**ROLSTON’S METHOD**

#RK Method (2ND order or ROLSTON'S method)

import matplotlib.pyplot as plt

[xo,xn,yo,h]=[0,4,1,0.5]

n=(xn-xo)/h

#f(x.y)=dy/dx (i.e derivative/slope of original; function)

print("Ralston's METHOD")

def f1(x,y):

return -2\*(x)\*\*3+12\*(x)\*\*2-20\*(x)+8.5

#f(x,y)=y(i.e integral of f1 which is original function)

def f2(x,y):

return (-0.5\*x\*\*4+4\*x\*\*3-10\*x\*\*2+8.5\*x+1)

print("xi \t yi \t f(xi,yi) \t xi+h \t yi+1 \t y(true)")

print("==================================================================")

a=[]

b=[]

c=[]

for i in range(0,int(n)+1):

yi\_ral=yo+(((1/3)\*(f1(xo,yo)))+((2/3)\*(f1(xo+(3/4)\*h,yo+(3/4)\*(f1(xo,yo))\*h))))\*h

print(round(xo,4), ' \t',round(yo,4), '\t',round(f1(xo,yo),4), '\t',round(xo+h,4), '\t',round(yi\_ral,4), '\t',round(f2(xo,yo),4))

a.append(round(xo,2))

b.append(round(yo,2))

c.append(f2(xo,yo))

xo=xo+h

yo=yi\_ral

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

print("x=",a,"\ny(euler)=",b,"\ny(true)=",c)

plt.plot(a,b,label = "y(euler)=dy/dx")

plt.plot(a,c,label = "y(true)=y")

plt.title('EULER\'S METHOD')

plt.xlabel('x')

plt.ylabel('f(x,y)')

plt.legend()

**OUTPUT**

Ralston's METHOD

xi yi f(xi,yi) xi+h yi+1 y(true)

==================================================================

0 1 8.5 0.5 3.2773 1.0

0.5 3.2773 1.25 1.0 3.1016 3.2188

1.0 3.1016 -1.5 1.5 2.3477 3.0

1.5 2.3477 -1.25 2.0 2.1406 2.2188

2.0 2.1406 0.5 2.5 2.8555 2.0

2.5 2.8555 2.25 3.0 4.1172 2.7188

3.0 4.1172 2.5 3.5 4.8008 4.0

3.5 4.8008 -0.25 4.0 3.0312 4.7188

4.0 3.0312 -7.5 4.5 -3.8164 3.0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

x= [0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0]

y(euler)= [1, 3.28, 3.1, 2.35, 2.14, 2.86, 4.12, 4.8, 3.03]

y(true)= [1.0, 3.21875, 3.0, 2.21875, 2.0, 2.71875, 4.0, 4.71875, 3.0]

