Analysis of Numerical Root-finding Methods

By Carl De Vries

Calculus II – Honors – Fall 2013

Introduction:

Numerical root-finding methods offer an alternative process for approximating roots of a function. Numerical methods can be used when an analytical solution is too complicated. Using a computer can speed up the numerical methods process. One implementation can be found on Texas Instruments graphing calculators. When searching for the roots on a graph, the calculator uses numerical methods to solve for the root. The purpose of this analysis is to identify the theories, nuances, and algorithms for four fundamental numerical root-finding methods. This paper covers both bracketing methods and open methods. The bracketing methods analyzed are Bisection and False Position and the open methods are Newton-Raphson and Secant. My analysis shows bracketing methods are more reliable, but slower than the open methods.

Theory:

Bracketing Methods

The Bisection and False Position methods are two examples of bracketing methods. The difference between the two methods is in the evaluation of the root approximation [x_n]. All bracketing methods require two initial guesses, a lower bound and an upper bound. The upper and lower bounds are assigned the variables [x_l] and [x_u], respectively. The bounds must surround or "bracket" a root. The number of iterations required to find an acceptable solution is dependent on the quality of the initial guesses.

Bracketing methods take advantage of a change in the function value f(x), from positive to negative, on either side of the root. After calculating a root approximation, the product of $f(x_n)$ and the function value at one bound can be used to identify which interval contains the root. Either bound will work, but for consistency, the lower bound [x_l] will be used. If the product of $f(x_l)$ and $f(x_n)$ is a negative number, one multiplicand must be negative and the other is positive. This means the x values, x_l and x_n, must bracket the root. If the product is a positive number, the lower bound and the root approximation don't straddle the root. Thus, the root is located between the root approximation and the upper bound. The root approximation then replaces the unneeded bound and the root finding process is repeated.

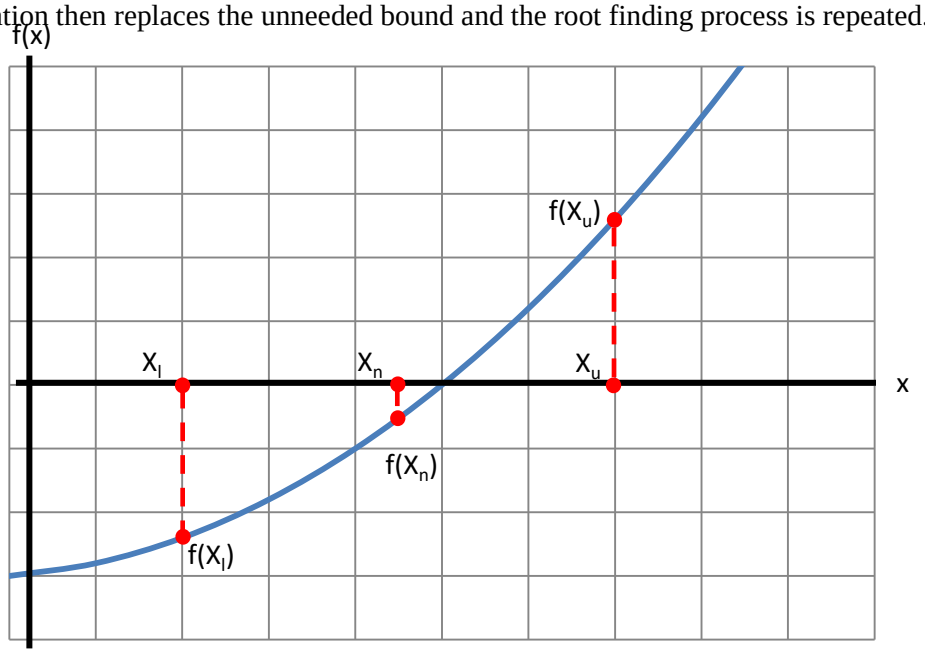

Figure 1: In this example, [$f(x_l) \cdot f(x_n) = +c$] and [$f(x_u) \cdot f(x_n) = -c$]. Therefore the root is bounded by x_u and x_n.

Bisection

The Bisection method is a binary approach to numerical root-finding. The root approximation is the midpoint of the upper and lower bounds.

$$x_n = (\frac{1}{2})(x_l + x_u)$$

The interval containing the root is reduced by half at the end of each iteration. The root approximation [x_n] approaches the true root with no regard to the function. Figure 2 is a graphical representation of the intervals and approximations.

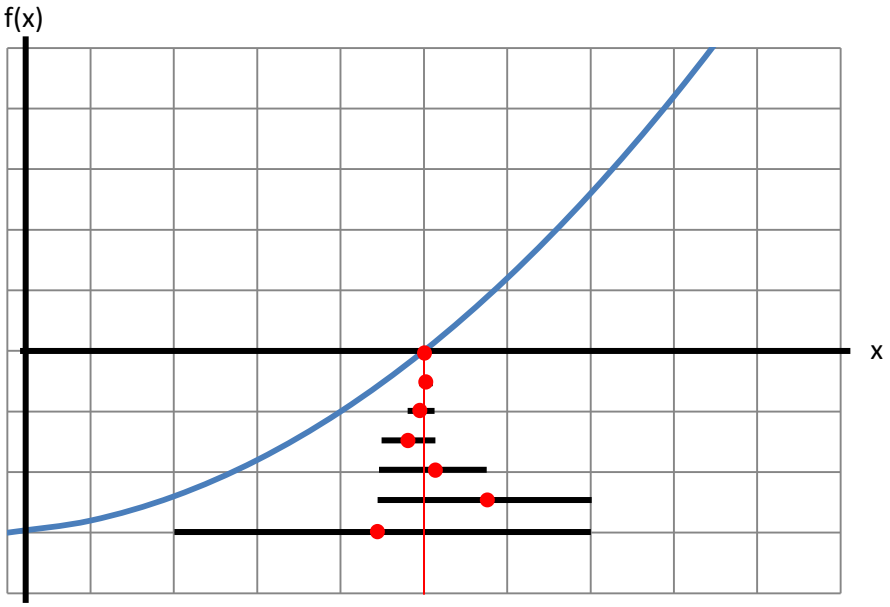

Figure 2: The black lines represent the current interval between the upper and lower bounds. The red dot represents the root approximation for the interval.

With Bisection, it's likely the root approximation will cross from one side of the root to the other. The root approximation may also approach the root and then recede slightly if one bound is much closer to the root than the other. An advantage of the Bisection method is that it will eventually converge on the root, but it may take longer than more sophisticated methods. In Figure 3, an examination of iterations three and four shows the approximation both oscillating about and slightly receding from the root.

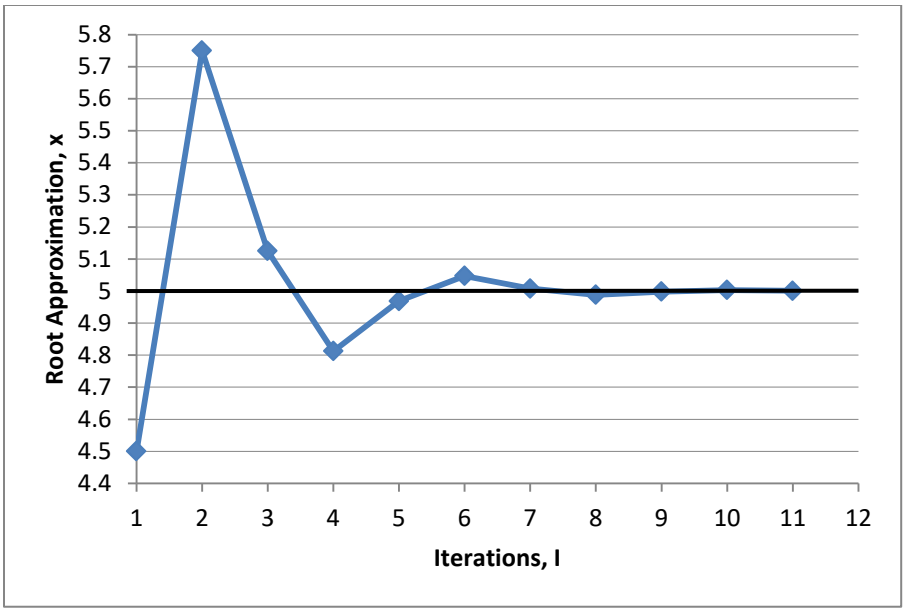

Figure 3: Root Approximation vs. Iteration for Bisection.

False Position

The False Position method estimates the root using a secant line connecting the upper and lower bounds. The method assumes the magnitude of a function, f(x), grows smaller as it approaches the root. Therefore, a bound is likely closer to the root if it evaluates to a smaller f(x).

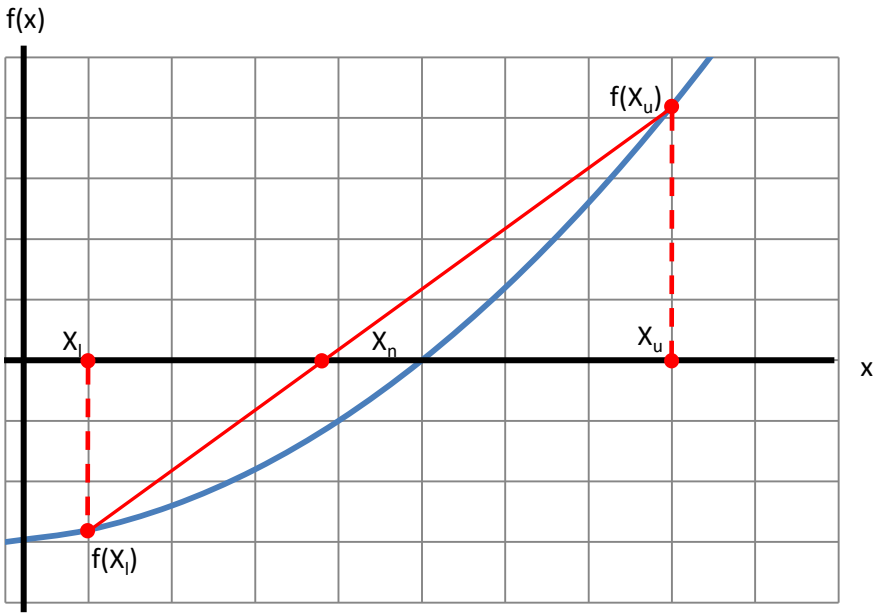

Figure 4: Drawing the secant line.

The secant line is created by the convergence of two separate lines at the point (x_n, 0). The first line starts from the point (x_l, f(x_l)), and the second line from (x_u, f(x_u)). An equation is be created by setting the slopes of both lines equal to each other. The following steps are used to derive the equation for the root approximation.

$$\frac{f(x_l) - 0}{x_l - x_n} = \frac{f(x_u) - 0}{x_u - x_n}$$

Create a slope equation that passes through the point (x_n, 0) for both the upper and lower bounds and them equal to each other.

$$\frac{f(x_l)}{x_n - x_l} = \frac{f(x_u)}{x_n - x_u}$$

Multiply both sides by -1 and distribute across the denominators.

$$f(x_l)(x_n - x_u) = f(x_u)(x_n - x_l)$$

Multiply the equation by both denominators.

$$f(x_l)(x_n) - f(x_l)(x_u) = f(x_u)(x_n) - f(x_u)(x_l)$$

Distribute both sets of terms.

$$f(x_l)(x_n) - f(x_u)(x_n) = f(x_l)(x_u) - f(x_u)(x_l)$$

Collect all x_n terms on one side.

$$x_n\big(f(x_l) - f(x_u)\big) = f(x_l)(x_u) - f(x_u)(x_l)$$

Factor out the x_n term.

$$x_n = \frac{f(x_l)(x_u) - f(x_u)(x_l)}{f(x_l) - f(x_u)}$$

Divide by the remaining terms to get x_n by itself.

Entering the upper and lower bounds and solving for [x_n] returns an approximation to the true root. The bounds must then be updated to reflect the new interval. Repeatedly solving for x_n and updating the interval will narrow the interval containing the root until x_n is an acceptable approximation of the actual root.

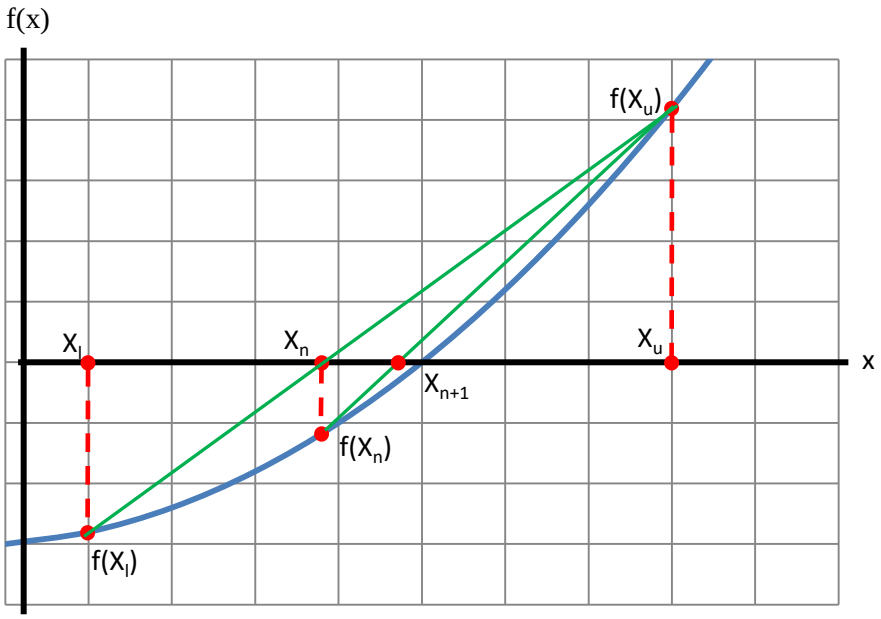

Figure 5: Shows two iterations of the False Position method. The first root approximation is x_n and the second is x_{n+1}.

False Position has several advantages over the Bisection method. First, False Position approximations tend to converge on the true root faster than Bisection, because each estimate accounts for the magnitude of $f(x_n)$ i.e., the nearness of $f(x_n)$ to the zero. Also, it is more likely x_n will approach the root from the same side. As with Bisection and other bracketing methods, an approximation is guaranteed as long as the true root is contained in the brackets.

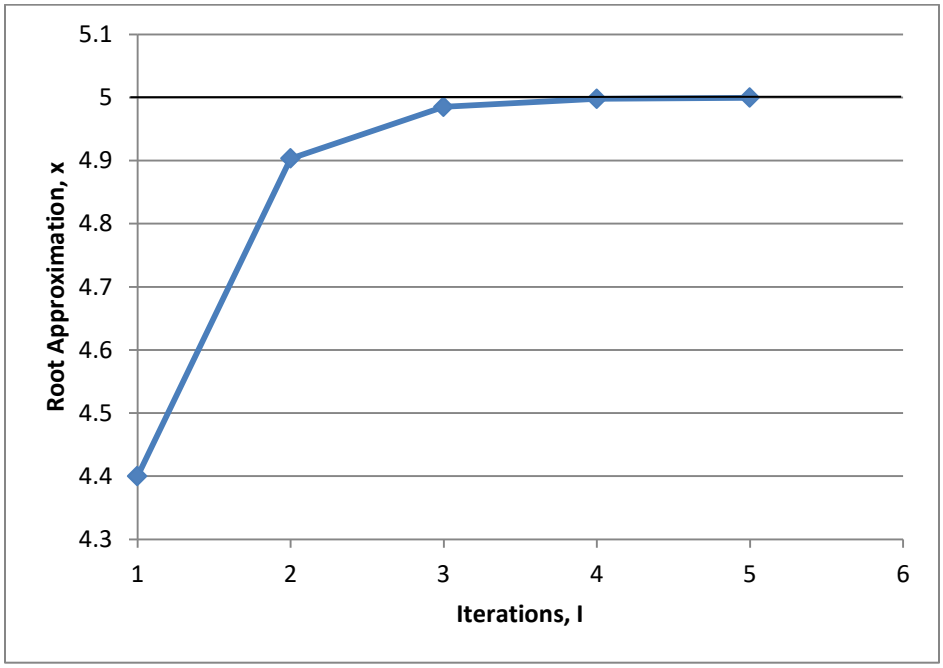

Figure 6: Root Approximation vs. Iterations for False Position.

Open Methods

The Newton-Raphson and Secant methods are both part of the open methods family. Open methods don't require initial guesses to bracket the root. Therefore, under some circumstances the approximation may diverge. This may be caused by the function itself or the quality of the initial guess. The advantage of open methods is they typically converge on the root faster than bracketing methods.

In some scenarios, the approximation will diverge from the true root. For example, a cubic function with roots that don't overlap (ex. not x^3) can cause problems. If a

Newton-Raphson

The Newton-Raphson method calculates root approximations using a function's derivative, f'(x). The method requires the analytical derivative. The analytical derivative is important, because using a slope approximation would make the equation nearly identical to the Secant method. Because the method is dependent on an analytical expression, it is not suitable for complicated functions. The following steps are used to derive the equation for the Newton-Raphson root approximation.

$$\frac{d}{dx}f(x_i) = f'(x_i)$$

Solve for the derivative of f(x).

$$f'(x_i) = \frac{f(x_i) - 0}{x_i - x_n}$$

Set the derivative equal to the slope of a tangent line. The tangent line is expressed using a point on the f(x) curve and the point where the tangent line crosses the x-axis.

$$f'(x_i)(x_i - x_n) = f(x_i)$$

Multiply each side by the denominator of the slope formula.

$$x_i - x_n = \frac{f(x_i)}{f'(x_i)}$$

Divide each side by the derivative f'(x).

$$-x_n = -x_i + \frac{f(x_i)}{f'(x_i)}$$

Subtract x_i from both sides of the equation leaving x_n by itself.

$$x_n = x_i - \frac{f(x_i)}{f'(x_i)}$$

Clean up the equations by multiplying by -1.

After solving for x_n, the final equation contains a ratio with the derivative f'(x) located in the denominator. This is an issue for any point on the curve with a slope of zero. Dividing f(x) by zero cannot be evaluated. This scenario can be represented graphically by drawing a line tangent to a point

with a slope of zero. The line never crosses the x-axis; therefore, a root-approximation cannot be calculated.

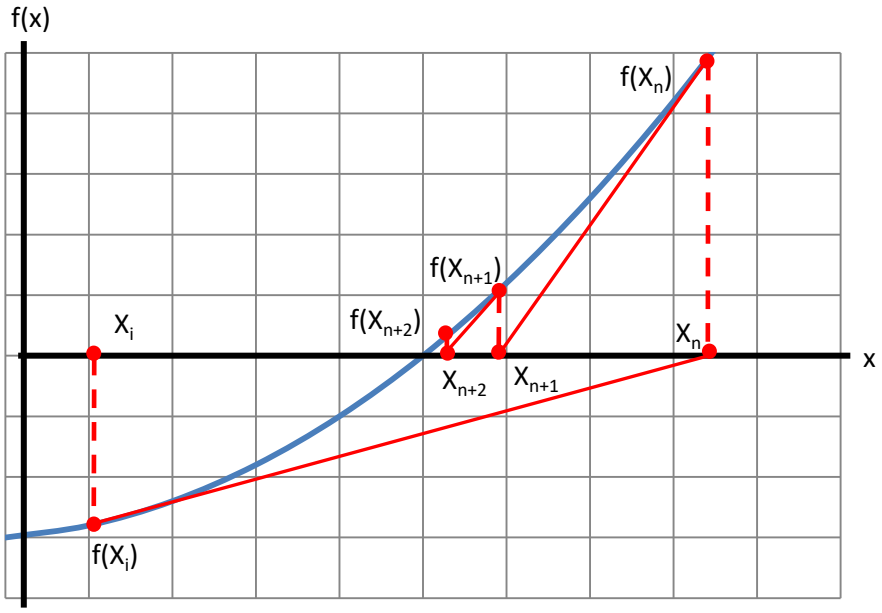

Figure 7: Newton-Raphson graphical representation

In Figure 7, it becomes clear how sporadic Newton-Raphson approximations can be. There is a large change between the initial guess [x_i],and the first approximation, [x_n]. The second and third approximations begin to converge nicely on the root of the function. The function in Figure 7 is concave up, so while the first approximation crosses the root, it will eventually converge back towards the root.

The number of iterations required for Newton-Raphson is dependent on the quality of the initial guess. If the initial guess were closer to the true root for the function in Figure 7, all subsequent approximations would be better, and the number of iterations to reach an accurate approximation would decrease. Conversely, a poor initial guess may increase the number of iterations. Compare the slope of $f(x_n)$ and the slope of the function at the true root. As the slope of the approximation approaches the slope of the true root, better approximations are made.

Secant

The Secant method is an open method, and requires two initial guesses [x_n] and [x_{n-1}]. However, unlike bracketing methods, the guesses are not required to bracket the root. This method uses a slope approximation to estimate the root. The slope approximation is created by evaluating the function at the two points [x_n] and [x_{n-1}]. In a graphical representation, a line is drawn which passes through both points and intersects the x-axis. The point where the line and x-axis intersect is the root approximation. The approximation becomes [x_n], and the earlier approximation now becomes [x_{n-1}]. The value of the function, $f(x_n)$, is calculated and then used

to determine a new slope. A line is drawn between the new point and the remaining point and the process is repeated. The Secant method addresses the disadvantage of needing an analytical derivative for Newton-Raphson. The Secant method equation can be derived from the Newton-Raphson equation using the following steps.

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

Start with the Newton-Raphson root approximation equation.

$$f'(x_n) \cong \frac{f(x_{n-1}) - f(x_n)}{x_{n-1} - x_n}$$

Use the idea that the derivative of f(x) is approximately equal to the change in two y coordinates over the change in the two corresponding x coordinates.

$$x_{n+1} = x_n - \frac{f(x_n)}{\frac{f(x_{n-1}) - f(x_n)}{x_{n-1} - x_n}}$$

Replace the analytical derivative with the slope approximation.

$$x_{n+1} = x_n - \frac{f(x_n)}{1} * \frac{x_{n-1} - x_n}{f(x_{n-1}) - f(x_n)}$$

Multiply $f(x_n)$ by the reciprocal of the denominator.

$$x_{n+1} = x_n - \frac{f(x_n)(x_{n-1} - x_n)}{f(x_{n-1}) - f(x_n)}$$

Combine the fractions to reach the final equation.

Coding:

All the numerical root-finding methods covered rely on repeating a series of steps to calculate increasingly accurate root approximations. Converting the algorithm for each method into computer code greatly reduces the time and effort needed to reach a reasonable approximation. The first step to create a computer programs is to identify an algorithm for the each method.

Algorithms:

The algorithms for all four methods followed a similar pattern. The first step in the algorithm is to collect required parameters. The first piece of data common to all the methods is the function, f(x). From this point, the data needed for each method varies slightly.

Both Bisection and False Position require two bounds which must bracket the root. The Secant method also requires two initial guess, but they aren't required to bracket the root. Newton-Raphson is the only method which requires only one initial guess.

The next decision is which stopping criteria to use. Each program uses a loop to repeat the approximation process. If the stopping criteria are met before the maximum number of iterations is reached, the program is terminates prematurely. The first stopping method calculates approximations until the function evaluated at the root, $f(x_n)$, is below a given tolerance , ε. The second method is a combination of the first stopping criterion and a minimum Δx value. The third stopping method calculates a value called approximate error and exits the program when the error drops below a given threshold. There is more on the individual stopping criteria in the following section.

When programming the root-finding methods it's important to include a parameter for the maximum number of iterations the program should try before terminating. There are situations where the stopping criteria may be unreasonable, the function may not cross the x-axis, or an open method loops through the same approximations. A limit to the number of iterations the method should perform ensures the method will stop at some point and not continue on infinitely. The value of max iterations will depend on the function and the level of accuracy desired.

Stopping Criteria:

The first stopping criterion calculates approximations until the function evaluated at the root approximation, $f(x_n)$, is below a given tolerance , ε (i.e., $|f(x_n) < \varepsilon$). The tolerance represents a magnitude on the y-axis. Once the absolute value of $f(x_n)$ is less than the given tolerance the function displays the root and number of iterations required. This method works well in most cases, but accuracy may suffer if the function approaches the x-axis asymptotically. The value $f(x_n)$ may satisfy the tolerance before the approximation x_n approaches the true root of the function. This pitfall can be examined in an extreme case by using the function $f(x)=1/x$. The function has a horizontal asymptote at the x-axis, and therefore never crosses the x-axis. Yet, with a tolerance of 0.001 and an initial guess of $x_0=1$, the Newton-Raphson method calculates an acceptable approximation at $x_{10}=1024$.

The second stopping criterion addresses the main disadvantage of the first for bracketing methods. The criterion still requires a tolerance (ε) representing a magnitude on the y-axis, but it also requires a value for the minimum difference between both bounds, Δx. Both criteria must be met before an estimate is considered acceptable. Requiring a minimum Δx decreases the x range for the location of the root approximation and brings about a more accurate answer.

The third stopping criterion uses a percent error approach to determine how close an approximation is to the true root. Traditional methods for calculating percent error require a measured value and a true value. In real applications the true root is not known, thus the need for an approximation

using numerical methods. This percent error approach calculates an approximate error [ε_a] using both the current and previous root approximations. The equation used is as follows:

$$\left|\frac{x_n - x_{n-1}}{x_n}\right| * 100 = \epsilon_a$$

A graph representing Iterations vs. Approximate Error and Iterations vs. True Error for Bisection shows the approximate error is consistently larger than the true error. This indicates that when the approximate error is within the accepted tolerance, the true error will be as well. The following figure shows the percent error for Bisection may fluctuate depending on how close the true root is to one of the bounds.

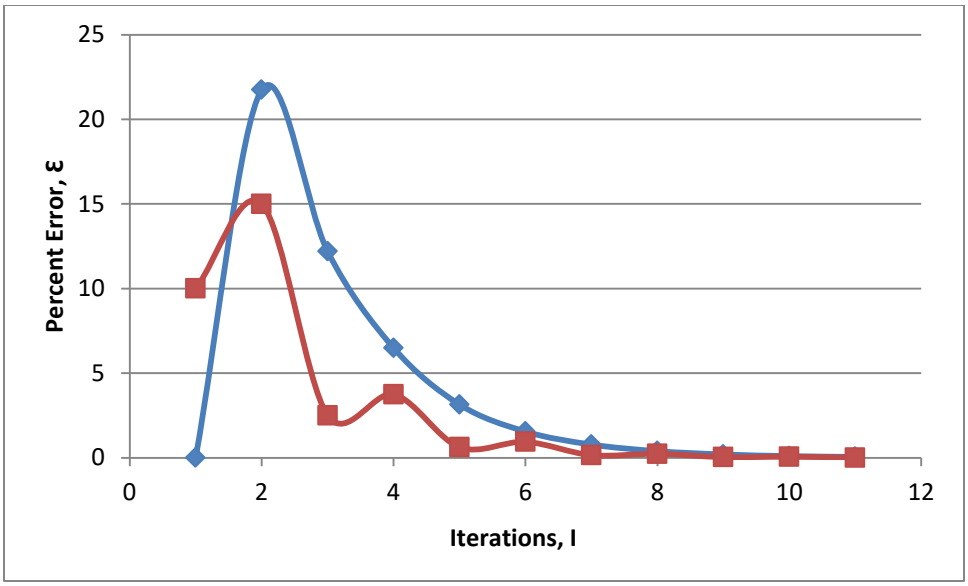

Figure #: Iterations vs. Percent Error for Bisection method.

Validation:

I tested my code using two different problems. For the first problem, I used a simple function with known roots. I tested each numerical method using the same equation for consistency. I used the function [$f(x)=x^2+x-30$] as my test case. The function factors to [$f(x)=(x+6)(x-5)$], and the roots are identified at [$x=5$] and [$x=-6$]. Below are the parameters and results using the Bisection program.

Parameters

Lower bound, L = 2
Upper bound, U = 7
Tolerance, E=f(x) = 0.001
Max Iterations, M = 20

Results

X_n = 5.000030518

I = 15

My second problem came from a case study in *Numerical Methods for Engineers*. The problem examines the transition between steady-states in a circuit. When a circuit changes state, the energy storage in capacitors and inductors oscillates. The oscillations eventually dissipate due to resistance in the circuit. The goal is to find the resistance needed to dissipate the fluctuation in charge when given values for the variables time, inductance, and capacitance. A second-order differential equation can be created to represent the fluctuations in charge. The differential equation solves to a function representing the charge in a circuit as a function of time, q(t). This equation is:

$$q(t) = q_0 e^{-Rt/2L}\cos\left(\sqrt{\frac{1}{LC} - \left(\frac{R}{2L}\right)^2} * t\right)$$

The problem specifies the charge must be dissipated to 1% of its initial value. The case study provides the constants:

Dimension	Value	Unit
charge/maximum charge, q/q_0	0.01	Percent
Time, t	0.05	Seconds, s
Inductance, L	5	Henries
Capacitance, C	10^{-4}	Farads

Rearranging the equation and entering the given values creates the following equation in terms of resistance. The accompanying graph displays the function as it crosses the x-axis.

$$f(R) = e^{-0.005R}\cos\left(\sqrt{2000 - 0.01R^2}(0.05)\right) - 0.01$$

The case study used the Bisection method to approximate the root, and identified the lower and upper bounds as 0 and 400, respectively. The stopping criteria for the problem was an approximate error of ε_a= 0.0001%. The answer given by the book is $x_{21} \approx 328.1515\ \Omega$. The Bisection program reached the same approximation in the same number of iterations. For comparison, the False Position method solved evaluated to $x_{17} \approx 328.1517\ \Omega$. Again, the False Position method was slightly faster than Bisection. In the open method family, Newton-Raphson is not an ideal choice, because solving for the derivative of the equation f(R) would be difficult. However, Secant method could be used with little difficulty.

Conclusion:

The analysis of all four root-finding methods demonstrates advantages and disadvantages for each family of methods. The bracketing methods are more reliable, because they will eventually reach an approximation, even at the cost of additional iterations. The open methods are less reliable, but tend to converge on the root faster than bracketing methods. The Newton-Raphson method may be difficult to use in some situations because it requires an analytical derivative. When choosing a method, time constraints must be weighed against the risk of the approximation diverging from the root. Combinations of these methods can be used in certain instances to reach guarantee an accurate approximation is reached quickly. The methods covered in this paper are basic numerical methods. There are many advanced methods which can be used to solve for the roots of a function; however they also have their advantages and disadvantages.