# Table of Contents



```
% AMS595 - Assignment2
% Amol Arora, 116491705

% mandelbrot_boundary_analysis.m

% I have created a folder to save the results
folder_name = 'mandelbrot_results';
if ~exist(folder_name, 'dir')
    mkdir(folder_name);
end
```

# 1. Mandelbrot Set Function

This function calculates the number of iterations a point 'c' in the complex plane takes to escape the Mandelbrot set (up to 100 iterations).

```
function it = fractal(c)
    z = 0;
    for it = 1:100
        z = z^2 + c;
        if abs(z) > 2.0 % If the magnitude of z exceeds 2, it escapes the
Mandelbrot set
            return;
        end
    end
    it = 100; % If the loop completes, return 100 (indicates the point is
within the set)
end
```

# 2. Mandelbrot Set Visualization

Now, I Generate a grid of complex numbers (x + iy) over the region of interest

```
N = 200; % Resolution (number of points in each axis)
x = linspace(-2.0, 1.0, N);
y = linspace(-1.0, 1.0, N);
[X, Y] = meshgrid(x, y);
C = X + Y * 1i;
```

```
IT = zeros(N, N);
for i = 1:N
    for j = 1:N
        IT(i, j) = fractal(C(i, j)); % Here, I store the iteration count for
each point
    end
end

figure;
imshow(IT, []);
colormap(jet);
colorbar;
title('Mandelbrot Set');
saveas(gcf, fullfile(folder_name, 'mandelbrot_set.png'));
```

# 3. Bisection Method

```
function m = bisection(fn_f, s, e)
    tol = 1e-10; % I am using tolerance, instead of strict equality for
breaking loop

    while abs(e - s) > tol
        m = (s + e) / 2;
        if fn_f(m) == 0 % If the function at the midpoint is zero, we found
the exact root
            return; % Exact solution found
        elseif fn_f(m) * fn_f(s) > 0
            s = m;
        else
            e = m;
        end
    end
end
```

# 4. Find the Fractal Boundary

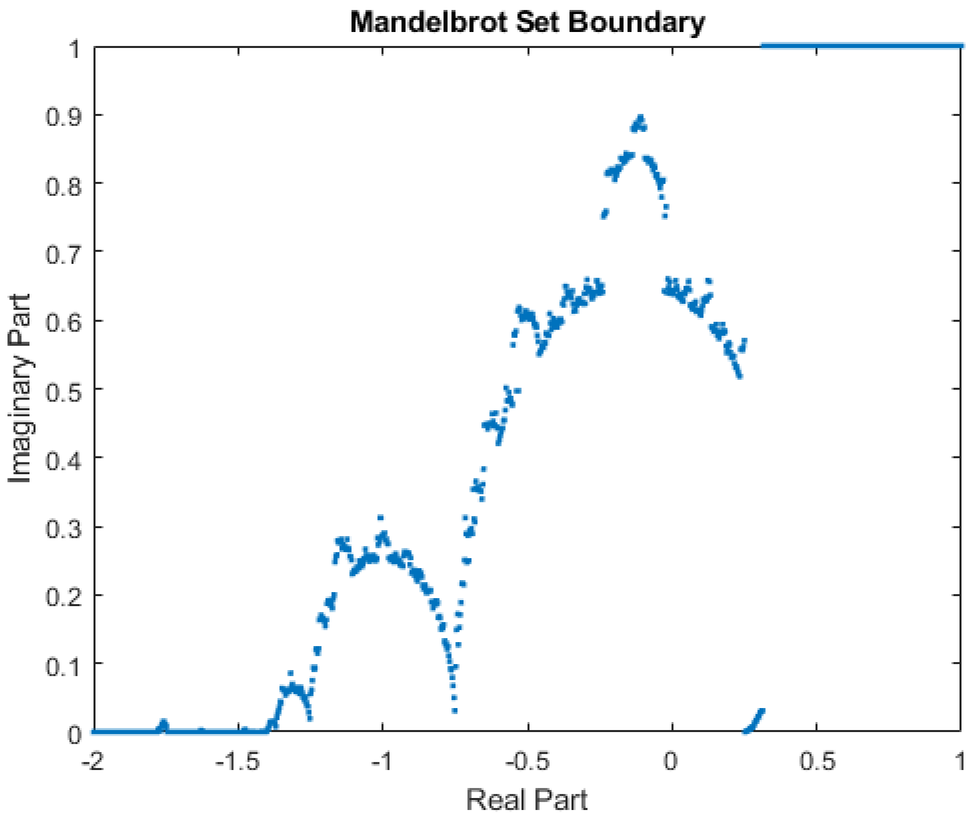
Here, I compute the boundary of the Mandelbrot set by applying the bisection method along the real axis for various x-values, estimating the corresponding y-values on the boundary.

```
x_values = linspace(-2, 1, 1000);
y_boundary = zeros(size(x_values));

for i = 1:length(x_values)
    x = x_values(i);
    % Fix indicator function logic
    indicator_fn = @(y) (fractal(x + 1i * y) == 100) * 2 - 1;
    y_boundary(i) = bisection(indicator_fn, 0, 1);
end

figure;
plot(x_values, y_boundary, '.');
title('Mandelbrot Set Boundary');
xlabel('Real Part');
```

```
ylabel('Imaginary Part');
saveas(gcf, fullfile(folder_name, 'boundary_points.png'));
```




# 5. Polynomial Fit

```
valid_indices = (x_values > -1.8) & (x_values < 0.5); % I filter out some
extreme values

% I extract valid x & y values.
x_fit = x_values(valid_indices);
y_fit = y_boundary(valid_indices);

% Reducing polynomial degree to avoid overfitting
p = polyfit(x_fit, y_fit, 7);

% Generate smooth curve for plotting the fitted polynomial
x_plot = linspace(min(x_fit), max(x_fit), 1000);
y_plot = polyval(p, x_plot);

% Plotting the boundary points and the fitted polynomial
figure;
plot(x_fit, y_fit, '.', x_plot, y_plot, 'r-');
legend('Boundary Points', 'Fitted Polynomial');
title('Polynomial Fit to Mandelbrot Set Boundary');
xlabel('Real Part');
```

```
ylabel('Imaginary Part');
saveas(gcf, fullfile(folder_name, 'polynomial_fit.png'));
```

# 6. Curve Length Calculation

```
% This function calculates the length of a polynomial curve by integrating
the square root of
% (1 + (dy/dx)^2) over the interval [s, e]. The derivative of the polynomial
is used for dy/dx.
function l = poly_len(p, s, e)
    dp = polyder(p);
    ds = @(x) sqrt(1 + polyval(dp, x).^2);
    l = integral(ds, s, e);
end
```

```
Approximate length of the Mandelbrot set boundary: 3.3278
Results saved in folder: mandelbrot_results
```

# 7. Calculate Boundary Length

```
% Here, I calculate the length of the fitted boundary curve using the
polynomial fit
s = min(x_fit);
e = max(x_fit);
boundary_length = poly_len(p, s, e);
fprintf('Approximate length of the Mandelbrot set boundary: %.4f\n',
boundary_length);

% Saving results
results_file = fullfile(folder_name, 'results.txt');
fid = fopen(results_file, 'w');
fprintf(fid, 'Approximate length of the Mandelbrot set boundary: %.4f\n',
boundary_length);
fprintf(fid, '\nPolynomial coefficients:\n');
fprintf(fid, '%.6e\n', p);
fclose(fid);

% Saving workspace
save(fullfile(folder_name, 'workspace.mat'));

disp(['Results saved in folder: ' folder_name]);
```

*Published with MATLAB® R2024a*