Lab 7

LAB 9: Solution of ODE of first order and first degree by Taylor's series and Modified Euler's method

9.1 Objectives:

Use python

1. to solve ODE by Taylor series method.
2. to solve ODE by Modified Euler method.
3. to trace the solution curves.

9.2 Taylor series method to solve ODE

Solve: $\frac{dy}{dx} - 2y = 3e^x$ with $y(0) = 0$ using Taylor series method at $x = 0.1(0.1)0.3$.

```
## module taylor
'''X,Y = taylor(deriv,x,y,xStop,h).
4th-order Taylor series method for solving the initial value problem {y
                                            }' = {F(x,{y})}, where

{y} = {y[0],y[1],...y[n-1]}.
x,y = initial conditions
xStop = terminal value of x
h = increment of x
'''
from numpy import * ~~array~~
def taylor(deriv,x,y,xStop,h):
    X = []
    Y = []
    X.append(x)
    Y.append(y)
    while x < xStop:                # Loop over integration steps
        D = deriv(x,y)              # Derivatives of y
        H = 1.0
        for j in range(3):          # Build Taylor series
            H = H*h/(j + 1)
            y = y + D[j]*H          # H = h^j/j!
        x = x + h
        X.append(x) # Append results to
        Y.append(y) # lists X and Y

    return array(X),array(Y) # Convert lists into arrays

# deriv = user-supplied function that returns derivatives in the 4 x n
                                            array
'''
[y'[0] y'[1] y'[2] ... y'[n-1]
y''[0] y''[1] y''[2] ... y''[n-1]
y'''[0] y'''[1] y'''[2] ... y'''[n-1]
y''''[0] y''''[1] y''''[2] ... y''''[n-1]]
'''
def deriv(x,y):
    D = zeros((4,1))
```

```
    D[0] = [2*y[0] + 3*exp(x)]
    D[1] = [4*y[0]+ 9*exp(x)]
    D[2] = [8*y[0]+ 21*exp(x)]
    D[3] = [16*y[0]+ 45*exp(x)]
    return D

x = 0.0          # Initial value of x
xStop = 0.3      # last value
y = array([0.0])            # Initial values of y
h = 0.1             # Step size
X,Y = taylor(deriv,x,y,xStop,h)
print("The required values are :at x= %0.2f, y=%0.5f, x=%0.2f, y=%0.5f,
                    x = %0.2f, y=%0.5f,x = %0.2f, y=%0
                    .5f"%(X[0],Y[0],X[1],Y[1],X[2],Y[2]
                    ,X[3],Y[3] ))
```

```
he required values are :at x= 0.00, y=0.00000, x=0.10, y=0.34850,
 = 0.20, y=0.81079,x = 0.30, y=1.41590
```

```
x = 0.20, y=0.81079,x = 0.30, y=1.41590
```

Solve $y' + 4y = x^2$ with initial conditions $y(0) = 1$ using Taylor series method at $x = 0.1, 0.2$.

```
from numpy import* array
def taylor(deriv,x,y,xStop,h):
    X = []
    Y = []
    X.append(x)
    Y.append(y)
    while x < xStop:                    # Loop over integration steps
        D = deriv(x,y)                  # Derivatives of y
        H = 1.0
        for j in range(3):              # Build Taylor series
            H = H*h/(j + 1)
            y = y + D[j]*H              # H = h^j/j!
        x = x + h
        X.append(x) # Append results to
        Y.append(y) # lists X and Y

    return array(X),array(Y) # Convert lists into arrays

# deriv = user-supplied function that returns derivatives in the 4 x n
                                            array
'''
[y'[0] y'[1] y'[2] ... y'[n-1]
y"[0] y"[1] y"[2] ... y"[n-1]
y'''[0] y'''[1] y'''[2] ... y'''[n-1]
y""[0] y""[1] y""[2] ... y""[n-1]]
'''
def deriv(x,y):
    D = zeros((4,1))
    D[0] = [x**2-4*y[0]]
    D[1] = [2*x-4*x**2+16*y[0]]
    D[2] = [2-8*x+16*x**2-64*y[0]]
    D[3] = [-8+32*x-64*x**2+256*y[0]]
```

```
    return D

x = 0.0            # Initial value of x
xStop = 0.2        # last value
y = array([1.0])            # Initial values of y
h = 0.1                # Step size
X,Y = taylor(deriv,x,y,xStop,h)

print("The required values are :at x= %0.2f, y=%0.5f, x=%0.2f, y=%0.5f,
                                    x = %0.2f, y=%0.5f"%(X[0],Y[0],X[1
                                    ],Y[1],X[2],Y[2]))
```

```
The required values are :at x= 0.00, y=1.00000, x=0.10, y=0.66967,
x = 0.20, y=0.45026
```

9.4 Modified Euler's method

The iterative formula is:

$$y_1^{(n+1)} = y_0 + \frac{h}{2}[f(x_0, y_0) + f(x_1, y_1^{(n)})], \qquad n = 0, 1, 2, 3, \ldots,$$

where, $y_1^{(n)}$ is the n^{th} approximation to y_1.

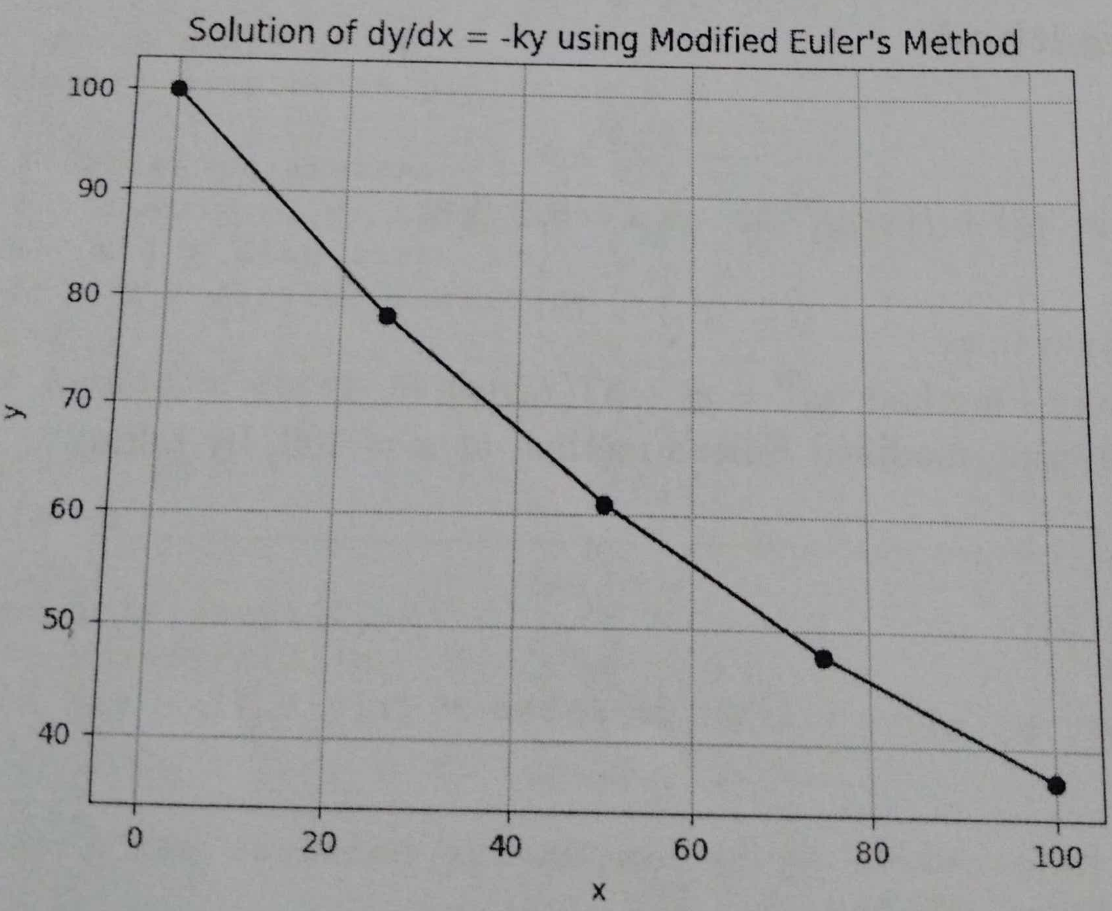
The first iteration will use Euler's method: $y_1^{(0)} = y_0 + hf(x_0, y_0)$.

Solve $y' = -ky$ with $y(0) = 100$ using modified Euler's method at $x = 100$, by taking $h = 25$.

```
import numpy as np
import matplotlib.pyplot as plt

def modified_euler(f, x0, y0, h, n):
    x = np.zeros(n+1)
    y = np.zeros(n+1)

    x[0] = x0
    y[0] = y0

    for i in range(n):
        x[i+1] = x[i] + h
        k1 = h * f(x[i], y[i])
        k2 = h * f(x[i+1], y[i] + k1)
        y[i+1] = y[i] + 0.5 * (k1 + k2)

    return x, y
```

```
def f(x, y):
    return -0.01 * y             # ODE dy/dx = -ky

x0 = 0.0
y0 = 100.0
h = 25
n = 4

x, y = modified_euler(f, x0, y0, h, n)

print("The required value at x= %0.2f, y=%0.5f"%(x[4],y[4]))
print("\n\n")

# Plotting the results
plt.plot(x, y, 'bo-')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Solution of dy/dx = -ky using Modified Euler\'s Method')
plt.grid(True)
plt.show()
```

```
The required value at x= 100.00, y=37.25290
```



9.5 Exercise:

1. Find $y(0.1)$ by Taylor Series exapnsion when $y' = x - y^2, y(0) = 1$.

   Ans: $y(0.1) = 0.9138$

2. Find $y(0.2)$ by Taylor Series exapnsion when $y' = x^2y - 1, y(0) = 1, h = 0.1$.

   Ans: $y(0.2) = 0.80227$

3. Evaluate by modified Euler's method: $y' = ln(x+y), y(0) = 2$ at $x = 0(0.2)0.8$.

   Ans: $2.0656, 2.1416, 2.2272, 2.3217$

4. Solve by modified Euler's method: $y' = x + y, y(0) = 1, h = 0.1, x = 0(0.1)0.3$.

   Ans: $1.1105, 1.2432, 1.4004$