Differential Equations Computational Practicum

Pavel Baharuev, BS20-01, Variant 16

$$\begin{cases} y' = e^y - 2/x \\ y(1) = -2 \\ x \in (1,7) \end{cases}$$

Let's solve $\begin{cases} y' = e^y - \frac{2}{x} \\ y(1) = -2 \end{cases}$ first order nonlinear nonhomogeneous ODE

$\frac{dy}{dx} = e^y - \frac{2}{x}$

$-\frac{e^y x - x\frac{dy}{dx}}{x} = 0 \quad | \cdot x$

$e^y x - x\frac{dy}{dx} - 2 = 0$

Let $y(x) = \log\left(\frac{v(x)}{x}\right) \Rightarrow \frac{dy(x)}{dx} = \frac{x\left(\frac{dv(x)}{dx} - \frac{v(x)}{x^2}\right)}{v(x)}$

$v(x) - \frac{x^2\left(\frac{dv(x)}{dx} - \frac{v(x)}{x^2}\right)}{v(x)} - 2 = 0$

$v(x) - \frac{\cancel{x^2}\frac{dv(x)}{dx}}{\cancel{x^2}v(x)} - 1 = 0$

Solve for $\frac{dv(x)}{dx}$

$\frac{dv(x)}{dx} = \frac{(v(x)-1)v(x)}{x} \quad | : (v(x)-1)v(x)$

$\frac{\frac{dv(x)}{dx}}{(v(x)-1)(v(x))} = \frac{1}{x}$

$\int \frac{\frac{dv(x)}{dx}}{(v(x)-1)v(x)}\,dx = \int \frac{1}{x}\,dx$

$\log(-v(x)+1) - \log(v(x)) = \log(x) + C, \; C - \text{const}$

Finally solve for y:

$y(x) = -\log(x) - \log(cx+1)$ general solution

Let's find partial solution.

$y(1) = -2 \Rightarrow$

$-\log(c+1) = -2$

$c_1 = -1 + e^2$

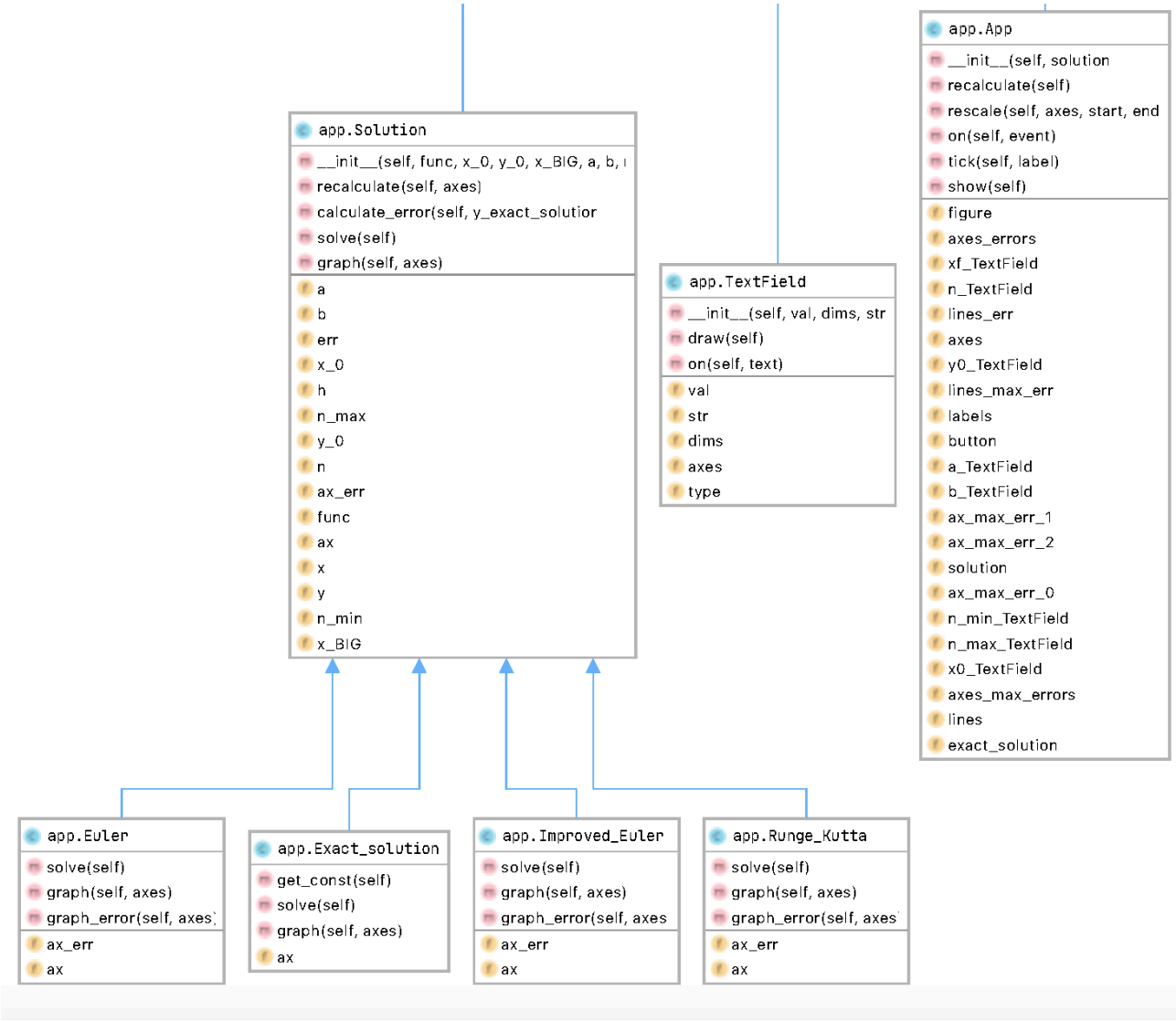
$y(x) = -\log(x) - \log((-1+e^2)x + 1)$

$v(x) = \frac{1}{e^c x + 1}$

$y(x) = \log\left(\frac{v(x)}{x}\right) \Rightarrow v(x) = xe^y$

$e^{y(x)}x = \frac{1}{cx+1}$

UML Diagram of classes



The program represents Solution class, App class and Classes of Euler, Improved_Euler, Runge-Kutta and Exact_solution methods. Also there is a function() and initiate_process() functions.

Interesting parts of code

```
def solve(self):
    k = numpy.zeros([4])
    for i in range(1, self.n + 1):
        k[0] = self.func(self.x[i - 1], self.y[i - 1], self.a, self.b)
        k[1] = self.func(self.x[i - 1] + self.h / 2, self.y[i - 1] + self.h * k[0] / 2, self.a, self.b)
        k[2] = self.func(self.x[i - 1] + self.h / 2, self.y[i - 1] + self.h * k[1] / 2, self.a, self.b)
        k[3] = self.func(self.x[i - 1] + self.h, self.y[i - 1] + self.h * k[2], self.a, self.b)
        self.y[i] = (self.h / 6) * (k[0] + 2 * k[1] + 2 * k[2] + k[3]) + self.y[i - 1]
```

Solution method of Runge-Kutta method

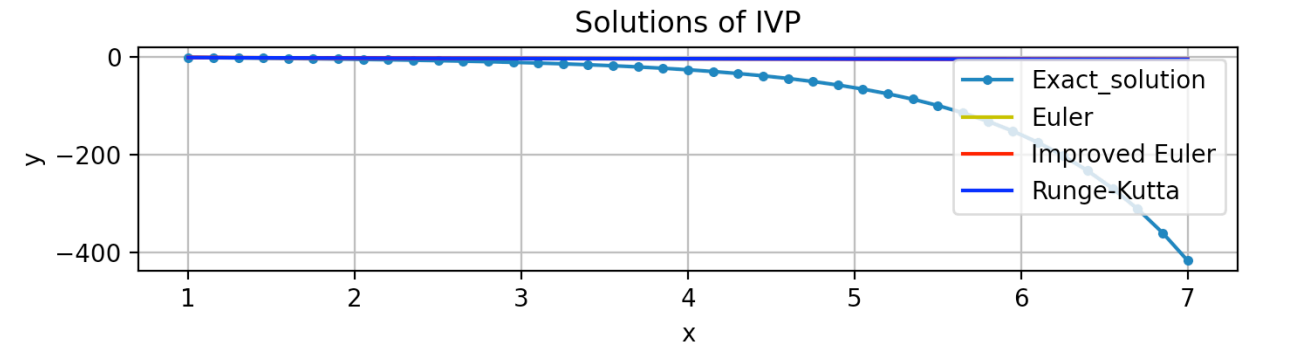
```
for i in range(self.n_min_TextField.val, self.n_max_TextField.val + 1):
    exact_solution = Exact_solution(function, self.x0_TextField.val, self.y0_TextField.val, self.xf_TextField.val, self.a_TextField.val, self.b_TextField.val, i)
    euler = Euler(function, self.x0_TextField.val, self.y0_TextField.val, self.xf_TextField.val, self.a_TextField.val, self.b_TextField.val, i)
    i_euler = Improved_Euler(function, self.x0_TextField.val, self.y0_TextField.val, self.xf_TextField.val, self.a_TextField.val, self.b_TextField.val, i)
    r_k = Runge_Kutta(function, self.x0_TextField.val, self.y0_TextField.val, self.xf_TextField.val, self.a_TextField.val, self.b_TextField.val, i)
    euler.calculate_error(exact_solution.y)
    i_euler.calculate_error(exact_solution.y)
    r_k.calculate_error(exact_solution.y)
    e_total_errors[i - self.n_min_TextField.val] = max(numpy.amax(euler.err), abs(numpy.amin(euler.err)))
    i_total_errors[i - self.n_min_TextField.val] = max(numpy.amax(i_euler.err), abs(numpy.amin(i_euler.err)))
    r_total_errors[i - self.n_min_TextField.val] = max(numpy.amax(r_k.err), abs(numpy.amin(r_k.err)))
```

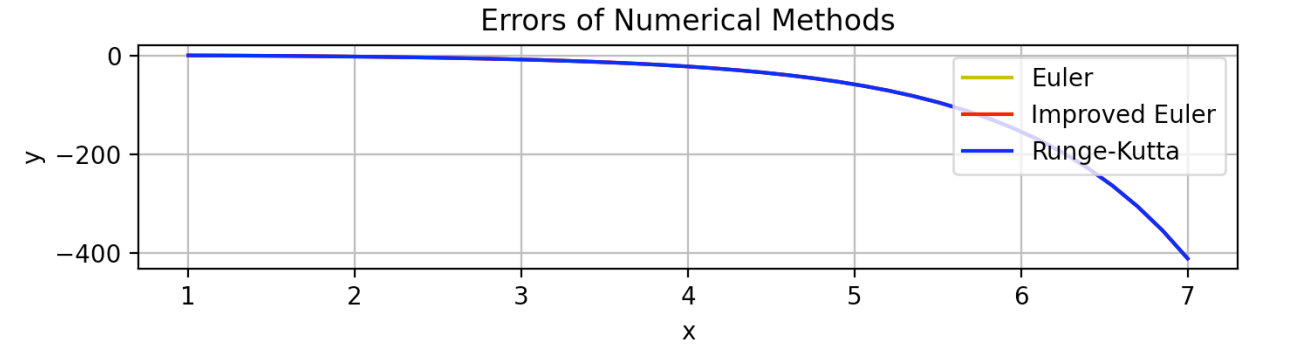
Recalculate() method. This method calls Euler, Improved_Euler, Runge-Kutta methods

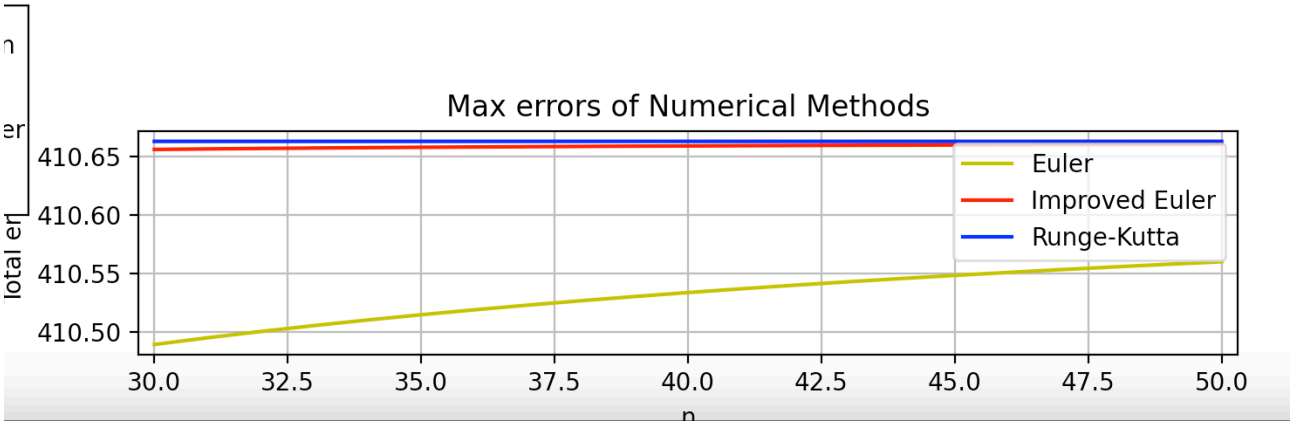
```
def solve(self):
    const = self.get_const()
    for i in range(1, self.n + 1):
        self.y[i] = (-2) * self.x[i] + 1 + const * math.exp(self.x[i])
```

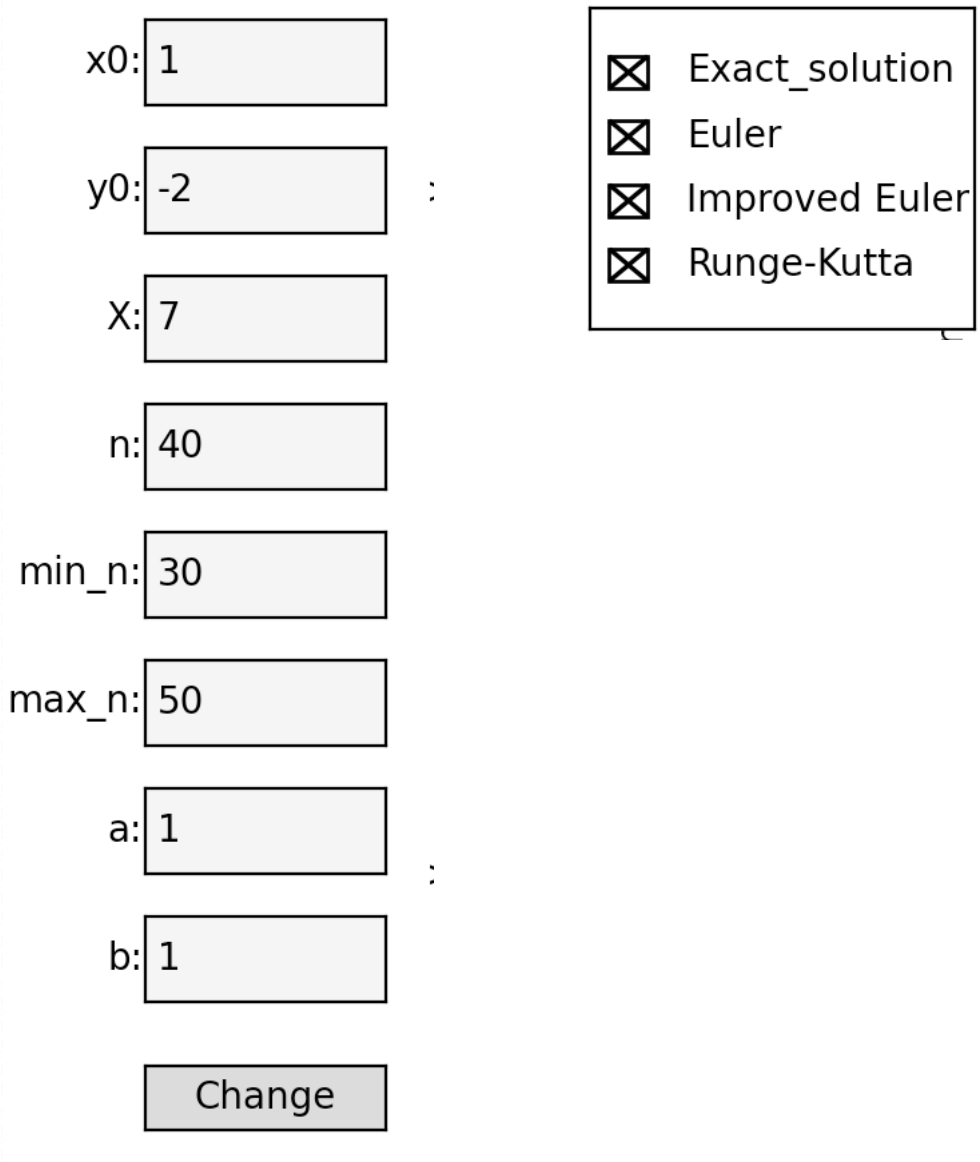
Exect_solution solve() method

GUI








x0:
1
y0:
-2
X:
7
n:
40
min_n:
30
max_n:
50
a:
1
b:
1
Change
Exact_solution
Euler
Improved Euler
Runge-Kutta