Exploring Newton's Method Some More!

1 What IS it?

Newton's classic root-finding algorithm is fairly mundane in practice: exploiting the calculus of tangent lines to "nice" functions, we can often produce a sequence of values that (may or may not) converge to a *root* of said function. In other words, we can construct an iterated function system that produces a sequence $\{x_0, x_1, ..., x_n\}$ such that $\lim_{n\to\infty} x_n = x^*$, where $f(x^*) = 0$. Interesting enough in and of itself, the method becomes far more interesting when we consider the question of the relationship between the starting value x_0 and the root x^* that the method converges to. We are currently examining this question by using functions in the complex plane, functions that take complex numbers as input and which produce complex numbers as output.

So, for a "nice" (i.e. differentiable) function $f(z)$, one that has roots at values given by $\{z_0^*, z_1^*, ..., z_n^*\}$, we define the function to be iterated from some initial value z_0 as follows:

$$z_{n+1} = z_n - \frac{f(z_n)}{f'(z_n)} = newtons(z_n) \tag{1}$$

2 What are you going to do?

In your programs you will explore the relationship between z_0 and the root that this scheme converges to (if it converges to anything!) You will do this by color-coding each starting value z_0 with a unique color corresponding to each root. In order to do this you will need to come up with several examples of functions $f(z)$ for which you already know

- the first derivative $f'(z)$
- the roots $\{z_0^*, z_1^*, ..., z_n^*\}$ of f

If you have these things, then you can set up the iteration scheme and start exploring!

A (possibly incomplete) laundry-list outlining what your program should do:

(1) define $f(z)$ and $f'(z)$

(2) using the above two functions, define $newtons(z)$, the iterated function [see equation (1) above]

(3) define a list that contains all of the roots of $f(z)$

(4) define a second list, of colors, that is parallel to the list in the previous step.
Note: your list of colors must have at least as many values as your list of roots and since you will play around with several different functions, you might as well make your color list have as many values in it as the number of roots that your function with the most roots has.

(5) import the `cmath` library (perhaps *as cm* to keep it distinct from the `math` library?). You will probably want to use the `isclose` function in `cmath`.

(6) create an `NLDGraphWin` object

i set `autoflush` to `False`

ii use `setCoords` to set up custom coordinates to an appropriate value

Here is some sample code that accomplishes the steps immediately above:

```
myWindow = NLDGraphWin(width=400, height=400)
myWindow.autoflush = False
myWindow.setCoords(-4,-4,4,4)
```

(7) For *every* point in your window (you will use a set of nested `while` loops here)

i create the complex number that corresponds to that point

ii iterate that value a set number of times using $newtons(z)$

iii determine which root your value has come closest to and ...

iv plot the original point with the color that corresponds to that root

Then try zooming in on your image at interesting points and see what you get! Some examples:

A. $f(z) = z^2 - 1$:

$f'(z) = 2z$
roots are $z_0 = 1$ and $z_1 = -1$

B. $f(z) = (z-1)(z+1)z = z^3 - z$:

$f'(z) = 3z^2 - 1$
roots are $z_0 = 1$, $z_1 = -1$, and $z_2 = 0$

Note: We did examples A and B in class.

C. $f(z) = (z-1)(z+1)(z-i)z = z^4 - iz^3 - z^2 + iz$:

$f'(z) = 4z^3 - 3iz^2 - 2z + i$
roots are $z_0 = 1$, $z_1 = -1$, $z_2 = 0$, and $z_3 = i$

D. the so-called *roots of unity* of $f(z) = z^n - 1$ for $n > 0$:

$f'(z) = nz^{n-1}$
roots are given by $e^{i \cdot k \cdot \frac{2\pi}{n}}$ for $k = 0, 1, ...(n-1)$