RK4-ODEsolverX

RK4-ODEsolverX is a Java-based project that allows users to solve ordinary differential equations (ODEs) using numerical methods like Runge-Kutta 4th Order (RK4), Euler's method, and the Adams-Moulton method.

Tech Stack

- Java

- Maven

- exp4j library

- (Frontend in progress)

Implemented Numerical Methods

- Euler's Method

- Runge-Kutta 4th Order (RK4)

- Adams-Moulton Method

Features

- Easy usability

- User-friendly design

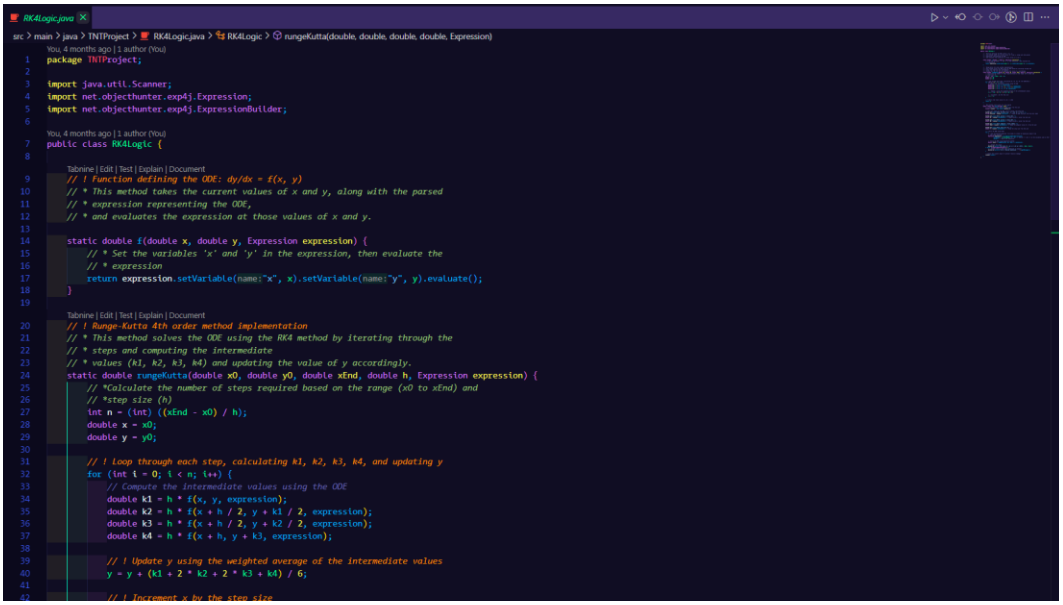
- Expression-based ODE input

Libraries Used

- exp4j

- Maven for dependency management

Code Snippet Screenshot


RK4Logic.java
src > main > java > TNTProject > RK4Logic.java > RK4Logic > rungeKutta(double, double, double, double, Expression)
You, 4 months ago | 1 author (You)
Tabnine | Edit | Test | Explain | Document

```java
package TNTProject;

import java.util.Scanner;
import net.objecthunter.exp4j.Expression;
import net.objecthunter.exp4j.ExpressionBuilder;

public class RK4Logic {

    // ! Function defining the ODE: dy/dx = f(x, y)
    // * This method takes the current values of x and y, along with the parsed
    // * expression representing the ODE,
    // * and evaluates the expression at those values of x and y.

    static double f(double x, double y, Expression expression) {
        // * Set the variables 'x' and 'y' in the expression, then evaluate the
        // * expression
        return expression.setVariable(name:"x", x).setVariable(name:"y", y).evaluate();
    }

    // ! Runge-Kutta 4th order method implementation
    // * This method solves the ODE using the RK4 method by iterating through the
    // * steps and computing the intermediate
    // * values (k1, k2, k3, k4) and updating the value of y accordingly.
    static double rungeKutta(double x0, double y0, double xEnd, double h, Expression expression) {
        // *Calculate the number of steps required based on the range (x0 to xEnd) and
        // *step size (h)
        int n = (int) ((xEnd - x0) / h);
        double x = x0;
        double y = y0;

        // ! Loop through each step, calculating k1, k2, k3, k4, and updating y
        for (int i = 0; i < n; i++) {
            // Compute the intermediate values using the ODE
            double k1 = h * f(x, y, expression);
            double k2 = h * f(x + h / 2, y + k1 / 2, expression);
            double k3 = h * f(x + h / 2, y + k2 / 2, expression);
            double k4 = h * f(x + h, y + k3, expression);

            // ! Update y using the weighted average of the intermediate values
            y = y + (k1 + 2 * k2 + 2 * k3 + k4) / 6;

            // ! Increment x by the step size
```

Check the Full Project at: https://github.com/ArjiJethin/RK4-ODEsolverX

-Made by Arji Jethin