Computational Practicum Report

Student: Bekzat Rakhimbayev

Group: B20-03

E-mail: b.rakhimbayev@innopolis.university

Part 1

Task

Given the initial value problem with the ODE of the first order and some interval

$$\begin{cases} y' = f(x,y) \\ y(x_0) = y_0 \\ x \in (x_0, X) \end{cases} \quad \rightarrow \quad \begin{cases} y' = \frac{\sqrt{y-x}}{\sqrt{x}} + 1 \\ y(1) = 10 \\ X = 15 \end{cases}$$

Solution:

The equation is homogeneous differential equations therefore we can do this substitution

$$y = xt(x), y' = t(x)'x + t(x)$$

We get this equation:

$$t(x)'x + t(x) = \frac{\sqrt{t(x)x - x}}{\sqrt{x}} + 1$$

simplify:

$$t(x)' = \frac{\sqrt{t(x)-1} + 1 - t(x)}{x}$$

Divide both side by:

$$\sqrt{t(x)-1} + 1 - t(x) \text{ —> } \frac{\frac{dt}{dx}}{\sqrt{t(x)-1}+1-t(x)} = \frac{1}{x}$$

Multiply both side by dx :

$$\int \frac{dt}{\sqrt{t(x)-1}+1-t(x)} = \int \frac{dx}{x}$$

$$\int \frac{dx}{x} = ln(x) + c$$

for solving: $\int \frac{dt}{\sqrt{t(x)-1}+1-t(x)}$

we use substitution: $u = \sqrt{t-1}; \frac{du}{dt} = \frac{1}{2\sqrt{t-1}} \rightarrow dt = 2\sqrt{t-1}du; t = u^2 + 1$

after substituting we get:

$$\int \frac{2u}{u+1-u^2-1} du = 2\int \frac{1}{1-u} du = 2ln(1-u) = 2ln(1-\sqrt{t(x)-1})$$

so we get this:

$$2ln(1-\sqrt{t(x)-1}) = ln(x) + c$$

$$ln(1-\sqrt{t(x)-1}) = \frac{ln(x)}{2} + \frac{c}{2}$$

$$\sqrt{t(x)-1} = 1 - x^{-2} - e^{\frac{c}{2}}$$

$$t(x) - 1 = (1 - x^{-2} - e^{\frac{c}{2}})^2$$

Solve for $t(x)$:

$$t(x) = -\frac{2e^{-\frac{c1}{2}}}{\sqrt{x}} + \frac{e^{-c1}}{x} + 2$$

Simplify:

$$t(x) = -\frac{2}{c_1\sqrt{x}} + \frac{1}{c_1^2 x} + 2$$

Substitute $t(x) = \frac{y}{x}$

$$y = x(-\frac{2}{c\sqrt{x}} + \frac{1}{c^2 x} + 2)$$

General Solution:

$$y = -\frac{2\sqrt{x}}{c} + \frac{1}{c^2} + 2x$$

to find coefficient will rewrite suitable equation for discriminant

$yc^2 = -2c\sqrt{x} + 1 + 2xc^2$

$(y - 2x)c^2 + (2\sqrt{x})c - 1 = 0$

$c_1 = \frac{(-2\sqrt{x}+\sqrt{4x+4(y-2x)})}{2(y-2x)}$

$c_2 = \frac{(-2\sqrt{x}-\sqrt{4x+4(y-2x)})}{2(y-2x)}$

Substitute $y(1) = 10$

$$c_1 = \frac{-2+\sqrt{4+4*8}}{2*8} = \frac{1}{4}$$

$$c_c = \frac{-2-\sqrt{4+32}}{16} = -\frac{1}{2}$$

For $c = -\frac{1}{2}$:

$$y = 2(2\sqrt{x} + x + 2)$$

For $c = \frac{1}{4}$:

y=2(-4\sqrt{x}+x+8)

Exact Solution Answer:

- $y = 2(2\sqrt{x} + x + 2)$
- $y = 2(-4\sqrt{x} + x + 8)$

Part 2 and 3

My Applications allow building graph of exact solution, numerical solution, local truncation error and global truncation error. In the project used methods such as Euler, Improved Euler and Runge Kutta.

The project is written in python3 language and user interface is created on Matplotlib library

How to use:

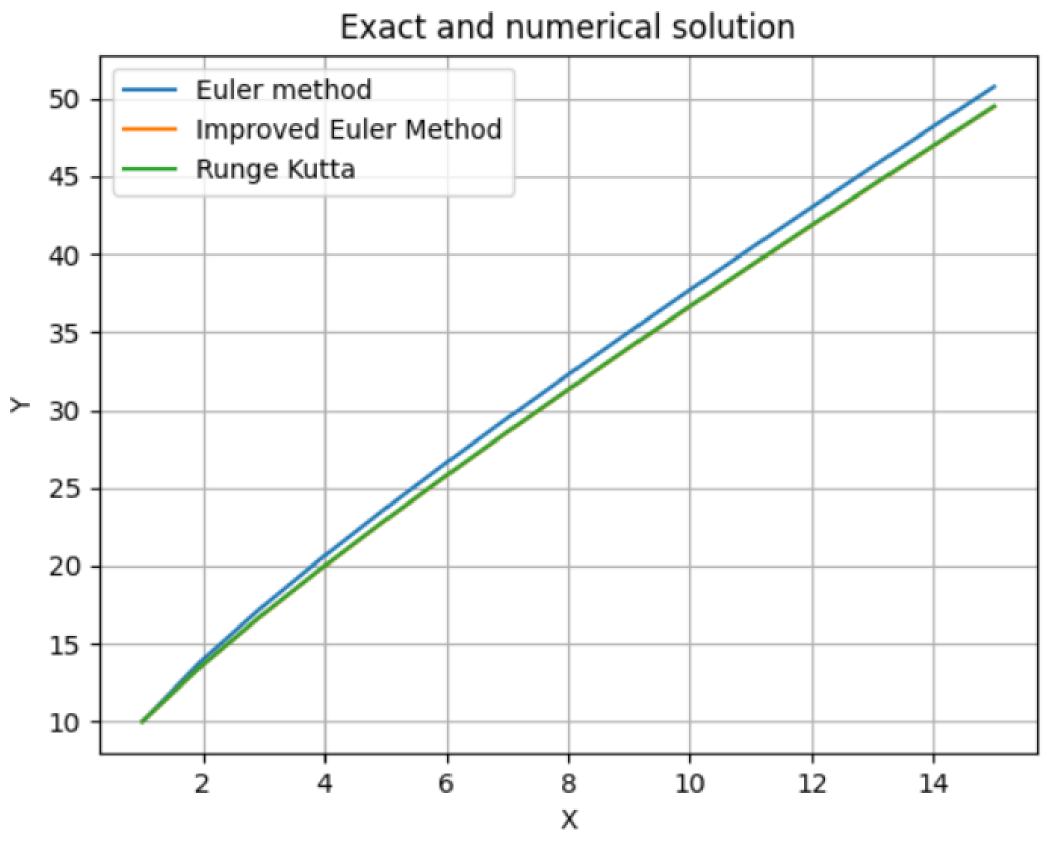
you can install and run from source with:

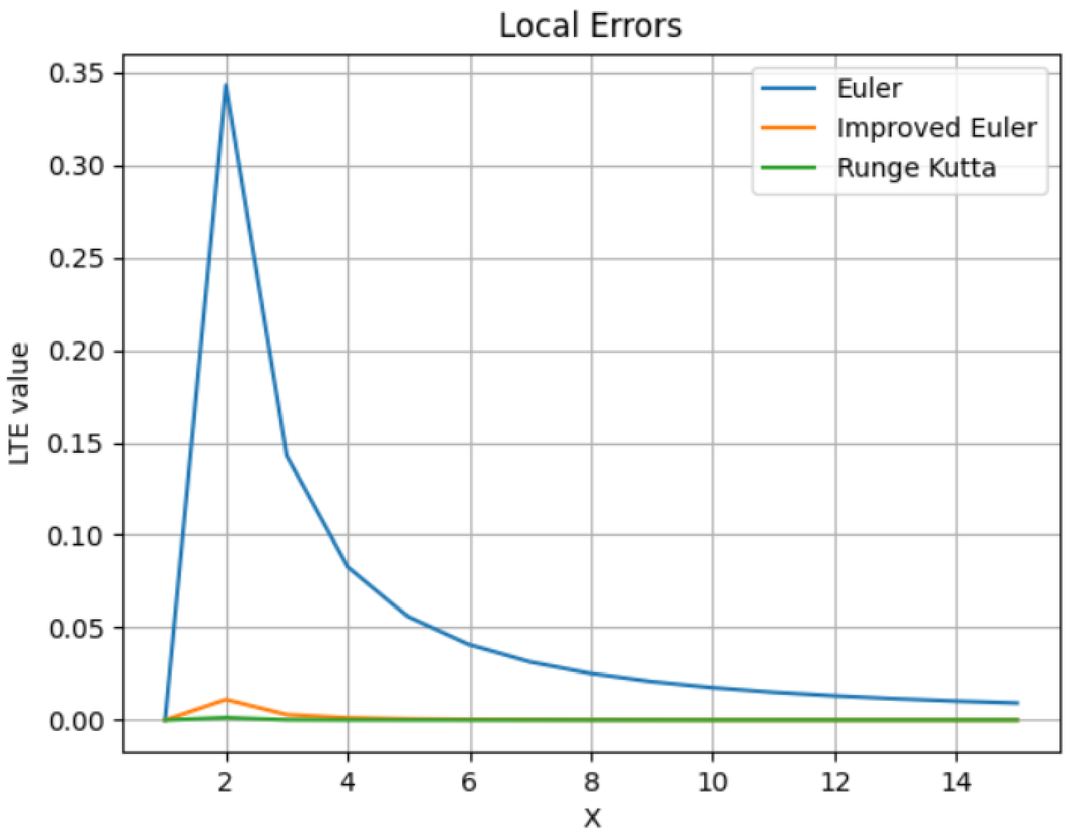
- `git clone https://github.com/rkBekzat/ComputationalPracticum.git --recursive`
- `cd ComputationalPracticum`

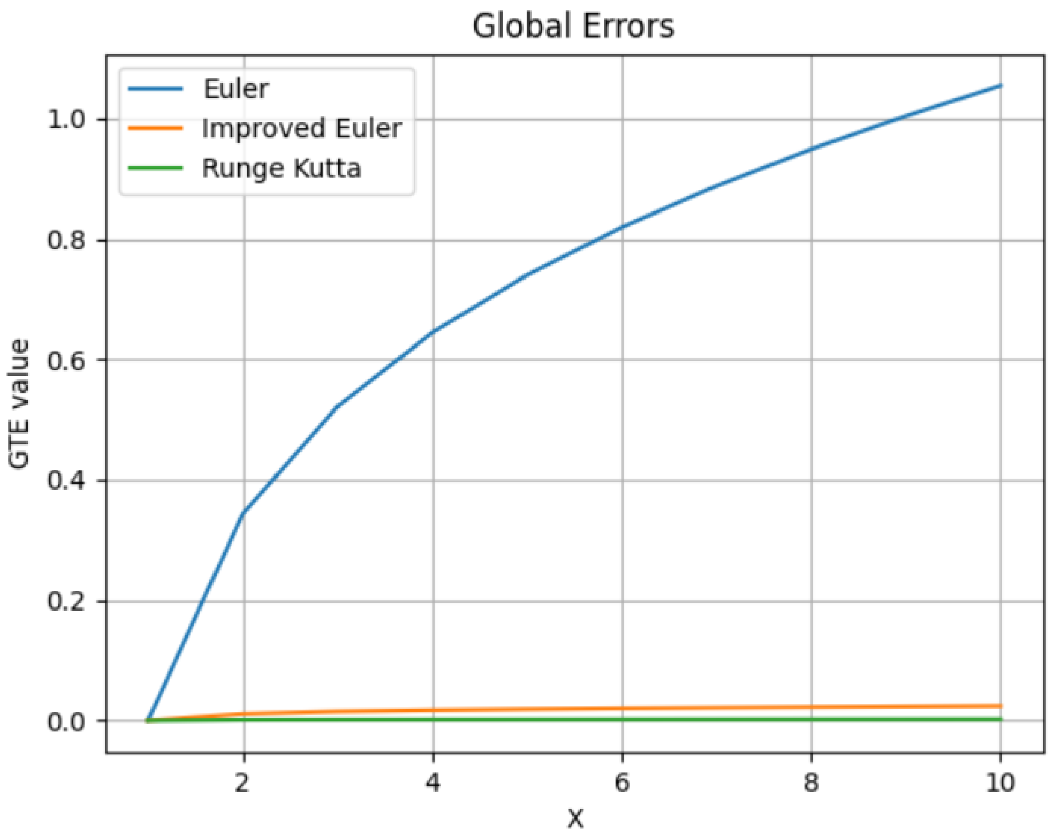
- `pip install -r requirements.txt`
- `python main.py`

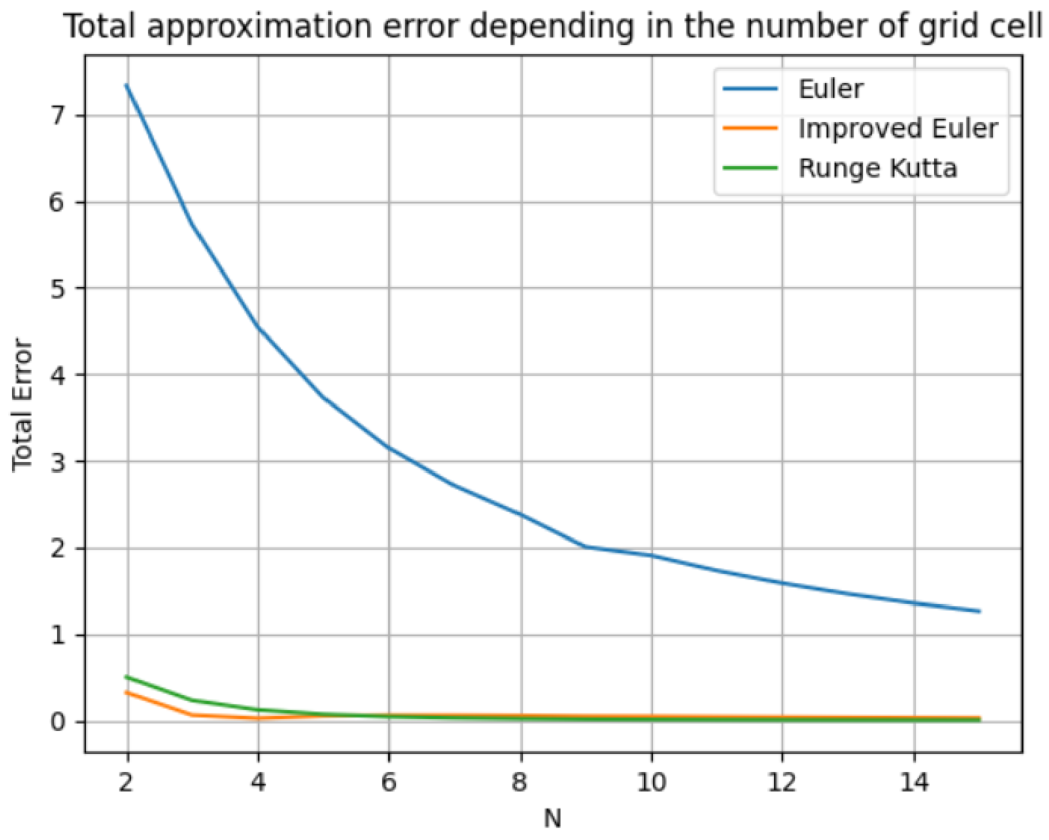
.

Graphs



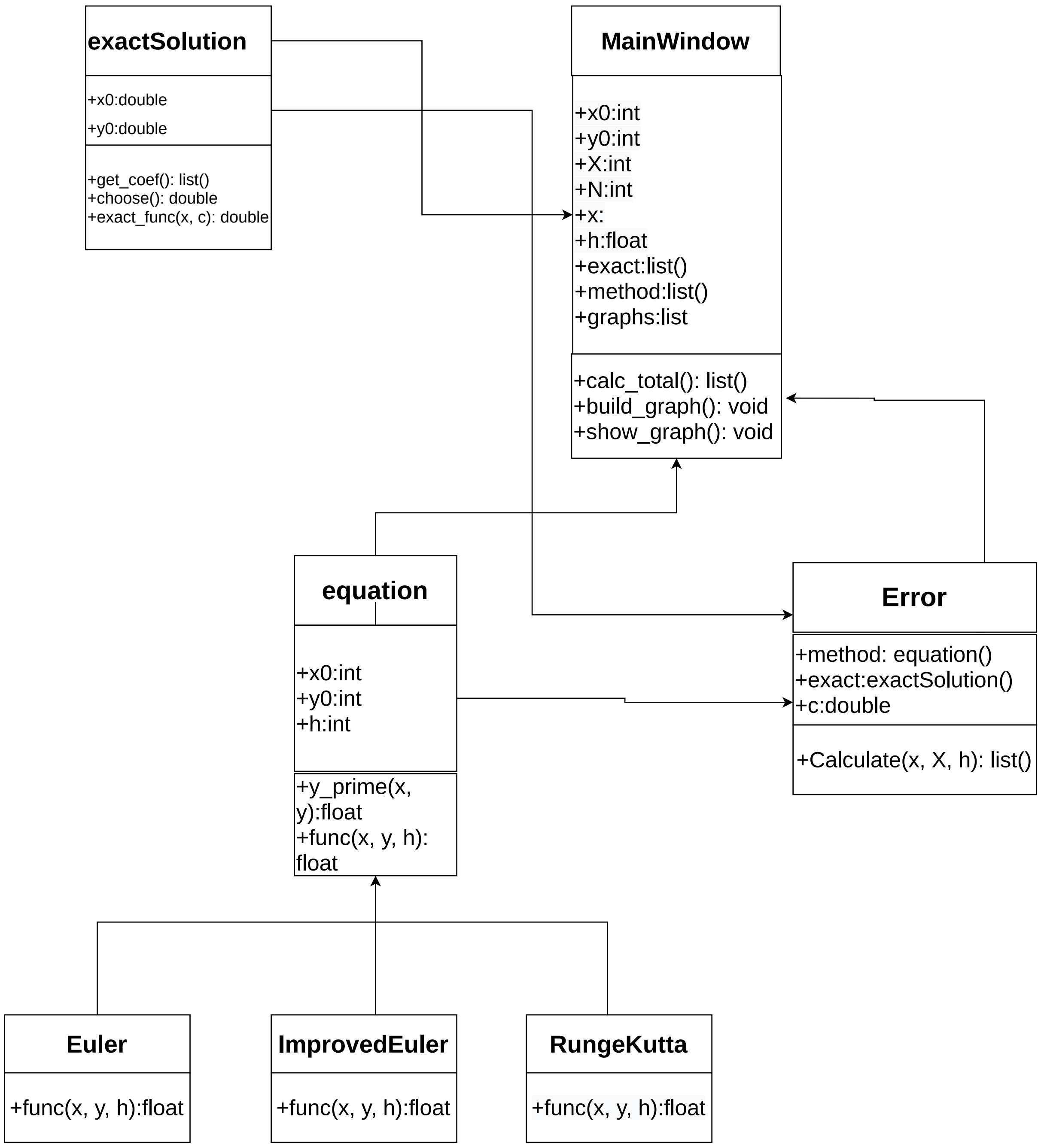

Local Errors
Euler
Improved Euler
Runge Kutta
LTE value
X
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
2
4
6
8
10
12
14


Global Errors
Euler
Improved Euler
Runge Kutta
GTE value
X
1.0
0.8
0.6
0.4
0.2
0.0
2
4
6
8
10



UML

The project has following UML diagram:



Source code:

The program start from entering the data by user on terminal, also user can't enter the number less or equal to the 0 for initial x_0. Because if initial x_0 less than 0 then the equation contain complex number which impossible to draw on 2 dimension graph. If initial x_0 equal to the 0 then we obtain calculating error which mean division by zero. After enter the data, creating **MainWindow** class and initialize with x_0, y_0, X, N variables. The class **MainWindow** is responsible for all user interface action and consist 4 methods: constructor(_*init*_), calc_total(), build_graph(), show-graph().After calling last function the program show graph as above mentioned.

```
#main.py
x0, y0, X, N = Enter_data()
```

```
gui=MainWindow(x0, y0, X, N)
gui.build_graph()
gui.show_graphs()
```

The function $gui.build\_graph()$ intended to build exat and numerical solution, LTE, GTE and Total approximation error graphs. The function $Calculate(self, x, X, h, sz)$ return lists of local truncation error and global truncation error.

LTE - difference between exact function and approximation from previous exact function.

LTE[i]=|y_exact(i)-y_approx(i, y_exact(i-1))|

GTE - difference between exact and approximation function.

GTE(x)=|y_exact(x)-y_approx(x)|

TotalError(n) - max(GTE); n is number of grid cell on graph

```
def Calculate(self, x, X, h, sz):
        #LTE
        lte = list()
        lte.append(0)
        for i in range(sz-1):
            lc = self.method.func(x, self.exact.exact_func(x, self.c), h)
            lte.append(abs(lc - self.exact.exact_func(x + h, self.c)).real)
            x += h

        #GTE
        gte = list()
        gte.append(0)
        x = self.method.x0
        y = self.method.y0
        for i in range(sz-1):
            gte.append(abs(self.method.func(x, y, h) - self.exact.exact_func(x + h, self.c)))
            y = self.method.func(x, y, h).real
            x += h

        return [lte, gte]
```

The method calc_total() return list of total approximation error for methods Euler, Improved Euler and Runge Kutta. Here grid cells will change from 2 to N. To calculate total error for certain method call function Calculate()[1] from Error class and get maximum among GTE answers.

```
def calc_total(self):
        total = [list(), list(), list()]
        for i in range(2, self.N+1):
            x=np.linspace(self.x0, self.X, i)
            h=(self.X-self.x0+1)/i
            for j in range(3):
                total[j].append(max(Error(self.methods[j], self.exact_y, self.c).Calculate(self.x0, self.X, h, i)[1]))
        return total
```

To find exact solution for any $x_0$and y_0 here created exactSolution class which has method get_coef(), choose(), exact_func(). The method get_coef() return two coefficient which calculated by discriminant of this equation $(y-2x)c^2+(2\sqrt{x})c-1=0$. The method exac_func(x, c) return this function $y=-\frac{2\sqrt{x}}{c}+\frac{1}{c^2}+2x$. The method choose() return coefficient which close to numerical methods.

```
#exactSolution class
def get_coef(self):
    c1=(-2*math.sqrt(self.x0)+math.sqrt(self.x0*4+4*(self.y0-self.x0*2)))/(2*(self.y0-self.x0*2))
    c2=(-2*math.sqrt(self.x0)-math.sqrt(self.x0*4+4*(self.y0-self.x0*2)))/(2*(self.y0-self.x0*2))
    return [c1, c2]

def choose(self):
    h=0.1
    method=RungeKutta(self.x0, self.y0, h)
    next_y=method.func(self.x0, self.y0, h) # we find y for self.x0+0.1
    coef=self.get_coef()
    dif1=next_y-self.exact_func(self.x0+h, coef[0])
    dif2=next_y-self.exact_func(self.x0+h, coef[1])
    if dif1.real > dif2.real:
        return coef[1]
    return coef[0]

def exact_func(self, x, c):
    return -2*math.sqrt(x)/c+1/(c**2)+2*(x)
```

Conclusion

In conclusion, among numerical methods Runge Kutta and Improved Euler methods are very close to exact solution. Moreover, if the number of grids cell will increase then total approximation error will decrease.