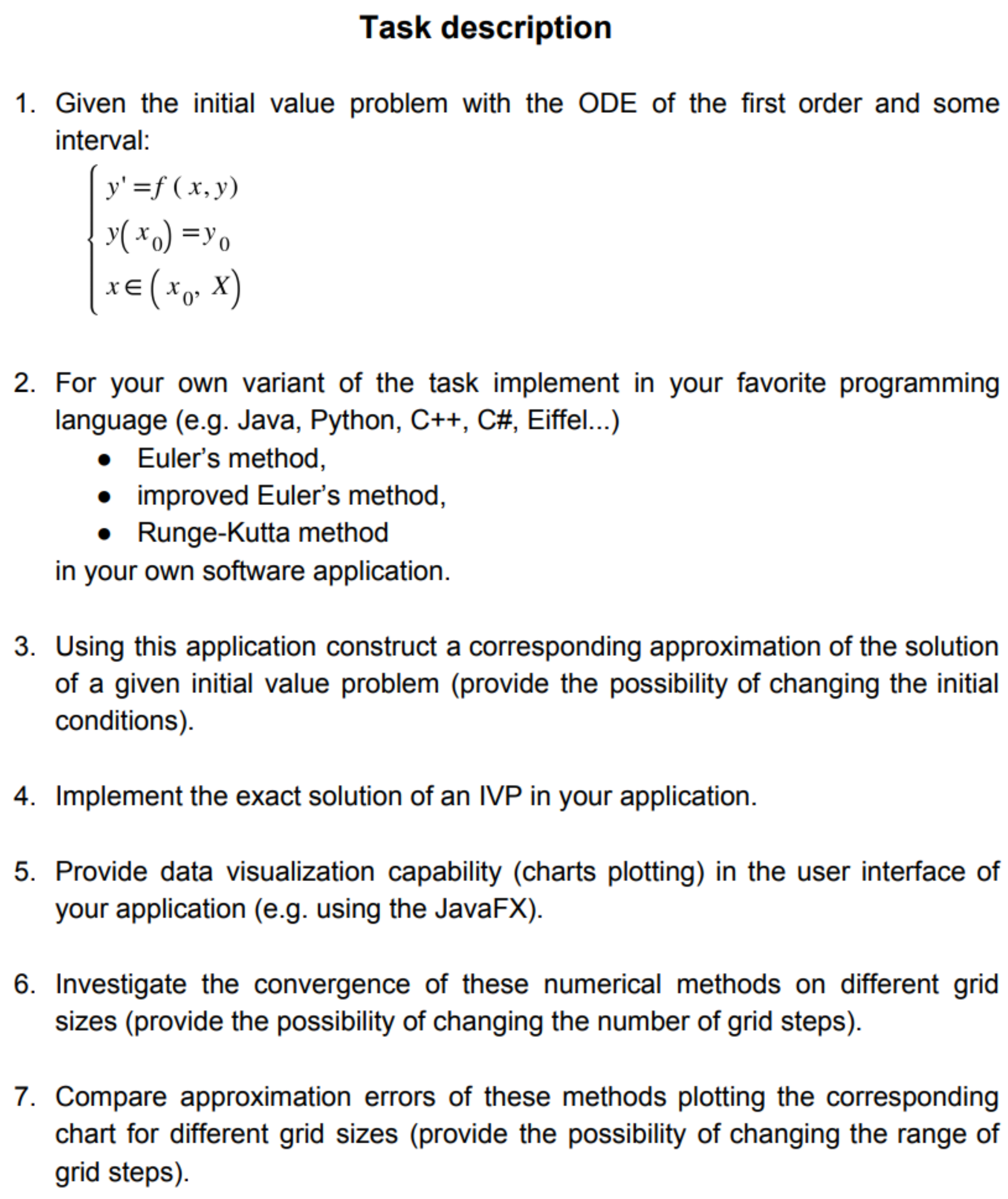
Polina Romanenkova BS19-05
Variant №1 DE

Part 0
The idea behind the assignment(Introduction)

The idea of this assignment is to inspect, implement and visualize different numerical methods to solve a differential equation by using GUI and user's inputs.
Definition of the task:



Task description

1. Given the initial value problem with the ODE of the first order and some interval:
$$\begin{cases} y' = f(x, y) \\ y(x_0) = y_0 \\ x \in (x_0, X) \end{cases}$$

2. For your own variant of the task implement in your favorite programming language (e.g. Java, Python, C++, C#, Eiffel...)
   - Euler's method,
   - improved Euler's method,
   - Runge-Kutta method

   in your own software application.

3. Using this application construct a corresponding approximation of the solution of a given initial value problem (provide the possibility of changing the initial conditions).

4. Implement the exact solution of an IVP in your application.

5. Provide data visualization capability (charts plotting) in the user interface of your application (e.g. using the JavaFX).

6. Investigate the convergence of these numerical methods on different grid sizes (provide the possibility of changing the number of grid steps).

7. Compare approximation errors of these methods plotting the corresponding chart for different grid sizes (provide the possibility of changing the range of grid steps).

Polina Romanenkova BS19-05
Variant №1 DE

Part 1
Exact solution for the problem

$$y' = 1 + \frac{2y}{x}$$
$$y(1) = 2$$
$$x \in (x0, X)$$

Firstly, we will subtract $2y/x$ from both sides

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

Now a brilliant idea will be to transform this equation somehow so that we will be able to apply product differential rule, but in reverse order. The solution for this can be to divide everything by x2 because 2/x is a differential of 1/x2, so on the left we will have products

As well we can express y' as $\frac{dy}{dx}$

$$\frac{\frac{dy}{dx}}{x^2} + \frac{2y}{x^3} = \frac{1}{x^2}$$
$$\frac{\frac{dy}{dx}}{x^2} + \frac{1}{x^2} \cdot x' = \frac{1}{x^2}$$

At this step we can apply reverse product rule

$$\frac{y}{x^2} = \frac{1}{x^2}$$

.. and integrate with respect to x...

$$\int \frac{y}{x^2} \cdot dx = \int \frac{1}{x^2} \cdot dx$$
$$\frac{y}{x^2} = -\frac{1}{x} + C_1$$
$$y = -x + C_1 \cdot x^2$$

Now by initial condition we can substitute and find C_1

$$2 = -1 + C_1$$
$$C_1 = 3$$

And by returning C_1 back to the solution we get:

Answer: $y = -x + 3 \cdot x^2$

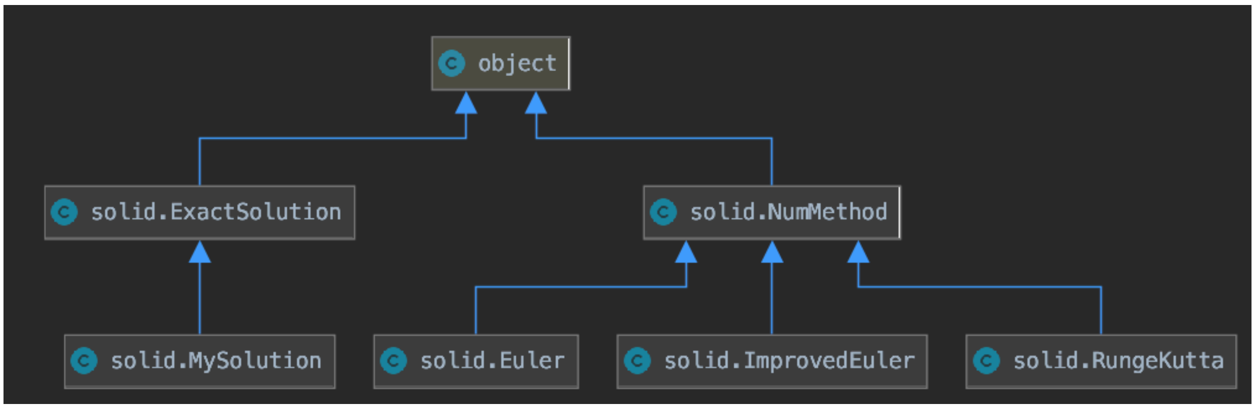
Polina Romanenkova BS19-05
Variant №1 DE

Part 2
Implementation

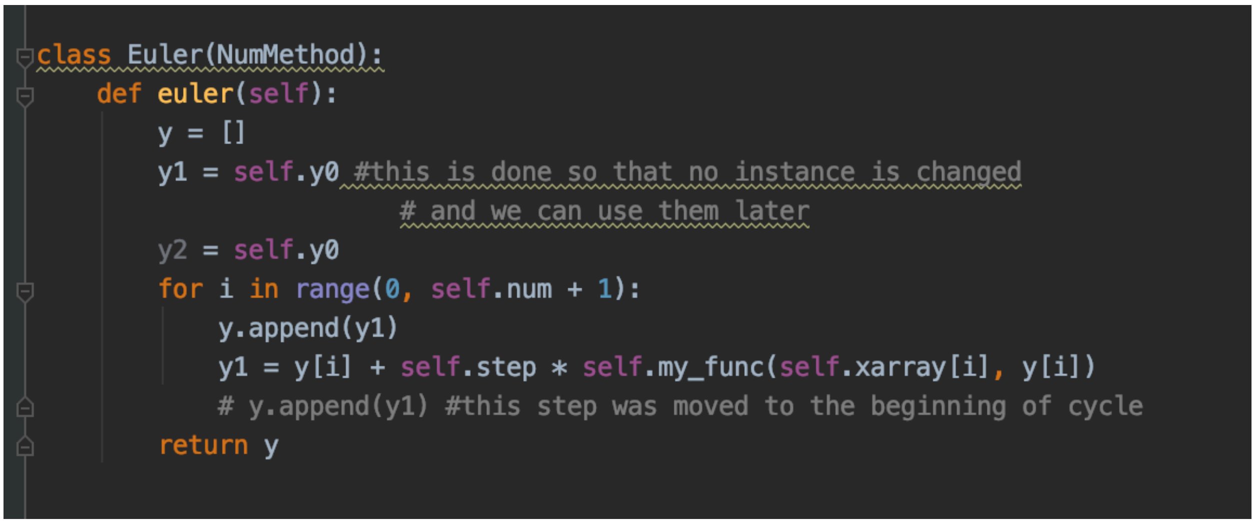
Tools used to accomplish these tasks:

- Python as a main programming language
- PyCharm as an IDE
- Tkinter as GUI Interface Accomplisher
- Matplotlib as a library for plotting
- Numpy as a helper for graph plotting (to do x-arrays for plots)

The code is based on OOP principles. I have all my classes containing numerical methods in file solid(by the name of OOP principles I tried to follow with).

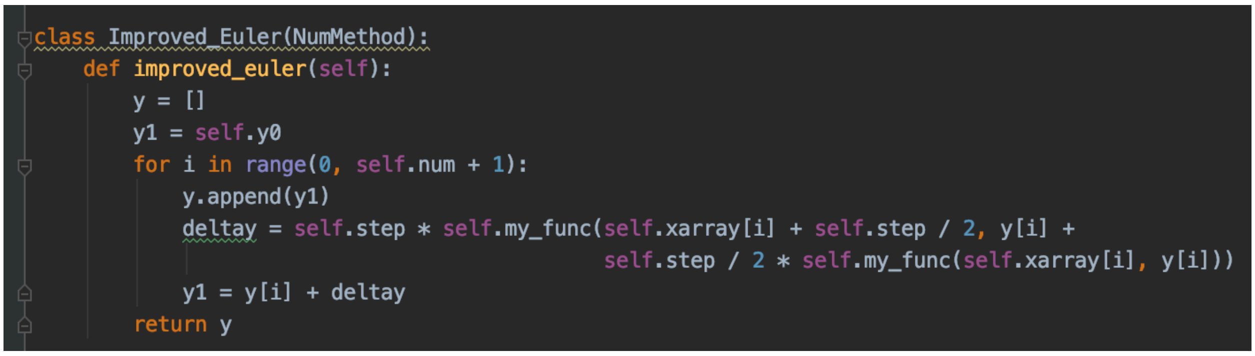


Euler's method:



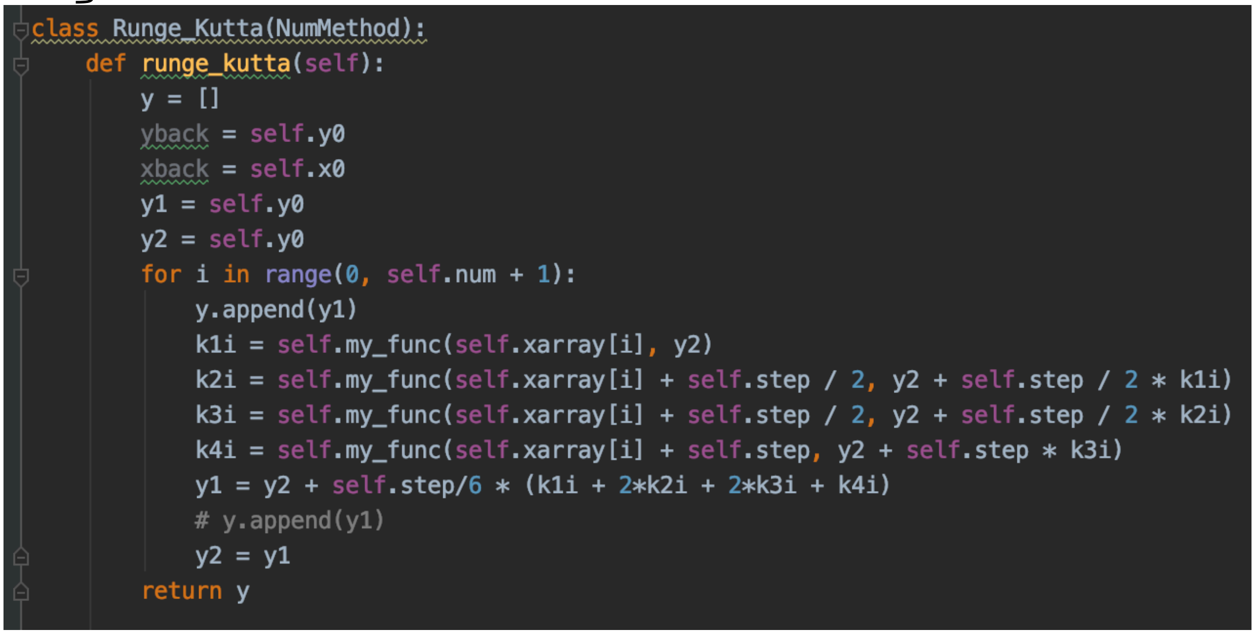
```python
class Euler(NumMethod):
    def euler(self):
        y = []
        y1 = self.y0 #this is done so that no instance is changed
                        # and we can use them later
        y2 = self.y0
        for i in range(0, self.num + 1):
            y.append(y1)
            y1 = y[i] + self.step * self.my_func(self.xarray[i], y[i])
            # y.append(y1) #this step was moved to the beginning of cycle
        return y
```

Improved Euler's method:



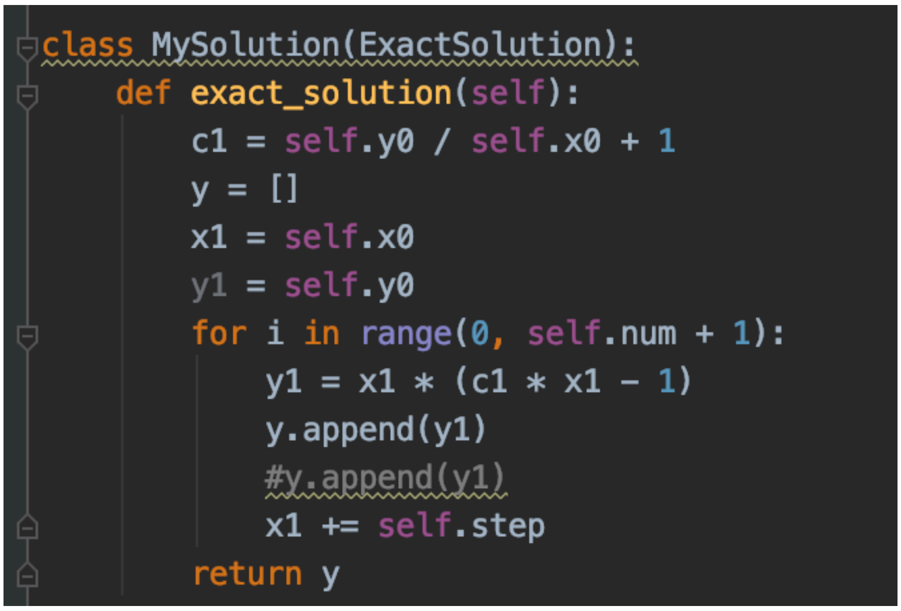
```python
class Improved_Euler(NumMethod):
    def improved_euler(self):
        y = []
        y1 = self.y0
        for i in range(0, self.num + 1):
            y.append(y1)
            deltay = self.step * self.my_func(self.xarray[i] + self.step / 2, y[i] +
                                              self.step / 2 * self.my_func(self.xarray[i], y[i]))
            y1 = y[i] + deltay
        return y
```

Runge-Kutta's method:



```python
class Runge_Kutta(NumMethod):
    def runge_kutta(self):
        y = []
        yback = self.y0
        xback = self.x0
        y1 = self.y0
        y2 = self.y0
        for i in range(0, self.num + 1):
            y.append(y1)
            k1i = self.my_func(self.xarray[i], y2)
            k2i = self.my_func(self.xarray[i] + self.step / 2, y2 + self.step / 2 * k1i)
            k3i = self.my_func(self.xarray[i] + self.step / 2, y2 + self.step / 2 * k2i)
            k4i = self.my_func(self.xarray[i] + self.step, y2 + self.step * k3i)
            y1 = y2 + self.step/6 * (k1i + 2*k2i + 2*k3i + k4i)
            # y.append(y1)
            y2 = y1
        return y
```

Exact Solution:



```python
class MySolution(ExactSolution):
    def exact_solution(self):
        c1 = self.y0 / self.x0 + 1
        y = []
        x1 = self.x0
        y1 = self.y0
        for i in range(0, self.num + 1):
            y1 = x1 * (c1 * x1 - 1)
            y.append(y1)
            #y.append(y1)
            x1 += self.step
        return y
```

Classes illustrating numerical methods are inherited from NumericalMethod class, which sets constructors for these child classes.

Polina Romanenkova BS19-05
Variant №1 DE

Part 3
Evaluating the results

It is worth noting that it is necessary to provide a check for the validity of the input. A thorough analysis revealed possible problems:

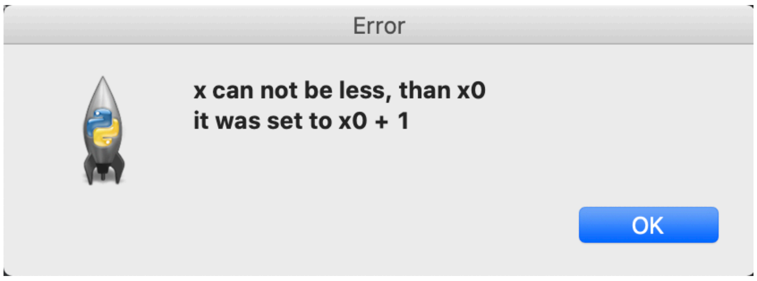
- The value of x can't be less than x_0
- The value of number of steps can't be less than 1;
- The value of x1 can't be greater than x2 for plotting the grid.

After each reading of the values from the fields, they are checked for validity. If an error occurs, the exception "Error" and the error window is displayed. When you click the "ok" button, the window with default values instead of wrong ones will be displayed again.

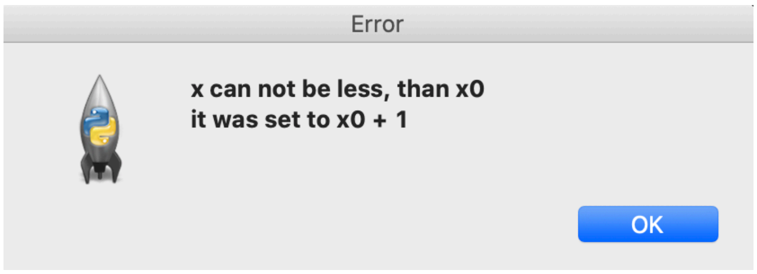
Polina Romanenkova BS19-05
Variant №1

Part 4
Possible improvements and Additional images

Errors' images:

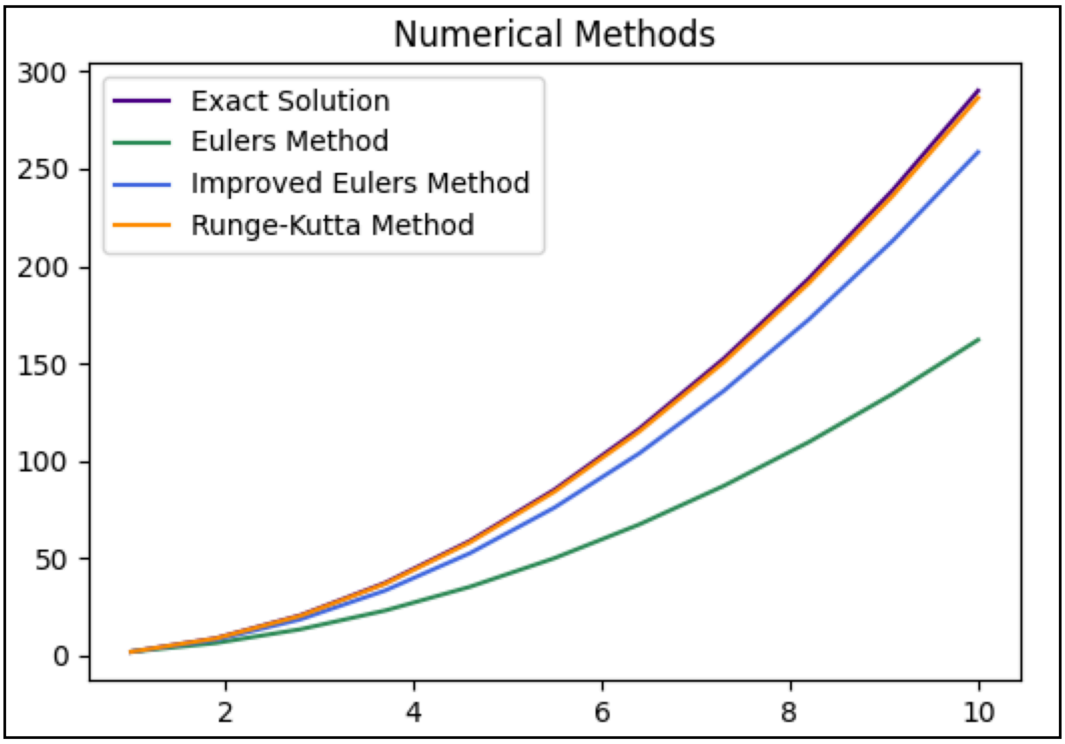

Error

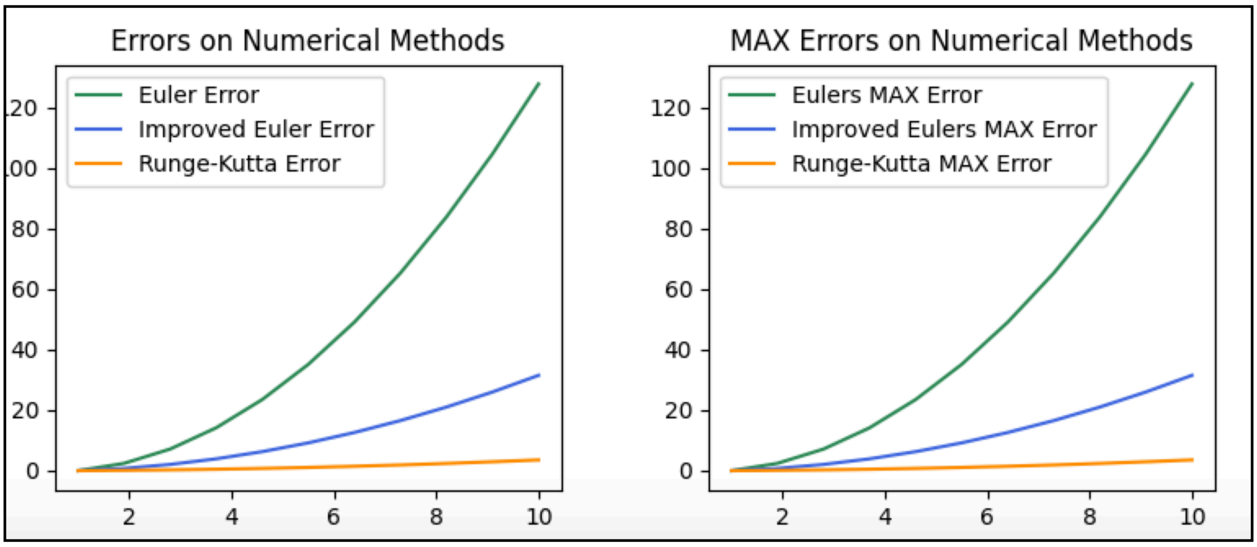
x can not be less, than x0
it was set to x0 + 1


Error

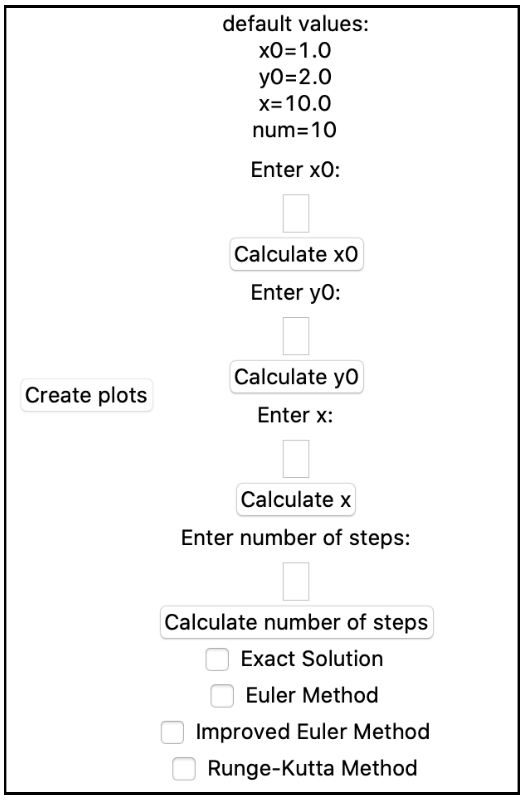
x can not be less, than x0
it was set to x0 + 1

Plots for Default IVP:

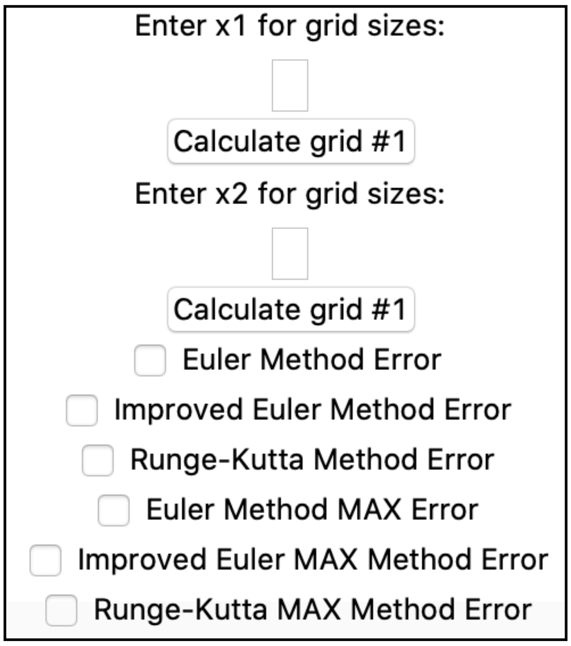




GUI User-dealing interface:



default values:
x0=1.0
y0=2.0
x=10.0
num=10



Definition of buttons:
On the right we can enter x0, y0, x and number of steps. By pressing 'Calculate' the string from user goes into data_storage for the plot

Checkbuttons evaluate, whether a user wants a particular graph line to be plotted

By pressing 'Create Plots' user loops a window and the new creating one goes specified by user's previous inputs.