proj 14: Newton's Method and Successive Approximation

Introduction

Successive approximation is a problem-solving method where you try to guess the right answer to a problem and then check your guess. If the guess is good enough, you're done. Otherwise, you keep improving your guess in small increments and checking it, getting closer and closer to the right answer, until you determine that the guess is good enough. We will look at Newton's method, which uses successive approximation to find the roots of a function.

Polynomials

For this project, we will be representing polynomials as tuples. The index of a number in the tuple represents the power, and the value at that index represents the coefficient for that term. So or example, the polynomial $x^4 + 3x^3 + 17.5x^2 – 13.39$ would be represented by the tuple (-13.39, 0.0, 17.5, 3.0, 1.0). This is because the tuple represents $-13.39x^0 + 0.0x^1 + 17.5x^2 + 3.0x^3 + 1.0x^4$, which is the same as $x^4 + 3.0x^3 + 17.5x^2 – 13.39$.

Problem #1

Implement the evaluate_poly function. This function evaluates a polynomial function for the given x value. It takes in a tuple of numbers poly and a number x. By number, we mean that x and each element of poly is a float. evaluate_poly takes the polynomial represented by poly and computes its value at x. It returns this value as a float.

Derivatives

We will need to find $f'(x_n)$, where $f'(x)$ is the derivative of $f(x)$. The derivative of a polynomial $f(x) = ax^b$ is $f'(x) = abx^{b-1}$, unless b=0, in which case $f'(x) = 0$. To compute the derivative of a polynomial function with many terms, you just do the same thing to every term individually. For example, if $f(x) = x^4 + 3x^3 + 17.5x^2 – 13.39$, then $f'(x) = 4x^3 + 9x^2 + 35x$.

Problem #2

Implement the compute_deriv function. This function computes the derivative of a polynomial function. It takes in a tuple of numbers poly and returns the derivative, which is also a polynomial represented by a tuple.

Newton's Method

Newton's method (also known as the Newton-Raphson method) is a successive approximation method for finding the roots of a function. The roots of a function f(x) are the values of x such that $f(x) = 0$.

Here is how Newton's method works:

1. We guess some x_0.
2. We check to see if it's a root or close enough to a root by calculating $f(x_0)$. If $f(x_0)$ is within some small value epsilon of 0, we say that's good enough and call x_0 a root.
3. If $f(x_0)$ is not good enough, we need to come up with a better guess, x_1. x_1 is calculated by the equation: $x_1 = x_0 - f(x_0)/f'(x_0)$.

4. We check to see if x_1 is close enough to a root. If it is not, we make a better guess x_2 and check that. And so on and so on. For every x_n that is not close enough to a root, we replace it with $x_{n+1} = x_n - f(x_n)/f'(x_n)$ and check if that's close enough to a root. We repeat until we finally find a value close to a root.

For simplicity, we will only be using polynomial functions.

Implementing Newton's Method

Problem #3

Implement the compute_root function. This function applies Newton's method of successive approximation as described above to find a root of the polynomial function. It takes in a tuple of numbers poly, an initial guess x_0, and an error bound epsilon. It returns a tuple. The first element is the root of the polynomial represented by poly; the second element is the number of iterations it took to get to that root.

The function starts at x_0. It then applies Newton's method. It ends when it finds a root x such that the absolute value of f(x) is less than epsilon, i.e. f(x) is close enough to zero. It returns the root it found as a float.