**Implementation of the Bisection Method in C**

**1. Introduction**

The **Bisection Method** is a numerical technique used to find the root of a function within a specified interval. It works by repeatedly halving the interval and selecting the subinterval where the function changes sign. This ensures that the method converges to the root if the function is continuous and the given interval contains a root.

This document explains the implementation of the **Bisection Method in C**, including a breakdown of the logic, input requirements, and code functionality.

**2. Algorithm of the Bisection Method**

The Bisection Method follows these steps:

1. **Choose an interval** [a,b][a, b] such that f(a)×f(b)<0f(a) \times f(b) < 0, indicating a root exists in the interval.
2. **Calculate the midpoint** c=a+b2c = \frac{a + b}{2}.
3. **Evaluate the function** at cc:
   * If f(c)=0f(c) = 0, then cc is the root.
   * If f(a)×f(c)<0f(a) \times f(c) < 0, update b=cb = c (root is in the left half).
   * Else, update a=ca = c (root is in the right half).
4. **Repeat steps 2-3** until the desired error tolerance is met.

**3. C Program Implementation**

Below is the C implementation of the **Bisection Method** with user input for polynomial coefficients, interval limits, and error tolerance.

**Code Implementation**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <stdbool.h>

// Function to evaluate the polynomial at a given x-value

double function(double number, double equation[], int size) {

double result = 0.0;

for (int i = 0; i < size; i++) {

result += equation[i] \* pow(number, size - i - 1);

}

return result;

}

int main() {

double targetErrorRatio;

printf("Enter target error ratio: ");

scanf("%lf", &targetErrorRatio);

double topLimit, bottomLimit;

printf("\nEnter top limit: ");

scanf("%lf", &topLimit);

printf("\nEnter bottom limit: ");

scanf("%lf", &bottomLimit);

int numberOfTerms;

printf("\nEnter number of terms: ");

scanf("%d", &numberOfTerms);

double equation[numberOfTerms];

for (int i = 0; i < numberOfTerms; i++) {

printf("Enter term %d: ", i + 1);

scanf("%lf", &equation[i]);

}

// Validate if a root exists in the given interval

if (function(topLimit, equation, numberOfTerms) \* function(bottomLimit, equation, numberOfTerms) >= 0) {

printf("Invalid interval! No root in [%lf, %lf]\n", bottomLimit, topLimit);

return 1;

}

double errorRatio = 1.0;

int count = 0;

bool flag = true;

double midpoint;

// Implementing the Bisection Method

do {

count++;

errorRatio = (topLimit - bottomLimit) / pow(2.0, count);

midpoint = (topLimit + bottomLimit) / 2.0;

double f\_mid = function(midpoint, equation, numberOfTerms);

double f\_bottom = function(bottomLimit, equation, numberOfTerms);

if (f\_mid == 0.0) {

bottomLimit = midpoint;

break;

} else if (f\_mid \* f\_bottom < 0) {

topLimit = midpoint;

} else {

bottomLimit = midpoint;

}

} while (errorRatio > targetErrorRatio);

printf("%lf is root of equation.\n", bottomLimit);

return 0;

}

**4. Explanation of the Code**

**Function to Evaluate the Polynomial**

* function() takes three arguments:
  + number: The value at which the polynomial is evaluated.
  + equation[]: Array of polynomial coefficients.
  + size: Number of terms in the polynomial.
* It computes the polynomial value using a loop and returns the result.

**Main Program Workflow**

1. **User Input:**
   * Reads the target error ratio.
   * Reads the interval [a, b].
   * Reads the number of terms and coefficients of the polynomial.
2. **Validates the Interval:**
   * Ensures f(a) \* f(b) < 0, confirming the presence of a root.
3. **Applies the Bisection Method:**
   * Iteratively refines the interval until the error ratio is within the target range.
   * Stops when the root is found, or error is minimized.
4. **Outputs the Approximate Root.**

**5. Sample Input & Output**

**Example Run**

**Input:**

Enter target error ratio: 0.001

Enter top limit: 2

Enter bottom limit: 0

Enter number of terms: 3

Enter term 1: 2

Enter term 2: -4

Enter term 3: -1

**Output:**

0.736816 is root of equation.

**6. Advantages & Limitations**

**Advantages:**

✅ **Guaranteed Convergence:** Always converges if the function is continuous. ✅ **Simple & Easy to Implement:** Does not require derivatives. ✅ **Controlled Accuracy:** Error tolerance can be adjusted.

**Limitations:**

❌ **Slow Convergence:** Takes many iterations for high precision. ❌ **Requires an Interval with Opposite Signs:** Cannot find roots where the function does not cross the x-axis.

**7. Conclusion**

The **Bisection Method** is a powerful and reliable numerical technique for root-finding. This C implementation allows users to input their own polynomial equations and compute roots within a given error tolerance. Despite its slow convergence, it is widely used in numerical analysis due to its simplicity and robustness.

This project can be further extended to:

* Implement **other root-finding methods** like Newton-Raphson.
* Improve efficiency using **adaptive interval selection**.