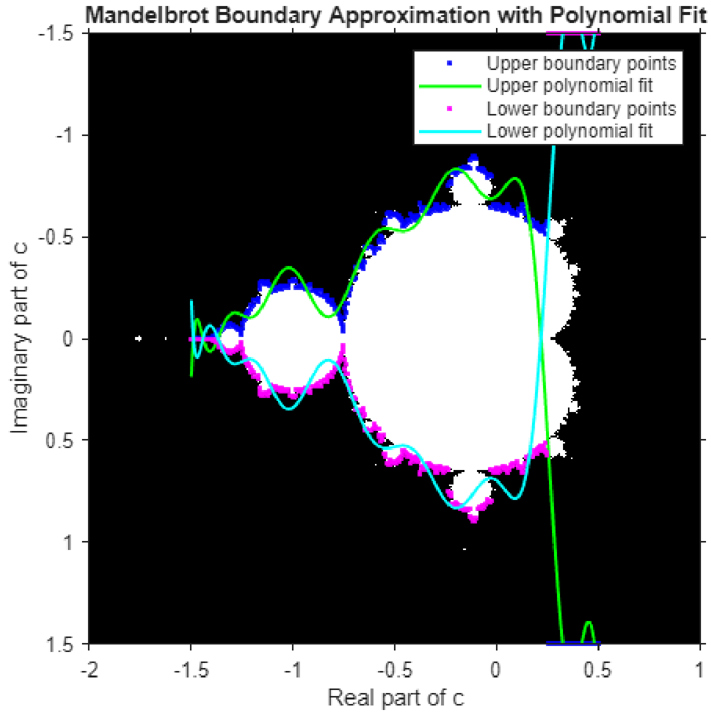
```
% Full code for computing the Mandelbrot set:


% Collecting the boundary points along the Mandelbrot set

% For x- values, we find the corresponding y-value where
% the boundary of the Mandelbrot set occurs, using the
% bisection method on vertical lines.

x_values = linspace(-2, 1, 1000); % Generating 1000 x-values in [-2, 1]
interval
y_values_upper = zeros(size(x_values)); % Initializing array for upper
boundary y-values
y_values_lower = zeros(size(x_values)); % Initializing array for lower
boundary y-value

for i = 1:length(x_values)
    x = x_values(i); % Picking one x-value

    % Creating an indicator function for this x-value. The function
    % should take a y-value and return:
    %  1 if point is outside the Mandelbrot set
    % -1 if point is inside the Mandelbrot set
    fn = indicator_fn_at_x(x);

    % Applying the bisection method to find the boundary along this
    % vertical line.

    y_values_upper(i) = bisection(fn, -1.5, 1.5); % Upper boundary
    y_values_lower(i) = bisection(fn,1.5,-1.5); % Lower boundary
end

% Plotting Mandelbrot set
n = 600;
x_range = linspace(-2, 1, n);
y_range = linspace(-1.5, 1.5, n);
mandelbrot_set = zeros(n, n);

for i_x = 1:n
    for i_y = 1:n
        c = x_range(i_x) + 1i*y_range(i_y);
        mandelbrot_set(i_y, i_x) = fractal(c);
    end
end

mask = mandelbrot_set == 0;

figure;
imshow(mask, 'XData', [x_range(1), x_range(end)], 'YData', [y_range(1),
y_range(end)]);
colormap(gray);
axis on;
```

```matlab
xlabel('Real parts of c');
ylabel('Imaginary parts of c');
title('Mandelbrot Set with Boundary and Polynomial Fit');
hold on;


% Fitting a polynomial to the boundary:
% To approximate the shape of the Mandelbrot boundary, we fit a polynomial
% to the boundary.

% Get rid of boundary points that are flat at the far left and far
% right ends to prevent distortion of the polynomial fit.

valid_idx = (x_values > -1.5 & x_values < 0.5); % Setting range of x-values

x_fit = x_values(valid_idx); % Real parts of c to fit
y_fit_upper = y_values_upper(valid_idx); % Imaginary parts of c to fit
(upper)
y_fit_lower = y_values_lower(valid_idx) ; % Imaginary parts of c to fit
(lower)

% Fitting a 15-th order polynomial to the boundary points

p_upper = polyfit(x_fit, y_fit_upper, 15); % Upper boundary
p_lower = polyfit(x_fit, y_fit_lower, 15); % Lower boundary

% Evaluating polynomial fit at a dense set of x-values
% Gives a smooth curve so that we can compare with raw values

x_x = linspace(min(x_fit), max(x_fit), 500); % 500 points in same range
y_y_upper = polyval(p_upper, x_x); % Evaluating polynomial at 'x_x'
y_y_lower = polyval(p_lower,x_x); % Evaluating polynomial at 'x_x'

% Plotting raw boundary points vs. fitted polynomial

plot(x_fit, y_fit_upper, 'b.');
plot(x_x, y_y_upper, 'g-', 'LineWidth', 1.5);
plot(x_fit, y_fit_lower, 'm.');
plot(x_x, y_y_lower, 'c-', 'LineWidth', 1.5);
legend('Upper boundary points','Upper polynomial fit', 'Lower boundary
points','Lower polynomial fit');
xlabel('Real part of c');
ylabel('Imaginary part of c');
title('Mandelbrot Boundary Approximation with Polynomial Fit');


% Curve length
l = poly_len(p_upper, min(x_fit), max(x_fit));
fprintf('Approximate boundary length: %.4f\n', l);
```

Approximate boundary length: 5.9211



Published with MATLAB® R2025a