

Differential Equations Programming Assignment Report

Kamilla Kryachkina
BS17-05

1. Exact solution:

$$y' = (1-2y)e^x + y^2 + e^{2x}, \quad y(-5) = 2$$

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

First-order nonlinear ordinary differential equation

Riccati's equation

$$-\frac{dy(x)}{dx} = -y(x)^2 + 2e^x y(x) - e^{2x} - e^x$$

$$-\frac{dy(x)}{dx} = -(y(x) - e^x)^2 - e^x$$

$$\text{Let } v(x) = -e^x + y(x), \quad y(x) = e^x + v(x), \quad \frac{dy(x)}{dx} = e^x + \frac{dv(x)}{dx}$$

$$-e^x - \frac{dv(x)}{dx} = -v(x)^2 - e^x$$

$$\frac{dv(x)}{dx} = v(x)^2 \quad \text{Divide both sides by } v(x)^2:$$

$$\frac{\frac{dv(x)}{dx}}{v(x)^2} = 1 \quad \text{Integrate both parts with respect to } x:$$

$$\int \frac{\frac{dv(x)}{dx}}{v(x)^2} dx = \int 1 dx$$

$$-\frac{1}{v(x)} = x + C_1$$

$$v(x) = -\frac{1}{x + C_1} \quad \text{Substitute } y(x) = e^x + v(x)$$

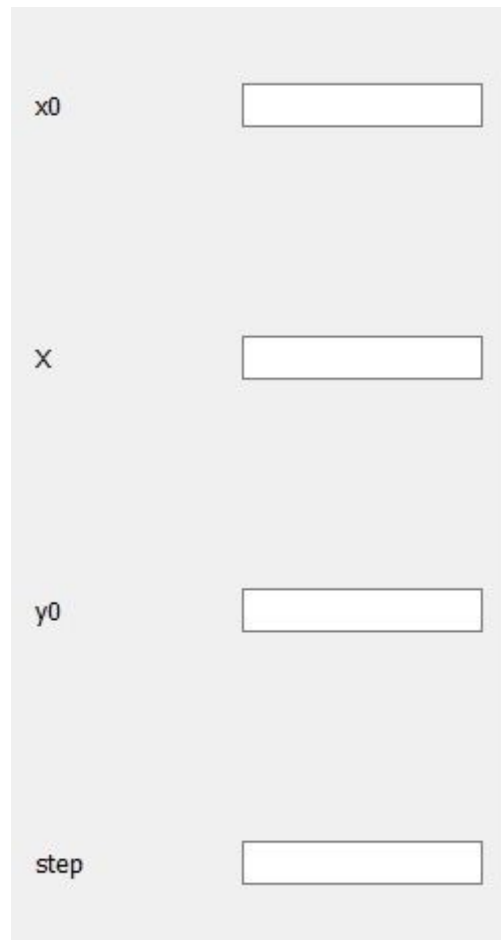
$$y(x) = e^x - \frac{1}{x + C_1} \quad \text{Substitute } y(-5) = 2$$

$$-\frac{1}{C_1 - 5} + \frac{1}{e^5} = 2$$

$$C_1 = \frac{5 - 9e^5}{1 - 2e^5}$$

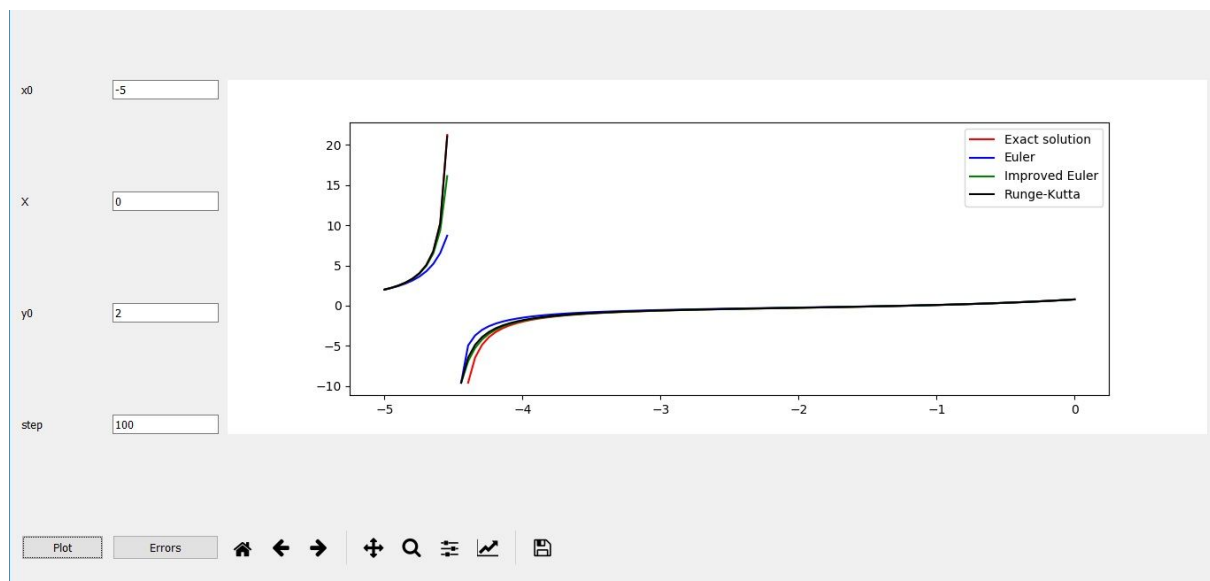
$$\text{Answer: } y(x) = e^x - \frac{1}{x + \frac{-5 + 9e^5}{-1 + 2e^5}}$$

2. I used Python as a programming language, PyQt as a GUI toolkit and matplotlib to plot corresponding graphs. There's an ability to change x_0 , X , y_0 and the number of grid cells:

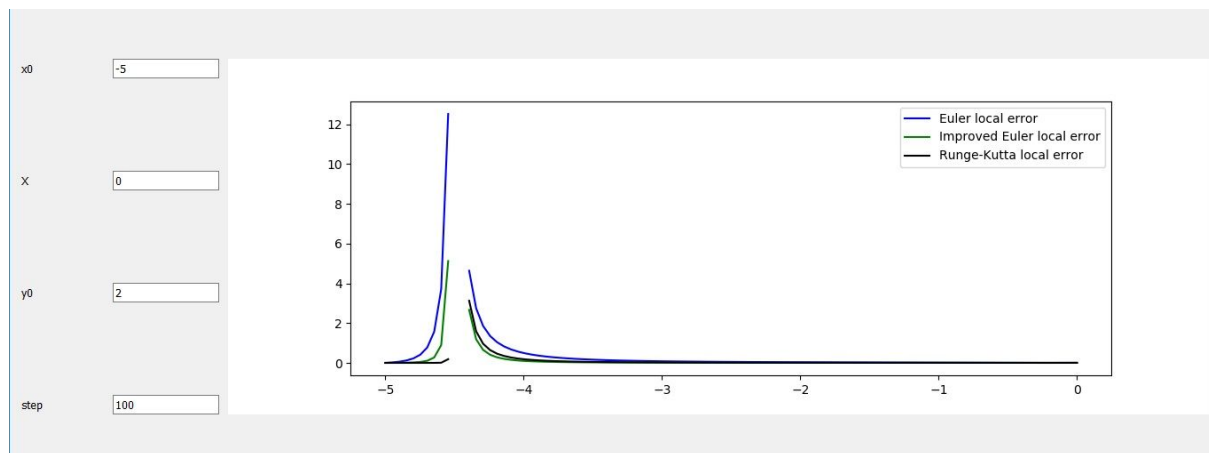


A vertical stack of four input fields with labels x_0 , X , y_0 , and $step$. Each label is to the left of an empty rectangular input box.

My app can plot graphs depending on input values:



3.... and also the local errors of each of three methods:



There's an asymptote and the program can make insufficient errors in the range of it. This is because the value of $y(x)$ becomes NaN at asymptote and can take critically high or low values near this point.

6. Catching double-type overflow to avoid unacceptable $y'(x)$ values:

```
def f(x, y):
    try:
        return (1 - 2*y)*np.exp(x) + y*y + np.exp(2*x)
    except FloatingPointError:
        print("Double overflow raised. Plot cannot be displayed")
        sys.exit(0)
```

Escaping an asymptote locality (range) in all three methods (and also in exact solution):

```
if x + dx*i > -constant - dx/limit and x + dx*i < -constant + dx/limit:
    escape = getProperX(xArr, constant, len(xArr), dx)
    yArr[escape] = yExact[escape]
    if yExact[escape] == np.inf:
        yArr[escape] = yExact[escape+1]
    i = escape + 1
    continue
```

*Euler method's code was taken as an example

getProperX method finds the most appropriate X-value to continue the graph after an asymptote (i.e. not very high and not very low value of Y at such X-value). I used binary search algorithm to improve performance:

```

def getProperX(xArr, const, n, h):
    l, r = -1, n-1
    while (r - l > 1):
        m = (l + r) // 2
        if xArr[m] > -const + h/limit:
            r = m
        else:
            l = m
        #print(l)
        #print(r)

    return r

```

Binary search algorithm.

Class PlotCanvas to introduce geometry (i.e. graph plotting):

```

class PlotCanvas(FigureCanvas):

    def __init__(self, parent=None, width=5, height=4, dpi=100):
        fig = Figure(figsize=(width, height), dpi=dpi)
        FigureCanvas.__init__(self, fig)
        self.ax = self.figure.add_subplot(111)
        self.ax.set_title(r'$(1-2y)e^x + y^2 + e^{2x}$')
        self.setParent(parent)

        FigureCanvas.setSizePolicy(self,
                                   QSizePolicy.Expanding,
                                   QSizePolicy.Expanding)
        FigureCanvas.updateGeometry(self)
        self.plot()

    def plot(self, x=[], y=[], color='', _label=''):
        self.ax.plot(x, y, color, label=_label)
        self.ax.legend()
        self.draw()

```

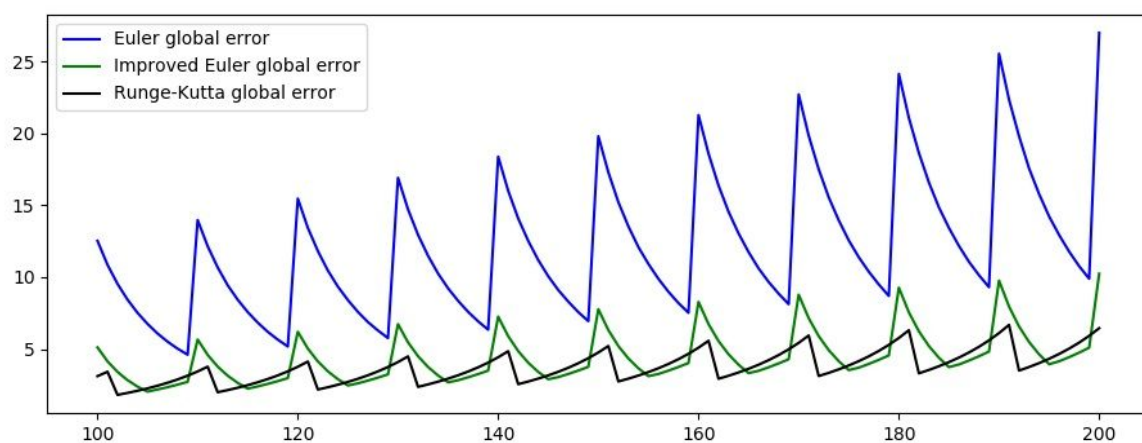
7. My app provides a function to analyze the total approximation error depending on the number of grid cells:

Globals

step range

100

200



Global errors plotting behaves in such a strange way because solution of IVP contains asymptote.

Conclusion

During the work I managed to analyze three numerical methods to solve differential equations - Euler's, Euler's (improved) and Runge-Kutta's. Euler's method is the most erroneous, improved one - a little bit less and the best one is Runge-Kutta's method. However, this method can cause more errors than improved Euler's in some cases. This is because we need to calculate four dependent coefficients in each step of this method, which is also implies calculation errors.