Lab 02 - Solution of Non-linear Equation Using NR method and Secant Method

Objective

To understand and implement the NR and Secant methods for finding roots of non-linear equations:

1. Implement the **NR method** to approximate a root of a continuous function.
2. Implement the **Secant method** to approximate a root of the same function.
3. Compare the convergence behaviour and accuracy of both methods.

Theory

Newton-Raphson (NR) Method

The Newton-Raphson method is an efficient iterative technique for finding roots of real-valued functions. It uses the idea of linear approximation, where the tangent to the curve at a current guess is used to estimate the next, closer approximation to the root.

Given a function $f(x)$ and its derivative $f'(x)$, the next approximation x_{n+1} is computed as:

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

Advantages:

- Rapid (quadratic) convergence near the root if the initial guess is close.
- Requires only one initial guess.

Limitations:

- Needs the derivative $f'(x)$, which may not always be easy to compute.
- May fail or diverge if $f'(x)$ is zero or the initial guess is far from the root.
- Not guaranteed to converge for all functions.

The NR method is widely used due to its speed and simplicity when the function and its derivative are well-behaved.

Secant Method

The Secant method is a root-finding algorithm that, like the NR method, uses linear approximation but does not require the explicit computation of the derivative. Instead, it approximates the derivative using two recent points.

Given two initial guesses x_{n-1} and x_n, the next approximation x_{n+1} is:

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

Advantages:

- Does not require the analytical derivative of the function.
- Typically converges faster than the Bisection and False Position methods.

Limitations:

- Requires two initial guesses.

- Convergence is not guaranteed and may be slower than NR if the initial guesses are not close to the root.
- May fail if $f(x_n) = f(x_{n-1})$.

The Secant method is especially useful when the derivative is difficult to compute or unavailable.

Algorithm

NR method

1. Define $f(x)$ and $f'(x)$.
2. Input initial guess x_0.
3. Check if $f'(x_0) = 0$. If yes, repeat step 2.
4. Compute $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$.
5. If $|f(x_1)| <$ tolerance, stop; x_1 is the root.
6. Set $x_0 = x_1$ and repeat steps 3–5 until convergence or maximum iterations reached.

Secant Method

1. Define $f(x)$.
2. Input two initial guesses x_0 and x_1.
3. Compute $x_2 = x_1 - f(x_1) \cdot \frac{x_1 - x_0}{f(x_1) - f(x_0)}$.
4. If $|f(x_2)| <$ tolerance, stop; x_2 is the root.
5. Set $x_0 = x_1$, $x_1 = x_2$ and repeat steps 3–4 until convergence or maximum iterations reached.

Source Code

NR method

```c
#include <stdio.h>
#include <math.h>

#define TOLERANCE 0.001
#define MAX_ITERATIONS 1000

// Calculatet the value of the function
double f(double x)
{
    return x * x - x - 1; // Example function: f(x) = x^2 - x - 1
}

// Calculate the derivative of the function
double f_prime(double x)
{
    return 2 * x - 1; // Derivative of the function: f'(x) = 2x - 1
}

// Newton-Raphson method
double newton_raphson(double initial_guess)
{
    double x0 = initial_guess;
    double x1;
    int iterations = 0;

    // Print the header for the table
    printf("\nNewton-Raphson Method\n");
    printf("Iteration\t x\t\t f(x)\t\t f'(x)\n");
    printf("-----------------------------------------------------------\n");

    while (iterations < MAX_ITERATIONS)
    {
        double fx0 = f(x0);
        double fpx0 = f_prime(x0);

        if (fabs(fx0) < TOLERANCE)
        {
            return x0; // Root found
        }

        if (fabs(fpx0) < TOLERANCE)
        {
            printf("Derivative is too small. No solution found.\n");
            return NAN; // Derivative is too small, no solution found
        }

        x1 = x0 - fx0 / fpx0; // Update the guess

        x0 = x1; // Update the guess for the next iteration
        iterations++;

        // Print the current details in table format
        printf("%d \t\t%.6lf, \t%.6lf, \t%.6lf\n", iterations, x0, fx0, fpx0);
    }

    printf("Maximum iterations reached. No solution found.\n");
    return NAN; // Maximum iterations reached, no solution found
}

// Driver method
int main(int argc, char const *argv[])
{
    double initial_guess;

    // Get the initial guess from the user - Ensure derivative is not zero
    do
    {
        printf("Enter an initial guess: ");
        scanf("%lf", &initial_guess);

        // Check if the derivative at the initial guess is zero
        if (f_prime(initial_guess) == 0)
        {
            printf("The derivative at the initial guess is zero. Please enter a different guess.\n");
            continue; // Ask for a new guess
        }
        else
        {
            printf("Initial guess accepted: %lf\n", initial_guess);
            break; // Valid initial guess
        }
    } while (1);

    // Call the Newton-Raphson method
    double root = newton_raphson(initial_guess);

    // Print the footer of the table
    printf("-----------------------------------------------------------\n");
    if (!isnan(root))
    {
        printf("Root found: %lf\n", root);
    }
    else
    {
        printf("No root found.\n");
    }

    return 0;
}
```

Output:

```
Enter an initial guess: 0.5
The derivative at the initial guess is zero. Please enter a different guess.
Enter an initial guess: 0.1
Initial guess accepted: 0.100000

Newton-Raphson Method
Iteration        x                f(x)            f'(x)
-----------------------------------------------------------
1               -1.262500,      -1.090000,      -0.800000
2               -0.735860,      1.856406,       -3.525000
3               -0.623651,      0.277350,       -2.471720
4               -0.618048,      0.012591,       -2.247301
-----------------------------------------------------------
Root found: -0.618048
```

Secant Method

```c
#include <stdio.h>
#include <math.h>

#define TOLERANCE 0.001
#define MAX_ITERATIONS 1000

// Calculatet the value of the function
double f(double x)
{
    return x * x - x - 1; // Example function: f(x) = x^2 - x - 1
}

// Secant method
double secant(double a, double b)
{
    double x0 = a;
    double x1 = b;
    double x2;
    int iterations = 0;

    // Print the header for the table
    printf("\nSecant Method\n");
    printf("Iteration\t x0\t\t x1\t\t f(x0)\t\t f(x1)\t\t x2\t\t f(x2)\n");
    printf("-----------------------------------------------------------------------------------------------------------\n");

    while (iterations < MAX_ITERATIONS)
    {
        double fx0 = f(x0);
        double fx1 = f(x1);

        if (fabs(fx1) < TOLERANCE)
        {
            return x1; // Root found
        }

        x2 = (x0 * fx1 - x1 * fx0) / (fx1 - fx0); // Update the guess

        x0 = x1; // Update the guess for the next iteration
        x1 = x2; // Update the guess for the next iteration
        iterations++;

        // Print the current details in table format
        printf("%d \t\t%.6lf\t%.6lf\t%.6lf\t%.6lf\t%.6lf\t%.6lf\n", iterations, x0, x1, fx0, fx1, x2, f(x2));
    }

    printf("Maximum iterations reached. No solution found.\n");
    return NAN; // Maximum iterations reached, no solution found
}

// Driver method
int main(int argc, char const *argv[])
{
    double a, b;

    // Get the initial guess from the user - Ensure derivative is not zero
    do
    {
        printf("Enter initial guesses (a b): ");
        scanf("%lf %lf", &a, &b);
        // Check if
        if (f(a) == f(b))
        {
            printf("Invalid initial guesses. Denominator can't be zero.\n");
            continue; // Ask for a new guess
        }
        else {
            printf("Initial guesses accepted: %lf, %lf\n", a, b);
            break; // Valid initial guess
        }
    } while (1);

    // Call the Secant method
    double root = secant(a, b);

    // Print the footer of the table
    printf("-------------------------------------------------------------------------------------------------------------\n");
    if (!isnan(root))
    {
        printf("Root found: %lf\n", root);
    }
    else
    {
        printf("No root found.\n");
    }

    return 0;
}
```

Output:

```
Enter initial guesses (a b): 5 5
Invalid initial guesses. Denominator can't be zero.
Enter initial guesses (a b): 1 5
Initial guesses accepted: 1.000000, 5.000000

Secant Method
Iteration        x0               x1               f(x0)            f(x1)            x2               f(x2)
-------------------------------------------------------------------------------------------------------------
1               5.000000         1.200000         -1.000000        19.000000        1.200000         -0.760000
2               1.200000         1.346154         19.000000        -0.760000        1.346154         -0.534024
3               1.346154         1.691542         -0.760000        -0.534024        1.691542         0.169773
4               1.691542         1.608226         -0.534024        0.169773         1.608226         -0.021835
5               1.608226         1.617720         0.169773         -0.021835        1.617720         -0.000701
-------------------------------------------------------------------------------------------------------------
Root found: 1.617720
```

Discussion

- **Convergence Speed:** The Newton-Raphson (NR) method converges faster than the Secant method when the initial guess is close to the root, due to its quadratic convergence. The Secant method, while generally slower (superlinear convergence), still outperforms methods like Bisection.
- **Reliability:** The NR method can fail if the derivative is zero or if the initial guess is far from the root, potentially leading to divergence. The Secant method, not requiring the derivative, is more robust in cases where the derivative is difficult to compute, but may also fail if the two initial guesses are not chosen well or if $f(x_n) = f(x_{n-1})$.

- **Accuracy:** Both methods achieved similar accuracy, reaching the specified tolerance of $|f(x)| < 0.001$ for the given function $f(x) = x^2 - x - 1$.
- **Computational Cost:** The NR method requires computation of both the function and its derivative at each iteration, which can be computationally expensive if the derivative is complex. The Secant method only requires function evaluations, making it more efficient when derivatives are hard to compute.

Conclusion

Both the Newton-Raphson and Secant methods are effective for finding roots of non-linear equations. The NR method offers faster convergence when the derivative is available and the initial guess is close to the root, but it requires the computation of the derivative and may fail if the derivative is zero. The Secant method, while slightly slower, is advantageous when the derivative is difficult or impossible to compute, requiring only function evaluations. For the example problem, both methods successfully found the root within the desired tolerance, demonstrating their practical utility in numerical analysis. The choice between the two depends on the nature of the function and the available information.

Submitted By: Kushal Prasad Joshi
Roll No: 230345
Faculty: Science and Technology
Program: BE Computer
Group: B