Physics 2211 - Lab 2

Motion of a falling object

Advaith Menon[1]

[1]Computer Engineering
Georgia Institute of Technology

February 13th, 2024

Outline

Outline

Aim

Purpose of this lab assignment

- Observe the motion of an object that is **dropped from rest and falls somewhat slowly**, straight down.
- Analyzing the motion using software tools (Tracker) and capturing the object's positions at uniform time intervals.
- Constructing two computational models as below:
    1. One that only includes the effect of gravity
    2. Both the effect of gravity and of drag (air resistance)

Outline

Newton's Second Law

Quantitative analysis of motion

"The net force acting on a body is defined as the change in its momentum per unit time."

$$\vec{F}_{net} = \frac{\mathrm{d}\vec{p}}{\mathrm{d}t} \tag{1}$$

In most daily life scenarios, mass doesn't change with respect to time, hence this equation is better known as:

$$\vec{F}_{net} = m \cdot \vec{a}$$

where,

- $\vec{F}_{net}$ = Net force acting on a body
- m = Mass of the body
- $\vec{a} = \frac{\mathrm{d}\vec{v}}{\mathrm{d}t} = \frac{\mathrm{d}^2\vec{r}}{\mathrm{d}t^2}$ = Net acceleration on body
- $\Delta\vec{p} = m \times \Delta\vec{v}$ = Change in momentum

Corollary: The Momentum Principle

Rearranging the equation

$$\Delta\vec{p} = \vec{F}_{net} \times \Delta t \tag{2}$$

where,

- $\vec{F}_{net}$ = Net force acting on a body
- $\Delta\vec{p}$ = Change in momentum

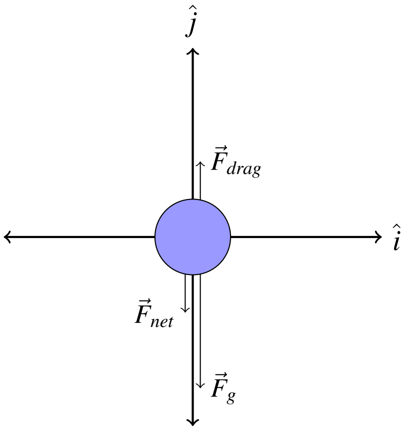
Initial Conditions

- Initial position, $\vec{r}_i = 0\hat{i} + 1.12 \times 10^{-2}\hat{j} + 0\hat{k}$ m
- Mass of the ball, $m_{ball} = 4.6$ g $= 4.6 \times 10^{-3}$ kg

System and Surroundings

- **System:** Paper ball (hereafter referred to as 'object')
- **Surroundings:** Everything else (table, air, etc.)

Free-Body Diagram



Outline

Analysis of Video

Getting displacement, velocity, acceleration at discrete time intervals using Tracker

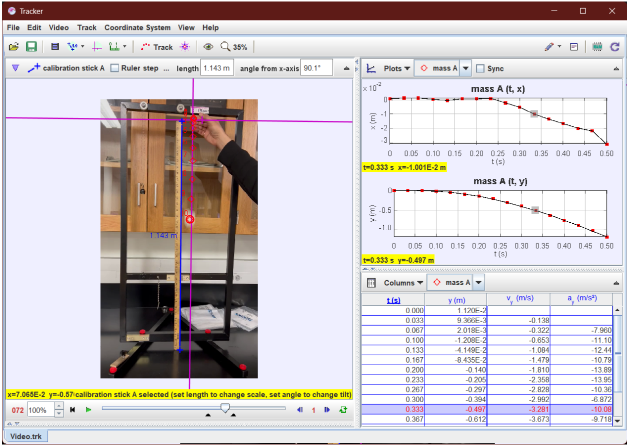


Figure 1: Using Tracker® to analyze the motion of the object.

Calculating Acceleration

Time t (s)	Y-Position r_y (m)
0.00E+00	1.12E-02
3.33E-02	9.37E-03
6.67E-02	2.02E-03
1.00E-01	-1.21E-02
1.33E-01	-4.15E-02
1.67E-01	-8.43E-02
⋮	⋮

- $$\vec{v}_{avg,2} = \frac{\vec{r}_3 - \vec{r}_1}{t_3 - t_1}$$
- Unlike the last lab, slightly larger time intervals have been used to ensure more accuracy.

Calculating Acceleration

- Repeating this step multiple times, we can get the average velocity for each interval
- Next step - calculate the acceleration

$$\vec{a}_{avg,3} = \frac{\vec{v}_4 - \vec{v}_2}{t_4 - t_2}$$

t (s)	r_y (m)	v_y (m)
0.00E+00	1.12E-02	
3.33E-02	9.37E-03	-1.38E-01
6.67E-02	2.02E-03	-3.22E-01
1.00E-01	-1.21E-02	-6.53E-01
1.33E-01	-4.15E-02	-1.08E+00
1.67E-01	-8.43E-02	-1.48E+00
	⋮	⋮

Georgia Tech

Calculating Acceleration

Some things to keep in mind

- Repeating this step multiple times, we can get the average acceleration for each interval
- Ideally, the average acceleration for each interval should be constant, but due to various factors such as air resistance , friction on the surface, etc, the acceleration is not constant.
- Hence, we average the accelerations of these small steps to get an accurate reading.

Model (without drag)

Data needed for simulation

- Net acceleration on object, `a_net` $= -g\hat{j}$ ms^{-2} $= -9.8\hat{j}$ ms^{-2}
- Vertical displacement, `delta_h` $= -1.17\hat{j}$ m
- Initial position $= 1.12 \times 10^{-2}\hat{j}$ m
- Mass of the ball, $m_{ball} = 4.6 \text{ g} = 4.6 \times 10^{-3} \text{ kg}$

Model (without drag)

Simulation code

```
# CONSTANTS
ACC_G = 9.8 # ms-2

ball = sphere(color=color.blue,
↪   radius=0.22) # sphere ball = new
↪   sphere(color.BLUE, 0.22);
trail = curve(color=color.green,
↪   radius=0.02)
origin = sphere(pos=vector(0,0,0),
↪   color=color.yellow, radius=0.04)
plot = graph(title="Position vs Time",
↪   xtitle="Time (s)", ytitle="Position
↪   (m)")
poscurve = gcurve(color=color.green,
↪   width=4)
plot = graph(title="Velocity vs Time",
↪   xtitle="Time (s)", ytitle="Velocity
↪   (m/s)")
velcurve = gcurve(color=color.green,
↪   width=4)
plot = graph(title="Acceleration vs
↪   Time", xtitle="Time (s)",
↪   ytitle="Velocity (m/s2)")
acccurve = gcurve(color=color.green,
↪   width=4)

# SYSTEM PROPERTIES & INITIAL CONDITIONS
ball.m = 4.6e-3 #kg
ball.pos = vector(0,1.12e-2,0) # m
ball.vel = vector(0,0,0) #
ball.acc = vector(0, -ACC_G, 0) #ms-2

# Time
t = 0             # where the clock starts
deltat = 0.001    # size of each timestep

while ball.pos.y >= -1.157057:
    rate(1000)

    # Apply the Momentum Principle
    ↪   (Newton's 2nd Law)
    ball.vel = ball.vel + ball.acc*deltat
    ball.pos = ball.pos + ball.vel*deltat

    t = t + deltat
    trail.append(pos=ball.pos)

    poscurve.plot(t,ball.pos.y)
    velcurve.plot(t,ball.vel.y)
    acccurve.plot(t,ball.acc.y);
    print(t,ball.pos.x)

print("All done!")
```

Model (with drag)

Data needed for simulation

- Net acceleration on object, `a_net` $= -g\hat{j}$ ms^{-2} $= -9.8\hat{j}$ ms^{-2}
- Vertical displacement, `delta_h` $= -1.17\hat{j}$ m
- Initial position $= 1.12 \times 10^{-2}\hat{j}$ m
- Mass of the ball, $m_{ball} = 4.6$ g $= 4.6 \times 10^{-3}$ kg
- **NOTE**: The data stays the same, the only thing to keep in mind is that we have to account for drag force too!
- `drag_force` = $b|v|^2\hat{j}$

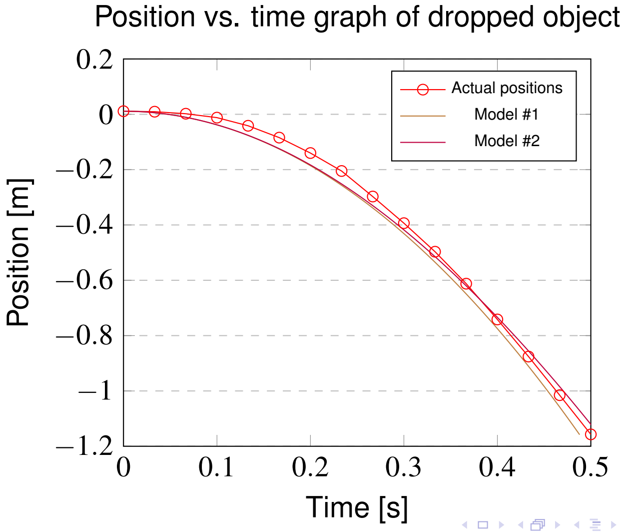
Model (with drag)

Simulation code

```
# CONSTANTS
ACC_G = 9.8 # ms-2
B = .001 # guessed by trial-and-error

ball = sphere(color=color.blue, radius=0.22) # sphere ball = new sphere(color.BLUE, 0.22);
trail = curve(color=color.green, radius=0.02)
origin = sphere(pos=vector(0,0,0), color=color.yellow, radius=0.04)
plot = graph(title="Position vs Time", xtitle="Time (s)", ytitle="Position (m)")
poscurve = gcurve(color=color.green, width=4)
plot = graph(title="Velocity vs Time", xtitle="Time (s)", ytitle="Velocity (m/s)")
velcurve = gcurve(color=color.green, width=4)
plot = graph(title="Acceleration vs Time", xtitle="Time (s)", ytitle="Velocity (m/s2)")
acccurve = gcurve(color=color.green, width=4)

# SYSTEM PROPERTIES & INITIAL CONDITIONS
ball.pos = vector(0,1.12e-2,0) # m
ball.vel = vector(0,0,0) #
ball.acc = vector(0, -ACC_G, 0) #ms-2

# Time
t = 0            # where the clock starts
deltat = 0.001   # size of each timestep

while ball.pos.y >= -1.157057:
    rate(1000)

    # Apply the Momentum Principle (Newton's 2nd Law)
    acc = (ball.acc + (B*ball.vel.y**2/ball.m)*vector(0, 1, 0))
    ball.vel = ball.vel + acc*deltat
    ball.pos = ball.pos + ball.vel*deltat

    t = t + deltat
    trail.append(pos=ball.pos)

    poscurve.plot(t,ball.pos.y)
    velcurve.plot(t,ball.vel.y)
    acccurve.plot(t,acc.y);
    print(t,ball.pos.x)
print("All done!")
```

Comparison of Results

Predicted vs. real



Outline

What does it mean?

Which model predicts terminal velocity?

- The second model is the only one to predict a terminal velocity.
- This is because at some point, $bv^2\hat{j} = mg\hat{j}$, after which this body has attained dynamic equilibrium.
- The graph does not evidently show it as the height is too small for the object to attain terminal velocity.

What if...

...the initial velocity was nonzero?

- Terminal velocity will still be attained at the same point.
- This is because for some v (say v', $bv^2\hat{j} = mg\hat{j}$ - note that this does not depend on the initial velocity of the body. If we threw the body at v', then there will be no net force on the body, thus it will be at terminal velocity.