Numerical Methods Lab

Lab report-1

Bisection Method

Submitted By:

Name:Md.Mursalin
Roll:16
Reg:202004018

Submitted To:

A F M Shahab Uddin
Assistant Professor
Department of Computer Science and Engineering,
Jashore University of Science and Technology,

Department of Computer Science and Engineering

Netrokona University Netrokona, Bangladesh

Lab 1 - Report and Code for Bisection Method

Objective:

To find the root of a given nonlinear equation using the **Bisection Method** with a predefined accuracy and iteration limit.

Theory:

The **Bisection Method** is a numerical technique to find the root of a continuous function f(x)f(x)f(x). If a function changes sign over an interval [a,b], i.e., $f(a) \cdot f(b) < 0$ then there exists at least one root in that interval.
The method works by repeatedly bisecting the interval and selecting the subinterval in which the function changes sign.

The new midpoint is calculated as:

mid=(a+b)/2;

if f(mid) s sufficiently close to 0 (within given error tolerance), we treat it as the root.

Given Function:

f(x)=x3−4x−9f(x) = x^3 - 4x - 9f(x)=x3−4x−9

Algorithm Steps:

1. Input initial guesses a, b, error tolerance ϵ, and maximum iteration count.
2. Check if $f(a) \cdot f(b) < 0$. If not, root is not guaranteed in the interval.

Calculate midpoint mid=(a+b)/2;

3. If $|f(mid)| < \epsilon$, return mid_ as the root.
4. Update interval:
   - If f(a) ·f(mid)<0, set b=mid ,a=mid
   - Else, set a=mid
5. Repeat for the specified number of iterations or until error condition is met.

Source Code (C++):

```
#include <bits/stdc++.h>
using namespace std;
float f(float x)
{
    return x * x * x - 4 * x - 9;
}
void solve()
{

    float a, b, x, mid, error;
    int max_iter;

    cin >> a;

    cin >> b;

    cin >> error;
    cin >> max_iter;

    if (f(a) * f(b) >= 0)
    {
        cout << "no value found" << endl;
        return;
    }

    for (int i = 0; i < max_iter; ++i)
    {
        mid = (a + b) / 2;

        cout << "Iteration " << i << " f(a):" << a << " " << "f(b): " << b << ": mid = "
<< mid << "  " << "f(c):" << f(mid) << endl;
```

```
        if (fabs(f(mid)) < error || fabs(b - a) < error)
        {
            cout << "Root found after " << i << " iterations: " << mid << endl;
            return;
        }

        if (f(a) * f(mid) < 0)
            b = mid;
        else
            a = mid;
    }

    cout << "solution not found" << endl;

    return;
}

int main()
{

    solve();

    return 0;
}
```

Sample Input:

4
-7
0.0001
20

Sample output:

4
-7
0.0001
20
Iteration 0 f(a):4 f(b): -7: mid = -1.5 f(c):-6.375
Iteration 1 f(a):4 f(b): -1.5: mid = 1.25 f(c):-12.0469
Iteration 2 f(a):4 f(b): 1.25: mid = 2.625 f(c):-1.41211
Iteration 3 f(a):4 f(b): 2.625: mid = 3.3125 f(c):14.0969
Iteration 4 f(a):3.3125 f(b): 2.625: mid = 2.96875 f(c):5.29001
Iteration 5 f(a):2.96875 f(b): 2.625: mid = 2.79688 f(c):1.69108
Iteration 6 f(a):2.79688 f(b): 2.625: mid = 2.71094 f(c):0.0794234
Iteration 7 f(a):2.71094 f(b): 2.625: mid = 2.66797 f(c):-0.681121
Iteration 8 f(a):2.71094 f(b): 2.66797: mid = 2.68945 f(c):-0.304573
Iteration 9 f(a):2.71094 f(b): 2.68945: mid = 2.7002 f(c):-0.113509
Iteration 10 f(a):2.71094 f(b): 2.7002: mid = 2.70557 f(c):-0.0172772
Iteration 11 f(a):2.71094 f(b): 2.70557: mid = 2.70825 f(c):0.0310145
Iteration 12 f(a):2.70825 f(b): 2.70557: mid = 2.70691 f(c):0.00685404
Iteration 13 f(a):2.70691 f(b): 2.70557: mid = 2.70624 f(c):-0.00521523
Iteration 14 f(a):2.70691 f(b): 2.70624: mid = 2.70657 f(c):0.000818492
Iteration 15 f(a):2.70657 f(b): 2.70624: mid = 2.70641 f(c):-0.0021986
Iteration 16 f(a):2.70657 f(b): 2.70641: mid = 2.70649 f(c):-0.000690109
Iteration 17 f(a):2.70657 f(b): 2.70649: mid = 2.70653 f(c):6.41769e-005
Root found after 17 iterations: 2.70653

Conclusion:

The Bisection Method successfully found a root of the equation f(x)=x3−4x−9f(x) = x^3 - 4x - 9f(x)=x3−4x−9 within the specified interval and tolerance. This method is reliable for continuous functions where a sign change occurs.

Advantages:

- Simple and easy to implement.

- Guaranteed convergence if conditions are met.

Limitations:

- Slow convergence.
- Requires function to have opposite signs at the endpoints.