Differential Equations: Computational Practicum

Khabib Khaysadykov
BS 20-03
Variant 13

Analytical solution

Initial system

$$\begin{cases} y' = e^{2x} + e^x + y^2 - 2ye^x \\ y(0) = 0 \\ x \in (0, 15) \end{cases}$$

Riccati equation

$$y' + a(x)y + b(x)y^2 = c(x)$$
$$y' + 2e^x y - y^2 = e^{2x} + e^x$$

where:

$$a(x) = 2e^x$$
$$b(x) = -1$$
$$c(x) = e^{2x} + e^x$$

finding particular solution:

$$y_1 = e^x + a$$
$$y_1' = e^x$$
$$e^x + 2e^x(e^x + a) - (e^x + a)^2 = e^{2x} + e^x$$

subtitution:

$$y = y_1 + z$$
$$y = e^x + z$$
$$y' = e^x + z'$$
$$e^x + z' + 2e^x(e^x + z) - (e^x + z)^2 = e^{2x} + e^x$$
$$e^x + z' + 2e^{2x} + 2e^x z - e^{2x} - 2e^x z - z^2 = e^{2x} + e^x$$
$$e^x + z' = e^x + z^2$$
$$z' = z^2$$

transforming z'=$\frac{dz}{dx}$:

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

dividing on z^2both sides:

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

when dividing lost solution:

$$z = 0 \rightarrow y - e^x = 0 \rightarrow x = ln(y)$$

integrating both sides of equation:

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

$$-\frac{1}{z} = x + C$$

$$\frac{1}{z} = C - x$$

back substitution:

$$\frac{1}{y - e^x} = C - x$$

$$y = e^x - \frac{1}{x + C}$$

Initial value problem

$$y(0) = 0$$

check the lost solution:

$$y - e^x = 0$$

$$0 - e^0 = 0$$

$$0 = 1$$

it is not included in the solution

substituting for C:

$$0 = e^0 - \frac{1}{C}$$

$$0 = 1 - \frac{1}{C}$$

$$\frac{1}{C} = 1$$

$$C = 1$$

and we can obtain general formula for C:

$$C = -\frac{1}{y - e^x} - x$$

particular solution is:

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

Discontinuity points

The particular function has no discontinuity points on the given interval.

Final system

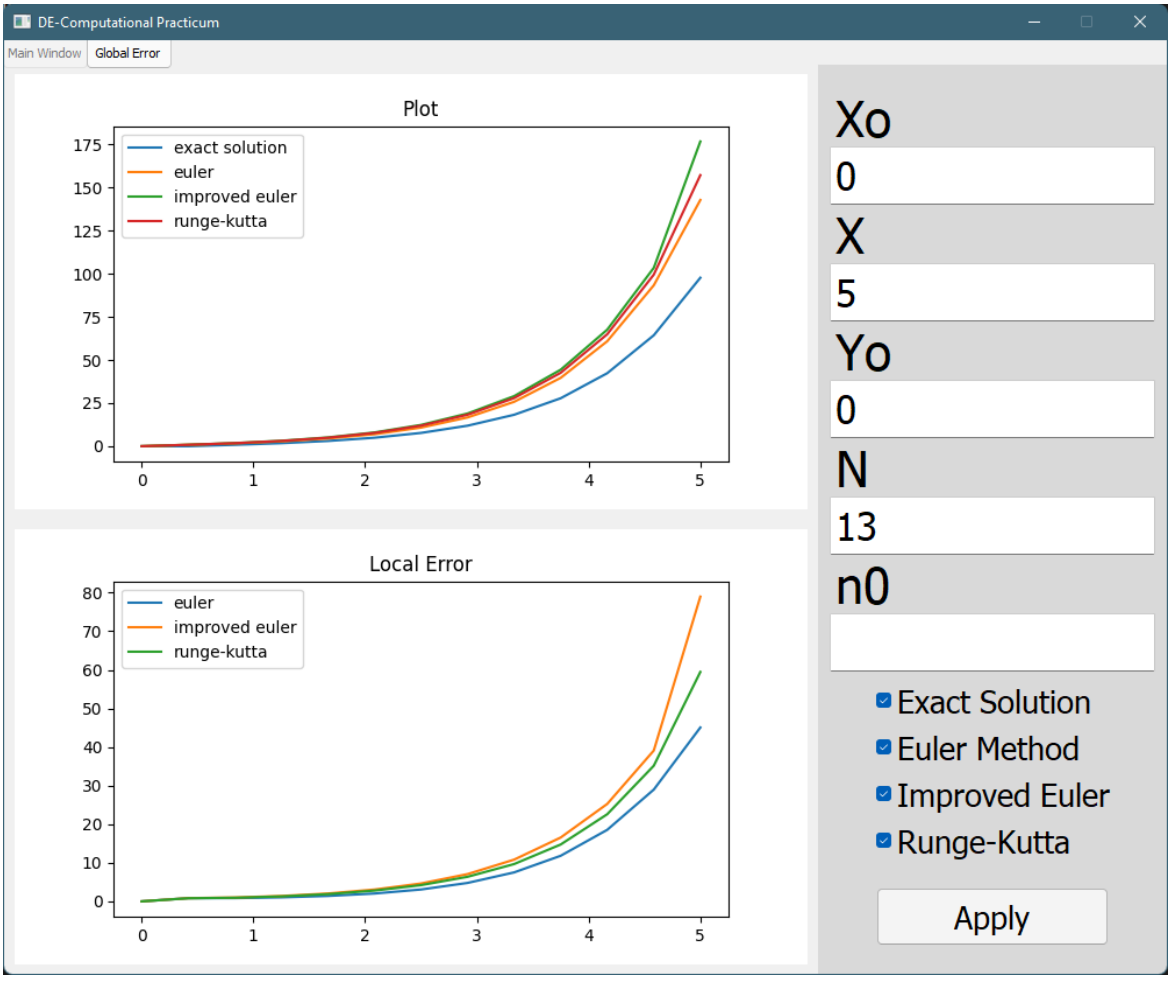
These equations were implemented in the code:

$$\begin{cases} y' = e^{2x} + e^x + y^2 - 2ye^x \\ y = e^x - \frac{1}{x+1} \end{cases}$$

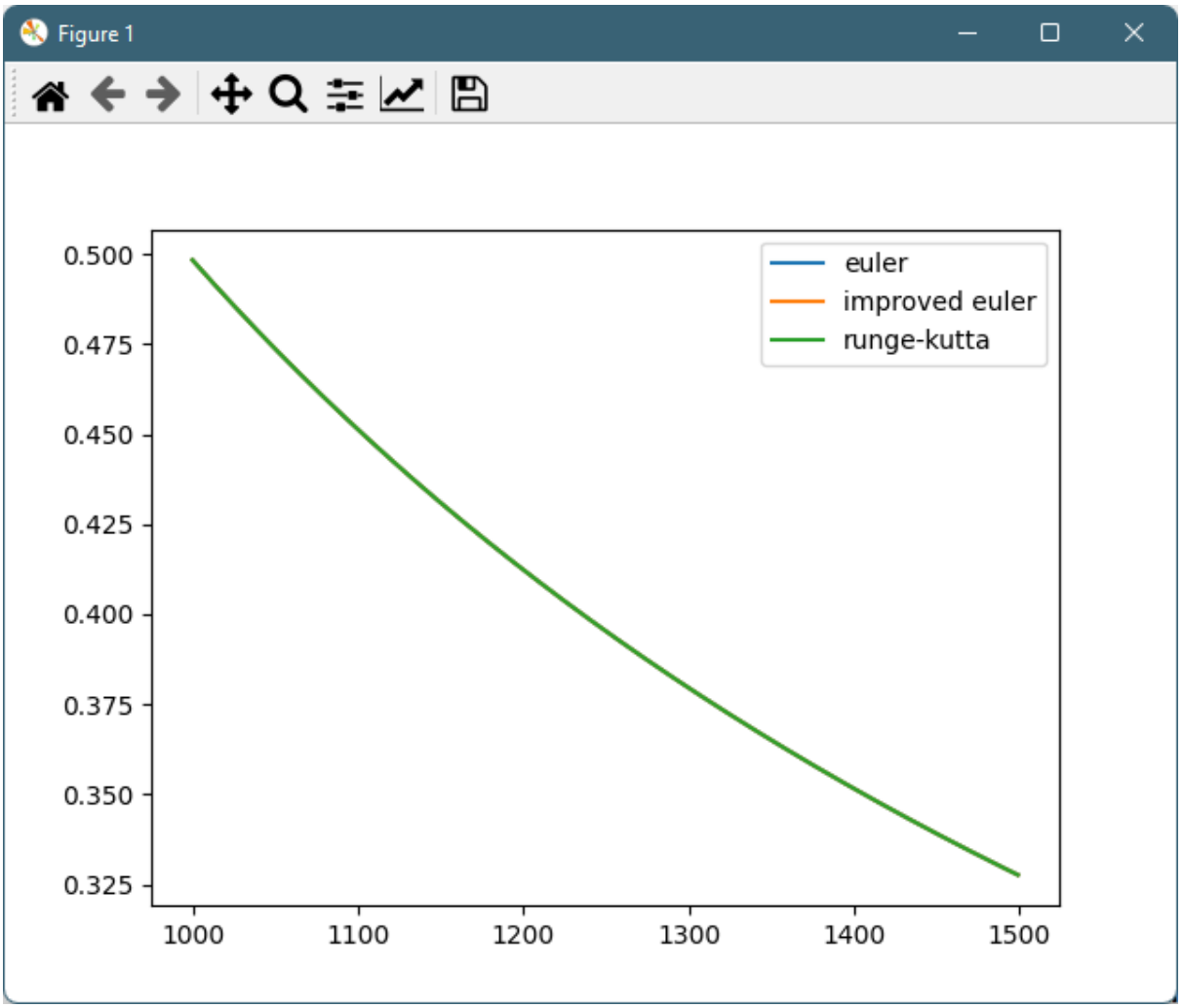
Application

My app provides the possibility to plot a graph of the exact solution that was provided above, and graphs of Euler, Improved Euler, Runge-Kutta methods.
In the app there are three graphs for local error, global error, and equation. All plots are drawn on the same canvas so we can compare all methods.

Program written in Python3
In the program I was usingQtDesigner, pyqt5 to create a user interface and matplotlib for plotting graphs.
Here some screenshots:



It is possible to change X0, X, Y0, N, N0 and also we can turn off and on plotting graphs for some of the methods.



Implementation

The application was developed as a windows app with python implemented logic. All the scripts can be divided into 3 categories:

- ui design
  - qt designer
- math classes
  - numerical method
  - exact solution
- graph plotter
  - matplotlib

Most notably, an attempt was made to generalize the abstract classes *numerical method* and *exact solution* so that adding new variants would be simple and their number could scale without requiring classes modification.

Qt Designer

I chose Qt designer because it has a good implementation of common widgets and it's easy to adjust it for your purposes. Also with Qt designer you can make design inside of application and after it generate python code by this ui file

Math classes

All math classes are divided into 2 categories.

Exact solution provides abstract methods Derivative and exactSolution, implementation of this methods are in MyOwnSolution class they provide analitical(exact) solution and solution of given derivative

```
class MyOwnSolution(ExactSolution):
    def __init__(self, N, y0, x0, X):
        ExactSolution.__init__(self, N, y0, x0, X)
        self.exactSolution()

    # overriding abstract method
    def Derivative(self, x, y):
        return np.power(np.e, 2*x) + np.power(np.e, x) + np.power(y, 2) - 2*y*np.power(np.e, x)

    # overriding abstract method
    def exactSolution(self):
        x = self.X
        y = self.Y
        constant = -1/(self.Y0 - np.power(np.e, self.X0)) - self.X0
        for i in range(self.N):
            if i == 0: y[i] = self.Y0
            else: y[i] = np.power(np.e, x[i-1]) - 1/(constant+x[i-1])

        self.Y = y
```

Numerical method provide abstract methods for different methods of solving differential equations. Based on this class I implemented three classes: euler, improved euler, runge-kutta methods.

```python
class NumericMethod(ABC, Grid):
    exactSolution : ExactSolution
    localError = []
    def __init__(self, N, y0, x0, X, DE):
        Grid.__init__(self, N, y0, x0, X)
        self.exactSolution = DE
        X = self.X
        Y = self.Y
        for i in range(N):
            if i == 0:
                Y[i] = y0
            else:
                Y[i] = self.calculateY(X[i-1], Y[i-1])
        self.Y = Y
        self.calculateLocalError()

    def calculateLocalError(self):
        exactY = self.exactSolution.Y
        numericalY = self.Y
        error = np.empty(self.N, dtype=float)

        for i in range(self.N):
            error[i] = np.abs(exactY[i]-numericalY[i])

        self.localError = error

    @abstractmethod
    def calculateGlobalError(self, n0,):
        pass

    @abstractmethod
    def calculateY(self, x, y):
        pass
```

This class consists of 3 implementations: initialization where we have the main method of calculating, finding local error and global error. And abstract class for each specific implementation.

Graph classes

I chose matplotlib for plotting graphs because this library provides great functionality for plotting graphs. I have imported a library and implemented a class to embed matplotlib graphs into Qt design.

```
class MplWidget (QWidget):

    def __init__(self, parent = None):

        QWidget.__init__(self, parent)

        self.canvas = FigureCanvas(Figure ())

        vertical_layout = QVBoxLayout()
        vertical_layout.addWidget(self.canvas)

        self.canvas.axes = self.canvas.figure.add_subplot(111)
        self.setLayout(vertical_layout)
```

UML diagram

The systems described above can be inspected in the following UML diagram: