Numerical Methods in Python

This repository contains Python implementations of various Numerical Methods used in mathematical and engineering applications. These methods help solve interpolation problems, numerical integration, solving ODEs, solving systems of linear equations, root-finding techniques, and curve fitting.

1 Interpolation

Interpolation is a method to estimate values between known data points.

1.1 Lagrange's Interpolation

Formula:

$$P(x) = \sum_{i=0}^{n} y_i \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}$$

1.2 Newton's Forward Difference Interpolation

Formula:

$$P(x) = y_0 + \Delta y_0 \frac{(x - x_0)}{h} + \frac{\Delta^2 y_0}{2!} \frac{(x - x_0)(x - x_1)}{h^2} + \dots$$

1.3 Newton's Backward Difference Interpolation

Formula:

$$P(x) = y_n + \nabla y_n \frac{(x - x_n)}{h} + \frac{\nabla^2 y_n}{2!} \frac{(x - x_n)(x - x_{n-1})}{h^2} + \dots$$

1.4 Newton's Divided Difference Interpolation

Formula:

$$P(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \dots$$

2 Numerical Integration

Methods to approximate the integral of a function.

2.1 Trapezoidal Rule

Formula:

$$\int_a^b f(x)dx \approx \frac{h}{2}[f(a) + 2\sum f(x_i) + f(b)]$$

2.2 Simpson's 1/3 Rule

Formula:

$$\int_a^b f(x)dx \approx \frac{h}{3}[f(a) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \ldots + f(b)]$$

2.3 Simpson's 3/8 Rule

Formula:

$$\int_a^b f(x)dx \approx \frac{3h}{8}[f(a) + 3f(x_1) + 3f(x_2) + 2f(x_3) + 3f(x_4) + \ldots + f(b)]$$

3 Numerical Solution of ODEs

3.1 Euler's Method

Formula:

$$y_{n+1} = y_n + hf(x_n, y_n)$$

3.2 Modified Euler's Method

Formula:

$$y^* = y_n + hf(x_n, y_n)$$
$$y_{n+1} = y_n + \frac{h}{2}[f(x_n, y_n) + f(x_{n+1}, y^*)]$$

3.3 Taylor Series Method

Formula:

$$y_{n+1} = y_n + hf(x_n, y_n) + \frac{h^2}{2}f'(x_n, y_n)$$

3.4 Runge-Kutta 2nd Order (RK2)

Formula:

$$k_1 = hf(x_n, y_n)$$
$$k_2 = hf(x_n + h, y_n + k_1)$$
$$y_{n+1} = y_n + \frac{1}{2}(k_1 + k_2)$$

3.5 Runge-Kutta 4th Order (RK4)

Formula:

$$k_1 = hf(x_n, y_n)$$
$$k_2 = hf(x_n + \frac{h}{2}, y_n + \frac{k_1}{2})$$
$$k_3 = hf(x_n + \frac{h}{2}, y_n + \frac{k_2}{2})$$
$$k_4 = hf(x_n + h, y_n + k_3)$$
$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

3.6 Runge-Kutta 4th Order for 2nd Order ODEs

Solving second-order ODEs of the form:

$$\frac{dy}{dx} = f(x, y, z), \quad \frac{dz}{dx} = g(x, y, z)$$

3.7 Milne's Predictor-Corrector Method

3.7.1 Predictor:

$$y_{n+1} = y_{n-3} + \frac{4h}{3}(2f_{n-1} - f_{n-2} + 2f_n)$$

Corrector:

$$y_{n+1} = y_{n-1} + \frac{h}{3}(f_{n-1} + 4f_n + f_{n+1})$$

3.8 Adams-Bashforth Method

A multi-step method using past function evaluations.

4 Numerical Solution of System of Linear Equations

4.1 Checking for Diagonal Dominance

A matrix is diagonally dominant if:

$$|a_{ii}| \geq \sum_{j \neq i} |a_{ij}| \quad \forall i$$

4.2 Gauss Elimination Method

A systematic method to reduce a system of equations to an upper triangular form.

4.3 Gauss-Seidel Method

Iterative method using:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j<i} a_{ij} x_j^{(k+1)} - \sum_{j>i} a_{ij} x_j^{(k)} \right)$$

4.4 Jacobi Iteration Method

Iterative method using:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right)$$

5 Methods of Root Finding

5.1 Bisection Method

If $f(a)f(b) < 0$, the root lies in the interval (a, b) and is given by:

$$c = \frac{a + b}{2}$$

5.2 Secant Method

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

5.3 Regula Falsi Method

$$x_{n+1} = \frac{af(b) - bf(a)}{f(b) - f(a)}$$

5.4 Newton-Raphson Method

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

5.5 Newton-Raphson for Multiple Roots

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

6 Curve Fitting

6.1 Straight Line Fit

$$y = ax + b$$

6.2 Parabolic Fit

$$y = ax^2 + bx + c$$

6.3 Exponential Fit

$$y = ae^{bx}$$