Numerical Solutions of Equations of Motion

Physics Staff-Graded Assignment (2023eBCS072)

Import libraries

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
```

Initialize global values in simulations

dt = Time interval between iterations
T = Total time of the simulation
N = Number of iterations
g = (x,y) vector for acceleration to due gravity

```
In [2]: global dt, T, N, g
        dt = 0.01
        T = 10
        N = int(T/dt)
        g = np.array([0, -9.8])
```

Define logic for a simulation with euler integration

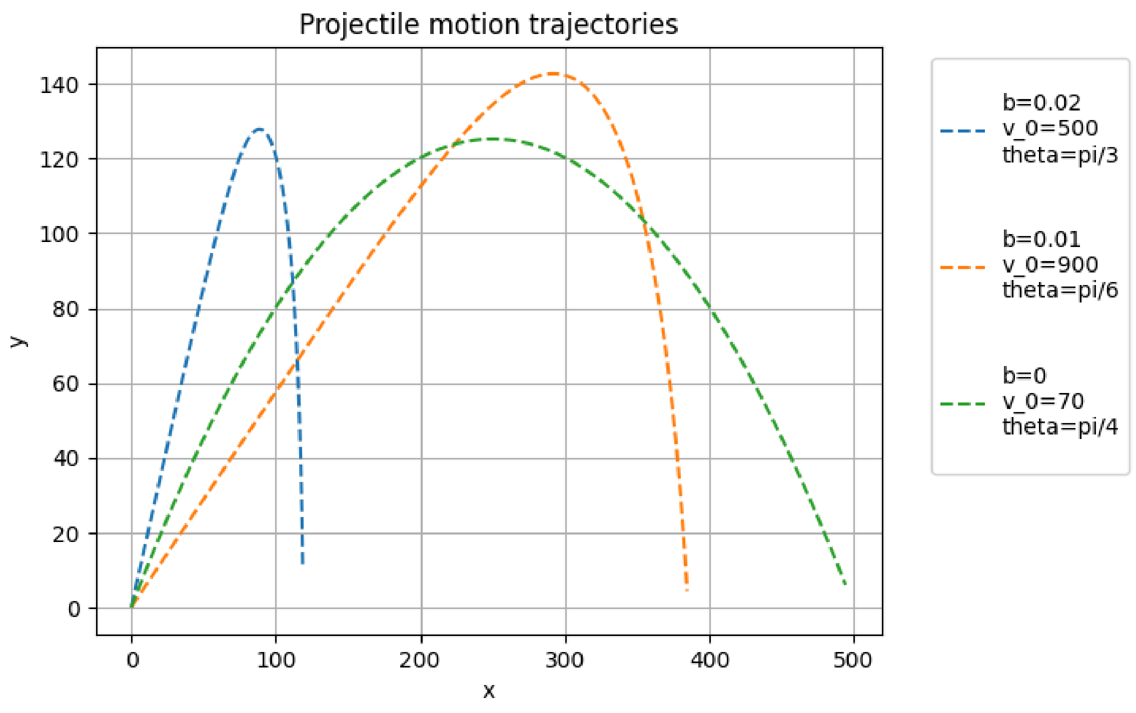
- Take initial position (q_0), intial speed (v_0), angle (theta), and coefficient of air resistance (b) as parameters for the simulation.
- Initialize an empty array with 2 rows (x and y coordinates) and N columns (for N iterations).
- Initialize the first column of the array with x and y co-ordinates of initial position.
- Initialize velocity with x component = $v_0 \cdot \cos(\theta)$ and y component = $v_0 \cdot \sin(\theta)$.
- Iterating i from 1 to N-1:
  - update position by: $q_i = q_{i-1} + v_{i-1} \cdot \Delta t$
  - find acceleration by: $a = -b \cdot |v_{i-1}| \cdot v_{i-1} + g$
  - update velocity by: $v_i = v_{i-1} + a \cdot \Delta t$

```
In [3]: def projectile_motion(q_0, v_0, theta, b):
            q_t = np.zeros((2, N))
            q_t[:, 0] = q_0
            v = np.array([v_0*np.cos(theta), v_0*np.sin(theta)])
            for i in range(1,N):
                q_t[:, i] = q_t[:, i-1] + v*dt
                a = g - b*np.linalg.norm(v)*v
                v = v + a*dt
            return q_t
```

Tuning Parameters for simulations and corresponding observations

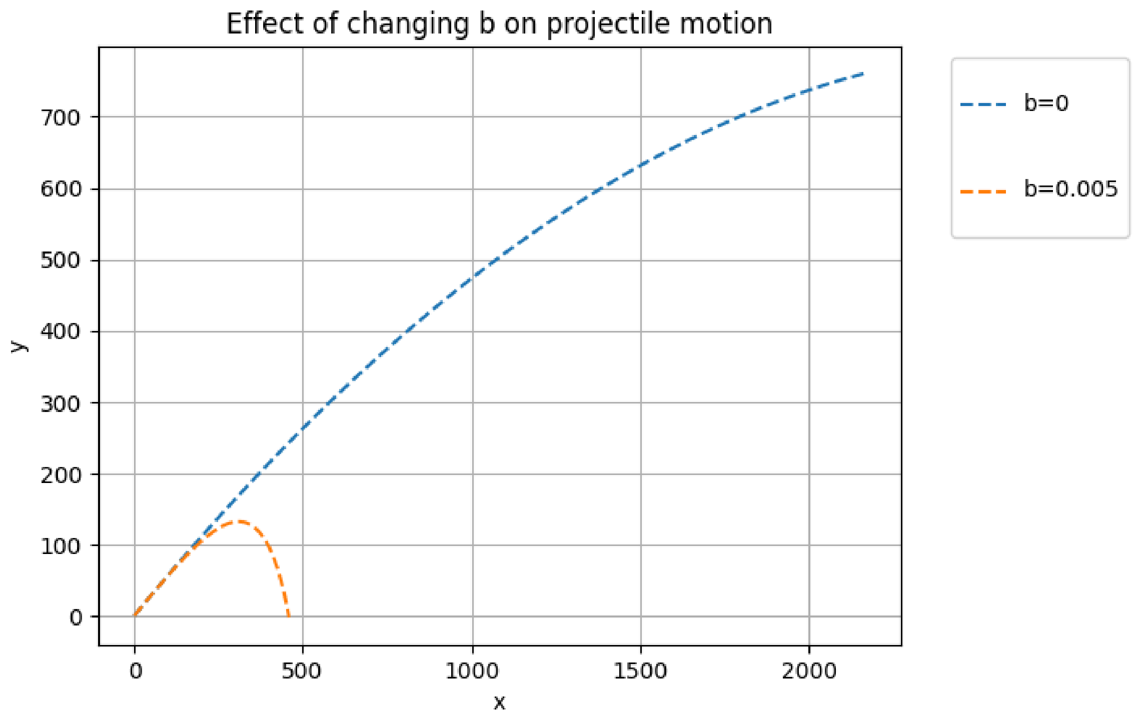
First, we take 3 arbitrary values for each parameter and perform the simulation, and plot the corresponding trajectories

In [4]:
```
q_0 = np.array([0, 0])

v_01 = 500
theta1 = np.pi/3
b1 = 0.02
q1 = projectile_motion(q_0, v_01, theta1, b1)

v_02 = 900
theta2 = np.pi/6
b2 = 0.01
q2 = projectile_motion(q_0, v_02, theta2, b2)

v_03 = 70
theta3 = np.pi/4
b3 = 0
q3 = projectile_motion(q_0, v_03, theta3, b3)

plt.plot(q1[0, :], q1[1, :], linestyle='--', antialiased=True, label=
f'\nb={b1}\nv_0={v_01}\ntheta=pi/{int(1/theta1*np.pi)}\n')
plt.plot(q2[0, :], q2[1, :], linestyle='--', antialiased=True, label=
f'\nb={b2}\nv_0={v_02}\ntheta=pi/{int(1/theta2*np.pi)}\n')
plt.plot(q3[0, :], q3[1, :], linestyle='--',antialiased=True, label=
f'\nb={b3}\nv_0={v_03}\ntheta=pi/{int(1/theta3*np.pi)}\n')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Projectile motion trajectories')
plt.grid()
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```



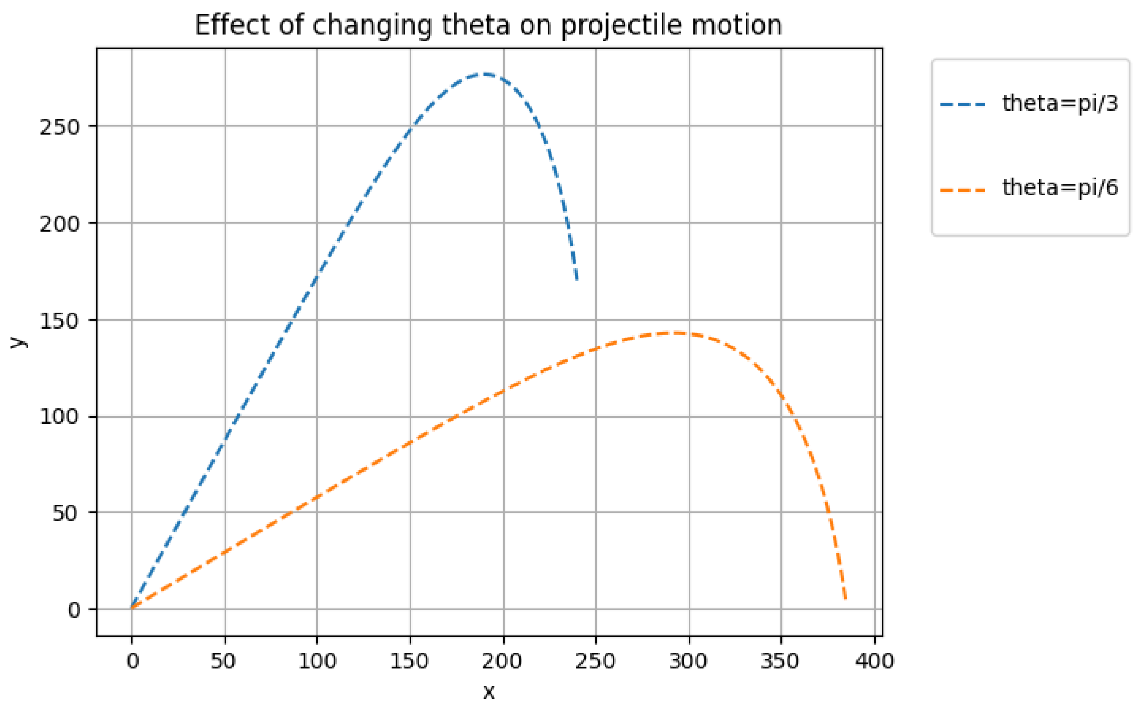
Varying b and keeping other parameters constant

In [5]:
```
v_0 = 250
theta = np.pi/6
b1 = 0
b2 = 0.005
q1 = projectile_motion(q_0, v_0, theta, b1)
q2 = projectile_motion(q_0, v_0, theta, b2)

plt.plot(q1[0, :], q1[1, :], linestyle='--', antialiased=True, label=
f'\nb={b1}\n')
plt.plot(q2[0, :], q2[1, :], linestyle='--', antialiased=True, label=
f'\nb={b2}\n')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Effect of changing b on projectile motion')
plt.grid()
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```



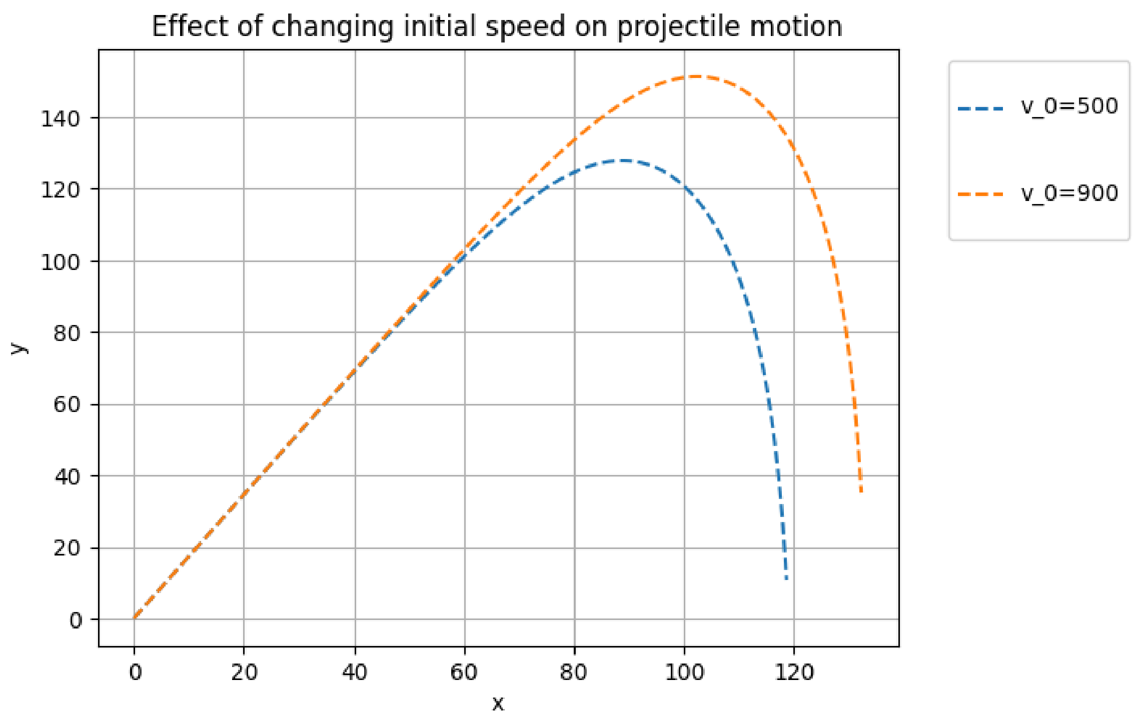
Varying θ and keeping other parameters constant

```
In [6]: b = 0.01
        v_0 = 900
        theta1 = np.pi/3
        theta2 = np.pi/6

        q1 = projectile_motion(q_0, v_0, theta1, b)
        q2 = projectile_motion(q_0, v_0, theta2, b)

        plt.plot(q1[0, :], q1[1, :], linestyle='--', antialiased=True, label=
        f'\ntheta=pi/{int(1/theta1*np.pi)}\n')
        plt.plot(q2[0, :], q2[1, :], linestyle='--', antialiased=True, label=
        f'\ntheta=pi/{int(1/theta2*np.pi)}\n')
        plt.xlabel('x')
        plt.ylabel('y')
        plt.title('Effect of changing theta on projectile motion')
        plt.grid()
        plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
        plt.show()
```



Varying v_0 and keeping other parameters constant

In [7]:
```
v_01 = 500
v_02 = 900
theta1 = np.pi/3
b = 0.02
q1 = projectile_motion(q_0, v_01, theta1, b)
q2 = projectile_motion(q_0, v_02, theta1, b)

plt.plot(q1[0, :], q1[1, :], linestyle='--', antialiased=True, label=
f'\nv_0={v_01}\n')
plt.plot(q2[0, :], q2[1, :], linestyle='--', antialiased=True, label=
f'\nv_0={v_02}\n')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Effect of changing initial speed on projectile motion')
plt.grid()
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```


Effect of changing initial speed on projectile motion
y
x
v_0=500
v_0=900