DIFFERENTIAL EQUATIONS

COMPUTATIONAL PRACTICUM

Lev Svalov BS18-05

https://github.com/LeoSvalov/Fall2019_DE_practicum

Variant #19:

$f(x,y) = 2x + y - 3$

$y_0 = 1$

$x_0 = 1$

$X = 7$

Innopolis University

2019

1. Exact solution of the DE.

$y' = 2x + y - 3$, $y(1) = 1$, $X = 7$

$y' - y = 2x - 3$

1) *Complementary part*

   $y_c' - y_c = 0$; $dy_c/y_c = dx$; —> $\ln(y_c) = x$;

   —> $\underline{y_c = e^x}$

2) *Variation of parameter*

   $u' = (2x - 3) / e^x = (2x - 3) * e^{(-x)}$

   $u = \int(2x - 3) * e^{(-x)}\, dx =$

   // apply integration by parts

   $= -(2x - 3) * e^{(-x)} - \int -2e^{(-x)}\, dx =$

   $= -2e^{(-x)} * x + e^{(-x)} + C = \underline{e^{(-x)} * (1 - 2x) + C}$,

   where C - constant

3) $y = y_c * u = e^x * (e^{(-x)} * (1 - 2x) + C) = \underline{1 - 2x + C*e^{(x)}}$

   solution — $y = 1 - 2x + C*e^{(x)}$

Answer for I.V.P. y(1) = 1:

from solution we derive formula for constant C:

$C = (y + 2x - 1) / e^x$

and put $y_0 = 1$, $x_0 = 1$ —>

$C = (1 + 2 - 1) / e = 2 / e \approx 0{,}736$

Should be noticed that there are **no** points of discontinuity.

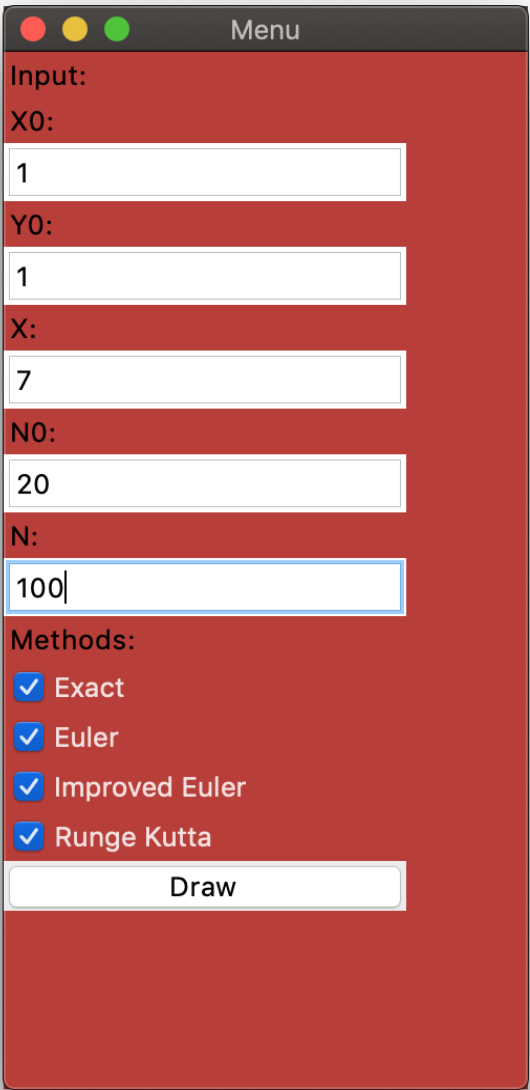
2. Implementation of numerical methods and the GUI.

1. Implementation

Language: Python

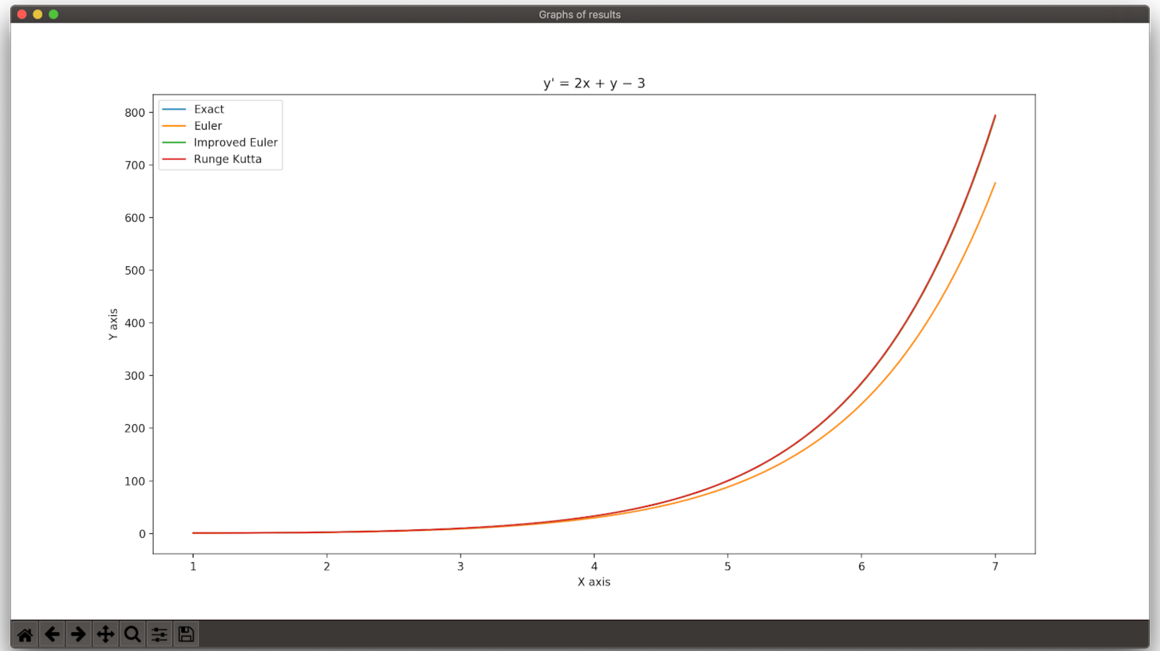
Used libraries: tkinter, matplotlib, numpy, math

The graphical interface looks the following:

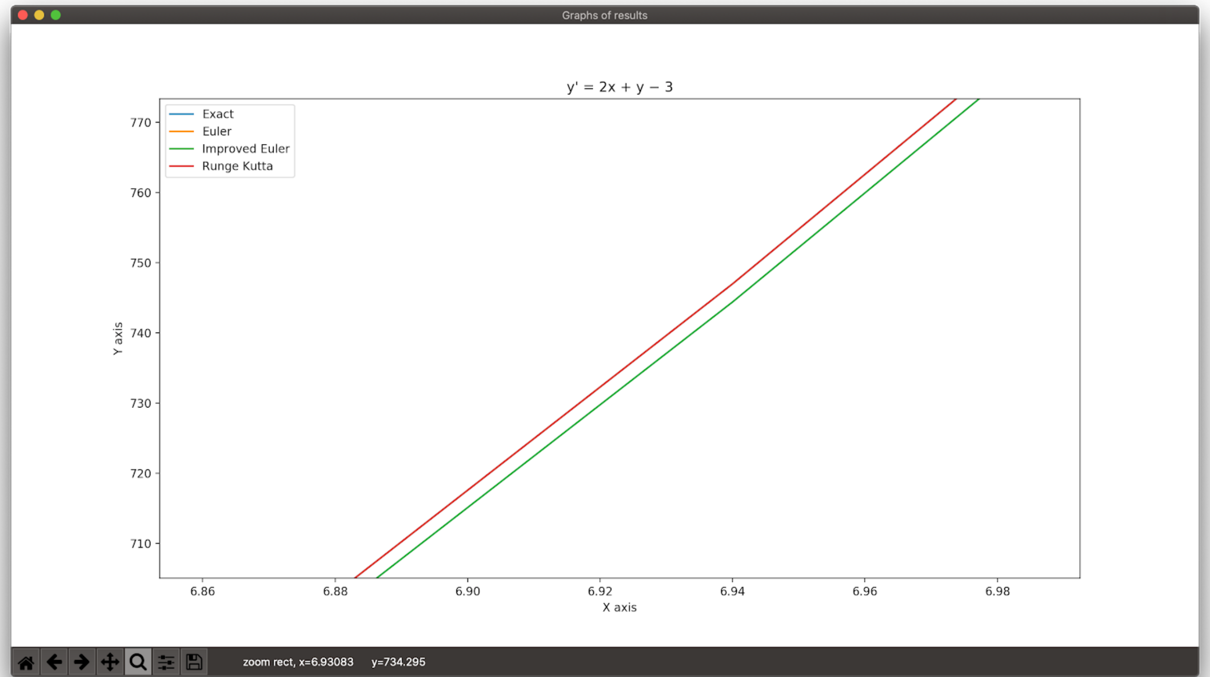


2. **The approximation of the solution of a given initial value problem: *(x0 = 1, y0 = 1, X = 7):***
   N0 = 20
   N = 100

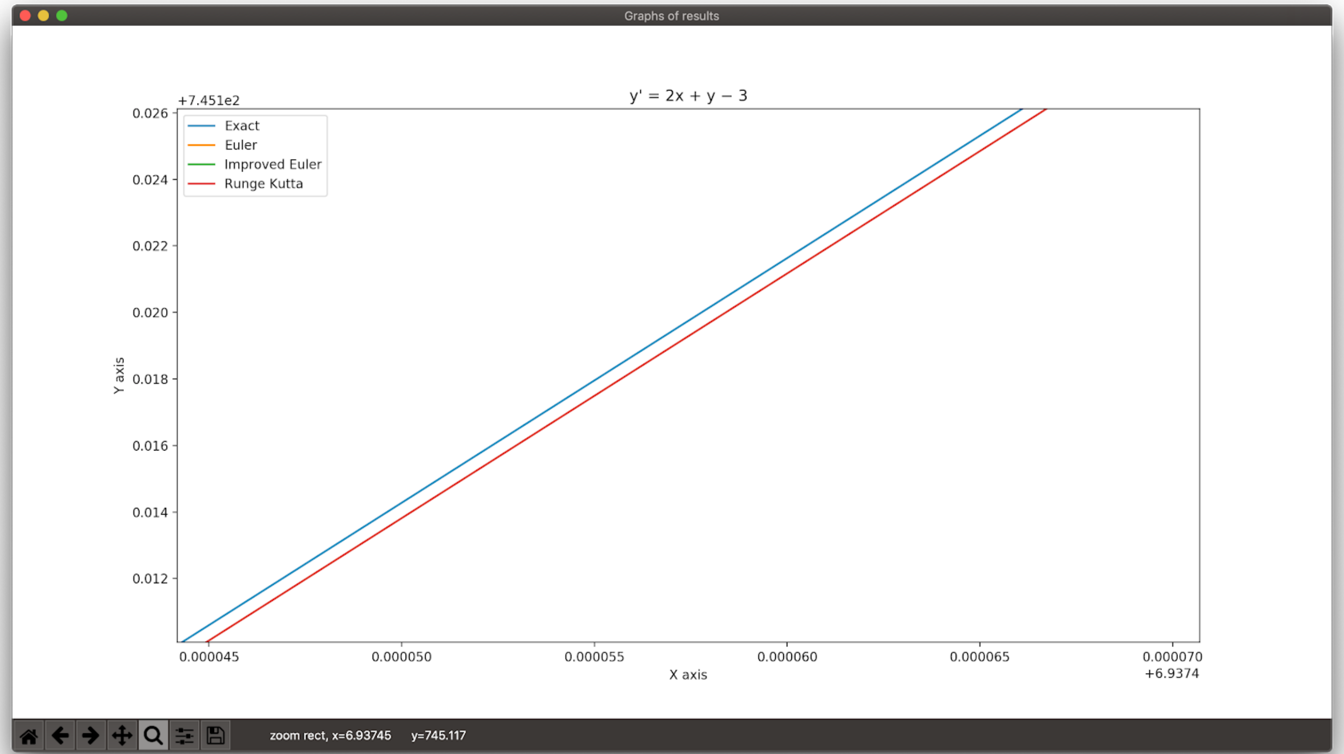
a) **Graphs of results**:



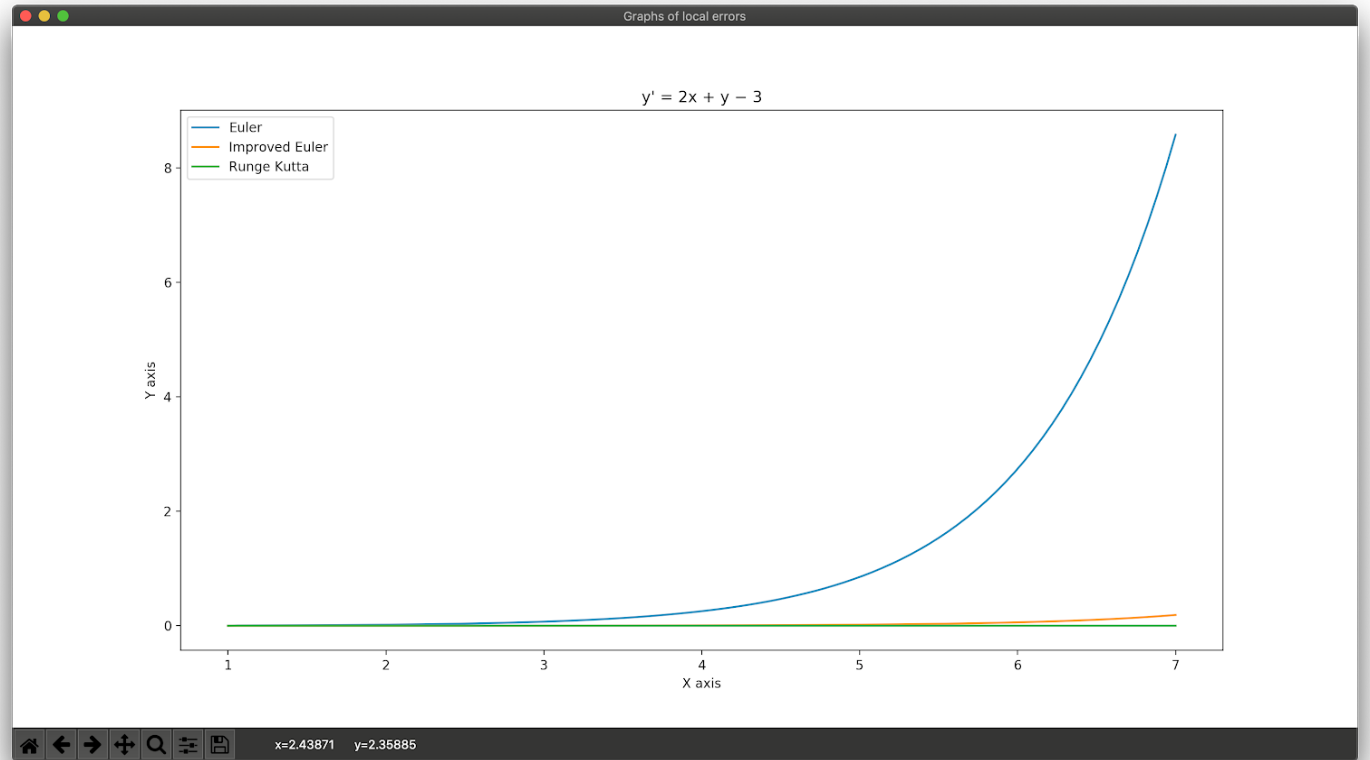
because of simplicity of the given DE, it seems that there are only 2 graphs, but if we zoom we will see that there are 4, just Exact, Imp.Euler and Runge Kutta are close to each other:



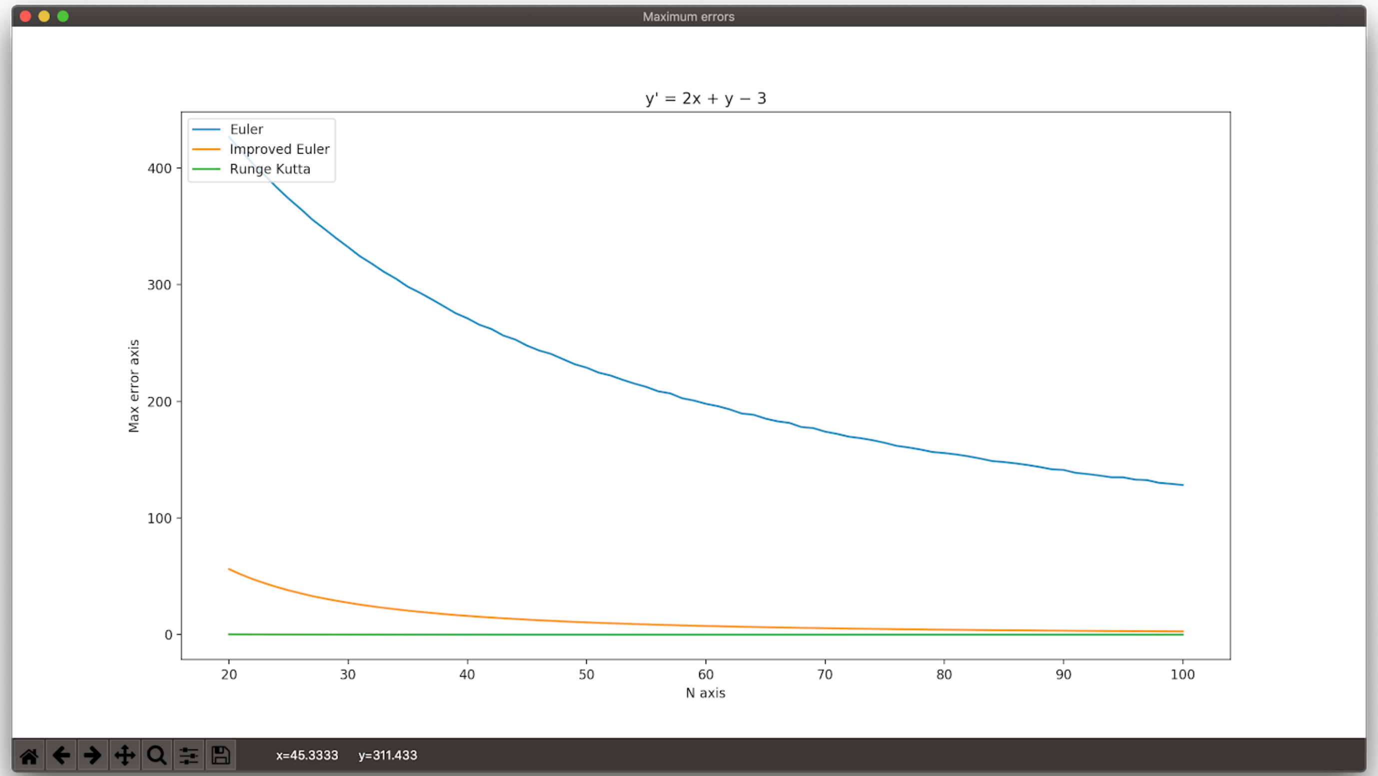
Actually, Exact and Runge Kutta are extremely close to each other



b) **Graphs of local errors:**



c) **Graphs of global errors**



Analysis Of The Results

The chart of results shows the graph of the exact solution and 3 numerical solutions. It is easily seen that the **Euler** method is the worst approach, **Runge-Kutta** is the best one and **Improved Euler** is in between. More precisely we make graphs (by increasing N in the graphical interface), graphs becomes closer and the **Runge-Kutta** graph and **Exact** graph almost coincides with each other.

The chart of local errors shows the graph of local errors of all 3 numerical methods. The **Euler** method has the biggest one, **Improved Euler** also has error when x becomes bigger, and **Runge-Kutta** approach almost does not have an error with respect to the original exact values of the DE.

The chart of maximum errors shows the graph of maximum errors of all 3 numerical methods. The **Euler** method has the biggest error with respect to exact values. We also can see that in the small Ns there is a sizeable difference between the numerical methods. But, for large N, like 100, maximum error for **Improved Euler** and **Runge Kutta** methods is almost the same.

Analysis of the code.
Following Object-Oriented-Programming style.

My version of the implementation of the computation practicum has the following logical points:
My code is logically divided by 3 files:

- **calculations.py** - *It contains all mathematical calculations that are related to getting the approximation of the particular method.*
- ***methods.py*** *- There are declarations of errors and all methods (as classes) and all used functions(like, get result of method), but without calculations, because it is in other file for better understanding.*
- ***main.py -*** *It contains the implementation of the interface and collecting all data of methods for putting into graphs.*

I also check the input data for the validity, in class *Interface,* it checks that input has to be a number and at least one method to draw should be selected.
I use an abstraction approach, for computing results of the numerical methods, I use function that calculates values according to the formula of the particular numerical method.
All functions I use are associated to the particular class. You can see the UML diagram with all classes below:

UML diagram
Computation Practicum Report
Differential Equations
Lev Svalov BS18-05