Newton-Raphson

The Newton-Raphson method, also known as Newton's method, is a powerful numerical technique used to find the roots of real-valued functions. This method iteratively refines an initial guess to converge toward the desired root.

Formula

: The Newton-Raphson formula for approximating the next iteration is:

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

where:

- (x_n) is the current approximation.
- (f(x_n)) is the value of the function at (x_n).
- (f′(x_n)) is the value of the first derivative of the function at (x_n).

Advantages of Newton-Raphson Method:

1. **Fast Convergence**:
    - The method converges quickly (if it converges) to the root. In most cases, you obtain the root in fewer steps.
2. **Single Initial Guess**:
    - Only one initial guess ((x_0)) is required.
3. **Simple Formulation**:
    - The formula is straightforward, making it easy to apply.
4. **Intuitive Derivation**:
    - The derivation is more intuitive, allowing better understanding of its behavior.

Disadvantages of Newton-Raphson Method:

1. **Dependence on Initial Guess**:
    - The method's success heavily relies on choosing a good initial guess. If the guess is far from the actual root, convergence may fail.
2. **Need for Derivative**:
    - Calculating the derivative (($f'(x)$)) is necessary. If the derivative is zero, the method breaks down.
3. **Poor Global Convergence Properties**:
    - The method may not converge globally for all functions.
4. **Possible Infinite Loops**:
    - Around critical points, thousands of iterations may occur, leading to slow convergence.

Applications of Newton-Raphson Method:

1. **Optimization Problems**:
    - Used to find local minima or maxima of functions.
2. **Solving Transcendental Equations**:
    - Finds solutions for equations involving transcendental functions.
3. **Obtaining Zeros of Special Functions**:
    - Useful for finding zeros of functions like Bessel functions.
4. **Numerical Verification of Nonlinear Equations**.

Example:

$$f(x) = x^3 - x - 1$$

```
from scipy.optimize import newton

f = lambda x: x**3 - x - 1
root = newton(f, 1.5)
print(root)
```