Title: Finding root of x^3-2x-5 by Iteration and Newton-Raphson Method.

Theoretical Background:

Iteration Method:

The Iteration Method, also known as the Fixed-Point Iteration Method, is a numerical technique for finding the roots of an equation of the form f(x) = 0. It is based on the idea of rearranging the equation into the form x = $\phi(x_n)$, where g(x) is a function that is easier to work with.

The general iterative formula for the Iteration Method is given by:

$x_{n+1} = \phi(x_n)$

Here, x_{n+1} is the next approximation, and x_n is the current approximation. The process is repeated until the sequence x_n converges to the root.

For convergence to occur, the iteration function $\phi(x_n)$ must satisfy the conditions:

1. $\phi(x_n)$ must be continuous at an interval containing the root.

2. $|\phi'(x_n)| < 1$ for all x in the interval.

Newton-Raphson Method:

The Newton-Raphson method is another iterative technique for finding the roots of a real-valued function f(x) = 0. It is based on linear approximation of the function around an initial guess.

The iteration formula for the Newton-Raphson method is given by:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Here, f'(x) is the derivative of f(x), and x_{n+1} is the next approximation.

The conditions for the Newton-Raphson method to converge include:

1. The initial guess should be close to the actual root.

2. $f'(x_n)$ should not be close to zero to avoid division by a very small number.

3. The function f(x) and its derivative f'(x) should be continuous.

Program:

```
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;
double arr[100];
long double polynom2(long double x, int d, int h)
{
    long double sum = 0;

    if (h == 0 || h == 2)
    {
        for (int i = (d - 2); i >= -1; i--)
            sum += (arr[i + 1] * pow(x, i));
    }
    else if (h == 1)
    {
        for (int i = d; i >= 0; i--)
            sum += (arr[i] * pow(x, i));
    }
    if (h == 0)
        return sqrt(-sum);
    else if (h == 1)
        return (x - (sum / ((3 * pow(x, 2)) - 2)));
    else if (h == 2)
        return round(abs(-2.5 * pow(x, -2) * pow(-sum, -0.5)) * 10) / 10;
    return 0;
}
void method(int de, long double a, long double b, int h)
{
    int i = 1;
    long double prev = a, chk, err, k = polynom2(b, de, 2);
    long double e = (h == 0) ? ((1 - k) / k) * 0.0001 : 0.0001;
    (h == 0) ? cout << "Iteration Method" << endl : cout << "Newton Raphson 
Method" << endl;
    cout << "step  "
         << "Xr    "
         << "error" << endl;
    while (i > 0)
    {
        chk = polynom2(prev, de, h);
        err = (chk - prev);
        prev = chk;
        cout << i << "    " << chk << "    " << abs(err) << endl;
```

```cpp
        if (abs(err) < e)
            break;
        i++;
    }

    cout << "ANSWER : " << prev << endl;
}
int main()
{
    long double a, b;
    int deg;
    int h;
    cout << "degree, a, b value : ";
    cin >> deg >> a >> b;
    cout << "What method do you want: (0 for Iteration 1 for Newton Raphson) ";
    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:

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Tonmo\OneDrive\Documents\RUET_CSE> cd "c:\Users\Tonmo\OneDrive\Documents\RUET_CSE\CSE2204\Lab2\" ;
sk }
degree, a, b value : 3 2 3
What method do you want: (0 for Iteration 1 for Newton Raphson) 0
Give the 4 Coefficients: 1 0 -2 -5
Iteration Method
step  Xr    error
1    2.12132    0.12132
2    2.08735    0.0339721
3    2.09652    0.00916883
4    2.09402    0.00249989
5    2.0947   0.000679723
ANSWER : 2.0947
PS C:\Users\Tonmo\OneDrive\Documents\RUET_CSE\CSE2204\Lab2> cd "c:\Users\Tonmo\OneDrive\Documents\RUET_CSE\CSE2
f ($?) { .\task }
degree, a, b value : 3 2 3
What method do you want: (0 for Iteration 1 for Newton Raphson) 1
Give the 4 Coefficients: 1 0 -2 -5
Newton Raphson Method
step  Xr    error
1    2.1    0.1
2    2.09457    0.00543188
3    2.09455    1.66394e-05
ANSWER : 2.0946
PS C:\Users\Tonmo\OneDrive\Documents\RUET_CSE\CSE2204\Lab2>
```

Comparisons:

Convergence Rate:

- The Newton-Raphson method typically converges faster than the Iteration Method.
- Newton-Raphson has quadratic convergence (approximately doubles the number of correct digits with each iteration) when close to the root.

Derivative Requirement:

- Newton-Raphson requires the derivative of the function, which may not always be readily available.
- Iteration Method only requires the function itself.

Ease of Implementation:

- The Iteration Method is generally easier to implement since it only requires the function $\phi(x_n)$ without the need for derivatives.