Mixing Open-Bracketing Methods and Modifications

Kaitlyn Colaw, Marissa Murphy, M. Catherine Yopp

MA348 - Numerical Analysis

Dr. Sirani M. Perera

Spring 2022

Embry-Riddle Aeronautical University

Abstract

This paper aims to describe the development and implementation of an adaptable Python code that identifies when the derivative of a function goes to zero. Knowing this, the code will guide the user to use the appropriate numerical method to approximate solutions to user-given functions. This program can be applied within academic, research, and other professional or personal-use contexts to quickly identify applicable numerical methods and more efficiently find root solutions.

Introduction

In the early 12th century. Leonardo of Pisa described the following polynomial equation, known as Fibonacci's cubic:

$$x^3 + 2x^2 + 10x - 20 = 0 \qquad (1)$$

He identified the root to be x = 1.368808107. While his utilized method is unknown, this achievement would be the beginning of the study of root-finding algorithms. This paved the way for the applications of root-finding to advance and be applied to solve real-world problems, such as efficiency testing and the finding of local critical values and inflections in data points, such as weather conditions or aircraft performance.

The Newton-Raphson method was first described in Joseph Raphson's *Analysis Aequationum Universalis* in 1690. It is used to find the roots of an equation and show that the method is convergent as it approaches the root. The Newton-Raphson method accounts for an initial guess, x_0, so that a function f(x) can be described by the Taylor series expansion:

$$f(x) = f(x_0) + f'(x_0)(x-x_0) + \frac{1}{2}f''(x_0)(x-x_0)^2 + \ldots = 0 \qquad (2)$$

In comparison to the Newton-Raphson method, the modified Newton-Raphson method converges faster and is preferable for systems with multiple roots. However, it is more computationally expensive and less efficient than the unmodified counterpart for simple roots. Both methods converge relatively quickly.

$$x_{i+1} = x_i - \frac{f(x_i)f'(x_i)}{[f'(x_i)]^2 - f(x_i)f''(x_i)} \qquad (3)$$

The secant method is another root-finding algorithm that dates to 3000 years before the Newton-Raphson method. Egyptian Rhind Papyrus made the connection that the Rule of Double False Position coincides with the secant method when applied to a linear equation. The secant method can be described as Newton's method using a finite difference approximation to the derivative.

$$x_{i+1} = x_i - \frac{f(x_i)(x_{i-1}-x_i)}{f(x_{i-1})-f(x_i)} \qquad (4)$$

In creating the root-finding adaptable code, these methods will be referenced and utilized. The code will guide the user to use the appropriate numerical method to approximate solutions to user-given functions.

Methodology

The Python codes developed for these root finding methods, works on the tolerance of an approximation error of $10e^{-5}$. While this tolerance does not account for the possibilities of oscillations in the approximations, it does prevent the code from attempting to compute infinitely due to a severe convergence from the true root value. For this reason, the approximate error was chosen over the true error for all iterative functions.

Mixing methods

While the Newton-Raphson method is the most efficient method of the aforementioned, it has the potential to break an iterative loop due to the division by zero if the derivative of the function is zero at one of the approximations. But, due to Anthony Ralston's and Philip Rabinowitz's theoretical demonstration that a function approaches zero quicker than its derivative, a conditional statement can be placed in one's code that will force an automatic switch to the secant method for only the iterations necessary.

Multiple Root

The Python code developed to find multiple roots requires knowledge of how many roots are being searched for and where (the initial guesses). By having the number of roots as an input into the function developed for finding multiple roots, the loop created to find a single root can be placed inside another loop that will iterate for as many roots are being searched for.

Numerical Results

Newton-Raphson and its Modification

Equation (5) with root -2.5 was used to test the computational differences between the Newton-Raphson method and its modification. Both functions were run with the initial guess of -1.

$$x^2 + 5x + 6.25 \qquad (5)$$

The classic Newton-Raphson method was able to approximate the root after 16 iterations with a true relative error of $9.2E-6$ but, the modified version was able to find the exact—with respect to machine precision—root in only two iterations. The modified version was capable of calculating an approximation so close to the root that after the first iteration, Equation (5) at that first estimated root caused the function to automatically switch to the secant method to satisfy the stopping condition.

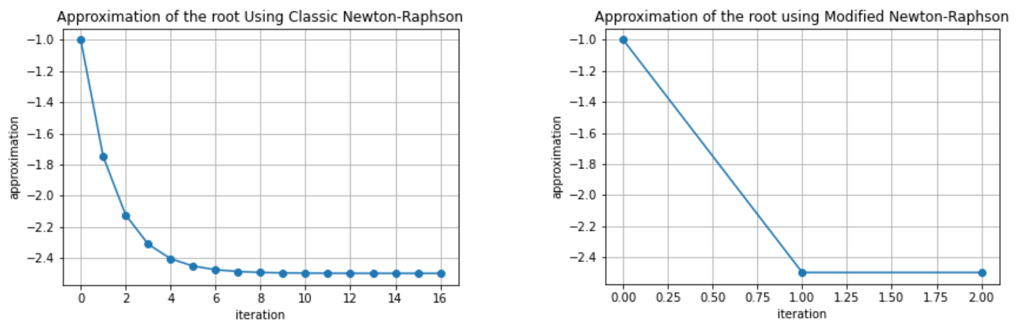

Figure 1: The new approximated root for every iteration using the classic Newton-Raphson method (left) and the modified Newton-Raphson method (right) with the initial guess of -1.

When the same functions were used but with an initial guess of -3, the original Newton-Raphson method was able to calculate the root in 15 iterations and with a true relative error of $6.1E-6$ while the modified version still only required two iterations to reach the exact solution.

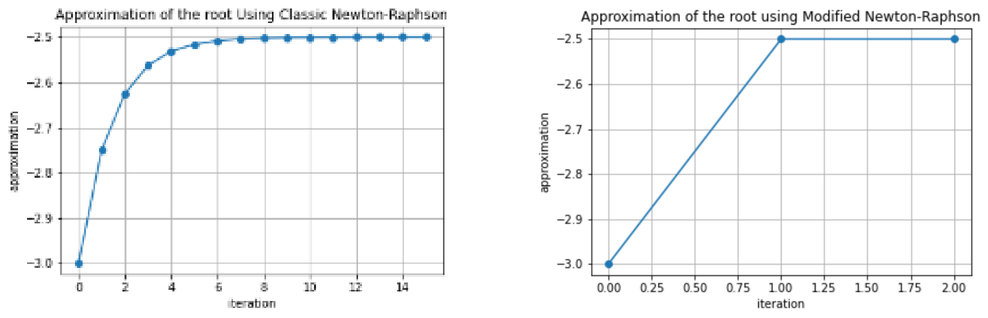

Figure 2: The new approximated root for every iteration using the classic Newton-Raphson method (left) and the modified Newton-Raphson method (right) with the initial guess of -3.

The Netwon-Raphson Method Considering Multiplicity

Another modification to the Newton-Rapshon method includes multiplying the second term of the classic Netwon-Raphson formula by the multiplicity, m, of the root that is to be approximated creating Equation (6).

$$x_{i+1} = x_i - m\frac{f(x_i)}{f'(x_i)} \tag{6}$$

To compare the classic Newton-Raphson method to Equation (6), Equation (7) was used which gives the single root 2 with a multiplicity of 3.

$$(x-2)^3 \tag{7}$$

With the initial guess of 1 given for both methods, the classic Newton-Raphson method took 25 iterations to approximate the value 1.9999603978719576, producing a true relative error of $1.98E-5$. Equation (6) was capable of computing the exact root with only two iterations. Similar to the modified Newton-Raphson method, the first approximation was so close to the approximation that for the second iteration, the function was forced to switch to secant method to meet the stopping condition tolerance. Because this secant method is related to the consideration of the multiplicity of the root, the second term in Equation_ is also multiplied by the multiplicity.

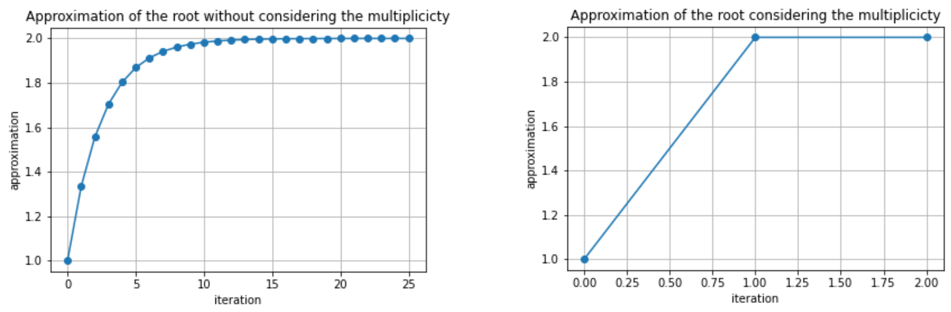

Figure 3: The new approximation of the root for each iteration without the consideration of the root's multiplicity (left) and with consideration of the multiplicity (right) using the initial condition 1.

Changing the initial guess of both methods to 4, the classic Newton-Raphson method took 25 iterations to calculate the root to be 2.0000352018915932 with a true relative error of $1.76E-5$ with Equation (6) producing the same results as before.

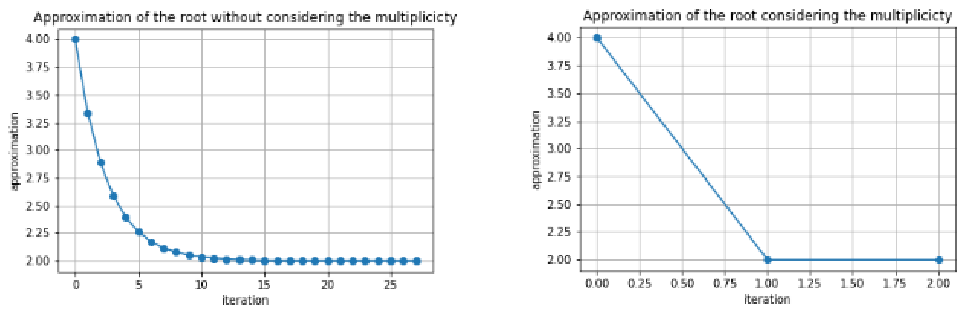

Figure 4: The new approximation of the root for each iteration without the consideration of the root's multiplicity (left) and with consideration of the multiplicity (right) using the initial condition 4.

The Secant Method and its Modification

Similar to the Newton-Rapshon method, there is a suggested modification of the Secant method (Equation (8)). The method utilized in this paper includes the zeroth and first derivative of the function under investigation. This is to avoid the need for the three initial guess that would be necessary if the first derivative of the second initial guess were to be approximated using finite difference.

$$x_i - \frac{f'(x_i)(x_{i-1} - x_i)}{f(x_{i-1}) - f(x_i)} - \frac{f(x_1)(x_{i-1} - x_i)\big(f'(x_{i-1}) - f'(x_i)\big)}{\big(f(x_{i-1}) - f(x_i)\big)^2} \tag{8}$$

To compare the two methods, Equation (9) was used focusing on finding the root -12.008 with the two initial guesses of -11 and -11.5.

$$x^5 + 12x^4 - 13x \tag{9}$$

Both methods took only three iterations to converge based on the approximate error stopping condition. The classic secant method converged to a value of -12.25512169497461 having a relative true error of $.021$ and the modified version converged to a value of -12.067604821327151 having a .5 true relative error. Running the methods with different initial guesses, it is found that the modified method is more sensitive to the difference between the initial guesses and the distance they both are from the actual root. For example, with initial conditions -11 and -10, the classic secant method converges to a value of -13.765432098765432 in three iterations while the modified version converges to a value of -5.217982292056367. When the initial conditions are changed to -9 and -9.5, the classic secant method converges to 5.428200573848887 and the modified method converges to 28.432464976462242 in three iterations. These changes in initial guesses shows that the modified secant method deviates much quicker from the actual root than the classic secant method.

Applications

The combination of the Secant and Newton Raphson methods in this adaptable code makes evaluating the roots of a function or set of data far more efficient. This code can be applied to any area of mathematics or data analysis that would otherwise use the Newton Raphson or Secant methods. Both methods on their own are adequate at finding roots for polynomial functions, but the combined adaptable method is advantageous for more complex equations and graphs. Furthermore, this combined method can be used to solve for both real and complex roots of a function.

An example of a polynomial that would utilize this code, found in Chapra and Canale's *Numerical Methods for Engineers* (2015), would be:

$$f(x) = x^5 - 16.05x^4 + 88.75x^3 - 192.0375x^2 + 116.35x + 31.6875 \quad (10)$$

This complex equation, given as a homework example in the aforementioned textbook, requests students use both the Newton Raphson method and the Secant method to find the roots of the equation within an error margin of 0.01%.

Using just the Newton Raphson method and an initial guess of $x = 0.5825$, a simple computer program calculates an approximate root of this function to be $x = 2.3000977355142824$. The Secant method, using the same initial guess and an upper value of $x = 1$ approximates a more accurate answer: $x = 1.6089601405037746$.

By itself, this problem would be far too complex to compute by hand. Both root-finding methods are capable of solving the equation on their own, but the combination of the methods provides us with the following, even more accurate answer $x = 1.499999$ ...

Beyond the calculation of complex equations, as seen above, the combined Newton-Raphson and Secant methods in this adaptable code can be used in the educational environment. It has applications in teaching students and as a basis for further research into root-finding methods. This method and its code are applicable in a broad range of mathematical subjects.

Conclusion

The modified Newton-Rapshon method was shown to be more efficient in calculating the root than its classic version. Due to the quick accuracy to approximate the root, on the modified version of the Newton-

Raphson method required mitigating the division by zero error. Therefore, for small computations, the modified Newton-Raphson method may be preferred

Using the classic Secant and Newton-Rapshon methods with consideration of the roots multiplicity, like the modified Newton-Raphson, proved to be the faster and more accurate method of calculating a root with multiplicity of two or more.

While the modified Secant method proved to be more accurate with initial guesses close to the actual root and with a small difference between them, the method's sensitivity to this closeness and differences makes it the less ideal method compared to the classic Secant method. Therefore, the best combination of open-bracket methods to quickly obtain a more accurate result would be that of the modified Newton-Raphson method and classic Secant with both accounting for the multiplicity of the root to be found.

References

Chapra, S. C., & Canale, R. P. (2015). *Numerical Methods for Engineers* (Seventh). McGraw-Hill Education.

Kreider, K. (n.d.). Introducton to Rootfinding. Akron; The University of Akron.

Smith, M. D. (1998, October 1). Newton-Raphson Technique. Retrieved April 3, 2022, from https://web.mit.edu/10.001/Web/Course_Notes/NLAE/node6.html

Papakonstantinou, J. M., & Tapia, R. A. (2013). Origin and evolution of the secant method in one dimension. The American Mathematical Monthly, 120(6), 500. https://doi.org/10.4169/amer.math.monthly.120.06.500