Solving Nonlinear Equations using
Newton-Raphson and Bisection Methods

Ancentus Makau
Jomo Kenyatta University of Agriculture and Technology
Registration Number: SCT211-0469/2021

1 Introduction

In this report, we will solve the following nonlinear equation:

$$f(x) = x^5 + x^3 - 2x - 5 = 0 \tag{1}$$

We will use the Newton-Raphson method and the bisection method to find the root of the equation.

2 Newton-Raphson Method

The Newton-Raphson method is an iterative method for finding the root of a function. The method starts with an initial guess x_0 and uses the derivative of the function $f'(x)$ to improve the guess. The iteration formula is:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{2}$$

To use the Newton-Raphson method to solve Equation (1), we need to find the derivative of the function $f(x)$:

$$f'(x) = 5x^4 + 3x^2 - 2 \tag{3}$$

We will start with an initial guess of $x_0 = 2.0$. The Python program to implement the Newton-Raphson method is shown below:

```
def newton_raphson(x0, tol=1e-6, max_iterations=100):
    """
    This function implements the Newton-Raphson method to solve the nonlinear
    x0: initial guess
    tol: tolerance for convergence
    max_iterations: maximum number of iterations to perform
    """
    x = x0
```

```
    for i in range(max_iterations):
        fx = f(x)
        dfx = 5*x**4 + 3*x**2 - 2
        if abs(fx) < tol:
            return x
        x = x - fx / dfx
    return None

x0 = 2.0
start_time = time.time()
root = newton_raphson(x0)
elapsed_time = time.time() - start_time
if root is not None:
    print(f"The root found by Newton-Raphson is: {root:.6f}")
    print(f"Time taken by Newton-Raphson: {elapsed_time:.6f} seconds")
else:
    print("Newton-Raphson failed to converge.")
```

Running the program gives the following output:
The root found by Newton-Raphson is: 1.385768
Time taken by Newton-Raphson: 0.000028 seconds

3 Bisection Method

The bisection method is another iterative method for finding the root of a function. The method works by repeatedly dividing an interval in half and selecting the subinterval that contains the root. The iteration formula is:

$$x_{n+1} = \frac{a_n + b_n}{2} \tag{4}$$

where a_n and b_n are the endpoints of the subinterval at iteration n. To use the bisection method to solve Equation (1), we need to find two initial points a_0 and b_0 such that $f(a_0)$ and $f(b_0)$ have opposite signs.

We can start with $a_0 = 1.0$ and $b_0 = 2.0$.

The Python program to implement the bisection method is shown below:

```
def bisection(a, b, tol=1e-6, max_iterations=100):
    """
    This function implements the bisection method to solve the nonlinear equa
    a: left endpoint of the initial interval
    b: right endpoint of the initial interval
    tol: tolerance for convergence
    max_iterations: maximum number of iterations to perform
    """
    fa = f(a)
    fb = f(b)
```

```
    if fa * fb > 0:
        return None
    for i in range(max_iterations):
        c = (a + b) / 2
        fc = f(c)
        if abs(fc) < tol:
            return c
        if fa * fc < 0:
            b = c
            fb = fc
        else:
            a = c
            fa = fc
    return None

a = 0.0
b = 3.0
start_time = time.time()
root = bisection(a, b)
elapsed_time = time.time() - start_time
if root is not None:
    print(f"The root found by bisection is: {root:.6f}")
    print(f"Time taken by bisection: {elapsed_time:.6f} seconds")
else:
    print("Bisection failed to converge.")
```

Running the program gives the following output:
The root found by bisection is: 1.385768
Time taken by bisection: 0.000022 seconds

4 Conclusion

In this report, we have solved the nonlinear equation $f(x) = x^5 + x^3 - 2x - 5 = 0$ using the Newton-Raphson method and the Bisection method. Both methods give the same answer. The bisection method is slightly faster than the Newton-Raphson method for this equation.