free_fall_object-27June2025

June 29, 2025

[1]:
```
## Autor: S Rajarajan - Date 27 June 2025
import turtle
import time
import matplotlib.pyplot as plt


# Setup screen
screen = turtle.Screen()
screen.setup(width=700, height=850, startx=790, starty=5)

# Constants
g = 9.81  # Gravity (m/s²)
dt = 0.1  # Time step (s)
terminal_vel = 50  # Approximate terminal velocity (m/s)
ground_level = -270  # Lower ground level for longer fall

C = 0.6  # Drag coefficient (depends on shape)
rho = 1.2  # Air density (kg/m³)
A = 0.2  # Cross-sectional area of object (m²)
m = 6  # Mass of the object (kg)

# Create distance scale (vertical ruler)
scale_turtle = turtle.Turtle()
scale_turtle.speed(0)
scale_turtle.hideturtle()
scale_turtle.penup()

# Define uniform spacing
spacing = 50
num_divisions = 13

# Draw evenly spaced scale
for i in range(num_divisions):
    y_pos = 300 - (spacing * i)  # Consistent spacing per division
    scale_turtle.goto(-230, y_pos)
    scale_turtle.pendown()
    scale_turtle.goto(-210, y_pos)  # Short horizontal tick mark
```

```
    scale_turtle.penup()
    scale_turtle.goto(-200, y_pos - 5)  # Label position
    scale_turtle.write(f"{i * spacing} m", font=("Caladea", 12, "bold"))

# Add labels
scale_turtle.goto(-280, 330)
scale_turtle.write("Distance fallen", font=("Caladea", 14, "bold"))
scale_turtle.goto(-280, 380)
scale_turtle.write("Velocity in m/s", font=("Caladea", 14, "bold"))
scale_turtle.goto(-280, 360)
scale_turtle.write("Acceleration in m/s²", font=("Caladea", 14, "bold"))
scale_turtle.goto(-30, 400)
scale_turtle.write("In Air", font=("Caladea", 14, "bold"))
scale_turtle.goto(150, 400)
scale_turtle.write("In Vacuum", font=("Caladea", 14, "bold"))
scale_turtle.goto(-280,355)
scale_turtle.pendown()
scale_turtle.goto(300,355)
scale_turtle.penup()
scale_turtle.goto(-20,-300)
scale_turtle.pendown()
scale_turtle.goto(240,-300)
scale_turtle.penup()
scale_turtle.goto(50,-340)
scale_turtle.write("Ground Level", font=("Caladea", 14, "bold"))


# Create air resistance ball
falling_obj = turtle.Turtle()
falling_obj.hideturtle()
falling_obj.shape("circle")
falling_obj.shapesize(3)
falling_obj.color("blue")  # Air resistance ball
falling_obj.penup()
falling_obj.goto(20,320)
falling_obj.showturtle()
# Start at same height

# Create vacuum ball (falls with pure gravity)
vacuum_obj = turtle.Turtle()
vacuum_obj.hideturtle()
vacuum_obj.shape("circle")
vacuum_obj.shapesize(3)
vacuum_obj.color("red")  # Vacuum ball
vacuum_obj.penup()
vacuum_obj.goto(200,320)
vacuum_obj.showturtle()
```

```
# Start at same height, shifted right

# Create speed display
speed_display = turtle.Turtle()
speed_display.hideturtle()
speed_display.penup()
speed_display.goto(-30, 380)  # Position speed label lower

# Create acceleration display (right of speed label)
accel_display = turtle.Turtle()
accel_display.hideturtle()
accel_display.penup()
accel_display.goto(-30, 360)  # Position acceleration label lower

# Create vacuum speed display (initialized once)
vacuum_display = turtle.Turtle()
vacuum_display.hideturtle()
vacuum_display.penup()

# Variables
velocity = 0
position_y = falling_obj.ycor()

# Vacuum object variables (independent from air resistance ball)
vacuum_velocity = 0
vacuum_position_y = vacuum_obj.ycor()

acceleration = g * (1 - velocity / terminal_vel)
# Initial acceleration due to gravity

time_steps = []
velocities = []
accelerations = []
positions = []

# Simulation loop
while position_y > ground_level or vacuum_position_y > ground_level:
    ## AIR RESISTANCE BALL MOTION (BLUE)
    if velocity < 5:
        acceleration = g
    elif velocity < terminal_vel * 0.8:
        drag_force = 0.5 * C * rho * A * velocity**2  # Air resistance present
        acceleration = g - (drag_force / m)  # Air resistance reduces␣
 ↪acceleration
        acceleration = max(g * (1 - velocity / terminal_vel), 0.1)
    else:
        acceleration = 0  # Terminal velocity reached
```

```
    ## VACUUM BALL MOTION (RED) - ONLY GRAVITY
    # Ensure vacuum ball stops at ground level **and stores final velocity**
    if (vacuum_position_y - ground_level) > 5:
        vacuum_velocity += g * dt  # Constant acceleration in vacuum
        vacuum_position_y -= vacuum_velocity * dt  # Falling position update
    else:
        vacuum_velocity = vacuum_velocity  # Store last velocity BEFORE stopping
        vacuum_position_y = ground_level-3  # Stop at ground
        scale_turtle.goto(240,-270)
        scale_turtle.write("Impact", font=("Caladea", 14,  "bold") )

    # Update air resistance ball
    velocity += acceleration * dt
    position_y -= velocity * dt

    time_steps.append(len(time_steps) * dt)
    velocities.append(velocity)
    accelerations.append(acceleration)
    positions.append(position_y)

    falling_obj.goto(20, position_y)  # Air resistance case
    #falling_obj.showturtle()

    vacuum_obj.goto(200, vacuum_position_y)  # Vacuum case
    #vacuum_obj.showturtle()

    # Update speed display
    speed_display.clear()
    speed_display.write(f"{velocity:.2f} m/s", font=("Caladea", 14, "bold"))

    # Update acceleration display
    accel_display.clear()
    accel_display.write(f"{acceleration:.2f} m/s²", font=("Caladea", 14,␣
↪"bold"))

    # Update vacuum speed & acceleration display **inside the loop**
    vacuum_display.clear()
    vacuum_display.goto(150, 380)  # Position under "In Vacuum"
    vacuum_display.write(f"{vacuum_velocity:.2f} m/s", font=("Caladea", 14,␣
↪"bold"))

    vacuum_display.goto(150, 360)  # Position acceleration display
    vacuum_display.write("9.81 m/s²", font=("Caladea", 14, "bold"))  # Constant␣
↪acceleration
    # Delay for visualization
    time.sleep(dt)
```

```
scale_turtle.goto(60,-270)
scale_turtle.write("Impact", font=("Caladea", 14, "bold") )
```

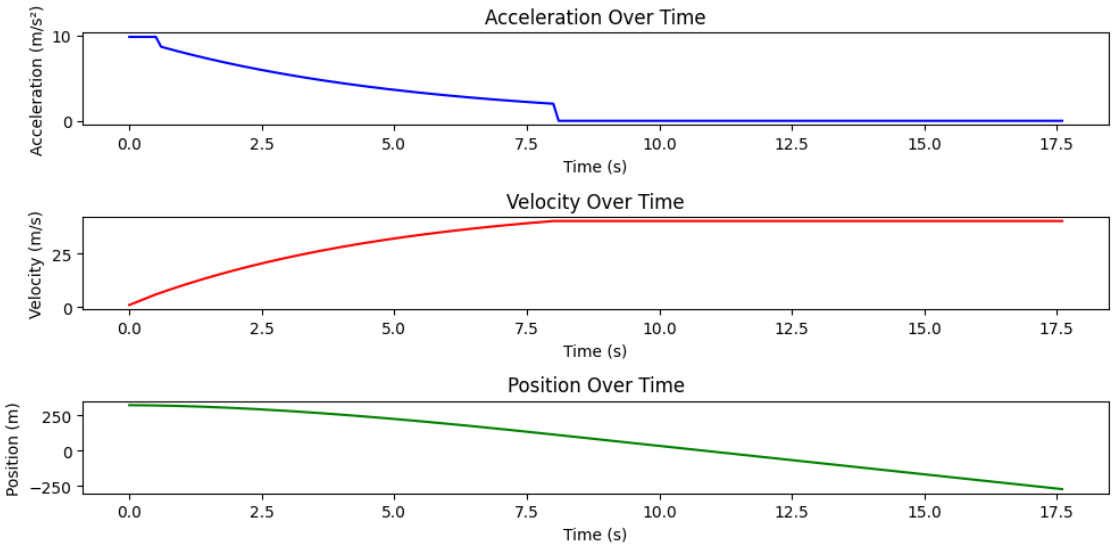
[2]:
```
plt.figure(figsize=(10,5))

# Acceleration plot
plt.subplot(3,1,1)
plt.plot(time_steps, accelerations, color='blue')
plt.xlabel("Time (s)")
plt.ylabel("Acceleration (m/s²)")
plt.title("Acceleration Over Time")

# Velocity plot
plt.subplot(3,1,2)
plt.plot(time_steps, velocities, color='red')
plt.xlabel("Time (s)")
plt.ylabel("Velocity (m/s)")
plt.title("Velocity Over Time")

# Position plot
plt.subplot(3,1,3)
plt.plot(time_steps, positions, color='green')
plt.xlabel("Time (s)")
plt.ylabel("Position (m)")
plt.title("Position Over Time")

plt.tight_layout()
plt.show(block=True)
```



[3]:
```
print("Time_steps:", time_steps[:10])
print("Accelerations:", accelerations[:100])
print("Velocities:", velocities[:10])
print("Positions:", positions[:10])
```

```
Time_steps: [0.0, 0.1, 0.2, 0.30000000000000004, 0.4, 0.5, 0.6000000000000001,
0.7000000000000001, 0.8, 0.9]
Accelerations: [9.81, 9.81, 9.81, 9.81, 9.81, 9.81, 8.6551668,
8.485352427383999, 8.318869812758727, 8.1556535870324, 7.995639663654825,
7.838765213453917, 7.684968639965952, 7.534189555249819, 7.386368756175817,
7.241448201179648, 7.099370987472502, 6.960081328698292, 6.823524533029232,
6.689646981691198, 6.558396107910418, 6.429720376273215, 6.3035692624907345,
6.179893233560666, 6.058643728318206, 5.939773138368602, 5.823234789393811,
5.708982922825904, 5.596972677880061, 5.4871600739400535, 5.37950199328935,
5.273956164181011, 5.170481144239781, 5.069036304189796, 4.969581811901592,
4.872078616752083, 4.776488434291408, 4.68277373121061, 4.590897710604258,
4.500824297522201, 4.412518124804817, 4.325944519196146, 4.241069487729518,
4.1578597043802645, 4.076282496980323, 3.9963058343895694, 3.9178983139188457,
3.8410291489997577, 3.7656681570963824, 3.6917857478541514, 3.6193529114812537,
3.5483412073579914, 3.478722752869627, 3.4104702124583257, 3.3435567868898923,
3.2779562027311133, 3.2136427020335283, 3.15059103221963, 3.0887764361674814,
3.028174642489876, 2.968761856004224, 2.9105147483894207, 2.853410449026019,
2.797426536016129, 2.742541027379492, 2.6887323724223067, 2.6359794432753803,
2.5842615265983175, 2.533558315446459, 2.4838499012974, 2.4351167662339446,
2.3873397752804353, 2.3405001688894327, 2.2945795555758224, 2.2495599046954236,
2.2054235393653, 2.1621531295229537, 2.1197316851217134, 2.0781425494596246,
2.0373693926392265, 1.997396205155645, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0]
Velocities: [0.9810000000000001, 1.9620000000000002, 2.9430000000000005,
3.9240000000000004, 4.905, 5.886, 6.75151668, 7.6000519227384,
8.431938904014272, 9.247504262717513]
Positions: [319.9019, 319.70570000000004, 319.4114, 319.019, 318.5285, 317.9399,
317.264748332, 316.50474313972614, 315.6615492493247, 314.73679882305294]
```

[4]:
```
## Command to print notebook (*.ipynb) file in pdf format:
## jupyter nbconvert --to pdf free_fall_objec.ipynb
```