Throwing a Ball Can Be Such a Drag

Chris McCarthy & Kujtim Bardhyll

City University of New York
Borough of Manhattan Community College

Joint Mathematics Meetings, Denver, Colorado
January 15, 2020

SIMIODE DEMARC Workshop

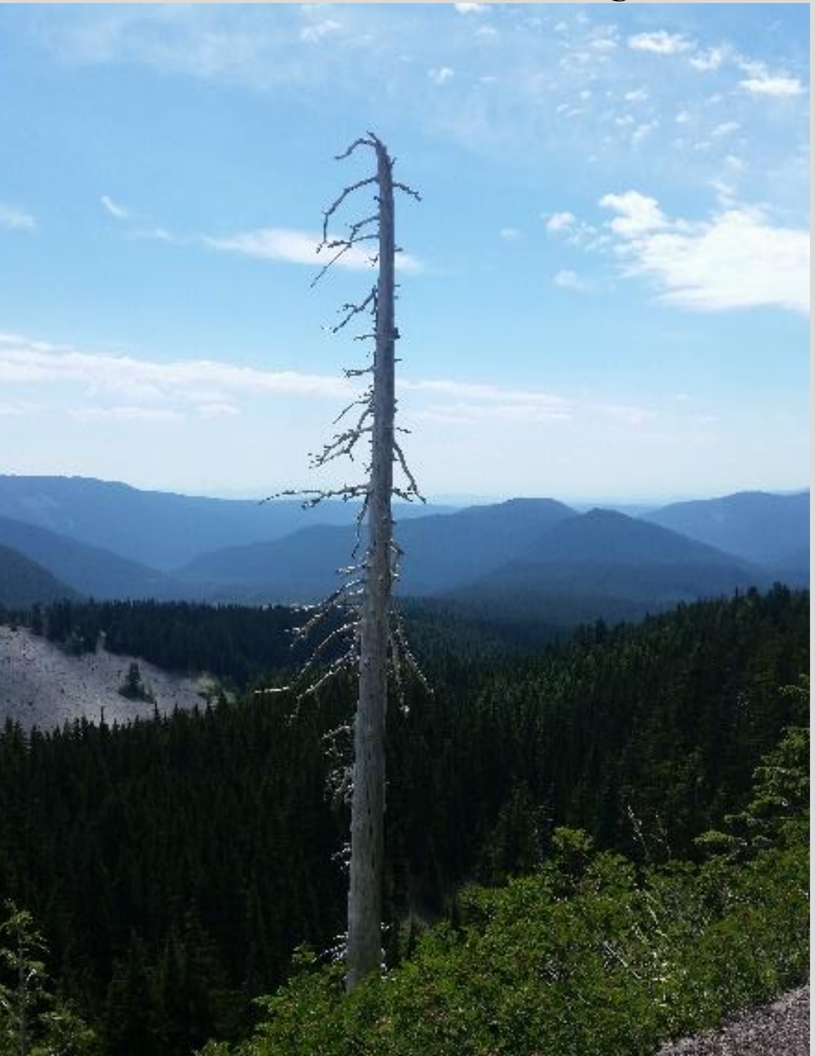
George Fox University

Oregon, July 2019

Beautiful campus
in
beautiful Oregon.

Differential Equations
Model And Resource
Creators Workshop

View from Mt. Hood, Oregon



SIMIODE DEMARC Workshop

George Fox University
Oregon, July 2019

SYSTEMIC INITIATIVE FOR MODELING INVESTIGATIONS & OPPORTUNITIES WITH DIFFERENTIAL EQUATIONS
SIMIODE

NSF

GEORGE FOX UNIVERSITY
1891
CHRISTIANITY AND CULTURE

DEMARC Goal

To Develop Diff Eq Modeling Projects

(That are good for students)

Leonhard Euler 1707 - 1783

I came up with a modeling project involving Euler's Method and drag

Professor McCarthy Mat 501 BMCC

Differential Equations

Euler, Drag, and How Far Can You Throw a Ball?

Associated Files

- detailed PDF on using Euler's method to calculate trajectories and drag (air resistance).
- Python script (*ThrownBallEulerStudent.py* – scroll down to see the script. You can copy and paste it into a Python compiler).

Description of scenario. If a tennis ball is thrown through the air it will eventually hit the ground due to gravity.

If you can throw a tennis ball 12 meters/second (about 26.8 mph) how far can you throw it; meaning how far away from you can you make it land?

Assume that when the ball leaves your hand it is at a height of 2 meters (about 6 feet). Your

My Differential Equations Class' Website

Just **Google**

McCarthy Differential Equations

The modeling project is also on **SIMIODE**

I show the class how to solve the following problem WITHOUT drag.
They have to solve the same problem WITH drag.

If you can throw a tennis ball 12 meters/second (about 26.8 mph) how far can you throw it; meaning how far away from you can you make it land?

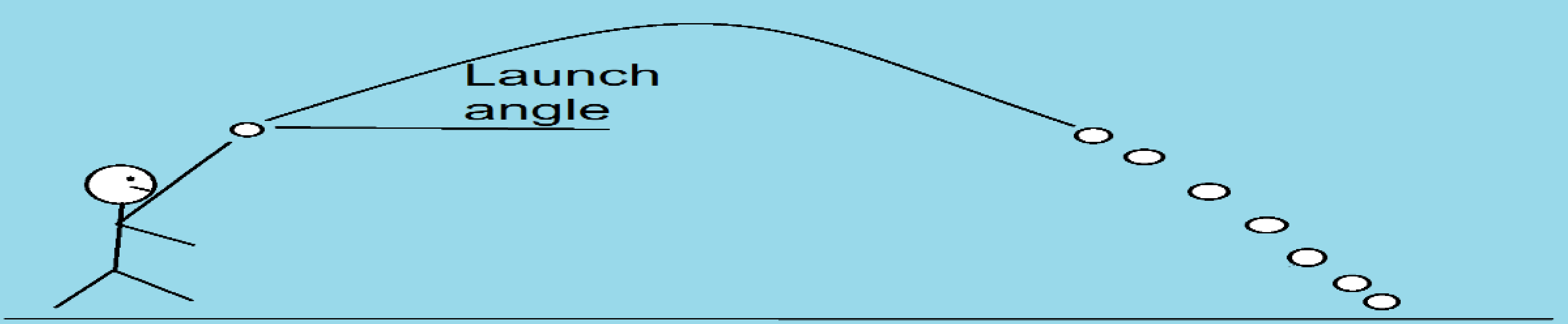
Assume that when the ball leaves your hand it is at a height of 2 meters (about 6 feet). Your task is to find the best angle to throw the ball, so that it will land the furthest from you.

ground due to gravity.

If you can throw a tennis ball 12 meters/second (about 26.8 mph) how far can you throw it; meaning how far away from you can you make it land?

Assume that when the ball leaves your hand it is at a height of 2 meters (about 6 feet). Your

SIMIODE



Method of solution:

1. For each launch angle between 0 and 90 degrees use Euler's method to numerically calculate where the ball will land. Store these values.
2. Pick the angle corresponding to the furthest away that the ball lands.
3. Re-run Euler for that optimal angle and plot the optimal trajectory.

Not elegant. But it works.

Differential Equations Home Page   Presentations   Programming and compilers   *Projects and Modeling Scenarios*

Euler, Drag, and How Far Can You Throw a Ball?

Associated Files

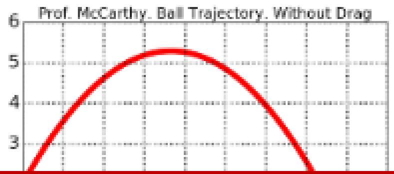
- detailed PDF on using Euler's method to calculate trajectories and drag (air resistance).
- Python script (*ThrownBallEulerStudent.py* – scroll down to see the script. You can copy and paste it into a Python compiler).

Description of scenario. If a tennis ball is thrown through the air it will eventually hit the ground due to gravity.

If you can throw a tennis ball 12 meters/second (about 26.8 mph) how far can you throw it; meaning how far away from you can you make it land?

Assume that when the ball leaves your hand it is at a height of 2 meters (about 6 feet). Your task is to find the best angle to throw the ball, so that it will land the furthest from you.

The Python script *ThrownBallEulerStudent.py* (shown below) solves this problem if we neglect drag (air resistance). The Python script outputs a graph of the ball's trajectory and the solution:



My Differential Equations Class' Website

Just **Google**

McCarthy Differential Equations

The modeling project is also on

SIMIODE

Output from ThrownBallEulerStudent.py

Assuming drag = 0. If when you throw a tennis ball you release it at a height of 2 meters and a speed of 12 meters/second, and you want it to land furthest from you, you should throw the ball at an angle of 42 degrees: it will land about 16.6 meters away.

The PDF **Using Euler to Simulate a Thrown Ball – Student Version (click to open PDF)** explains how to solve this problem using Euler's method, and gives suggestions on how to implement the algorithm in Python. The PDF also discusses how to include drag (air resistance).

Your project is to solve this problem if we include drag (air resistance). With the help of Euler's Method, write a short script (in Python) to find the ideal launch angle to throw the ball, so that it will result in the ball landing as far as possible from you. Also find that maximal distance and plot that trajectory. Make sure to include air resistance (drag) in your model. The height you launch the ball at is 2 meters and the speed is 12 meters/second. Don't worry about rotational effects.

What to do? Read the PDF. Run the Python code (below). Once it is working, change the Python code so that it takes into account drag. Also change the wording of the plot's title to be: *Your name. Ball trajectory. With drag.* Change the beginning part of the Python's script's output to say: *With drag.*

What to hand in? Hand in one sheet of paper. Copy the graph (with your name in the title) and the solution from Python's output and paste them into a single document (E.g. Word, OpenOffice, etc.). Print that out. Hand in the print out.

You can run the following python code (just copy and paste it):

My Differential Equations Class' Website

Just **Google**

McCarthy Differential Equations

The modeling project is also on **SIMIODE**

implement the algorithm in Python. The PDF also discusses how to include drag (air resistance).

Your project is to solve this problem if we include drag (air resistance). With the help of Euler's Method, write a short script (in Python) to find the ideal launch angle to throw the ball, so that it will result in the ball landing as far as possible from you. Also find that maximal distance and plot that trajectory. Make sure to include air resistance (drag) in your model. The height you launch the ball at is 2 meters and the speed is 12 meters/second. Don't worry about rotational effects.

What to do? Read the PDF. Run the Python code (below). Once it is working, change the Python code so that it takes into account drag. Also change the wording of the plot's title to be: *Your name. Ball trajectory. With drag.* Change the beginning part of the Python's script's output to say: *With drag.*

What to hand in? Hand in one sheet of paper. Copy the graph (with your name in the title) and the solution from Python's output and paste them into a single document (E.g. Word, OpenOffice, etc.). Print that out. Hand in the print out.

You can run the following python code (just copy and paste it):

- on your own computer. You can download a free version of Python 3. Anaconda Python IDE (free): https://www.anaconda.com/distribution/
- online (for free) at: https://repl.it/languages/python3

Python code: **ThrownBallEulerStudent.py**

```
# Written by Chris McCarthy July 2019 SIMIODE DEMARC
# Drag = 0 Student Version
#========================================  for online compi
import matplotlib as mpl
mpl.use('Agg')
#========================================  usual packages
import numpy as np
from matplotlib import pyplot as plt
plt.rcParams.update({'font.size': 22})
#========================================= constants
g = 9.8    # gravitation
m = 0.058  # mass tennis ball in kg
#=========================================
class Ball:
    def __init__(self, x,y,vx,vy,t):
        self.x = x
        self.y = y
```

My Differential Equations Class' Website

Just **Google**

McCarthy Differential Equations

The modeling project is also on **SIMIODE**

Python code: **ThrownBallEulerStudent.py**

```
# Written by Chris McCarthy July 2019 SIMIODE DEMARC
# Drag = 0 Student Version
#========================================  for online compi
import matplotlib as mpl
mpl.use('Agg')
#========================================  usual packages
import numpy as np
from matplotlib import pyplot as plt
plt.rcParams.update({'font.size': 22})
#========================================= constants
g = 9.8    # gravitation
m = 0.058  # mass tennis ball in kg
#=========================================
class Ball:
    def __init__(self, x,y,vx,vy,t):
        self.x = x
        self.y = y
        self.vx = vx
        self.vy = vy
        self.t = t
    def update_ball(self, delta_t,g):
        self.x = self.x + delta_t*self.vx
        self.y = self.y + delta_t*self.vy
        self.vx = self.vx
        self.vy = self.vy + delta_t*(-g )
        self.t = self.t + delta_t
#========================================= initial conditio
x0 = 0      # initial x position in meters
y0 = 2      # initial y position in meters
t0 = 0      # initial time in seconds
speed = 12  # initial speed of the ball in meters/sec
#========================================= Delta t
dt = .001 # Delta t
#========================================= Delta t
xDistance = []   # store the horizontal distance ball travelle
Theta = []       # store the angle the ball is thrown at
#========================================= run Euler for th
for theta in range(0, 91):
    vx0 = speed*np.cos(theta * np.pi/180) # initial vx
    vy0 = speed*np.sin(theta * np.pi/180) # initial vy
    ball = Ball(x0 , y0,  vx0, vy0, t0)   # initialize ball ob
    while 0 &amp;lt;= ball.y:                          # Euler Metho
        ball.update_ball(dt, g)
    xDistance.append(ball.x)             # collect x value whe
    Theta.append(theta)                  # collect theta
#========================================= find max x dista
maxpos = xDistance.index(max(xDistance))
```

My Differential Equations Class' Website

Just **Google**

McCarthy Differential Equations

The modeling project is also on **SIMIODE**

```
    vy0 = speed*np.sin(theta * np.pi/180) # initial vy
    ball = Ball(x0 , y0,  vx0, vy0, t0)   # initialize ball ob
    while 0 &amp;lt;= ball.y:                          # Euler Metho
        ball.update_ball(dt, g)
    xDistance.append(ball.x)                # collect x value whe
    Theta.append(theta)                     # collect theta
#============================================ find max x dista
maxpos = xDistance.index(max(xDistance))
#============================================= run Euler (agai
best_theta = Theta[maxpos]
best_vx0 = speed*np.cos(best_theta * np.pi/180) # initial vx
best_vy0 = speed*np.sin(best_theta * np.pi/180) # initial vy
best_ball = Ball(x0 , y0,  best_vx0, best_vy0, t0)   # initial
xvalues = [x0]
yvalues = [y0]
times =   [t0]
while 0 &amp;lt;= best_ball.y:                        # Euler Meth
    best_ball.update_ball(dt, g)
    xvalues.append(best_ball.x)
    yvalues.append(best_ball.y)
    times.append(best_ball.t)
#=============================================
print(' ')
print('Assuming drag = 0. If when you throw a tennis ball you
      ' meters and a speed of', speed,
      'meters/second, and you want it to land furthest from y
     'you should throw the ball at an angle of', Theta[maxpos
      ' degrees: it will land about', np.round(max(xDistance),
#============================================= plot best traje
plt.plot(xvalues, yvalues, 'r-',linewidth=7.0)
plt.grid(linewidth='3', color='black')
plt.title('Prof. McCarthy. Ball Trajectory. Without Drag', fon
plt.savefig('NoDragGraph.png')
```

This entry is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International license.

My Differential Equations Class' Website

Just **Google**

McCarthy Differential Equations

The modeling project is also on **SIMIODE**

Second order ODE

Reduce to first order so we can apply Euler

First order ODE

Newton's 2^{nd} Law of Motion

$$\mathbf{F_g} = m\mathbf{a}$$
$$\langle 0, -mg \rangle = m\langle \ddot{x}, \ddot{y} \rangle$$

Students are shown solution assuming no drag force. Only gravity.

Separate the x and y components then reduce the order.

$$m\ddot{x} = 0$$
$$m\ddot{y} = -mg$$

$$\dot{x} = v_x$$
$$\dot{y} = v_y$$
$$\dot{v}_x = 0$$
$$\dot{v}_y = -g$$

Let $\dot{x} = v_x$, so $\ddot{x} = \dot{v}_x$

Let $\dot{y} = v_y$, so $\ddot{y} = \dot{v}_y$

Divide by m

Euler's Method in Vector Form

$(x, y) =$ position
$(v_x, v_y) =$ velocity
$t =$ time

$$\begin{pmatrix} x \\ y \\ v_x \\ v_y \\ t \end{pmatrix}_{\text{at } t+\Delta t} \approx \begin{pmatrix} x \\ y \\ v_x \\ v_y \\ t \end{pmatrix}_{\text{at } t} + \frac{d}{dt}\begin{pmatrix} x \\ y \\ v_x \\ v_y \\ t \end{pmatrix}_{\text{at } t} \Delta t$$

Assuming
no drag force.
Only gravity.

$$\begin{pmatrix} x \\ y \\ v_x \\ v_y \\ t \end{pmatrix}_{\text{at } t+\Delta t} \approx \begin{pmatrix} x \\ y \\ v_x \\ v_y \\ t \end{pmatrix}_{\text{at } t} + \begin{pmatrix} v_x \\ v_y \\ 0 \\ -g \\ 1 \end{pmatrix}_{\text{at } t} \Delta t$$

velocity

acceleration

Python code: **ThrownBallEulerStudent.py**

```
# Written by Chris McCarthy July 2019 SIMIODE DEMARC
# Drag = 0 Student Version
#===========================================  for online compi
import matplotlib as mpl
mpl.use('Agg')
#===========================================  usual packages
import numpy as np
from matplotlib import pyplot as plt
plt.rcParams.update({'font.size': 22})
#============================================ constants
g = 9.8    # gravitation
m = 0.058  # mass tennis ball in kg
#============================================
class Ball:
    def __init__(self, x,y,vx,vy,t):
        self.x = x
        self.y = y
        self.vx = vx
        self.vy = vy
        self.t = t
    def update_ball(self, delta_t,g):
        self.x = self.x + delta_t*self.vx
        self.y = self.y + delta_t*self.vy
        self.vx = self.vx
        self.vy = self.vy + delta_t*(-g )
        self.t = self.t + delta_t
```

Euler Method implemented in Python

repl.it

Most students paste the Python code into an online Python IDE such as repl.it

Online Python compiler, On...
Code Python, compile Python, run Python, ...

save | run ▶ | share | + new repl | talk | Sign up

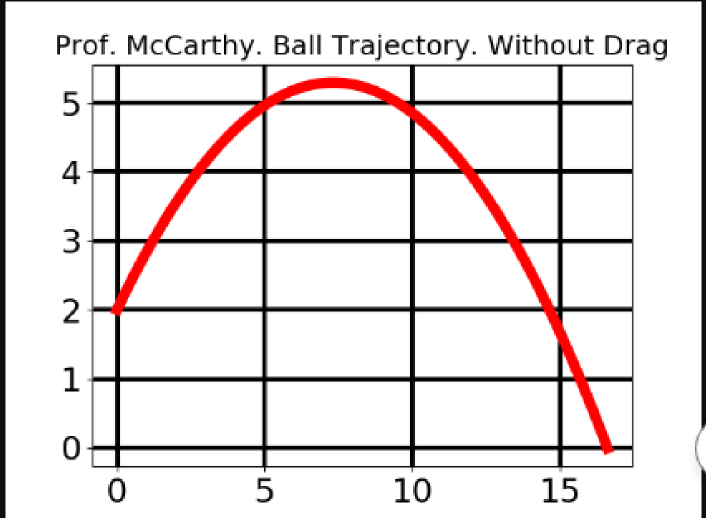
Files

main.py · saved

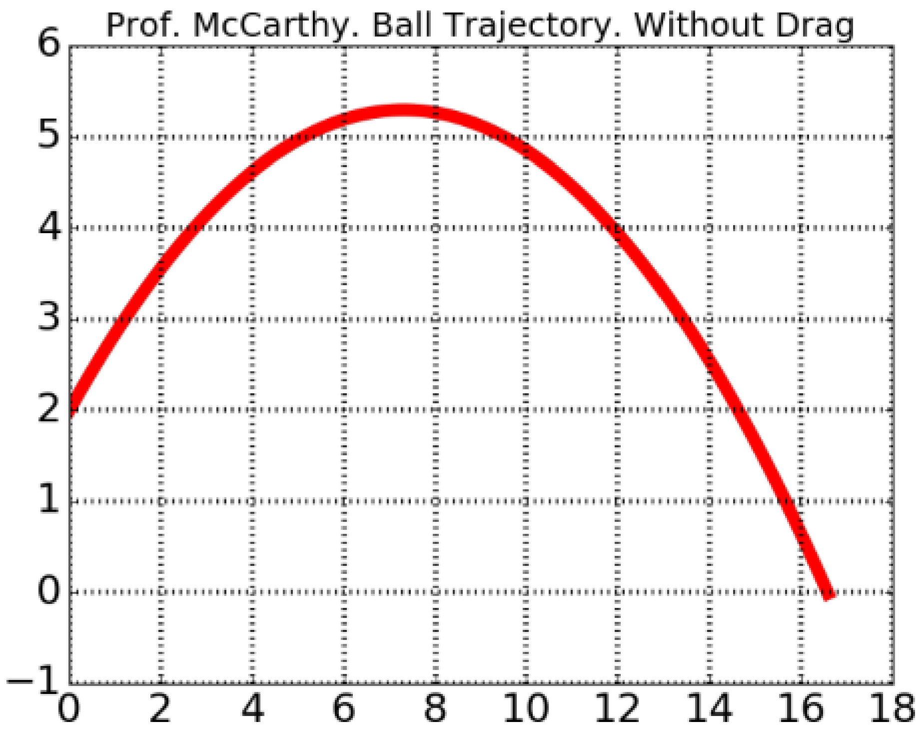
```
distance over theta, print it
maxpos = xDistance.index(max(xDistance))
#========================================== run Euler (again) for best theta
best_theta = Theta[maxpos]
best_vx0 = speed*np.cos(best_theta * np.pi/180) # initial vx
best_vy0 = speed*np.sin(best_theta * np.pi/180) # initial vy
best_ball = Ball(x0 , y0,  best_vx0, best_vy0, t0)   # initialize ball object
xvalues = [x0]
yvalues = [y0]
times =   [t0]
while 0 <= best_ball.y:                     # Euler Method applied to that ball
    best_ball.update_ball(dt, g)
    xvalues.append(best_ball.x)
    yvalues.append(best_ball.y)
    times.append(best_ball.t)
#==========================================
print(' ')
print('Assuming drag = 0. If when you throw a tennis ball you release it at a height of', y0,
      ' meters and a speed of', speed,
      'meters/second, and you want it to land furthest from you,',
      'you should throw the ball at an angle of', Theta[maxpos],
      ' degrees: it will land about', np.round(max(xDistance),1),'meters away.' )
#========================================== plot best trajectory
plt.plot(xvalues, yvalues, 'r-',linewidth=7.0)
plt.grid(linewidth='3', color='black')
plt.title('Prof. McCarthy. Ball Trajectory. Without Drag', fontsize = 18)
plt.savefig('NoDragGraph.png')
```

```
Assuming drag = 0. If when you throw a tennis ball you release it at a height of 2  meters and a speed of 12 meters/second, and you want it to land furthest from you, you should throw the ball at an angle of 42  degrees: it will land about 16.6 meters away
.
```

Some files may fail to load on the browser. Click here to download



The no drag solution

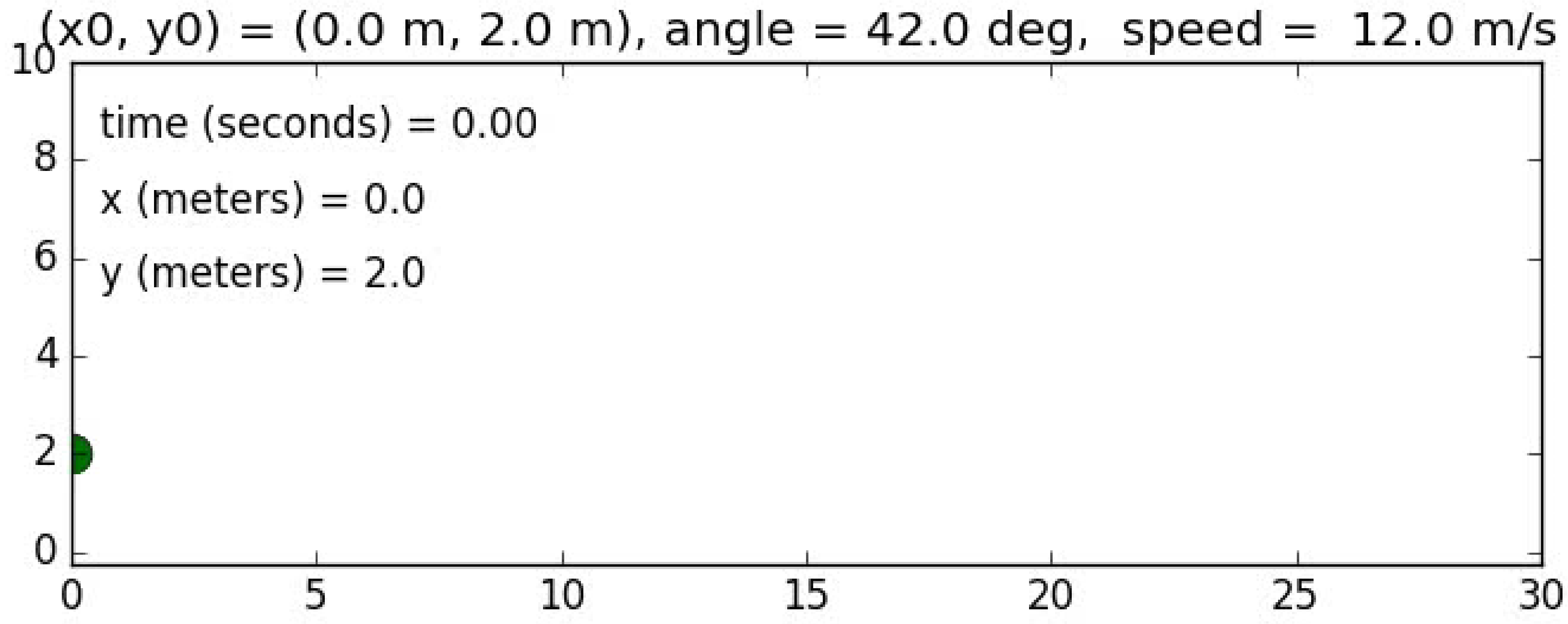


Output from ThrownBallEulerStudent.py

Assuming drag = 0. If when you throw a tennis ball you release it at a height of 2 meters and a speed of 12 meters/second, and you want it to land furthest from you, you should throw the ball at an angle of 42 degrees: it will land about 16.6 meters away.

A desktop version of Python gives a faster, better development experience. E.g. Anaconda Python (free).

No Drag Trajectory for Maximum Distance



Heuristic argument: drag force proportional to v^2

$$F = ma = \frac{d}{dt}mv \approx \frac{\Delta mv}{\Delta t}$$

$\rho =$ density of air. $\quad v =$ velocity of projectile

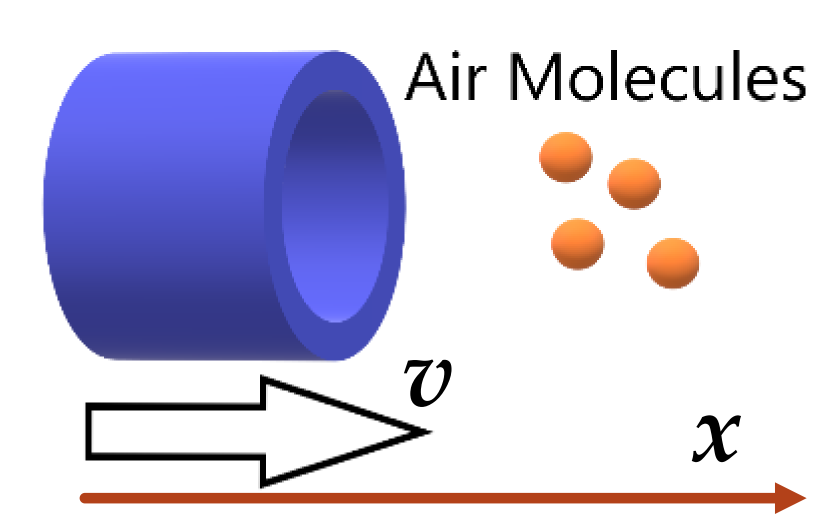
$A =$ cross sectional area. $\quad \Delta x = v\Delta t$

Mass of air collided with in $\Delta t = \rho \underbrace{A\Delta x}_{volume}$

$$\boxed{\Delta v_{air} \propto v_{projectile} = v}$$

$$F_{drag} \propto \frac{\overbrace{\rho A \Delta x}^{mass\ air} \quad \Delta v_{air}}{\Delta t} \approx \rho A v\, v = \rho A v^2$$

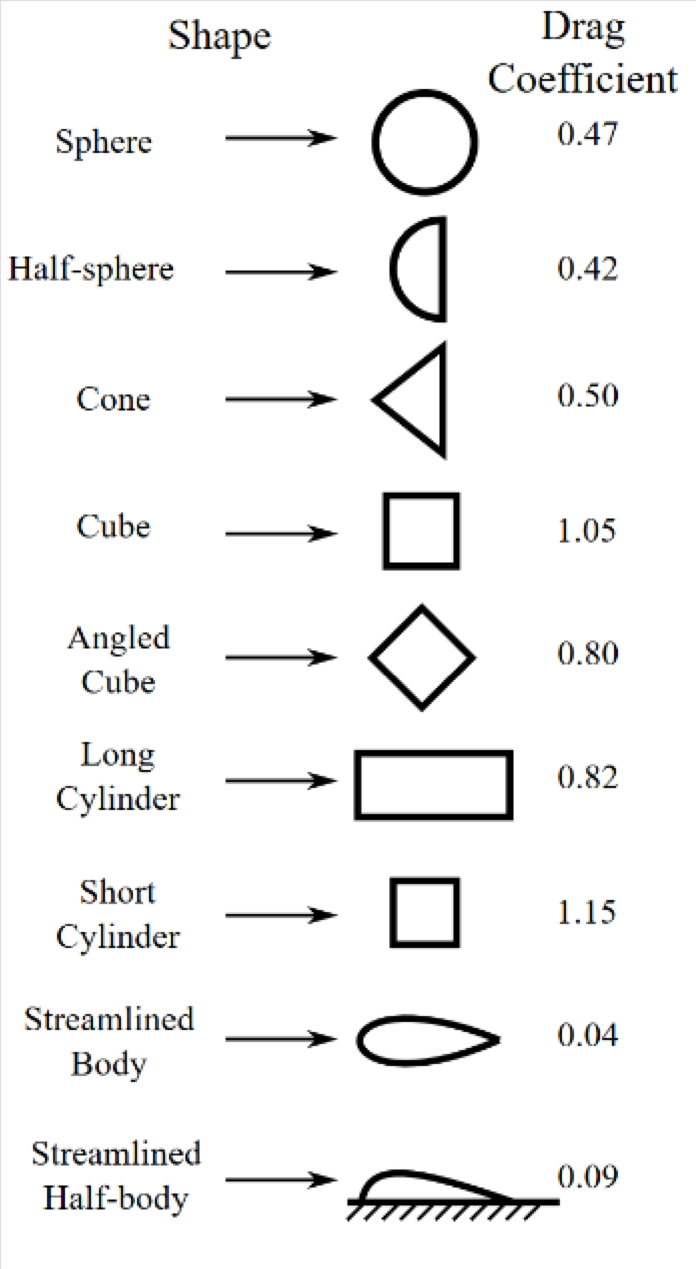
F_{drag} is in opposite direction of v.



Drag Equation

$F_{drag} = \frac{1}{2} C_D\ \rho A v^2$

$C_D =$ drag coefficient



Measured Drag Coefficients

Drag Equation

$$F_{drag} = \frac{1}{2} C_D \, \rho A v^2$$

C_D = drag coefficient

Including drag force in tennis ball model

Magnitude of drag force:

$$F_D = \frac{C_D \; \rho \; A \; ||\mathbf{v}||^2}{2} = c\,||\mathbf{v}||^2$$

$C_D =$ coefficient of drag $= 0.5$

$\rho =$ density of air $= 1.21 \text{ kg/m}^3$

$A =$ cross sectional area $= \pi r^2$

$r = 0.0335$ m

$\mathbf{v} =$ velocity of ball in m/s

$m = 0.058$ kg and g $= 9.8 \text{ m/s}^2$

$$\boxed{\begin{gathered} \mathbf{F_g} + \mathbf{F_D} = m\mathbf{a} \\ \langle 0, -mg \rangle - c\,||\mathbf{v}||\;\langle v_x, v_y \rangle = m\langle \ddot{x}, \ddot{y} \rangle \end{gathered}}$$

Drag Force Vector

$$\begin{aligned} \mathbf{F_D} &= -F_D \frac{\mathbf{v}}{||\mathbf{v}||} \\ &= -\frac{C_D \; \rho \; A \; ||\mathbf{v}||^2}{2} \; \frac{\mathbf{v}}{||\mathbf{v}||} \\ &= -\frac{C_D \; \rho \; \pi r^2 \; ||\mathbf{v}||}{2} \; \mathbf{v} \\ &= -c\,||\mathbf{v}|| \; \mathbf{v} \end{aligned}$$

$$c = \frac{C_D \; \rho \; \pi r^2}{2}$$

Euler recursive relation including drag

$$\begin{pmatrix} x \\ y \\ v_x \\ v_y \\ t \end{pmatrix}_0 = \begin{pmatrix} 0 \\ 2 \\ 12\cos\theta \\ 12\sin\theta \\ 0 \end{pmatrix}$$

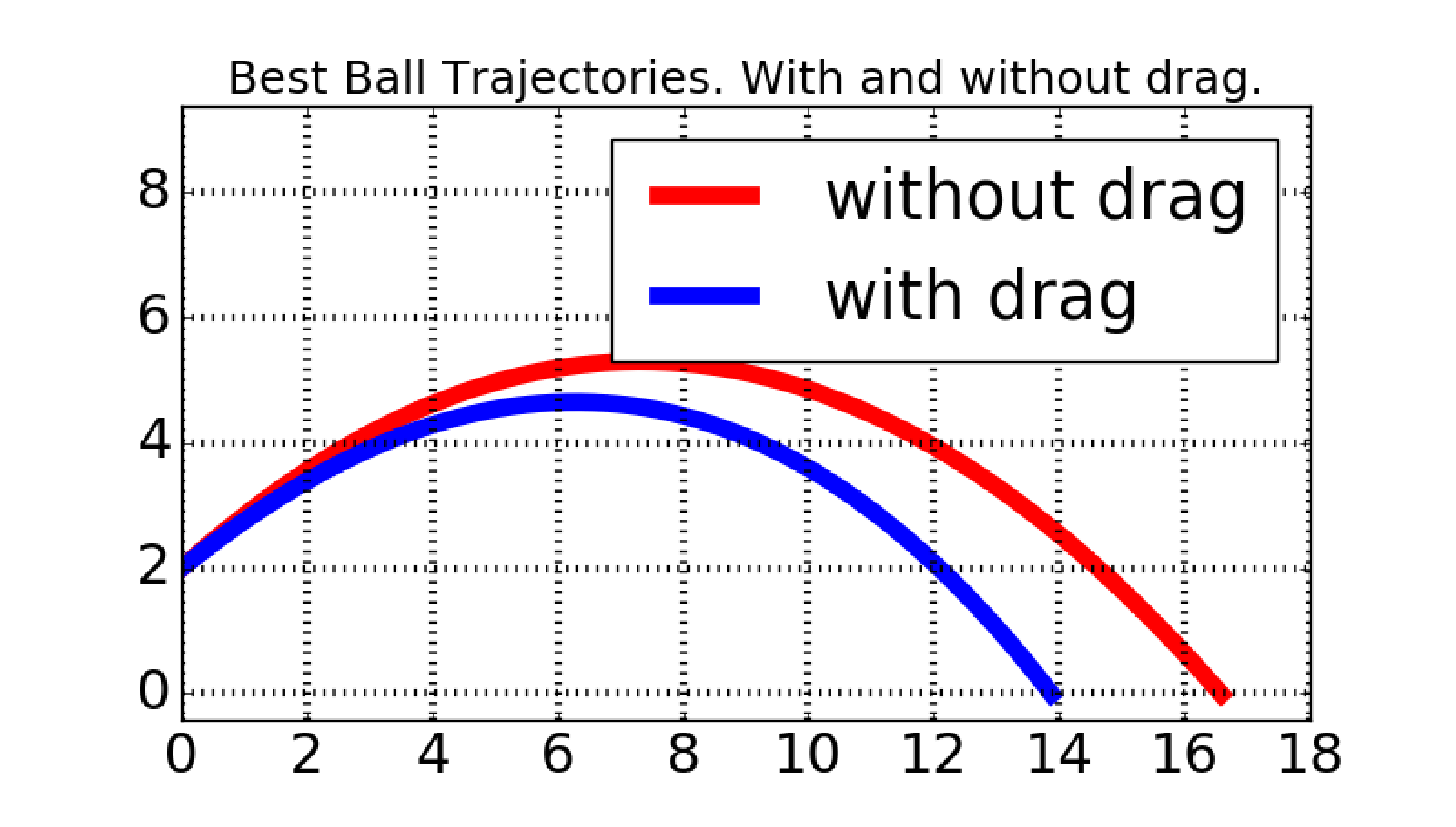
Initial conditions

Position 2 meters up

Speed 12 m/s

Launch angle θ varies

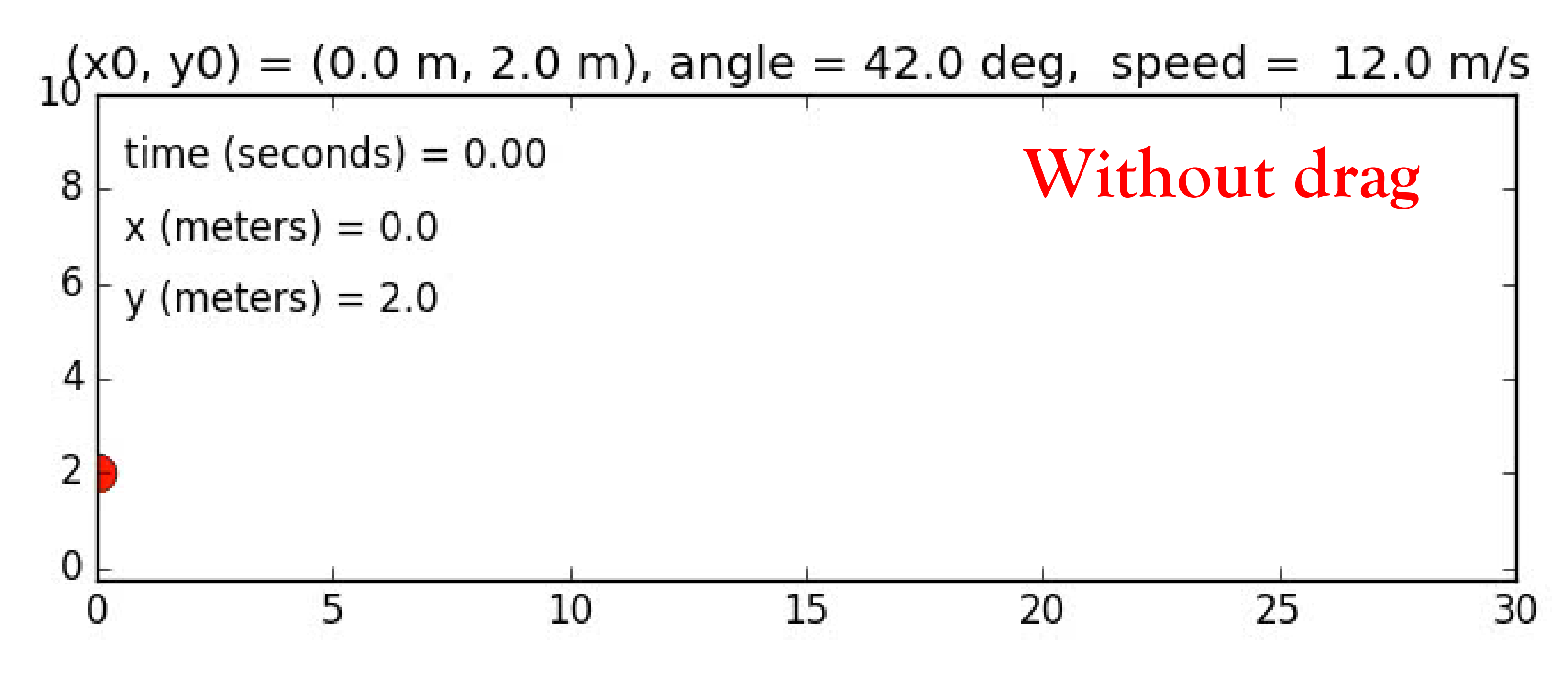
$$\begin{pmatrix} x \\ y \\ v_x \\ v_y \\ t \end{pmatrix}_{n+1} = \begin{pmatrix} x \\ y \\ v_x \\ v_y \\ t \end{pmatrix}_n + \begin{pmatrix} v_x \\ v_y \\ -\frac{c}{m}\sqrt{v_x^2 + v_y^2}\; v_x \\ -g - \frac{c}{m}\sqrt{v_x^2 + v_y^2}\; v_y \\ 1 \end{pmatrix}_n \cdot \Delta t$$

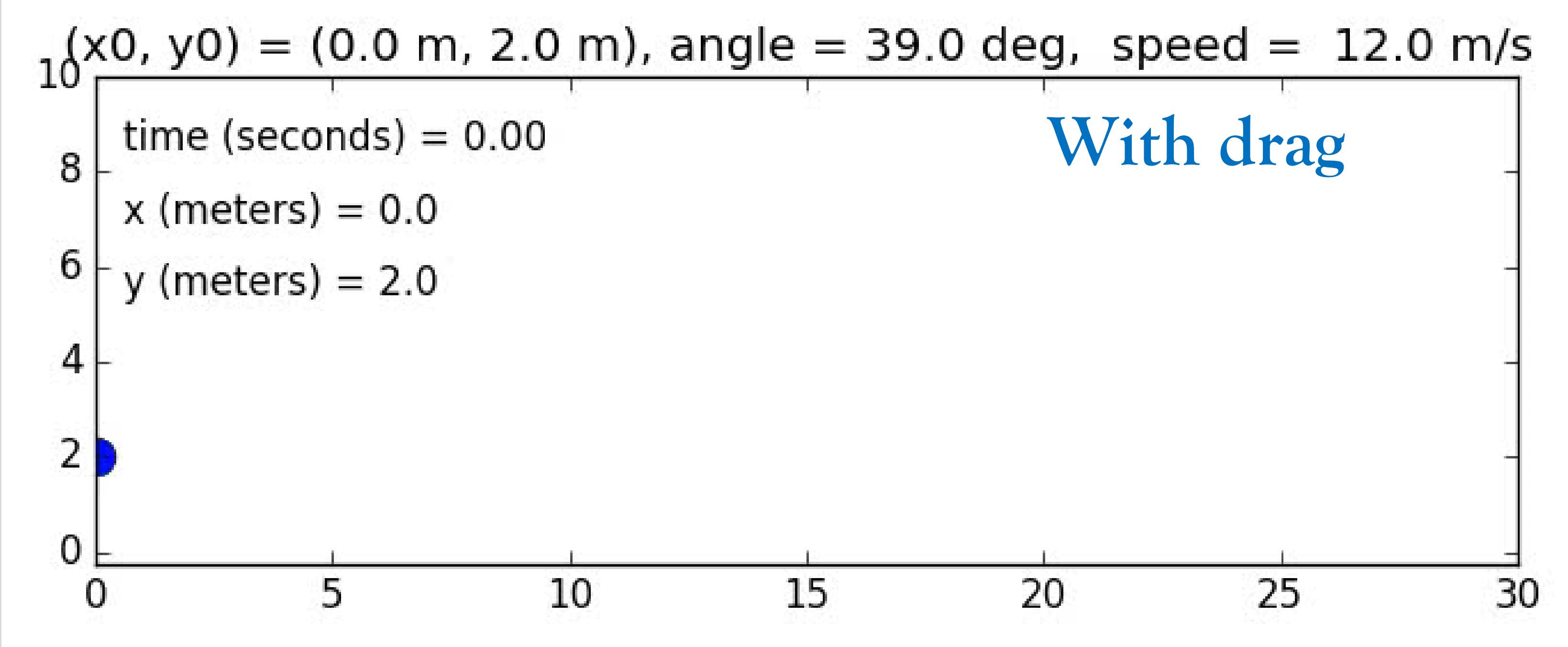


If when you throw a tennis ball you release it at a height of 2 meters and a speed of 12 meters/second, and you want it to land furthest from you:

Without drag. you should throw the ball at an angle of 42 degrees: it will land about 16.6 meters away.

With drag. you should throw the ball at an angle of 39 degrees: it will land about 13.9 meters away.



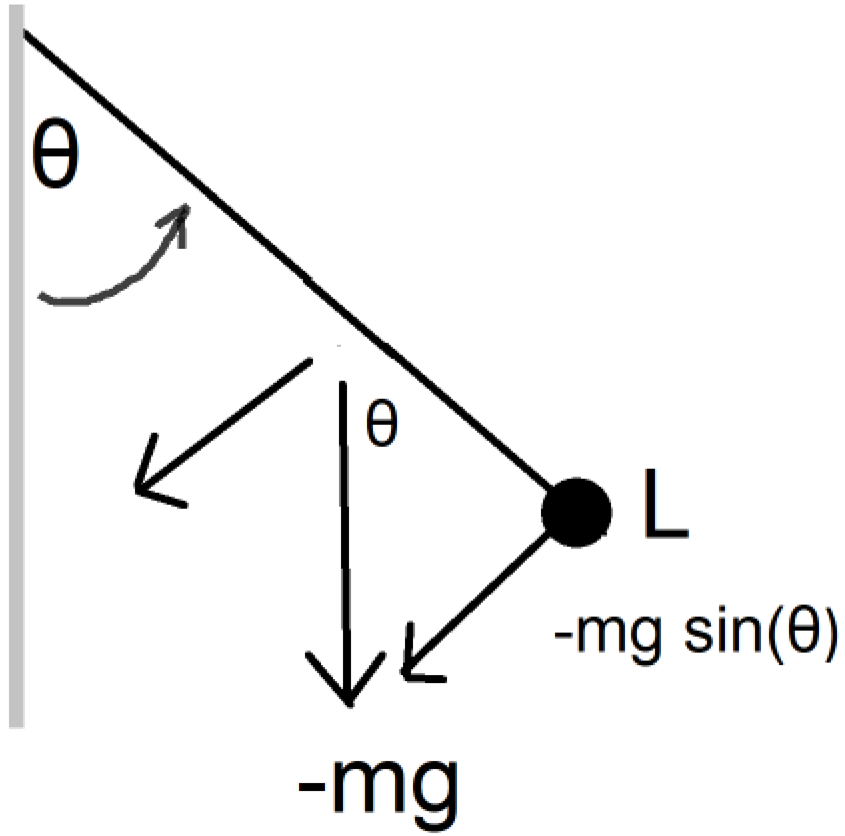


If when you throw a tennis ball you release it at a height of 2 meters and a speed of 12 meters/second, and you want it to land furthest from you:

Without drag. you should throw the ball at an angle of 42 degrees: it will land about 16.6 meters away.

With drag. you should throw the ball at an angle of 39 degrees: it will land about 13.9 meters away.

My honors student Kujtim Bardhyll worked with me to test the drag model on a real pendulum.



$$\mathbf{F_g} + \mathbf{F_D} = m\mathbf{a}$$

$$\langle 0, -mg \rangle - c\,||\mathbf{v}||\;\langle v_x, v_y \rangle = m\langle \ddot{x}, \ddot{y} \rangle$$

with $c = \dfrac{C_D\;\rho\;\pi r^2}{2}$

$\Delta\theta$ results in the bob moving $L\Delta\theta$.

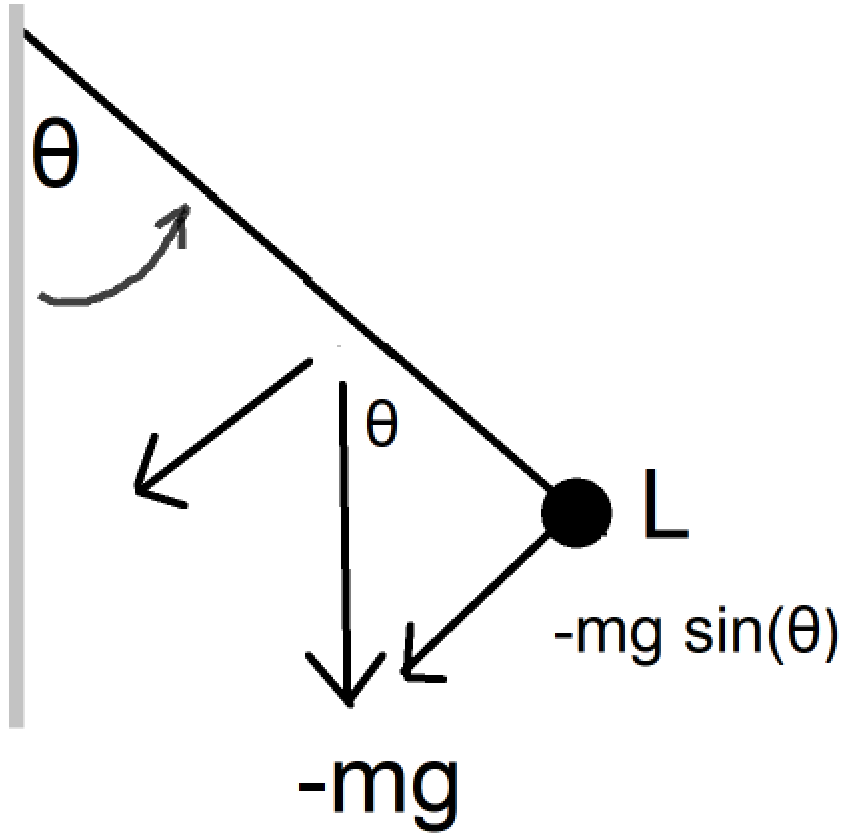
$v = L\dot{\theta}$

Tangential drag force: $-\mathrm{sign}(\dot{\theta})cL^2\dot{\theta}^2$

Tangential gravitational force: $-mg\sin(\theta)$

$$-mg\sin(\theta) - \mathrm{sign}(\dot{\theta})cL^2\dot{\theta}^2 = mL\ddot{\theta}$$

$$\ddot{\theta} + \frac{c}{m}\mathrm{sign}(\dot{\theta})L\dot{\theta}^2 + \frac{g}{L}\sin(\theta) = 0$$



Euler recursion step for the pendulum with drag.

$$\begin{pmatrix}\theta\\ \dot\theta\\ t\end{pmatrix}_{n+1} = \begin{pmatrix}\theta\\ \dot\theta\\ t\end{pmatrix}_{n} + \left(\begin{matrix} v_x \\ -\frac{c}{m}\mathrm{sign}(\dot\theta)L\dot\theta^2 - \frac{g}{L}\sin(\theta) \\ 1\end{matrix}\right)_n \cdot \Delta t$$

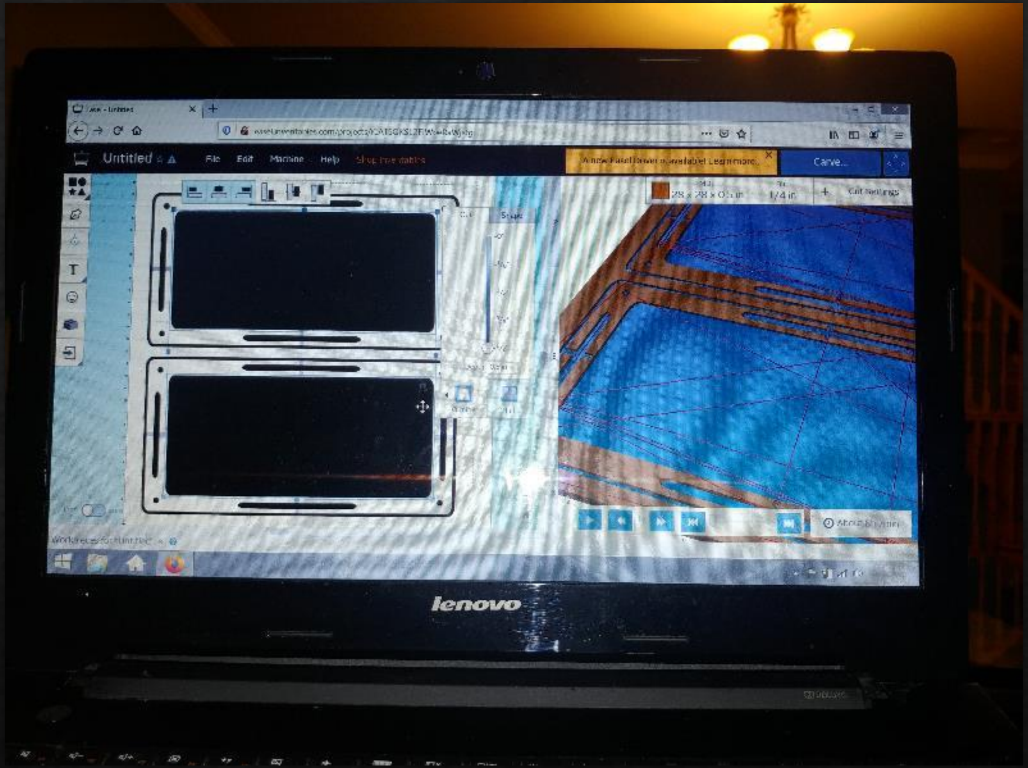
Pendulum & Drag

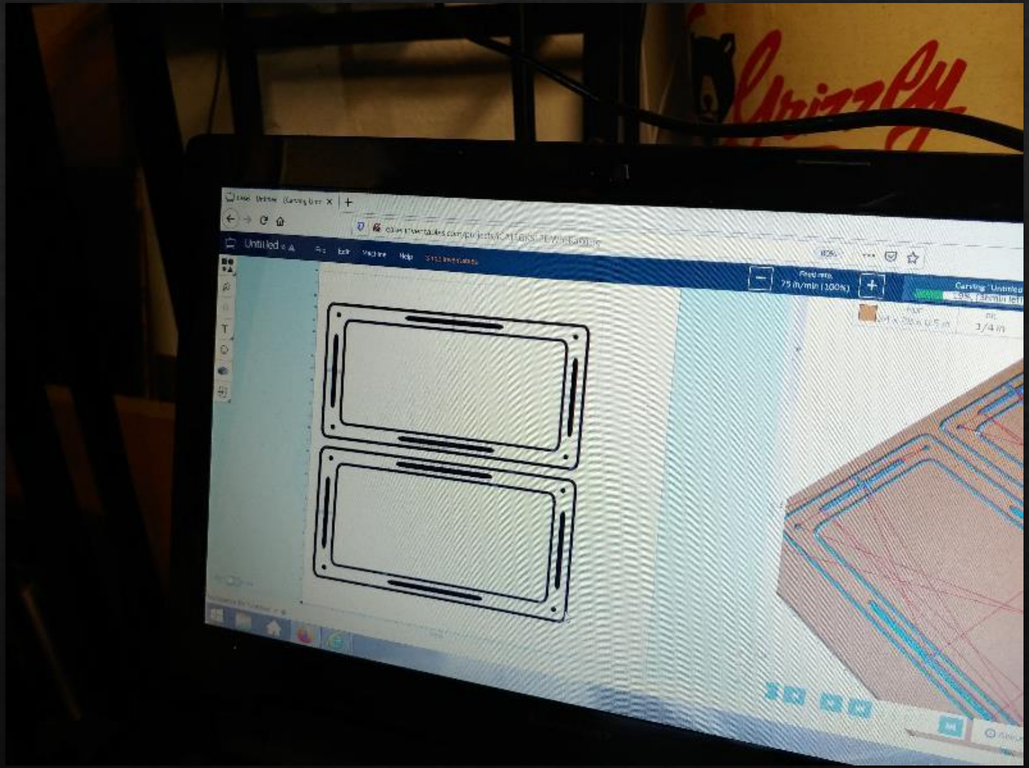
Presented by :

Kujtim Bardhyll

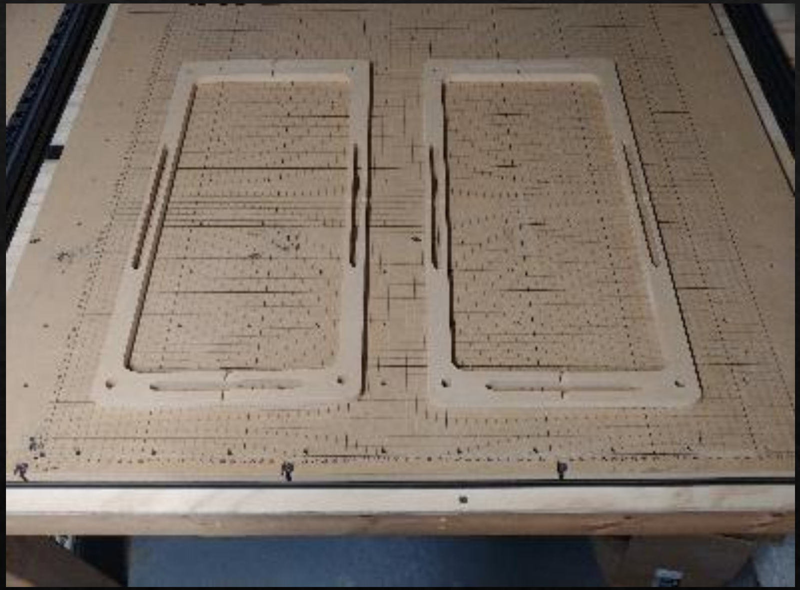
- I began the process by drawing a few sketches to see what might work.
- When I was done, I used a program on easel an application created by Inventables.
- I designed it and used my CNC (**Computer Numerical Control**) **machine** to make cuts.
- I chose to use MDF (**Micro Density Fiberboard)** because of its light weight and sturdy composition.
- I decide to use sliding arms to adjust the width of the pendulum allowing easy leveling of the pendulum. It also helped with the pivot because its smooth surface which does not cause much friction on the fishing line.
- After I had built the pendulum and tested it, I created a graph to help me easily track the bob.
- I then recorded the experiment using different parameters values (different weights for the bob and different lengths of string). I used the stopwatch from vclock.com to help time the experiment.
- I used the Tracker App to plot data points.
- With that data I used Python to create a graph that uses the real time data and the mathematical data to show how drag affects a pendulums motion.

Designing the Pendulum

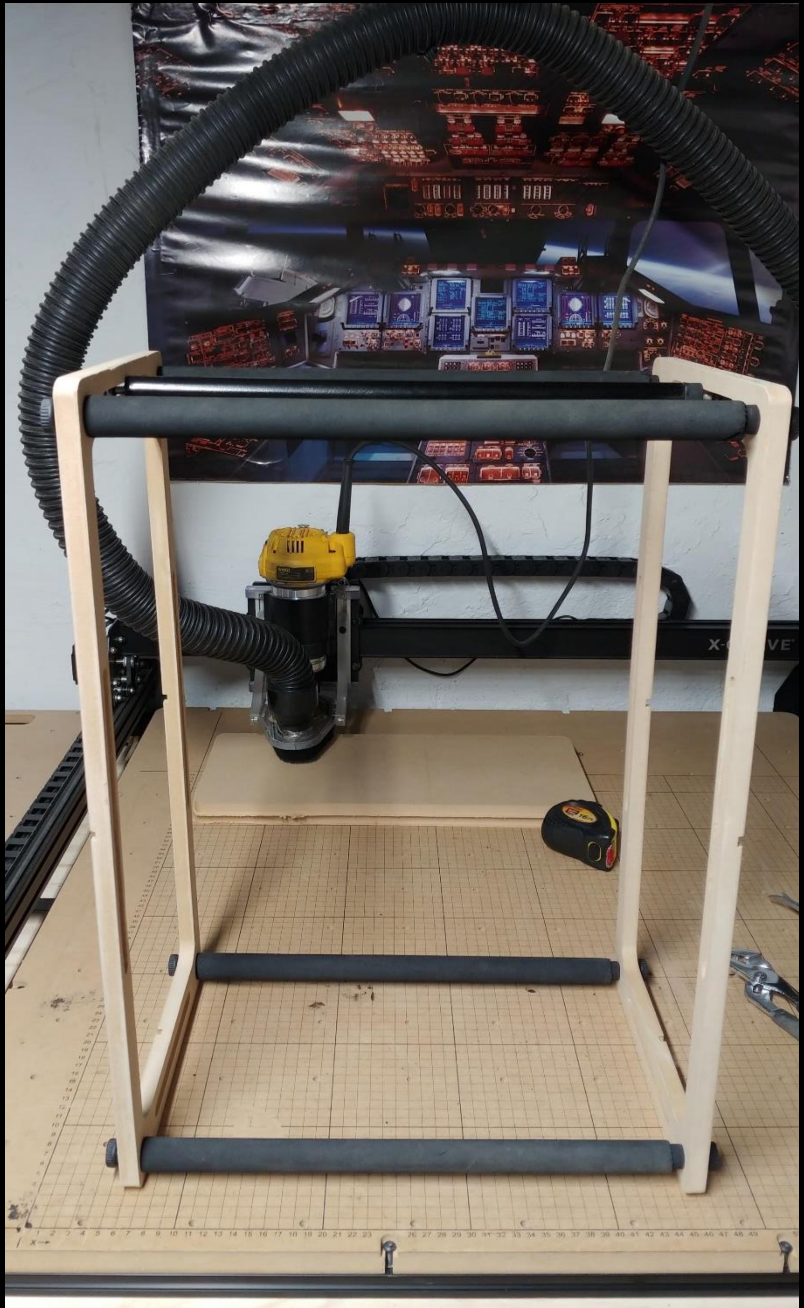




Milling MDF

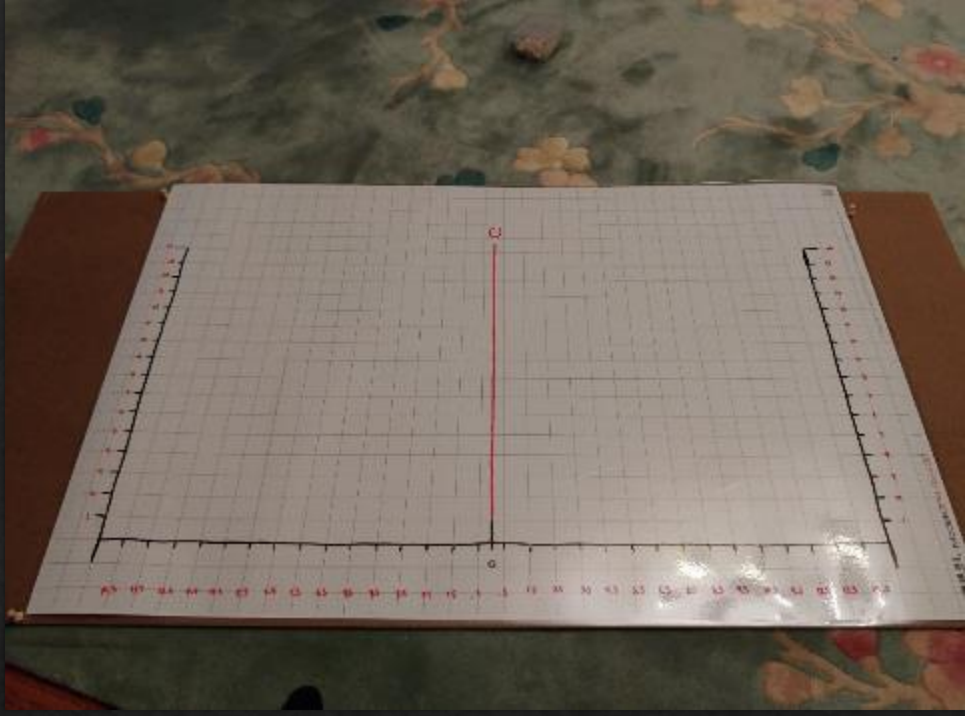



inventables
X-CARVE

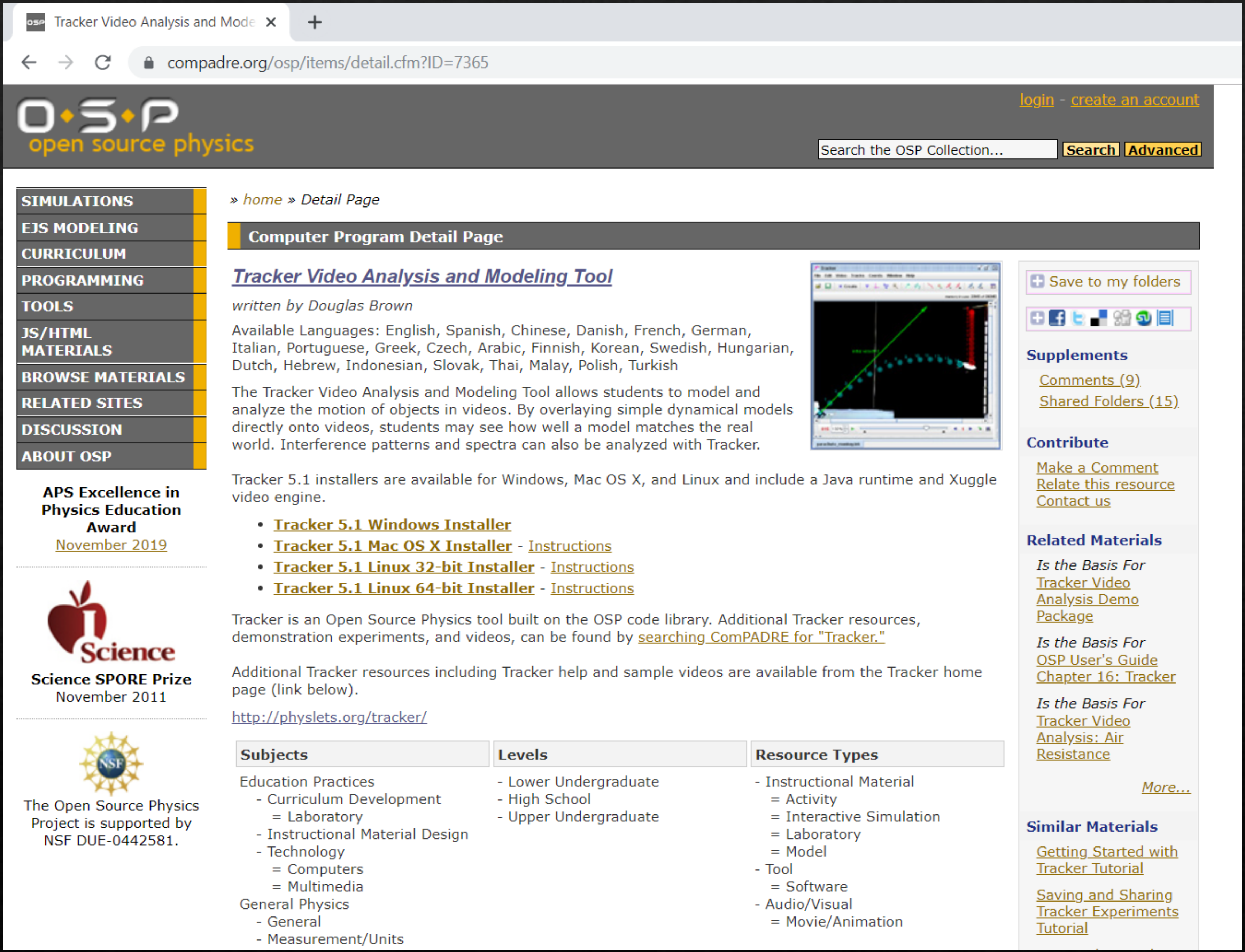
X-CARVE

X-CARVE

- Then I used a laminated graph 1”x 1” to help track the position of the bob as it moved from side to side.

- I also used fishing weights to increase the weight of the bob.



- I used **Tracker Video Analysis and Modeling Tool** from Open Source Physics to plot the points of the tennis ball.
- This app tracks objects in motion. It helped me see the oscillation points of the pendulum.
- These points are helpful because they use real time tracked data points against the calculations made in python.
- It creates a graph of the points showing the user where they are on the x and y axis.



Tracker Video Analysis and Mode

compadre.org/osp/items/detail.cfm?ID=7365

O·S·P open source physics

login - create an account

Search the OSP Collection... Search Advanced

SIMULATIONS
EJS MODELING
CURRICULUM
PROGRAMMING
TOOLS
JS/HTML MATERIALS
BROWSE MATERIALS
RELATED SITES
DISCUSSION
ABOUT OSP

APS Excellence in Physics Education Award
November 2019

Science
Science SPORE Prize
November 2011

NSF
The Open Source Physics Project is supported by NSF DUE-0442581.

» home » Detail Page

Computer Program Detail Page

Tracker Video Analysis and Modeling Tool

written by Douglas Brown

Available Languages: English, Spanish, Chinese, Danish, French, German, Italian, Portuguese, Greek, Czech, Arabic, Finnish, Korean, Swedish, Hungarian, Dutch, Hebrew, Indonesian, Slovak, Thai, Malay, Polish, Turkish

The Tracker Video Analysis and Modeling Tool allows students to model and analyze the motion of objects in videos. By overlaying simple dynamical models directly onto videos, students may see how well a model matches the real world. Interference patterns and spectra can also be analyzed with Tracker.

Tracker 5.1 installers are available for Windows, Mac OS X, and Linux and include a Java runtime and Xuggle video engine.

- **Tracker 5.1 Windows Installer**
- **Tracker 5.1 Mac OS X Installer** - Instructions
- **Tracker 5.1 Linux 32-bit Installer** - Instructions
- **Tracker 5.1 Linux 64-bit Installer** - Instructions

Tracker is an Open Source Physics tool built on the OSP code library. Additional Tracker resources, demonstration experiments, and videos, can be found by searching ComPADRE for "Tracker."

Additional Tracker resources including Tracker help and sample videos are available from the Tracker home page (link below).

http://physlets.org/tracker/

Subjects	Levels	Resource Types
Education Practices - Curriculum Development = Laboratory - Instructional Material Design - Technology = Computers = Multimedia General Physics - General - Measurement/Units	- Lower Undergraduate - High School - Upper Undergraduate	- Instructional Material = Activity = Interactive Simulation = Laboratory = Model - Tool = Software - Audio/Visual = Movie/Animation

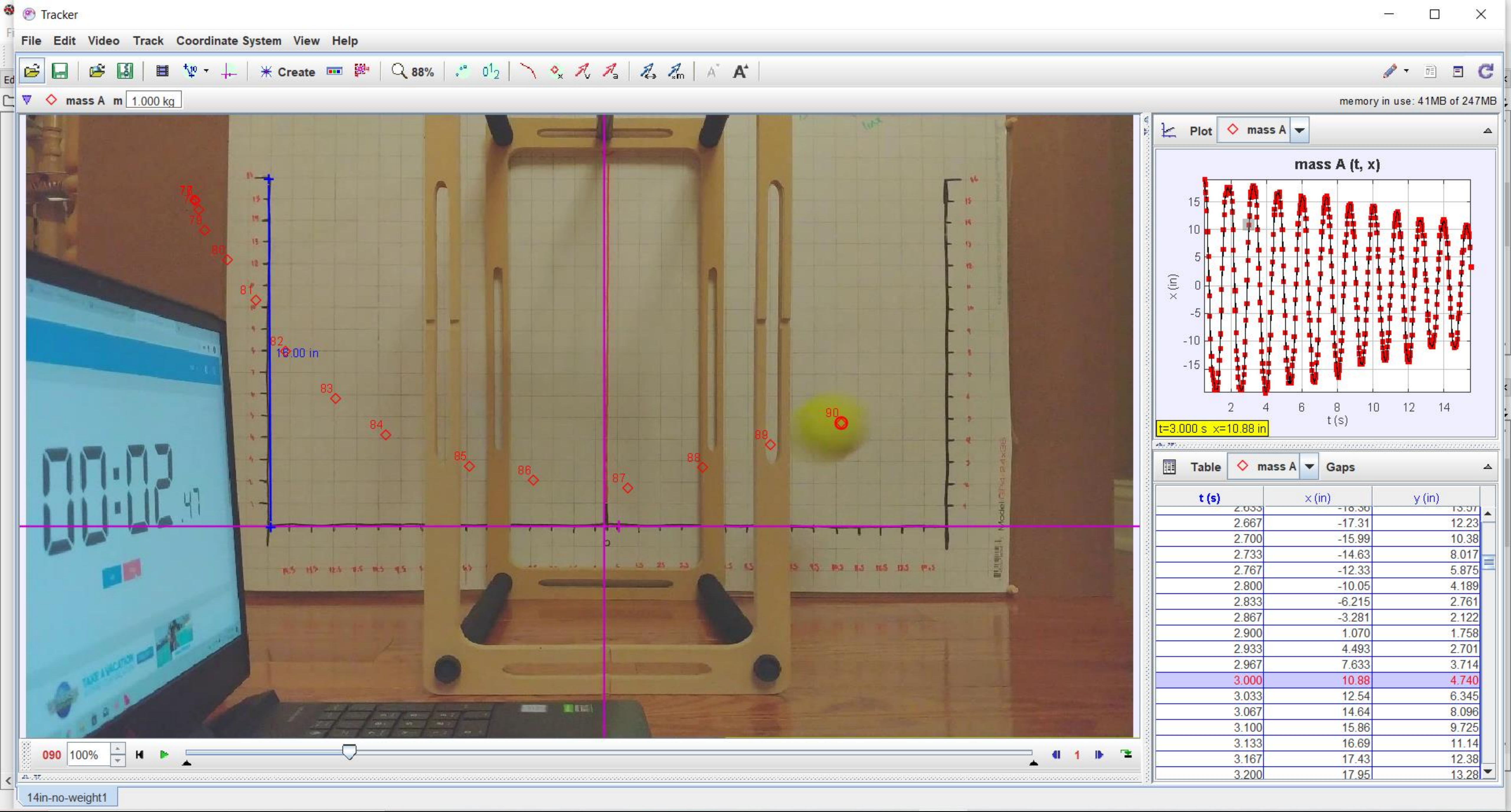
Save to my folders

Supplements
Comments (9)
Shared Folders (15)

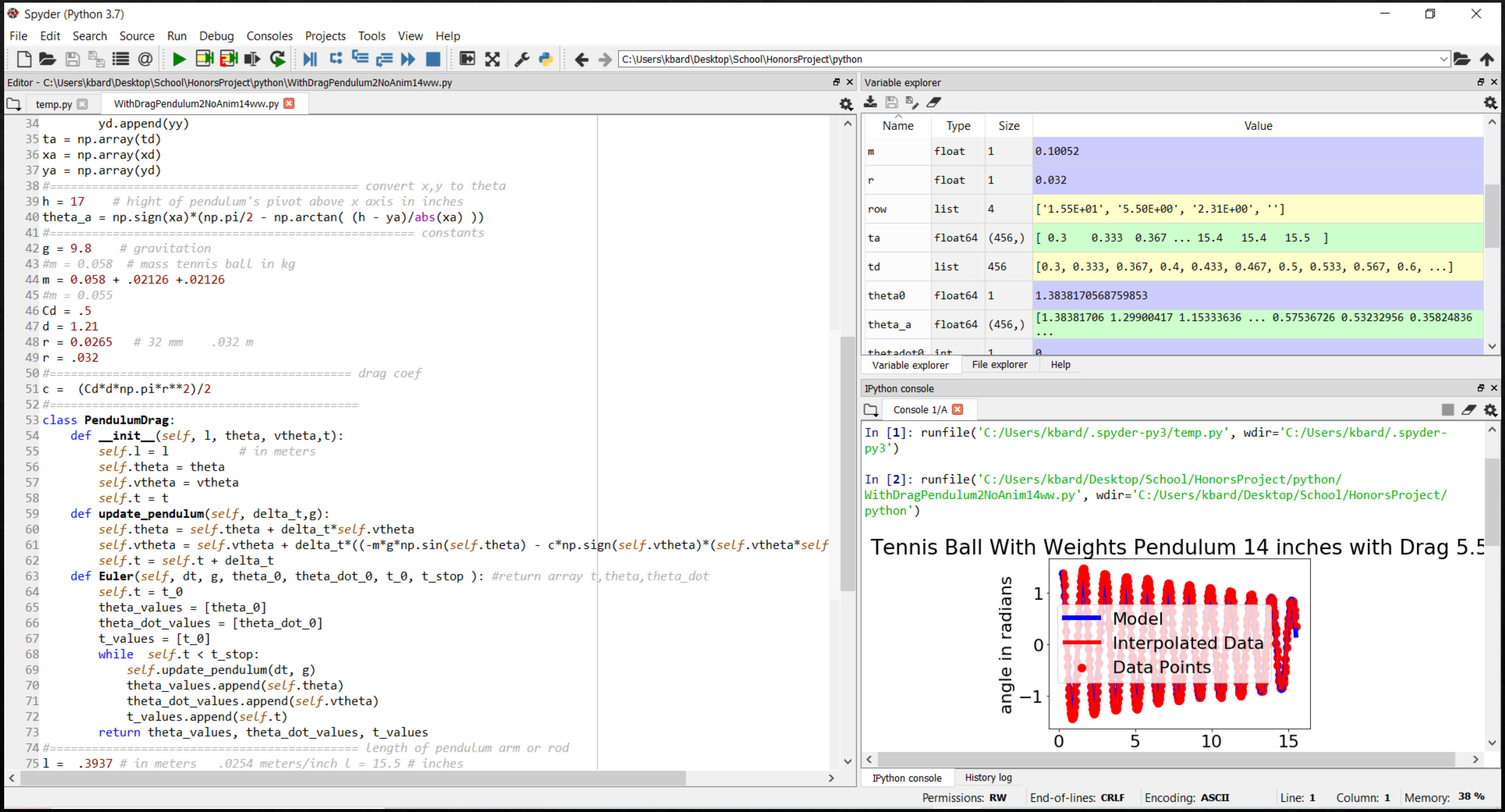
Contribute
Make a Comment
Relate this resource
Contact us

Related Materials
Is the Basis For Tracker Video Analysis Demo Package
Is the Basis For OSP User's Guide Chapter 16: Tracker
Is the Basis For Tracker Video Analysis: Air Resistance
More...

Similar Materials
Getting Started with Tracker Tutorial
Saving and Sharing Tracker Experiments Tutorial

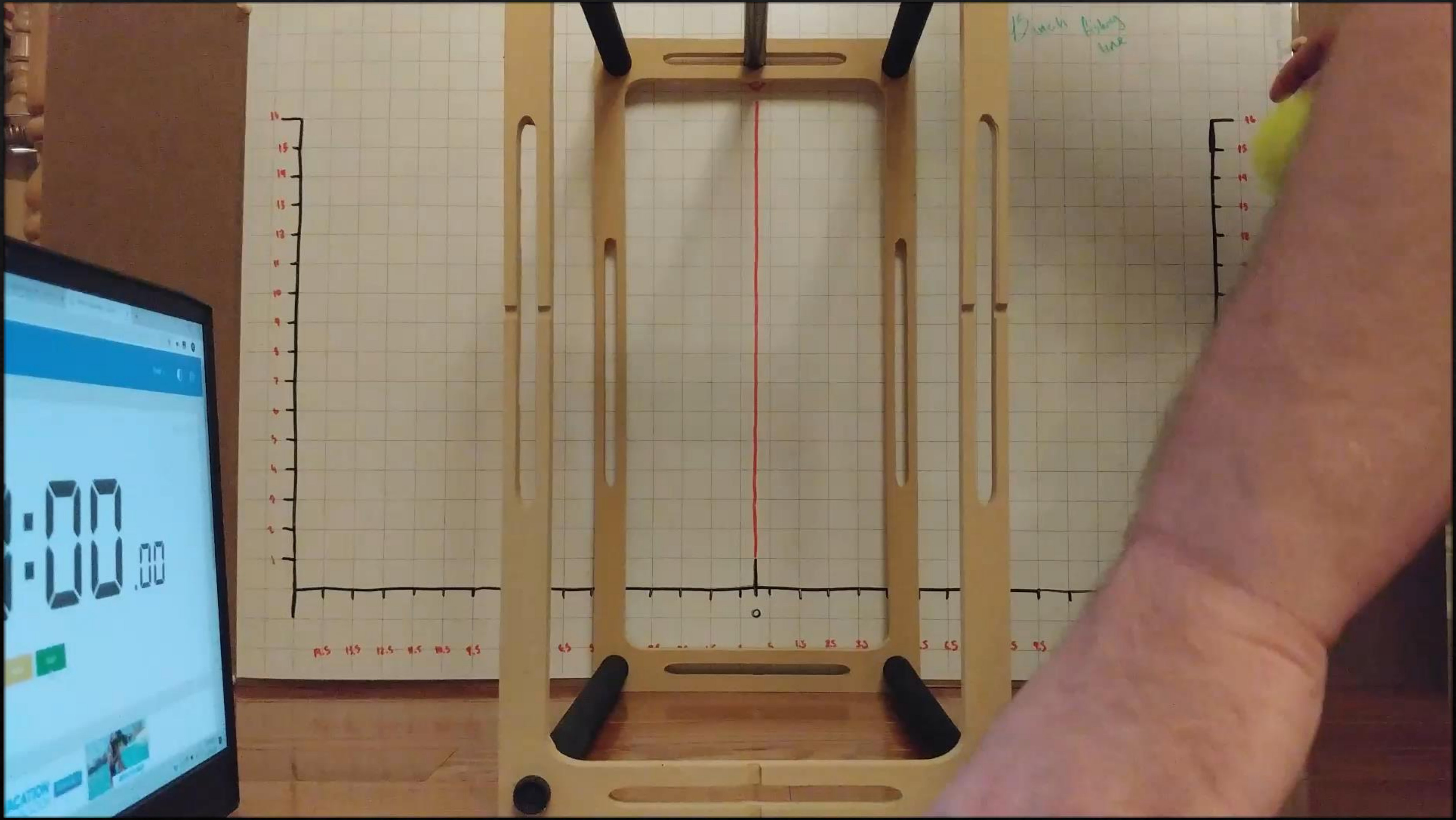


t (s)	x (in)	y (in)
2.667	-17.31	12.23
2.700	-15.99	10.38
2.733	-14.63	8.017
2.767	-12.33	5.875
2.800	-10.05	4.189
2.833	-6.215	2.761
2.867	-3.281	2.122
2.900	1.070	1.758
2.933	4.493	2.701
2.967	7.633	3.714
3.000	10.88	4.740
3.033	12.54	6.345
3.067	14.64	8.096
3.100	15.86	9.725
3.133	16.69	11.14
3.167	17.43	12.38
3.200	17.95	13.28

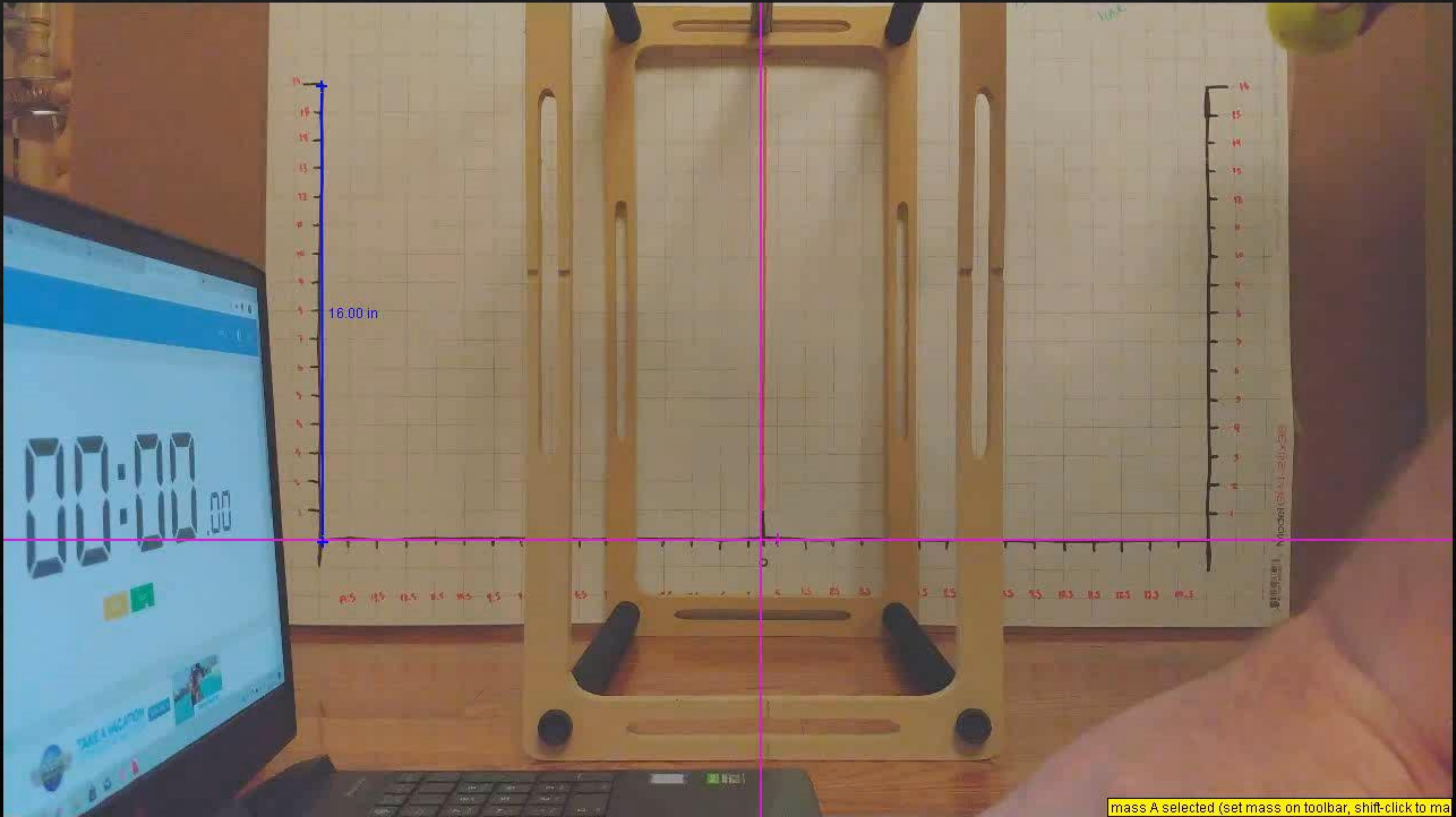


The pendulum data from the Tracker software was imported into Python where it was combined with our ODE model, which was solved using Euler's method.

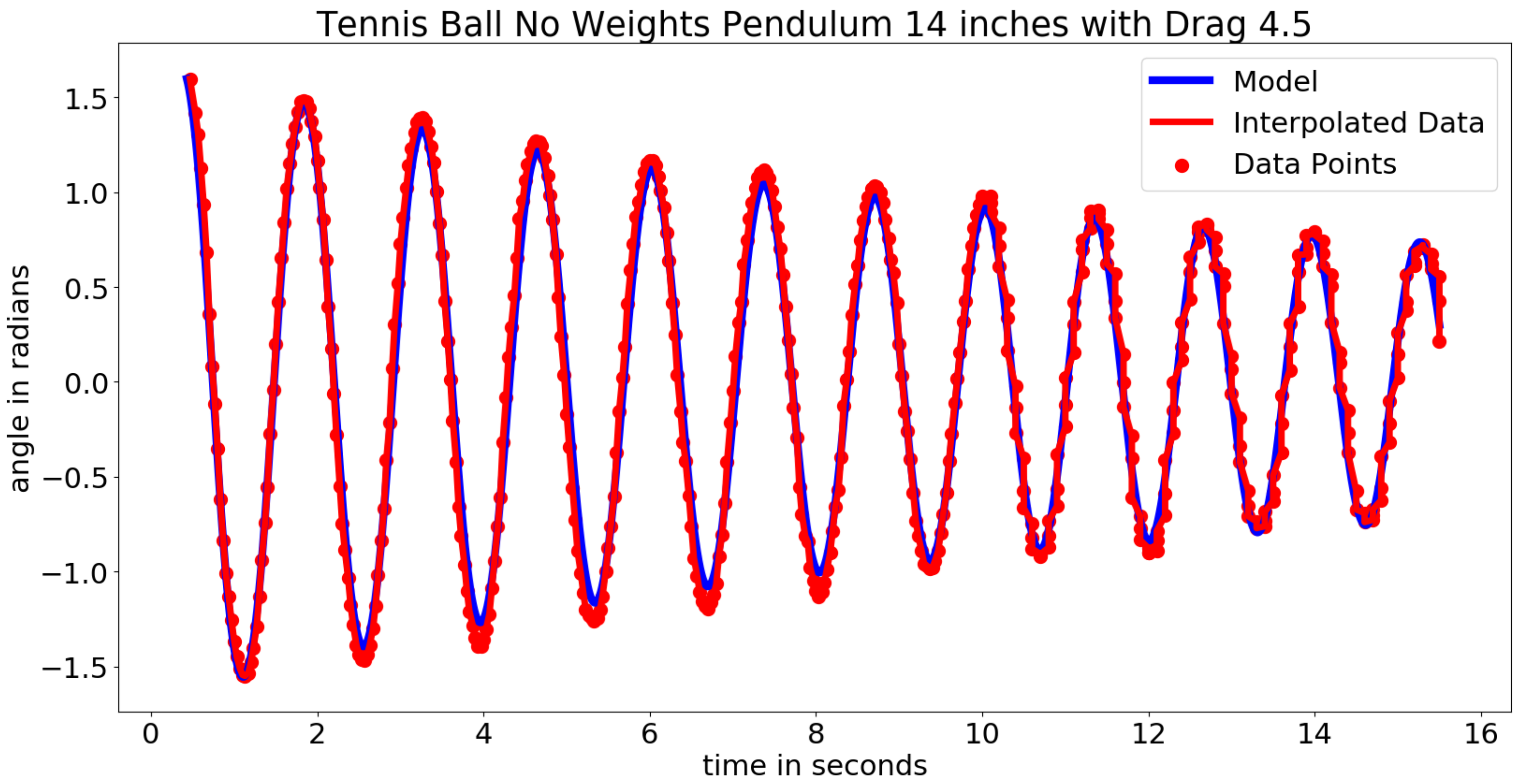
14 in of fishing line with no weights



14 in fishing line with no weights / tracked



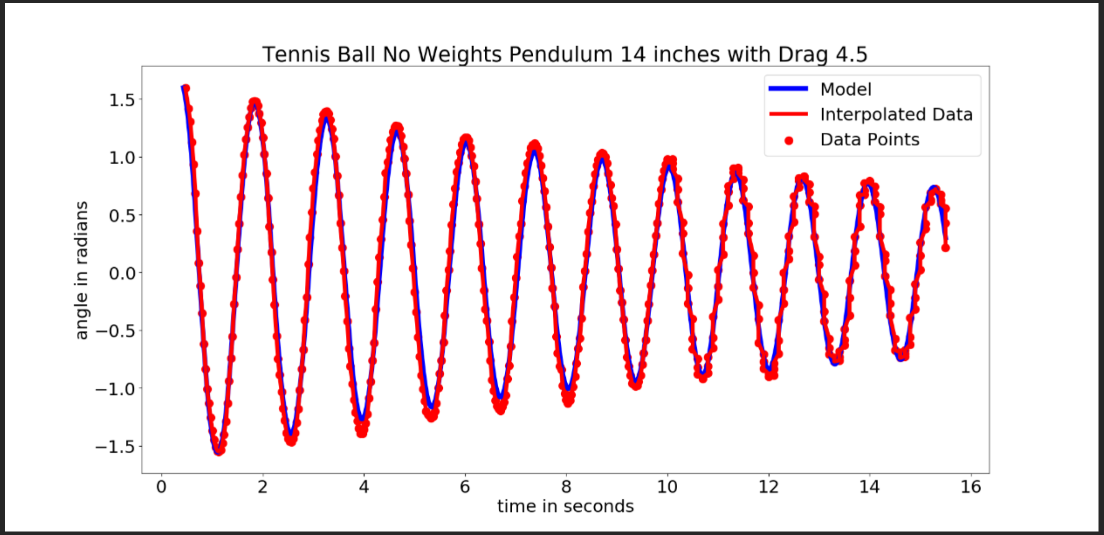
14 in of fishing line with no weights data points vs model

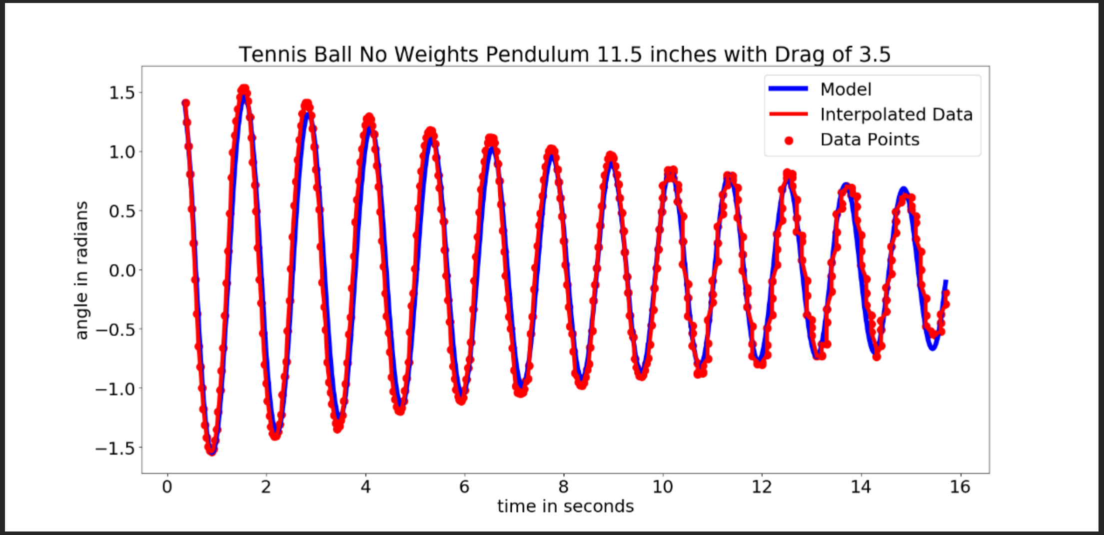


L = 14 inches

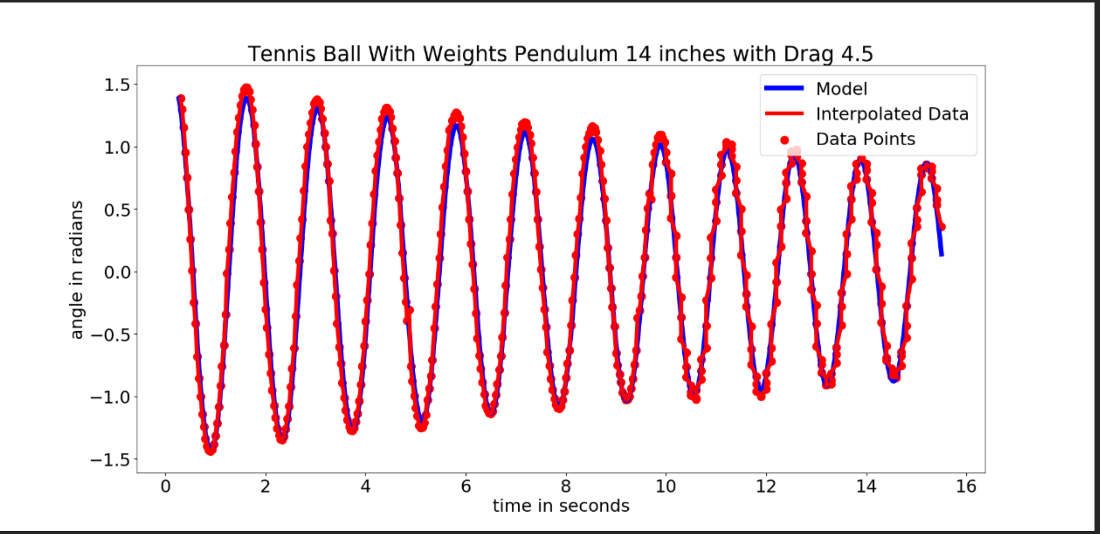
L = 11 (top)/11.5 (bottom) inches

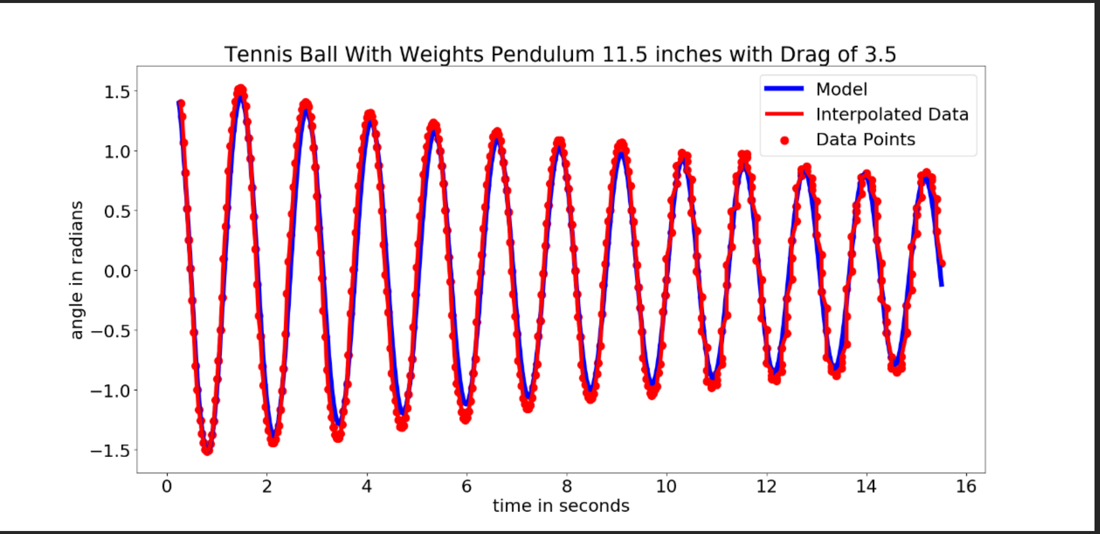
No added weight





With added weight





$$T \approx 2\pi\sqrt{L/g}$$

Thank you!

Please email me for more information.

cmccarthy@bmcc.cuny.edu

Or visit my webpage

McCarthy Differential Equations

Or see the scenario at

SIMIODE.org

NYS OER Initiative Grant

SIMIODE DEMARC Workshop

Presented at the 2020 Joint Mathematics Meetings in Denver Colorado

Wednesday January 15, 2020, 2:15 p.m.-3:10 p.m.
MAA Contributed Paper Session on Modeling-First Inquiry-Based Course Activities, II

Room 502, Meeting Room Level,
Colorado Convention Center

Organizers:
Ben Galluzzo, Clarkson University
Brian Winkel, SIMIODE brianwinkel@simiode.org
Corban Harwood, George Fox University

2:35 p.m.
Throwing a ball can be such a drag.
Chris McCarthy*, Borough of Manhattan Community College City University of New York
(1154-L5-2196)