Project Expectations

The project must be completed by a team of two students. A single hard copy of each project should be submitted by the project deadline. A cover sheet should be included with the project submission. The cover sheet should include the names of the two students on the project team. It should also include a brief summary of the work completed by each student. Project team members need to sign the cover sheet indicating their agreement with the summary of work they completed. In addition, MATLAB m-files should be e-mailed to shontz@ku.edu by the project deadline. Please use ‘EECS 639: Project 3’ as the subject line of your e-mail. Do not submit work unless you have adhered to the principles of academic integrity as described in the syllabus. Points will be deducted by poorly commented code, redundant computation that seriously effects efficiency, and failure to use features of MATLAB that are part of the course syllabus. In particular, use vector operations whenever possible. In addition, you should make an effort to minimize the error in your computations (where possible).

Solving Nonlinear Equations for Use in Fractal Generation

Fractals are geometric objects in which self-similar patterns recur at progressively smaller scales. Fractals occur throughout nature and are observed in patterns in leaves, crystal growth, fluid turbulence, and galaxy formation to name just a few applications. They also occur in patterns in vibrations and sounds, e.g., heart beats.

In this project, you will solve nonlinear equations using various starting points. The resulting patterns of convergence to the various roots of the equation (or divergence) will then be used to generate fractals.

Figure 1: Fractals in nature

Part A: Implementation of Newton’s Method

Write a Matlab function which implements Newton’s method. In particular, your function should compute a root of the complex function $f(z)$ given the starting point $z_0 = x_0 + iy_0$. (Note that the roots of $f(z)$ may be complex and/or real.) It should then return a color corresponding to the root it found in at most 20 iterations or to failure (in the case it does not find a root within 20 iterations).

Here is a template for the Matlab function which can be completed and applied to the case when Newton’s method is used to solve a nonlinear equation containing four roots. For example, consider solving $f(z) = (z^2 - 1)(z^2 + 0.16)$, where $z = x + iy$. The equation has 4 roots given by $z_1 = 1, z_2 = -1, z_3 = 0.4i$, and $z_4 = -0.4i$. 
Matlab function template:

```
function color = Newton(z)
    
    % Input:
    % z = complex number
    
    % Output:
    % color = the color associated with the root found by Newton’s Method
    
    if Newton's Method found z_1
        color = 'y';
    elseif Newton's Method found z_2
        color = 'r';
    elseif Newton's Method found z_3
        color = 'b';
    elseif Newton's Method found z_4
        color = 'g';
    elseif Newton's Method was unsuccessful after 20 iterations.
        color = 'k';
    
    % More precisely, "unsuccessful" means that Matlab cannot reduce
    % (z^2-1)(z^2+0.16) to less than 0.0001 (in absolute value) in at most
    % 20 Newton iterations.
    
    Part B: Generation of a Fractal

    Second, write a script that calls Newton.m with various starting values z=x+iy where
    
    x = linspace(0.15,0.55,150);
    y = linspace(-0.15,0.15,150).
    
    Save the x and y coordinates and the colors returned by Newton.m in vectors as appropriate. Finally, plot the points according to the obvious scheme: if using z = x+iy as a starting point for Newton’s method yielded color = 'r', then plot z = x+iy red.
    
    If done correctly, your code will generate a beautiful image of a fractal. If not, please debug your code before proceeding any further.

    HELPFUL HINT:
    • It is helpful to know how to convert between complex numbers and their real and imaginary parts. See the help for the following Matlab function: complex, real, and imaginary for examples of how this is done.

    Part C: Generation of Additional Fractals

    Next, solve other nonlinear equations to generate five additional fractal images.
    
    For each nonlinear equation used in your project, be sure to specify the nonlinear equation being solved and its roots. In addition, generate a fractal plot illustrating the convergence of Newton’s method to the various roots from an appropriate rectangle of starting points.

    Part D: Secant Approximation to the Derivative

    Repeat Parts A through C using the secant method (in place of Newton’s method). How does each of the fractal pictures change when the secant method is used to compute a root of the nonlinear equation? How sensitive is the fractal image to the distance that z_1 is from z_0? Experiment with at least two different values for z_1.

    Items to Submit: Submit your Matlab code (for Newton’s method, the secant method, and the corresponding scripts), the fractal diagrams (illustrating the convergence patterns for Newton’s method and for the secant method), and a succinct discussion of your results.

    Grading Criteria: Your project will be graded on the basis of: accuracy and efficiency of the implementations of the numerical methods, correctness of the fractal diagrams, thoroughness of the results, and the writing itself. The project is worth a total of 100 points.