2019

KERIM KOCHEKOV

DIFFERENTIAL EQUATIONS ASSIGMENT – VARIANT 10

https://github.com/KerimKochekov/DE

Part I

The differential equation of the form:

$$y'(x) = -\frac{1}{3}\, y(x)^2 - \frac{2}{3\,x^2}$$

Here is the general solution of the equation:

$$y(x) = -\frac{3\left(\frac{1}{3}\left(\frac{2\,c_1}{c_1 + \sqrt[3]{x}} - 1\right) - 1\right)}{2\,x}$$

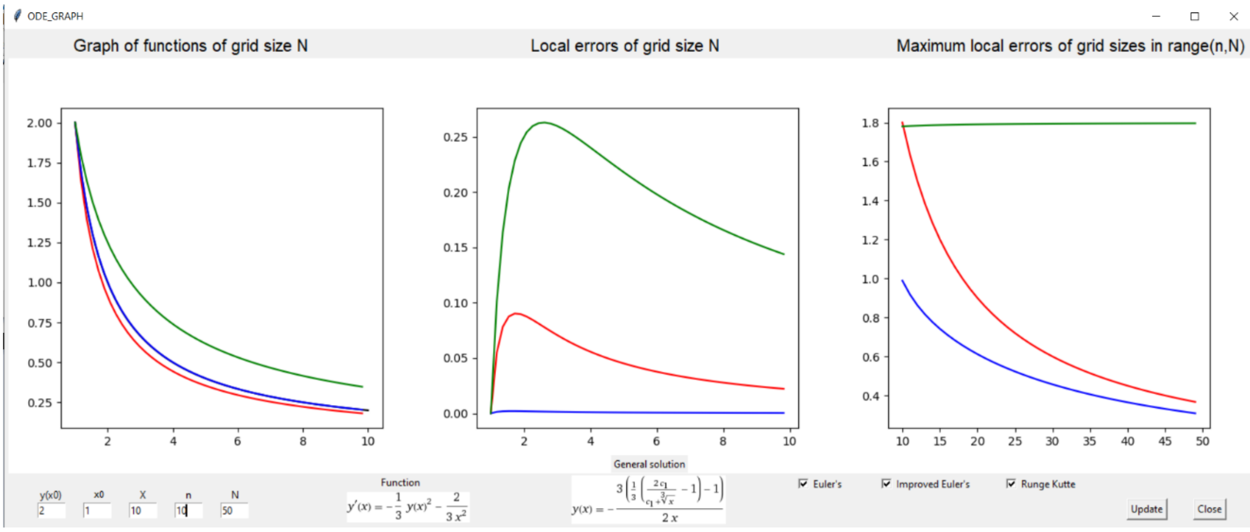
Function has discontinuity at point x = 0.

If we have given IVP, which y (1) = 2 then it forms:

$$y(x) = \frac{2}{x}$$

Part II

Application interface:



In application each technique drawn with different colors:

1) Euler's – red color
2) Improved Euler's – blue color
3) Runge Kutte – green color
4) Exact solution – black color

Used libraries:

1) Numpy
2) Mathplotlib
3) Tkinter
4) Scipy.integrate

Application is fully dynamic, so you can change all values in any way, which you are given 5 input boxes for each variable. Also, we have 3 checkboxes for selecting which technique to draw for us. After every change we have to click button "Update" to see the changes and drawings.

Important parts of code

```
def derivative(y,x):
    x = float("{0:.5f}".format(x))
    return -((y*y)/3)-(2/(3*x*x))
```

Here it is my derivative function, which rounds x directly to 5 digits after point. The reason that after some time dividing by x^2 creates us precision errors.

```
if(self.x0.get()<=0 & 0<=self.X.get()):
    print("Interval contains discontinued point")
    return
```

If our function includes discontinuity point in given range (x0, X) then it directly gives error and returns.

```
 draw.Graph(self.f1,self.x0.get(),self.X.get(),self.y0.get(),self.derivative)
```

I implement another module(draw), which contains computation part of code.

```
def Euler(f1,f2,f3,x0,X,y0,derivative,n,N):
```

When self.m1 checkbox clicked, it calls the Euler function from draw module.

```
def Improved_Euler(f1,f2,f3,x0,X,y0,derivative,n,N):
```

When self.m2 checkbox clicked, it calls the Improved_Euler function from draw module.

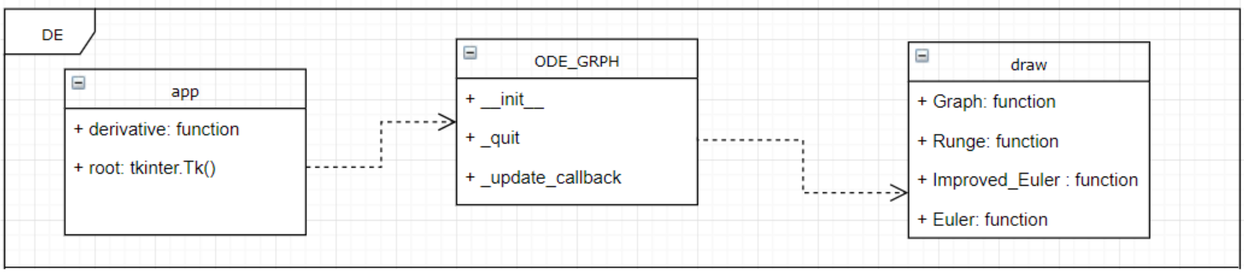
```
def Runge(f1,f2,f3,x0,X,y0,derivative,n,N):
```

When self.m3 checkbox clicked, it calls the Runge function from draw module.

For each technique has its own code, but I would like to show Runge maximum error computation for given range of grid sizes (n,N)

```
    xn,yn = normalize(n,N,1),[]
    for x in xn:
        h,yy,mx = (X-x0)/x,y0,0
        xf = normalize(x0,X,h)
        yf = odeint(derivative,y0,xf)
        for i in range(len(xf)):
            k1 = derivative(yy,xf[i])
            k2 = derivative(yy+h/2*k1,xf[i]+h/2)
            k3 = derivative(yy+h/2*k2,xf[i]+h/2)
            k4 = derivative(yy+h*k3,xf[i]+h)
            yy = float("{0:.5f}".format(yy))
            if abs(yy-yf[i]) > mx:
                mx = abs(yy-yf[i])
        yn.append(mx)
    f3.add_subplot(111).plot(xn,yn,color="green")
```

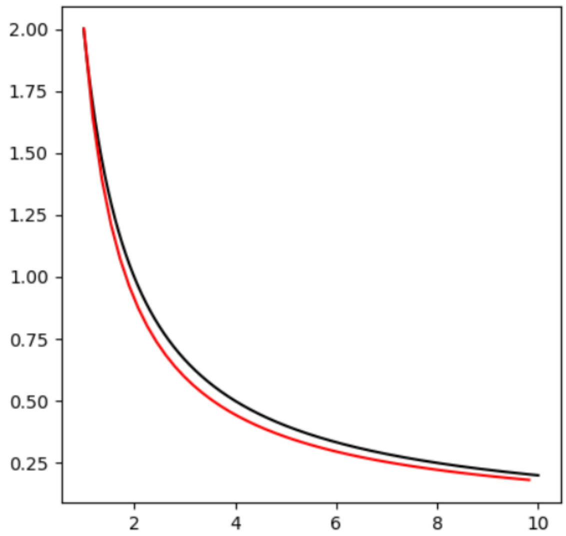
UML

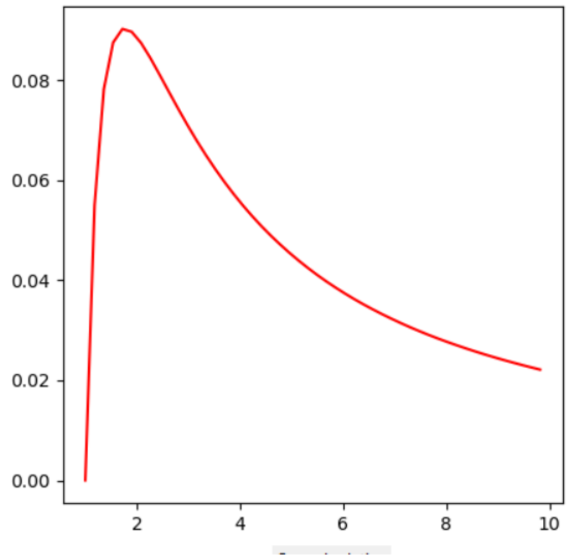


Graphs

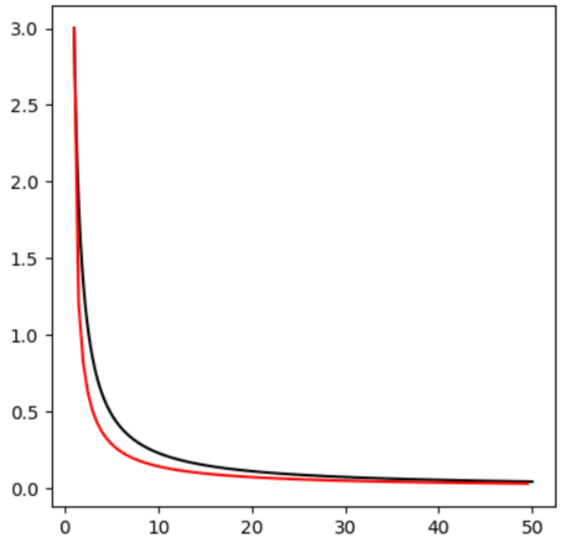
1) Euler's method

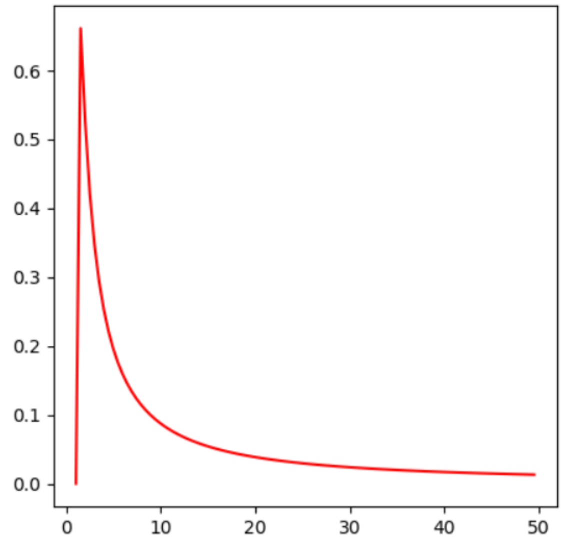
x0 = 1, y(x0) = 2, X = 10, N = 50





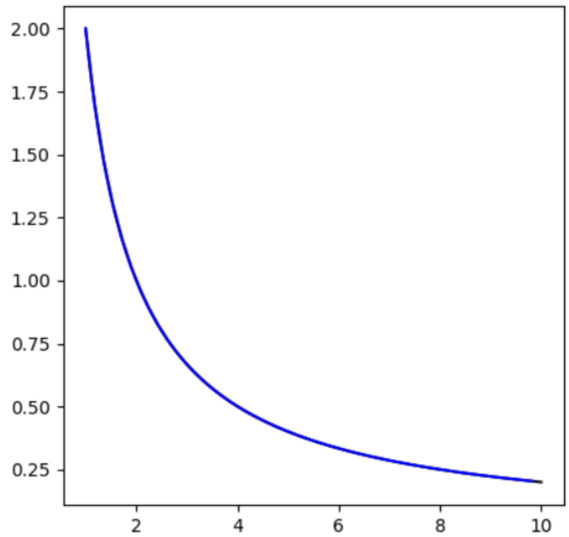
x0 = 1, y(x0) = 3, X = 50, N = 100

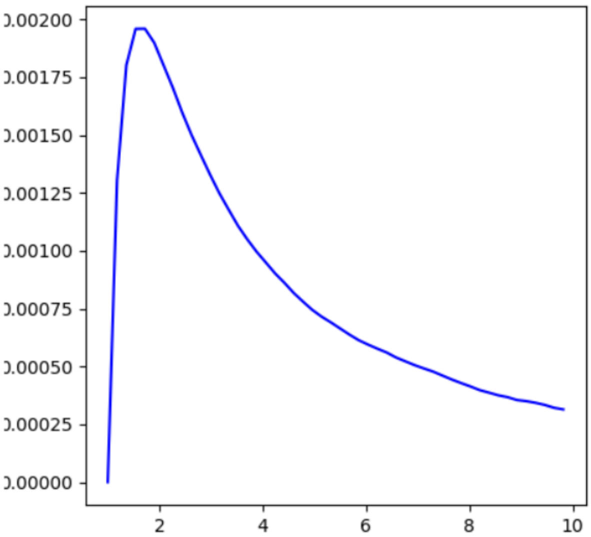




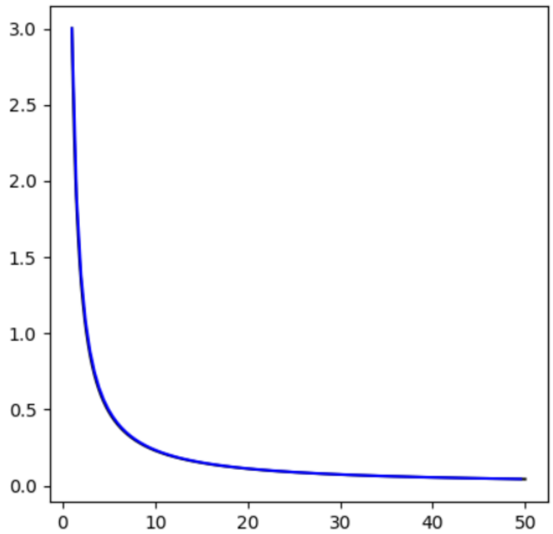
2) Improved Euler's method

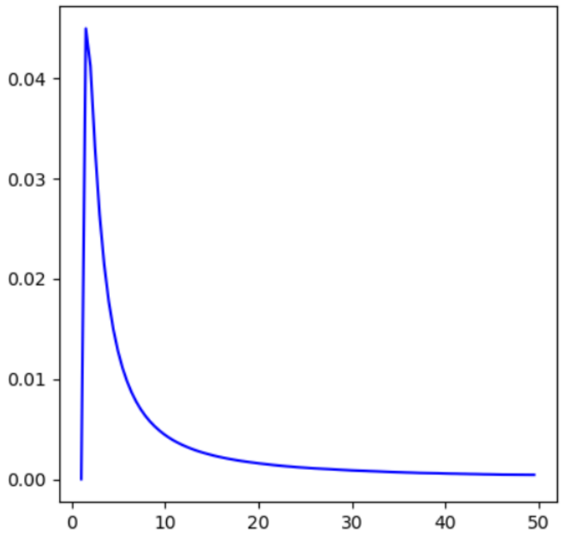
x0 = 1, y(x0) = 2, X = 10, N = 50





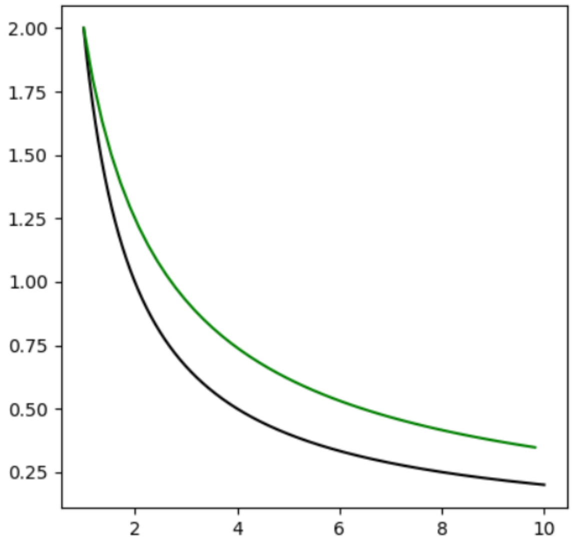
x0 = 1, y(x0) = 3, X = 50, N = 100

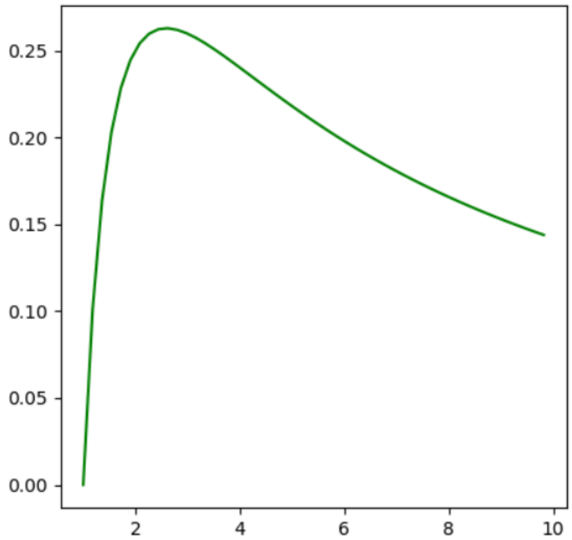




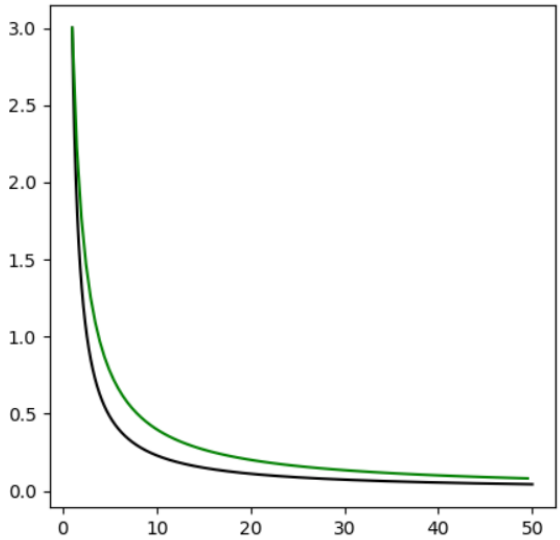
3) Runge Kutte method

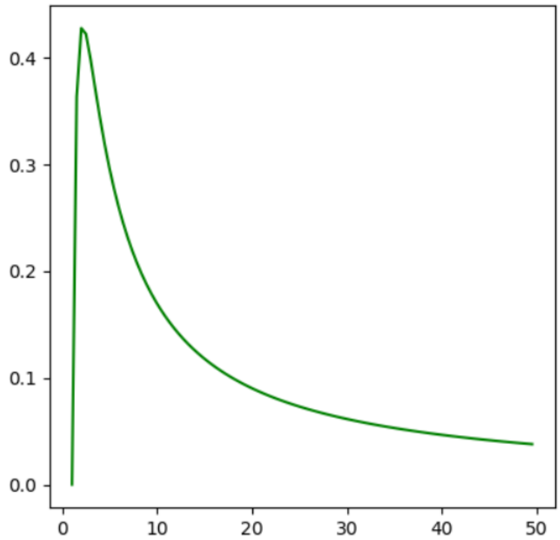
x0 = 1, y(x0) = 2, X = 10, N = 50





x0 = 1, y(x0) = 3, X = 50, N = 100

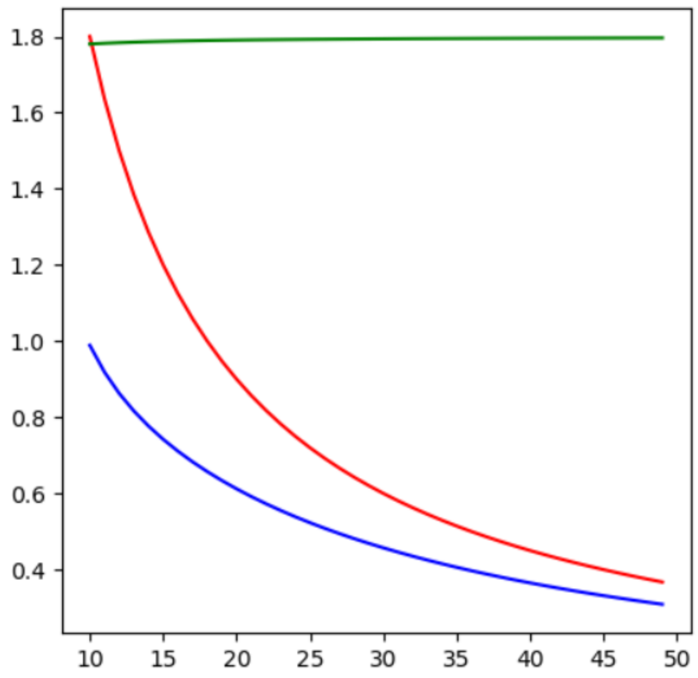




PART III

Maximum error graph in given range of grid sizes (n,N)

x0 = 1, y(x0) = 2, X = 10, n=10 , N = 50



x0 = 1, y(x0) = 3, X = 50, n=10, N = 100, Runge Kutte