$$\begin{cases} y' = -\frac{y^2}{3} - \frac{2}{3x^2} \\ y_0 = 2 \\ x_0 = 1 \end{cases}$$

X = 5

Exact solution.

$$y' = -\frac{y^2}{3} - \frac{2}{3x^2} \ (\mathbf{1})$$

As we can see $x \neq 0$

Let $y(x) = -3v(x)$, then $y'(x) = -3v'(x)$
Substitute this in equation (1):

$$-3v' = -\frac{9v^2}{3} - \frac{2}{3x^2} \mid : -3$$

$$v' = v^2 + \frac{2}{9x^2} \ (\mathbf{2})$$

Let $v(x) = -\frac{u\prime(x)}{u(x)}$, then $v'(x) = -\frac{u''(x)}{u(x)} + \frac{u\prime(x)^2}{u(x)^2}$
Substitute this in equation (2):

$$-\frac{u''}{u} + \frac{u'^2}{u^2} = \frac{u'^2}{u^2} + \frac{2}{9x^2}$$

$$\frac{u''}{u} + \frac{2}{9x^2} = 0 \mid * 9\mathrm{u}x^2$$

$$9x^2u'' + 2u = 0 \ (\mathbf{3})$$

(3) is Euler-Cauchy equation. So, we assume that the solution is in the form

$$u(x) = x^a, \ u'(x) = ax^{a-1}, \ u''(x) = a(a-1)x^{a-2}$$

Substitute it to **(3)**

$$9x^2a(a-1)x^{a-2} + 2x^a = 0$$

$$9a(a-1)x^a + 2x^a = 0$$

$$x^a(9a^2 - 9a + 2) = 0$$

Now we need to solve $9a^2 - 9a + 2 = 0$:

$$9a^2 - 9a + 2 = 0$$

$$(3a-2)(3a-1) = 0$$

So, $a_1 = \frac{2}{3}$ and $a_2 = \frac{1}{3}$

Hence, $u(x) = c_1x^{\frac{1}{3}} + c_2x^{\frac{2}{3}}$.

$$v(x) = -\frac{u'(x)}{u(x)} = -\frac{c_1 + 2c_2 x^{\frac{1}{3}}}{3c_1 x + 3c_2 x^{\frac{4}{3}}} = -\frac{c_2\left(2x^{\frac{1}{3}} + \frac{c_1}{c_2}\right)}{c_2\left(\frac{3c_1 x}{c_2} + 3x^{\frac{4}{3}}\right)} = -\frac{2x^{\frac{1}{3}} + C}{x(3C + 3x^{\frac{1}{3}})}$$

$$y(x) = -3v(x) = \frac{2x^{\frac{1}{3}} + C}{x(C + x^{\frac{1}{3}})}$$

For given y_0 and x_0 we can calculate C:

$$C = \frac{x_0^{\frac{1}{3}}(y_0 x_0 - 2)}{(1 - y_0 x_0)}$$

So, $y_0 x_0 \neq 1$

in my case,

$$\begin{cases} y' = -\frac{y^2}{3} - \frac{2}{3x^2} \\ y_0 = 2 \\ x_0 = 1 \end{cases}$$

Substituting y_0 and x_0, we get that C = 0

Points of Discontinuity

As we saw while solving the differential equation:

$$x \neq 0$$
$$y_0 x_0 \neq 1$$
$$x^{\frac{1}{3}} \neq -C$$

Implementation Details.

Used Software.

Language: Python, matplotlib library is used for plotting
GUI Framework: Tkinter Library

Classes.

To start with, there are no abstract classes in Python. Nevertheless, there exists a library called ABC, and classes that inherit from it are considered to be abstract. Their abstract methods are denoted using @abstractmethod.

Below is an example of abstract class *Function* with abstract method *compute*

```
class Function(ABC):
    """An abstract class representing some f(x, y)"""

    @abstractmethod
    def compute(self, x, y):
        """if y is none then method computes y = g(x)
           otherwise, the method computes y' = f(x, y)"""
        pass
```

Class View.
Is one of the MVC classes. Used to display GUI window, its widgets and graphs of solutions for user.

Class Model.
Is one of the MVC classes. All the computational work is done here.

Class Controller.
Is one of the MVC classes. Used to read data from user and send the necessary information to Model and View classes, so they could perform needed actions.

Class Function.
An abstract class representing some function f(x, y).

Class MyFunction.
A class representing y' = f(x, y) of 10^{th} variant.

Class MySolutionFunction.
A class representing y = g(x) which is a solution of y'=f(x,y) of 10^{th} variant.

Class Grid.
A class representing a grid in which we need to find approximated solutions.

Class Solution.
An abstract class representing different solution methods (e.g Exact, Numerical) for a given function y' = f(x,y) and grid.

Class ExactSolution.
A class representing Exact Solution of a given function y' = f(x,y) for a given grid.

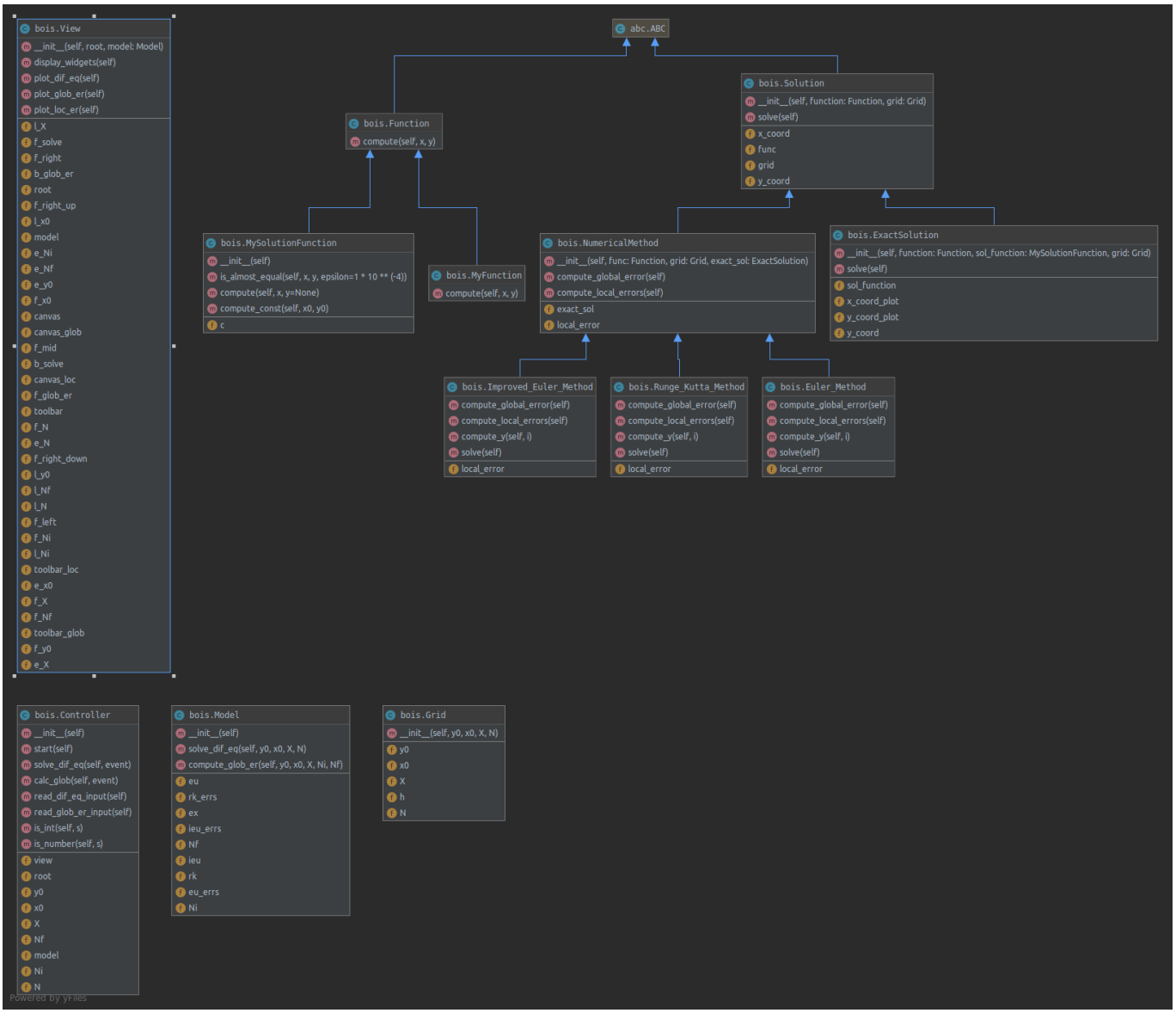
Class NumericalMethod.
An abstract class Numerical Solutions (e.g Euler's, Improved Euler's, Runge-Kutta's methods) for a given function y' = f(x,y) and grid.

Class Euler_Method.
A class representing Euler's method for a given function y'=f(x,y) and grid.

Class Improved_Euler_Method.
A class representing Improved Euler's method for a given function y'=f(x,y) and grid.

Class Runge_Kutta_Method.
A class representing Runge-Kutta's method for a given function y'=f(x,y) and grid.

Below you can see a UML diagram representing relations between Classes.



Implementations of Numerical Methods:

- **Euler's Method:**

```
def compute_y(self, i):
    """computes i-th y by Euler's method using (i-1)-th y and x"""
    if i == 0:
        return self.grid.y0
    else:
        return self.y_coord[i - 1] + self.grid.h * self.func.compute(self.x_coord[i - 1],
                                                                     self.y_coord[i - 1])
```

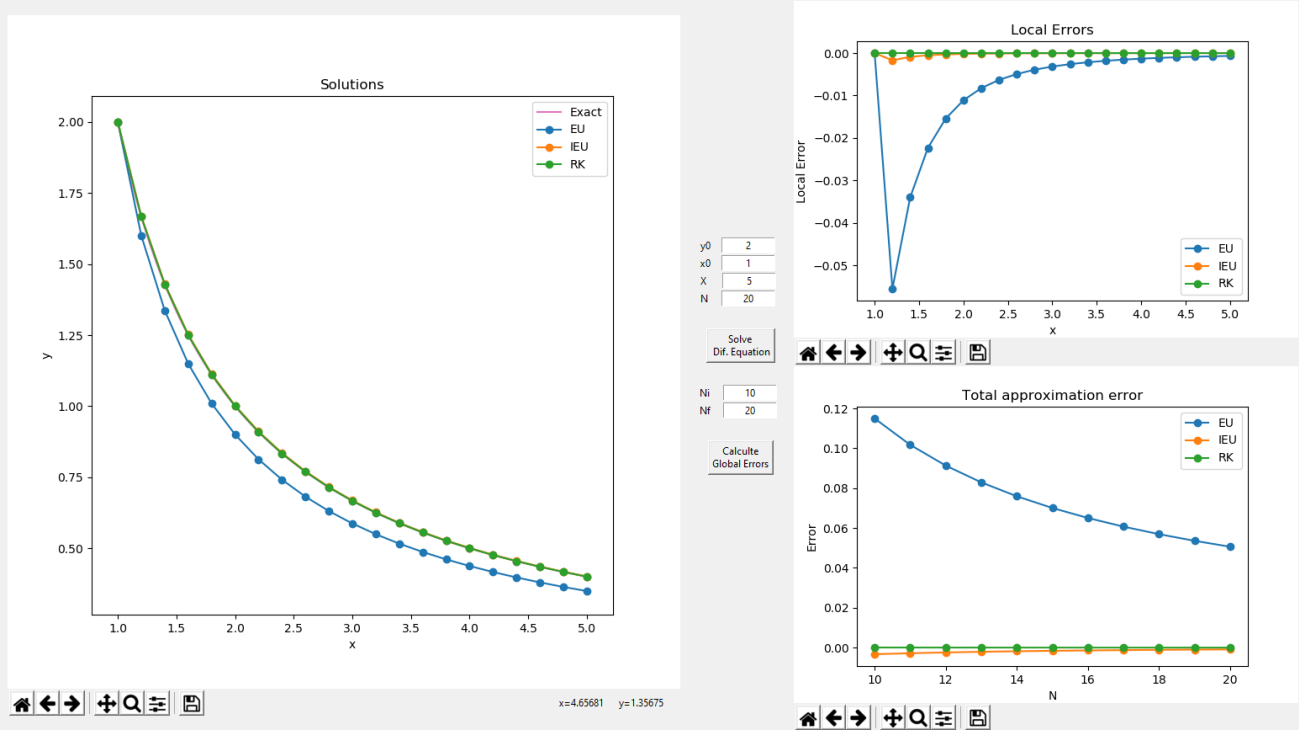
- **Improved Euler's Method:**

```
def compute_y(self, i):
    """computes i-th y by Improved Euler's method using (i-1)-th y and x"""
    if i == 0:
        return self.grid.y0
    else:
        k1 = self.func.compute(self.x_coord[i - 1], self.y_coord[i - 1])
        k2 = self.func.compute(self.x_coord[i - 1] + self.grid.h,
                               self.y_coord[i - 1] + self.grid.h * k1)
        return self.y_coord[i - 1] + (k1 + k2) * self.grid.h / 2
```

- **Runge-Kutta's Method:**

```
def compute_y(self, i):
    """computes i-th y by Runge-Kutta's method using (i-1)-th y and x"""
    if i == 0:
        return self.grid.y0
    else:
        k1 = self.func.compute(self.x_coord[i - 1], self.y_coord[i - 1])
        k2 = self.func.compute(self.x_coord[i - 1] + self.grid.h / 2,
                               self.y_coord[i - 1] + self.grid.h * k1 / 2)
        k3 = self.func.compute(self.x_coord[i - 1] + self.grid.h / 2,
                               self.y_coord[i - 1] + self.grid.h * k2 / 2)
        k4 = self.func.compute(self.x_coord[i - 1] + self.grid.h,
                               self.y_coord[i - 1] + self.grid.h * k3)
        return self.y_coord[i - 1] + self.grid.h * (k1 + 2 * k2 + 2 * k3 + k4) / 6
```

How to use GUI.



The application allows user to change the values of y0, x0, X and N.

User must make sure that:

- y0, x0, X are float
- X > x0
- N is Natural number greater than 0
- no points of discontinuities lie in [x0, X]

After entering valid values, user must press button “Solve Dif. Equation” in order to obtain the graph of original curve, charts of different approximations and their local errors.

Also, user may analyze how total approximation errors of used numerical methods change for different values of N.
For this, he/she needs to input initial N (Ni) and final N (Nf).

User must make sure that:

- Ni and Nf are Natural numbers greater than 0
- Ni < Nf

After entering valid values, user must press button “Calculate Global Errors” in order to obtain charts of total approximation errors of various methods for Ns in [Ni, Nf].

User may also use toolbar under graphs to inspect them in more details.

Analyses. Methods comparison.

The results of comparison of Numerical methods may be predicted by their global and local truncation errors complexities (written below from worst to best):

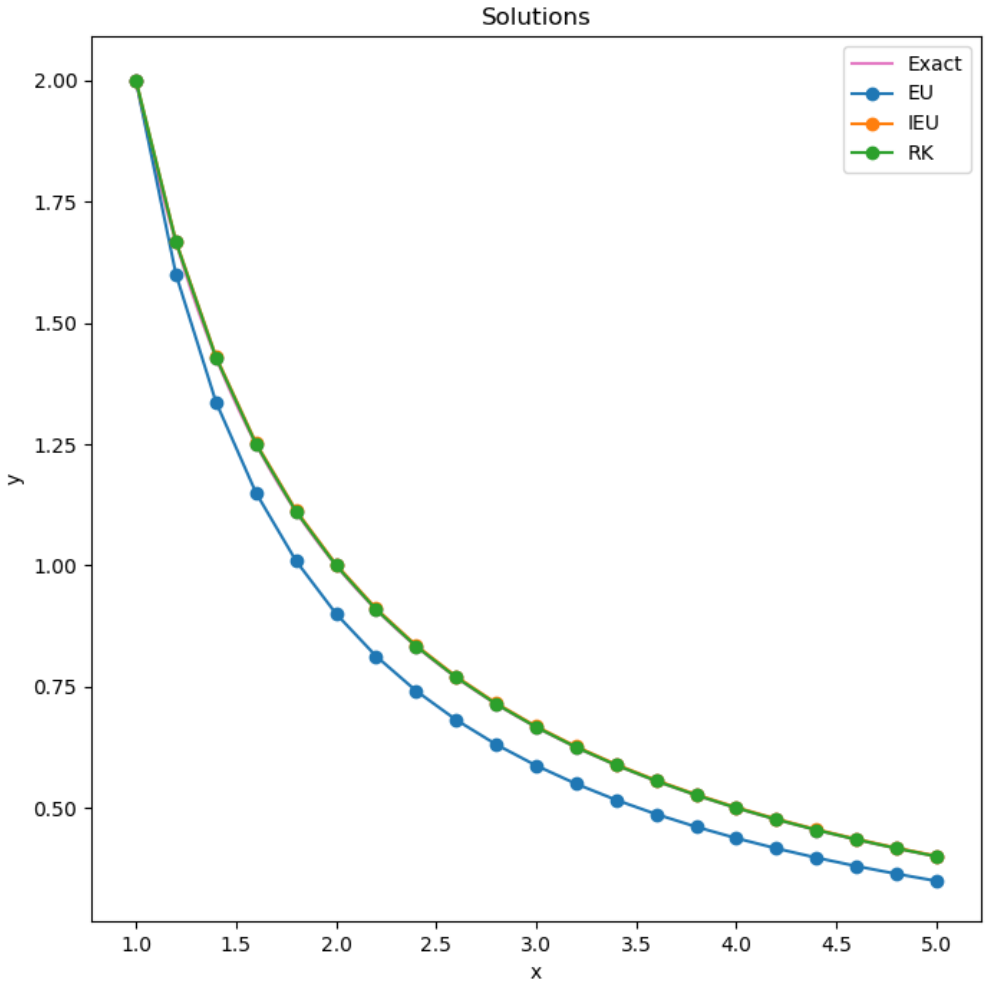
- Euler local: h^2
  Euler global: h

- Improved Euler local: h^3
  Improved Euler global: h^2

- Runge-Kutta local: h^5
  Runge-Kutta global: h^4

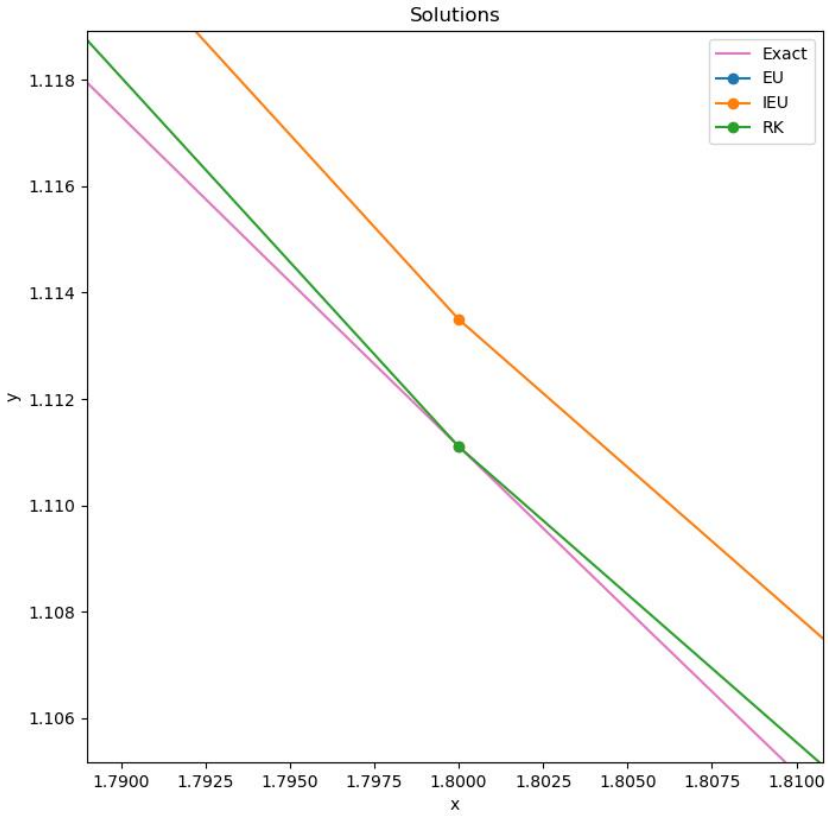
This statement is valid as h is usually chosen to be very small (it is a distance between 2 consecutive points).
In my example, x0 = 1, X = 5 and N = 20, so $h = \frac{X-x0}{N} = 0.2$

Now, let's see actual approximations to check if our assumption is right.



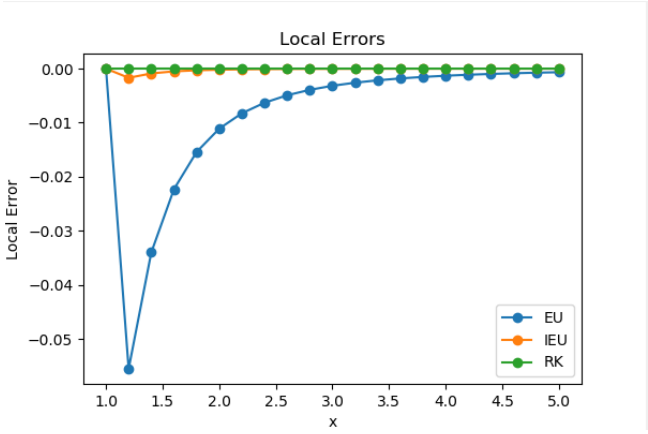
The charts of Exact Solution, Improved Euler's Method and Runge-Kutta's Method are so close to each other that at first glance you can't even tell the difference.

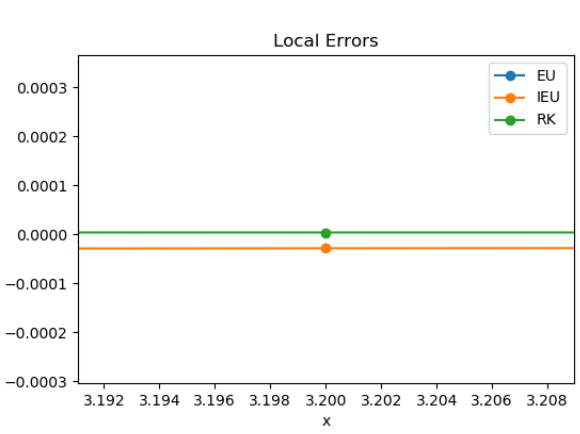
Let’s zoom in a bit.



Now, charts of those methods are clearly distinguishable. You can see how Runge-Kutta’s Method approximation is close to the original graph

Below you can see the plots of local errors (zoomed in at the left side).





As a result, our assumption was right: Runge-Kutta's Method showed itself as the best of the used approximations and Euler's method showed itself as the worst one.

Analyses. Total approximation error.

Now, let's see how the global error at the last point X depends on the number of points N in interval [x0, X].

It's not hard to guess that the more points we have, the less the error is going to be as greater N means smaller h, and global truncation errors are proportional to h (remember, that h is usually very small)

Indeed, as we can see, each method's total approximation error decreases as N increases.