Project 1: Fractal Geometry
MAT128B Winter 2020

Caitlin Brown, Nikos Trembois, and Shuai Zhi

February 14, 2020

Contents

1 Introduction

In this project, numerical analysis is used to understand and demonstrate fractals and their characteristics. The fractals will be generated from the orbits of complex functions. Orbits are the sequence of numbers that results from the process of applying a function to the output of the same function over and over, like a recursive function. So $orb(z_0) = z_0, z_1 = \phi(z_0), z_2\phi(\phi(z_0))...$ is the orbit of the initial point z_0 under the function ϕ. It is not hard to imagine that for certain initial values this process will diverge while other initial values will converge, or at least remain bounded by some value. The filled Julia set is all points whose orbit, using a polynomial function, remains bounded. The boundary of the filled set is called the Julia set.

2 Introduction to Fractals

The orbit of complex values whose real and imaginary part were within [-1, 1] were calculated for the function $\phi(z) = z^2$. The filled Julia set of $\phi(z) = z^2$, shown in figure 1, shows the map the orbit, which is the unit disk. Under the same conditions the Julia set would create the unit circle, as it is the boundary of the filled Julia set.

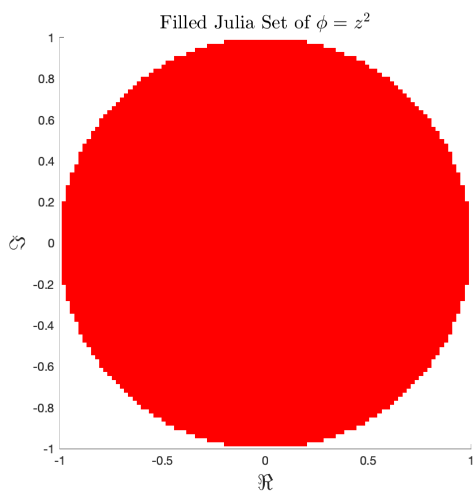
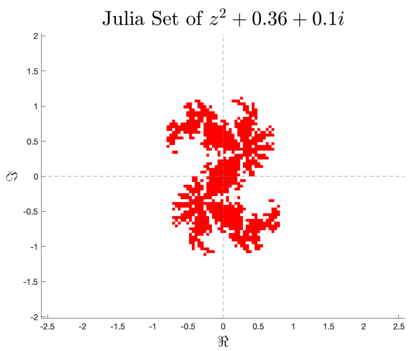
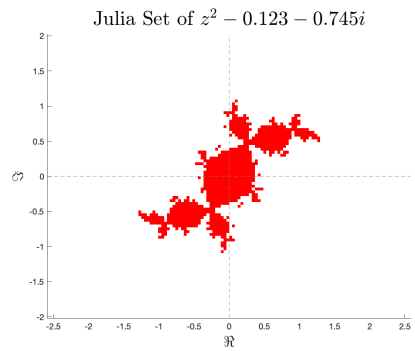


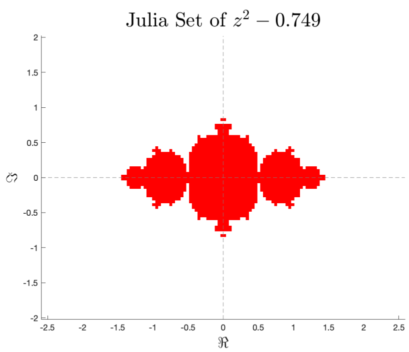
Figure 1: Orbit of z under $\phi = z^2$



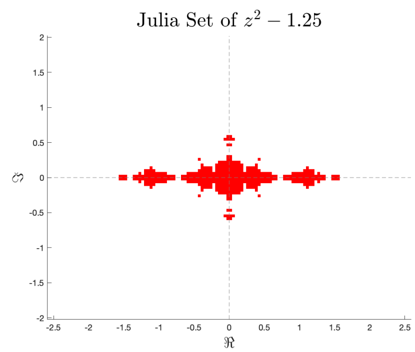
(a) Filled Julia Set of $z = 0.36 + 0.1i$



(b) Filled Julia Set of $z = -0.123 + 0.745i$



(c) Filled Julia Set of $z = -0.749$



(d) Filled Julia Set of $z = -1.25$

Figure 2: Filled Julia Sets with 100 points in real and imaginary axis

By adding a constant to the function (i.e. $\phi(z) = z^2 + c$) the map of the orbit becomes more complex shapes, which turn out to be fractals.

3 Julia Set

Figure 2a

4 Fractal Dimensions

5 Julia Set Connectivity

6 Divergent Orbits

7 Newton's Method in Complex Plane

Root finding can be a surprisingly difficult task. The linear case is trivial and roots of second order polynomials are solved with the quadratic equation. However, as the order increases the analytic equations to solve for the roots become more intricate and no known analytic equation exists for polynomials of orders higher than 6. Not to mention, this is just for real functions! Finding the roots of complex functions is even more difficult. Fortunately iterative methods, with the help of plots, simplify the process; although, some accuracy will be lost. The plots in figure 3 are not just interesting, but also insightful. When plotting the Newton fractals and coloring the points with the number of iterations required to converge, the location of the roots become evident. For Newton's method, and any iterative method I can think of, the closer an initial guess is to a root, the less iterations are required. So the roots are found where the plots indicate the least amount of iterations are, in this case dark blue. Looking at figure 3b it appears the roots around $(1, 0i), (-0.5, 0.86i), (0.5, -0.86i)$. This agrees with the known values of $(1, 0i), (-0.5, \frac{\sqrt{3}}{2}i), (-0.5, -\frac{\sqrt{3}}{2}i)$. The process extends to all subfigures in figure 3 and to any complex function whose roots are desired.

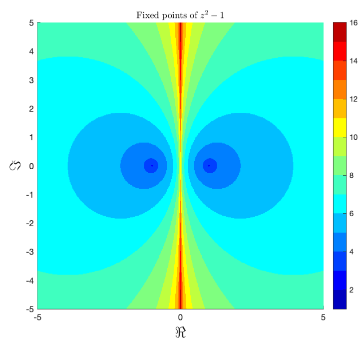
8 The Mandelbrot Set

9 Conclusion

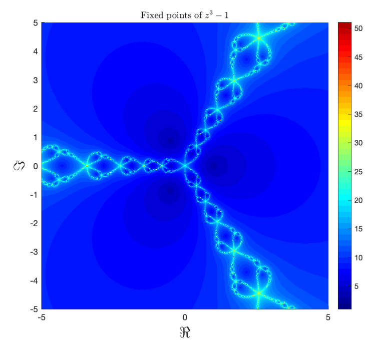
10 Appendix

10.1 Code

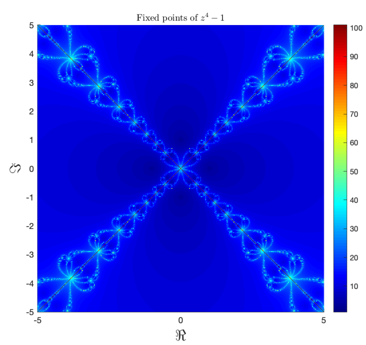
```
% MAT 128B: Project 1
% UC Davis Winter 2020
% Nikos Trembois, Caitlin Brown, and Shuai Zhi

global c
```



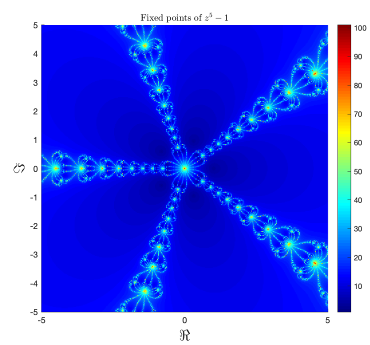
(a) Iterations to roots of $z^2 - 1$



(b) Iterations to roots of $z^3 - 1$



(c) Iterations to roots of $z^4 - 1$



(d) Iterations to roots of $z^5 - 1$

Figure 3: Roots of complex functions of form $z^n - 1$

```
c = [0.36 + 0.1i, -.123 - .745i,-.749, -1.25];

%% Part 1: Fractals
clearvars -except c
phi = @(z,c) z^2;
M = FilledJuliaSet(phi,1,1,100,0);

figure(); hold on
title('Filled Julia Set of $\phi = z^2$','Fontsize',16,'
    Interpreter','Latex')
xlabel('\Re','Fontsize',18)
ylabel('\Im','Fontsize',18)
colormap([1 0 0; 1 1 1]);
image( [-1 1], [-1 1], M')
axis xy
axis equal
axis([ -1 1 -1 1])
hold off
saveas(gcf,'../Figures/UnitDisk.png')

%% Part 2:  Fractals
clearvars -except c
phi = @(z,c) z^2 + c;
xrange = 2; yrange = 2;
M = cell(length(c),1);
for i = 1:length(c)
    M{i} = FilledJuliaSet(phi,xrange,yrange,100,c(i));
end

for i = 1:length(c)
    figure(); hold on
    if (real(c(i) > 0 ))
        titles = strcat('Julia Set of $z^2 + ',num2str(c(i)),'$');
    else
        titles = strcat('Julia Set of $z^2 ',num2str(c(i)),'$');
    end
    title(titles,'Interpreter','Latex','FontSize',24)
    colormap([1 0 0; 1 1 1]);
    image( [-xrange xrange], [-yrange yrange], M{i}')
    axis xy
    axis equal
    ax = gca;
    plot(ax.XLim,[0,0],'LineStyle','--','Color',[.5,.5,.5])
    plot([0,0],ax.YLim,'LineStyle','--','Color',[.5,.5,.5])
    xlabel('\Re','Fontsize',18)
    ylabel('\Im','Fontsize',18)
    hold off
    ssave = strcat('../Figures/FilledJulia',num2str(i),'.png');
    saveas(gcf,ssave)
```

```
end

%% Part 3: Julia Sets
clearvars -except c
psi = @(z,c) sqrt(z - c);
x = zeros(100,4);
y = zeros(100,4);
for k = 1:length(c)
    clear z;
    z = c(k);
    for j = 1:1000
        x(j,k) = real(z(j));
        y(j,k) = imag(z(j));
        if randi([0 1],1,1) == 1
            z(j+1) = psi(z(j),c(k));
        else
            z(j+1) = -psi(z(j),c(k));
        end
    end
end

for i = 1:length(c)
    figure(); hold on
    if (real(c(i) > 0 ))
        titles = strcat('Julia Set of $z^2 + ',num2str(c(i)),'$');
    else
        titles = strcat('Julia Set of $z^2 ',num2str(c(i)),'$');
    end
    title(titles,'Interpreter','Latex','FontSize',24)
    scatter(x(:,i),y(:,i),'filled')
    ax = gca;
    plot(ax.XLim,[0,0],'LineStyle','--','Color',[.5,.5,.5])
    plot([0,0],ax.YLim,'LineStyle','--','Color',[.5,.5,.5])
    xlabel('\Re','Fontsize',18)
    ylabel('\Im','Fontsize',18)
    hold off
    ssave = strcat('../Figures/Julia',num2str(i),'.png');
    saveas(gcf,ssave)
end

%% Part 4: Computing the Fractal Dimension

% FractalDimension(M{end},.02)

%% Part 5: Connectivity of the Julia Set
clearvars -except c
psi = @(z) z^2 + 3;
z = 0;
max_iter = 1000;
```

```
for i = 1:max_iter
    z = psi(z);
    if abs(z) > 100
        fprintf('The orbit diverged after %i iterations, the set
            is not connected\n',i)
        break
    end
end
if abs(z) < 100
    fprintf('The set did not diverge after %i iterations\n',
        max_iter)
    fprintf('It is reasonable to assume the Julia set is connected
        \n')
end


%% Part 6: Coloring Divergent Orbits
clearvars -except c
phi = @(z,c) z^2 - c;
rl = -1; ru = -rl; %1.6
il = -1;  iu = -il; %.7
a = linspace(rl,ru,100);
b = linspace(il,iu,100);
M = cell(length(c),1);
for k = 1:length(c)
    M{k} = ones(length(a),length(b));
    for r = 1:length(a)
        for i = 1:length(b)
            clear z;
            z = a(r) + 1i*b(i);
            for j = 1:100
                z(j+1) = phi(z(j),c(k));
                if abs(z(j+1)) > 100
                    M{k}(r,i) = j;
                    break;
                end
            end
        end
    end
end

for i = 1:length(c)
    figure(); hold on
    image( [rl ru], [il iu], M{i}')
    axis xy
    axis equal
    ax = gca;
    ax.XLim = [rl,-rl]; ax.YLim = [il,-il];
    plot(ax.XLim,[0,0],'LineStyle','--','Color',[.5,.5,.5])
```

```
    plot([0,0],ax.YLim,'LineStyle','--','Color',[.5,.5,.5])
    xlabel('\Re','Fontsize',18)
    ylabel('\Im','Fontsize',18)
    colormap(jet(max(max(M{i}))))
    colorbar
    hold off
    ssave = strcat('../Figures/ColoredJulia',num2str(i),'.png');
    saveas(gcf,ssave)
end

%% Part 7: Newton's Method in the Complex Plane
clearvars -except c
g = @(z,n) (z^n - 1)/(n*z^(n-1));
a = linspace(-5,5,500);
b = linspace(-5,5,500);
nmax = 5;
M = cell(nmax,1);
for n = 2:nmax
    M{n-1} = 100*ones(length(a),length(b));
    gn = @(z) (z^n - 1)/(n*z^(n-1));
    for r = 1:length(a)
        for i = 1:length(b)
            z = a(r) + 1i*b(i);
            for j = 1:100
                diff = z^n - 1;
                if abs(z^n-1) > 0.001
                    z = z - gn(z);
                else
                    if j < 3
                        fprintf('For n = %i, The root is near %2
                            .4f, %2.4f\n',n,a(r),b(i));
                    end
                    M{n-1}(r,i) = j;
                    break;
                end
            end
        end
    end
end

stitle1 = 'Fixed points of';
for i = 1:nmax-1
    figure(); hold on
    stitle2 = strcat( ' $z^',num2str(i+1),' - 1$');
    image( [min(a) max(a)], [min(b) max(b)], M{i}')
    colormap(jet(max(max(M{i}))))
    colorbar
    axis xy
    ax = gca;
```

```
    axis equal
    ax.XLim = [min(a) max(a)]; ax.YLim = [min(b) max(b)];
    title(strcat(stitle1,stitle2),'Interpreter','Latex')
    xlabel('\Re','Fontsize',18)
    ylabel('\Im','Fontsize',18)
    hold off
    ssave = strcat('../Figures/Newton',num2str(i),'.png');
    saveas(gcf,ssave)
end


%% Part 8: Mandelbrot Set
clearvars -except c
phi = @(z,c) z^2 + c;
a = linspace(-1,1,100);
b = linspace(-1,1,100);
M = ones(length(a),length(b));

for r = 1:length(a)
    for i = 1:length(b)
        clear z;
        z = 0;
        c = a(r) + b(i)*1i;
        for j = 1:100
            z(j+1) = phi(z(j),c);
            if abs(z(j+1)) > 100
                M(r,i) = j;
                break;
            end
        end
    end
end

figure(); hold on
title('Mandelbrot Set')
xlabel('Real')
ylabel('Imaginary')
colormap(jet(100))
image( [-1 1], [-1 1], M')
colorbar
axis xy
axis('equal')
axis([ -1 1 -1 1])
saveas(gcf,'../Figures/Madelbrot.png')
%% Part 8 (cont) Zoom in
% zoom in on a fractal by changing limits
% Weird it seems to change significantly when using higher
    resolution in
% this section from what is shown in the previous plot
```

```
phi = @(z,c) z^2 + c;
a = linspace(-1,-.8,100);
b = linspace(0,-.2,100);
clear M;
M = ones(length(a),length(b));

for r = 1:length(a)
    for i = 1:length(b)
        clear z;
        z = 0;
        c = a(r) + b(i)*1i;
        for j = 1:100
            z(j+1) = phi(z(j),c);
            if abs(z(j+1)) > 100
                M(r,i) = j;
                break;
            end
        end
    end
end

figure(); hold on
title('Mandelbrot Set')
xlabel('Real')
ylabel('Imaginary')
colormap(jet(100))
image( [min(a) max(a)], [min(b) max(b)], M')
colorbar
axis xy
axis('equal')

%% Functions
function [result] = R(x,y)
    result = sqrt(x^2 + y^2);
end

function [result] = T(x,y)
    if x > 0
        t = atan(y/x);
    elseif x == 0
        if y > 0
            t = pi/2;
        else
            t = 3*pi/2;
        end
    else
        t = atan(y/x) + pi;
    end
    result = t;
```

```
end

function FractalDimension(M,r)
    n = length(M);
    N = 0; % Initialize value of N for box-counting
    for i = 1:n
        for j = 1:n
            if M(i,j) == 1
                N = N+1;
            end
        end
    end
    d = log(N)/log(1/r);
    fprintf('The fractal dimension is %5.4f\n', d);
end

% This function determines if a Julia
function [ M ] = FilledJuliaSet(phi, xrange, yrange, pts, c)
    a = linspace(-xrange, xrange, pts);
    b = linspace(-yrange, yrange, pts);
    for k = 1:length(c)
        M = ones(length(a), length(b));
        for r = 1:length(a)
            for i = 1:length(b)
                clear z;
                z = a(r) + 1i*b(i);
                for j = 1:100
                    z(j+1) = phi( z(j), c(k) );
                    if abs( z(j+1) ) > 2
                        M(r,i) = 2;
                        break;
                    end
                end
            end
        end
    end
end
```