Computational Practicum report

Latypov Daniil BS20-01

Analytical solution

General solution

The following differential equation is given:

$$y' = \frac{4}{x^2} - \frac{y}{x} - y^2 \tag{1}$$

The domain of x is $x \in \mathbf{R}\backslash\{0\}$.
Equation can be rewritten in the following way:

$$y' + (-\frac{1}{x})y + (-1)y^2 = \frac{4}{x^2}$$

From this we can see, that the equation is a Ricatti equation.
We suppose that $y = y_2 + u(x)$, $y_1 = \frac{a}{x}$. Then $y_1' = -\frac{a}{x^2}$.

Substituting $y = y_1$ into (1), we get:

$$-\frac{a}{x^2} = \frac{4}{x^2} - \frac{a}{x^2} - \frac{a^2}{x^2}$$

From this equation we can easily find $a = 2$.
Then, $y_1 = \frac{2}{x}$. $y = \frac{2}{x} + u(x)$, $y' = -\frac{2}{x^2} + u'(x)$
Substituting into (1) we get:

$$-\frac{2}{x^2} + u' = -\frac{2}{x^2} - \frac{u}{x} - \frac{4}{x^2} - u^2 \;\; - \frac{4u}{x} + \frac{4}{x^2}$$

Which is equivalent to:

$$u' = -u^2 - \frac{5u}{x} \tag{2}$$

Then, with the use of substitution $u = \frac{1}{z}$, $u' = -\frac{1}{z^2}z'$ we get:

$$-\frac{z'}{z^2} + \frac{5}{zx} = -\frac{1}{z^2}$$

Multiplying both sides by z^2 and rewriting we get the following equation:

$$z' = 1 + \frac{5z}{x} \tag{3}$$

It's complementary equation can be written as:

$$\frac{1}{z}dz = \frac{5}{x}dx$$

Taking integral of both sides:

$$ln|z| = 5ln|x|$$

Therefore,

$$z = x^5 * C_1 \tag{4}$$

Where $C_1 = C_1(x)$, $z' = 5x^4 * C_1 + x^5 * C_1'$.
Substituting into (3), we get:

$$5x^4 * C_1 + x^5 * C_1' = \frac{x^5 * C_1}{x} - 1$$

From this we find:

$$x^5 * C_1' = -1 \Rightarrow C_1 = \frac{1}{4x^4} + C_2$$

Substituting into (4) we get:

$$z = -\frac{x}{4} + x^5 * C_2$$

Backsubstituting $z = \frac{1}{u}$ we get:

$$u = \frac{1}{-\frac{x}{4} + x^5 * C_2} \Rightarrow u = \frac{4}{4x^5 * C_2 - x}$$

Backsubstituting $y = \frac{2}{x} + u$ we get the general solution on domain $x \in \mathbf{R} \backslash \{0\}$, where $4C * x^5 - \frac{x}{4} \neq 0$:

$$y = \frac{2}{x} + \frac{4}{4C * x^5 - x}$$

Initial value problem

Solving the Initial value problem:

$$0 = 2 + \frac{4}{4C - 1} \Rightarrow C = -\frac{1}{4}$$

Therefore, the particular solution is:

$$y = \frac{2}{x} + \frac{4}{-x - x^5}$$

Analysis of points of discontinuity

From the general solution we can express C

$$C = \frac{1}{x_0^4}\left(\frac{1}{x_0 y_0 - 2} + 1\right)$$

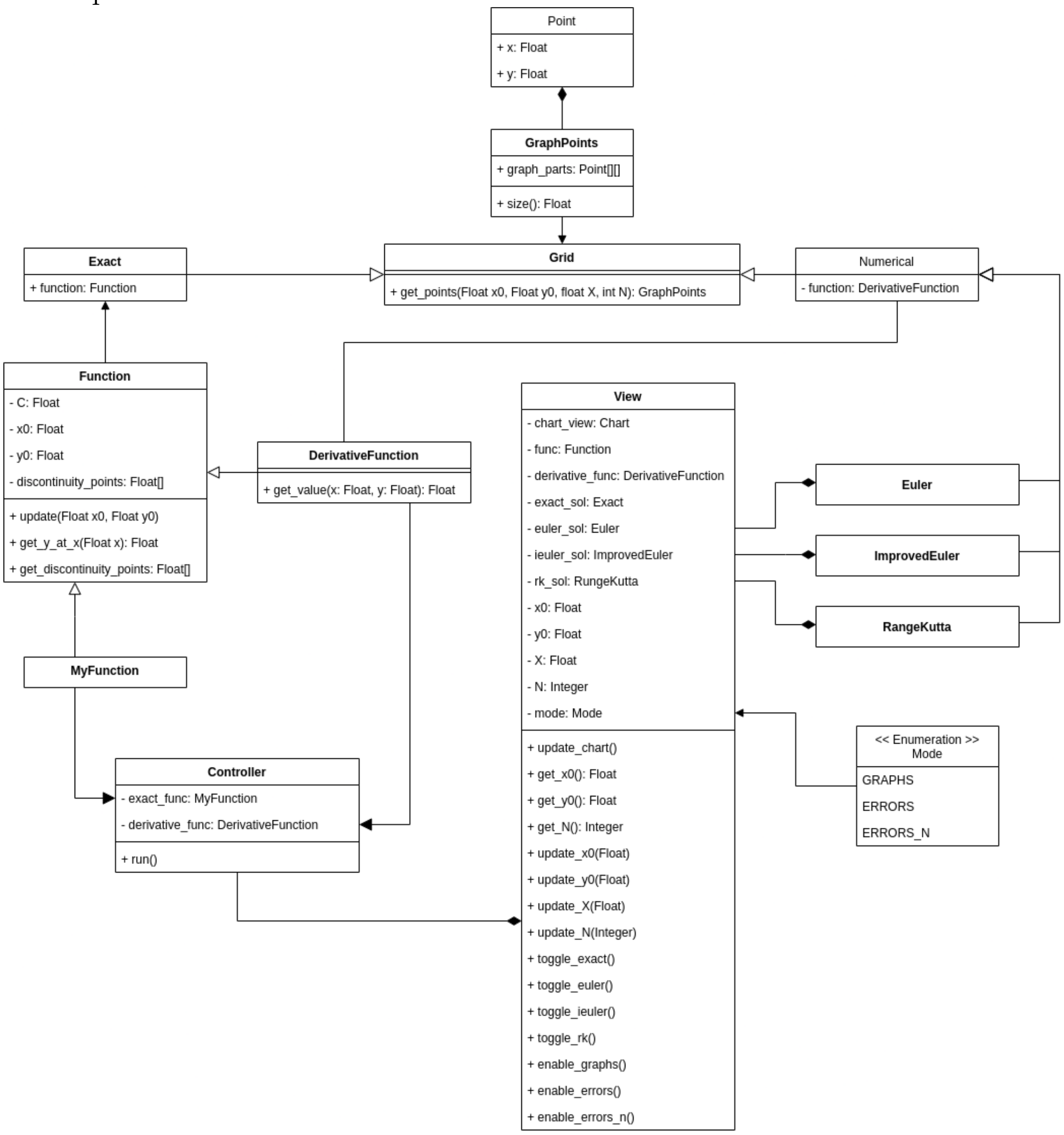
If $C = 0$, then there is only one point of discontinuity, $x = 0$.
If $C \neq 0$, then there are 3 points of discontinuity, that can be found in the following way:

$$\begin{cases} x \neq 0, \\ 4C * x^5 - x \neq 0 \end{cases} \Rightarrow \begin{cases} x \neq 0, \\ x \neq \pm\sqrt[4]{\frac{1}{4C}} \end{cases}$$

Code

UML Diagram

The following diagram shows the class structure of my program. It follows the model-view-controller pattern.



Implementation

The program was written using the QT framework using C++. It follows SOLID principles and the Model-View-Controller pattern.

The implementation includes plotting of the exact solution and numberical solutions (Euler's method, Improved Euler's method and Runge-Kutta method), plotting of local and average errors based on grid size, and provides control over variables to the user.

The following code snippet shows how the value is found for each point for Runge-Kutta method:

```
...
float x_prev = x;
x += step;
float k1 = step * function->get_value(x_prev, y);
float k2 = step * function->get_value(x_prev + step/2, y + k1/2);
float k3 = step * function->get_value(x_prev + step/2, y + k2/2);
float k4 = step * function->get_value(x_prev + step, y + k3);
y += 1.0/6.0 * (k1+2*k2+2*k3+k4);
points.graph_parts[points.size()-1].push_back( Point(x, y) );
...
```

Implementation supports existance of points of discontinuity. If the graph includes points of discontinuity, it will be split into multiple parts. At each point, the function checks if it is equal or bigger than the next point of discontinuity, if it bigger and the current graph part has more than 0 points, it creates a new graph part (and skips the point completely if it is exactly equal to the point of discontinuity).

```
GraphPoints Exact::get_points(float x0, float y0, float X, int N){
...
    if(std::find(next_disc_pt, disc_pts.end(), x) != disc_pts.end()) {
        if ( points[points.size()-1].size() != 0 )
            points.add_graph_part();
        if ( next_disc_pt != disc_pts.end() )
            next_disc_pt++;
        continue;
    }
    else if ( x > *next_disc_pt && next_disc_pt != disc_pts.end()){
        while (x > *next_disc_pt && next_disc_pt != disc_pts.end())
            next_disc_pt++;
        if ( points[points.size()-1].size() != 0 )
            points.add_graph_part();
    }
...
```

Graphs of Errors

Program includes 3 pages:

- Graphs of numerical and analytical solutions
- Errors of each numerical method
- Average error of each method based on step length (controlled through N)

Each page has controls of x_0, y_0, X, N and allows the user to choose which numerical solutions they wish to see (and if they wish to see the exact solution on the Graph page).

The following code snippet shows how local error is calculated for each part of the graph for the Euler's method:

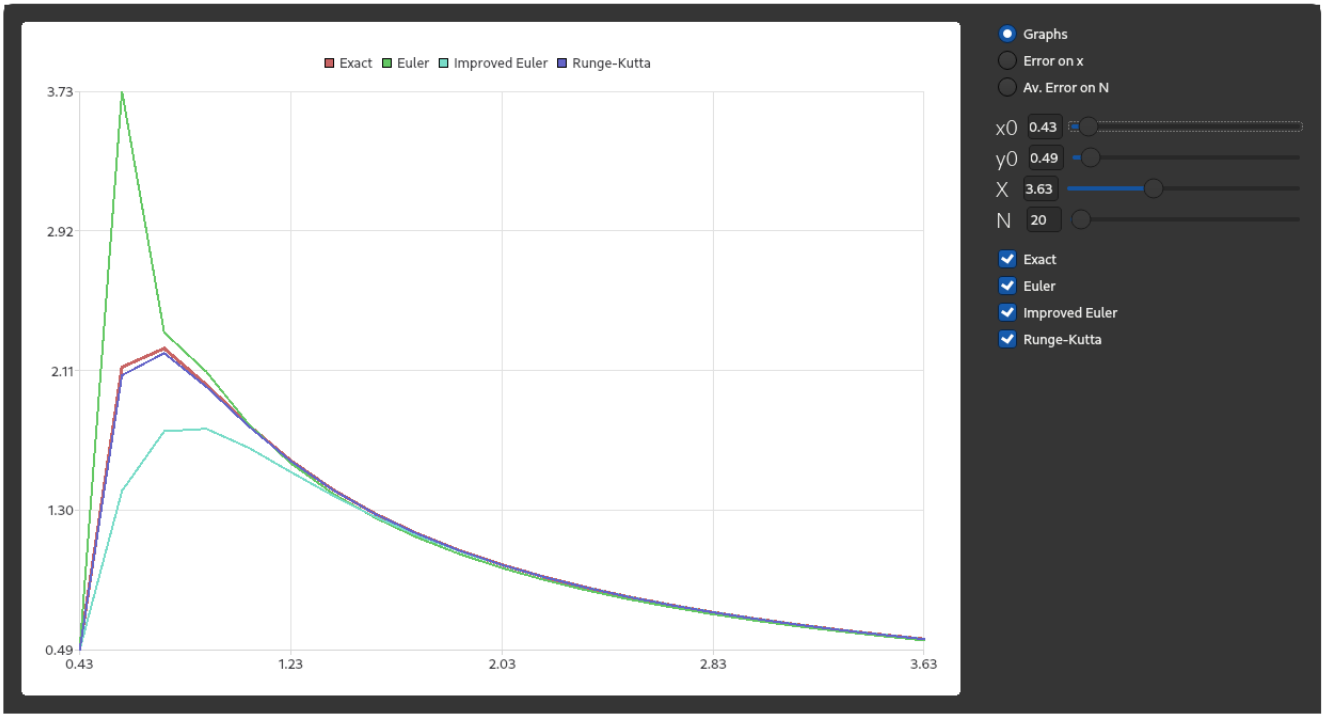
```
QLineSeries* series = new QLineSeries();
series->setName("Euler");
for (int j = 0; j < graph_points_euler[i].size(); ++j){
    float err = abs(graph_points_euler[i][j].y - graph_points_exact[i][j].y);
    series->append(graph_points_euler[i][j].x, err);
}
chartView->chart()->addSeries(series);
```

Miscelaneous points

Implementation is somewhat general. For example, it is easy to replace the differential equation by inheriting the Function class and supplying the new equation to the Controller.

Some attention was paid to optimization, the program calculates the constant C and points of discontinuity only once for each x0, y0 and stores them. Runge-Kutta implementation goes through each enabled numerical method in the same loop.

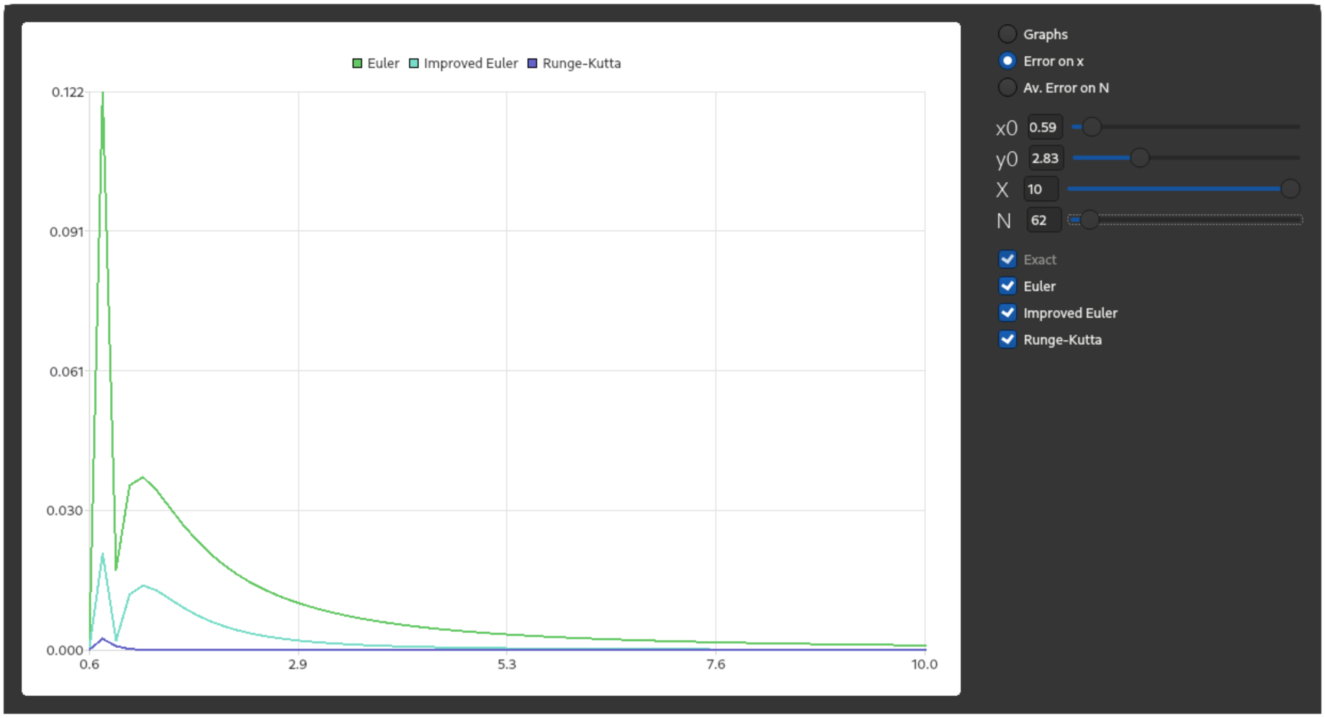
Screenshots



● Graphs
○ Error on x
○ Av. Error on N

x0 0.43
y0 0.49
X 3.63
N 20

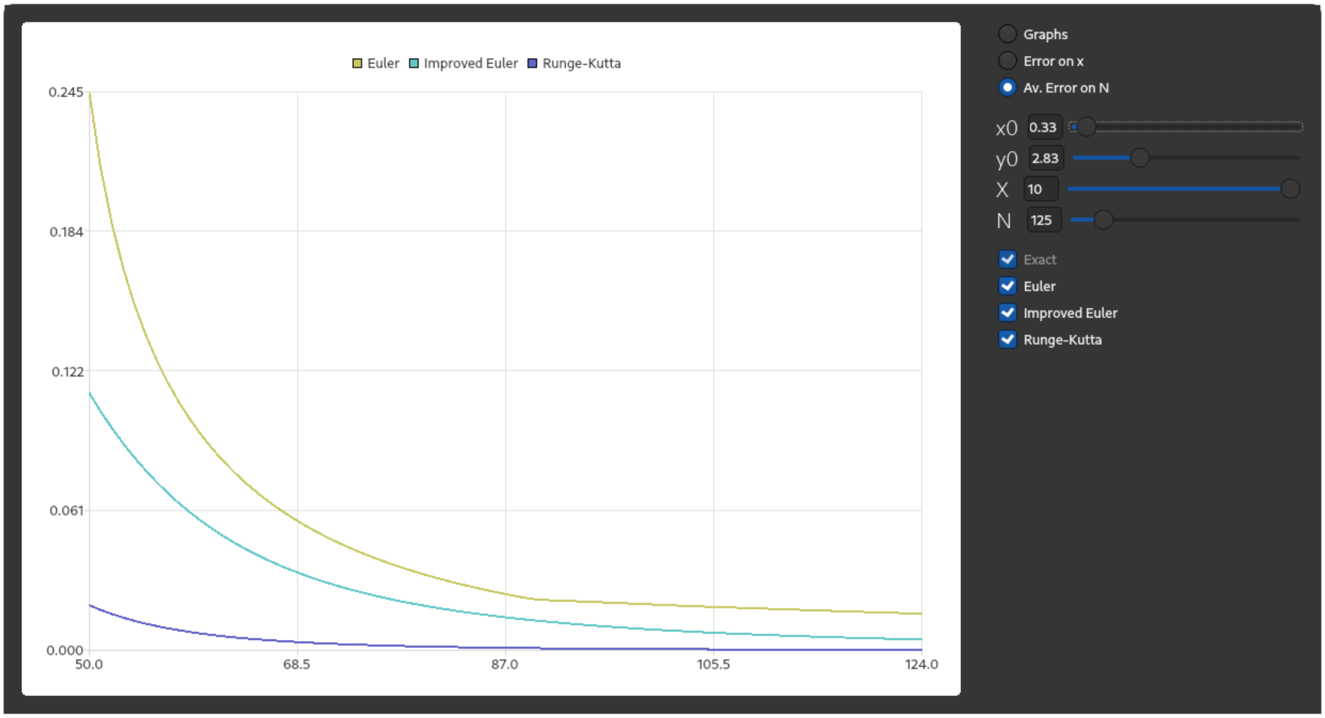
☑ Exact
☑ Euler
☑ Improved Euler
☑ Runge-Kutta



○ Graphs
● Error on x
○ Av. Error on N

x0 0.59
y0 2.83
X 10
N 62

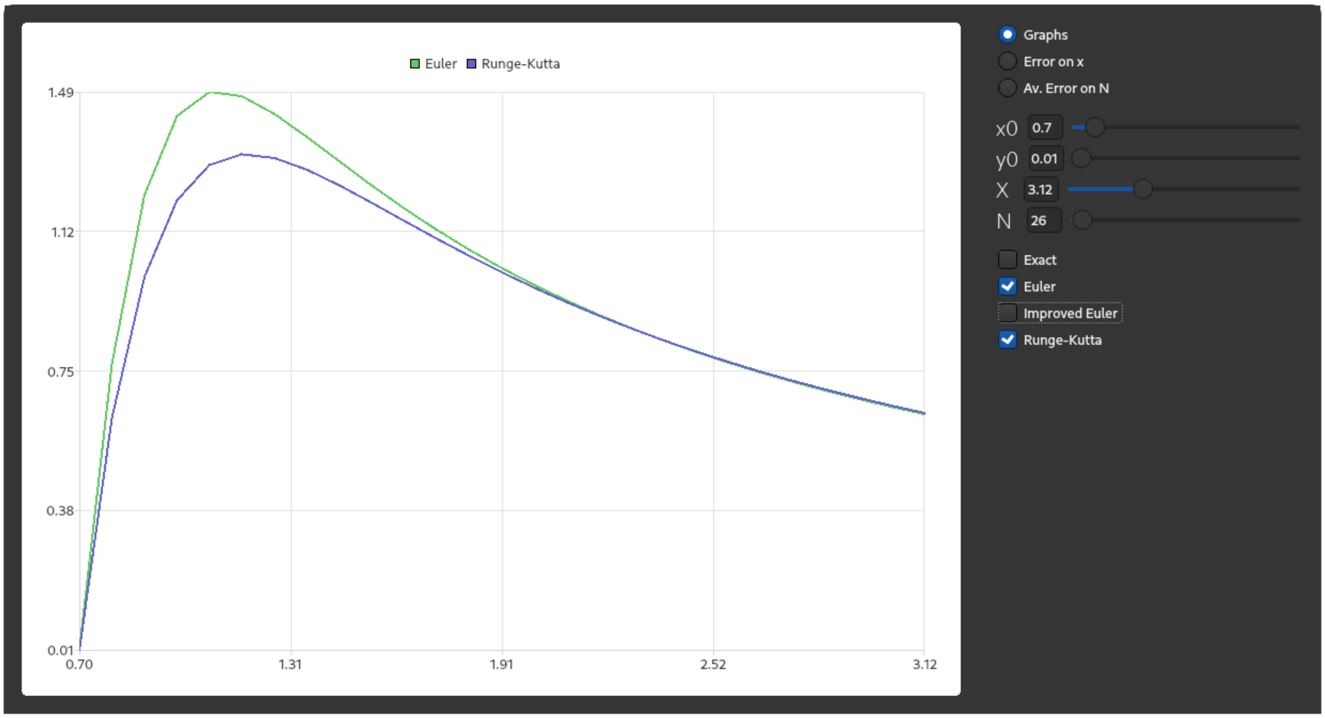
☑ Exact
☑ Euler
☑ Improved Euler
☑ Runge-Kutta



○ Graphs
○ Error on x
● Av. Error on N

x0 0.33
y0 2.83
X 10
N 125

☑ Exact
☑ Euler
☑ Improved Euler
☑ Runge-Kutta



● Graphs
○ Error on x
○ Av. Error on N

x0 0.7
y0 0.01
X 3.12
N 26

☐ Exact
☑ Euler
☐ Improved Euler
☑ Runge-Kutta