Computational practicum

Name: Mahmoud Mousatat
Group: BS20-01

Exact solution

$$\begin{cases} y' = 3y - xy^{\frac{1}{3}} \\ y(1) = 2 \end{cases}$$

$$y' - 3y = xy^{\frac{1}{3}}$$

So, this is Bernoulli equation.

Let's solve it. First of all we should divide both parts by $y^{\frac{2}{3}}$

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

the make substitution

$$z = y^{\frac{2}{3}} \quad z' = \frac{2}{3} y^{-\frac{1}{3}} y'$$

we get

$$\frac{2}{3} z' - 3z = -x \qquad (1)$$

Equation (1) is first-order non-homo linear ordinary differential equations.

So, first of all we need to solve complementary equation, $\frac{3}{2} z' - 3z = 0$

$$z' = 2z$$

$$\int \frac{dz}{z} = 2 \int dx$$

$$e^{\ln|z|} = e^{2x + C_1}$$

$$z = e^{2x} C_2$$

$$z' = 2e^{2x}C_2 + C_2' e^{2x}$$

substitute to equation(1)

$$3e^{2x}C_2 + \frac{3}{2}C_2' e^{2x} - 3e^{2x}C_2 = -x$$

$$\frac{3}{2}C_2' e^{2x} = -x$$

$$C_2' = -\frac{2}{3} x e^{-2x}$$

$$C_2 = -\frac{2}{3}\int x e^{-2x} dx = \frac{2}{3}\cdot\frac{(2x+1)}{4}\cdot e^{-2x} + C_3$$

$$z = \frac{2x+1}{6} + e^{2x}C_3$$

$$z = \frac{x}{3} + \frac{1}{6} + e^{2x}C_3$$

Back substitution

$$y^{\frac{2}{3}} = \frac{x}{3} + \frac{1}{6} + e^{2x}C_3$$

$$y = \left(\frac{x}{3} + \frac{1}{6} + e^{2x}C_3\right)^{\frac{3}{2}}$$

Lets find C_3:

$$C_3 = \frac{y^{\frac{2}{3}} - \frac{x}{3} - \frac{1}{6}}{e^{2x}} \qquad y(1) = 2$$

$$C_3 = \left(2^{\frac{2}{3}} - \frac{1}{2}\right)e^{-2}$$

$$\boxed{y = \left(\frac{x}{3} + \frac{1}{6} + e^{2(x-1)}\cdot\left(2^{\frac{2}{3}} - \frac{1}{2}\right)\right)^{\frac{3}{2}}}$$

the answer

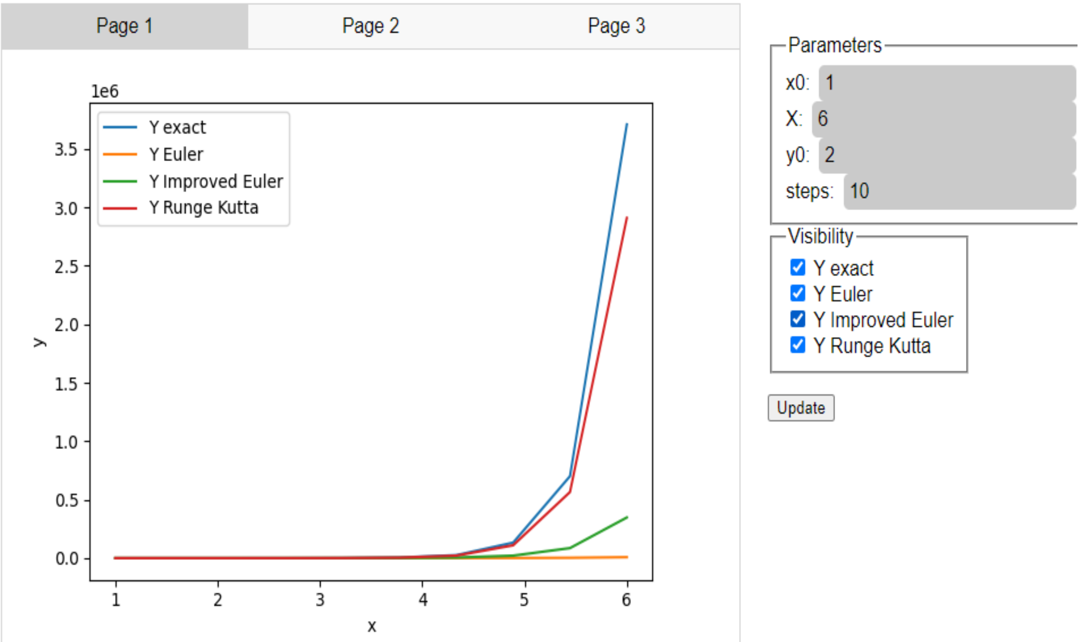
Graphs

Graph of solutions

There are 4 lines represented different types of the solution:

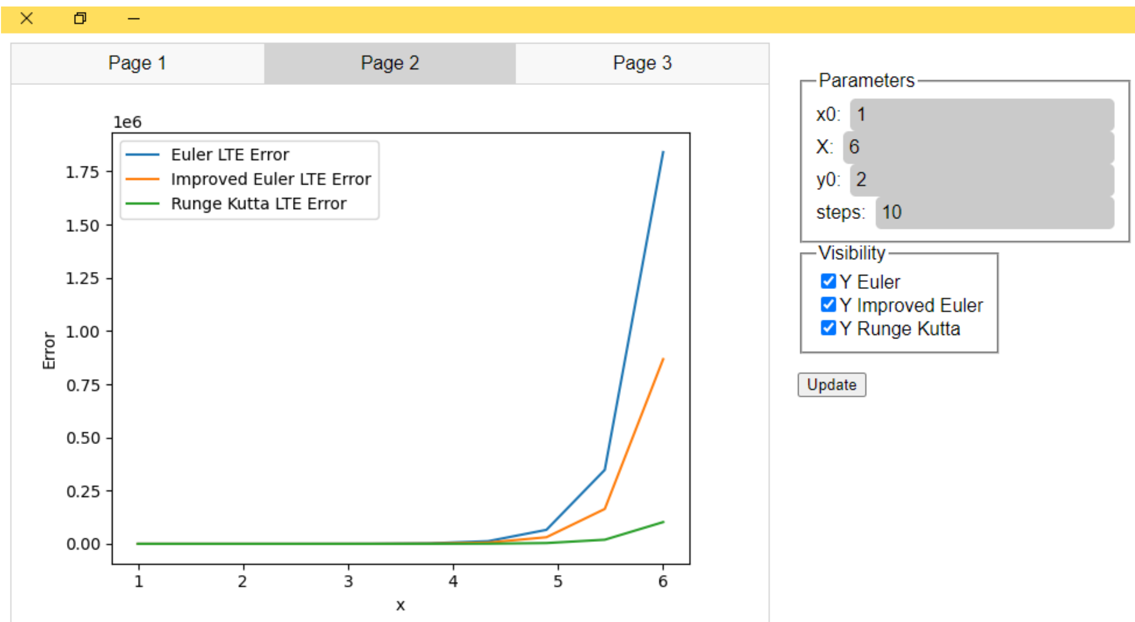
- Exact solution;
- Approximate solution using Euler's method;
- Approximate solution using Improved Euler's method;
- Approximate solution using Runge Kutta method.

$y - axis$ represents solution for given x with values $\in [0, 2.7 * 10^{43}]$.



Graph of local errors

There are local truncation errors (LTE) of each method.

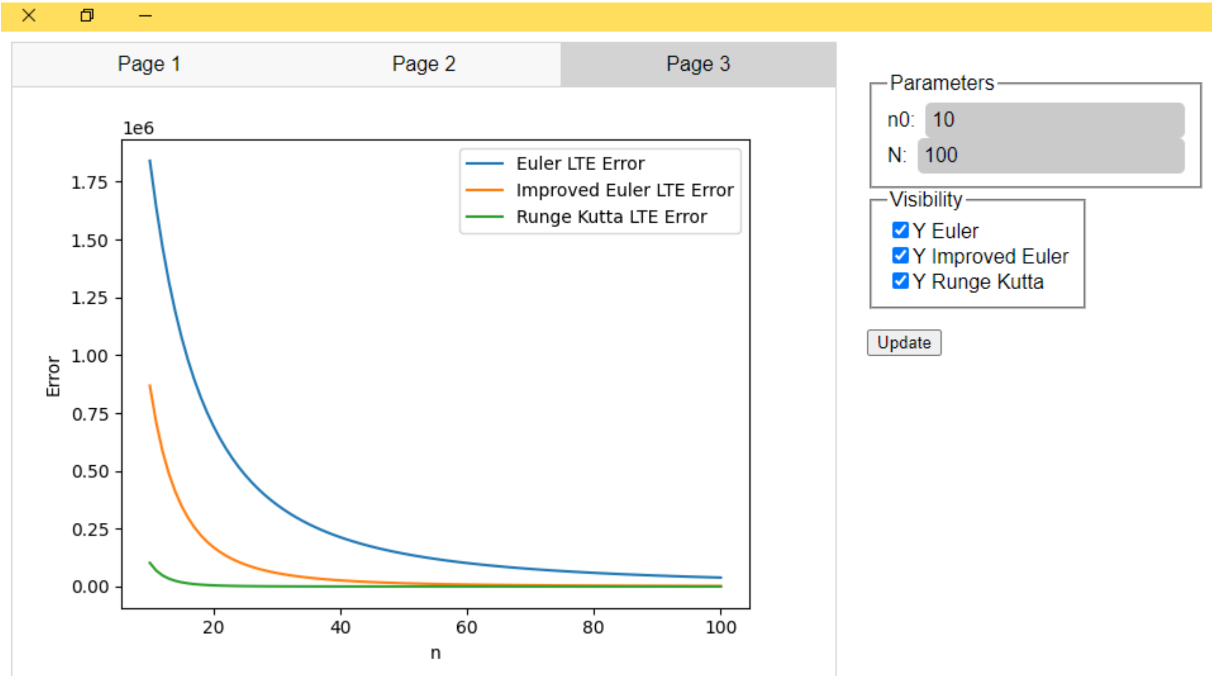


Graph of total approximation error

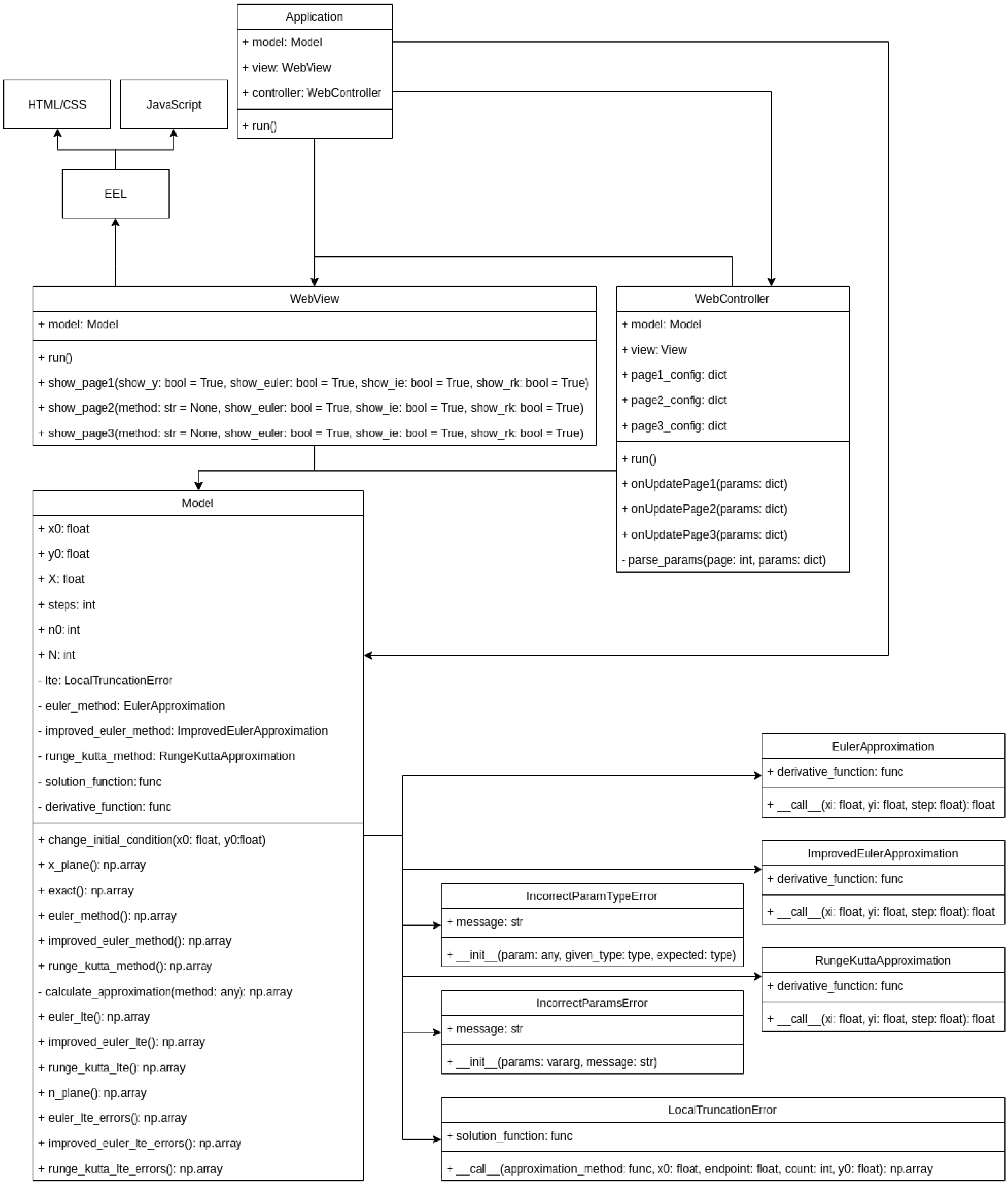
There are changing LTE of each approximation method depending on the given step.

It calculates the maximum local error on the range $[x0, X]$ for each number of steps on the range $[n0, N]$ with step 1.

- n_0 - starting number of steps
- N - end number of steps



UML class diagram



Parts of the code

Project structure

```
project

|   ├── controller
|   |   ├── console_controller.py
|   ├──___init__.py
|   ├──___main__.py  // Starting point for the application
|   |   ├── approximations  // Folder for approximation functions
|   |   |   ├── euler_method.py
|   |   |   ├── improved_euler_method.py
|   |   ├── errors  // Folder for truncation error classes
|   |   |   ├── gte.py
|   |   ├── exceptions  // Custom exceptions
|   |   |   ├── incorrect_params_error.py
|   |   └── model.py  // Business logic
|   └── view
|       ├── console_view.py
|       |   ├── css
|       |   |   └── style.css
|       |   ├── img
|       |   |   └── graph.png
|       |   ├── index.html
|       |   └── js
|       |       ├── controller.js
├── README.md
├── report
|   └── Report.pdf  // File with report
├── requirements.txt
└── tests  // Folder for tests
    └── model
        ├── test_euler_method.py
```

Run application (___main__.py)

```
if___name__ == '__main__':
    app = Application(model, view, controller)
    app.run()
```

Run graphical user interface (web_view.py)

```
def run(self) -> None:
    self._change_image({}, 1, callback_needed=False)
    eel.init('view/static')
    eel.start('index.html', size=(1000, 600))
```

Calculation of LTE (lte.py)

```
arr = np.zeros(shape=steps, dtype=np.float64)
xi = x0
y_real = y0
for i, x in enumerate(np.linspace(x0, endpoint, steps)):
    if i == 0:
        continue
    y_approximate = approximation_method(xi, y_real, step)
    y_real = self.solution_function(x)
    arr[i] = abs(y_real - y_approximate)
```

Plotting and saving a graph (web_view.py)

```
def _change_image(table: dict, page_number: int, callback_needed=True) -> None:
    for key in table.keys():

        if key == 'X':
            continue

        plt.plot(table['X'], table[key], label=key)
    if page_number == 1:

        plt.xlabel('x')

        plt.ylabel('y')
    elif page_number == 2:

        plt.xlabel('x')
        plt.ylabel('Error')

    elif page_number == 3:
        plt.xlabel('n')
        plt.ylabel('Error')

    if len(table) > 1:
        plt.legend()
```

Tests of the application

Code for testing **local truncation error** using 3 methods of approximation:

```
def setUp(self):

test_func = lambda x: (x/3+1/6+np.exp(2*(x-1))*(2**(2/3)-1/2))**(3/2)

self.derivative_func = lambda x, y: 3 * y - x * (y ** (1/3))


euler_method = EulerApproximation(self.derivative_func) improved_euler_method = ImprovedEulerApproximation(self.derivative_
runge_kutta_method = RungeKuttaApproximation(self.derivative_func)


    self.lte = LocalTruncationError(test_func)


def test_euler(self):

    expected = np.array([0., 0.087150835, 0.13986887, 0.2441393, 0.48296002, 1.1715976], dtype=np.float32)

    val = self.lte(EulerApproximation(self.derivative_func), 1, 1.5, count=6)


    self.assertIs(type(val), np.ndarray)
    self.assertEqual(len(val), 6)
    self.assertEqual(len(val), len(expected))
    np.testing.assert_array_almost_equal(val,  expected)


def  test_improved_euler(self):

    expected = np.array([0., 0.01368145, 0.023602538, 0.04599498, 0.106638946, 0.32514724], dtype=np.float32)

    val = self.lte(ImprovedEulerApproximation(self.derivative_func), x0=1.0, endpoint=1.5, count=6, y0=2.0)


    self.assertIs(type(val), np.ndarray)
    self.assertEqual(len(val), 6)
    self.assertEqual(len(val), len(expected))
    np.testing.assert_array_almost_equal(val,  expected)
```