Math 448 – Numerical Analysis: Root Finding Project.

Option 1: Convergence at a Double Zero

Consider the cubic $f(x) = (x - 1.1)(x - 2.1)^2 = x^3 - 5.3x^2 + 9.03x - 4.581$, which clearly has a double zero at $x = 2.1$.

1. Verify that using Newton's method with initial value 2 converges linearly for this cubic (look at the magnitude of the successive errors) and that we don't get more than about eight digits accuracy.

2. A variant of Newton's method is $x_{n+1} = x_n - (mf(x_n))/(f'(x_n))$, where $m = 2$ for a double root (the proof is tedious so we won't do it here). Use this version for this cubic and verify that convergence is quadratic, but you still only get about eight digits accuracy.

3. The μ method suggests applying Newton's method to $\mu(x) = f(x)/f'(x)$. If $\mu(x) = (x - a)^m g(x)$, show that $\mu'(x) = (x - a)g(x)/[mg(x) + (x - a)g'(x)]$ so it and $f(x)$ have the same zeros, but the multiplicity of $x = a$ has been reduced to one. Show that the Newton iteration in this case is

$$x_{n+1} = x_n - \frac{f(x_n)f'(x_n)}{(f'(x_n))^2 - f(x_n)f''(x_n)}.$$

   Apply this method to this cubic again, verify quadratic convergence, but how many digits of accuracy do you get?

4. Apply Steffensen's method to this cubic with the standard (linear) Newton iteration. We should get quadratic convergence, but verify that yet again, we lose accuracy.

5. Apply the Durand-Kerner method to this cubic, and see what happens.

6. Apply the Aberth method to this cubic, and see what happens.

7. Use the Matlab routine `roots`, which is based upon turning root finding into an eigenvalue problem. What sort of accuracy do we get?

8. The μ method above worked because $f(x)/f'(x)$ has the same zeros as $f(x)$, but all single. Actually, $g(x) = f(x)/\gcd(f(x), f'(x))$ has the same roots as $f(x)$, all of multiplicity one, and no extraneous asymptotes that were introduced by the μ method, and so is more reliable. Luckily, finding the greatest common divisor of polynomials is done exactly as the standard Euclidean algorithm, with polynomial long division. If $p_1(x)$ has higher order than $p_2(x)$, let $p_1(x) = q(x)p_2(x) + r(x)$, then $\gcd(p_1(x), p_2(x)) = \gcd(p_2(x), r(x))$. Apply this algorithm (by hand) to find the gcd for our particular function, then do the division and apply Newton's method to $g(x)$ with $x = 2$ one more time. *Finally* do we get decent accuracy?

Option 2: Fractals in Newton's Method

The aim of this project is to write a function that produces a fractal image where the color of a square on the plane indicates which zero Newton's method converges to from that complex value. We will stick to cubics in this case.

- To achieve a C, write a function that takes as input a single complex number, and uses Newton's method to find where it converges when solving $x^3 + 1 = 0$. You can write the code so it only solves this one problem, so a single while loop should be all you need – it will always converge as long as you don't start at zero. The output from the code should be a single number, a one, two, or three, depending on which of the possible solutions -1 and $\frac{1}{2} \pm \frac{\sqrt{3}}{2}i$ appear. Since there is always the chance of roundoff error, when you are comparing the solution from Newton's method to the true solutions, a condition like `abs(x+1)<1e-12` is better than `x==-1`.

- To achieve a B, in addition write a function that takes as input four numbers representing the top, left, bottom and right bounds of a rectangle, and produces a plot of the fractal in that rectangle, using three different colors for the three different solutions. I would suggest that you choose a grid of points (say 200 by 200?) in the rectangle, and produce a two dimensional array by calling the function you write previously at each point. To plot the actual picture, I suggest you look at Matlab's help files on `linspace` to generate x and y vectors of data, then look at `pcolor`. An example piece of code to consider might be `x=linspace(3.5,6.5,4); y=x; c=[1,1,2,0;1,2,2,0;2,3,3,0;0,0,0,0]; pcolor(x,y,c); shading flat`.

  Use this code to draw a number of pictures showing the fractal behavior as you zoom in near a boundary.

- To achieve an A, alter your code so that rather than solving $x^3 + 1 = 0$ you are solving $ax^3 + bx^2 + cx + d = 0$, which means you will need to input four additional values. Check out the Matlab function `roots` to find them the solutions to this equation. Do we still see the same fractal type structure for other cubics? And what happens when you have a cubic where two of the zeros are close to each other?

Your write-up should include all the code you used (with moderate commenting) as well as any discussion on how you wrote the code. Include any pictures you feel are useful that were output, as well as discussion of the results. I am happy if your solution is a Word file rather than a printout so you don't have to deal with the expense of using a color printer.

In a perfect world, we could turn this into a GUI with sliders for position, zooming and changing coefficients, but that would be well beyond the scope of the course.