Differential Equations Assignment Report

Author: Temur Kholmatov

Group: B17-05

Variant: #25

O.D.E. and initial conditions: $\begin{cases} y' = xy^2 - 3xy \\ y(0) = 2 \\ x \in [0, 6.4] \end{cases}$

GitHub: link

Content:

1. Introduction

This project is created to demonstrate the usage and precision of such numerical methods like Euler, Improved Euler and Runge-Kuffa methods in plotting the graph of ordinary differential equations. Here I try to plot the graph of the function represented by the function $y' = xy^2 - 3xy$ on [0, 6,4] range and the initial value $y(0) = 2$ (the default grid step is put to 0.1). For such purpose, I implemented a GUI application drawing the functions of numerical solutions, their local errors w.r.t. the exact solution ($e_i = |y_i - y(x_i)|$) and total approximation errors of numerical solutions w.r.t. the exact solution depending on the grid size ($E(N) = \max(e_i \text{ when } h = \frac{X-x0}{N})$). Additionally, there is an opportunity to change the initial conditions of the o.d.e. and show the other graph. More information about the application is given in section 3.

2. Analytical Solution

$$y' = xy^2 - 3xy$$

$$\text{This is a Bernoulli equation}$$

$$y' + 3xy = xy^2 \mid : y^2$$

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

$$\text{Let's use substitution}$$

$$z = \frac{1}{y}, \qquad z' = -\frac{y'}{y^2}$$

$$-z' + 3xz = x$$

Method of variation of parameter

$$-z' + 3xz = 0$$

$$\frac{dz}{z} = 3xdx$$

$$\int \frac{dz}{z} = \int 3xdx$$

$$\log|z| = \frac{3}{2}x^2 + \log u$$

$$z = ue^{\frac{3}{2}x^2}$$

$$z' = u'e^{\frac{3}{2}x^2} + 3xe^{\frac{3}{2}x^2}u$$

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

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

$$u = -\int \frac{xdx}{e^{\frac{3}{2}x^2}} = \left| \begin{matrix} v = \frac{3}{2}x^2 \\ dv = 3xdx \end{matrix} \right| = -\frac{1}{3}\int \frac{dv}{e^v} = \frac{1}{3e^v} + C$$

$$= \frac{1}{3e^{\frac{3}{2}x^2}} + C$$

$$z = \frac{1}{3} + Ce^{\frac{3}{2}x^2}$$

$$y = \frac{1}{\frac{1}{3} + Ce^{\frac{3}{2}x^2}}, \qquad y(0) = 2$$

$$2 = \frac{1}{\frac{1}{3} + C}, \qquad C = \frac{1}{6}$$

Solution:

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

3. Information about the application

The source code of the application can be found via this link.

This is a GUI application created on Python with the use of matplotlib, numpy, pandas, tkinter and pandastable libraries. Using it you can check the graph of function for some specific differential equation as well as the error function. In this project the equation is $y' = xy^2 - 3xy$ (variant #25).

Clone this repository to your machine using Git:

```
$ git clone https://github.com/temur-kh/Differential-Equations.git
```

In order to run the application, you should install all required packages. You can do it (not necessary step) primarily creating and activating a virtual environment:

```
$ pip install virtualenv
```

```
$ virtualenv venv
```

```
$ source venv/bin/activate
```

Install all requirements from requirements.txt file:

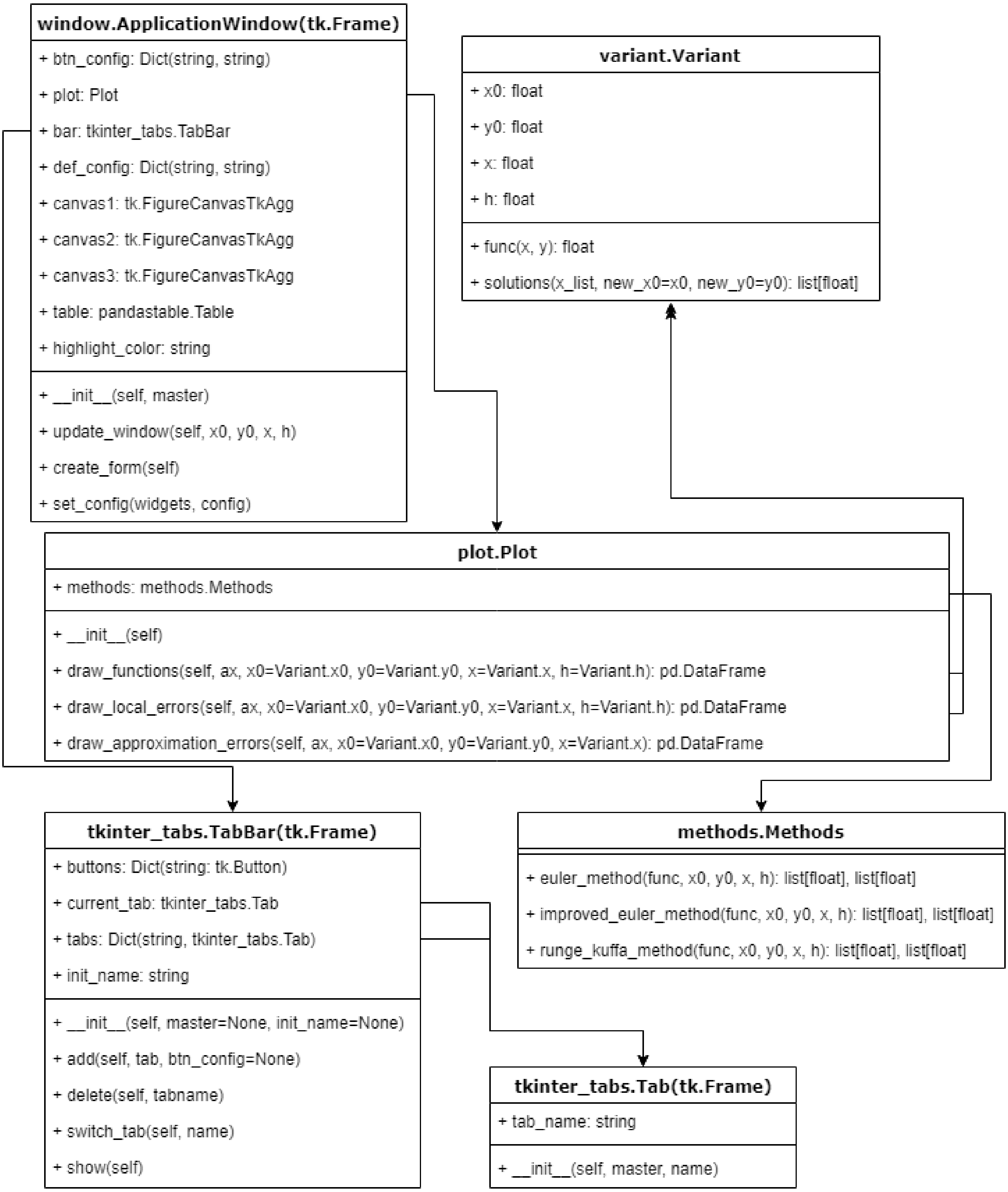
```
$ pip install -r requirements.txt
```

Now you can run the application:

```
$ python main.py
```

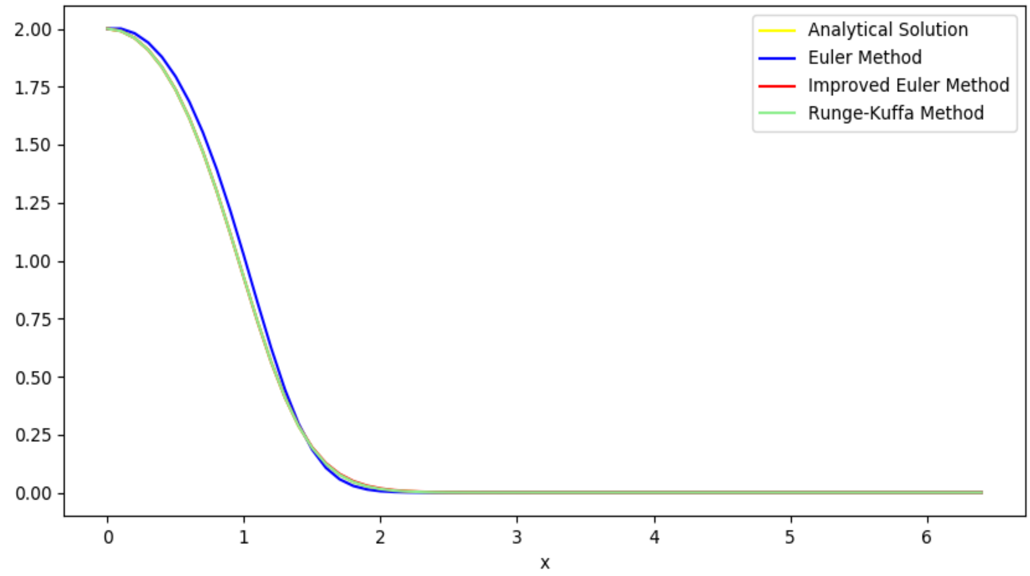
The values of initial conditions are changeable through the form in the window of app. However, to change the differential equation and the function of the analytical solution, you need to change the code.

4. UML diagram of the application

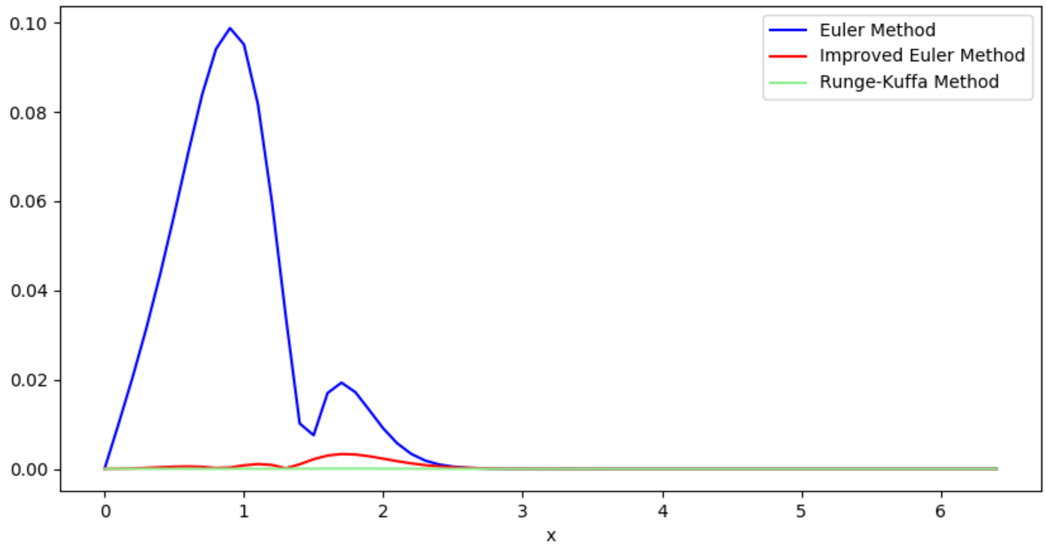

window.ApplicationWindow(tk.Frame)
+ btn_config: Dict(string, string)
+ plot: Plot
+ bar: tkinter_tabs.TabBar
+ def_config: Dict(string, string)
+ canvas1: tk.FigureCanvasTkAgg
+ canvas2: tk.FigureCanvasTkAgg
+ canvas3: tk.FigureCanvasTkAgg
+ table: pandastable.Table
+ highlight_color: string
+ __init__(self, master)
+ update_window(self, x0, y0, x, h)
+ create_form(self)
+ set_config(widgets, config)
variant.Variant
+ x0: float
+ y0: float
+ x: float
+ h: float
+ func(x, y): float
+ solutions(x_list, new_x0=x0, new_y0=y0): list[float]
plot.Plot
+ methods: methods.Methods
+ __init__(self)
+ draw_functions(self, ax, x0=Variant.x0, y0=Variant.y0, x=Variant.x, h=Variant.h): pd.DataFrame
+ draw_local_errors(self, ax, x0=Variant.x0, y0=Variant.y0, x=Variant.x, h=Variant.h): pd.DataFrame
+ draw_approximation_errors(self, ax, x0=Variant.x0, y0=Variant.y0, x=Variant.x): pd.DataFrame
tkinter_tabs.TabBar(tk.Frame)
+ buttons: Dict(string: tk.Button)
+ current_tab: tkinter_tabs.Tab
+ tabs: Dict(string, tkinter_tabs.Tab)
+ init_name: string
+ __init__(self, master=None, init_name=None)
+ add(self, tab, btn_config=None)
+ delete(self, tabname)
+ switch_tab(self, name)
+ show(self)
methods.Methods
+ euler_method(func, x0, y0, x, h): list[float], list[float]
+ improved_euler_method(func, x0, y0, x, h): list[float], list[float]
+ runge_kuffa_method(func, x0, y0, x, h): list[float], list[float]
tkinter_tabs.Tab(tk.Frame)
+ tab_name: string
+ __init__(self, master, name)

5. Screenshots of the results

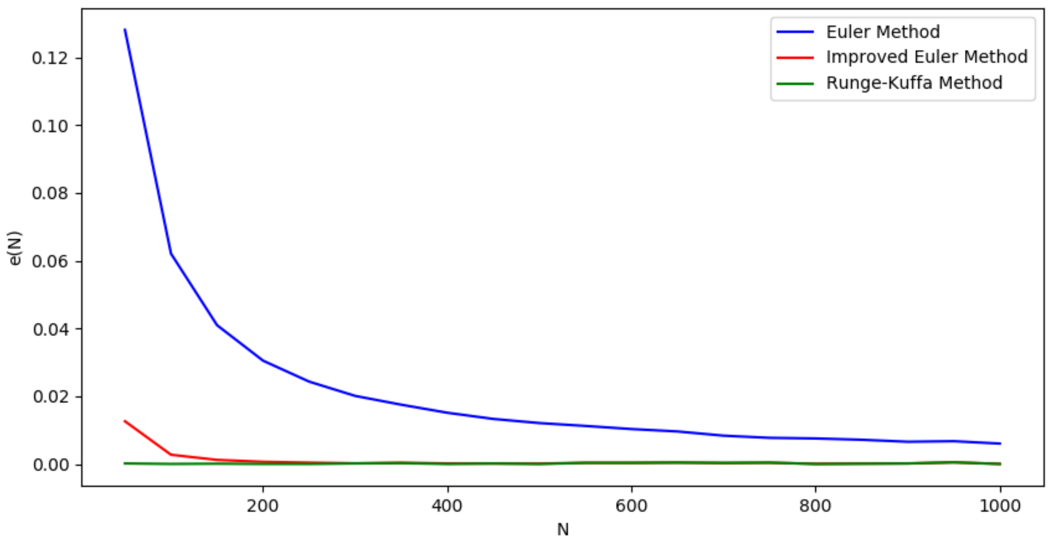
The graph of the function using different numerical methods and the function of the analytical solution



The graph of the function of local errors of different numerical methods comparing to the analytical solution



The graph of the function of total approximation errors of different numerical methods comparing to the analytical solution



The table with the results of different numerical methods and the analytical solution

	x	Analytical Solution	Euler Method	Improved Euler Method	Runge-Kuffa Method
1	0.0	2.0	2.0	2.0	2.0
2	0.1	1.989975125780137	2.0	1.99	1.98997517
3	0.2	1.9596081988013834	1.98	1.95970529	1.95960837
4	0.3	1.908071175970593	1.939608	1.90828031	1.90807159
5	0.4	1.8341617297510011	1.87790566	1.83450563	1.83416251
6	0.5	1.736617918821381	1.79361816	1.73708578	1.73661924
7	0.6	1.6146364155184167	1.68542874	1.61516242	1.61463845
8	0.7	1.468614966607913	1.55249177	1.46906256	1.46861787
9	0.8	1.3010396574400718	1.39518465	1.30121436	1.3010435
10	0.9	1.1172572540974886	1.21606355	1.11697573	1.11726201
11	1.0	0.9256846378913173	1.02081934	0.92490197	0.92569044
12	1.1	0.7370032989770018	0.81878075	0.73593589	0.73701096
13	1.2	0.5622643238269259	0.62232732	0.56139255	0.56227592
14	1.3	0.41049693063788184	0.44476444	0.41038346	0.41051566
15	1.4	0.28686384047890406	0.29702231	0.28788077	0.28689272
16	1.5	0.19215813057674871	0.18462405	0.19431424	0.1921981
17	1.6	0.12364639540917512	0.10665614	0.12662004	0.12369537
18	1.7	0.07660393423818114	0.05728128	0.07991966	0.07665741
19	1.8	0.04579306649837732	0.02862562	0.0490078	0.04584581
20	1.9	0.02646056291030139	0.01331528	0.02927198	0.0265082
21	2.0	0.014799146228146511	0.00575926	0.0170667	0.01483903
22	2.1	0.00801919549119668	0.00231034	0.00973098	0.00805044
23	2.2	0.004212724213146776	0.00085595	0.00543476	0.0042358
24	2.3	0.0021464150255688865	0.00029118	0.00297768	0.00216257
25	2.4	0.0010609460785248467	9.029e-05	0.00160284	0.00107172
26	2.5	0.0005088230965246155	2.528e-05	0.0008489	0.00051569
27	2.6	0.00023679412499200223	6.32e-06	0.00044302	0.00024098
28	2.7	0.00010693641239060816	1.39e-06	0.00022818	0.00010939
29	2.8	4.686421630501705e-05	2.6e-07	0.00011617	4.824e-05
30	2.9	1.993074239366642e-05	4e-08	5.856e-05	2.068e-05
31	3.0	8.225731964020555e-06	1e-08	2.928e-05	8.62e-06
32	3.1	3.294547828703518e-06	0.0	1.454e-05	3.49e-06
33	3.2	1.2805242960448408e-06	0.0	7.19e-06	1.38e-06
34	3.3	4.830041014218233e-07	0.0	3.55e-06	5.3e-07
35	3.4	1.76801077959469e-07	0.0	1.75e-06	2e-07
36	3.5	6.280440945367003e-08	-0.0	8.6e-07	7e-08
37	3.6	2.1650429637886215e-08	0.0	4.3e-07	3e-08
38	3.7	7.242926256111083e-09	-0.0	2.1e-07	1e-08
39	3.8	2.3514338837750537e-09	0.0	1.1e-07	0.0
40	3.9	7.408369680111272e-10	-0.0	5e-08	0.0
41	4.0	2.265080726396442e-10	0.0	3e-08	0.0
42	4.1	6.720721347419445e-11	-0.0	1e-08	0.0
43	4.2	1.9351706081006698e-11	0.0	1e-08	0.0
44	4.3	5.407465966976939e-12	-0.0	0.0	0.0
45	4.4	1.4663562544398593e-12	0.0	0.0	0.0
46	4.5	3.85883636487071e-13	-0.0	0.0	0.0
47	4.6	9.854722393881577e-14	0.0	0.0	0.0
48	4.7	2.4423257734318323e-14	-0.0	0.0	0.0
49	4.8	5.874000184222209e-15	0.0	0.0	0.0
50	4.9	1.3709937805528883e-15	-0.0	0.0	0.0
51	5.0	3.105333003481127e-16	0.0	0.0	0.0
52	5.1	6.825776108430344e-17	-0.0	0.0	0.0
53	5.2	1.4560190742379427e-17	0.0	0.0	0.0
54	5.3	3.0140694521507523e-18	-0.0	0.0	0.0
55	5.4	6.054950707110851e-19	0.0	0.0	0.0
56	5.5	1.180426998450933e-19	-0.0	0.0	0.0
57	5.6	2.233257605366299e-20	0.0	0.0	0.0
58	5.7	4.1002437456553694e-21	-0.0	0.0	0.0
59	5.8	7.305529411761561e-22	0.0	0.0	0.0
60	5.9	1.2631789440687133e-22	-0.0	0.0	0.0
61	6.0	2.11957714332049e-23	0.0	0.0	0.0
62	6.1	3.451475086899807e-24	-0.0	0.0	0.0
63	6.2	5.45420458149956e-25	0.0	0.0	0.0
64	6.3	8.364293627613338e-26	-0.0	0.0	0.0
65	6.4	1.244796281328927e-26	0.0	0.0	0.0

6. Conclusion

According to the results, the most accurate numerical method is the Runge-Kuffa method whereas the most inaccurate one is the Euler method. The function does not have any points of discontinuity and, since it is decreasing on the interval [0, 6.4], the local error functions are also decreasing.