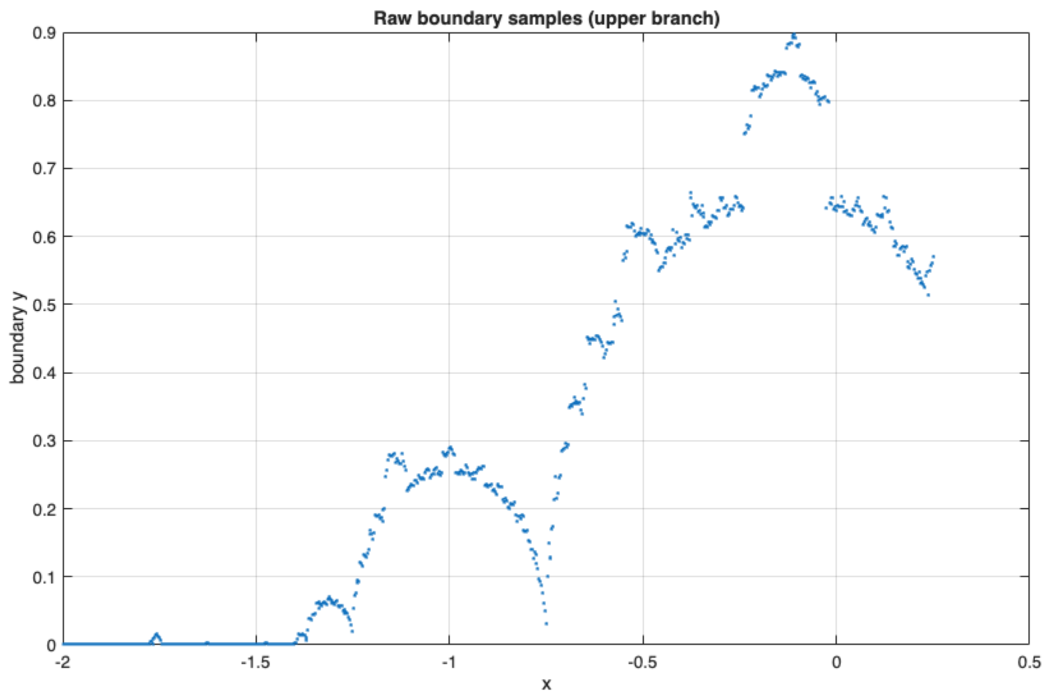
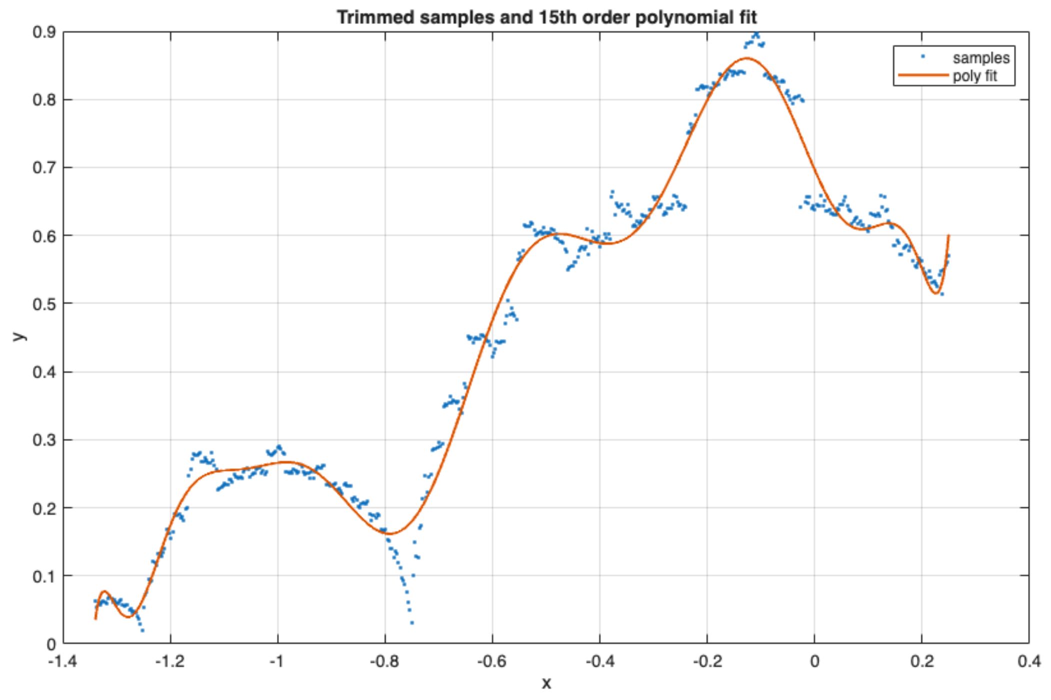
```
% MATLAB Project 2: How Long Is the Coast of Britain?
% Abby Bindelglass
% Due 10/5

close all;

% fractal boundary for at least 10^3 points
min_points = 1000;

% find fractal boundary for x in [-2,1]
x_min = -2.0;
x_max = 1.0;

xs = linspace(x_min,x_max,min_points); % grid of evenly spaced x for
boundary search
ys = nan(size(xs)); % preallocate y boundary values (same size as for x's)

s = 0.0; % lower bound, "y = 0 would do fine"
e = 1.5; % upper bound, above the fractal

tol = 1e-6; % tolerance for bisection method
max_steps = 60; % safety cap to prevent infinite loops

for i = 1:numel(xs) % loop through all x samples
    x = xs(i); % current x location
    fn = indicator_fn_at_x(x); % indicator along vertical line at x
    if fn(s) < 0 && fn(e) > 0 % if bracket is valid at x
        ys(i) = bisection(fn,s,e,tol,max_steps); % find boundary y
    else % if bracket invalid at x
        ys(i) = NaN; % mark as missing
    end
end

f1=figure(1); % create new figure
clf;
plot(xs,ys,'.','MarkerSize',5); % scatter plot of boundary samples
grid on; % add grid to plot
xlabel('x'); % x axis label
ylabel('boundary y'); % y axis label
title('Raw boundary samples (upper branch)'); % title of plot
drawnow;

lefttrim = -1.34; % left trim, chosen from looking at plot (where curve
upward starts)
righttrim = 0.25; % right trim, chosen from looking at plot (where curve
seems to flatten)

idx = (xs >= lefttrim) & (xs <= righttrim) & isfinite(ys); % keep curved
part and valid ys
x_fit = xs(idx); % x data for fitting
y_fit = ys(idx); % y data for fitting
```

```
p= polyfit(x_fit,y_fit,15); % fit 15th order polynomial to fractal boundary
data points
x = linspace(min(x_fit),max(x_fit),2000); % x grid for plotting fitted curve
y = polyval(p, x); % evaluate fitted polynomial at dense x

f2=figure(2); clf;
plot(x_fit,y_fit,'.','MarkerSize',6); % show trimmed data
hold on; % so another plot can be added to the same figure
plot(x,y,'-','LineWidth',1.5); % overlay polynomial fit
grid on; % add grid to figure
xlabel('x'); % label x axis
ylabel('y'); % label y axis
title('Trimmed samples and 15th order polynomial fit'); % add title
legend('samples','poly fit'); % add legend
drawnow;

s_interval = min(x_fit); % start of fitted x interval
e_interval = max(x_fit); % end of fitted x interval
arc_length_est = poly_len(p,s_interval,e_interval); % get arc length with
formula integral(sqrt(1+(dy/dx)^2) dx)

fprintf('Approximate boundary length on [%g, %g]: %.8f\n', s_interval,
e_interval, arc_length_est); % print the result

% export the figures
exportgraphics(f1,'figure_01.png','Resolution',300);
exportgraphics(f2,'figure_02.png','Resolution',300);

% FUNCTIONS

function it = fractal(c) % takes complex c and returns number of iterations
until divergence
    % stop if |z|>2, or cap at 100 iterations
    escape_radius = 2.0;
    max_iterations = 100;
    z = 0; % start with z = 0
    for k = 1:max_iterations % loop through up to 100 times
        z = z^2 + c; % equation 5.1, to test if point belongs to Mandelbrot
Set
        if abs(z) > escape_radius % if absolute value of z is greater than
2.0
            it = k; % record iteration where |z| exceeds escape_radius
            return; % early exit on escape
        end
    end
    it = 0; % |z| stayed bounded, treat as inside set
end

function fn = indicator_fn_at_x(x) % fn(y) = +1 if (x+i*y) escapes
(outside), -1 if not (inside)
    fn = @(y) (fractal(x + 1i*y) > 0)*2 - 1; % >0 returns 1 for escape; maps
{0,1}->{-1,+1}
end
```

```
function m = bisection(fn_f, s, e, tol, maxSteps) % takes function fn_f,
bounds s and e on initial guess, and returns point where sign of f changes
    fs = fn_f(s); % evaluate indicator at start
    fe = fn_f(e); % evaluate indicator at end
    if ~(fs<0 && fe>0) % enforce s is inside, e is outside
        m = NaN; % return NaN if bracket isn't valid at x value
        return
    end
    while (e-s) > tol && maxSteps > 0 % continue when interval is wide and
there are steps left
        m = 0.5*(s+e); % midpoint of interval
        fm = fn_f(m); % indicator at the midpoint
        if fm > 0 % outside (+) -> boundary in [s,m]
            e = m;
        elseif fm < 0 % inside (-) -> boundary in [m,e]
            s = m;
        else % on boundary
            s = m; e = m; % break
            break;
        end
        maxSteps = maxSteps - 1; % decrement counter each iteration
    end
    m = 0.5*(s+e); % return midpoint of final bracket
end

function L = poly_len(p,s,e) % takes polynomial p, starting and ending
points s,e and returns curve length of polynomial l
    dp = polyder(p); % derivative coefficients for f'(x)
    ds = @(x) sqrt(1 + (polyval(dp,x)).^2); % integrand sqrt(1+(f'(x))^2),
anonymous function with x as only argument
    L = integral(ds,s,e,'RelTol',1e-8,'AbsTol',1e-10); % use integral to
compute curve length of polynomial, applied to fractal boundary
end
```

```
Approximate boundary length on [-1.33934, 0.249249]: 2.42118429
```





Published with MATLAB® R2025a