Graphical representation of the application of the bisection and secant methods for obtaining roots of equations using Matlab

Nancy Velasco¹,³, Dario Mendoza¹, Vicente Hallo¹, Elizabeth Salazar-Jácome¹ and Victor Chimarro²

¹Universidad de las Fuerzas Armadas Espe, Sangolqui-Ecuador
²Universidad Técnica de Machala, Machala-Ecuador
³Email: ndvelasco@espe.edu.ec

Abstract. In this work, a Graphical User Interface in Matlab is implemented that allows visualizing the process animation performed by the Bisection and Secant methods to find the roots of simple equations and more complex expressions since the manual calculation becomes tedious. This interface represents a useful tool for a better understanding of the behavior of these methods and an aid to teaching in engineering. The interface allows the user to enter the function to be analyzed, the initial values, the minimum allowed error, the maximum accepted iterations and control the animation speed. In addition, in each iteration it allows the user to visualize the approximate value of the root, the current error and iteration.

1. Introduction
The calculation of roots of an equation can be done iteratively through numerical method [1, 2]. The roots of an equation or zeros of a function \( f(x) \), are the \( x \) values, which annul the function, that is, the roots of the equation \( f(x) = 0 \). Although the quadratic formula is useful for solving the second-degree equation which we can find in problems such as area calculation or age calculations, there are many functions where roots cannot be easily determined as in polynomials of degree greater than two because they model more complex but common processes in engineering like balance of energy, balance of forces or Newton movement laws [3].

The roots of equations appear in the context of various common problems in the area of design and modelling, where you have to specify the properties or composition of a system to ensure that it is operating in the desired way. Thus, such problems often require the determination of implicit parameters, the parameters that cannot easily be cleared from the equation [4].

The solution is provided by numerical methods for root equations [5]. These methods are based on interval analysis [6] and they are methods for solving algebraic equations using a computer [7]. The interface presented in this paper uses the Matlab Software, whose main characteristics include matrix calculation, numerical analysis and graphic visualization [8]. Numerical methods can also be developed in other software such as Excel [9]. The numerical methods for calculating roots are classified in closed and open methods.

Closed methods take advantage of the fact that a function changes sign in the vicinity of a root. They are also called intervals, because two initial values are needed for the root. Among the closed methods available, this work implements the Bisection method since it is categorized as highly...
didactic. In this method, the user must give the lower and upper values to ensure that the root is between these two values. The family of methods for root clogging is discussed in [10] and includes the methods of Laguerre, Halley, Ostrowski, Euler and Newton. A new iterative fourth order method to find multiple roots of nonlinear equations is presented [11]. In [12] poses a new high-order method for calculating a root of an equation.

Open methods are based on formulas that require only a single start value or start with a couple of them. Among the methods that are classified as open this work uses the method of Secant [13] to compare it with the closed method of Bisection, the main difference is that the user does not necessarily have to give two initial values that enclose the root. Sometimes these methods diverge or move away from the true root as you progress in the calculation. However, when open methods converge, they generally do much faster than closed methods [4].

1.1. Closed method of bisection or bolzanh
Based on the user's need to find the root of a complex function, he is asked to enter the equation of the function f. This method is based on the fact that f(x) changes sign to each side of the root. In general, if f(x) is real and continues in the interval from the lower value x_l to the upper value x_u and also the function evaluated at the lower limit f(x_l), and the function evaluated at the upper limit f(x_u) have opposing signs, that is:

\[ f(x_l)f(x_u) < 0 \]  

Then there is at least one real root between x_l and x_u. It is an incremental search type in which the interval between x_l and x_u is always divided in half, resulting in the approximate root x_r:

\[ x_r = \frac{x_l + x_u}{2} \]  

For the location of the subinterval where the root is found it is used:
- If f(x_l)f(x_r) < 0 \( \rightarrow \) Root in the lower subinterval
- If f(x_l)f(x_r) > 0 \( \rightarrow \) Root in the upper subinterval
- If f(x_l)f(x_r) = 0 \( \rightarrow \) Root is equal to the approximate root x_r, the calculation stops.

The relative percentage error calculated e_r uses the value of the root in the current item x_{rl} and the value of the root in the previous iteration x_{rl-1}:

\[ e_r = \left| \frac{x_{rl} - x_{rl-1}}{x_{rl}} \right| \times 100\% \]  

1.2. The open method of the secant
The method requires the user to enter two initial values of x, these being x_{l-1} and x_l. A root approximation is predicted by extrapolating a Secant of the function to the x-axis, the point where this line intersects the x-axis represents an improved approximation of the root. The iterative equation uses the current values x_l and earlier x_{l-1}, as well as the evaluations in the function f(x_l) and f(x_{l-1}) respectively, for the calculation of the new approximate root x_{l+1}:

\[ x_{l+1} = x_l \frac{f(x_l)(x_{l-1}-x_l)}{f(x_{l-1})-f(x_l)} \]  

The approximate error is calculated in a manner similar to the Bisection method. Both the open method and the closed method are iterative, so that in order to reach the answer, they must perform several iterations, demanding that if they want to do it manually, it would take a lot of time.

2. Development of the graphical user interface GUI in Matlab
The graphical user interface created is shown in figure 1. The user must enter the function to be analyzed, in addition can select if the approach is done with the method of Bisection or Secant. Both methods require two initial values but only for the closed method of the Bisection it is necessary that
between these values the root be found. The interface allows to establish the minimum required error
or, in its absence, the number of maximum iterations possible for the processing, besides it allows to
establish the animation speed with which it is shown each iteration of the method, given this number
in seconds. In each iteration, the interface shows the approximate root, the error obtained and the
iteration until that moment.

Matlab GUIs are created using a tool called guide, the GUI Development Environment. This tool
allows a programmer to layout the GUI, selecting and aligning the GUI components to be placed in it.
Once the components are in place, the programmer can edit their properties: name, colour, size, font,
text to display, and so forth. When guide saves the GUI, it creates working program including skeleton
functions that the programmer can modify to implement the behaviour of the GUI. When guide is
executed, it creates the Layout Editor.

![Graphical user interface made in Matlab.](image)

**Figure 1.** Graphical user interface made in Matlab.

2.1. Blocks diagram
The code developed in Matlab is based on the block diagrams described in figure 2 for both the
bisection method and the secant method. The process is repeated until the calculated error is less than
desired or the maximum number of iterations has been reached.
Figure 2. Block diagram for the Bisection (left) and Secant (right) method. Includes data entry, calculations and obtaining the approximate root.

3. Tests and results
The graphical development of the methods is illustrated in figure 3. For the iteration method in each iteration the graph indicates the values below $x_l$ and superior $x_u$ and the approximate root $x_r$, it also tells the user if there is a change of sign between $f(x_l)$ and $f(x_u)$, which means that the root is in the lower subinterval $(x_l + x_u)/2$, thus reassigning the value higher than $x_r$, thus suppressing the search to a shorter interval. Conversely, if there is no change of sign, it means that the root is in the upper range and the reallocation is performed by assigning the value lower than $x_r$, thus also reducing the search interval of the root to the Half of its original value, the advantage is that as long as the two initial values enclose the root, the method can find the best approximate root.

For the Secant method, the user must enter two arbitrary values. The advantage of this method is that there is no restriction that the two values are returned to the root.

Drawing a red line (figure 3-right) joining the points $(x_l, f(x_l))$ and the value where the line intersects the x-axis is an approximate root by this method. The update of values for the next iteration is done by eliminating the previous value $x_{i-1}$. This method is usually faster than the Bisection, but the disadvantage of this method is that it does not always converge, that is, to find the root, since the two initial values can cause that with each iteration the error increases and the approximate root will move farther and farther from the real root.
Figure 3. Development of the Bisection Method (left) and Secant (right). The function used is \( x^2 - 7 \) with initial values 0 and 5, 3 iterations and desired error of 0.10.

Both algorithms were tested with various functions taking as data for error 0.05 or 10 iterations and the initial values are -5 and 0. The results obtained for the approximate root, error and iteration are summarized in table 1 for the Bisection method and in table 2 for the Secant method. The Bisection method works correctly as long as the initial values enclose the root, otherwise an error will be displayed in the calculation of 100% as shown by the example of the logarithmic function in table 1. The secant method is often used to refine the responses obtained with other techniques, such as the bisection method. As these methods require a good first approximation, they generally give a rapid convergence.

Table 1. Data obtained with the Bisection method.

| Function Type | Approximate root | Error  | Iteration |
|---------------|------------------|--------|-----------|
| Linear: \( 3x + 5 \) | -1.665 | 0.29326 | 10 |
| Quadratic: \( 9x^2 - 100 \) | -3.335 | 0.14641 | 10 |
| Cubic: \((x + 3)^3 - \sqrt{0.25}\) | -2.2021 | 0.22173 | 10 |
| Order 6: \((x - 1)^6 - 5\) | -0.30762 | 1.5873 | 10 |
| Trigonometric: \( tan(x - 1000)sin(x - 1) \) | -3.7354 | 0.13072 | 10 |
| Exponential: \( e^{-x} - 6.45 \) | -1.8604 | 0.26247 | 10 |
| Logarithmic: \( log(0.39x + 2.44) + 1 \) | -0.0048 | 100 | 10 |
| Combination: \( sin(-x + 1)e^{-\sqrt{-x+1}} - 0.1 \) | -1.6162 | 0.30211 | 10 |
| Combination: \( -\sqrt{[(x^3 - 4) + 25] + 3.95} \) | -3.3154 | 0.14728 | 10 |

Note that the method does not converge for the logarithmic function because the set limits are -5 and 0, but the root is not within these limits. If the lower limit is changed to -6, the approximate root
obtained is -5.3145, with an error of 0.11025 and a total of 10 iterations. If the iterations parameter is increased to 12, an error of less than 0.05% is obtained: 0.027571%.

We use the Secant method to obtain the result with the least number of iterations, but it is noted that for the last function shown in table 2 that represents a combination of square root function and polynomial, the error is 100% after the 10 iterations what represents that the method does not converge.

**Table 2. Data obtained with the Secant method.**

| Function Type | Approximate root | Error    | Iteration |
|---------------|------------------|----------|-----------|
| Linear: 3x + 5 | -1.667           | 0        | 2         |
| Quadratic: 9x^2 - 100 | -3.3333 | 0.00051  | 7         |
| Cubic: (x + 3)^3 - √0.25 | -2.2064 | 0.04032  | 7         |
| Order 6: (x - 1)^6 - 5 | -0.30766 | 0.00320  | 9         |
| Trigonometric: tan(x - 1000)sin(x - 1) | -5.3098 | 0.02091  | 8         |
| Exponential: e^{-x} - 6.45 | -1.8447 | 5.0365   | 10        |
| Logarithmic: log(0.39x + 2.44) + 1 | -5.3131 | 0.00447  | 8         |
| Combination: sin(-x + 1)e^{-x+1} + 0.1 | -1.6138 | 0.00283  | 5         |
| Combination: -√[(x^3 - 4) + 25] + 3.95 | 0.3511  | 100      | 10        |

In the last combination of functions, the method does not find the root because there are actually two roots in the interval between -5 and 0, so for this function the initial data are not a good approximation, the method is confused between the two. If the upper limit is changed to -3, the obtained root is -3.3202 with an error of 0.029134% with 4 iterations.

A comparison of the approximate error to find the root of the function (x + 2)^3 + 1 with the two methods is illustrated in figure 4. The error in the Secant method approaches zero more rapidly in approximately 8 iterations, while the Bisection method takes longer to descend to zero (approximately 13 iterations).

**Figure 4.** The approximate error is plotted against the number of iterations for both the closed Bisection method and the open method of the Secant.

4. **Conclusions**

In the interface, it can be seen that the Secant method operates in a smaller number of iterations with respect to the Bisection method, therefore, it uses less processing time.
Manual calculations can take a long time, but with the interface created, it saves time and can also serve as a tool to test the speed in the exercises to be solved by students convergence.

Both interfaces are illustrative, allowing students a better understanding of the step-by-step operation of the methods.

The interface allows limiting when processing ends through the desired error parameters or maximum iterations, since any of those criteria could be determinant in a particular application.

The interface controls allow to adapt to user needs, since if required for teaching, the animation speed should be low and if the response is required quickly, the animation speed should be high.

When it is not guaranteed that the initial data in the Bisection method are each on one side of the root, the interface displays a 100% error in the root search.

When the value of the approximate root goes away from the real value, the graphical interface displays a 100% error indicating this particular one.

It is demonstrated that software can be used to implement didactic aids for better (visual) compression of numerical methods that in many cases become unfriendly to teaching.

References
[1] Conte S D and Kahaner D K 2003 Numerical analysis (John Wiley and Sons Ltd)
[2] Traub Joseph Frederick 1982 Iterative methods for the solution of equations American Mathematical Soc.
[3] Lang M and Frenzel B C 1994 Polynomial root finding in IEEE Signal Processing Letters 1(10) 141-143
[4] Chapra Steven, C Canale, et al. 2007 Métodos numéricos para ingenieros McGraw-Hill
[5] Lindfield G and Penny J 2012 Numerical methods: using MATLAB (Academic Press)
[6] Moore R E 1979 Methods and applications of interval analysis Society for Industrial and Applied Mathematics
[7] Muller D E A 1956 method for solving algebraic equations using an automatic computer Mathematical Tables and Other Aids to Computation 10(56) 208-215
[8] Nakamura S 1997 Análisis numérico y visualización gráfica con MATLAB Prentice Hall
[9] Billo E J 2007 Excel for Scientists and Engineers: Numerical Methods Roots of Equations 147-188
[10] Hansen E and Patrick M 1976 A family of root finding methods Numerische Mathematik 27(3) 257-269
[11] Li S, Liao X and Cheng L 2009 A new fourth-order iterative method for finding multiple roots of nonlinear equations Applied Mathematics and Computation 215(3) 1288-1292
[12] Anderson N and Björck Å 1973 A new high order method of regula falsi type for computing a root of an equation BIT Numerical Mathematics 13(3) 253-264
[13] Wolfe P 1959 The Secant method for simultaneous nonlinear equations Communications of the ACM 2(12) 12-13