Physics Lab 1: Constant Motion

Wesley Lu

Introduction

Introduction: Lab Goals

Lab Goals:

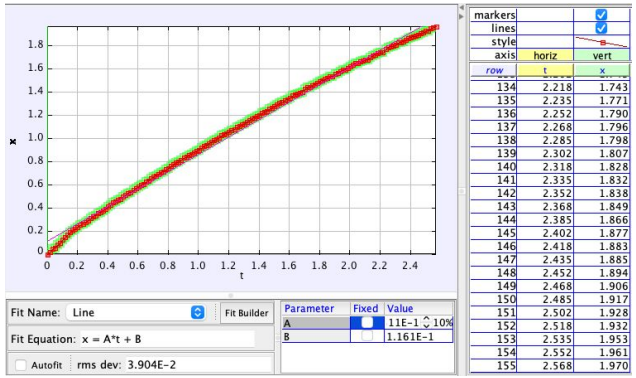
- Observe object moving in the same direction at same speed
- Capture the motion of the video using Tracker
- Analyze the tracked data
- Construct a computational model that describe my observations

Introduction: Physics Principles Explored

The **purpose** of this lab is to observe an object whilst explaining the principles of **Newton's 1st Law**, **Newton's 2nd Law**, and the **momentum principle**.

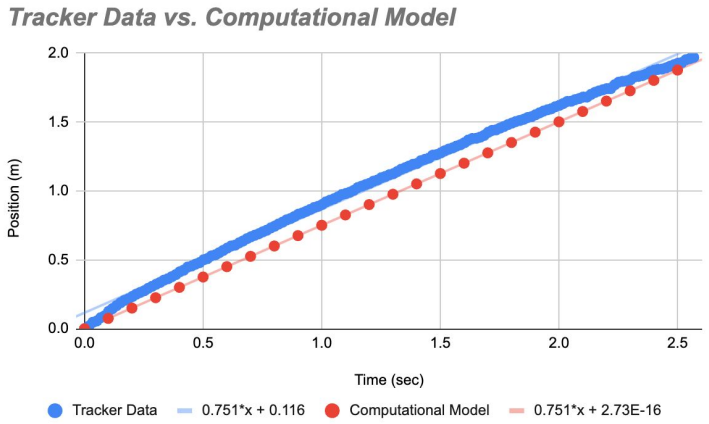
$$F_{net} = ma \qquad\qquad p = mv$$

Introduction: Results



markers		
lines		
style		
axis	horiz	vert
row	t	x
134	2.218	1.743
135	2.235	1.771
136	2.252	1.790
137	2.268	1.796
138	2.285	1.798
139	2.302	1.807
140	2.318	1.828
141	2.335	1.832
142	2.352	1.838
143	2.368	1.849
144	2.385	1.866
145	2.402	1.877
146	2.418	1.883
147	2.435	1.885
148	2.452	1.894
149	2.468	1.906
150	2.485	1.917
151	2.502	1.928
152	2.518	1.932
153	2.535	1.953
154	2.552	1.961
155	2.568	1.970

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

while t < 2.5:
    # Control how fast the program runs (larger number runs faster)
    rate(100)

    # Calculate Net Force -- EDIT THIS NEXT LINE (add more lines if necessary)
    Fnet = vector(0,0,0)

    # Apply the Momentum Principle (Newton's 2nd Law)
    # Update the object's velocity -- EDIT THIS NEXT LINE

    ball.vel = ball.vel

    # 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)

    # Plot position and velocity as a function of time
    poscurve.plot(t,ball.pos.x)
    velcurve.plot(t,ball.vel.x)

    # Displaying the position values
    # The information that gets printed is the same as the information that is p
    # EDIT THIS NEXT LINE if you want to output velocity instead
    print(t,ball.pos.x)

print("All done!")
```



Concepts + Explanation

Newton's Law and Momentum Principle

The 1st law states "an object at rest remains at rest, and an object in motion remains in motion at constant speed and in a straight line unless acted on by an unbalanced force." The 2nd law states "the force acted upon is equal to the mass times the acceleration upon the object." Finally the momentum principle says position is equal to velocity multiplied by its velocity.

$$F_{net} = ma$$

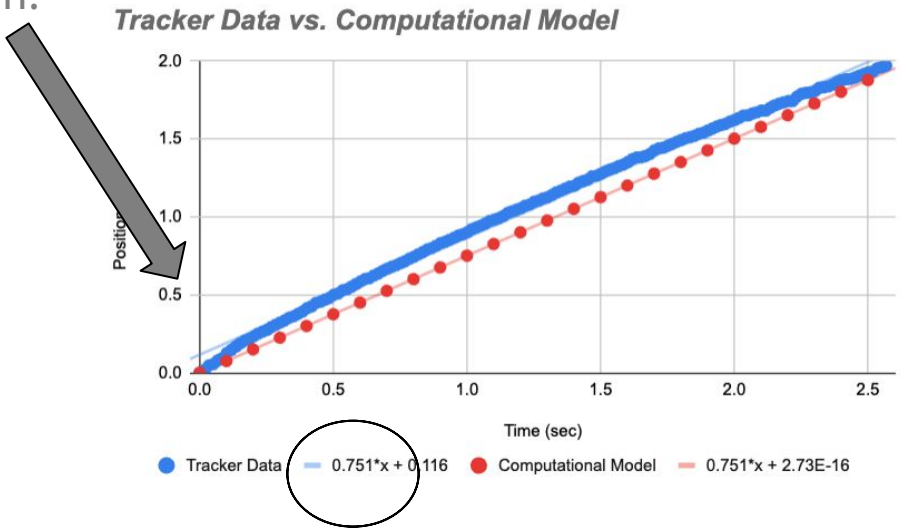
$$p = mv$$

$$F_{net} = \frac{\Delta p}{\Delta t}$$

Position and Velocity

The position of an object can be measured using different coordinate system labeled as <x, y, z>. This video observed the horizontal change in position across the table. The velocity is the change in position over the change in time. This is also the instantaneous rate of change in a position graph.

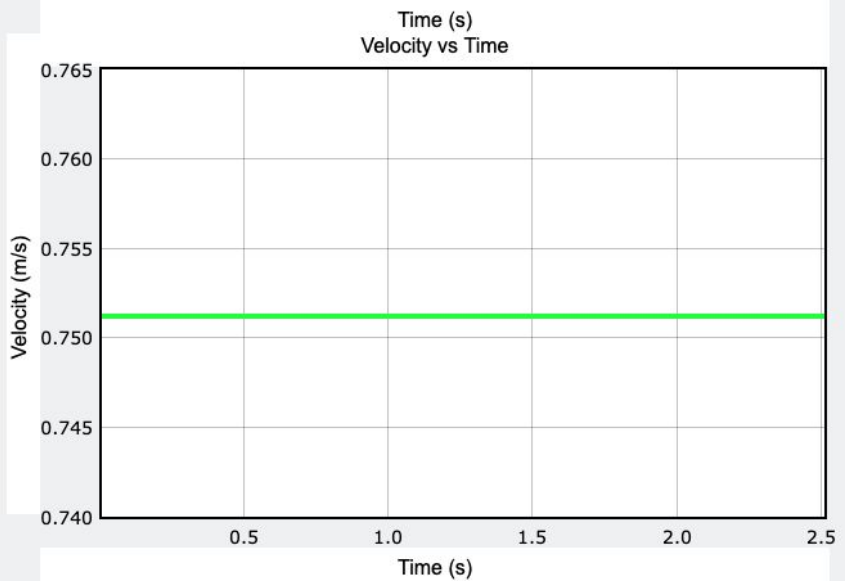
$$\overline{v} = \frac{\Delta x}{\Delta t}$$



Acceleration

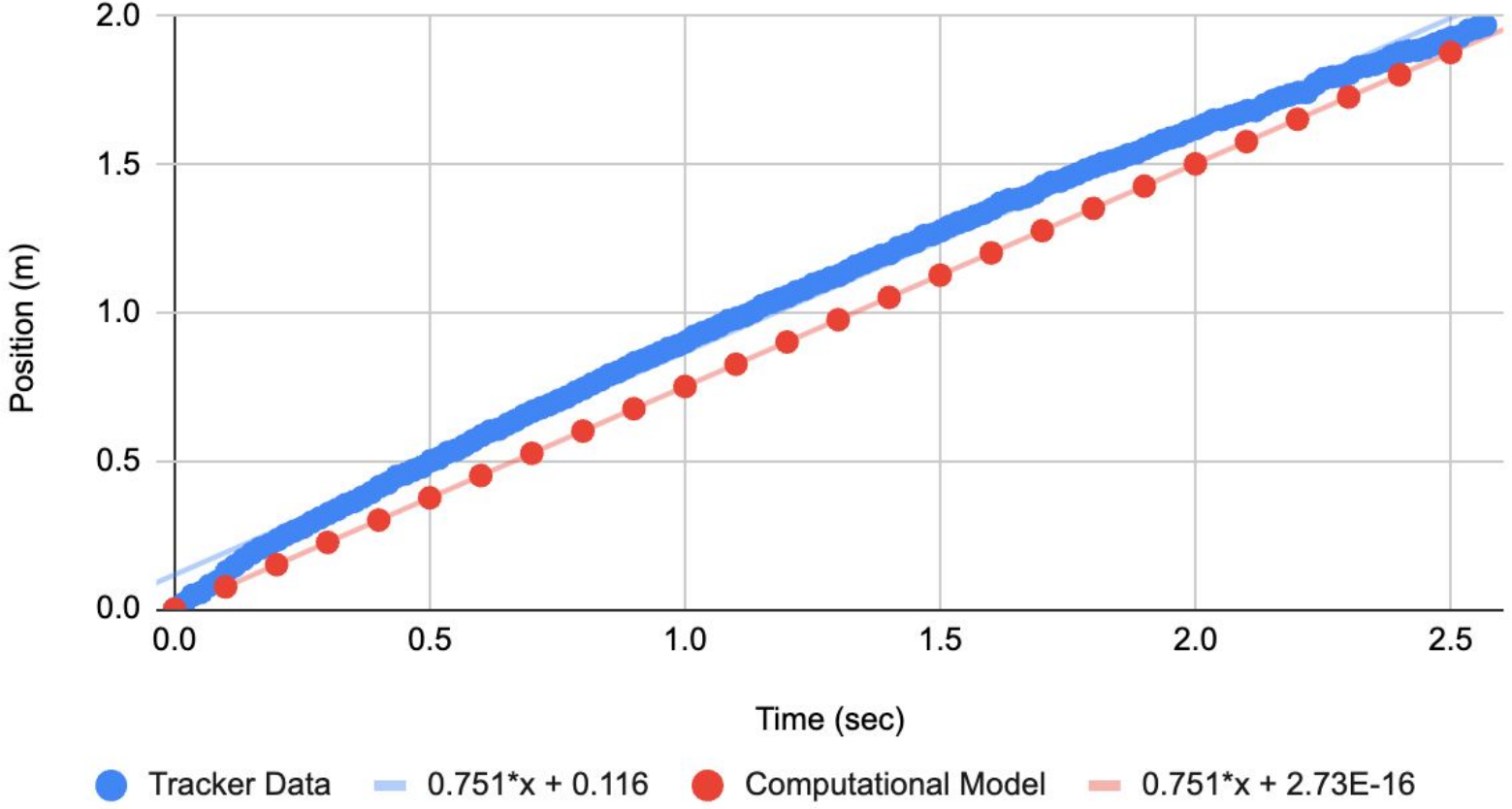
The instantaneous change for velocity is acceleration. Newton's second law includes acceleration. An object moving at constant motion with constant velocity has an acceleration of zero. This is because the change in velocity is zero.

$$a = \frac{\Delta v}{\Delta t}$$



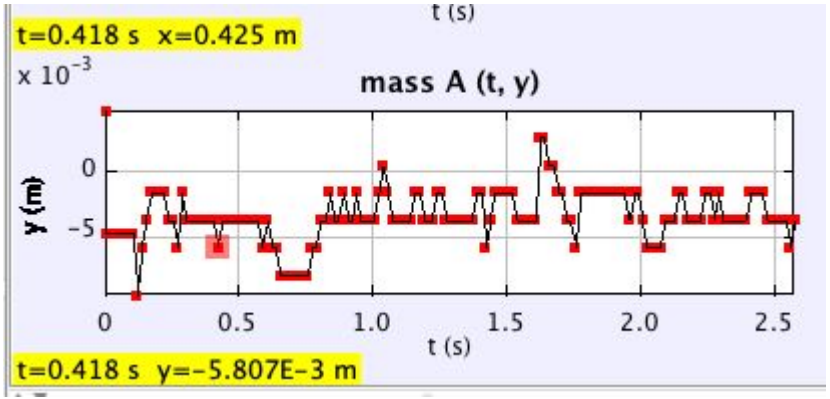
Models

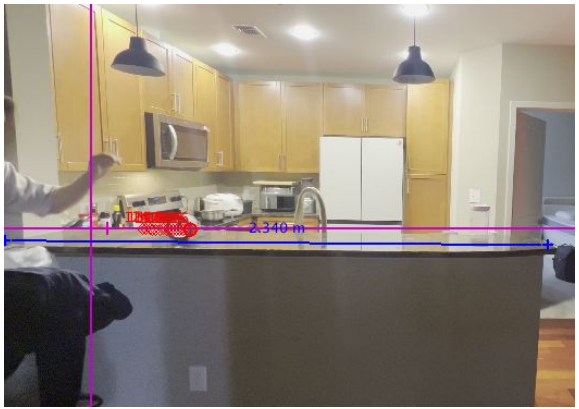
Tracker Data vs. Computational Model



Assumptions

- No Friction => Decrease in velocity
- No change in velocity => Constant Motion
- No deviation in horizontal change => Continuous Path
- Estimation of mass, length in tracker, and tracking errors can all lead to incorrect inputs into computational model



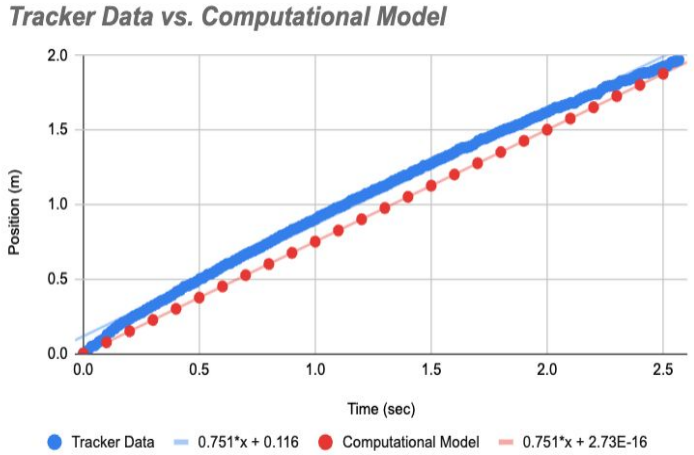


Tracker vs Python

- Both the tracker and computational model has a line of best fit velocity of 0.751 m/s. In the computation model, this is simple to configure however, the tracker models has imperfections based on how accurate a student will measure, click, and assess the change in position over each frame.

Errors and Initial Conditions

- There shouldn't be a y intercept
- Position is off by 0.12 meters than tracker



- The initial position is set to zero, velocity was set to 0.7512 meter/second
- ball.pos = vector(0,0,0)
- ball.vel = vector(0.7512,0,0)

Experiment + Coding

The system was the duct tape, the surrounding is everything else. Considering a duct tape has the center of mass at the most center point, I choose this as my object.

```
from vpython import *

scene.background = color.white

ball = sphere(color=color.blue, radius=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)

ball.m = 1
ball.pos = vector(0,0,0)
ball.vel = vector(0.7512,0,0)

t = 0
deltat = 0.01

while t < 2.5:
    rate(100)
    Fnet = vector(0,0,0)
    ball.vel = ball.vel
    ball.pos = ball.pos + ball.vel * deltat
    t = t + deltat
    trail.append(pos=ball.pos)
    poscurve.plot(t, ball.pos.x)
    velcurve.plot(t, ball.vel.x)
    print(t, ball.pos.x)

print("All done!")
```

Coding

The while loop estimates the time it took to travel across the table in the experiment.

The ball.vel is always set to itself as it doesn't change. The ball position is set to represent the new position correctly. The time is update correctly as well.

```
from vpython import *

scene.background = color.white

ball = sphere(color=color.blue, radius=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)

ball.m = 1
ball.pos = vector(0,0,0)
ball.vel = vector(0.7512,0,0)

t = 0
deltat = 0.01

while t < 2.5:
    rate(100)
    Fnet = vector(0,0,0)
    ball.vel = ball.vel
    ball.pos = ball.pos + ball.vel * deltat
    t = t + deltat
    trail.append(pos=ball.pos)
    poscurve.plot(t, ball.pos.x)
    velcurve.plot(t, ball.vel.x)
    print(t, ball.pos.x)

print("All done!")
```

Discussion

Questions?

I purposely flipped my video such that the system is moving towards the right. Within the tracker software, it uses standard coordinate systems where right is +x and left is -x. Thus, having the object move toward the left will result in a negative velocity since the displacement is towards the negative side.

In the computational model, the force is set to zero to replicate the newton's first law of an object staying in constant motion. This it takes another force to stop this object from moving.

Conclusions

Wrap Up

- The Net Force of the computational model should be zero since the velocity is always constant (0.751 m/s).
- There are heavy assumptions that don’t guarantee a replication of reality
- Within an empty vacuum, it’s easier to replicate an experiment with little to no assumptions
- Both the models have similar line of best fits