Differential equations

Computational practicum

Alla Chepurova

November 2019

Problem statement:

$$\begin{cases} y' = sec(x) - ytan(x) \\ y(0) = 1 \\ x \in (\pi/2, 3\pi/2) \end{cases}$$

Exact solution of the Initial Value Problem

The type of equation - first-order linear ordinary differential equation.
Chosen solving method - Bernoulli method.

$$y' = \frac{1}{cos(x)} - y * \frac{sin(x)}{cos(x)} \quad (cos(x) \neq 0) \to y' - ytg(x) = \frac{1}{cos(x)} \Rightarrow y(x) = u(x) * v(x) \Rightarrow y' = v'u + uv'$$

$$y(x) = u(x) * v(x) \Rightarrow y' = v'u + uv' \Rightarrow v'u + uv' - uvtg(x) = \frac{1}{cos(x)}$$

$$v'u + uv' - uvtg(x) = \frac{1}{cos(x)} \Rightarrow u'v + u(v' - vtg(x)) = \frac{1}{cos(x)} \Rightarrow u'v + u(v' - vtg(x)) = \frac{1}{cos(x)}$$

$$Let's\ set\ v' - vtg(x) = 0 \Rightarrow v' - vtan(x) = 0 \to v' = vtg(x)$$

$$\frac{dv}{dx} = vtg(x) \to \frac{dv}{v} = tan(x)dx \to \int \frac{dv}{v} = \int tan(x)dx \to v = C1 * cos(x)$$

$$Let\ set\ C1 = 0\ \to v = cos(x) \to y = uv \to y = cos(x) * u \to u'cos(x) = \frac{1}{cos(x)}$$

$$\frac{du}{dx} = \frac{1}{cos(x)} \to u = C + tan(x) \Rightarrow y = C * cos(x) + sin(x)\ \ (But\ cos(x) \neq 0!)$$

$$\to\ Solution\ of\ IVP\ is:\ C = 1 \Rightarrow y = cos(x) + sin(x)$$

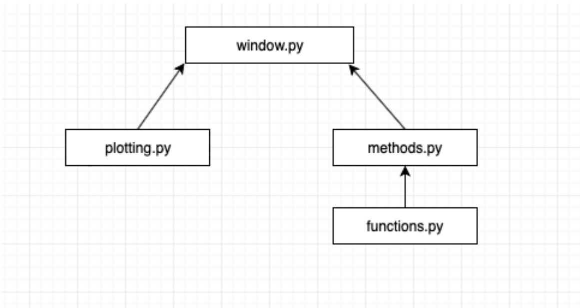
There are no points of discontinuity in a given range.

Implementation

I choose Python language for implementation, with 2 libraries imported - MatPlotlib and Tkinter, first for plotting graphs and second one for GUI.

Program Structure

for structuring, I divided the code into 4 modules: *window.py, plotting.py, methods.py, functions.py.*



There are corresponding functions in each of them:

plotting.py × window.py × methods.py × functions.py ×

```
from math import*

def f(x, y):...

def const(x, y):...

def y(x, x0, y0):...
```

plotting.py × window.py × methods.py × functions.py ×

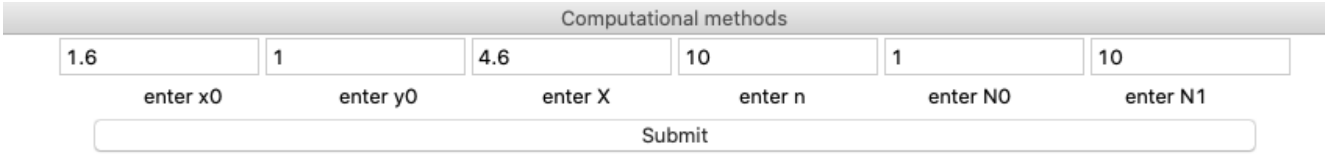
```
from functions import*

def exact_solution(x0, y0, X, n):...


def euler_method(x0, y0, b, n):...

def improved_euler_method(x0, y0, b, n):...

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

def compute_error(method1, method2):...

def global_error(x0, y0, X, n1, n2):...

```

window.py × plotting.py × methods.py × functions.py ×

```
import ...


def plotMethods(exact, euler, improved, runge_kutta, f, a, dataPlot, x0, X):...

def plotErrors(error1, error2, error3, f, a, exact, dataPlot, x0, X):...

def plot_total_errors(error1, error2, error3, f, a, dataPlot, n0, n1):...

```

Module *window.py* contains Tkinter objects for GUI building and function *input()* for getting input, which, in turn, firstly call the functions from *methods.py* to obtain arrays of values of corresponding methods and errors and secondly call the functions from *plotting.py* to plot relevant graphs on the GUI window using values in array obtained previously.

GUI



GUI provides the possibility to input values of x0, y0, X, n, N0, N1 (two last are grid sizes to plot the graph of errors in dependence of N).

Plotting

#1. plotMethods(exact, euler, improved, runge_kutta, f, a, dataPlot, x0, X)

Responsible for plotting the methods for solving the DE and the function y(x) (exact solution of DE) itself. Parameters are arrays gotten by corresponding functions, MatPlotlib objects already placed on the Tkinter canvas in the module *window.py*.

```
def plotMethods(exact, euler, improved, runge_kutta, f, a, dataPlot, x0, X):
    a.clear()
    a.grid()
    a.set_xlabel('X', fontsize = 5)
    a.set_ylabel('Y', fontsize = 5)

    line1, = a.plot(exact[0], exact[1], linewidth = 0.7)
    line2, = a.plot(euler[0], euler[1], linewidth = 0.7)
    line3, = a.plot(improved[0], improved[1], linewidth = 0.7)
    line4, = a.plot(runge_kutta[0], runge_kutta[1], linewidth = 0.7)

    a.legend([line1, line2, line3, line4],
             ['Exact solution', 'Euler method', 'Improved Euler method', 'Runge-Kutta method'], loc=4, prop={'size': 4})
    axes = f.gca()
    axes.set_xlim([x0, X])
    a.tick_params(axis='both', which='major', labelsize=3)
    a.set_xlabel('X', fontsize = 5)
    a.set_ylabel('Y', fontsize = 5)
    a.set_title("Solutions")
    dataPlot.show()
    dataPlot.get_tk_widget().pack(side=LEFT, fill=BOTH, expand=1)
```

drawing area preparation

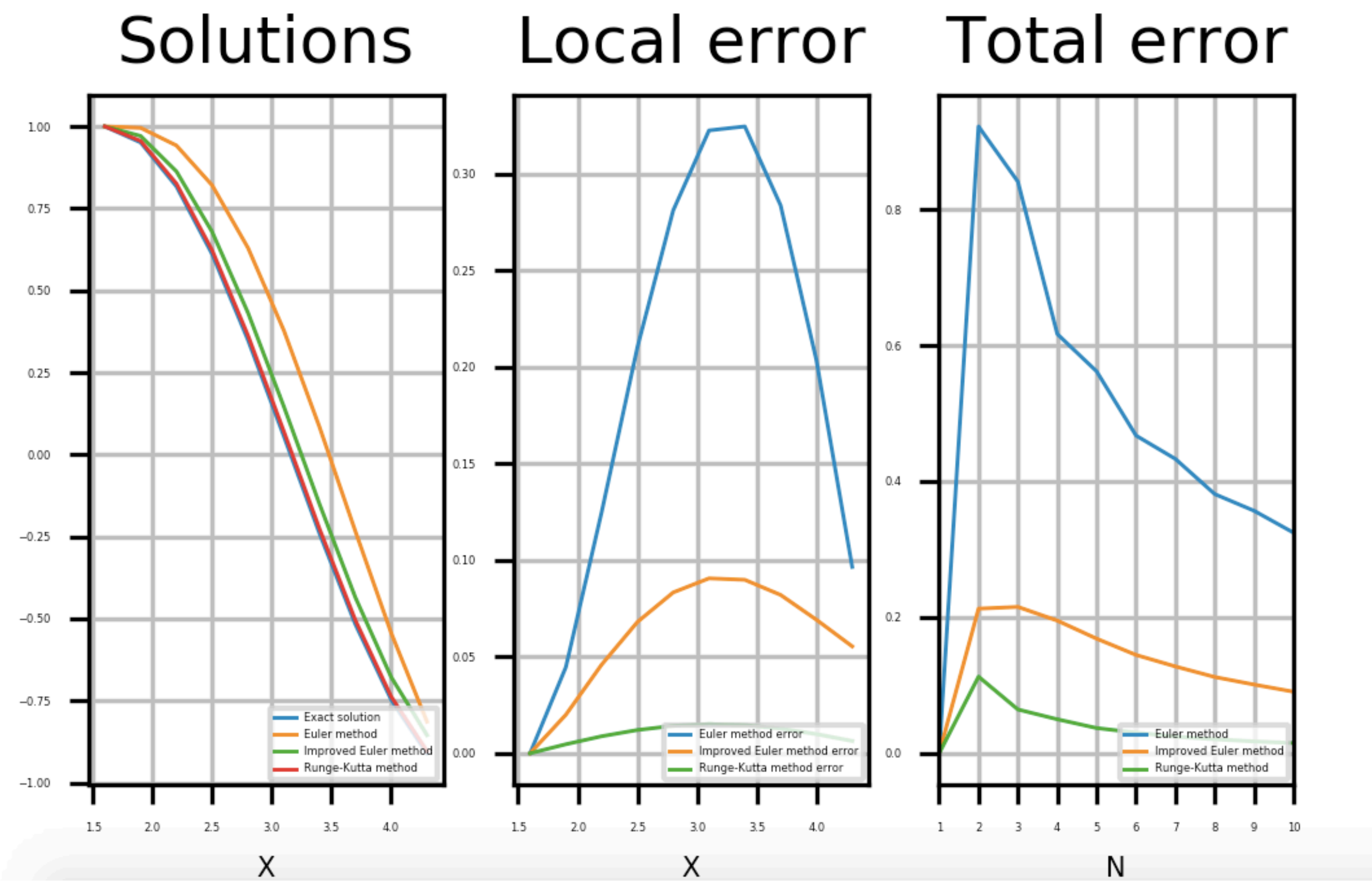
plotting the graphs

making it more readable

#2. plotErrors(error1, error2, error3, f, a, exact, dataPlot, x0, X) and
#3. plot_total_errors(error1, error2, error3, f, a, dataPlot, n0, n1):

Has the same structure as function *plotMethods(exact, euler, improved, runge_kutta, f, a, dataPlot, x0, X).*

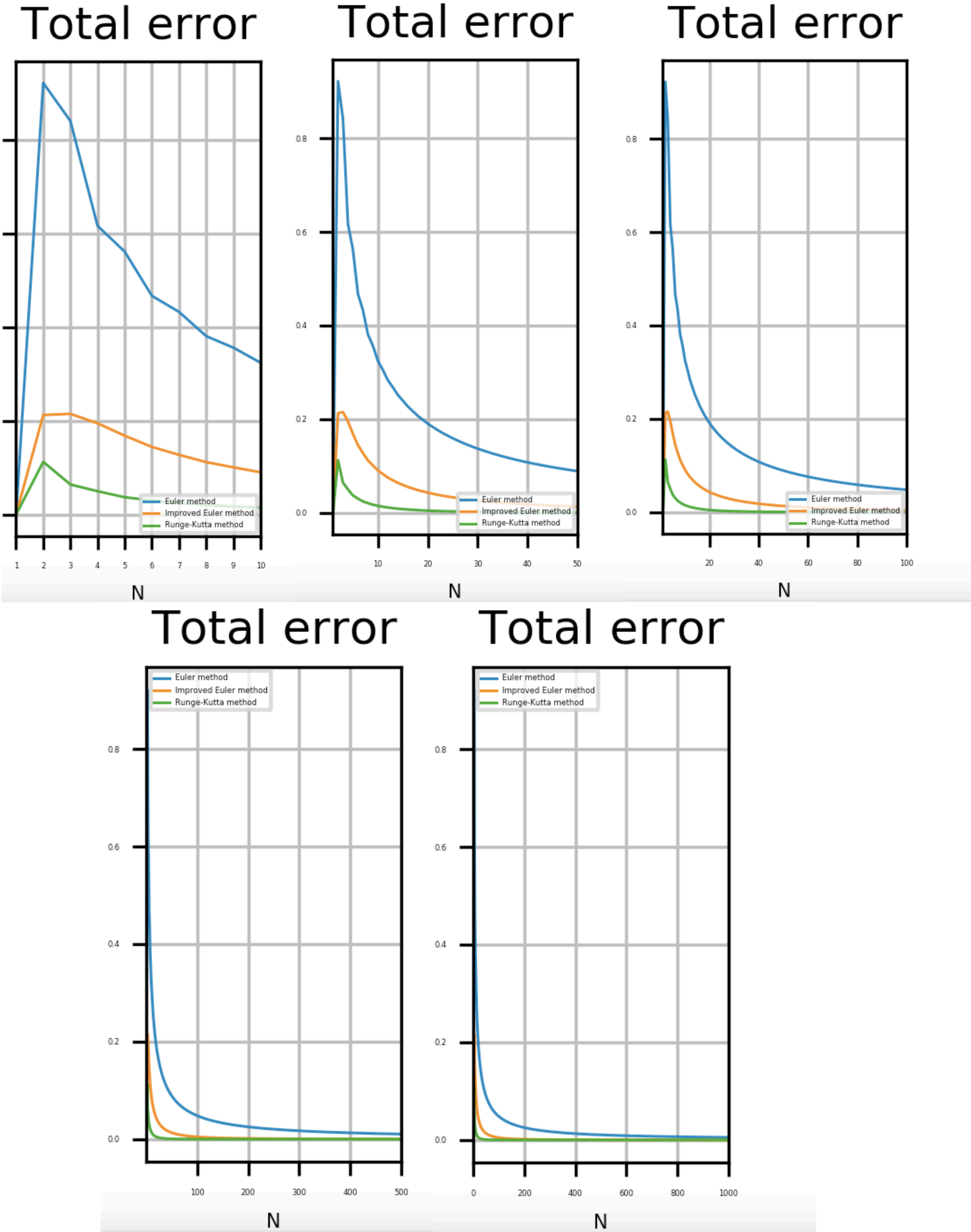
The graph examples of #1, #2, #3 respectively (input values are condition of IVP, N0 = 1, N1 = 10):



Convergence

Testing my program on different grid numbers showed that the value of error of each method tends to zero as grows, but the rate of error decreasing is different for the methods.

The third graph shows the dependence of maximum error for each method on the number of grid cells (N). Graphs for IVP in my task and N1 =1 and N2 = 10, 50, 100, 500, 1000 respectively are:


Total error
Euler method
Improved Euler method
Runge-Kutta method
N
0.0
0.2
0.4
0.6
0.8
1 2 3 4 5 6 7 8 9 10
10 20 30 40 50
20 40 60 80 100
100 200 300 400 500
0 200 400 600 800 1000