# Report

*Grigorii Kostarev BS19-03*

## Exact solution

$$\begin{cases} y' = 3y - xy^{1/3} \\ y(1) = 2 \end{cases}$$

$$y' - 3y = xy^{1/3}$$

So, this is Bernoulli equation. Let's solve it. First of all we should divide both parts by

$$y^{2/3}$$

We get

$$y'y^{-1/3} - 3y^{2/3} = -x$$

Then, make substitution

$$z = y^{2/3} z' = 2/3y{-}1/3y'$$

We get

$$\frac{3}{2}z' - 3z = -x \tag{1}$$

Equation (1) is a first-order non-homogeneous linear ordinary differential equations. So, first of all we need to solve complementary equation,

$$\frac{3}{2}z' - 3z = 0$$

$$z' = 2z$$

$$\int \frac{dz}{z} = 2\int dx$$

$$e^{ln|z|} = e^{2x+C_1}$$

$$z = e^{2x}C_2$$

$$z' = 2e^{2x}C_2 + C_2'e^{2x}$$

Substitute to equation (1)

$$3e^{2x}C_2 + \frac{3}{2}C_2'e^{2x} - 3e^{2x}C_2 = -x$$

$$\frac{3}{2}C_2'e^{2x} = -x$$

$$C_2' = -\frac{2}{3}xe^{-2x}$$

$$C_2 = -\frac{2}{3}\int xe^{-2x}dx = \frac{2}{3}\cdot\frac{(2x+1)e^{-2x}}{4} + C_3$$

$$z = \frac{2x+1}{6} + e^{2x}C_3$$

$$z = \frac{x}{3} + \frac{1}{6} + e^{2x}C_3$$

Back substitution

$$y^{2/3} = \frac{x}{3} + \frac{1}{6} + e^{2x}C_3$$

$$y = (\frac{x}{3} + \frac{1}{6} + e^{2x}C_3)^{3/2}$$

So, let's find $C_3$

$$C_3 = \frac{y^{2/3} - \frac{x}{3} - \frac{1}{6}}{e^{2x}}$$

$$y(1) = 2$$

$$C_3 = (2^{2/3} - \frac{1}{2})e^{-2}$$

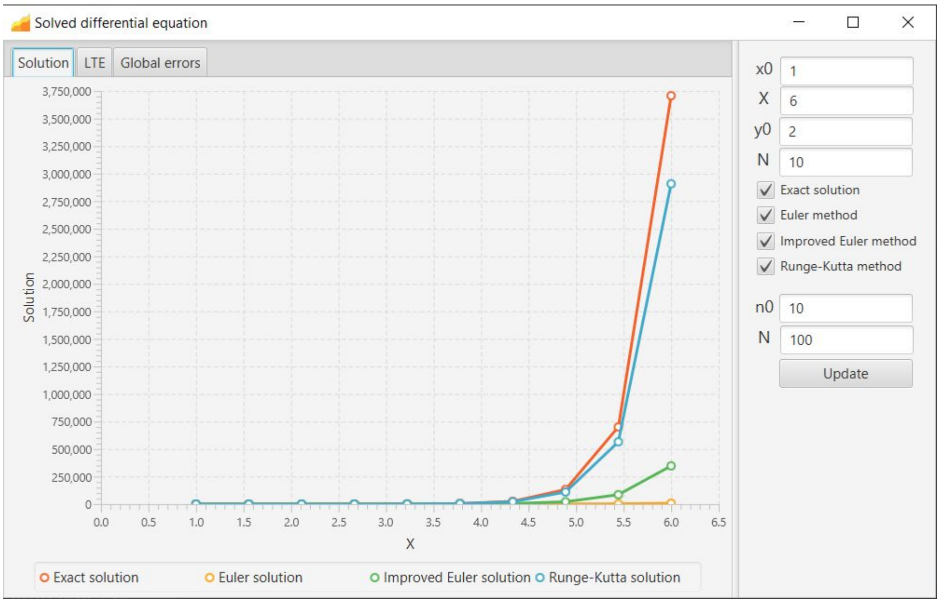
$$y = (\frac{x}{3} + \frac{1}{6} + e^{2(x-1)}(2^{2/3} - \frac{1}{2}))^{3/2}$$

**Answer**

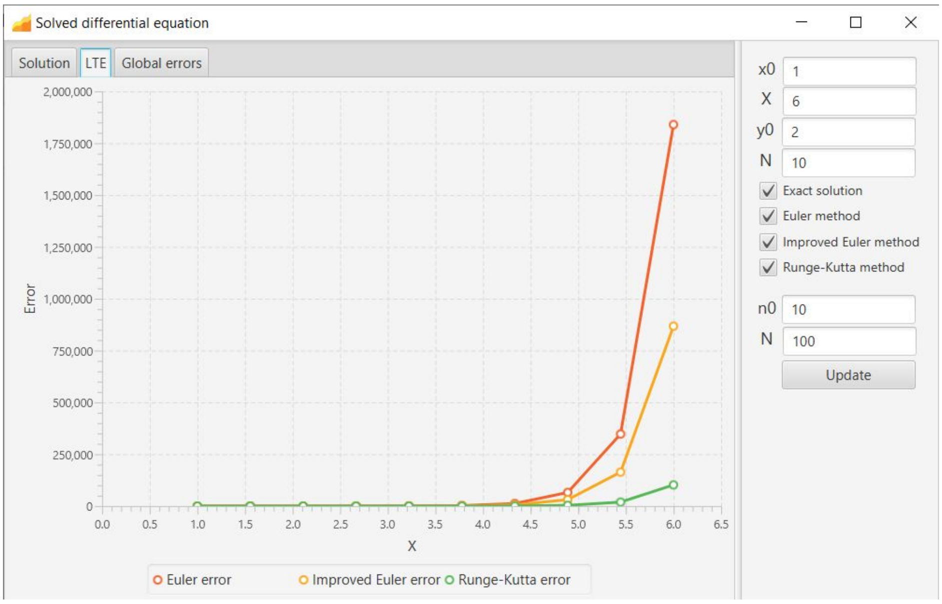
$$\boldsymbol{y = (\frac{x}{3} + \frac{1}{6} + e^{2(x-1)}(2^{2/3} - \frac{1}{2}))^{3/2}}$$

# Results

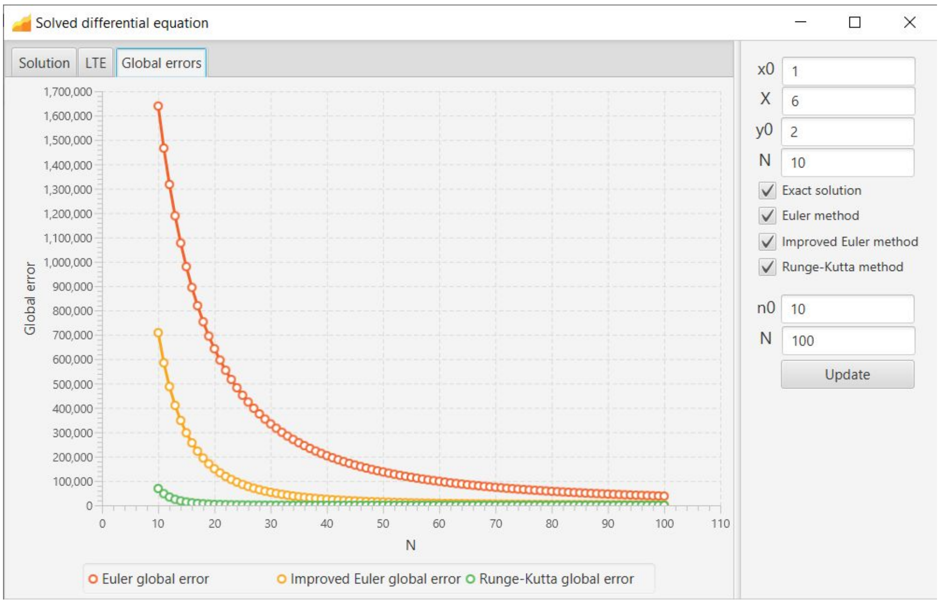
Chart of solution and approximate values. We can notice that the Runge-Kutta method calculates the most approximate values, the worst approximation is done by the Euler method.



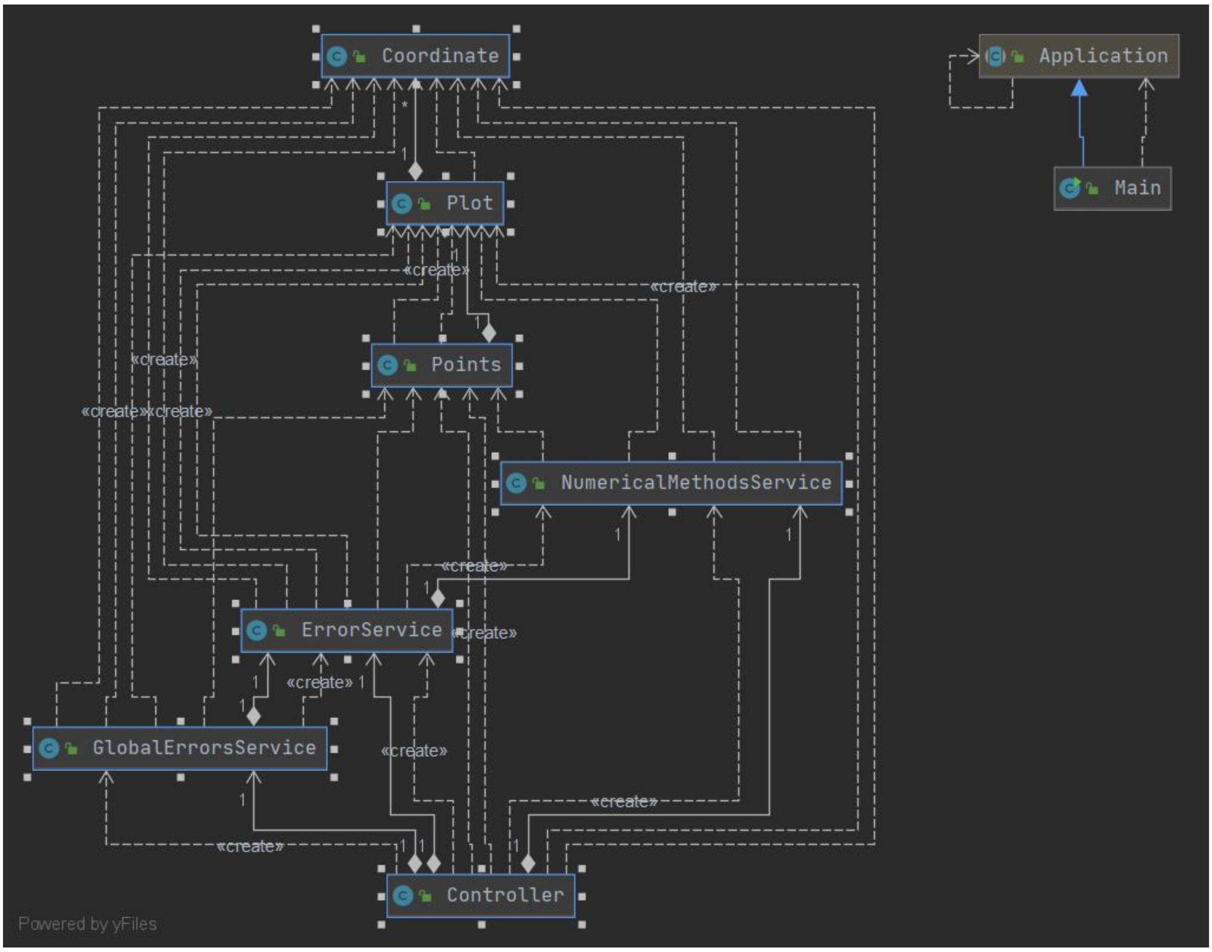

In addition, the Runge-Kutta method also has the smallest error, and the Euler method has a bigger error.




The third chart shows us that the more steps we take, the less error methods have.



# UML-diagram


Coordinate
Application
Plot
Main
«create»
«create»
Points
«create»
«create»«create»
NumericalMethodsService
«create»
ErrorService
«create»
«create»
«create»
GlobalErrorsService
«create»
«create»
Controller
Powered by yFiles

# Most interesting parts of code

```java
public void rungeKuttaMethod(Plot plot) {
    ArrayList<Coordinate> resultCoordinates = new ArrayList<>();

    double left = Plot.getX0();
    double right = Plot.getX();

    double h = (right - left) / Plot.getN();

    double prevX = Plot.getX0();
    double prevY = Plot.getY0();
    resultCoordinates.add(
            new Coordinate(prevX, prevY)
    );
    for (int i = 1; i <= Plot.getN(); i++) {
        double x = i * h + left;
        resultCoordinates.add(
                new Coordinate(x, rungeKuttaFunction(prevX, prevY, h))
        );
        prevX = x;
        prevY = resultCoordinates.get(resultCoordinates.size() - 1).getY();
    }

    plot.setCoordinates(resultCoordinates);
}
```

```java
public Plot rungeKuttaErrors(double steps){
    ArrayList<Coordinate> resultErrors = new ArrayList<>();

    double left = Plot.getX0();
    double right = Plot.getX();

    double h = (right - left) / steps;

    double prevX = left;
    double prevY = 0;
    resultErrors.add(
            new Coordinate(prevX, prevY)
    );
    for (int i = 1; i <= steps; i++) {
        double x = i * h + left;
        resultErrors.add(
                new Coordinate(x, rungeKuttaError(prevX, h))
        );
        prevX = x;
    }

    return new Plot(resultErrors);
}
```

```java
public void rungeKuttaGlobalErrors(Plot plot){
    ArrayList<Coordinate> resultGlobalErrors = new ArrayList<>();

    for (int i = Plot.getN0(); i <= Plot.getN1(); i++){
        resultGlobalErrors.add(
                new Coordinate(
                        i,
                        findMaximum(errorService.rungeKuttaErrors(i).getCoordinates())
                )
        );
    }

    plot.setCoordinates(resultGlobalErrors);
}

private double findMaximum(List<Coordinate> globalErrors){
    double maxError = 0;

    for (Coordinate error :
            globalErrors) {
        maxError = Math.max(error.getY(), maxError);
    }

    return maxError;
}
```

```java
public class Plot {

    private List<Coordinate> coordinates;
    private static double x0;
    private static double X;
    private static double y0;
    private static int N;
    private static int n0;
    private static int n1;
```

```java
public class Coordinate {

    private double x;
    private double y;
```

```java
public class Points {

    private Plot exactSolution;
    private Plot eulerSolution;
    private Plot improvedEulerSolution;
    private Plot rungeKuttaSolution;
    private Plot eulerError;
    private Plot improvedEulerError;
    private Plot rungeKuttaError;
    private Plot eulerGlobalError;
    private Plot improvedEulerGlobalError;
    private Plot rungeKuttaGlobalError;
```