IMPLEMENTATION OF BISECTION METHOD USING PYTHON FOR SOLVING NONLINEAR EQUATIONS

Shweta Sachin Bibave[1]
[1]*Department of Mathematics,*
S. N. Arts, D. J. Malpani Commerce and B. N. Sarda Science College (Autonomous)
Sangamner, Ahilyanagar 422 605, Maharashtra – India.

Abstract

This paper presents the implementation of the Bisection Method using Python to find the roots of nonlinear equations. The method is a simple and robust numerical technique based on the Intermediate Value Theorem. A Python-based approach is used to visualize the convergence of the method through graphical representation. The results demonstrate accuracy and efficiency in approximating roots.

Keywords

Bisection Method, Python, Numerical Methods, Root Finding, Nonlinear Equations.

Introduction

The Bisection Method is one of the simplest numerical methods used to find the root of a nonlinear equation. It is based on the principle that if a continuous function changes sign over an interval, then a root must exist within that interval. This method is widely used due to its reliability and ease of implementation, especially for beginners in numerical analysis.

Finding the roots of nonlinear equations is a fundamental problem in numerical analysis with broad applications in science, engineering, economics, and applied mathematics. Since many real-world problems cannot be solved analytically due to the complexity of the equations involved, numerical methods are essential tools for obtaining approximate solutions [1]. Of the many numerical techniques available, the Bisection Method stands out as one of the most straightforward, dependable, and simple approaches for root-finding problems.
The Intermediate Value Theorem, a key idea in calculus, is the foundation of the Bisection Method. In accordance with this theorem, if a continuous interval exists and if a continuous function exists within this interval, then a continuous function exists [3]. The technique selects the subinterval where the sign shift takes place by periodically splitting the interval in half. Until the root is estimated with the required degree of accuracy, this iterative process is repeated. As a result of its Bisection.

The Bisection Method is useful in numerical calculation despite its simplicity, particularly where dependability is more crucial than speed [5]. Unlike methods such as Newton-Raphson or Secant Method, which require derivatives or may fail under specific conditions, the Bisection Method ensures convergence as long as the function is continuous and the beginning interval is suitably set. Because of its resilience, it is very helpful for teaching and for resolving issues where other approaches might not be appropriate.

Programming languages like Python have become into effective tools for implementing numerical algorithms as computing technologies have advanced. Python offers a set of mathematical calculation [2]. Python allows for the efficient implementation of complex numerical algorithms and the graphical visualization of their behaviour, which improves conceptual understanding.

Visualization of convergence is an important part of understanding a solution. The Bisection Method's graphical representation shows the root's progressive approximation and the shrinking interval, clearly illustrating the method's operation and aiding comprehension. Python modules such as Matplotlib and NumPy effectively visualize functions and iterative processes, making them suitable tools for such purposes [7].

Combining theoretical ideas with computational tools is also consistent with contemporary teaching methods that prioritize conceptual clarity and active learning [6]. By integrating the Bisection Method with Python-based visualization, learners may bridge the gap between theory and practice. Because it enables learners to experiment with various functions, intervals, and tolerance levels, this method is very helpful for students studying numerical methods because it helps them gain a deeper knowledge of the algorithm.
The primary objective of this study is to implement the Bisection Method using Python and to analyse its effectiveness in solving nonlinear equations. The study also focuses on visualizing the convergence process through graphical representations, providing a clear insight into how the method approaches the root. By evaluating the accuracy and efficiency of the method, this work aims to highlight its practical significance and applicability.

Furthermore, this paper contributes to the growing body of research that integrates computational techniques with mathematical problem-solving. It demonstrates how programming can be used not only as a tool for computation but also as a medium for enhancing understanding. The results obtained through this implementation confirm that the Bisection Method remains a valuable and dependable technique in numerical analysis.

Example:

Find a real root of equation $x^3 - 2x - 5 = 0$.

Let $f(x) = x^3 - 2x - 5$. Then $f(2) = -1$ and $f(3) = 16$.

Hence a root lies between 2 and 3 we take

$$x_0 = \frac{2+3}{2} = 2.5$$

Since $f(x_0) = 5.6250,$ we choose $[2, 2.5]$ as the new interval. Then,

$$x_1 = \frac{2+2.5}{2} = 2.25 \text{ and } f(x_1) = 1.89625$$

Proceeding in this way, the following table is obtained.

n	a	b	x	$f(x)$
1	2	3	2.5	5.6250
2	2	2.5	2.25	1.8906
3	2	2.25	2.125	0.3457
4	2	2.125	2.0625	-0.3513
5	2.0625	2.125	2.09375	-0.0089
6	2.09375	2.125	2.10938	0.1668
7	2.09375	2.10938	2.10156	0.07856
8	2.09375	2.10156	2.09766	0.03471
9	2.09375	2.09766	2.09570	0.01286

ISBN – 978-81-994619-6-3 | https://rchubpublisher.com/dm-sdg/

10	2.09375	2.09570	2.09473	0.00195
11	2.09375	2.09473	2.09424	-0.0035
12	2.09424	2.09473		

At $n = 12$, it is seen that the difference between two successive iterates is 0.0005, which is less than 0.001.

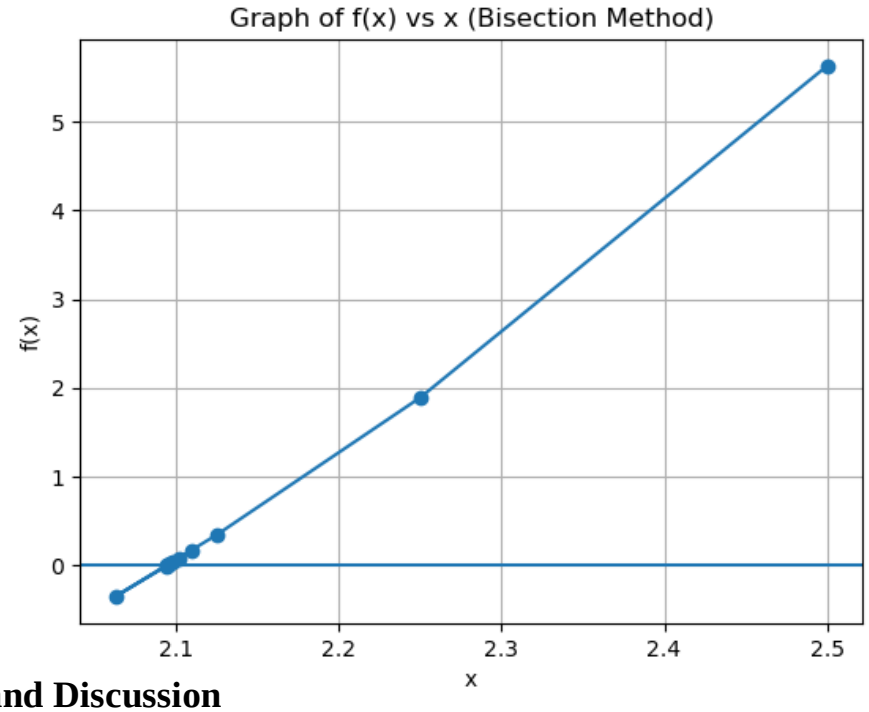
Now the novelty of this research work is that we use python programming to show above iteration in graphical method.

```
import matplotlib.pyplot as plt

x = [2.5, 2.25, 2.125, 2.0625, 2.09375, 2.10938, 2.10156, 2.09766, 2.09570,
2.09473, 2.09424]
fx = [5.6250, 1.8906, 0.3457, -0.3513, -0.0089, 0.1668, 0.07856, 0.03471,
0.01286, 0.00195, -0.0035]

plt.plot(x, fx, marker='o')
plt.axhline(0)  # x-axis
plt.xlabel("x")
plt.ylabel("f(x)")
plt.title("Graph of f(x) vs x (Bisection Method)")
plt.grid()

plt.show()
```



Result and Discussion

The Bisection Method was successfully implemented using Python to approximate the roots of selected nonlinear equations. The method was tested on continuous functions over specified intervals where a change in sign was observed, ensuring the existence of at least one root

according to the Intermediate Value Theorem. The implementation demonstrated consistent and reliable convergence toward the root with each iteration.

The results obtained show that the Bisection Method produces accurate approximations of roots within a predefined tolerance level. As the number of iterations increased, the interval containing the root progressively decreased, leading to improved precision. The stopping criteria based on tolerance and maximum iterations were effectively applied to control the accuracy and computational effort. In all test cases, the method converged to the expected root values, confirming its correctness and robustness.

Graphical visualization played a significant role in analysing the convergence behaviour of the method. Using Python libraries such as Matplotlib, the function curves and the midpoints generated during each iteration were plotted. These visualizations clearly illustrated how the interval was repeatedly halved and how the approximated root approached the actual root. The shrinking interval and movement of midpoints toward the root provided an intuitive understanding of the algorithm's working process.

From a computational perspective, the Bisection Method was found to be efficient for problems where guaranteed convergence is required. Although the rate of convergence is relatively slower compared to methods like Newton-Raphson, its simplicity and reliability make it a preferred choice when derivative information is unavailable or when the function is not well-behaved. The method does not depend on initial guesses being very close to the root, which adds to its practical usability.

The discussion also highlights that the accuracy of the method depends on the selection of the initial interval and the tolerance level. A smaller tolerance leads to higher accuracy but requires more iterations, thereby increasing computational time. Conversely, a larger tolerance reduces the number of iterations but may affect precision. Therefore, a balance between accuracy and efficiency must be maintained based on the requirements of the problem.

Conclusion

In conclusion, the Bisection Method, when combined with Python programming and visualization, offers an effective approach for solving nonlinear equations. Its simplicity, reliability, and ease of implementation make it an essential topic in numerical methods. This study emphasizes the importance of computational tools in modern mathematics education and provides a practical framework for exploring numerical techniques in a more interactive and engaging manner.

References

[1.] Etesami, R., Madadi, M., Mashinchi, M., & Ganjoei, R. A. (2021). *A new method for rooting nonlinear equations based on the bisection method*. **MethodsX, 8**, 101502.https://doi.org/10.1016/j.mex.2021.101502

[2.] Romadani, M. N. H. (2025). *From theory to code: Transforming classical root-finding methods into efficient Python implementations*. ICCK Journal of Applied Mathematics.

[3.] Gulshan, G., Budak, H., Hussain, R., & Sadiq, A. (2023). *Generalization of the bisection method and its applications in nonlinear equations*. Advances in Continuous and Discrete Models.

[4.] García, G. (2025). *A bisection-type method based on α-dense curves*. Open Journal of Mathematical Sciences.
[5.] Islam, M. M., Islam, M. S., & Saiduzzaman, M. (2022). *An advanced approach of Regula-Falsi method exploiting the bisection method*. Journal of Interdisciplinary Mathematics.
[6.] Zhu, K., Shi, G., Liu, J., & Shi, J. (2022). *Fast high-precision bisection feedback search algorithm and its application*. Journal of Marine Science and Engineering.
[7.] Widyanto, W., Abiyyu, R. F., & Rachman, A. (2022). *Design of numerical solution computation using the bisection method in Python*. KERNEL Journal of Informatics.