Computational Practicum

Jaffar Totanji

October 2021

Abstract

The goal of this assignment is to find the exact solution of a given initial value problem and to construct a corresponding approximation of the given problem using Euler's, Improved Euler's, and Runge-Kutta methods. Then to compare the findings and errors to decide which approximation is the most accurate.

1 Finding the Exact Solution

$$y' = f(x, y) = \frac{4}{x^2} - \frac{y}{x} - y^2 \tag{1}$$

The initial values are:

$$x_0 = 1$$
$$y_0 = 0$$
$$X = 7$$

Where x cannot be null as it occurs in the denominator.

Equation (1) is a nonlinear non homogeneous equation differential equation which falls into Riccati equations of the format:

$$y' + a(x)y + b(x)y^2 = c(x)$$

And, to solve equation (1), we need to find any particular solution y_1, then use the substitution $y = y_1(x) + z(x)$.
To find y_1 we can see that the particular solution is of the format $y1(x) = \frac{C}{x}$, since both functions $a(x)$ and $c(x)$ follow this format or its derivative.

Now to find C, we substitute y_1 in equation (1):

$$y_1' = \frac{4}{x^2} - \frac{y_1}{x} - y_1^2$$

$$-\frac{C}{x^2} = \frac{4}{x^2} - \frac{C}{x^2} - \frac{C^2}{x}$$

$$C^2 = 4$$
$$C = \pm 2$$

Choosing $C = -2$ and substituting $y = y_1 + z$ in equation (1):

$$(y_1 + z)' = \frac{4}{x^2} - \frac{(y_1 + z)}{x} - (y_1 + z)^2$$
$$(\frac{-2}{x^2})' + z' = \frac{4}{x^2} - \frac{(\frac{-2}{x} + z)}{x} - (\frac{-2}{x} + z)^2$$
$$\frac{2}{x^2} + z' = \frac{4}{x^2} + \frac{2}{x^2} - \frac{z}{x} - \frac{4}{x^2} - z^2 + \frac{4z}{x}$$
$$z' - \frac{3z}{x} = -z^2 \tag{2}$$

We can see that the equation (2) is a nonlinear first order ordinary differential equation, which has the format of Bernoulli equations which is:

$$z' + g(x)z = f(x)z^k$$

where

$$g(x) = \frac{-3}{x}$$
$$f(x) = -1$$
$$k = 2$$

To solve equation (2), we first need to find any non-trivial solution z_c of the complementary equation:

$$z' + g(x)z = 0$$

Then the solution of equation (2) would be $z = uz_c$, where $u(x)$ can be found using the following equation:

$$\frac{du}{u^k} = f(x)z_c^{k-1}dx \tag{3}$$

Now we find z_c:

$$z' - \frac{3z}{x} = 0$$
$$z' = \frac{3z}{x}$$
$$\int \frac{dz}{z} = \int \frac{3dx}{x}$$
$$\ln(z) = 3\ln(x) + K$$

We can choose $K = 0$ since we need any non-trivial solution for z_c:

$$z_c = x^3$$

Now to find u, we substitute in the equation (3):

$$\begin{aligned} \int \frac{du}{u^2} &= \int -x^3 dx \\ -\frac{1}{u} &= -\frac{x^4 + D}{4} \\ u &= \frac{4}{x^4 + D} \end{aligned}$$

To find the solution of equation (2), we substitute in the following equation:

$$z = uz_c$$

Substituting u and z_c:

$$z = \frac{4x^3}{x^4 + D}$$

Now we write the solution of equation (1) using the previous equation:

$$y = y_1(x) + z(x)$$

Substituting y_1 and z we get:

$$y = \frac{-2}{x} + \frac{4x^3}{x^4 + D} \tag{4}$$

The constant C can be found substituting the initial values and solving equation (4) for D:

$$D = x_0^4(\frac{2 - x_0 y_0}{x_0 y_0 + 2})$$

We can see that any initial conditions should satisfy $x_0 y_0 \neq -2$ and that y is defined when $x \neq 0$ and when $x^4 + D \neq 0$, and when substituting the constant D:

$$x_0^4 + x_0^4(\frac{2 - x_0 y_0}{x_0 y_0 + 2}) \neq 0$$

dividing by x^4:

$$\frac{4}{x_0 y_0 + 2} \neq 0$$

Which is always true assuming the previously mentioned conditions.

The final answer is:

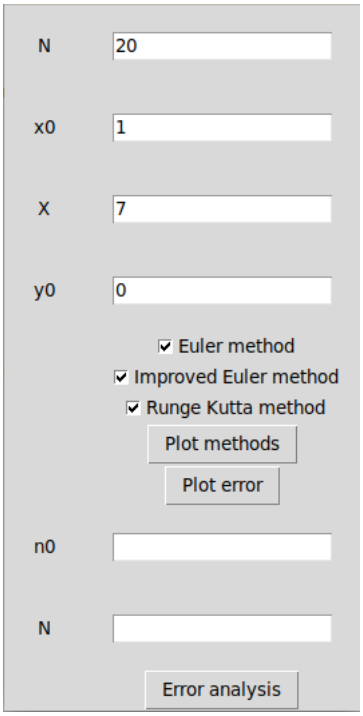
$$y = \frac{-2}{x} + \frac{4x^3}{x_0^4 + x_0^4(\frac{2-x_0y_0}{x_0y_0+2})}$$

$$on\{x|x \in R \ \& \ x_0 \in R \ \& \ y_0 \in R \ \& \ x \neq 0 \ \& \ x_0y_0 \neq -2\}$$

2 Overview of the application

The application has been made using Python, with the help of external libraries.

The main window of the application is responsible for collecting configuration from the user.



The first 4 text boxes are used to specify $N, x0, X, y0$:

- $x0, y0$: The initial values for the given equation.
- X: The end point for the given x interval.
- N: The number of steps that the methods will perform.

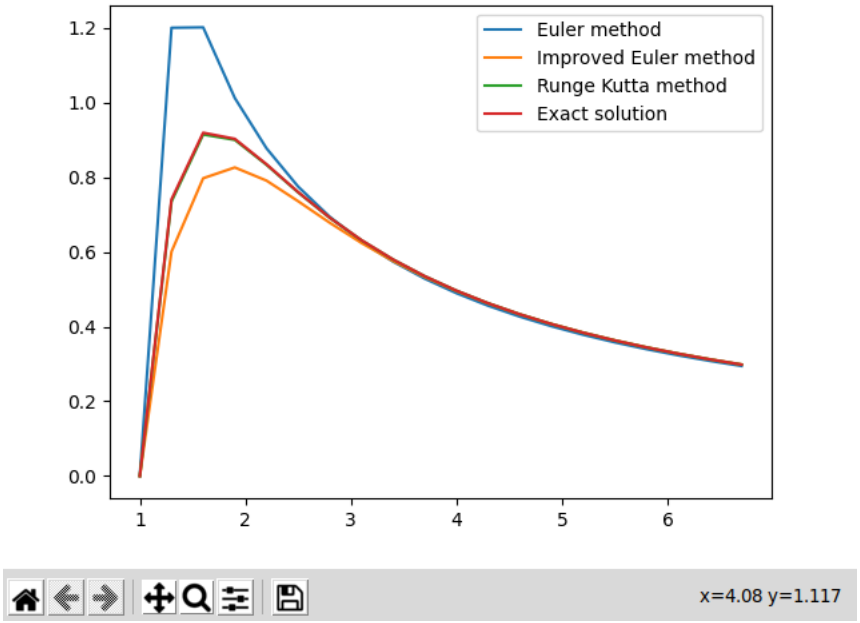
The next two boxes (Plot methods, Plot error) are used to show plotting of the approximations as well as the exact solution and to plot the error function respectively.

The last 2 test boxes are used to specify $n0$ and N:

- $n0$: Starting number of intervals.
- N: : Ending number of intervals.

The last button is used to show plotting of the comparison of the error function of the methods on from $n0$ to N.

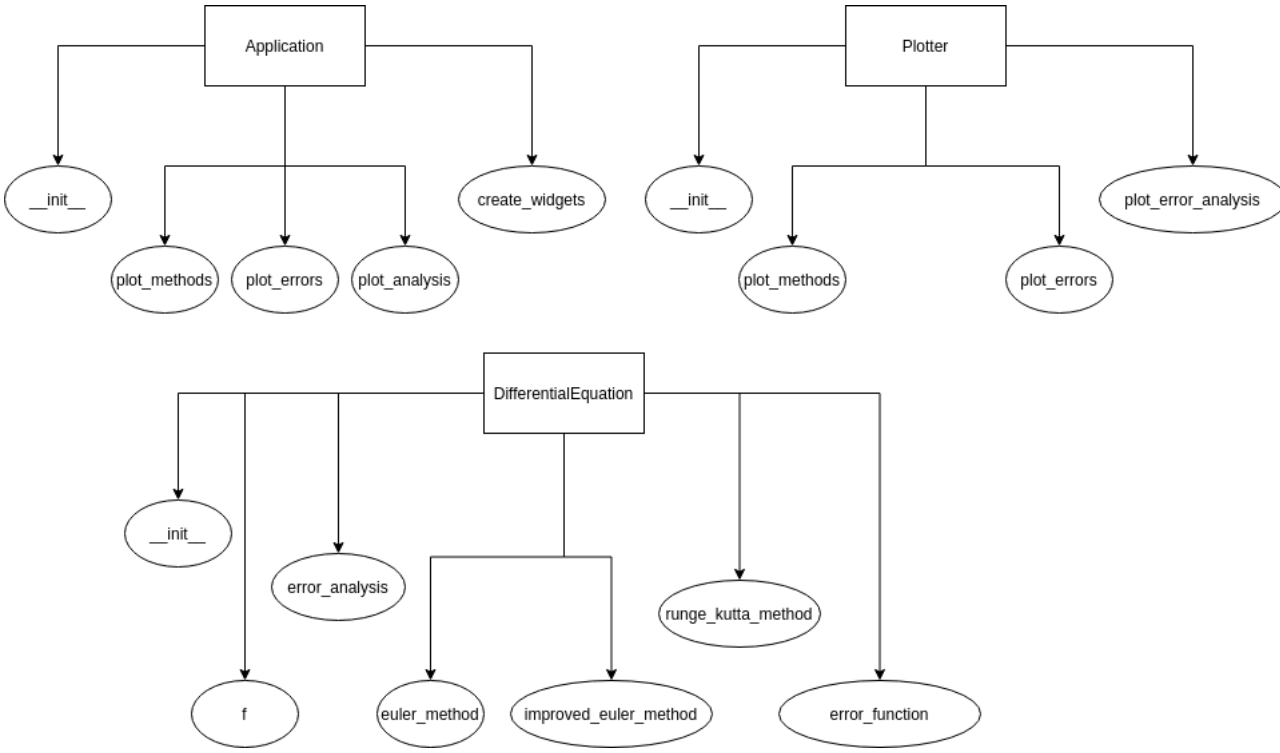
The plotting window is displayed after clicking on one of the buttons:



It contains the plot of the chosen functionality, the lines are shown in different colors and a legend is provided. The user can zoom in and out and move around in the plot.

3 Code Analysis

The UML diagram below describes the classes of the code:



Three classes are used in the code: Application, Plotter and DifferentialEquation.

- The Application class is used to display the main window. It contains the function create_widgets which is responsible of creating and displaying the widgets on the main window (textboxes, checkboxes and buttons):

```
def create_widgets(self):
    self.label1 = tk.Label(self, text="N")
    self.n = tk.Entry(self, width=20)
    self.n.insert(0,'20')
    self.label1.grid(row=0, pady=20, padx=20)
    self.n.grid(row=0, column=1, padx=20)

    self.label2 = tk.Label(self, text="x0")
    self.x0 = tk.Entry(self, width=20)
    self.x0.insert(0,'1')
    self.label2.grid(row=1, pady=20)
```

```
self.x0.grid(row=1, column=1)
...
```

It also contains 3 functions used as callback functions for the 3 available buttons, they are responsible of initializing the plotter class and passing the user given settings:

```
def plot_methods(self):
    self.plotter = Plotter(int(self.n.get()),
    int(self.x0.get()), int(self.X.get()), int(
        ↪ self.y0.get()), bool(self.eulerVar.get
        ↪ ()),bool(self.improvedEulerVar.get()),
    bool(self.rungeKuttaVar.get()))
    self.plotter.plot_methods()

def plot_errors(self):
    self.plotter = Plotter(int(self.n.get()), int
        ↪ (self.x0.get())
    , int(self.X.get()),int(self.y0.get()),
    bool(self.eulerVar.get()),bool(self.
        ↪ improvedEulerVar.get()),
    bool(self.rungeKuttaVar.get()))
    self.plotter.plot_error()

def plot_analysis(self):
    self.plotter = Plotter(int(self.n.get()),
    int(self.x0.get()), int(self.X.get()), int(
        ↪ self.y0.get()),
    bool(self.eulerVar.get()),bool(self.
        ↪ improvedEulerVar.get()),
    bool(self.rungeKuttaVar.get()))
    self.plotter.plot_error_analysis(int(self.n0.
        ↪ get()),int(self.N.get()))
```

- The Plotter class is implemented using matplotlib, it contains 3 functions each of them is responsible of plotting and showing each of the three functionalities of the app (methods, error function, and error analysis) :

```
def plot_methods(self):
    if self.euler:
        plt.plot(self.DE.euler_method()[0],
        self.DE.euler_method()[1], label="Euler␣
            ↪ method")
    if self.improvedEuler:
```

```
            plt.plot(self.DE.improved_euler_method()
                ↪ [0],
            self.DE.improved_euler_method()[1], label
                ↪ ="Improved Euler method")
        if self.rungeKutta:
            plt.plot(self.DE.runge_kutta_method()[0],
            self.DE.runge_kutta_method()[1], label="
                ↪ Runge Kutta method")
        plt.plot(self.DE.exact_solution()[0],
        self.DE.exact_solution()[1], label="Exact
            ↪ solution")
        plt.legend()
        plt.show()
```

- The DifferentialEquation class is responsible of doing the mathematical calculations of the application. It contains the Differential equation function f, 3 functions to calculate the approximation using the 3 methods, function to calculate the error function of a method passed as a parameter and a function to calculate the error analysis function of a method passed as a parameter:

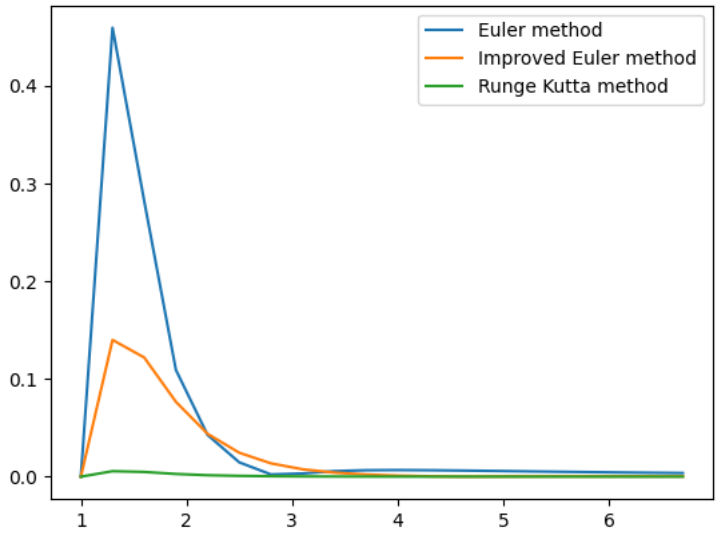
```
def f(self, x, y):
    return 4 / (x * x) - y / x - y * y

def euler_method(self):
    x = [self.a]
    u = [self.c]
    for i in range(1, self.n):
        x += [x[i - 1] + self.h]
        d = self.h * self.f(x[i - 1], u[i - 1])
        u += [u[i - 1] + d]
    return x, u

def error_function(self, exact, approx):
    return exact[0], [abs(exact[1][i]-approx[1][i
        ↪ ]) for i in range(len(exact[1]))]
...
```

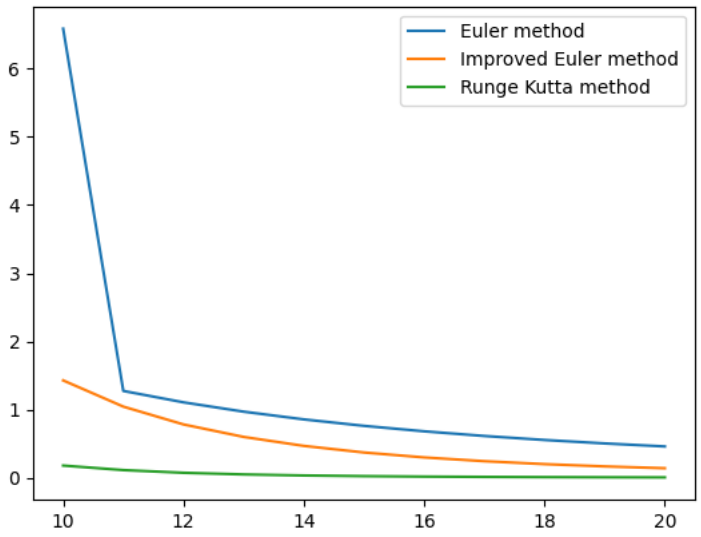
4 Results Analysis

The following graph shows the error function of the three function using the initial parameters:



As noticed the error using Euler is the largest. Using the Improved Euler method shown a great improvement of results. Runge Kutta approximation was the closest to the exact solution.

The following graph shows the error function over a range:



Starting from 10 steps the Euler method has a very large error value, and even at 20 steps it's still showing errors. While Runge Kutta had a very small error value at 10 steps and became extremely small at 12 steps and more.

5 Conclusion

As a result, imperially has been proven that for the given function and continuous functions from different variants Runge-Kutta method performs much better that others achieving relatively small error even on functions that grow exponentially fast.

The srouce code can be found on GitHub.