment
            end
            %if hitting iteration limit assume infinity and is in set
            if i == N
                manr(indexm) = real(c);%store real number
                mani(indexm) = imag(c);%store imag number
                indexm = indexm + 1;%increment
            end
            %No Limit met so point is outside set
            if i ~= N
                outr(indexo) = real(c);%store real number
                outi(indexo) = imag(c);%store imag number
                indexo = indexo + 1;%increment
            end

        end
    end
```

```
    figure(1)
    scatter(manr, mani,[], 'blue', 'filled')%plotting the points in set
    hold on
    scatter(outr, outi,[], 'black', 'filled')%plotting points outide set
    hold off
    title('Mandelbrot')
    xlabel('Real')
    ylabel('Imaginary')
end
%Monte Carlo Hit Or Miss Method
function area = mc(points, N)
    mandelbrotmc = NaN(points,N);%array set up for monte carlo mandelbrot
    r = -2 + (2+2) * rand(points, 1);%random real points
    im = -2 + (2+2) *rand(points, 1);%random imaginary points
    a = NaN(points, 1);%real array set up
    b = NaN(points, 1);%imaginary array set up
    count = 1;%count iter
    max = 0;
    %iterative Mandelbrot calculation
    for i = 1:points
        c = complex( r(i), im(i));%setting complex number
        mandelbrotmc(i,1) = 0;%storing value
```

```
        k = 1;%iteration set
        mandelbrotmc(i, k) = 0;%storing intial z
        while( abs(mandelbrotmc(i, k)) < 2 && k < N)%run unitl outside bounds
            mandelbrotmc(i, k+1) = mandelbrotmc(i, k).^2 + c;%mandlebrot equa
            k = k + 1;%increment
            %Limit Hit( in Set)
            if k == N
                a(count) = real(mandelbrotmc(i, k));%store real value
                b(count) = imag(mandelbrotmc(i, k));%store imaginary value
                max = max + 1;%increment count
            end
            count = count +1;%tally of inset
        end

    end
    area = 16 * (max./points);%area calculation
    %%%%%%%%%%%FOR PLOTTING CHANGES%%%%%%%%%%%%%%
    %for Limit change being 45
    if N == 45
        figure(2)
        for i = 1:points
            if ~isnan(a(i)) %making sure value contained
```

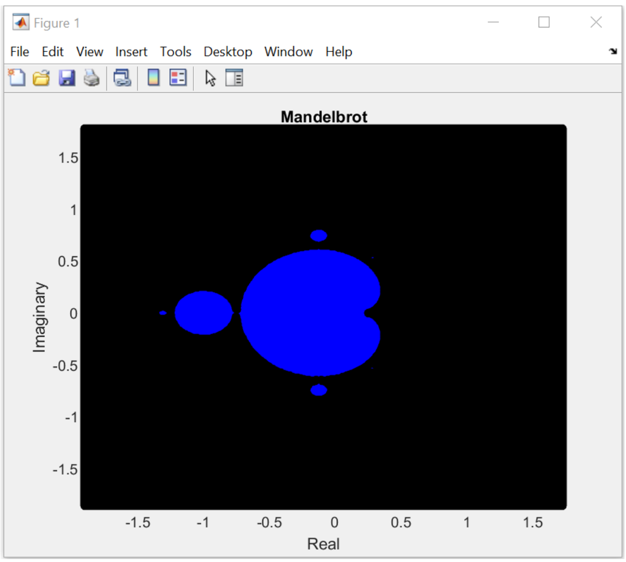
```
                plot(a(i), b(i), '.')
            end
            title('Monte Carlo Mandelbrot 45 iteration limit')
            xlabel('real')
            ylabel('imaginary')
            hold on

    end
    hold off
end
%for limit being 90
if N == 90
    figure(3)
    for i = 1:points
        if ~isnan(a(i))  %making sure value contained
            plot(a(i), b(i), '.')
        end
        hold on
         title('Monte Carlo Mandelbrot 90 iteration limit')
        xlabel('real')
        ylabel('imaginary')

```

```
        end
        hold off
    end
    %for limit being 135
    if N == 135
        figure(4)
        for i = 1:points
            if ~isnan(a(i))  %making sure value contained
                plot(a(i), b(i), '.')
            end
            title('Monte Carlo Mandelbrot 135 iteration limit')
            xlabel('real')
            ylabel('imaginary')
            hold on

        end
        hold off
    end
    %for point change of 64
    if points == 4^3
        figure(5)
        for i = 1:points
```

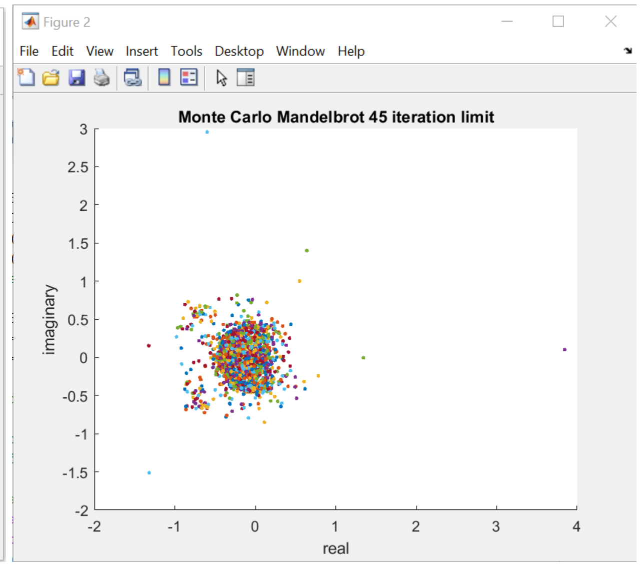
```
            if ~isnan(a(i))  %making sure value contained
                plot(a(i), b(i), '.')
            end
            title('Monte Carlo Mandelbrot for Change in Points: 64')
            xlabel('real')
            ylabel('imaginary')
            hold on

        end
        hold off
    end
    %for points change of 4096
    if points == 4^6
        figure(6)
        for i = 1:points
            if ~isnan(a(i))  %making sure value contained
                plot(a(i), b(i), '.')
            end
            title('Monte Carlo Mandelbrot for Change in Points: 4096')
            xlabel('real')
            ylabel('imaginary')
            hold on
```

```

        end
        hold off
    end
    %for point change of 262144
     if points == 4^9
        figure(7)
        for i = 1:points
            if ~isnan(a(i))  %making sure value contained
                plot(a(i), b(i), '.')
            end
            title('Monte Carlo Mandelbrot for Change in Points: 262144 ')
            xlabel('real')
            ylabel('imaginary')
            hold on

        end
        hold off
    end
end
```

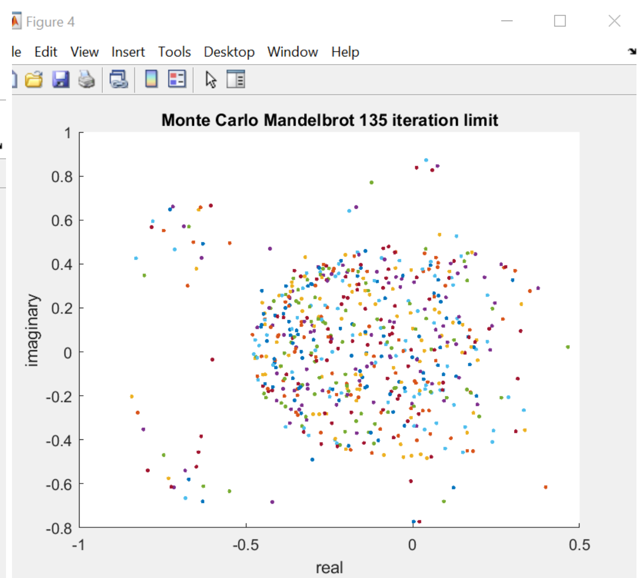
Figures(1):

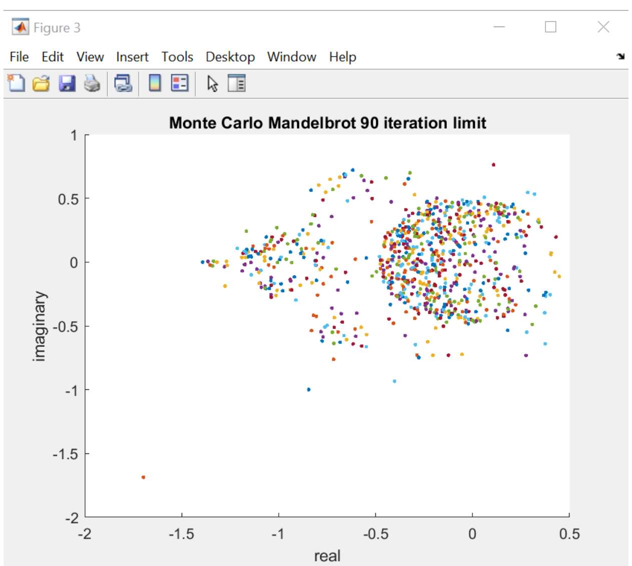


(a) The Mandelbrot Set



(b) Monte Carlo 100,000 Points 45 Iteration Limit

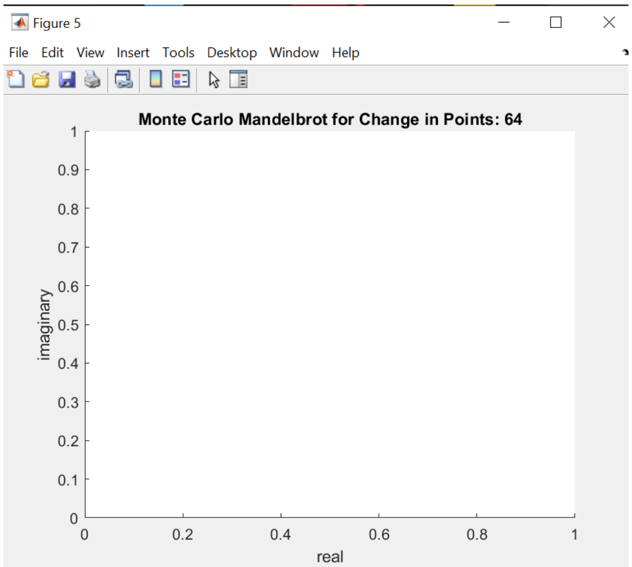




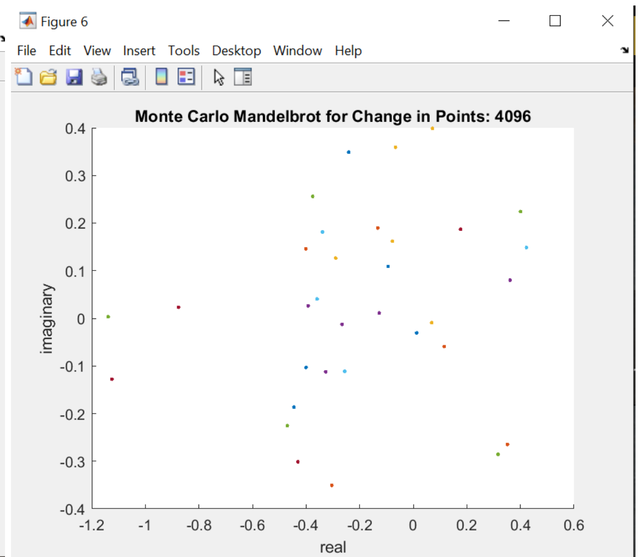
(d) Monte Carlo 100,000 Points 135 Iteration

(c) Monte Carlo 100,000 Points 90 Iteration Limit Limit

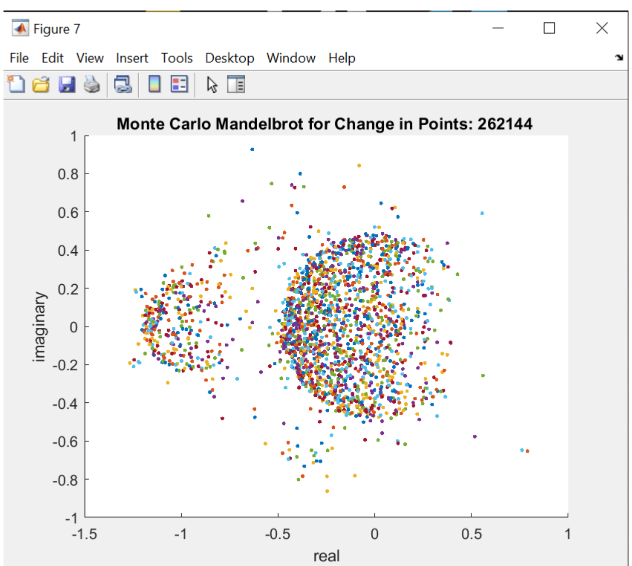
Figures(2):



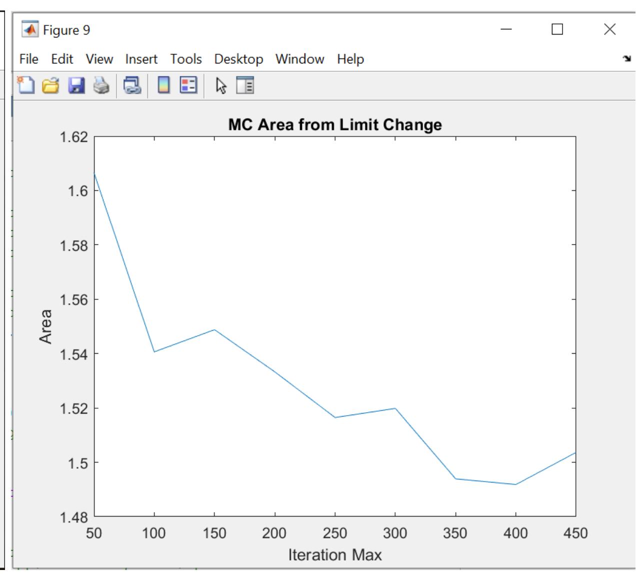
(a) Monte Carlo 64 Points 100 Iteration Limit



(b) Monte Carlo 4096 Points 100 Iteration Limit

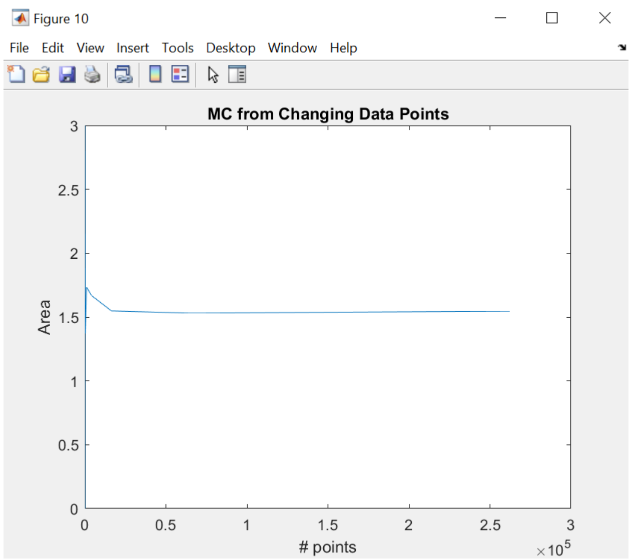


(c) Monte Carlo 262,144 Points 100 Iteration

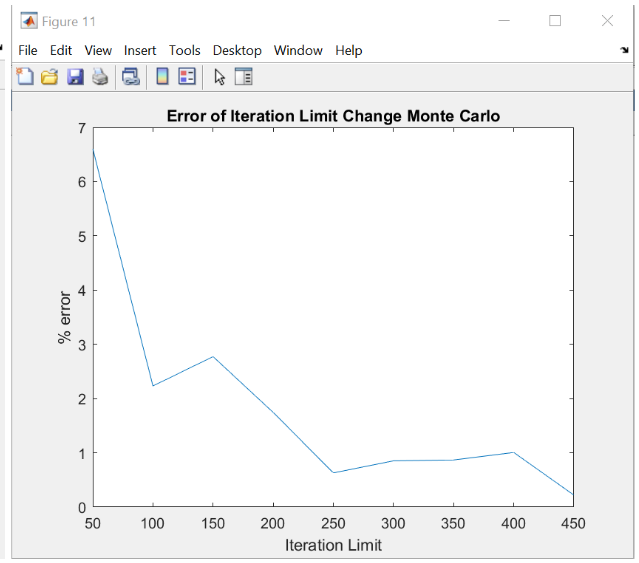


(d) Monte Carlo Area for Iteration Change

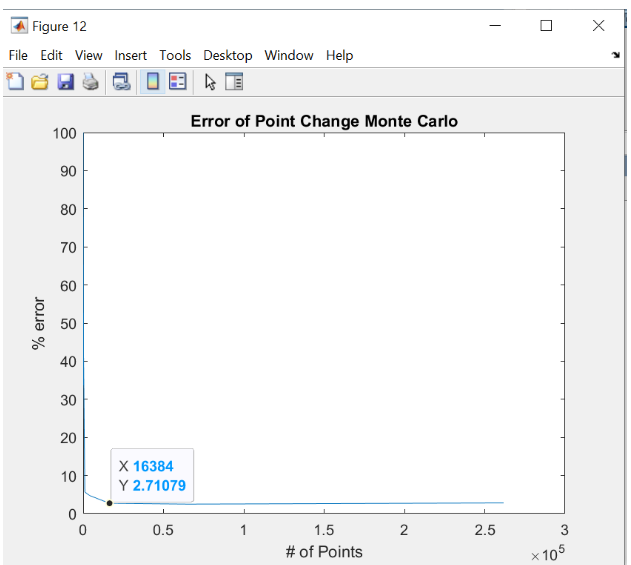
Figures(3):



(a) Monte Carlo Area for Point Change



(b) Monte Carlo Iteration Change Error Plot



(c) Monte Carlo Point Change Error Plot

In analysis of the code we can see that for the changing iteration limit, as the limit increased the more the number of points decreased as shown in Figures(1): (b), (c), (d). However the area decreased as well and converged closer to the true area, while most the areas fell within the acceptable error to the true area the only concerning one is 45 iteration limit as it has about 65 percent error Figure(3):(b). Seeing this the best number for a iteration limit would have to be 100 or greater to give the best approximate result. Furthermore, as the iteration increases it seems that the error follows suit as well and decreases. Now for Changing the number of points with the fixed iteration limit of 100 the results were surprising, while at 64 data points failed to have a point in set of the Mandelbrot, the others plotted well with them containing low amounts of error in the acceptable range. Looking from the plots it appears that changing the number of data points in the Monte Carlo Method seemed to have little effect except for relatively low numbers such as 64. However, the iteration limit seemed to have greater effect on changing the outcome of area.

Overall this Method proved quite useful for the Mandelbrot as it took less data points to get the area compared to running point by point in [-2, 2] saving time and memory space. One way to change this code would be to better optimize it to improve the time calculations while the method itself too little to no time, the plotting of some runs too significant amount of time.

Reference

Munafo, R. ”Area of the Mandelbrot Set.”

https://www.mrob.com/pub/muency/areaofthemandelbrotset.html.

Schober, Glenn, and Ewing, John H.. ”The area of the Mandelbrot set..” Numerische Mathematik 61.1 (1992): 59-72. ¡http://eudml.org/doc/133610¿.

Devaney, Robert L. “Unveiling the Mandelbrot Set.” Plus.maths.org, 21 Aug. 2020, plus.maths.org/content/unveiling-mandelbrot-set.

Peitgen, Heinz-Otto, et al. Chaos and Fractals: New Frontiers of Science. Springer, 2012.