Title: Polynomial Root Finder with Bisection and False Position Methods

Theoretical Background:

1. Bisection Method: The Bisection method, also known as the binary search method, is a simple and robust numerical technique to find the root of a real-valued function. It is based on the Intermediate Value Theorem, which states that if a continuous function changes sign over an interval, it must have at least one root within that interval. The Bisection method works as follows:

Start with an interval [a, b] where the function changes sign (f(a) * f(b) < 0).

Calculate the midpoint c = (a + b) / 2.

Check the sign of f(c):

If f(c) ≈ 0 (within a predefined tolerance), c is the root.

If f(c) has the same sign as f(a), replace a with c, else replace b with c.

Repeat the process until the interval becomes sufficiently small or the function value is close to zero.

2. False Position Method (Regula Falsi): The False Position method is another numerical technique for finding the root of a real-valued function. It's based on linear interpolation. This method is also known as the linear interpolation method. The False Position method works as follows:

Start with an interval [a, b] where the function changes sign (f(a) * f(b) < 0).

Calculate the point c where the line connecting (a, f(a)) and (b, f(b)) intersects the x-axis.

Check the sign of f(c):

If f(c) ≈ 0 (within a predefined tolerance), c is the root.

If f(c) has the same sign as f(a), replace a with c, else replace b with c.

Repeat the process until the interval becomes sufficiently small or the function value is close to zero.

Program:

```
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;
double arr[100];
long double polynom(long double x, int d)
{
    long double sum = 0;
    for (int i = d; i >= 0; i--)
        sum += (arr[i] * pow(x, i));
    return sum;
}
void method(int de, long double a, long double b, bool h)
{
    int i = 1;
    long double temp1 = polynom(a, de), temp2 = polynom(b, de), mid = (h ==
true) ? ((a * temp2) - (b * temp1)) / (temp2 - temp1) : ((a + b) / 2);
    cout << "step  "<< "a   "<< "b   "<< "Xr   "<< "f(xr)   "<< "error" <<
endl;
    cout << "0"<< "   " << a << "   " << b << "   " << mid << "   " <<
polynom(mid, de) << "   " << mid << endl;
    if (temp1 * temp2 < 0)
    {
        while (i > 0)
        {
            long double prev = 0;
            long double chk = polynom(mid, de);
            prev = mid;
            if (temp1 * chk < 0)
                b = mid;
            else if (temp1 * chk == 0)
                break;
            else if (temp1 * chk > 0)
                a = mid;
            temp1 = polynom(a, de);
            temp2 = polynom(b, de);
            mid = (h == true) ? ((a * temp2) - (b * temp1)) / (temp2 - temp1)
: ((a + b) / 2);
            long double err = (mid - prev);
            cout << i << "   " << a << "   " << b << "   " << mid << "   " <<
chk << "   " << abs(err) << endl;
            if (abs(err) < 0.0001)
                break;
```

```
            i++;
        }
    }
    cout << "ANSWER : " << setprecision(5) << mid << endl;
}
int main()
{
    long double a, b;
    int deg;
    bool h;
    cout << "degree, a, b value : ";
    cin >> deg >> a >> b;
    cout << "What method do you want: (0 for bisection 1 for false position)
";
    cin >> h;
    cout << "Give the " << deg + 1 << " Coefficients: ";
    for (int i = deg; i >= 0; i--)
        cin >> arr[i];
    method(deg, a, b, h);
}
```

Input & Output: Bisection

```
C:\Users\Tonmo\OneDrive\Documents\Work\RUET_CSE>cd  C:\Users\Tonmo\O
ork\RUET_CSE\CSE2204\Lab1\q1\"q1
degree, a, b value : 3 2 3
What method do you want: (0 for bisection 1 for false position) 0
Give the 4 Coefficients: 1 0 -2 -5
step  a    b    Xr    f(xr)    error
0   2   3   2.5   5.625   2.5
1   2   2.5   2.25   5.625   0.25
2   2   2.25   2.125   1.89062   0.125
3   2   2.125   2.0625   0.345703   0.0625
4   2.0625   2.125   2.09375   -0.351318   0.03125
5   2.09375   2.125   2.10938   -0.00894165   0.015625
6   2.09375   2.10938   2.10156   0.166836   0.0078125
7   2.09375   2.10156   2.09766   0.0785623   0.00390625
8   2.09375   2.09766   2.0957   0.0347143   0.00195312
9   2.09375   2.0957   2.09473   0.0128623   0.000976562
10   2.09375   2.09473   2.09424   0.00195435   0.000488281
11   2.09424   2.09473   2.09448   -0.00349515   0.000244141
12   2.09448   2.09473   2.0946   -0.000770775   0.00012207
13   2.09448   2.0946   2.09454   0.000591693   6.10352e-05
ANSWER : 2.0945
```

False Position:

```
C:\Users\Tonmo\OneDrive\Documents\work\RUET_CSE>cd "C:\Users\Tonmo\OneD
ork\RUET_CSE\CSE2204\Lab1\q1\"q1
degree, a, b value : 3 2 3
What method do you want: (0 for bisection 1 for false position) 1
Give the 4 Coefficients: 1 0 -2 -5
step  a   b   Xr   f(xr)   error
0   2   3   2.05882   -0.3908   2.05882
1   2.05882   3   2.08126   -0.3908   0.0224401
2   2.08126   3   2.08964   -0.147204   0.00837555
3   2.08964   3   2.09274   -0.0546765   0.00310036
4   2.09274   3   2.09388   -0.0202029   0.00114413
5   2.09388   3   2.09431   -0.00745051   0.000421743
6   2.09431   3   2.09446   -0.00274567   0.000155395
7   2.09446   3   2.09452   -0.00101157   5.72476e-05
ANSWER : 2.0945

c:\Users\Tonmo\OneDrive\Documents\work\RUET_CSE\CSE2204\Lab1\q1>
```

Discussion: The Bisection and False Position methods are essential tools in numerical analysis for finding the roots of single-variable polynomials. The Bisection method relies on the concept of interval halving and is guaranteed to converge to a root if the initial interval contains a sign change in the function. The False Position method, on the other hand, uses linear interpolation to iteratively narrow down the interval containing the root. Both methods are relatively simple to implement and can be effective even when dealing with complex or non-differentiable functions. Choosing between them often depends on the specific characteristics of the function and the desired convergence speed, as the Bisection method provides slower but more reliable convergence, while the False Position method can be faster but may encounter convergence issues with certain functions.