Final Project Report

Seth Richards

Bio Eng 217

For my final project in this course, I chose to explore Mandelbrot set generator. A mandelbrot set is a plot of points in the complex domain colored based on a mathematical definition that separates each point as black or white (in basic sets) based on whether the recursive square of that point approaches infinity or zero, In the real domain, any point whose position is equal to or greater than one discriminated in accordance with that definition will approach infinity. This means that the plot would look like a perfect sphere, with a constant radius of one. This is not the case in the complex domain, as the square of an imaginary number returns a negative number, meaning that it’s plot has a very strange shape with uneven radii, and actually results in fractal-like behavior. My goal for this project was to write MATLAB code to be able to evaluate any square area in the complex plane and generate a Mandelbrot plot.

Initially, I set out to encode a program to do a very basic and limited but functional set. I started by creating a function that would accept a square area in the complex domain. Inside it, I made a for loop that looped a random number of times. Within the for loop, I coded in the definition of the set by taking each point, and squaring in through each iteration of the for loop, and then testing to see if it was larger than an arbitrary number. If it was, that meant the value approaches infinity since it got much larger than it’s single square, so I colored the point white. Other points were colored black. But considering I just took a MATLAB class, I did that math the MATLAB way by doing this for all points at once with element wise squaring of the whole area in one matrix. Then, I used a logical expression with a thresholding value to get my simple matrix, which I returned to display. Since the matrix to display was just ones and zeros, I used the function imagesc() to display with high contrast to make the 1’s full white, and the 0’s full black. To get good image quality, I played around with the number of points, number of loops, and the threshold value. I later made a version of this function that would “zoom in” by using several preset areas with experimentally derived constants. This process was difficult and very time consuming, so if I were to continue this project in the future, I would use edge detection to automatically zoom in on a desired point’s area, and estimate certain variables for image quality.

Then, I saw that some Mandelbrot plots had been created and colored based on how fast they approached infinity, meaning that as a point got closer to a “black” area, it would be color coded a “closer” color on a color scale. I thought about doing some fancy math with derivatives, but I didn’t do that for my set, so I decided to do a simple implementation. In my solution, I added another matrix inside the function’s for loop, and simply added each iteration of the original returned logical matrix. This meant that points that exceeded the threshold in the earlier iterations of the for loops corresponded to a greater number in the new matrix, which was higher on the color scale. As the for loop looped around more, more points closer to the black mass eventually reached above the threshold, resulting in color bands that got closer together as the points got closer to the mass, simulating the rate of change in the set plot. At this point, the coronavirus hit, and my machine really struggles with MATLAB, so I couldn’t do much more optimizing and I cleaned up my code into a finished project.

% Seth Richards

% BIOEN 217 A

% 03/13/2020

% Final Project

close all;clear all; clc;

%% Single Detailed Basic Mandelrot Plot

% This portion of the code makes a single mandelbrot set with very high

% detail using a simple function below and a preset area. This plot is

% black and white only.

% creates complex domain area

[X, Y] = meshgrid( -2:0.0007:2 , -1.25:0.0007:1.25);

s = X + 1j\*Y;

% implements function

detailedPlot = mandelBrotPlot(s) \* 1.5;

% creates image

figure(1)

colormap(gray);

% Using imagesc to show more contrast as matrix is only 1s and 0s not 0-256

imagesc(detailedPlot);

xlabel('Real Axis)');

ylabel('Imaginary Axis');

title('Mandelbrot Plot');

% adds correct labels to match the complex domain above

xticklabels = -2:0.5:2;

xticks = linspace(1, size(detailedPlot, 2), numel(xticklabels));

set(gca, 'XTick', xticks, 'XTickLabel', xticklabels)

yticklabels = -1.25:0.25:1.25;

yticks = linspace(1, size(detailedPlot, 1), numel(yticklabels));

set(gca, 'YTick', yticks, 'YTickLabel', flipud(yticklabels(:)))

%% Single Detailed Colored Madlebrot Plot

% This portion of the code makes a single mandelbrot set with very high

% detail using a simple function below and a preset area. This plot is

% colored based on how fast each point approaches infinity simulated and

% displayed by a colorscale

% creates complex domain area

[X, Y] = meshgrid( -2:0.0007:2 , -1.25:0.0007:1.25);

s = X + 1j\*Y;

% implements function

detailedPlot = mandelBrotPlotRainbow(s) \* 50;

figure(1)

imagesc(detailedPlot);

xlabel('Real Axis)');

ylabel('Imaginary Axis');

title('Mandelbrot Plot');

%labels axes

xticklabels = -2:0.5:2;

xticks = linspace(1, size(detailedPlot, 2), numel(xticklabels));

set(gca, 'XTick', xticks, 'XTickLabel', xticklabels)

yticklabels = -1.25:0.25:1.25;

yticks = linspace(1, size(detailedPlot, 1), numel(yticklabels));

set(gca, 'YTick', yticks, 'YTickLabel', flipud(yticklabels(:)))

%% Multiple Detailed Basic Madlebrot (Zoom)

% This portion of code creates a mandlebrot plot, and simulates zooming in

% by creating several smaller plots on top of each other

% zoom constants, where index corresponds to loop number implemented

zoomXL = [-1.75,-1.1,-0.9, -0.7];

zoomXU = [0.95,0.05,-0.1, -0.3];

zoomYL = [-1.75,-1.4,-1.0, -0.8];

zoomYU = [0.95,0.55,0.2,-0.3];

scale = [0.001, 0.0009, 0.0005,0.00005];

loop = [12, 16, 19, 35];

magnitude = [1.5, 1.5, 1.4, 1.3];

for k = 1:4

% pause for dramatic effect

pause(5)

% creates complex domain area from specified constraints

[X1, Y1] = meshgrid( zoomXL(k):scale(k):zoomXU(k) , zoomYL(k):scale(k):zoomYU(k) );

s = X1 + 1j\*Y1;

% implements function

imageK = mandelBrotPlotZoom(s,loop(k),magnitude(k))\*1.5;

% creates image for each loop iteration

figure(k)

colormap(gray);

imagesc(imageK);

xlabel('Real Axis)');

ylabel('Imaginary Axis');

title('Mandelbrot Plot');

% Having some problems with labelling axis at certain values

if k <= 5

xticklabels = zoomXL(k):zoomXU(k);

xticks = linspace(1, size(imageK, 2), numel(xticklabels));

set(gca, 'XTick', xticks, 'XTickLabel', xticklabels);

yticklabels = zoomYL(k):zoomYU(k);

yticks = linspace(1, size(imageK, 1), numel(yticklabels));

set(gca, 'YTick', yticks, 'YTickLabel', flipud(yticklabels(:)))

end

end

%%

function [imageArray] = mandelBrotPlot(s)

% creates a mandlebrot plot by doing equation on an array multiple times,

% and testing whether each point goes to infinity by threshold

% input s - complex plane on which to create mandlebrot plot

% output imageArray is mandelbrot set

imageArray = zeros(size(s));

n = 0;

for k = 1:22

% mandelbrot definition

n = (n.^2) + s;

% thresholding with logical array

imageArray = imageArray > 1 | abs(n) > 1.5;

end

end

%%

function [imageArrayDiff] = mandelBrotPlotRainbow(s)

% creates a mandlebrot plot by doing equation on an array multiple times,

% and testing whether each point goes to infinity by threshold

% creates arrray of colors based on how how few loops it takes for point

% to exceed threshold

% input s - complex plane on which to create mandlebrot plot

% output imageArrayDiff is mandelbrot set with colors based on rate

imageArray = zeros(size(s));

imageArrayDiff = zeros(size(s));

n = 0;

for k = 1:20

% mandelbrot definition

n = (n.^2) + s;

imageArray = imageArray > 1 | abs(n) > 2;

% expression adding for each loop to show rate of approach to

% infinity

imageArrayDiff = imageArrayDiff + imageArray;

end

end

%%

function [imageArrayZoom] = mandelBrotPlotZoom(s,loopVar,MagnitudeVar)

% creates a mandlebrot plot by doing equation on an array multiple times,

% and testing whether each point goes to infinity by threshold

% input s - complex plane on which to create mandlebrot plot

% input loopVar - determines number of times array is squared

% input s - determines threshold value for plot

% output imageArrayZoom is mandelbrot set based on input zoom variables

imageArrayZoom = zeros(size(s));

n = 0;

for k = 1:loopVar

% madelbrot definition

n = (n.^2) + s;

% logical expression for threshold

imageArrayZoom = imageArrayZoom > 1 | abs(n) > MagnitudeVar;

end

end