Differential Equations
Computational practicum

Shohjahon Khamrakulov
BS20-03
Github: *https://github.com/ShohKhan-dev/DE_Computational_Practicum*

1. Analytical solution

1.1 Initial system

$$\begin{cases} y' = 5 - x^2 - y^2 + 2xy \\ y(0) = 1 \\ x \epsilon (0, 20) \end{cases}$$

2 Finding General solution

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

2.1 Solving

Using Riccati equation:

$$y' + a(x)y + b(x)y^2 = c(x)$$

where

$$a(x) = -2,$$
$$b(x) = 1,$$
$$c(x) = 5 - x^2$$

Private solution in the form: $y = Ax + B$

Substituting to original equation:

$$(Ax + b)^2 - 2x(Ax + B) + A = 5 - x^2 \quad \rightarrow \quad \begin{pmatrix} A = 1 \\ B = 2 \end{pmatrix}$$

We will get: $y = x + 2 \quad \Rightarrow \quad y = x + u + 2 \quad \Rightarrow \quad y' = u' + 1$

By substituting back

$$-x^2 + u' + u^2 + 4u + 5 = 5 - x^2 \quad \Rightarrow \quad u' + u^2 + 4u = 0 \quad \Rightarrow$$
$$\Rightarrow \quad u' = -u^2 - 4u$$

Derivative form: $u'(x) = \dfrac{du}{dx}$

$$\frac{du}{dx} = -u^2 - 4u$$

Multiply by differential dx: $du = (-u^2 - 4u)dx$

Divide by $-u^2 - 4u$: $-\dfrac{du}{u^2 + 4u} = dx$

Now equation became separable, therefore integrate both sides:

$$-\int \frac{du}{u^2 + 4u} = \int dx$$

By calculating integrals we get:

$$\frac{ln(u+4)}{4} - \frac{ln(u)}{4} = x + C$$

By rising both sides by power of e:

$$\sqrt[4]{\frac{u+4}{u}} = e^{x+C}$$

Substituting u back to $u = y - x - 2$:

$$\sqrt[4]{\frac{y-x+2}{y-x-2}} = Ce^{x}$$

Finding ODE solution:

$$y = \frac{4}{Ce^{4x}-1} + x + 2$$

By applying initial values: $y(0) = 1$ we can find C

$$1 = \frac{4}{C-1} + 2 \quad \Rightarrow \quad C = -3$$

Exact form of equation and final answer is:

$$y = \frac{4}{-3e^{4x}-1} + x + 2$$

2. Method Functions analysis

2.1 Euler Method

```python
def euler_method(x0, y0, b, n):

    h, x = cal_steps(x0, b, n)

    y = [y0]
    for i in range(1, n):

        y.append(round(y[i - 1] + h * f(x[i - 1], y[i - 1]), 5))

    euler = [x, y]

    return euler
```

2.2 Improved Euler Method

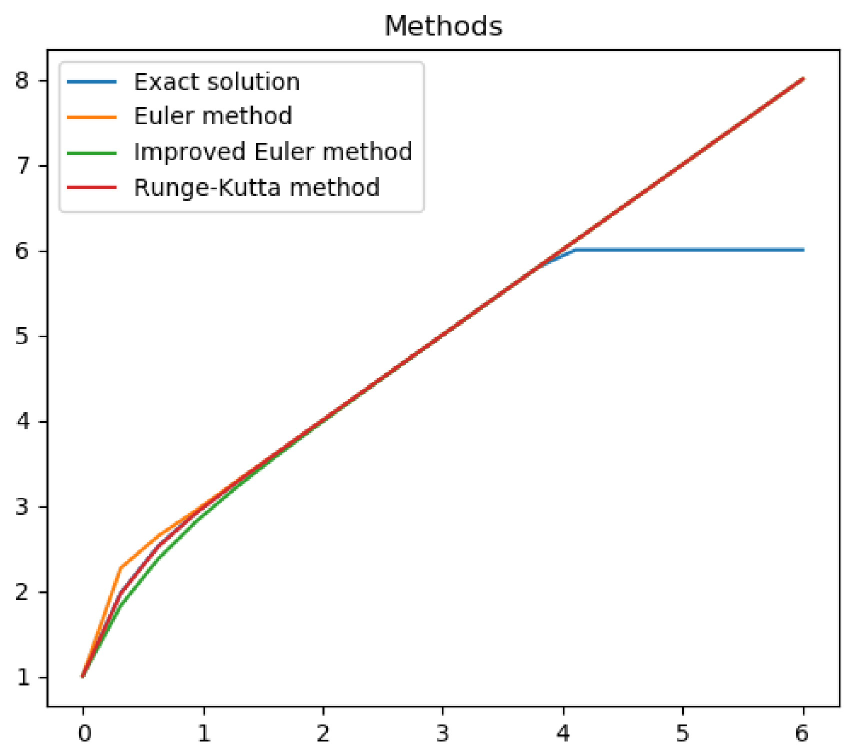
```python
def improved_euler_method(x0, y0, b, n):

    h, x = cal_steps(x0, b, n)

    y = [y0]
    for i in range(1, n):

        k1 = f(x[i - 1], y[i - 1])
        k2 = f(x[i - 1] + h, y[i - 1] + h * k1)

        y.append(round(y[i - 1] + (h / 2) * (k1 + k2), 5))

    improved = [x, y]


    return improved
```

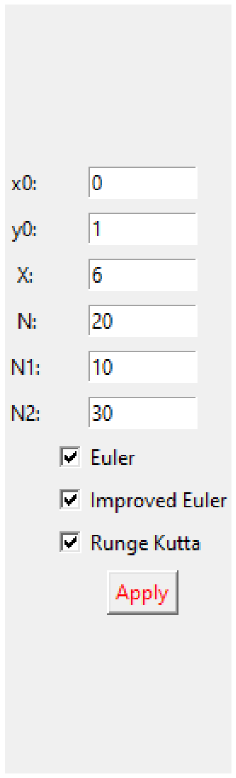
2.3 Runge-Kutta Method

```
def runge_kutta_method(x0, y0, b, n):

    h, x = cal_steps(x0, b, n)

    y = [y0]
    for i in range(1, n):
        k1 = f(x[i - 1], y[i - 1])
        k2 = f(x[i - 1] + h / 2, y[i - 1] + (h / 2) * k1)
        k3 = f(x[i - 1] + h / 2, y[i - 1] + (h / 2) * k2)
        k4 = f(x[i - 1] + h, y[i - 1] + h * k3)
        y.append(round(y[i - 1] + (h / 6) * (k1 + 2 * k2 + 2 * k3 + k4),
5))

    runge_kutta = [x, y]


    return runge_kutta
```

All method functions above takes x0, y0, X(b) and n (number of steps) and calculates y for each method based on x. The return value is 2 arrays with x values and y values respectively.

2.5 Graph visualization





x0:	0
y0:	1
X:	6
N:	20
N1:	10
N2:	30

✓ Euler

✓ Improved Euler

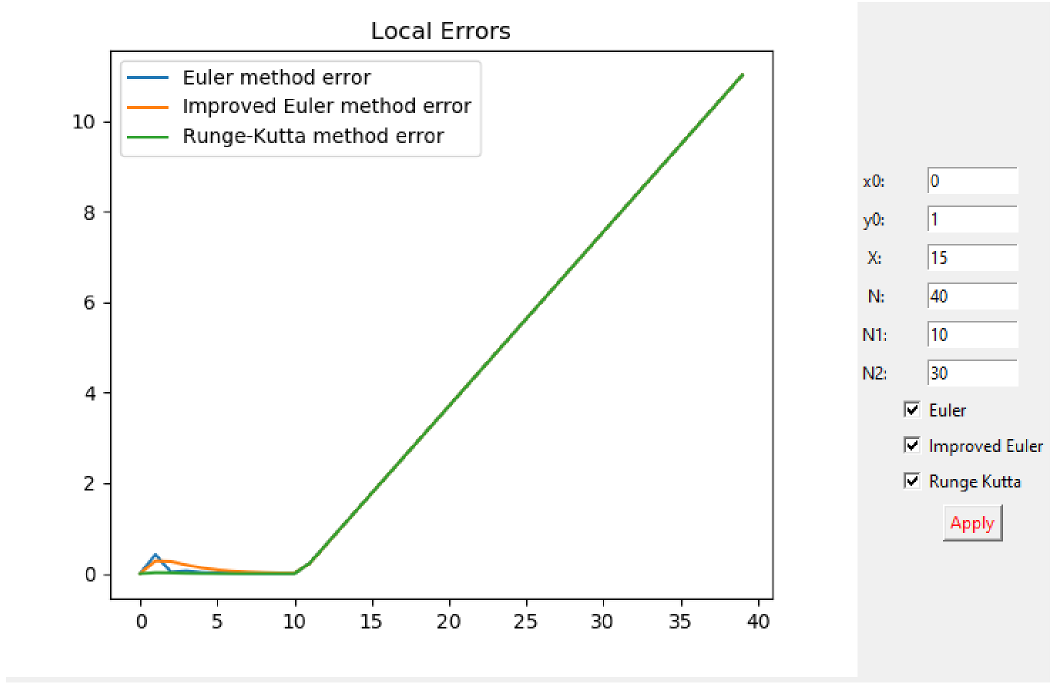
✓ Runge Kutta

It can be easily seen that all methods are almost the same as the exact solution. But when X reaches to its limit which is set to 6, it becomes stable but other methods continues

Because of this, errors in each method gradually increase.

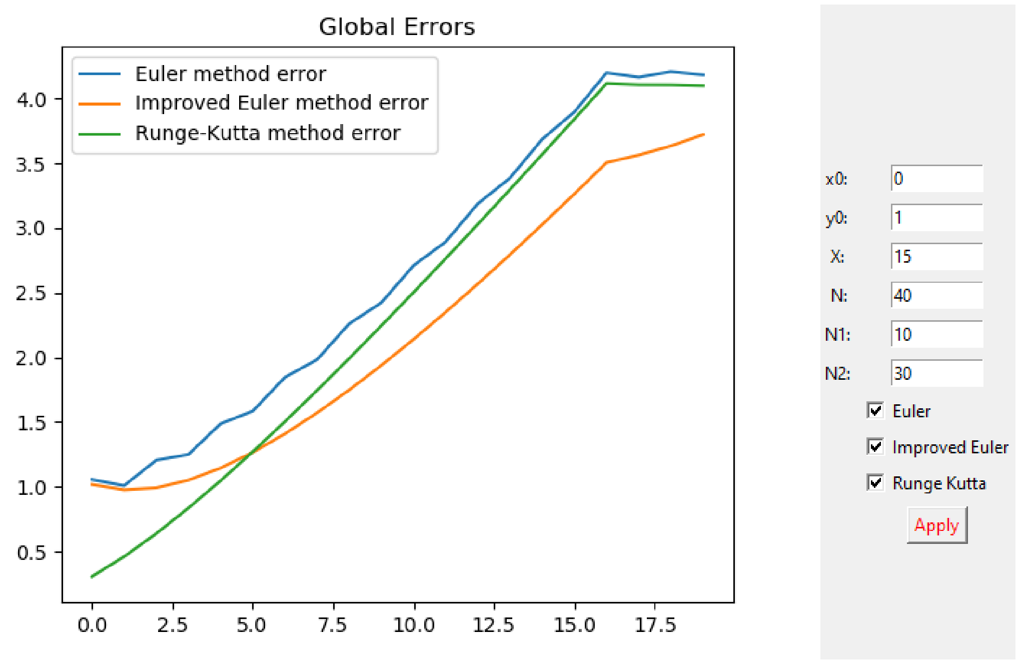
3. Error Functions

3.1 Local Errors



x0: 0
y0: 1
X: 15
N: 40
N1: 10
N2: 30
[✓] Euler
[✓] Improved Euler
[✓] Runge Kutta

3.2 Global Errors



x0: 0
y0: 1
X: 15
N: 40
N1: 10
N2: 30
[✓] Euler
[✓] Improved Euler
[✓] Runge Kutta

As you can see if we increase N, error is also increasing. The number of steps is 40 and between intervals (10, 30) Global Errors went to 4.

4. Implementation

This application developed as a Desktop software using Tkinter framework and Python as a programming language.
Consists of two .py files:
main.py for main operations such as user interface and Drawing graphs.
While calculate.py is for calculations operations for methods and errors.

4.1 Classes

Program consists of only two classes: PlotGraph and TkMenu.
PlotGraph class contains all graph operations with methods with no returns.

plotMethods() - to plot graphs for all methods. Gets exact, euler, improved, runge_kutta and checks parameters in list type.

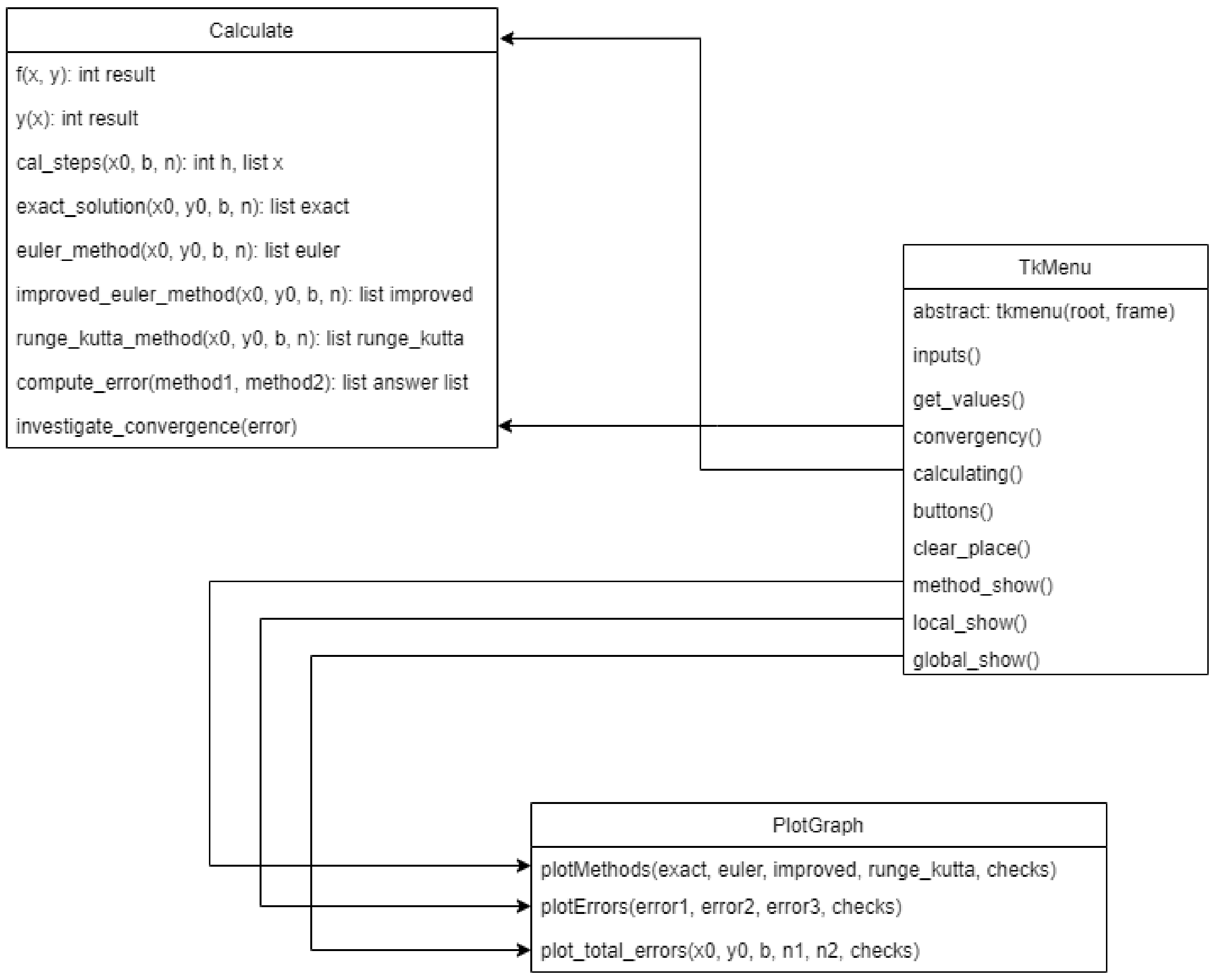
plotErrors() - to plot local errors. Gets error1, error2, error3, checks parameters in list type.

plot_total_errors() - to plot global errors. Gets x0, y0, b, n1, n2 integer parameters and one checks list type parameters.

Checks list parameters consist of 3 integers that specify checkboxes whether to plot or not to plot methods.

TkMenu class handles all implementations of the user interface.
methods divided into for each buttons, inputs, checkboxes and some configurations.

4.2 UML Diagram


Calculate
f(x, y): int result
y(x): int result
cal_steps(x0, b, n): int h, list x
exact_solution(x0, y0, b, n): list exact
euler_method(x0, y0, b, n): list euler
improved_euler_method(x0, y0, b, n): list improved
runge_kutta_method(x0, y0, b, n): list runge_kutta
compute_error(method1, method2): list answer list
investigate_convergence(error)
TkMenu
abstract: tkmenu(root, frame)
inputs()
get_values()
convergency()
calculating()
buttons()
clear_place()
method_show()
local_show()
global_show()
PlotGraph
plotMethods(exact, euler, improved, runge_kutta, checks)
plotErrors(error1, error2, error3, checks)
plot_total_errors(x0, y0, b, n1, n2, checks)