Root Finding Algorithms Visualizer

A Root Finding Algorithms Visualizer project is a software tool designed to help users understand and visualize the behavior of various numerical methods for finding roots of mathematical equations. Root finding is a fundamental concept in numerical analysis and is commonly used in fields such as engineering, physics, computer science, and more. The primary goal of this project is to provide a user-friendly interface where users can input mathematical functions and observe how root-finding algorithms iteratively converge to find solutions.

Here are the key components and details of a Root Finding Algorithms Visualizer project:

1. **User Interface**:
   - The project should have a user-friendly graphical interface where users can input mathematical functions they want to analyze.
   - Users can specify the initial guess for the root and choose from a selection of root-finding algorithms to apply to the function.
2. **Supported Algorithms**:
   - The visualizer should support multiple root-finding algorithms, such as:
     - **Bisection Method**: A simple algorithm that iteratively reduces the search interval by halving it.
     - **Newton-Raphson Method**: An iterative method that uses the derivative of the function to estimate the root.
     - **Secant Method**: A numerical approximation of the derivative used in the Newton-Raphson method.
     - **Regula Falsi (False Position) Method**: A variation of the bisection method that interpolates between two points.
     - Any other algorithms the project aims to support.
3. **Visualization**:
   - The core of the project is the visualization of how these algorithms work. Users should be able to see the following:
     - The mathematical function they input.
     - The initial guess on the graph.
     - The convergence process, showing how the algorithms iteratively adjust the guess to get closer to the root.
     - The progression of the search interval for methods like bisection.
4. **Convergence Criteria**:
   - The project should implement convergence criteria to stop the iteration when a sufficiently accurate root is found, e.g., a small tolerance or a maximum number of iterations.
5. **Error Estimation**:
   - Provide error estimation and display how the error decreases as the iteration progresses.
6. **User Control**:

- Allow users to control the speed of visualization, pause, resume, or step through the iterations.
- Offer the option to switch between different algorithms for comparison.

7. **Result Presentation**:
   - Display the final root that each algorithm converges to and provide a summary of the number of iterations taken.
8. **Error Analysis**:
   - Include error analysis tools that show how the choice of initial guess and algorithm affects convergence.
9. **Documentation and Tutorials**:
   - Include detailed documentation and tutorials on how to use the tool effectively, explaining the theory behind the algorithms and practical examples.
10. **Extensibility**:
    - Make the project extensible so that additional root-finding algorithms can be added in the future.