1 Exact Solution

Differential equation with IVP:

$$y' = 3xe^x - y(1 - \frac{1}{x}), \quad x_0 = 1, \quad y_0 = 0 \tag{1}$$

1.1 Solution

Domain: x $\neq 0$

$y' = 3xe^x - y(1 - \frac{1}{x})$

$y' + y(1 - \frac{1}{x}) = 3xe^x$

Complementary equation: $y' + y(1 - \frac{1}{x}) = 0$

$y' = -y(1 - \frac{1}{x})$

$\frac{y'}{y} = (\frac{1}{x} - 1)$

$\int \frac{dy}{y} = \int (\frac{1}{x} - 1)dx$

$ln(y) = ln(x) - x + c$

$y = cxe^{-x}$

Substitution: $y = cxe^{-x}$

$y' = u'xe^{-x} + u(e^{-x} - xe^{-x}) = u'xe^{-x} + ue^{-x} - uxe^{-x}$

$u'xe^{-x} + u'xe^{-x} + ue^{-x} - uxe^{-x} = 3xe^{-x}$

$u'xe^{-x} = 3xe^{-x}$

$u'e^{-x} = 3e^{-x}$

$u' = 3e^{2x}$

$\int du = \int 3e^{2x}dx$

$\int du = \frac{1}{2}\int 3e^{2x}d(2x)$

$u = \frac{1}{2}3e^{2x} + c$

$y = 1.5e^x + ce^{-x}x$

$y = 1.5e^x + ce^{-x}x$

$x \neq 0$

1.2 Initial Value Problem

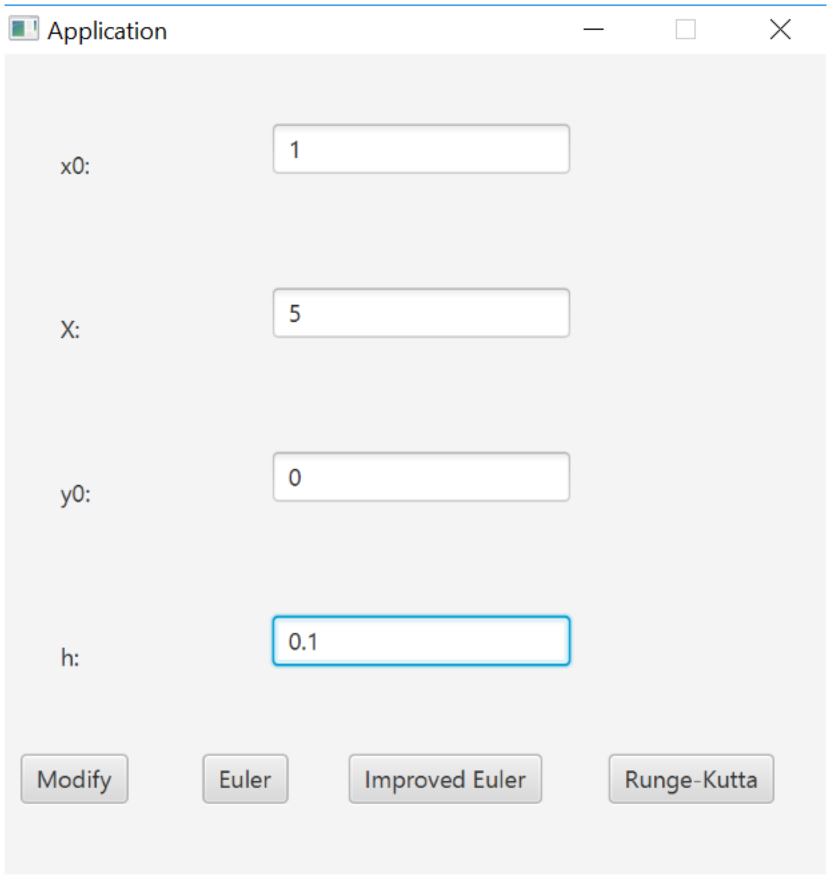
$$c = \frac{y}{(xe^{-x})} - 1.5e^{2x};$$
Initial value: y = 0, x = 1
$$c = \frac{0}{(1e^{-1})} - 1.5e^{2};$$
$$c = -1.5e^{2};$$
$$y = 1.5e^{x} + -1.5e^{2}e^{-x}x$$
$$x \neq 0$$

2 Application

Programming language: Java
GUI: JavaFx

2.1 Initial Values

It is plot which was made by application. x$_0 = 1$, $y_0 = 0$, $X = 5$, $h = 0.1$



Application

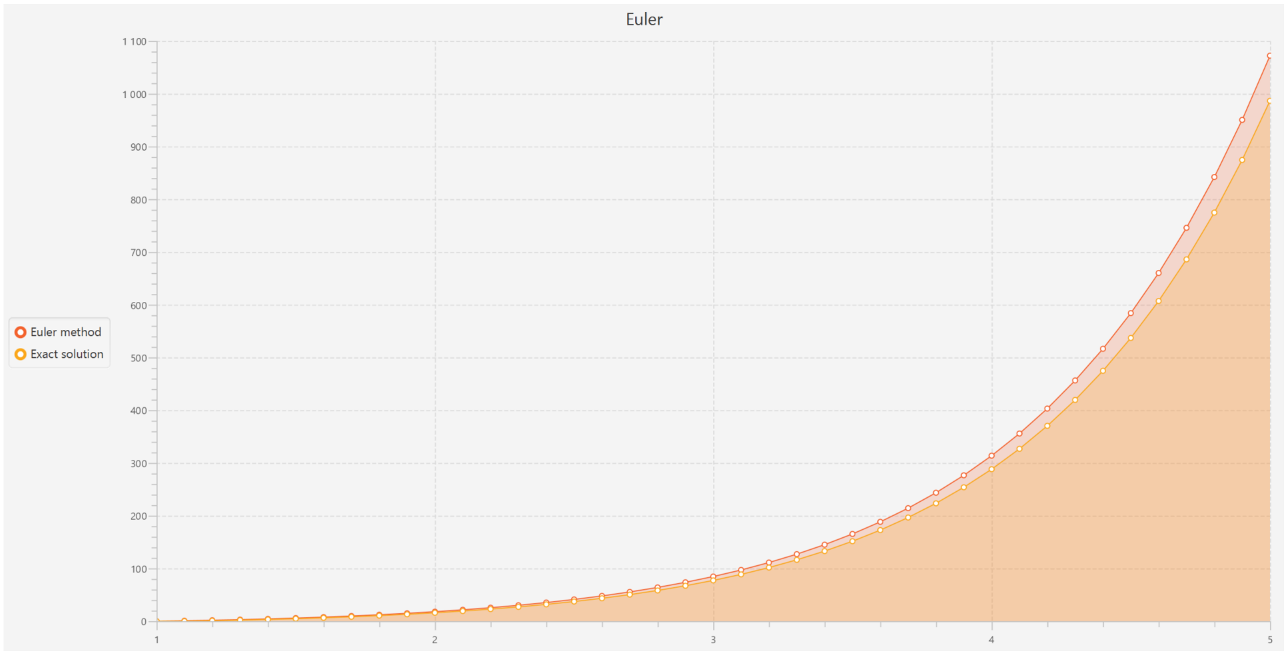
x0: 1

X: 5

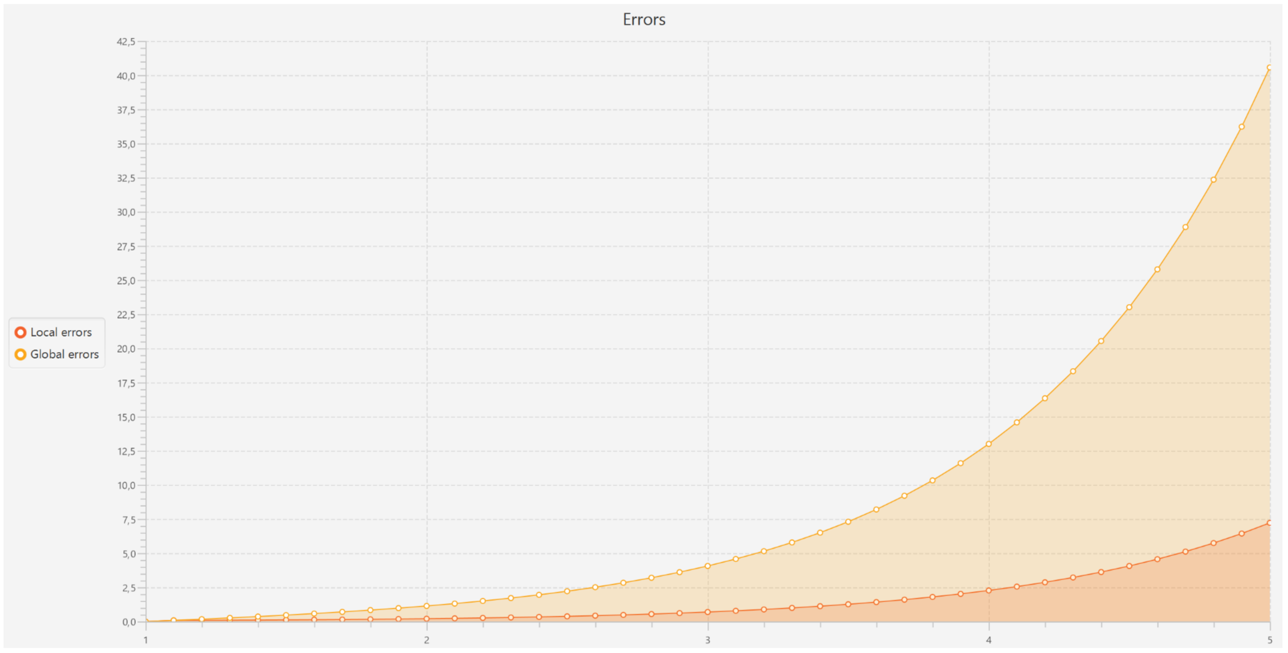
y0: 0

h: 0.1

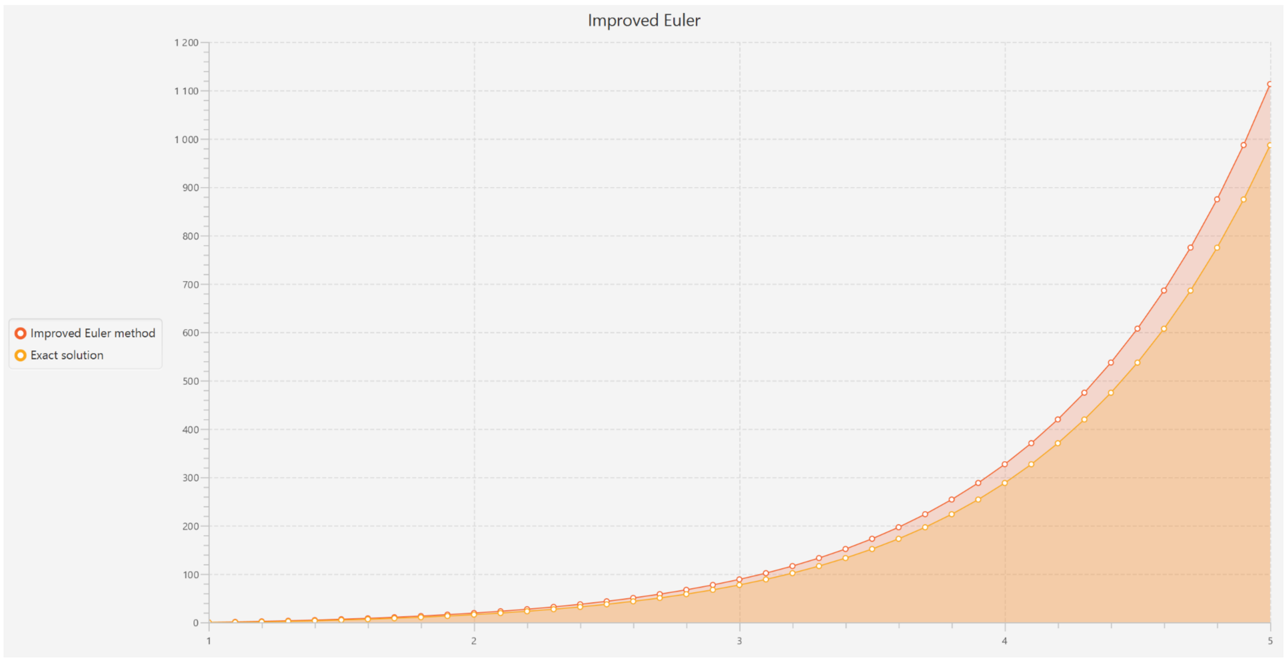
2.2 Euler Method



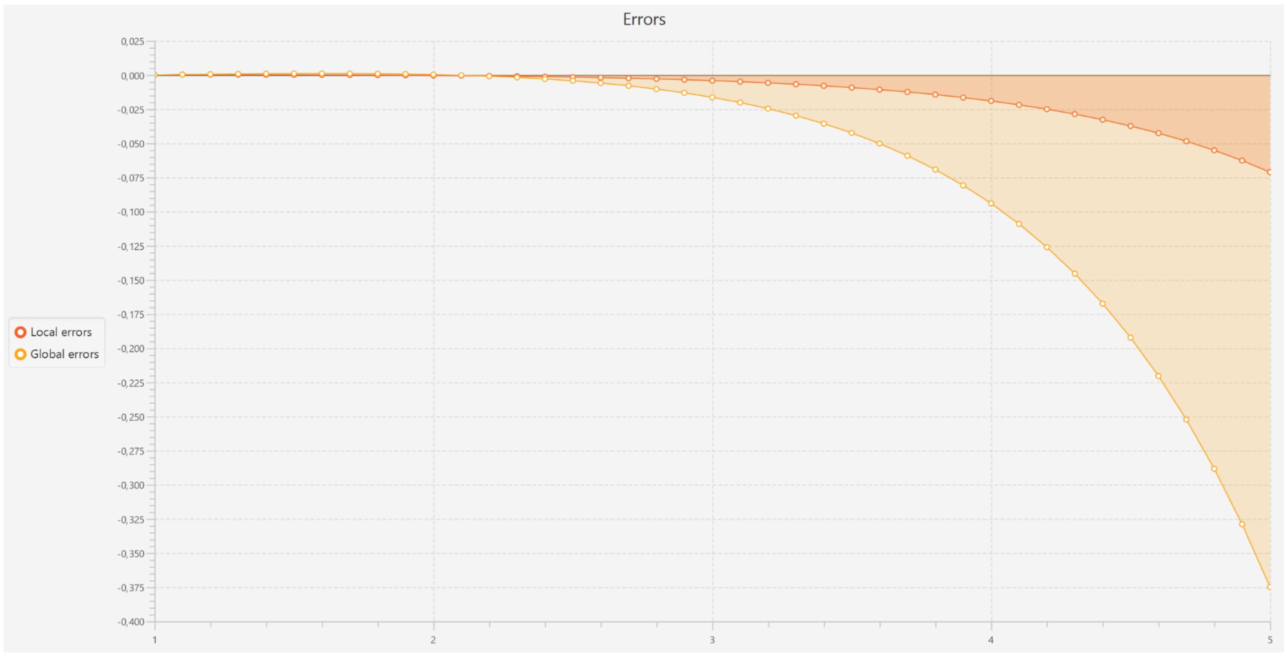
Local and Global errors



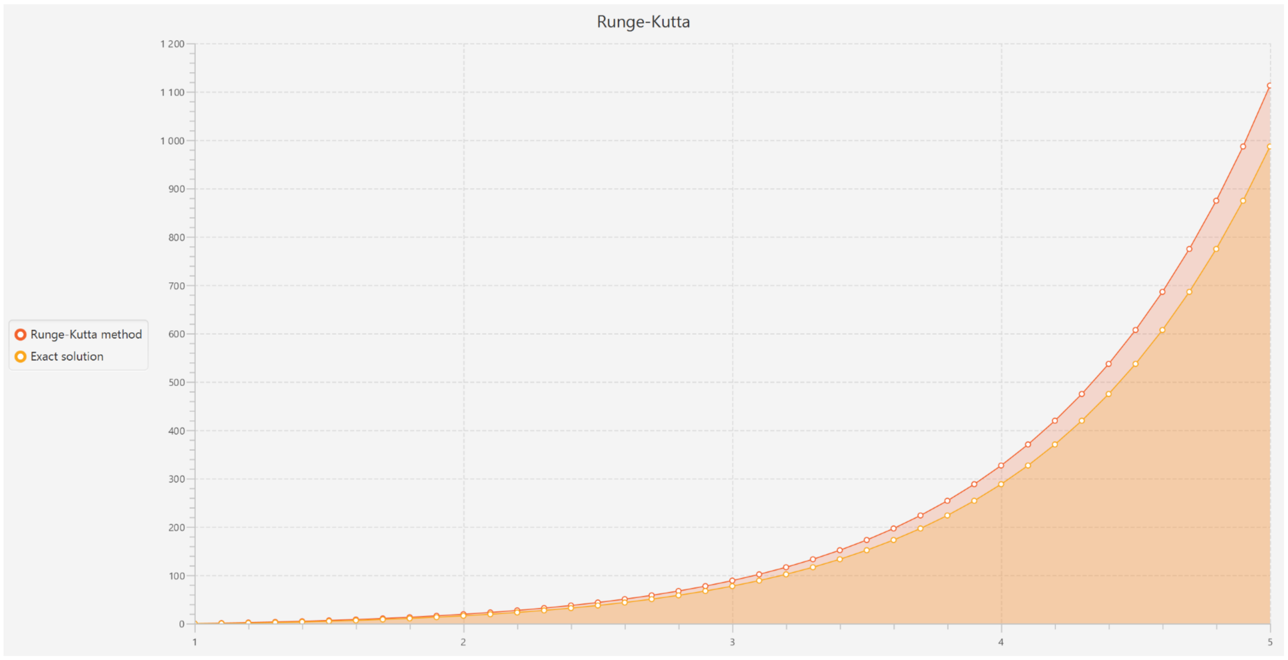
2.3 Improved Euler Method



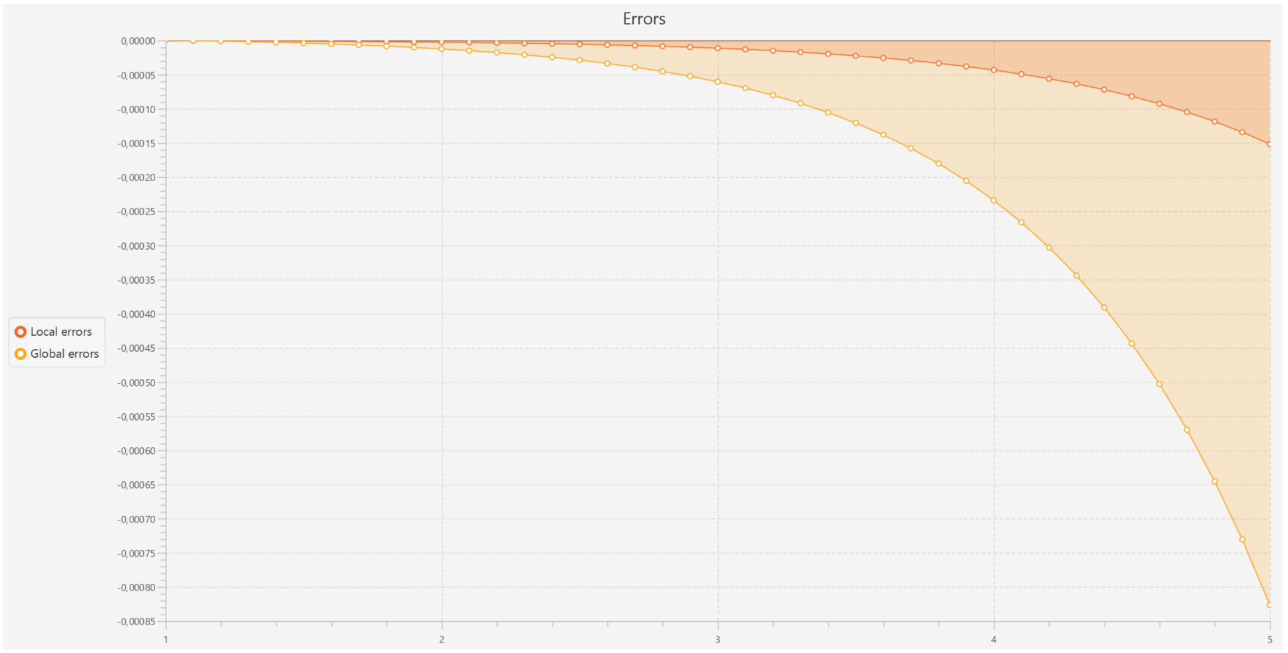
Local and Global errors



2.4 Runge-Kutta Method



Local and Global errors



2.5 Analysis

As we see on plot of errors, Runge-Kutta method is better than Improved Euler method and Euler method.

2.6 Additional part

As x isn't equal 0, numerical methods broken on range with 0. Before 0 it worked but in point 0 we have uncertainty. That is why application throw the exception when there is range with 0.

3 Code

There are incomplete class Euler.

```
public class Euler {
    ...
    private void countC() {
        c = yMin / (xMin * Math.exp(-xMin)) - 1.5 * Math.exp(2 * xMin);
    }

    private double solutionFunction(double x) {
        return 1.5 * Math.exp(x) * x + c * x * Math.exp(-x);
    }

    public double formula(double x, double y) {
        return y + h * function(x, y);
    }

    public double function(double x, double y) {
        if (x != 0) {
            return 3 * x * Math.exp(x) - y * (1 - 1 / x);
        } else {
            return 0;
        }
    }

    public void countPoints() {
        if (points != null) {
            points = null;
        }
        points = new LinkedList<Point>();
        ((LinkedList<Point>) points).addFirst(new Point(xMin, yMin));
        double xCurrent = xMin;
        double yCurrent = yMin;
        while (xCurrent + h <= xMax) {
            yCurrent = formula(xCurrent, yCurrent);
            xCurrent += h;
            points.add(new Point(xCurrent, yCurrent));
        }
    }

    public void countLocalErrors() {
       if (localErrors != null) {
            localErrors = null;
        }
        localErrors = new LinkedList<>();
        localErrors.add((double) 0);
        for (int i = 1; i < points.size(); i++) {
            double error = solutionFunction(points.get(i).x) -
                formula(points.get(i - 1).x, solutionFunction(points.get(i - 1).x));
            localErrors.add(error);
        }
    }

    public void countGlobalErrors() {
        if (globalErrors != null) {
            globalErrors = null;
        }
        globalErrors = new LinkedList<>();
        for (Point point : points) {
            double yExact = solutionFunction(point.x);
            globalErrors.add(yExact - point.y);
        }
    }
    ...
}
```

Complete classes ImprovedEuler and RungeKutta(inherit by Euler).

```
public class ImprovedEuler extends Euler {
    public ImprovedEuler(double xMin, double yMin, double xMax, double h) {
        super(xMin, yMin, xMax, h);
    }


    @Override
    public double formula(double x, double y) {
        return y + h * function(x + h / 2, y + (h / 2) * function(x, y));
    }
}

public class RungeKutta extends Euler {
    public RungeKutta(double xMin, double yMin, double xMax, double h) {
        super(xMin, yMin, xMax, h);
    }

    @Override
    public double formula(double x, double y) {
        double k1 = function(x, y);
        double k2 = function(x + h / 2, y + h * k1 / 2);
        double k3 = function(x + h / 2, y + h * k2 / 2);
        double k4 = function(x + h, y + h * k3);
        double deltaY = (h / 6) * (k1 + 2 * k2 + 2 * k3 + k4);
        return y + deltaY;
    }
}
```

4 Links

GitHub: https://github.com/kamilyagimatova/DEAssignment2k18