Physics Lab 2: Motion of Falling Object

Wesley Lu

Introduction

Introduction - Fundamentals

Newton’s Second Law

$$\vec{v_f} = (\frac{\vec{F_{net}}}{m})\Delta t + \vec{v_i} \qquad \vec{F_{net}} = m \cdot a$$

$$\vec{p_f} = \vec{F_{net}} \cdot \Delta t + \vec{p_i}$$

These laws are used in order to calculate the position and velocity over time.

Gravitational Force

$$\vec{F}_{grav} = -G\frac{m_{obs} * m_{source}}{|\vec{r}|^2} * \hat{r}$$

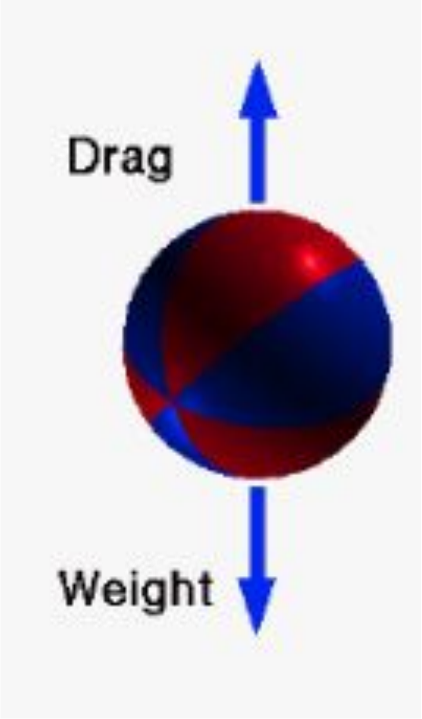
$$|\vec{F}_{grav}| = G\frac{m_2 * m_1}{|\vec{r}|^2}$$

The force of gravity is approximately 9.8 m/s^s. The force felt upon is equal to the mass time the force of gravity

Introduction - Drag Force

- The force known as air resistance that acts upon a falling object
- proportional to velocity by the drag coefficient
- drag coefficient is different for various objects
- as velocity increases the drag force becomes greater as the velocity is greater due to acceleration
- as drag equals to weight, acceleration is zero thus velocity is constant, called terminal velocity

$$\vec{F}_{\text{drag}} = -bv^2\,\hat{v}$$

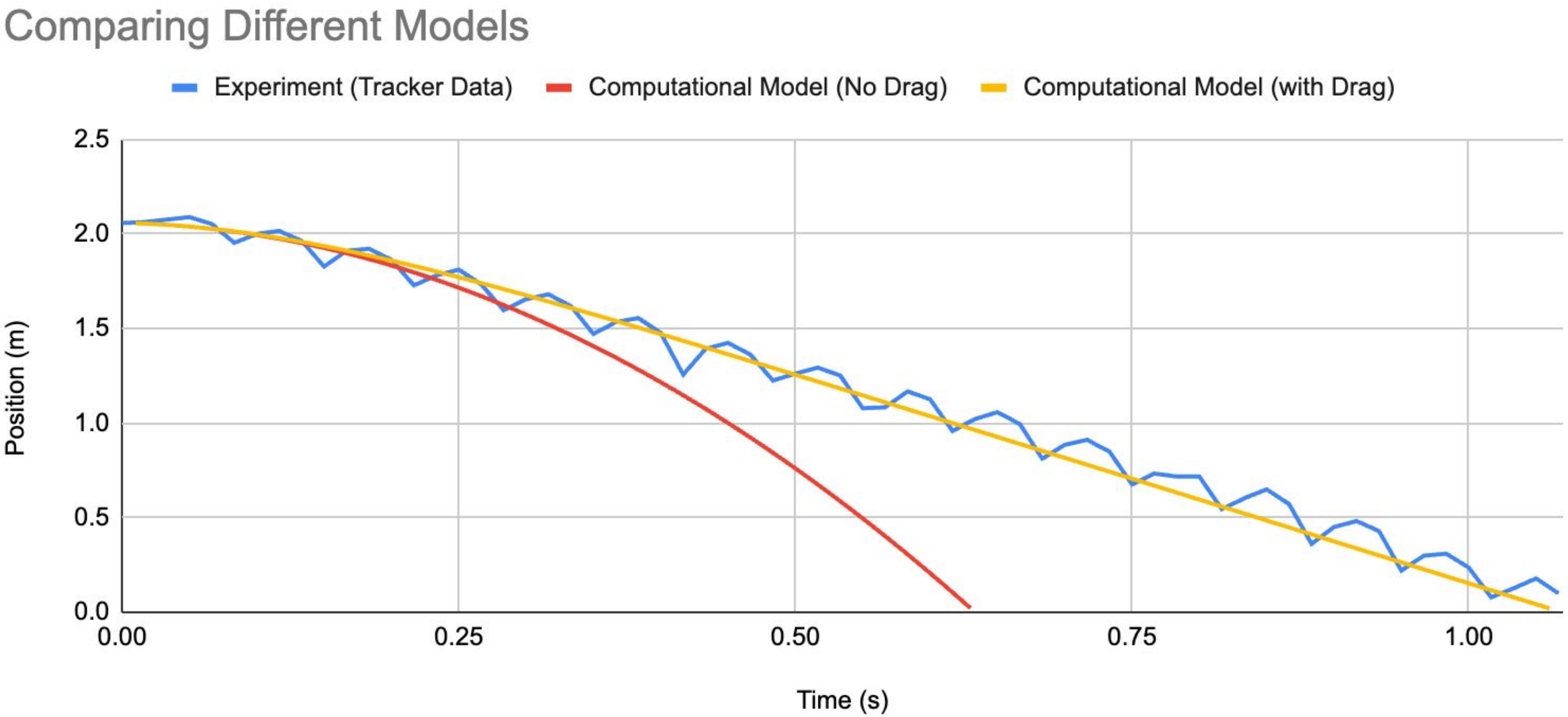


Introduction - Purpose

Observe the motion of an object that is dropped from rest whilst comparing the implications of drag force on difference computational models.

Use Tracker software to enable tracking and use observations as initial conditions for the computational model.

Introduction - Brief Preview

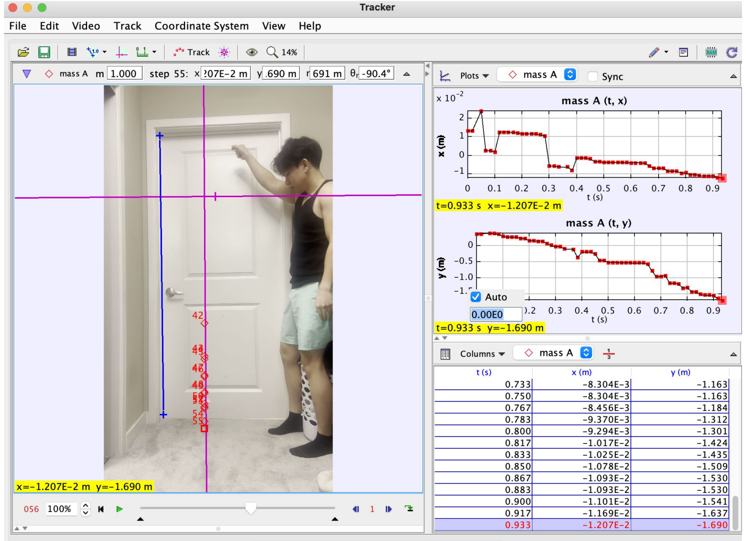


Experiment

Details of Experiment

1. Record the video used in tracker
2. Label weight, adjust y-axis and x-axis, and calibrate length
3. Plot video using each frame as a reference for position
4. Export the data
5. In all it should look like the following

System: Cotton Pad, Surrounding: Everything around
Weight: 0.5 Grams or 0.005 KG
Length of Door: 2 Meters
Assumptions: Motions is
in the y-axis



mass A m 1.000 step 55: x ?07E-2 m y .690 m r 691 m θr -90.4°

t=0.933 s x=-1.207E-2 m

t=0.933 s y=-1.690 m

0.00E0

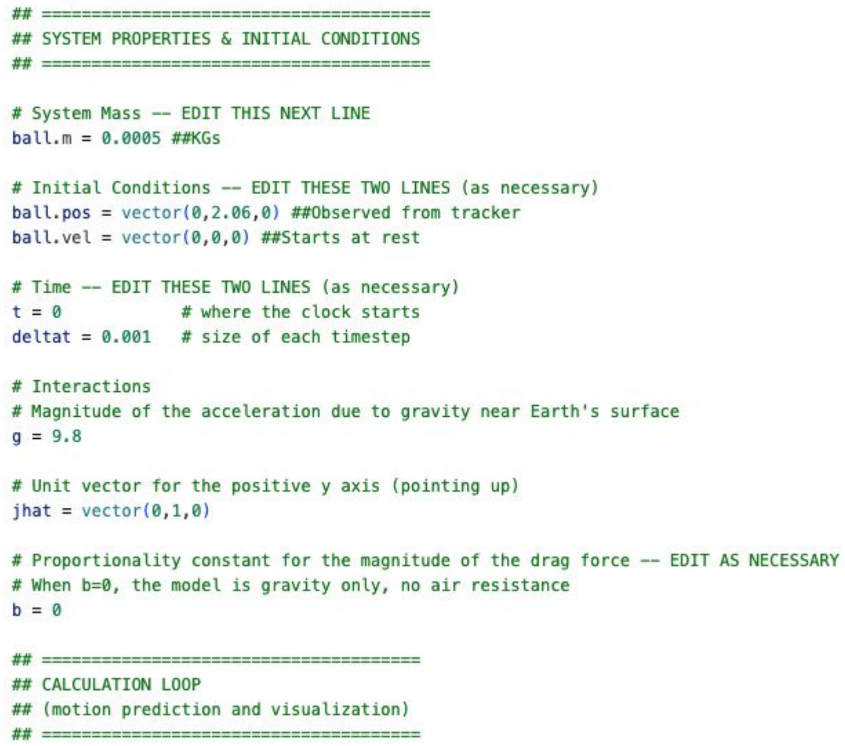
x=-1.207E-2 m y=-1.690 m

t (s)	x (m)	y (m)
0.733	-8.304E-3	-1.163
0.750	-8.304E-3	-1.163
0.767	-8.456E-3	-1.184
0.783	-9.370E-3	-1.312
0.800	-9.294E-3	-1.301
0.817	-1.017E-2	-1.424
0.833	-1.025E-2	-1.435
0.850	-1.078E-2	-1.509
0.867	-1.093E-2	-1.530
0.883	-1.093E-2	-1.530
0.900	-1.101E-2	-1.541
0.917	-1.169E-2	-1.637
0.933	-1.207E-2	-1.690

Code

Computational Model (No Drag) - Initial Conditions

- Ball weight = 0.005 KG
- Position: 2.06 Meter above the ground
- At rest initially so no velocity



```
## =======================================
## SYSTEM PROPERTIES & INITIAL CONDITIONS
## =======================================

# System Mass -- EDIT THIS NEXT LINE
ball.m = 0.0005 ##KGs

# Initial Conditions -- EDIT THESE TWO LINES (as necessary)
ball.pos = vector(0,2.06,0) ##Observed from tracker
ball.vel = vector(0,0,0) ##Starts at rest

# Time -- EDIT THESE TWO LINES (as necessary)
t = 0            # where the clock starts
deltat = 0.001   # size of each timestep

# Interactions
# Magnitude of the acceleration due to gravity near Earth's surface
g = 9.8

# Unit vector for the positive y axis (pointing up)
jhat = vector(0,1,0)

# Proportionality constant for the magnitude of the drag force -- EDIT AS NECESSARY
# When b=0, the model is gravity only, no air resistance
b = 0

## ======================================
## CALCULATION LOOP
## (motion prediction and visualization)
## ======================================
```

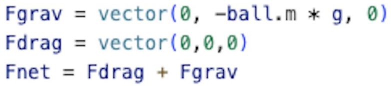
Computational Model (No Drag) - Equations

The Fnet only accounts for weight.

Newton’s Second Law/ Velocity Update Formula

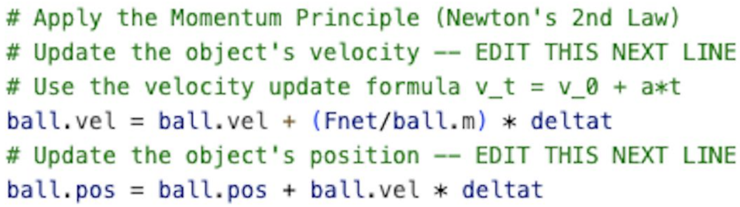
$$\vec{v_f} = (\frac{\vec{F_{net}}}{m})\Delta t + \vec{v_i}$$

$$\vec{F_{net}} = m \cdot a$$



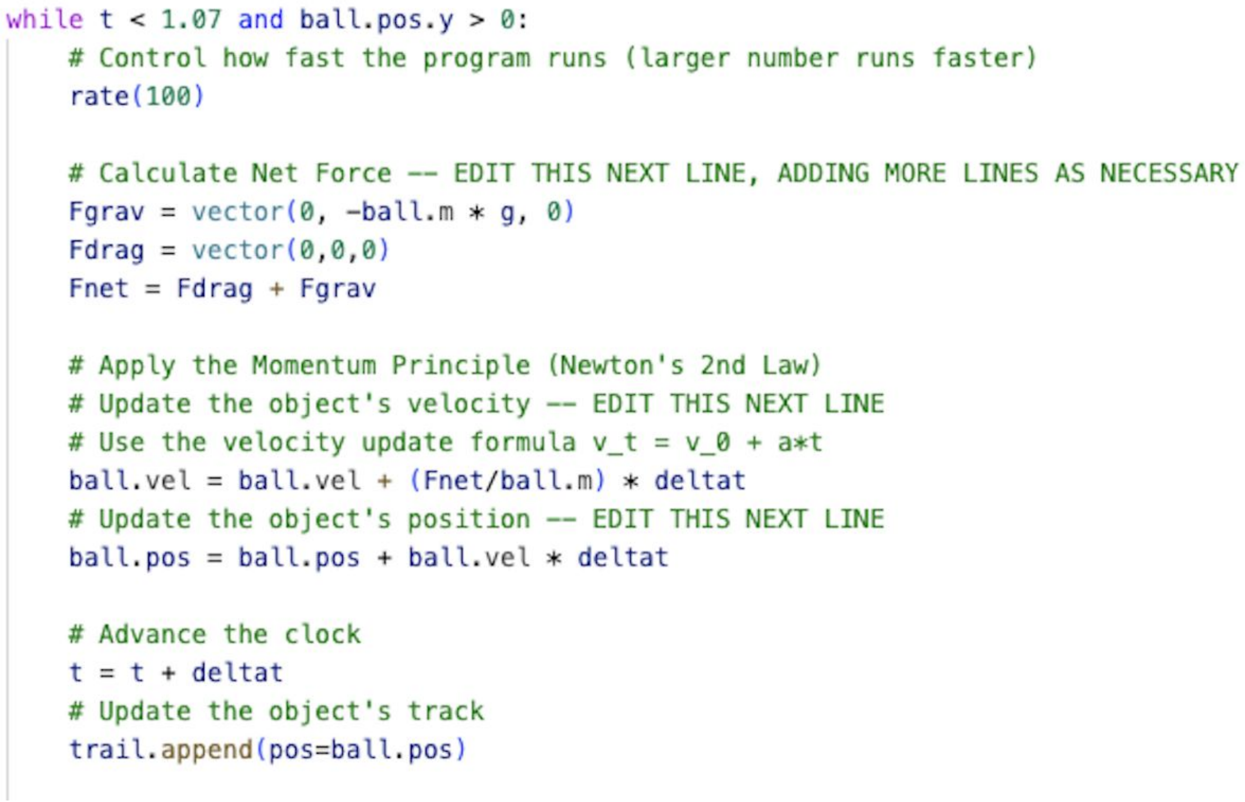
```
Fgrav = vector(0, -ball.m * g, 0)
Fdrag = vector(0,0,0)
Fnet = Fdrag + Fgrav
```

Update Formulas



```
# Apply the Momentum Principle (Newton's 2nd Law)
# Update the object's velocity -- EDIT THIS NEXT LINE
# Use the velocity update formula v_t = v_0 + a*t
ball.vel = ball.vel + (Fnet/ball.m) * deltat
# Update the object's position -- EDIT THIS NEXT LINE
ball.pos = ball.pos + ball.vel * deltat
```

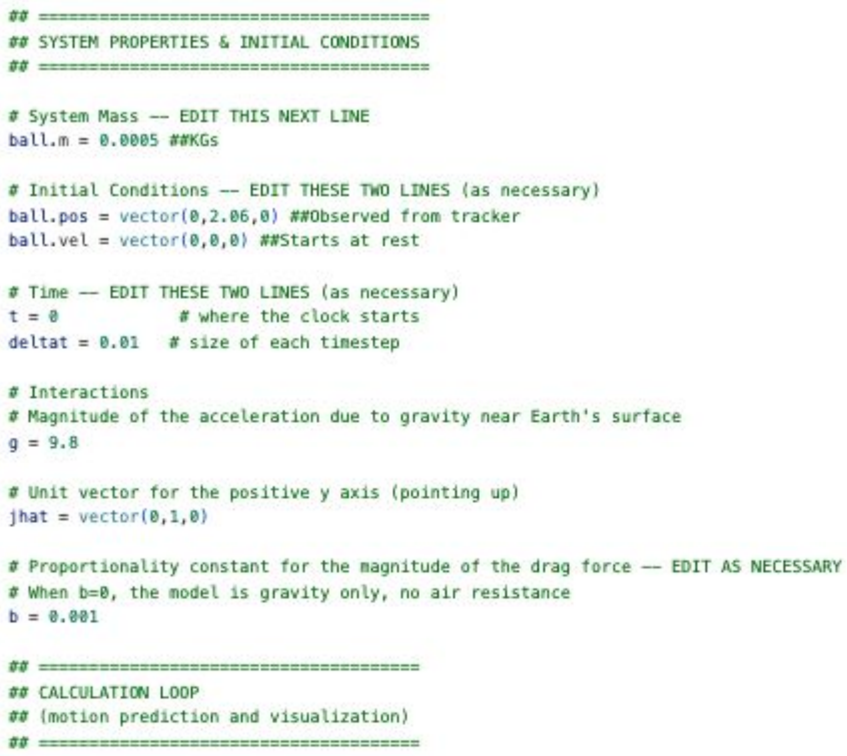
Calculation Loop



```
while t < 1.07 and ball.pos.y > 0:
    # Control how fast the program runs (larger number runs faster)
    rate(100)

    # Calculate Net Force -- EDIT THIS NEXT LINE, ADDING MORE LINES AS NECESSARY
    Fgrav = vector(0, -ball.m * g, 0)
    Fdrag = vector(0,0,0)
    Fnet = Fdrag + Fgrav

    # Apply the Momentum Principle (Newton's 2nd Law)
    # Update the object's velocity -- EDIT THIS NEXT LINE
    # Use the velocity update formula v_t = v_0 + a*t
    ball.vel = ball.vel + (Fnet/ball.m) * deltat
    # Update the object's position -- EDIT THIS NEXT LINE
    ball.pos = ball.pos + ball.vel * deltat

    # Advance the clock
    t = t + deltat
    # Update the object's track
    trail.append(pos=ball.pos)
```

Computational Model (With Drag) - Initial Conditions

- Ball weight = 0.005 KG
- Position: 2.06 Meter above the ground
- At rest initially so no velocity
- Drag Coefficient = 0.001

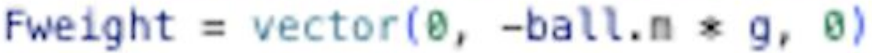


```
## =======================================
## SYSTEM PROPERTIES & INITIAL CONDITIONS
## =======================================

# System Mass -- EDIT THIS NEXT LINE
ball.m = 0.0005 ##KGs

# Initial Conditions -- EDIT THESE TWO LINES (as necessary)
ball.pos = vector(0,2.06,0) ##Observed from tracker
ball.vel = vector(0,0,0) ##Starts at rest

# Time -- EDIT THESE TWO LINES (as necessary)
t = 0            # where the clock starts
deltat = 0.01   # size of each timestep

# Interactions
# Magnitude of the acceleration due to gravity near Earth's surface
g = 9.8

# Unit vector for the positive y axis (pointing up)
jhat = vector(0,1,0)

# Proportionality constant for the magnitude of the drag force -- EDIT AS NECESSARY
# When b=0, the model is gravity only, no air resistance
b = 0.001

## ======================================
## CALCULATION LOOP
## (motion prediction and visualization)
## ======================================
```

Computational Model (With Drag) - Equations

Newton's Second Law

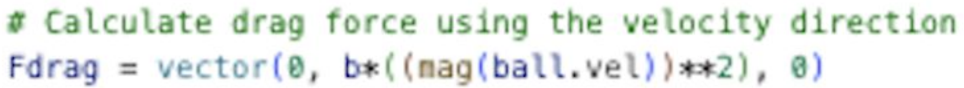
$$\vec{v_f} = (\frac{\vec{F_{net}}}{m})\Delta t + \vec{v_i} \qquad \vec{F_{net}} = m \cdot a$$



```
Fweight = vector(0, -ball.m * g, 0)
```

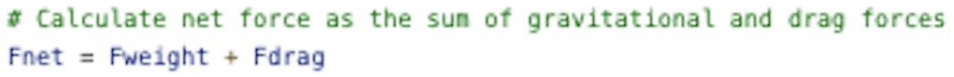
Drag Force

$$\vec{F}_{\text{drag}} = -bv^2\,\hat{v}.$$



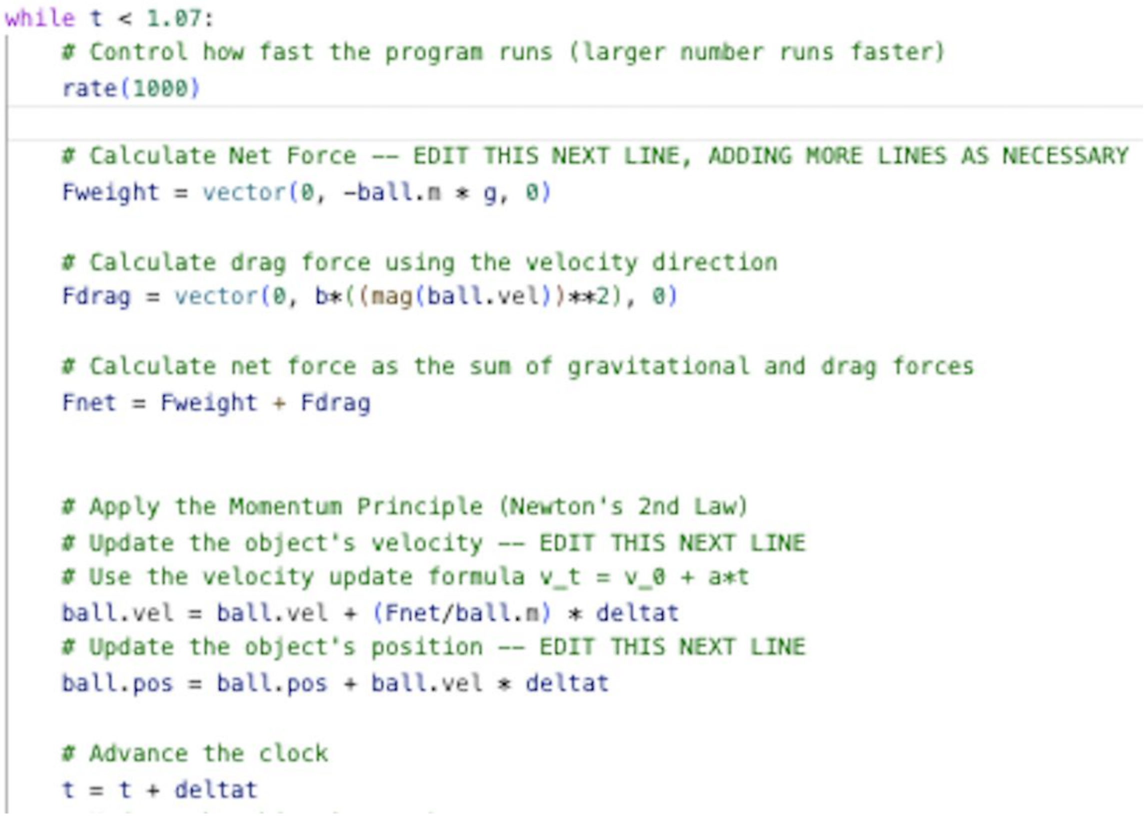
```
# Calculate drag force using the velocity direction
Fdrag = vector(0, b*((mag(ball.vel))**2), 0)
```

Total Net Force



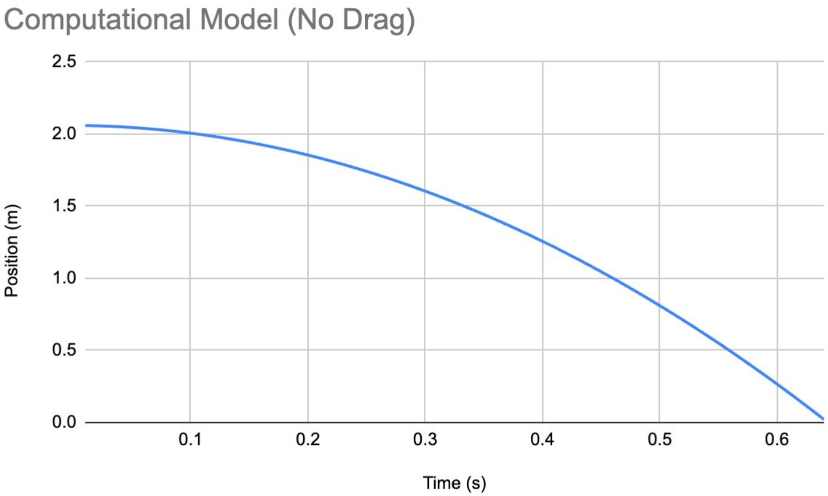
```
# Calculate net force as the sum of gravitational and drag forces
Fnet = Fweight + Fdrag
```

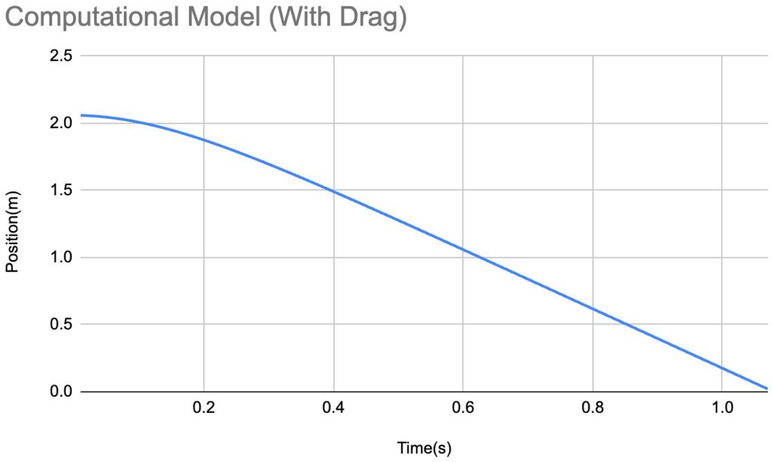
Calculation Loop



```
while t < 1.07:
    # Control how fast the program runs (larger number runs faster)
    rate(1000)


    # Calculate Net Force -- EDIT THIS NEXT LINE, ADDING MORE LINES AS NECESSARY
    Fweight = vector(0, -ball.m * g, 0)

    # Calculate drag force using the velocity direction
    Fdrag = vector(0, b*((mag(ball.vel))**2), 0)

    # Calculate net force as the sum of gravitational and drag forces
    Fnet = Fweight + Fdrag


    # Apply the Momentum Principle (Newton's 2nd Law)
    # Update the object's velocity -- EDIT THIS NEXT LINE
    # Use the velocity update formula v_t = v_0 + a*t
    ball.vel = ball.vel + (Fnet/ball.m) * deltat
    # Update the object's position -- EDIT THIS NEXT LINE
    ball.pos = ball.pos + ball.vel * deltat

    # Advance the clock
    t = t + deltat
```

Effects of Drag Force (Similarities + Differences)

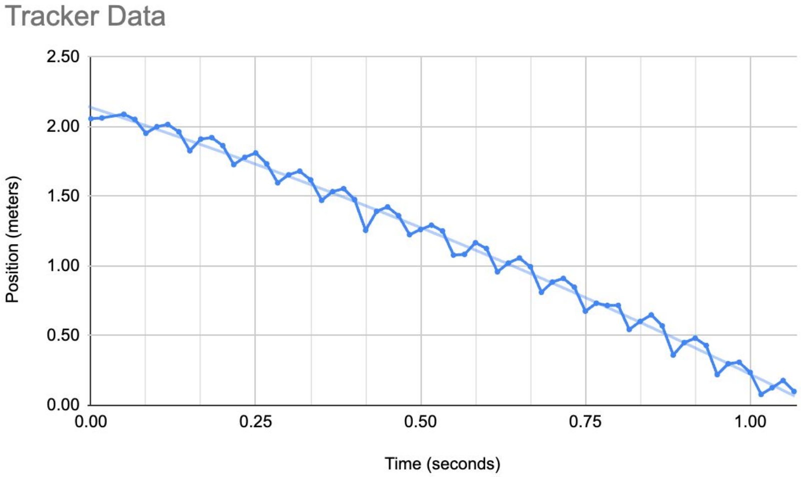




The model without drag has a significant difference in time.

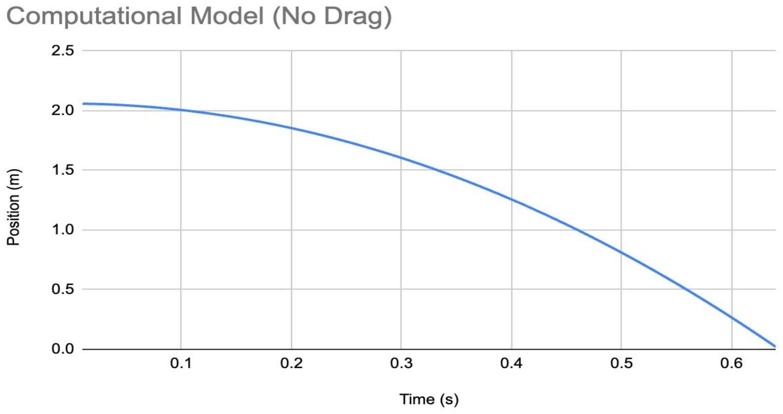
Potential Errors

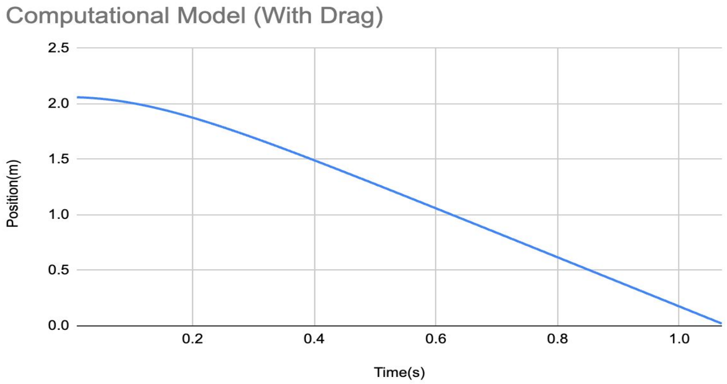
- tracker data has oscillations due to the inaccuracy to frame representation
- differences in the mass and position of simulation could be slightly different
- Inaccurate drag coefficient



Discussion

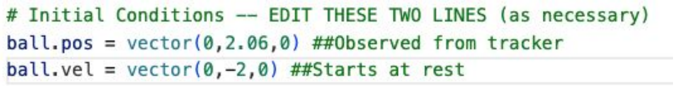
Discussion - Question 1)





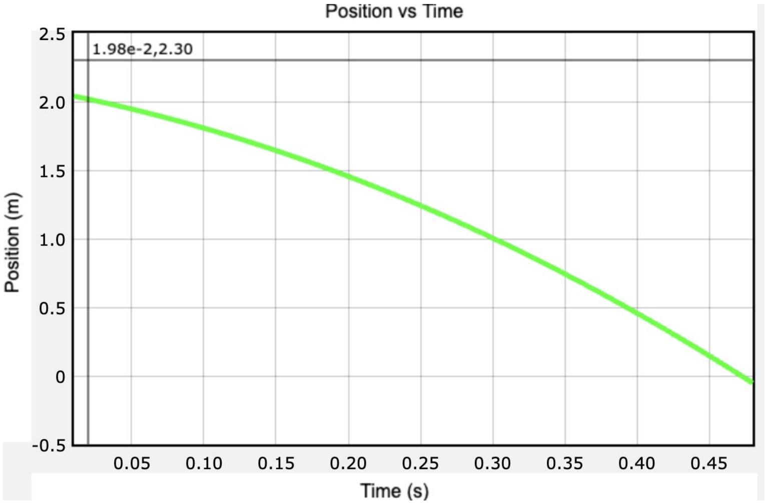
The computation model with drag force could potentially predict a terminal velocity. However there needs to be a greater height that the object is dropped from. This horizontal asymptote on the model would be the terminal velocity

Discussion - Question 2)



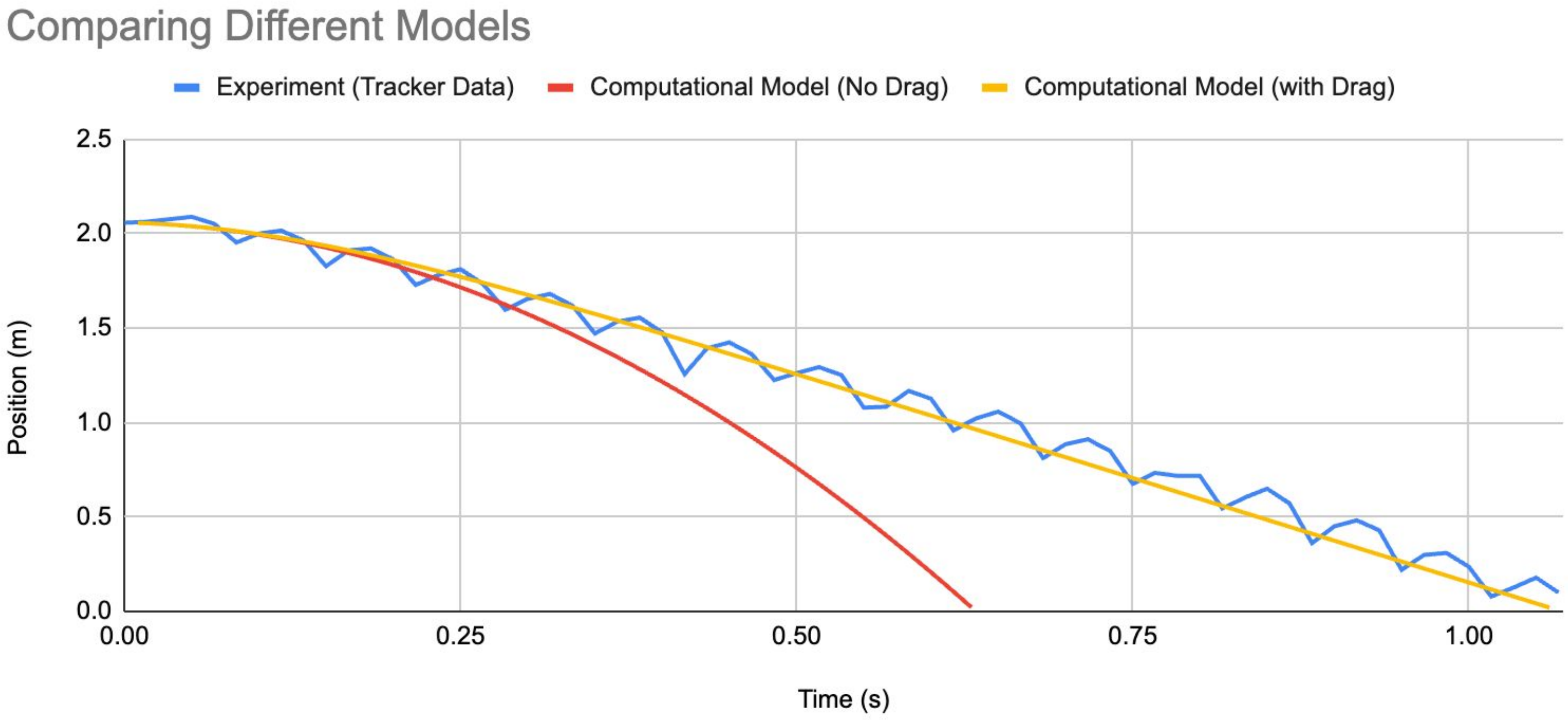
```
# Initial Conditions -- EDIT THESE TWO LINES (as necessary)
ball.pos = vector(0,2.06,0) ##Observed from tracker
ball.vel = vector(0,-2,0) ##Starts at rest
```

With a initial velocity thrown downwards, it would experience the terminal velocity quicker due to the acceleration from the initial throw. However, the terminal velocity would have the same velocity.



Conclusion

Conclusion - Result & Findings



Conclusion - Finding

- Model with drag had a better representation of the drop assuming a drag coefficient of 0.001
- The model without drag is to assume all other force besides gravity negligible