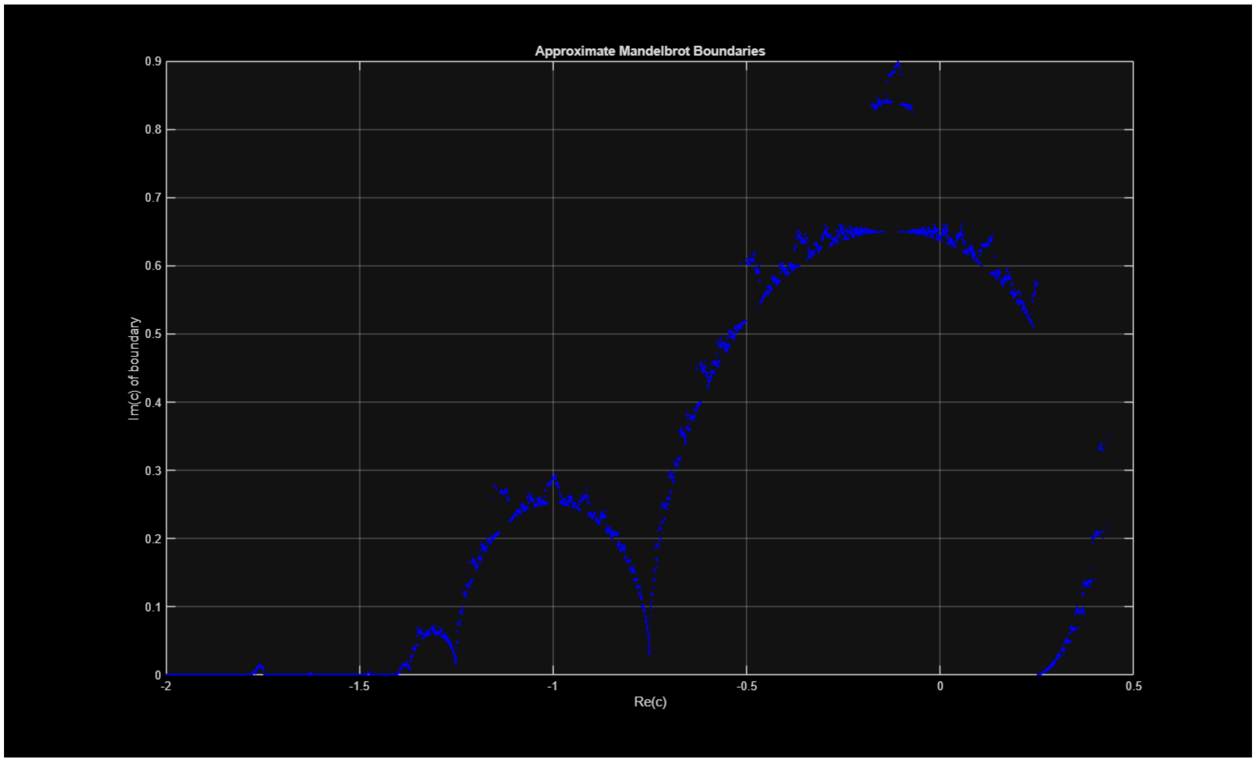
```matlab
%%Mandelbrot Boundary Approximation and Arc Length
%this approximates and plots a boundary for the Mandelbrot set boundary by
%scanning vertical slices in the complex plane. It then fits a polynomial
%of degree 15 to the non flat line portion of the boundary and estimates
%the fractal's arc length using numerical integration.
%
%Outputs:
%Figure with approximate Mendelbrot boundary
%Estimate of arc length
%
%Notes:
%The boundary of the Mandelbrot set is truly infinite as it is a fractal.
%This only approximates the length using a polynomial fit of degree 15.


function it = fractal(c)
    maxIter = 100; %max number of iterations
    z = 0; %take z=0 at the start

    for it = 1:maxIter
        z = z^2 + c; %recurrence fucntion
        if abs(z) > 2 %check if it has diverged
            return %if it diverges, we exit
        end
    end
    it = maxIter; %if we finish the for loop that means we never diverged
end

function fn = indicator_fn_at_x(x)
    %we return +1 if we escape
    %we return -1 if we stay bounded
    %@(y) makes it an anonymous function
    fn = @(y) (fractal(x + 1i * y)<100) * 2 -1; %tests if we are in or
outside of the set
    %this works since if we don't escape, we return 100 from fractal(c)
    %which will make fn -1
end

function m = bisection(fn_f, s, e)
    maxIter = 1000; %max number of steps to do
    tolerance = 1e-7; %can be changed, set to 1e-7 for now

    for k = 1:maxIter
        m = (s + e)/2; %midpoint
        if fn_f(m) == fn_f(s)
            s = m; %if we have the same sign we move lower bound
        else
            e = m; %else we move the upper bound
        end
        if abs(e-s)<tolerance
            m = (s + e)/2;
            return
        end
    end
end

numPoints = 10000; %min 1000, we use 10000
xValues = linspace(-2, 1, numPoints); %span [-2,1] with our numPoints
yValues = NaN(size(xValues)); %initialize an empty y vector

maxY = 1.5; %how far we search vertically
dy   = 0.01; %step size for checking

for i = 1:numPoints
    x = xValues(i);
    fn = indicator_fn_at_x(x);
    y = 0; %lower bound
    while y < maxY
        if fn(y) ~= fn(y+dy) %sign change detected
            yValues(i) = bisection(fn, y, y+dy); %use bisection function to
find boundary
            break
        end
        y = y + dy; %move up our step size for checking
    end
end

figure
plot(xValues, yValues, 'b.', 'MarkerSize', 4)
grid on
xlabel('Re(c)')
ylabel('Im(c) of boundary')
title('Approximate Mandelbrot Boundaries')

%we see a flat line of y=0 from x in [-2, -1.4] (eyeballing)

gate = xValues >= -1.4;
%apply gate
xFiltered = xValues(gate);
yFiltered = yValues(gate);

%second gate to remove NaNs
NaNGate = ~isnan(yFiltered);
xFiltered = xFiltered(NaNGate);
yFiltered = yFiltered(NaNGate);

%fitting polynomial of deg 15 to the filtered values
p = polyfit(xFiltered, yFiltered, 15);

%function to find length of polynomial
function l = poly_len(p, s, e)
    dp = polyder(p); %derivative of polynomial p
    ds = @(x) sqrt(1 + (polyval(dp, x)).^2);

    l = integral(ds, s, e); %integrate over [s, e]
end
```

```
%finding length of polynomial from [-1.4, 1]
length = poly_len(p, -1.4, 1);

%display approx length
fprintf("Approximate fractal length %.4f\n", length);
```

```
Approximate fractal length 54134.0885
```



Published with MATLAB® R2025a