JSSE
JOURNAL OF SCIENCE & SCIENCE EDUCATION
Faculty of Science, Ubon Ratchathani University
Vol. 8 No. 1 (Jan. – Jun. 2025): 152-167
http://doi.org/10.14456/jsse.2025.13

Research Article in Science

Empirical Comparison of Root-Finding Algorithms between Classical Methods and Hybrid Method

Jiratchaya Jaisaardsuetrong[1] and Wasana Ngaogate[1,*]

[1]Department of Mathematics, Statistics and Computer, Faculty of Science, Ubon Ratchathani University

**Email: wasana.n@ubu.ac.th*

Received <6 November 2024>; Revised <17 February 2025>; Accepted <18 February 2025 >

Abstract

This research presents a novel method derived from combination of traditional techniques, namely the Bisection method, False position method and Edmond-Halley's method, to enhance the efficiency of root finding algorithms. The research also compares the performance of this traditional methods with the new hybrid method by coding in Python. The results show that the new method demonstrates superior efficiency than the classical methods. Furthermore, a classe's structure based on the strategy design pattern was developed for code implementation, facilitating systematic coding and improving maintainability and scalability of the algorithms.

Keywords: Bisection method; False position method; Edmond-Halley's method; Hybrid method; Strategy design pattern

บทความวิจัยทางวิทยาศาสตร์

การเปรียบเทียบเชิงประจักษ์ของอัลกอริธึมการหารากระหว่างวิธีดั้งเดิมและวิธีผสม

จิรัชยา ใจสะอาดซื่อตรง[1] และวาสนา เหง้าเกษ[1,*]

[1]ภาควิชาคณิตศาสตร์ สถิติ และคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยอุบลราชธานี

**Email: wasana.n@ubu.ac.th*

บทคัดย่อ

การวิจัยนี้นำเสนอวิธีการใหม่ที่พัฒนาขึ้นจากการผสมผสานเทคนิคดั้งเดิม ได้แก่ วิธีแบ่งครึ่งช่วง วิธีแก้ตำแหน่งผิด และวิธีของเอ็ดมอนด์-ฮัลเลย์ เพื่อเพิ่มประสิทธิภาพของอัลกอริธึมในการหาค่าราก การวิจัยยังเปรียบเทียบประสิทธิภาพของวิธีการแบบดั้งเดิมเหล่านี้กับวิธีผสมใหม่โดยการเขียนโค้ดในภาษาไพธอน ผลการวิจัยแสดงให้เห็นว่าวิธีการใหม่มีประสิทธิภาพสูงกว่าวิธีการแบบดั้งเดิม นอกจากนี้ ยังได้พัฒนาโครงสร้างของคลาสที่อิงกับการออกแบบแบบรูปกลยุทธ เพื่อการนำโค้ดไปใช้ที่อย่างเป็นระบบ เพื่อช่วยเพิ่มความสามารถในการบำรุงรักษาและการขยายตัวของอัลกอริธึม

คำสำคัญ: วิธีแบ่งครึ่งช่วง; วิธีแก้ตำแหน่งผิด; วิธีของเอ็ดมอนด์-ฮัลเลย์; วิธีผสม; แบบรูปกลยุทธ

Introduction

In solving the equation $f(x) = 0$, sometimes it is not possible to find the solution directly, as the equation might be difficult to solve. However, there are numerical methods that can approximate the root. The simplest method is the bisection method, which involves dividing the interval in half. Another widely used method is Newton's method, which uses the slope of the graph and an initial reference point to approximate the root. Another method is the false position method. There are several research articles that study the approximation of roots. Gemechu and Thota (2020), proposed new iterative algorithms aimed at determining the roots of nonlinear transcendental equations. These algorithms utilize nonlinear Taylor polynomial interpolation combined with a modified error correction term grounded in fixed-point principles. Furthermore, the study examines the potential to extend these higher-order iterative methods from single-variable cases to higher dimensions. Jun and Jeon (2019), extended the bisection method to solve nonlinear equations. The paper discusses convergence properties and iteration counts, and includes visual graphs. Sabharwai (2019), proposed a dynamic blend of the bisection method and the regula falsi (false position) method to enhance the performance of root-finding algorithms. The blended algorithm demonstrated superior performance, requiring fewer computational steps to converge and offering a robust solution for root-finding problems where classical methods may struggle. Tanakan (2013), presented a computational algorithm that enhances the traditional bisection method for solving nonlinear equations, with the goal of increasing both efficiency and accuracy in root approximation. Burden and Faires (2021), introduced hybrid algorithm that combines the strengths of the trisection method and the false position method. The results show that the proposed algorithm surpasses the secant, trisection, Newton-Raphson, bisection, and regula falsi methods, in terms of both iterations count and average runtime. Bogdanov and Volkov (2013), modified quadratic interpolation method for finding the roots of a continuous function is proposed, focusing on the positioning of a parabola that interpolates the original function. The method identifies the conditions under which two interpolating parabolas will be situated on opposite sides of the given function. Cortez *et al.* (2023), introduced two novel hybrid methods for solving nonlinear equations by leveraging classical techniques like bisection, trisection, and modified false position. These hybrid methods, termed bisection-modified false position and trisection-modified false position.

This research brings together three well-known methods: The Bisection method, the False Position method, and Edmond Halley's Method, to create a new approach called the Hybrid Bisection False Position Edmond method (or Hybrid method). The goal is to develop an optimized method that combines the strengths of each individual approach, leading to more precise and efficient solutions. By leveraging the stability of the Bisection method, the adaptability of the False Position method, and the fast convergence of Edmond Halley's Method, the method aims to achieve faster convergence while maintaining high accuracy. The research emphasizes how each method's unique advantages enhance the overall effectiveness of this hybrid approach, ultimately improving performance in solving nonlinear equations.

Research Objectives

1. To develop an efficient estimation method for finding the roots of nonlinear equations by refining and integrating traditional techniques to enhance accuracy.
2. Evaluate and compare the effectiveness of traditional methods and hybrid methods to identify which approach demonstrates superior performance.
3. Designing appropriate coding strategies to achieve effective results in solving complex problems.

Review of Literature

In numerical analysis, finding the roots of nonlinear equations is a critical task with significant applications in various fields. There are many methods in numerical method for finding roots, two widely used methods for this purpose are the bisection method and the false position method. The bisection method employs the intermediate value theorem to progressively narrow down an interval containing a root, ensuring convergence but often at a slower rate. On the other hand, the false position method enhances this process by applying linear interpolation, typically leading to faster convergence when the function behaves consistently near the root. In the following sections, we will explore the algorithms, advantages, and limitations of these methods, providing a clear understanding of their roles in solving nonlinear equations.

Traditional Root-Finding Algorithm methods

1. Bisection Method

The most basic method for finding roots of f(x) = 0 is Bisection method. The bisection method divides the interval in half at each step. Before starting to approximate the root, it is necessary to check if there is a root within the interval by examining the signs of the function at the endpoints. If the signs at the endpoints are opposite, then a root exists within that interval. Following each halving of the interval, the newly calculated x-value undergoes evaluation to verify whether the value represents the root or lies within an acceptable error margin close to the root. If the criteria remain unmet, the method proceeds by identifying the subinterval most likely to contain the root, based on examining the signs of the function at the endpoints and at the newly calculated x-value. Any pair of points with opposite signs indicates the presence of a root within that interval. The interval then undergoes further bisection to compute the next x-value, and the iterative process continues until achieving a satisfactory solution. The formula for approximate value of the root finding with Bisection method is $x = \frac{x_L + x_R}{2}$, where x_L is x-value on the left of the interval and x_R is x-value on the right of the interval.

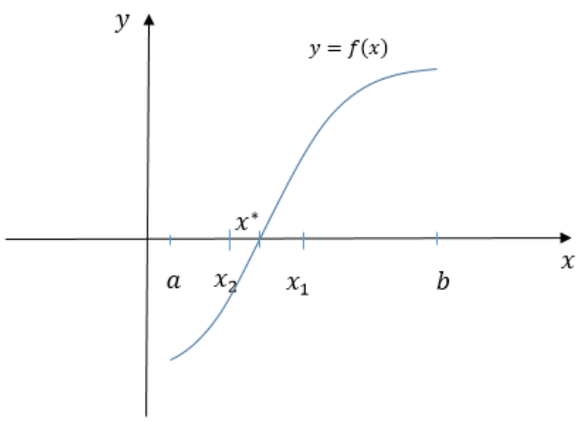

Figure 1 Bisection method

Figure 1 illustrates the bisection of the interval, with the initial interval is $[a, b]$. The first midpoint obtained is x_1, and the next interval to consider is $[a, x_1]$. This process continues, yielding x_2 and so forth, until a value close to or equal to the root x^* is reached.

The Bisection method, while conceptually straightforward, exhibits notable drawbacks. This method converges slowly, resulting in the potential loss of valuable intermediate approximations. However, the most significant advantage remains the guarantee of convergence to a solution. As a result, this method is frequently employed as a preliminary step before applying more efficient techniques, Burden and Faires (2001); Kincaid and Cheney (1990).

2. False Position Method

The False position method or Regular falsi method is an approximation technique for finding roots that begins with an initial interval, that requires examination to confirm the existence of a root. The two endpoints are then used to find the intersection point on the x-axis, yielding the first approximation of the root, denoted as x_1. Subsequently, the interval containing the root is considered, similar to the Bisection Method, by examining the signs of the function at the endpoints and at x_1. If the signs are opposite, a root exists within that interval. This process continues until a root or a value close to the root, within an acceptable error margin, is found. The formula for approximate value of the root finding with False position method is $x = \frac{x_L f(x_R) - x_R f(x_L)}{f(x_R) - f(x_L)}$, where x_L is x-value on the left of the interval and x_R is x-value on the right of the interval.

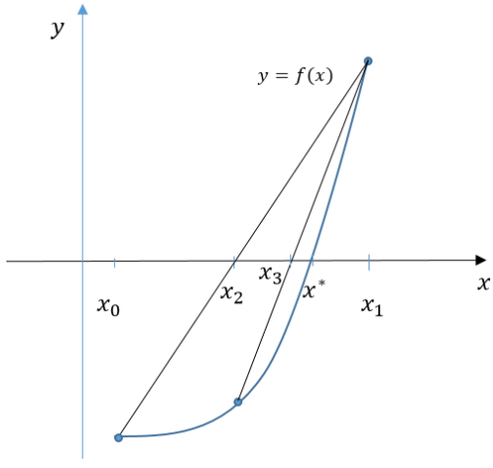

Figure 2 False position method

From Figure 2, the exact root value is x^*. The line connecting the points $(x_0, f(x_0))$ and $(x_1, f(x_1))$ intersects the x-axis at $(x_2, 0)$. The line formed by $(x_2, f(x_2))$ and $(x_1, f(x_1))$ intersects the x-axis at $(x_3, 0)$. This process continues until an approximate root value close to x^* or the true root is obtained.

3. Newton's method

The Newton's method or Newton Richardson's method uses the principle of the slope of the tangent line to the graph to find an approximate root value. At the beginning of root-finding, it is not necessary to specify an initial interval; only an initial approximation of the root is needed to estimate the next root value. The formula for estimating the next root value is $x_{i+1} = x_i - \frac{f(x_i)}{f'(x_{i+1})}$.

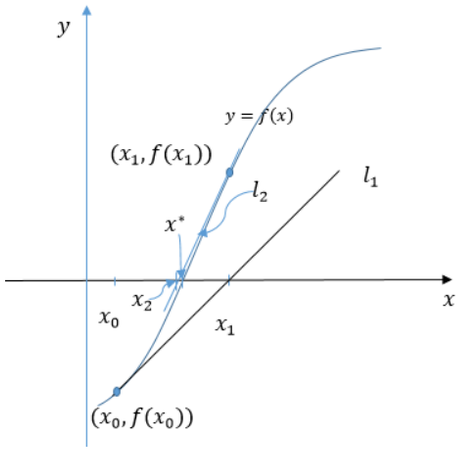

Figure 3 Newton's method

From Figure 3, the initial root approximation is x_0. The l_1 is the tangent line to the graph at the point $(x_0, f(x_0))$ and intersects the x-axis at $(x_1 0)$, where x_1 is the approximate root value. From $(x_1, f(x_1))$, the line l_2 forms the next tangent to the graph and intersects the x -axis at $(x_2, 0)$, where x_2 is the new root approximation. This process continues until the true root is obtained or an approximate root value within the desired error tolerance is reached.

4. Edmond-**Halley's Method**

Edmond-Halley' s Method or Halley' s Method is a root approximation technique that offers faster convergence than the Newton's method. While the formula resembles that of Newton's, this utilizes both the first and second derivatives for root estimation, Noor and Noor (2007). The Edmond-Halley method, named after the renowned English mathematician and astronomer who lived from 1656 to 1742, represents an extension of Newton's method through the incorporation of Taylor series expansion up to the second derivative. This method is particularly suitable for functions where the second derivative can be computed efficiently. In cases where the computation of the second derivative is challenging, Newton's method serves as a viable alternative, as it relies solely on the first derivative. This method requires only a single initial root approximation and does not need an initial interval. This method is particularly suitable for functions where calculating the first and second derivatives is straightforward.

To find $f(x) = 0$, Taylor's expansion can be used for the function $f(x)$, resulting in

$$f(x_i) + (x - x_i)f'(x_i) + \frac{(x-x_i)^2}{2}f''(x_i) = 0.$$

Therefore, $x_{i+1} = x_i - \frac{2f(x_i)f'(x_i)}{2(f'(x_i))^2 - f(x_i)f''(x_i)}$. This formula becomes representative of Newton's method when $f''(x_i) = 0$.

All four methods are effective in solving $f(x) = 0$. A comparison of their strengths and weaknesses is provided in the table 1.

Table 1 Comparison of root-finding method.

Method	Derivatives Required	Robustness	Speed	Applicability
Bisection	None	Highly robust	Slow	Suitable for any function that guarantees a root within the specified interval
False Position	None	Robust	Moderate	Appropriate for continuous functions, particularly those that are not excessively flat
Newton	First derivative	Sensitive to initial guess	Fast	Effective for functions with derivatives and when a good initial guess close to the root is available
Edmond-Halley	First and second derivatives	Sensitive to initial guess	Fast	Ideal for applications demanding highly accurate results and when second derivatives are accessible

The Hybrid Edmond-Halley Method

This section focuses on integrating and enhancing all three methods: bisection, false position and Edmond-Halley method to develop a new, more effective approach. The combined strengths of the bisection and false position methods will be utilized, as these methods rely on an initial interval where a root is assured to exist, guaranteeing a successful outcome.

Algorithm for Hybrid Edmond-Halley method (Hybrid method)

To solve $f(x) = 0$, this approach combines three methods: the Bisection method, the False Position method, and Edmond-Halley's method to improve efficiency. The bisection method is a reliable root-finding approach that ensures convergence when a sign change exists across the interval, though its convergence rate tends to be slow. In contrast, the false position method often achieves faster convergence through a secant-based approximation. While bisection is preferred for its reliability, false position is advantageous for faster convergence under suitable conditions. A hybrid approach that combines the strengths of both methods with an additional corrective technique, the Edmond-Halley method, can yield more efficient and accurate root approximations.

The process proceeds as follows:

1. Initialized Parameters:
   1.1 Define function $f(x)$ and the first and second derivatives of $f(x)$.
   1.2 Set the initial interval $[a, b]$.
   1.3 Define the tolerance *tol* and the maximum number of iterations *max_iter*.
2. Check Initial Interval Validity: If $f(a) \cdot f(b) < 0$, there is a root in the interval. If not, exit the algorithm.
3. Choose method: Use Bisection or False position only one iteration to find an approximated root x_i.
   3.1. If absolute error of f(x_Bisection) < absolute error of f(x_False Position), choose Bisection. Else choose False position.
   3.2 Check tolerance: if $|f(x_i)| < tol$, return x_i as the root.

4. Improve root: Use Edmond-Halley's method with initial guess x_i from 3.1.

Iterate to find root: For each iteration up to *max_iter*:

If both $f'(x_i)$and $f''(x_i)$ exist: $x_{i+1} = x_i - \frac{2f(x_i)f'(x_i)}{2f'(x_i)^2 - f(x_i)f''(x_i)}$. Evaluate $f(x_{i+1})$.

If x_{i+1} falls outside the interval $[a, b]$, reset the initial guess of Edmond- Halley' s method as the root from the previously selected root and repeat Edmond-Halley.

5. Check Convergence:

If $|f(x_{i+1})| < tol$, return x_{i+1} as a root.

6. Iteration Loop: Increment number of iteration. If number of iteration > *max_iter,* terminate and report no convergence.

7. Repeat Steps 3-6 until convergence is achieved

The hybrid method begins with the application of the bisection method and the false position method to estimate the root in the interval. If the estimated root does not meet the condition $|f(x_i)| < tol$, the root obtained from the method yielding the smaller absolute value of approximated root is selected as the initial root for the Edmond-Halley method. The Edmond-Halley method is then employed to refine the root approximation. The updated approximation is evaluated against the same convergence criterion, $|f(x_i)| < tol$. If this condition remains unmet, the process advances to the next iteration. If the updated root fall outside the interval, the root from the previous method is reselected as the initial guess for the Edmond-Halley method. This cycle continues until the condition $|f(x_i)| < tol$ is satisfied, at which point the root is obtained. Thus, each iteration consists of an initial estimation through the bisection and false position methods, followed by refinement of the root approximation using the Edmond-Halley method. The total number of iterations involves one iteration of the bisection method, one iteration of the false position method, and additional iterations from the Edmond-Halley method, as required to achieve convergence.

Advantages of the Hybrid Method

1. The hybrid method incorporates bisection and false position techniques to effectively manage cases where the initial guesses are far from the root, ensuring reliable starting approximations.

2. By utilizing the cubic convergence property of the Edmond-Halley method, the algorithm achieves rapid refinement of the root approximation, significantly reducing the number of iterations required near the root.

3. The method enhances reliability by seamlessly reverting to the more robust approaches of bisection or false position in situations where Newton's method or Edmond-Halley method fail to converge, thereby avoiding computational failure.

4. The hybrid approach dynamically selects the most appropriate method based on error minimization at each iteration, ensuring accuracy and efficiency throughout the root-finding process

Performance Evaluation

In order to test the efficiency of the Hybrid Edmond-Halley method, a series of examples will be employed with a tolerance of 10^{-10}. We will compare its performance to established techniques, including the bisection method, false position method, Newton's method, Edmond-Halley method, and Hybrid Edmond-Halley method. This approximation aims to highlight the strengths and weaknesses of each method in terms of accuracy and computational efficiency. There are 6 example tests.

1. $f(x) = 0.3x^2 - 4.5x - 15$, with exact solution is x =-2.807764064044151.
2. $f(x) = e^x + 0.2x^2 - 10$, with exact solution is x =-7.070767433290999.
3. $f(x) = x^2 + x - 48.75$, with exact solution is x =6.500000000000000.
4. $f(x) = \sin(x-1) + x^2 - 9$, with exact solution is x =2.834533011882032.
5. $f(x) = 10ln(x) + \frac{2}{x} - 5$, with exact solution is x =1.434103727265730.
6. $f(x) = x^3 + 9.41361457x^2 + 15.23940620x - 15.50505725$, with exact solution is x =0.696039315664213.

The results of these example tests are in Table 2-7.

Table 2 Approximate root of $f(x) = 0.3x^2 - 4.5x - 15$.

methods	Initial interval/ guess	Number of iterations	Root	Absolute error	CPU time (seconds)
Bisection	[-10,10]	32	-2.807764064054936	0.000000000066700	0.035902500152588
False position	[-10,10]	21	-2.807764064028455	0.000000000097076	0.000996589660645
Newton	-10	6	-2.807764064044152	0.000000000000000	0.000998258590698
Edmon Halley	-10	4	-2.807764064044152	0.000000000000000	0.002987146377563
Hybrid	[-10,10]	5	-2.807764064044151	0.000000000000000	0.001994609832764

Table 3 Approximate root of $f(x) = e^x + 0.2x^2 - 10$.

methods	Initial interval/guess	Number of iterations	Root	Absolute error	CPU time (seconds)
Bisection	[-10,0]	34	-7.070767433324363	0.000000000094336	0.041887283325195
False position	[-10,0]	16	-7.070767433281620	0.000000000026516	0.000996589660645
Newton	-10	5	-7.070767433290999	0.000000000000000	0.000996351242065
Edmon-Halley	-10	4	-7.070767433290999	0.000000000000000	0.003029823303223
Hybrid	[-10,0]	5	-7.070767433291000	0.000000000000003	0.035856962203979

Table 4 Approximate root of $f(x) = x^2 + x - 48.75$

methods	Initial interval/guess	Number of iterations	Root	Absolute error	CPU time (seconds)
Bisection	[-10,0]	38	6.500000000021828	0.000000000025466	0.031914949417114
False position	[-10,0]	13	6.499999999995794	0.000000000058876	0.000954151153564
Newton	-10	5	6.500000000000000	0.000000000000000	0.001015901565552
Edmon-Halley	-10	4	6.500000000000000	0.000000000000000	0.003029823303223
Hybrid	[-10,0]	6	6.500000000000000	0.000000000000000	0.003989458084106

Table 5 Approximate root of $f(x) = \sin(x-1) + x^2 - 9$

methods	Initial interval/guess	Number of iterations	Root	Absolute error	CPU time (seconds)
Bisection	[-2,4]	37	2.834533011860912	0.000000000003829	0.047030448913574
False position	[-2,4]	14	2.834533011879617	0.000000000013061	0.000996589660645
Newton	4	5	2.834533011882032	0.000000000000000	0.001000165939331
Edmon-Halley	4	4	2.834533011882032	0.000000000000000	0.002995967864990
Hybrid	[-2,4]	5	2.834533011881290	0.000000000004036	0.075797080993652

Table 6 Approximate root of $f(x) = 10lnx + \left(\frac{2}{x}\right) - 5$

methods	Initial interval/guess	Number of iterations	Root	Absolute error	CPU time (seconds)
Bisection	[0.2,4]	34	1.434103727276670	0.000000000065662	0.057822227478027
False position	[0.2,4]	24	1.434103727274182	0.000000000050734	0.000995874404907
Newton	0.2	Exceeded maximum iterations	Failed to identify the root	-	-
Edmon-Halley	0.2	-	Failed to identify the root	-	-
Hybrid	[0.2,4]	5	1.434103727265730	0.000000000000000	0.067799091339111

Table 7 Approximate root of $f(x) = x^3 + 9.41361457x^2 + 15.23940620x - 15.50505725$

methods	Initial interval/guess	Number of iterations	Root	Absolute error	CPU time (seconds)
Bisection	[0,10]	33	0.696039315662347	0.000000000055609	0.042885065078735
False position	[0,10]	Exceeded maximum iterations	Failed to identify the root	-	-
Newton	10	8	0.696039315664213	0.000000000000000	0.001022577285767
Edmon-Halley	10	6	0.696039315664213	0.000000000000000	0.024210691452026
Hybrid	[0.2,4]	5	0.696039315664213	0.000000000000000	0.002997159957886

Performance testing of the Hybrid Edmond-Halley method (hybrid method) across various types of functions reveals mixed results regarding time consumption. In some cases, it requires less time to find the root, while in others, it takes more time, with moderate results in certain examples. However, a clear advantage, as shown in Tables 2–7, is the hybrid method's consistent success in approximating the root across all test cases, unlike other methods that occasionally fail. For example, the Edmond-Halley method fails in Table 6, whereas the hybrid method successfully finds the root. Furthermore, in Table 6, the Newton method exceeds the iteration limit, and in Table 7, the false position method also surpasses the iteration limit of 1,000 iterations. Additionally, the hybrid method employs a number of iterations comparable to Newton's method and Edmond-Halley's method but fewer than the bisection and false position methods. Overall, the hybrid method demonstrates exceptional efficiency and the ability to handle all problem scenarios effectively.

Strategy Design Pattern

The Strategy Design Pattern addresses the growing complexity of mathematical problems in fields like computational mathematics, data science, and engineering by offering a flexible framework for algorithm management. This approach allows for seamless integration and interchange of diverse mathematical methods without altering the core codebase, significantly improving software maintainability, scalability, and efficiency. By enabling dynamic selection of appropriate algorithms based on specific contexts or constraints, it overcomes the limitations of traditional, rigid software solutions. This bridging of software design principles with mathematical problem-solving not only enhances system flexibility but also fosters innovation in computational mathematics, providing a versatile framework for managing algorithms across a wide range of applications and facilitating rapid advancements in the field.

Experts can provide class diagrams as recommended practices in addition to the fundamental ones made by software engineers. The most popular best practices are the Gang of Four (GoF) design patterns, which were first presented by Gamma *et al*. (1994). The programs written using most of the design patterns were simpler compared to the programs written without using design patterns, Qamar and Malik (2020). A family of algorithms is defined by the Strategy Design Pattern, Sarcar (2022), which encapsulates and renders replaceable each algorithm. It permits variations in the algorithm that are not dependent on the clients using it. Also, the clients ought not to be aware of the data. Avoiding exposing intricate, algorithm-specific data structures is encouraged by the Strategy pattern. It describes, enumerates, and makes a family of algorithms interchangeable. Khairin, Kusumo and Priyadi (2022), analyzed the impact of design patterns on mobile application performance. They found that design patterns can affect application performance depending on the design pattern used. The Strategy pattern and Visitor pattern optimize memory usage by 1%. Ngaogate *et al.* (2024), suggested the Strategy pattern for implementation of two machine learning techniques, Back propagation and Hybrid with Fixed point.

This study employs five distinct algorithms to determine root values for various functions. Implementing these algorithms in Python is essential to enable the mathematician to more easily assess their efficiency. To support this, we present the following class structure and strategy design for code implementation in the research on comparing machine learning algorithms, as shown in Figure 4.

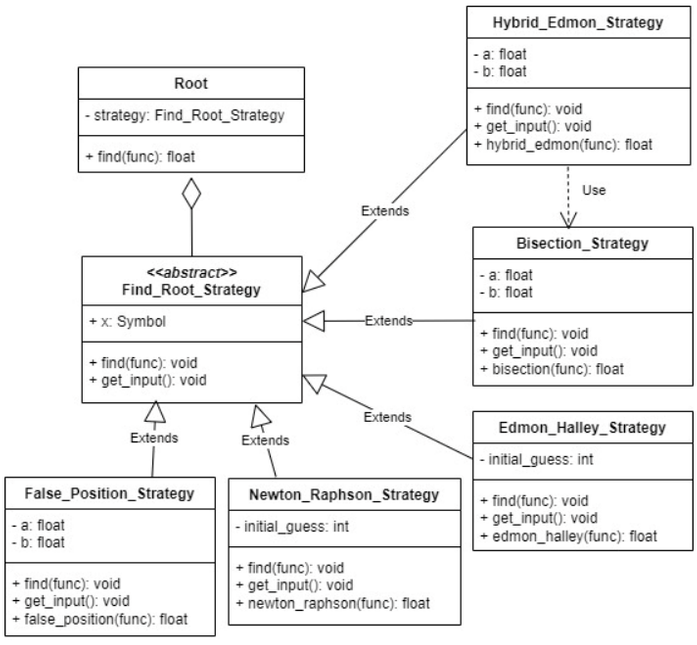

Figure 4 classes' structure based on the strategy design pattern

The Strategy pattern enhances flexibility and algorithm switching between methods dynamically without changing the structure of the underlying code. The main program simply calls each method by using the same instruction as below.

```
1.  # get function from user
2.  func = input("function: ")
3.
4.  x = symbols('x')
5.
6.  # call Bisection method
7.  root = Root(Bisection_Strategy(x))
8.  root.find(func)
9.
10. # call Hybrid Edmon method
11. root = Root(Hybrid_Edmon_Strategy(x))
12. root.find(func)
```

The codes at lines 7 and 11 above demonstrate how simple it was to switch between the Bisection approach and the Hybrid Edmond-Halley's method. Additionally, the Edmond- Halley, Newton Raphson, and False position were the three additional research methodologies that were simply called.

The Root class contains “strategy” as an attribute in order to support polymorphism of strategies defined by the Find_Root_Strategy class.

```
class Find_Root_Strategy (ABC):
        # attribute
    x = symbols('x')
    @abstractmethod
    def find(self, func):
        pass
    @abstractmethod
    def get_input(self):
        pass
```

Furthermore, the Strategy Pattern improves the mathematical software's maintainability and extensibility because it allows us to easily add a new class that implements the necessary method while maintaining the same interface and integrate it into the current system without changing other code. For instance, all we would have to do is develop a new class that implements the Find_Root_Strategy interface if a new algorithm called the Fixedpoint method were created.

The FixedPointStrategy : another mathematical method for finding Root

```
class FixedPointStrategy(Find_Root_Strategy):
    def find(self, func):
        its algorithms
```

The Strategy design pattern brings several benefits to mathematical applications by decoupling algorithm selection from core logic. This approach reduces code duplication, improves modularity, and streamlines testing. By encapsulating each algorithm in its own class, the codebase becomes more modular and easier to understand. Modularity is essential for testing and validation, allowing for the independent testing of individual algorithms. This separation simplifies debugging and ensures errors are traceable to specific algorithms rather than being intertwined within the larger system. For instance, to verify the functionality of the Hybrid Edmond method, the Hybrid_Edmon_Strategy class can be unit tested independently, without concern for the rest of the system.

Conclusion

The hybrid method integrates the bisection method, false position method, and Edmond-Halley method, utilizing the strengths of each to efficiently approximate roots. In all seven test cases, the method requiring the fewest iterations was the Hybrid Edmond-Halley method, followed by Newton’s method, the false Position method, and the bisection method, respectively, under the same tolerance of 10^{-10}. In some cases, the Edmond-Halley method and Newton's method fail to find roots due to the selection of an initial guess. A poorly selected initial guess can obstruct the convergence of these methods to a root. The Hybrid Edmond-Halley method addresses this limitation by incorporating adjustments that ensure the selection of

a suitable initial guess, enabling successful root-finding. Additionally, methods such as bisection and false position often require an excessive number of iterations in some cases. The Hybrid Edmond-Halley method, however, achieves convergence with a significantly reduced number of iterations when compared to these methods.

The Hybrid Edmond-Halley method provides an exceptionally efficient solution for root-finding, requiring very few iterations. Employing a hybrid framework effectively addresses the convergence limitations of both Newton's method and the Edmond-Halley method. Additionally, the computational time remains practical, underscoring reliability and suitability for practical implementations.

For coding, the Strategy design pattern is a valuable tool for building adaptable mathematical software systems. By decoupling algorithms from their clients, it allows for seamless switching between different solving methods without requiring extensive modifications. This flexibility is crucial in fields like computational fluid dynamics, data approximation, and machine learning, where problem characteristics or performance requirements may necessitate different approaches. The Strategy design pattern also enhances maintainability by reducing code duplication and promoting modularity, making it easier to integrate new algorithms and maintain existing ones.

Future work may involve developing a hybrid approach for root approximation, combining various methods to overcome the shortcomings of traditional techniques.

References

Bogdanov, V. V. and Volkov, Y. S. (2013). A modified quadratic interpolation method for root finding. **Journal of Applied and Industrial Mathematics**, 17(3), 491-497.

Burden, R. L. and Faires, J. D. (2001). **Numerical Analysis**. USA: Brooks/Cole.

Cortez, M. V., Ali, N. Z., Khan, A. G. and Awan M. U. (2023). Numerical analysis of new hybrid algorithms for solving nonlinear equations. **Axioms**, 12(7), 684.

Gemechu, T. and Thota, S. (2020). On new root finding algorithms for solving nonlinear transcendental equations. **International Journal of Chemistry Mathematics and Physics**, 4(2), 18-24.

Gamma, E., Helm, R., Johnson, R. and Vlissides J. (1994). **Design Patterns: Elements of Reusable Object-Oriented Software**. Addison-Wesley. Indianapolis: Addison-Wesley.

Jun, Y. and Jeon, J. (2019). Modified bisection method for solving nonlinear equations. **International Journal of Scientific and Innovation Mathematical Research**, 7(9), 8-11.

Khairin, A., Kusumo, D. and Priyadi, Y. (2022). Analysis of The Impact of Software Detailed Design on Mobile Application Performance Metrics. Building of Informatics. **Technology and Science (BITS)**, 4(1), 226–234.

Kincaid, D. and Cheney, W. (1990). **Numerical analysis mathematics of scientific computing**. USA: Brook/Cole.

Ngaogate, W., Jean, A., Wattanataweekul, R., Janngam, K. and Alherbe, T. (2024). Hybrid Machine Learning Algorithm with Fixed Point Technique for Medical Data Classification Problems Incorporating Data Cryptography. **Thai Journal of Mathematics**, 22(2), 295–310.

Noor, K. I. and Noor, M. A. (2007). Predictor-Corrector Halley method for nonlinear equations. **Applied Mathematics and Computation**, 188, 1587-1591.

Qamar, N and Malik, A. A. (2020). Impact of Design Patterns on Software Complexity and Size. **Mehran University Research Journal of Engineering and Technology**, 39(2), 342-352.

Sabharwai, C. L. (2019). Blended root finding algorithm outperforms bisection and regula falsi algorithm. **Mathematics**, 7(11), 1-16.

Sarcar, V. (2022). **Java Design Patterns: A Hands-On Experience with Real-World Examples (Third Edition)**. USA: Apress.

Tanakan, S. (2013). A new algorithm of modified bisection method for nonlinear equation. **Applied Mathematical Sciences**, 7(123), 16107-16114.