Problem 2

The differential equation for the velocity of a bungee jumper is different depending on whether the jumper has fallen to a distance where the cord is fully extended and begins to stretch. Thus, if the distance fallen is less than the cord length, the jumper is only subject to gravitational and drag forces. Once the cord begins to stretch, the spring and dampening forces of the cord must also be included. These two conditions can be expressed by the following equations:

$$\frac{dv}{dt} = \begin{cases} g - \operatorname{sign}(v)\frac{c_d}{m}v^2, & x \leq L \\ g - \operatorname{sign}(v)\frac{c_d}{m}v^2 - \frac{k}{m}(x-L) - \frac{\gamma}{m}v, & x > L \end{cases}$$

where v is velocity (m/s), t is time (s), g is gravitational constant (9.81 m/s2), sign(x) is a function that returns −1, 0, 1 for negative, zero, and positive x, respectively, cd is second-order drag coefficient (kg/m), m is mass (kg), k is cord spring constant (N/m), γ is cord dampening coefficient (Ns/m), and L cord length (m). Determine the position and velocity of the jumper given the following parameters: L = 30 m, m = 68.1 kg, cd = 0.25 kg/m, k =40 N/m, and γ = 8 kg/s. Perform the computation from t = 0 to 50 s and assume that the initial conditions are: x(0) = v(0) = 0. You are required to write a computer program implementing the Fourth order Runge-Kutta method.

Description of Problem:

We are tasked with simulating the velocity and position of a bungee jumper during their descent. The differential equation governing the bungee jumper's motion is dependent on whether the jumper has fallen to a distance where the cord is fully extended and begins to stretch. This simulation will challenge your ability to implement numerical methods and understand the physics of bungee jumping.

Given:

The given parameters are as follows:

- Gravitational constant, $g = 9.81\ m/s^2$
- Second-order drag coefficient, $c_d = kg/m$
- Mass of the jumper, $m = 68.1\ kg$
- Cord spring constant, $k = 40\ N/m$
- Cord dampening coefficient, $\gamma = 8\ kg/s$

- Initial conditions: $x(0) = 0$ (initial position), $v(0) = 0$ (initial velocity)

Method Used: Fourth Order Runge-Kutta Methods

The Fourth-order Runge-Kutta method, often abbreviated as RK4, is a widely used numerical technique for solving ordinary differential equations (ODEs). It is a member of the Runge-Kutta family of numerical methods and is particularly popular due to its accuracy and simplicity. The RK4 method is used to approximate the solution to an initial value problem for a first-order ODE.

Steps for implementing RK4:

$$y_{i+1} = y_i + slope \times h$$

Here, h is step size.

$$slope = \phi(x_i, y_i, h)$$

For fourth order runge-kutta method ϕ is

$$\phi = (k_1 + 2k_2 + 2k_3 + k_4)/6$$

Here k_1, k_2, k_3 *and* k_4 are as follows

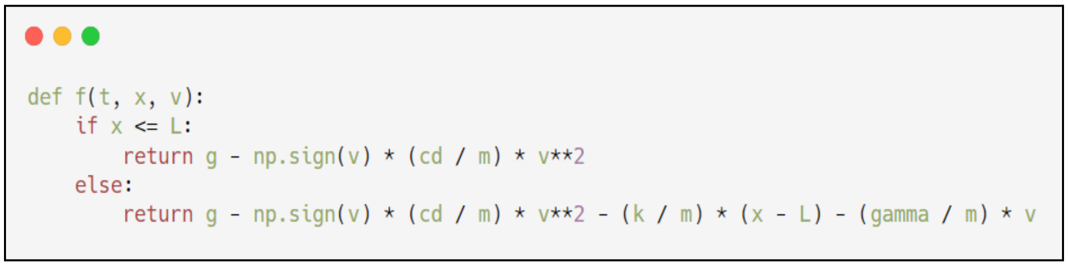
- $k_1 = f(x_i, y_i)$
- $k_2 = f(x_i + \frac{1}{2}h,\ y_i + \frac{1}{2}k_1 h)$
- $k_3 = f(x_i + \frac{1}{2}h,\ y_i + \frac{1}{2}k_2 h)$
- $k_4 = f(x_i + h,\ y_i + k_3 h)$

Hence,

$$\boxed{y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)h}$$

Procedure:

➢ Define the function $f(t, x, v)$

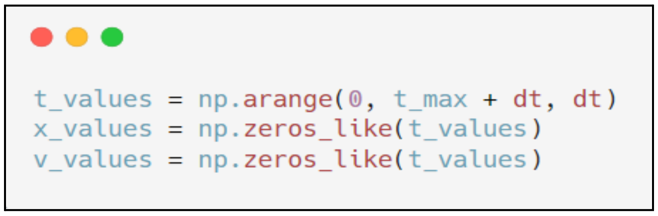


```
def f(t, x, v):
    if x <= L:
        return g - np.sign(v) * (cd / m) * v**2
    else:
        return g - np.sign(v) * (cd / m) * v**2 - (k / m) * (x - L) - (gamma / m) * v
```

➢ Step Size:

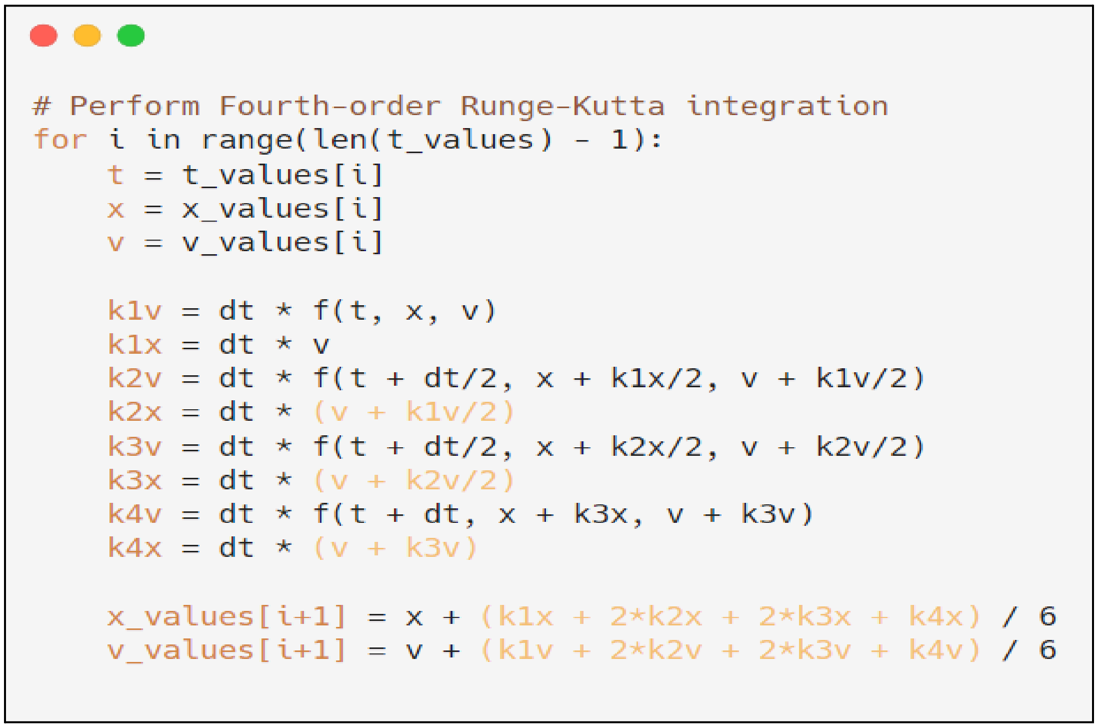
Take step size(h) to be 0.1

➢ Initialize arrays to store results



```
t_values = np.arange(0, t_max + dt, dt)
x_values = np.zeros_like(t_values)
v_values = np.zeros_like(t_values)
```

➢ Performing fourth-order Runge-Kutta method

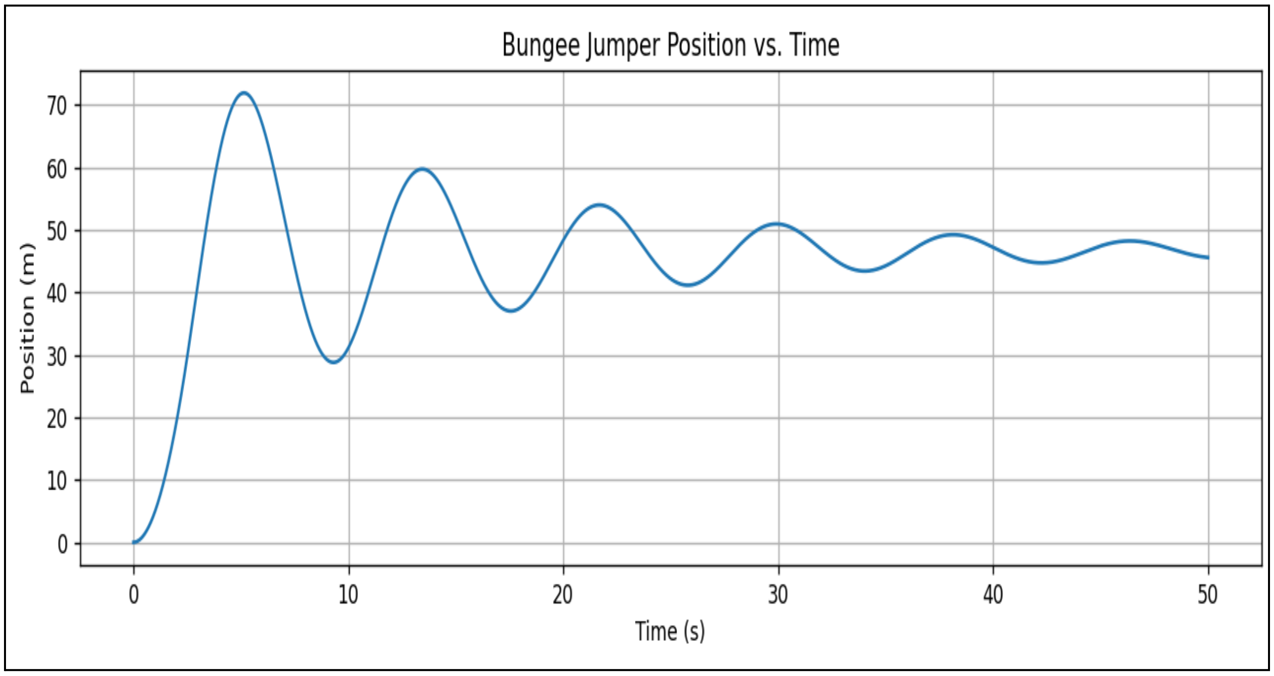


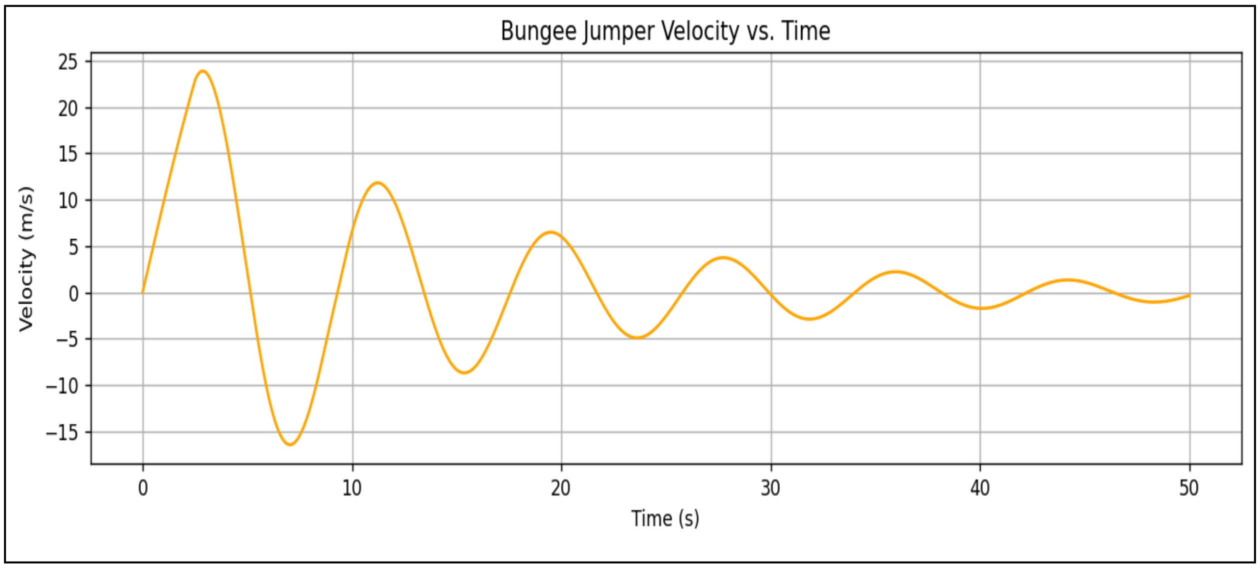
```
# Perform Fourth-order Runge-Kutta integration
for i in range(len(t_values) - 1):
    t = t_values[i]
    x = x_values[i]
    v = v_values[i]

    k1v = dt * f(t, x, v)
    k1x = dt * v
    k2v = dt * f(t + dt/2, x + k1x/2, v + k1v/2)
    k2x = dt * (v + k1v/2)
    k3v = dt * f(t + dt/2, x + k2x/2, v + k2v/2)
    k3x = dt * (v + k2v/2)
    k4v = dt * f(t + dt, x + k3x, v + k3v)
    k4x = dt * (v + k3v)

    x_values[i+1] = x + (k1x + 2*k2x + 2*k3x + k4x) / 6
    v_values[i+1] = v + (k1v + 2*k2v + 2*k3v + k4v) / 6
```

➢ Plot of the final result





Accuracy:

RK4 is a fourth-order method, which means that its local truncation error (the error introduced at each step) is on the order of h^5. This makes it significantly more accurate than simpler methods like Euler's method, which is only first-order.

Final Conclusion:

In solving the problem of simulating the velocity and position of a bungee jumper during their descent using the Fourth-order Runge-Kutta method, we have successfully obtained a numerical solution that describes the bungee jumper's motion over time. The simulation considered various physical factors, including gravitational forces, air resistance, spring forces from the bungee cord, and dampening effects, providing a comprehensive representation of the jumper's experience.