LAB ASSINGMENT

Submitted by : Group 4 (Ithal, Pranav, Edna, Krishnendu, Salva, karthik, parvi)

Department : 2^{nd} year Bsc physics (Aided)

Aim : To simulate projectile motion using Euler's method and plot Y vs x , Y vs t and x Vs t graphs. compare, Horizontal range, Time of flight and Maximum height reached with theoretical results, include the effect of air resistance and freefall. and plot Y vs x and Y vs t graphs.

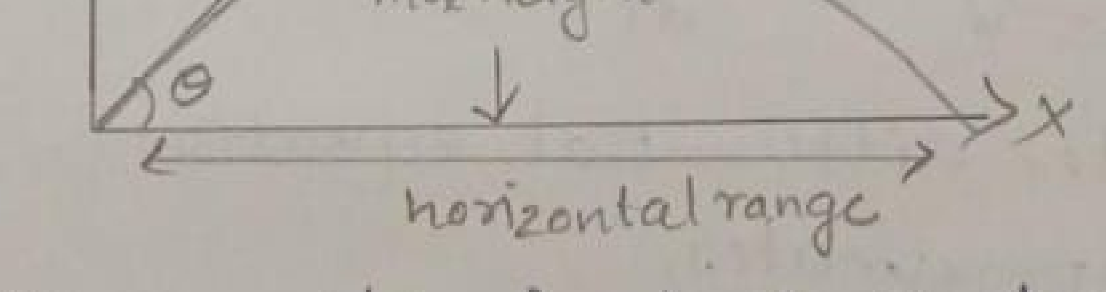
principle : consider a body projected at some angle 'θ' with an initial velocity 'u'. Neglecting the air resistance, components of acceleration exerted on the body along the two directions are

$g_x = 0, \quad g_y = 9.8 m/s^2$

$a = -g$

g_x = gravity along x-axis

g_y = gravity along y-axis



components of velocity vector at time 't'

$V_x(t) = V\cos\theta$

$V_y(t) = V\sin\theta - g_y t$

position vectors are given by

$X = V\cos\theta$

$Y = V\sin\theta - \frac{1}{2}gt^2$

Total horizontal distance 'R' covered by projectile is given by

Horizontal Range, $R = \frac{v^2 \sin 2\theta}{g}$

Maximum height reached by the projectile

$$H = \frac{v^2 \sin^2\theta}{2g}$$

Time of flight of the projectile

$$T = \frac{2v\sin\theta}{g}$$

If Air resistance is included,

$$a_x = -k v_x$$

$$a_y = -g - \frac{k v_y}{m}$$

Theory: Euler method is a numerical technique to solve ordinary differential equations by approximating the solution at small time steps.

General equation of Euler's method,

$$f_{n+1} = f_n + \frac{df}{dt} \cdot \Delta t$$

We know that, $\frac{d^2x}{dt^2} = a$ on integrating $\frac{dx}{dt} = v$

$$\frac{dx}{dt} = v$$

This is a first-order differential equation,

Euler's method approximates,

$$\frac{dx}{dt} \approx \frac{x_{n+1} - x_n}{\Delta t}$$

rearranging,

$$x_{n+1} \approx x_n + \frac{dx}{dt} \cdot \Delta t$$

$$x_{n+1} = x_n + v \cdot \Delta t$$

Thus we get the position update equation,

$$x_{n+1} = x_n + v_x \cdot \Delta t$$

$$y_{n+1} = y_n + v_y \cdot \Delta t$$

similarly,

$$a = \frac{dv}{dt}$$

This is also a first order differential equation

Euler Method approximates derivative,

$$\frac{dv}{dt} \approx \frac{v_{n+1} - v_n}{\Delta t}$$

Rearranging ~~equa~~ equation, we get

$$v_{n+1} = v_n + \frac{dv}{dt} \cdot \Delta t$$

Substitute $\frac{dv}{dt} = a$,

$$v_{n+1} = v_n + a \cdot \Delta t$$

Then we get the velocity update equation.

$$v_{n+1} = v_n + a \cdot \Delta t$$

Algorithm

Step 1 . Initialize parameters

Step 2 : Set initial conditions

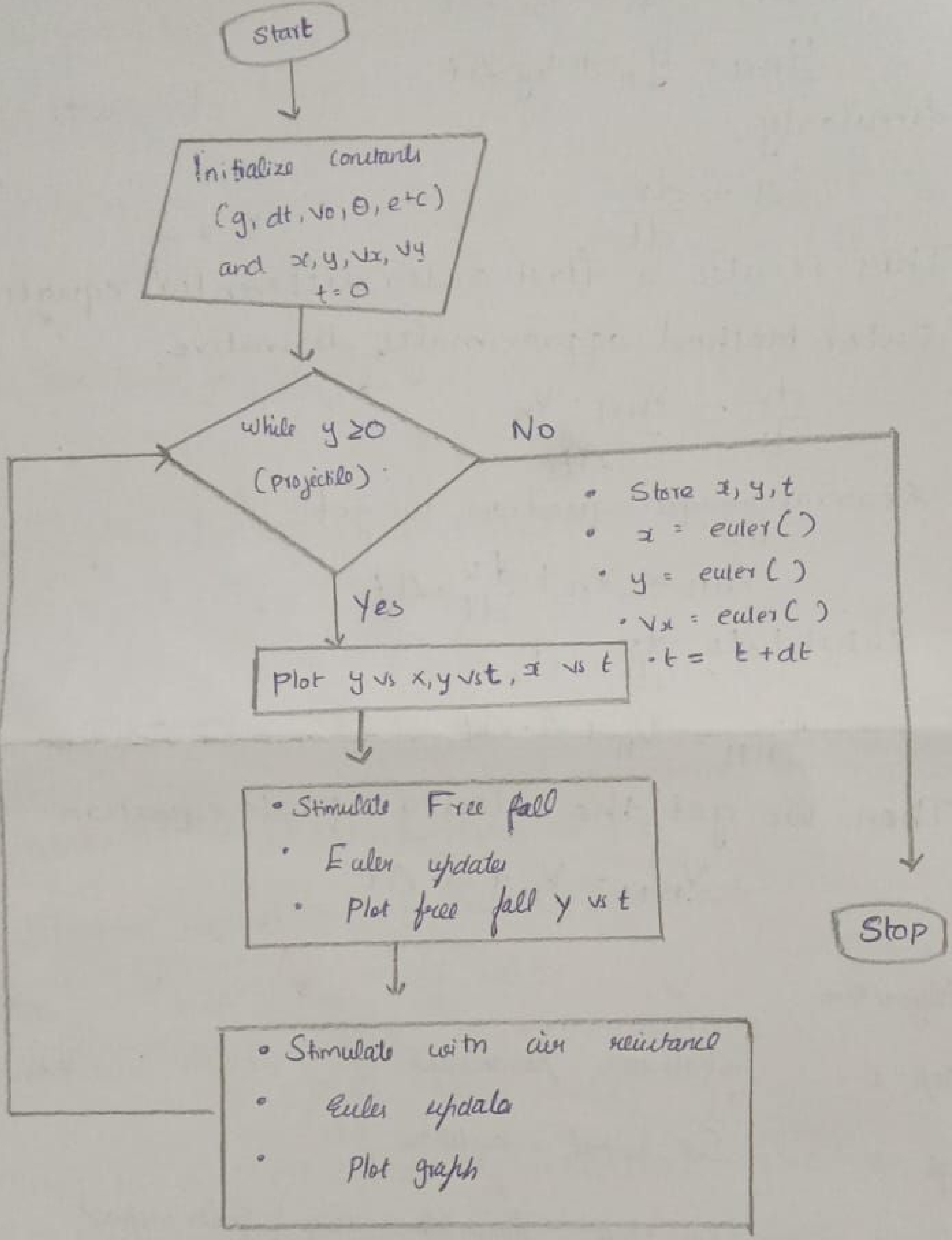
Step 3 : Stimulate projectile using Euler method

Step 4 : Plot Trajectories

Step 5 : Stimulate free fall

Step 6 : Stimulate with air resistance

Flow chart


Start
Initialize constants
(g, dt, v0, θ, etc)
and x, y, vx, vy
t=0
while y ≥0
(projectile)
No
Yes
• Store x, y, t
• x = euler()
• y = euler()
• vx = euler()
• t = t+dt
Plot y vs x, y vs t, x vs t
• Stimulate Free fall
• Euler updates
• Plot free fall y vs t
Stop
• Stimulate with air resistance
• Euler update
• Plot graph

python code:

```
import numpy as np
import matplotlib.pyplot as plt
def euler(x,y,fxy,h):
    return y+h*fxy(x,y)

g=9.81
dt=0.01
VO = 50
theta=45
theta_rad=np.radians(theta)
VxO = VO*np.cos(theta-rad)
VyO = VO*np.sin(theta_rad)

x, y = 0,0
vx, vy = VxO, VyO
t=0
X, Y, T= [],[],[]
while y>=0:
    X.append(x)
    Y.append(y)
    T.append(t)
    x=euler(t,x,lambda t,x: vx,dt)
    y= euler(t,y,lambda t,y: vy,dt)
    vx=euler(t,vx,lambda t,vx:0,dt)
    vy=euler(t,vy,lambda t,vy:-g,dt)
    t+=dt
plt.figure(figsize=(14,4))
    # YvsX
plt.subplot(1,3,1)
plt.plot(X,Y)
plt.title("Trajectory (Y vs x)")
plt.xlabel("x(m)")
```

```
plt.ylabel ("y (m)")
plt.grid()
  # yvst
plt.subplot (1, 3, 2)
plt.plot (T, Y)
plt.title ("Height vs Time")
plt.xlabel ("Time (s)")
plt.ylabel ("Height (m)")
plt.grid()
  #X vs t
plt.subplot (1, 3, 3)
plt.plot (T, X)
plt.title ("Distance vs Time")
plt.xlabel ("Time (s)")
plt.ylabel ("Distance (m)")
plt.grid()
plt.tight_layout ()
plt.show()
print ("simulation values")
print (f"Range : {x[-1]:.2f}m")
print (f" Maxheight : {max (Y):.2f}m")
print (f" Time of flight : {T[-1]:.2f}s")
  # FreeFall
Y =100
vy =0
t =0
yf, Tf = [], []
while y>=0:
   yf.append (y)
   Tf.append (t)
   Y=euler (t, Y, lambda t, Y: vy, dt)
```

```
    vy = euler(t, vy, lambda t, vy: -g, dt)
    t += dt
plt.plot (Tf, Yf)
plt.title ("Free fall (Y vs t)")
plt.xlabel ("Time (s)")
plt.ylabel ("Height (m)")
plt.grid()
plt.show()
    # with Air resistance

k = 0.1
m = 1.0
x, y = 0, 0
vx, vy = 0, 0
t = 0
xa, ya = [], []
while y >= 0:
    xa.append(x)
    ya.append(y)
    ax = lambda t, vx: -k*vx/m
    ay = lambda t, vy: -g - k*vy/m
    vx = euler (t, vx, ax, dt)
    vy = euler (t, vy, ay, dt)
    x = euler(t, x, lambda t, x: vx, dt)
    y = euler (t, y, lambda t, y: vy, dt)
    t += dt
plt.plot (x, y, label = 'No Air Resistance')
plt.plot (xa, ya, label = 'With Air Resistance')
plt.title ("projectile with and without Air resistance")
plt.xlabel ("x (m)")
plt.ylabel ("y (m)")
plt.legend ()
```

```
plt.grid()
plt.show()
```