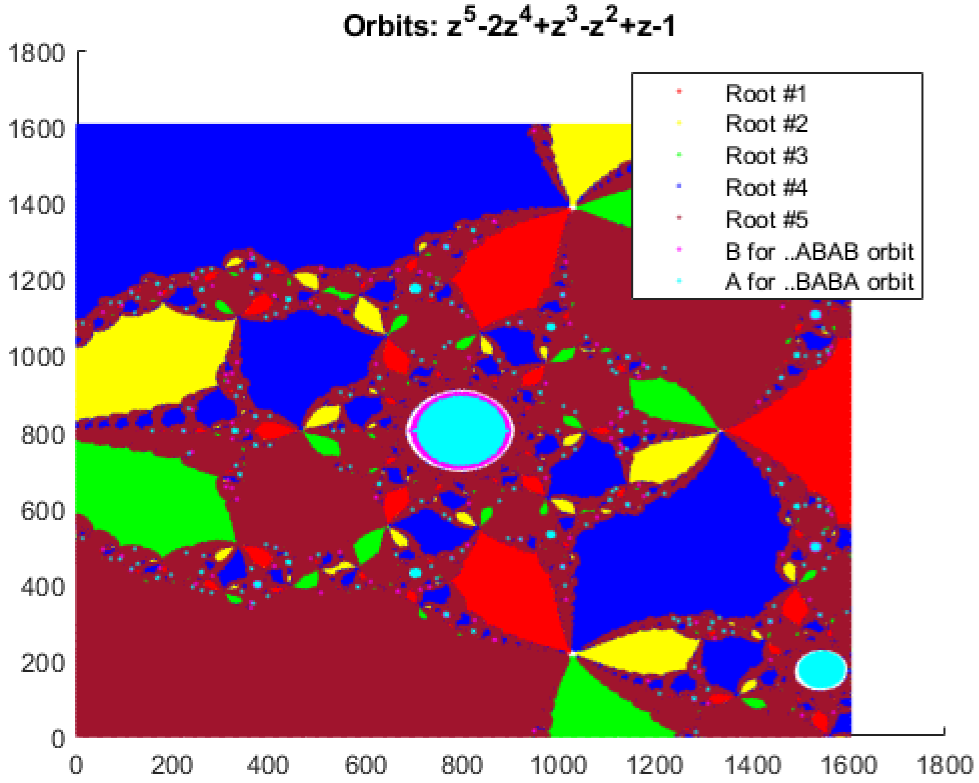
```
% Pierce Zhang, CMOR220, Fall 2023, Project 3: Newton Fractals
% newton_bonus.m
% Find attractive periodic orbits in Newton's method
% Last modified: 27 September 2023
```

It holds that x^5 - 2x^4 + x^3 - x^2 + x - 1 exhibits an attractive periodic orbit on Newton's Method.

```
% Driver function
% Uses qnewt to plot a polynomial that exhibits an attractive periodic orbit
% behavior
function newton_bonus
% Parameters of Newton's method
xt = [-0.2,0.00025,0.2];
yt = [-0.2,0.00025,0.2];
maxiter = 20;

qnewt([1 -2 1 -1 1 -1],xt,yt,maxiter);
end

% Inputs: q, a vector of coefficients of a polynomial
% Outputs: dq, a vector of coefficients of the analytical derivative of q
function [dq] = myownpolyder(q)
% Evaluates the analytical derivative of a polynomial
% Derivative function should have one less term
dq = zeros(1,length(q) - 1);
for i = 1:length(dq)
    % Where the coefficient is q(i) and the power is length(q) - i
    dq(i) = q(i) * (length(q) - i);
end
end

% Inputs:
% coeff, the coefficient of a monomial term
% img, a boolean indicating whether the coefficient is imaginary
% power, the power of z in the monomial term
% Outputs: term, a string containing the formatted monomial term
function [term] = qterm(coeff, img, power)
term = "";
% Sign handler
if (coeff < 0)
    term = term + '-';
elseif (coeff > 0)
    term = term + '+';
else
    return
end
% Coefficient handler
if (abs(coeff) ~= 1)
```

```
    term = term + abs(coeff);
elseif (power == 0 && img == false)
    term = term + abs(coeff);
end
%Imaginary unit handler
if (img)
    term = term + 'i';
end
% Power handler
if (power == 1)
    term = term + 'z';
elseif (power > 1)
    term = term + 'z^' + power;
end
end

% Inputs: q, a  vector of coefficients for a polynomial
% Outputs: name, a string which represents a polynomial
function [name] = qlab(q)
% Produces a title for a given polynomial as a coefficient vector
name = "";
for i=1:length(q)
    coeff = q(i);
    power = length(q) - i;
    % Skip over all 0 coefficient terms
    if (coeff ~= 0)
        name = name + qterm(real(coeff), false, power);
        name = name + qterm(imag(coeff), true, power);
    end
end
% Remove leading plus sign.
if (name ~= "" && extract(name,1) == '+')
    name = eraseBetween(name, 1, 1);
end
end

% Inputs:
% q, a vector of up to 5 complex coefficients of a polynomial
% xt, a vector of 3 x grid values, xlo, xinc, and xhi
% yt, a vector of 3 y grid values, ylo, yinc, and yhi
% maxiter, the maximum number of iterations
% Outputs: none (produces plots)
function qnewt(q, xt, yt, maxiter)

% Parameters of iteration
dq = myownpolyder(q);
x = xt(1):xt(2):xt(3);
y = yt(1):yt(2):yt(3);

% Initialize grid to store results of Newton's method iteration
[X,Y] = meshgrid(x,y);
% Initialize four complex planes, where Z is current and Z_3 is 3 iters ago
Z = X + 1i*Y; Z_1 = X + 1i*Y; Z_2 = X + 1i*Y; Z_3 = X + 1i*Y;
% Run Newton iterations
```

```
for k=1:maxiter
    % Newton's method on all grid values simultaneously
    dq_Z = polyval(dq,Z);
    dq_Z(dq_Z==0) = NaN; % detect division by zero and set to NaN
    Z = Z - polyval(q,Z)./dq_Z;
    % Populate last three iters
    if (k == maxiter - 3)
        Z_3 = Z;
    elseif (k == maxiter - 2)
        Z_2 = Z;
    elseif (k == maxiter - 1)
        Z_1 = Z;
    end
end

% Plot
figure()
hold on

r = roots(q); % find roots for comparison
colors = ['r','y','g','b',"#A2142F"]; % root colors
speccolors = ['k',"#EDB120",'m','c']; % special condition colors

[hdiv0,kdiv0] = find(isnan(Z)); % show break down points as black
plot(kdiv0,hdiv0,'.','color',speccolors(1),'MarkerSize',3,'DisplayName', ...
    'Breakdown (dy/dx=0)');

[hinf,kinf] = find(abs(Z)>1e18); % show infinite points as orange
plot(kinf,hinf,'.','color',speccolors(2),'MarkerSize',3,'DisplayName', ...
    'Divergence');

for i = 1:length(r)
    [h,k] = find(abs(Z - r(i))<0.001); % get points near root r(i)
    plot(k,h,'.','color',colors(i),'MarkerSize',3,'DisplayName',"Root #"+i); %
 plot
    Z(abs(Z - r(i))<0.001) = NaN; % remove all root points from histories
    Z_1(abs(Z_1 - r(i))<0.001) = NaN;
    Z_2(abs(Z_2 - r(i))<0.001) = NaN;
    Z_3(abs(Z_3 - r(i))<0.001) = NaN;
end

[hev,kev] = find(Z == Z_2); % plot Z, Z-2 iter orbit points
[hod,kod] = find(Z_1 == Z_3); % plot Z-1, Z-3 iter orbit points
plot(kev,hev,'.','color',speccolors(3),'MarkerSize',3,'DisplayName', ...
    'B for ..ABAB orbit') % magenta
plot(kod,hod,'.','color',speccolors(4),'MarkerSize',3, 'DisplayName', ...
    'A for ..BABA orbit') % cyan

title("Orbits: " + qlab(q));
legend;
end
```



Published with MATLAB® R2023a