Variant 12
Andrey Vagin **Computational Practicum Report** BS20-01

Initial Value Problem

$$\left\{ \begin{array}{l} y' = 5 - x^2 - y^2 + 2xy \\ y(0) = 1 \\ x \in (0, 20) \end{array} \right\}$$

Solution:

1. This is Nonlinear Non-homogeneous First order Differential Equation
$$y' = 5 - (y - x)^2 \tag{1}$$

2. Let's use the substitution $z(x) = y - x$:
$$\begin{gathered} y = z + x \\ y' = z' + 1 \end{gathered} \tag{2}$$

3. After substitution we obtain:
$$\begin{gathered} z' + 1 = 5 - z^2 \\ z' = -z^2 + 4 \end{gathered} \tag{3}$$

4. We got Riccati equation with particular solution $z = 2$. Now special substitution for Riccati equation can be applied:
$$\begin{gathered} u(x) = \frac{1}{z-2} \\ z(x) = \frac{1}{u} + 2 \\ z' = -\frac{1}{u^2}u' \end{gathered} \tag{4}$$

5. If we substitute (4) into (3) we will get:
$$\begin{gathered} -\frac{1}{u^2}u' = -(\frac{1}{u} + 2)^2 + 4 \\ u' - 4u = 1 \end{gathered} \tag{5}$$

6. We got Linear Differential Equation which can be solved in form:

$$\begin{gathered} u = (\textstyle\int 1 * e^{-4x} dx + C)e^{4x} \\ u = Ce^{4x} - \tfrac{1}{4} \end{gathered} \tag{6}$$

7. As $z = \frac{1}{u} + 2$ we get:

$$z = \frac{1}{Ce^{4x} - \frac{1}{4}} + 2 \tag{7}$$

8. From (2) $y = z + 2$, so we obtain the General Solution of our ODE:

$$y = \frac{1}{Ce^{4x} - \frac{1}{4}} + 2 + x, \text{ where } C \in \mathbb{R} \tag{8}$$

9. As we are solving IVP:

$$\left\{ \begin{array}{l} y = \frac{1}{Ce^{4x} - \frac{1}{4}} + x + 2 \\ y(x_0) = y_0 \end{array} \right\}$$

$$\begin{gathered} y_0 = \frac{1}{Ce^{4x_0} - \frac{1}{4}} + x_0 + 2 \\ C = e^{-4x_0}(\frac{1}{y_0 - x_0 - 2} + \frac{1}{4}) \end{gathered} \tag{9}$$

10. Let's substitute obtained value of C to get Exact Solution:

$$y = \frac{1}{e^{-4x_0}(\frac{1}{y_0 - x_0 - 2} + \frac{1}{4})e^{4x} - \frac{1}{4}} + x + 2 \tag{10}$$

11. Now we can investigate discontinuity points:

   (a) Firstly, C does not exist when $y_0 - x_0 = 2$:

$$y_0 - x_0 \neq 2 \tag{11}$$

   (b) If $C \leqslant 0$, then the determinant of (8) is always nonzero and there is no discontinuity points there

(c) If $C > 0$, then the determinant of (8) can be equal to zero

$$\begin{gathered} C > 0 \\ e^{-4x_0}(\tfrac{1}{y_0-x_0-2} + \tfrac{1}{4}) > 0 \\ y_0 - x_0 < -2 \end{gathered} \tag{12}$$

Now let's consider determinant of (10):

$$\begin{gathered} e^{-4x_0}(\tfrac{1}{y_0-x_0-2} + \tfrac{1}{4})e^{4x} - \tfrac{1}{4} \neq 0 \\ x \neq \tfrac{1}{4}\log(\tfrac{1}{4}e^{4x_0}(\tfrac{1}{y_0-x_0-2} + \tfrac{1}{4})^{-1}) \end{gathered} \tag{13}$$

Answer:

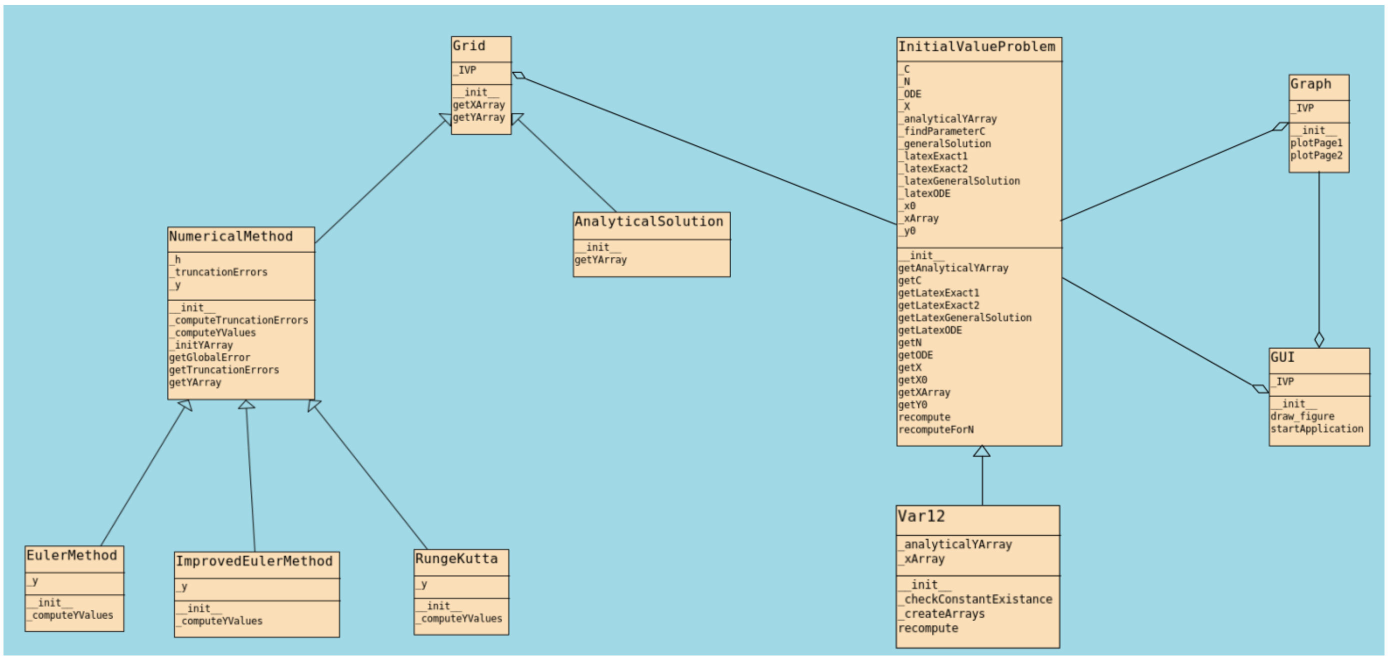
Given equation does not have trivial solution and has the most general nontrivial solution in form: $y = \frac{1}{e^{-4x_0}(\frac{1}{y_0-x_0-2}+\frac{1}{4})e^{4x}-\frac{1}{4}} + x + 2$ on

$$\left\{ \begin{gathered} x|x \in \mathbb{R} \wedge x_0 \in \mathbb{R} \wedge y_0 \in \mathbb{R} \wedge (y_0 - x_0 \neq 2)\wedge \\ ((y_0 - x_0 < -2) \wedge x \neq \tfrac{1}{4}\log(\tfrac{1}{4}e^{4x_0}(\tfrac{1}{y_0-x_0-2} + \tfrac{1}{4})^{-1})) \end{gathered} \right\}$$

Programming Part

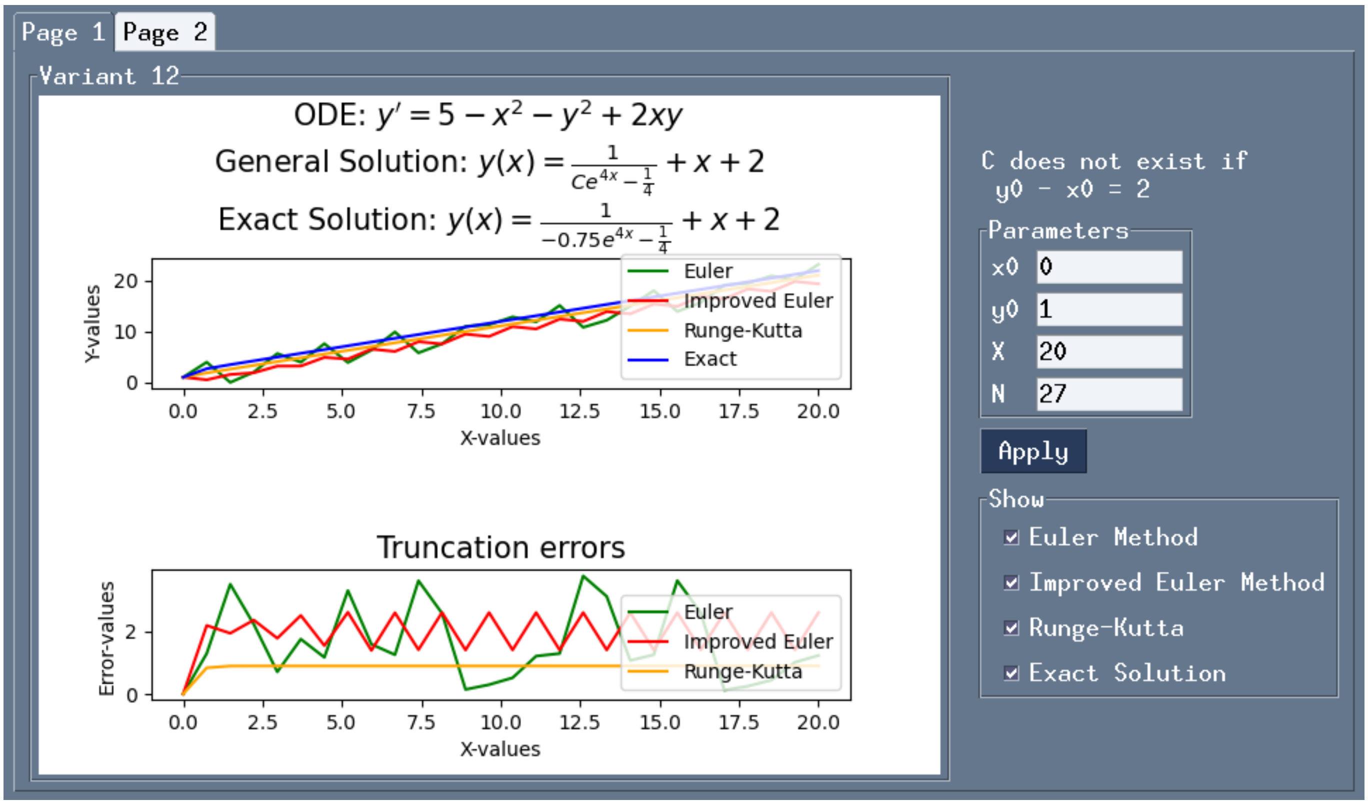
The application was written in Python with the use of NumPy, MatPlotLib, and PySimpleGUI.

UML Diagram:



Numerical investigations:

Let's look at the graphs of IVP var12 with 27 grid steps. We can observe that Euler's method and Improved Euler's method are not precise, and Runge-Kutta method shows a good result even with this grid steps size.



ODE: $y' = 5 - x^2 - y^2 + 2xy$

General Solution: $y(x) = \frac{1}{Ce^{4x} - \frac{1}{4}} + x + 2$

Exact Solution: $y(x) = \frac{1}{-0.75e^{4x} - \frac{1}{4}} + x + 2$

C does not exist if y0 − x0 = 2

Parameters

x0 0

y0 1

X 20

N 27

Show

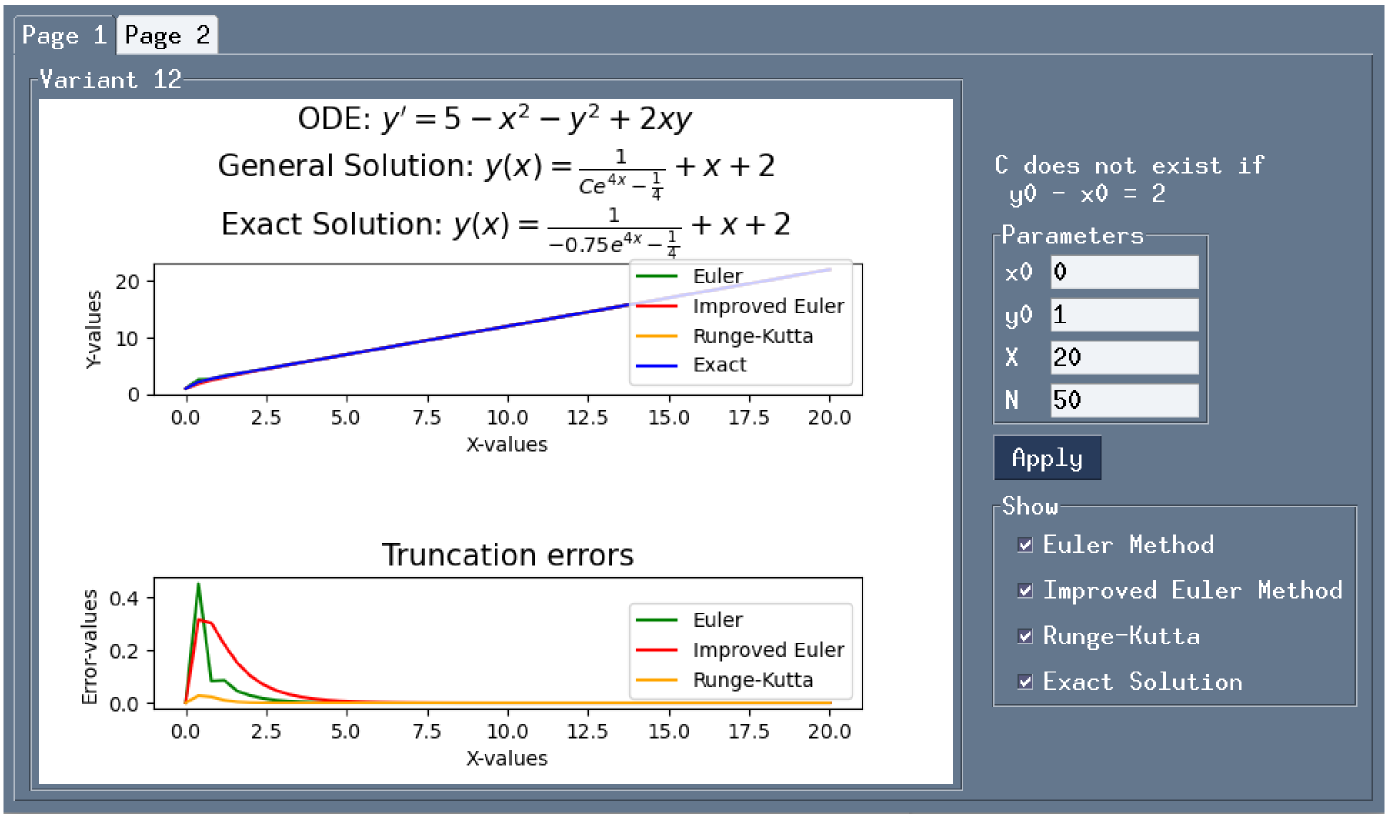
☑ Euler Method

☑ Improved Euler Method

☑ Runge-Kutta

☑ Exact Solution

With 50 grid steps, Euler's method and Improved Euler's method have significant errors in the beginning while Runge-Kutta method has almost zero errors during the whole interval. After the first quarter, all the methods have almost zero errors.



ODE: $y' = 5 - x^2 - y^2 + 2xy$

General Solution: $y(x) = \frac{1}{Ce^{4x} - \frac{1}{4}} + x + 2$

Exact Solution: $y(x) = \frac{1}{-0.75e^{4x} - \frac{1}{4}} + x + 2$

C does not exist if y0 - x0 = 2

Parameters

x0 0

y0 1

X 20

N 50

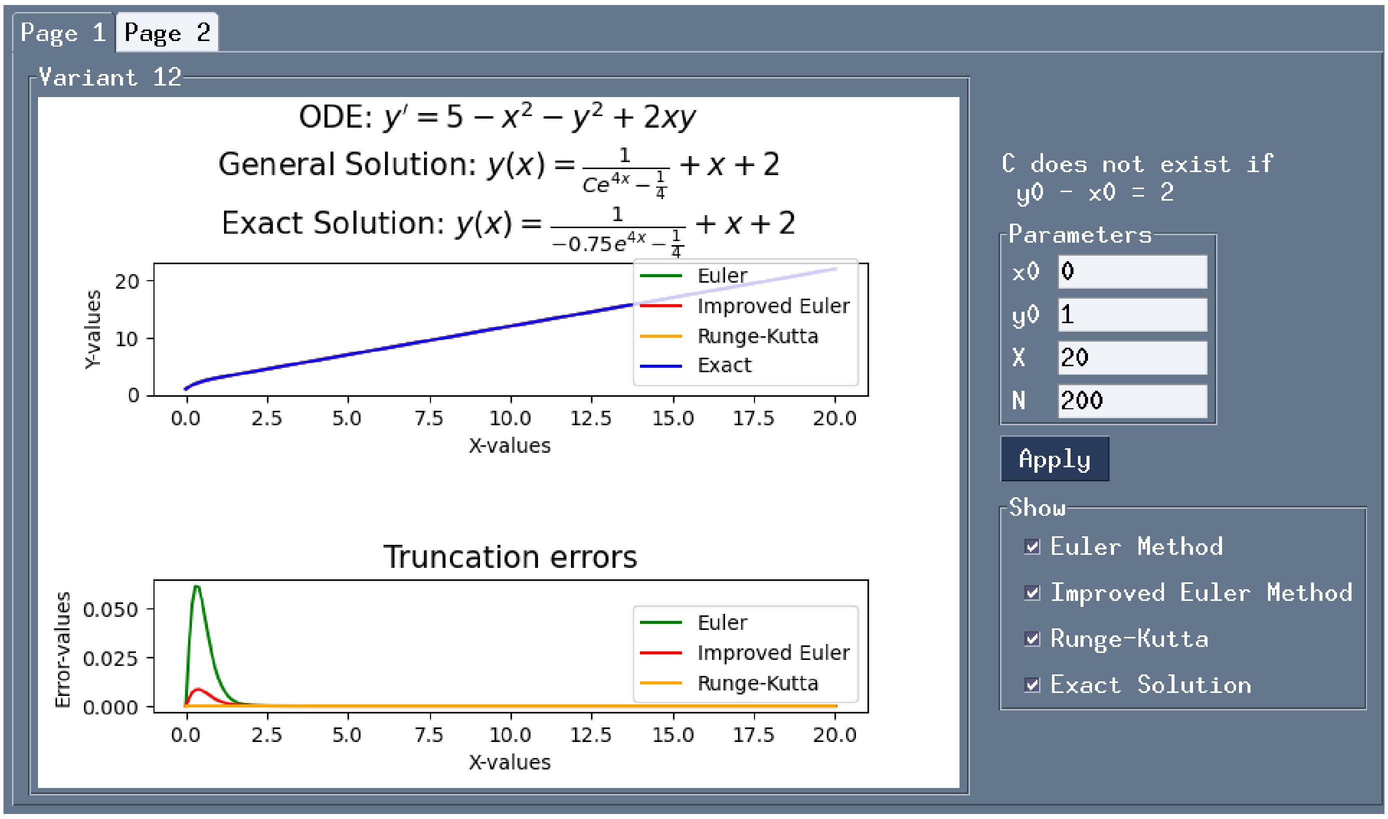
☑ Euler Method

☑ Improved Euler Method

☑ Runge-Kutta

☑ Exact Solution

With 200 grid steps size, only Euler's method shows significant error in the beginning but finally, truncation errors of all the methods go to zero.



ODE: $y' = 5 - x^2 - y^2 + 2xy$

General Solution: $y(x) = \frac{1}{Ce^{4x} - \frac{1}{4}} + x + 2$

Exact Solution: $y(x) = \frac{1}{-0.75e^{4x} - \frac{1}{4}} + x + 2$

C does not exist if y0 - x0 = 2

Parameters

x0 0

y0 1

X 20

N 200

☑ Euler Method

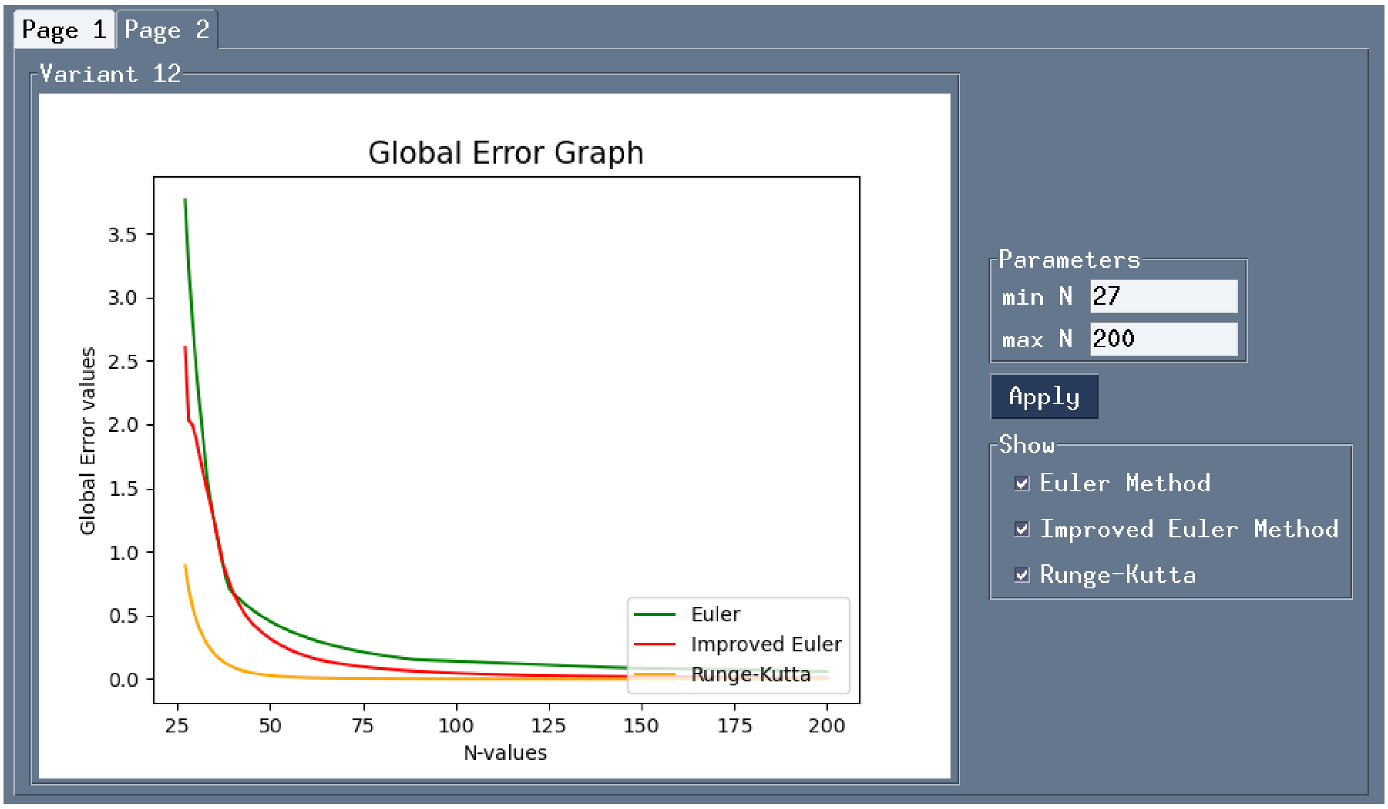
☑ Improved Euler Method

☑ Runge-Kutta

☑ Exact Solution

As we can conclude: Runge-Kutta method is the most precise one, and Euler's method is the most inaccurate.

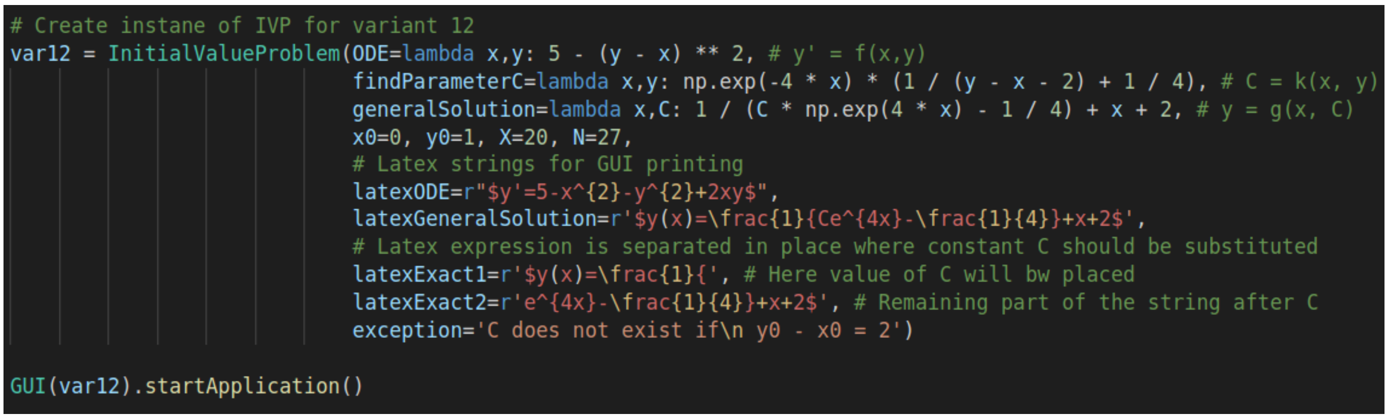
Investigation of graphs of Global Errors:



By increasing the number of steps we achieve almost absolute accuracy for Improved Euler's method and for Runge-Kutta method.

Some Implementation Features:

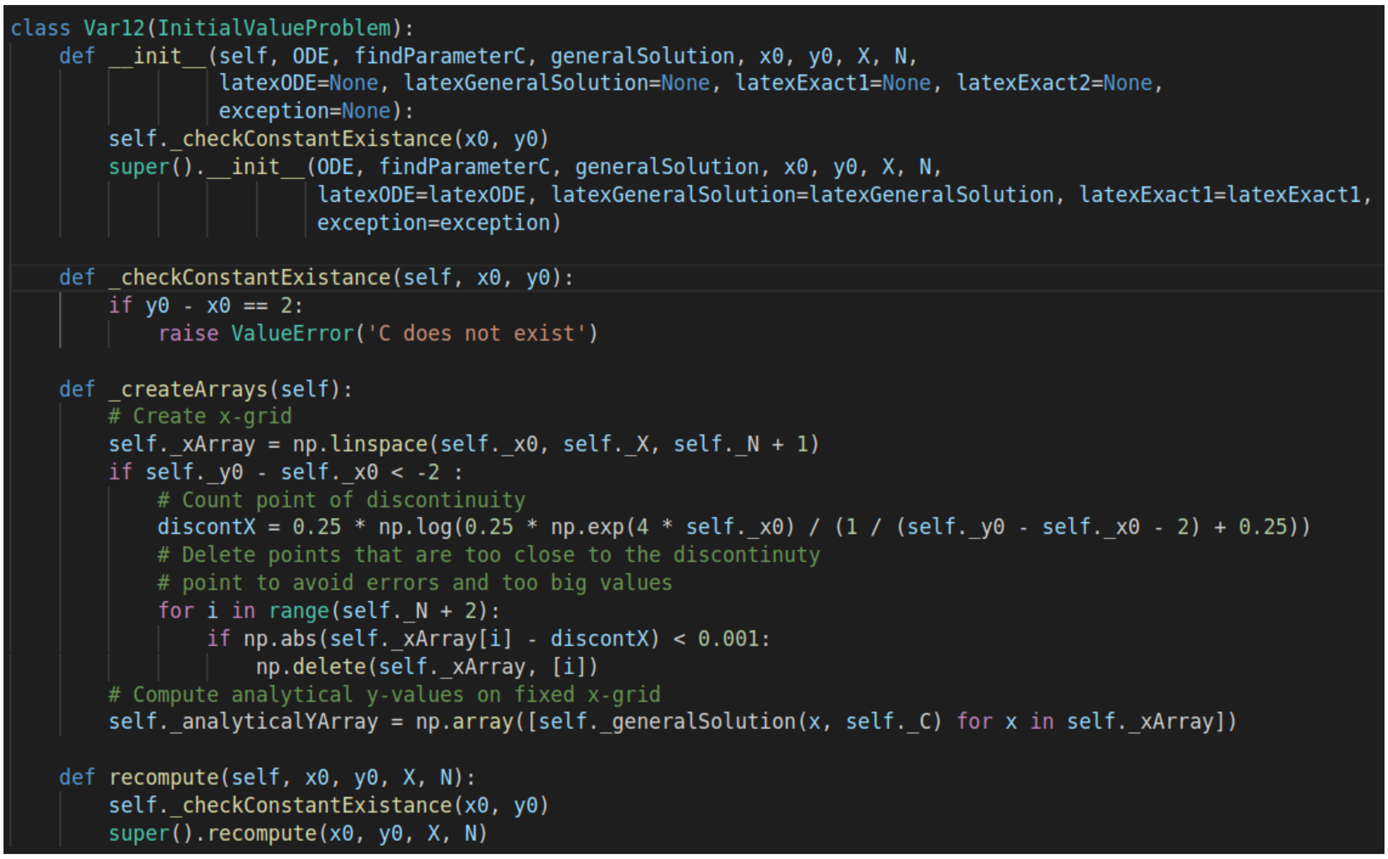
In my application, the user does not need to create separate classes for different variants of IVP if they do not have points of discontinuity. To start the application for the chosen variant the user should create an instance of the class "InintialValueProblem" and initialize it with appropriate lambda expressions for ODE and LaTeX expressions for GUI printing. After these actions, the user creates an instance of GUI for chosen variant and starts the application.



```python
# Create instane of IVP for variant 12
var12 = InitialValueProblem(ODE=lambda x,y: 5 - (y - x) ** 2, # y' = f(x,y)
                            findParameterC=lambda x,y: np.exp(-4 * x) * (1 / (y - x - 2) + 1 / 4), # C = k(x, y)
                            generalSolution=lambda x,C: 1 / (C * np.exp(4 * x) - 1 / 4) + x + 2, # y = g(x, C)
                            x0=0, y0=1, X=20, N=27,
                            # Latex strings for GUI printing
                            latexODE=r"$y'=5-x^{2}-y^{2}+2xy$",
                            latexGeneralSolution=r'$y(x)=\frac{1}{Ce^{4x}-\frac{1}{4}}+x+2$',
                            # Latex expression is separated in place where constant C should be substituted
                            latexExact1=r'$y(x)=\frac{1}{', # Here value of C will bw placed
                            latexExact2=r'e^{4x}-\frac{1}{4}}+x+2$', # Remaining part of the string after C
                            exception='C does not exist if\n y0 - x0 = 2')

GUI(var12).startApplication()
```

In the case of the 12-th variant, there is a discontinuity point. To handle it special class derived from “InintialValueProblem” should be created. The problem is solved by redefining grid creating method:



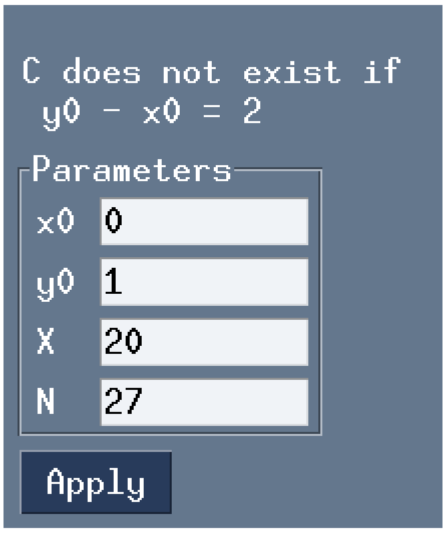
```python
class Var12(InitialValueProblem):
    def __init__(self, ODE, findParameterC, generalSolution, x0, y0, X, N,
                 latexODE=None, latexGeneralSolution=None, latexExact1=None, latexExact2=None,
                 exception=None):
        self._checkConstantExistance(x0, y0)
        super().__init__(ODE, findParameterC, generalSolution, x0, y0, X, N,
                         latexODE=latexODE, latexGeneralSolution=latexGeneralSolution, latexExact1=latexExact1,
                         exception=exception)

    def _checkConstantExistance(self, x0, y0):
        if y0 - x0 == 2:
            raise ValueError('C does not exist')

    def _createArrays(self):
        # Create x-grid
        self._xArray = np.linspace(self._x0, self._X, self._N + 1)
        if self._y0 - self._x0 < -2 :
            # Count point of discontinuity
            discontX = 0.25 * np.log(0.25 * np.exp(4 * self._x0) / (1 / (self._y0 - self._x0 - 2) + 0.25))
            # Delete points that are too close to the discontinuty
            # point to avoid errors and too big values
            for i in range(self._N + 2):
                if np.abs(self._xArray[i] - discontX) < 0.001:
                    np.delete(self._xArray, [i])
        # Compute analytical y-values on fixed x-grid
        self._analyticalYArray = np.array([self._generalSolution(x, self._C) for x in self._xArray])

    def recompute(self, x0, y0, X, N):
        self._checkConstantExistance(x0, y0)
        super().recompute(x0, y0, X, N)
```

Also, it is possible to add some hint’s for the user of GUI by initializing the field “exception” in class “InitialValueProblem”:



C does not exist if
y0 - x0 = 2

Parameters

x0	0
y0	1
X	20
N	27

Conclusion:

As a result of the practicum, an application was written to demonstrate the performance of different numerical methods for approximation of ordinary differential equations solutions.