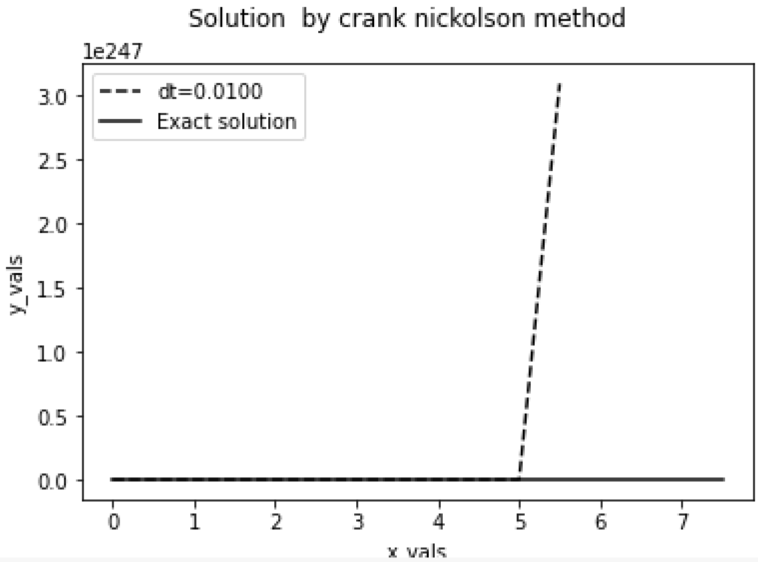
```
#PROBLEM 2

#in this project we gonna be coding the 2 ode functions as defined in the given paper

# crank nickolson

#importing important libraries
import numpy as np
import math
import matplotlib.pyplot as plt

# defining the ode
def f(x, y) :
    return x*y**2+x

# defining time step
h = 0.5

# defing a step length
dt=0.01

# defining the initial condition
y0 = 2

# the range of the initial condition
X = 8

# list of discretized time
x = np.arange(0, X, 0.5)

# lets define the euler's with crank nickolson method
y_approx = np.zeros(len(x))
y_approx[0] = y0;
for i in range(1, len(x)) :
    y_approx[i] = y_approx[i - 1] + f(x[i - 1], y_approx[i - 1]) * h


#calculating the y exact result
y_exact = np.tan (x**2/2)


# Calculating the  Error value and plotting
dif_val=y_exact-y_approx

# Plotting of solution with exact result
plt.plot(x,y_approx,'k--',label="dt=%.4f"%(dt))
plt.plot(x, y_exact,'k',label="Exact solution")
plt.xlabel("x_vals")
plt.ylabel("y_vals")
plt.legend(loc='best')
plt.suptitle("Solution  by crank nickolson method")
```

```
print(f'Table of errors between the exact and the approximated values {np.abs(dif_val)}')
```

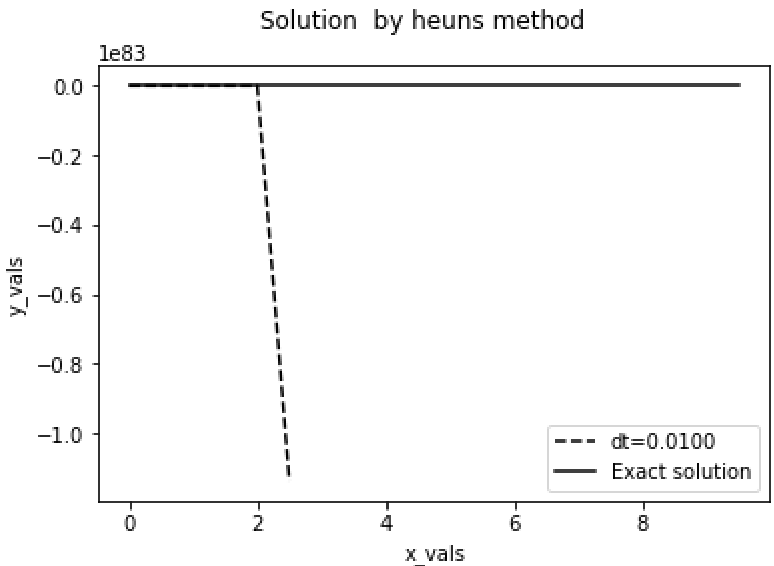
```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:15: RuntimeWarning: ove
  from ipykernel import kernelapp as app
Table of errors between the exact and the approximated values [2.00000000e+000 1.874
 7.31388973e+001 5.10642033e+003 3.25993020e+007 1.59407222e+015
 4.44686592e+030 3.95492330e+061 3.51931912e+123 3.09640177e+247
             inf             inf             inf             inf]
```




```
"""PROBLEM 1"""

# heun's method

# defining the problem
def f(x, y) :
    return y**2 *np.cos (x) +np.cos (x)


#defining the time step
h = 0.5

# stating the initial condition
y0 = 2

# step length
dt=0.01

# X value range for the initial condition
X = 10

# list of discretized time
x = np.arange(0, X, 0.5)

# heun's method
y_approx1 = np.zeros(len(x))
y_approx1[0] = y0;
for i in range(1, len(x)) :
    k1 = h * f(x[i - 1], y_approx1[i - 1])
    k2 = h * f(x[i], y_approx1[i - 1] + k1)
```

```
    y_approx1[i] = y_approx1[i - 1] + (k1 + k2) / 2



#calculation of exact result
y_exact =np.tan(np.sin(x)+np.arctan(2))


# Calculation of Error and plotting
dif_val=y_exact-y_approx1

# we now Plot the solution with exact result
plt.plot(x,y_approx1,'k--',label="dt=%.4f"%(dt))
plt.plot(x, y_exact,'k',label="Exact solution")
plt.xlabel("x_vals")
plt.ylabel("y_vals")
plt.legend(loc='best')

plt.suptitle("Solution  by heuns method")
print(f'Table of errors between the exact and the approximated values {np.abs(dif_val)}')
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8: RuntimeWarning: over

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:32: RuntimeWarning: inv
Table of errors between the exact and the approximated values [4.44089210e-16 7.1286
 1.90330388e+21 1.13793150e+83            inf            inf
           inf            inf            nan            nan
           nan            nan            nan            nan
           nan            nan            nan            nan]
```




"""END OF PROJECT AND IMPLEMENTATION .THANK YOU!!!"""

✓ 0s completed at 12:07 AM