Rock Drop Lab

2/5/20

PH 211

Katherine Skovborg

Lab Partners: Walker Davis, Casey Heiskell, Ryan Ross, Nic Hamel

Description:

In this lab we dropped birdies (with playdough in them for extra weight) as a replica for a rock. We measured the time it took for the birdie to hit the ground from different heights. The purpose of this experiment is to explore practical uses of 1 dimensional kinematics. The goal is to find a strategy for taking measurements in order to produce the most accurate data points. From here, we are to enter our data points to create a mathematical expression model that is a function of height with respect to time.

In [105]:
```
#import python tool libraries

import numpy as np
import matplotlib as mplot
import matplotlib.pyplot as plt
from numpy.polynomial import polynomial as ply
import matplotlib.patches as mpatches
```

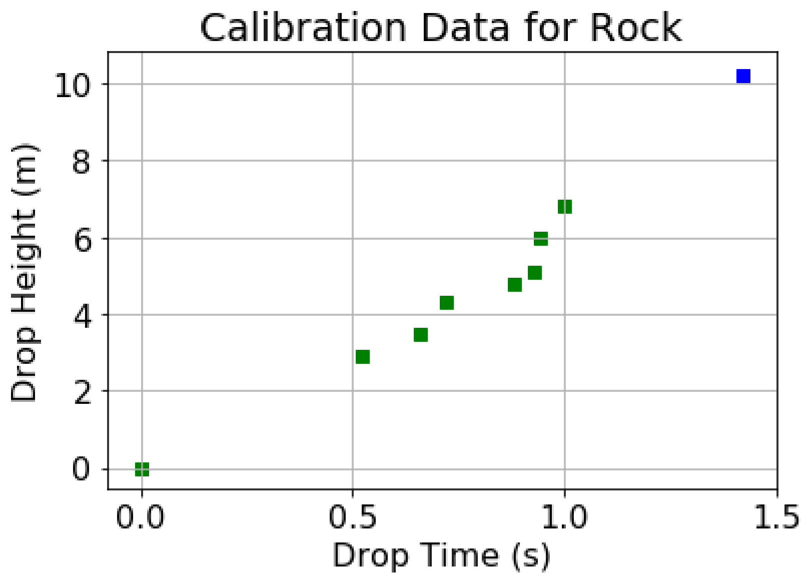
Deliverable One:

In the following code cell, we are entering our data points and checking to make sure they are entered correctly. The time data is the x variable (how long it takes to hit the ground in seconds) and the height data is the corresponding y value (the height at which the 'rock' is dropped).

In [106]:
```
#enter data

timedata = [0, .52, .66, .72, .88, .93, .94, 1.0]

heightdata = [0, 2.9, 3.5, 4.3, 4.8, 5.12, 5.96, 6.8]

#check data

print("Flight time: ", timedata)
print("Height: ", heightdata)

#check and print length of data

timedatalength = len(timedata)
heightdatalength = len(heightdata)

print("Number of data points (x): ", timedatalength)
print("Number of data points (y): ", heightdatalength)
```

```
Flight time:  [0, 0.52, 0.66, 0.72, 0.88, 0.93, 0.94, 1.0]
Height:  [0, 2.9, 3.5, 4.3, 4.8, 5.12, 5.96, 6.8]
Number of data points (x):  8
Number of data points (y):  8
```

In [110]:
```
#Generate data plot

fig, ax = plt.subplots()
ax.scatter(timedata, heightdata, color = "green", marker = "s")

ax.scatter(1.42, 10.2, color = 'blue', marker = 's')

plt.rcParams.update({'font.size': 16}) # make labels easier to read
ax.set(xlabel='Drop Time (s)', ylabel='Drop Height (m)',
       title='Calibration Data for Rock')
ax.grid()

#fig.savefig("myplot.png")
plt.show()
```



In [89]:
```
#generate polynomial coefficients

degree = 2
coefs = ply.polyfit(timedata, heightdata,degree)
print("Coefficients of the polynomial fit:", coefs)
```

```
Coefficients of the polynomial fit: [0.04432741 3.87312967 2.34706004]
```

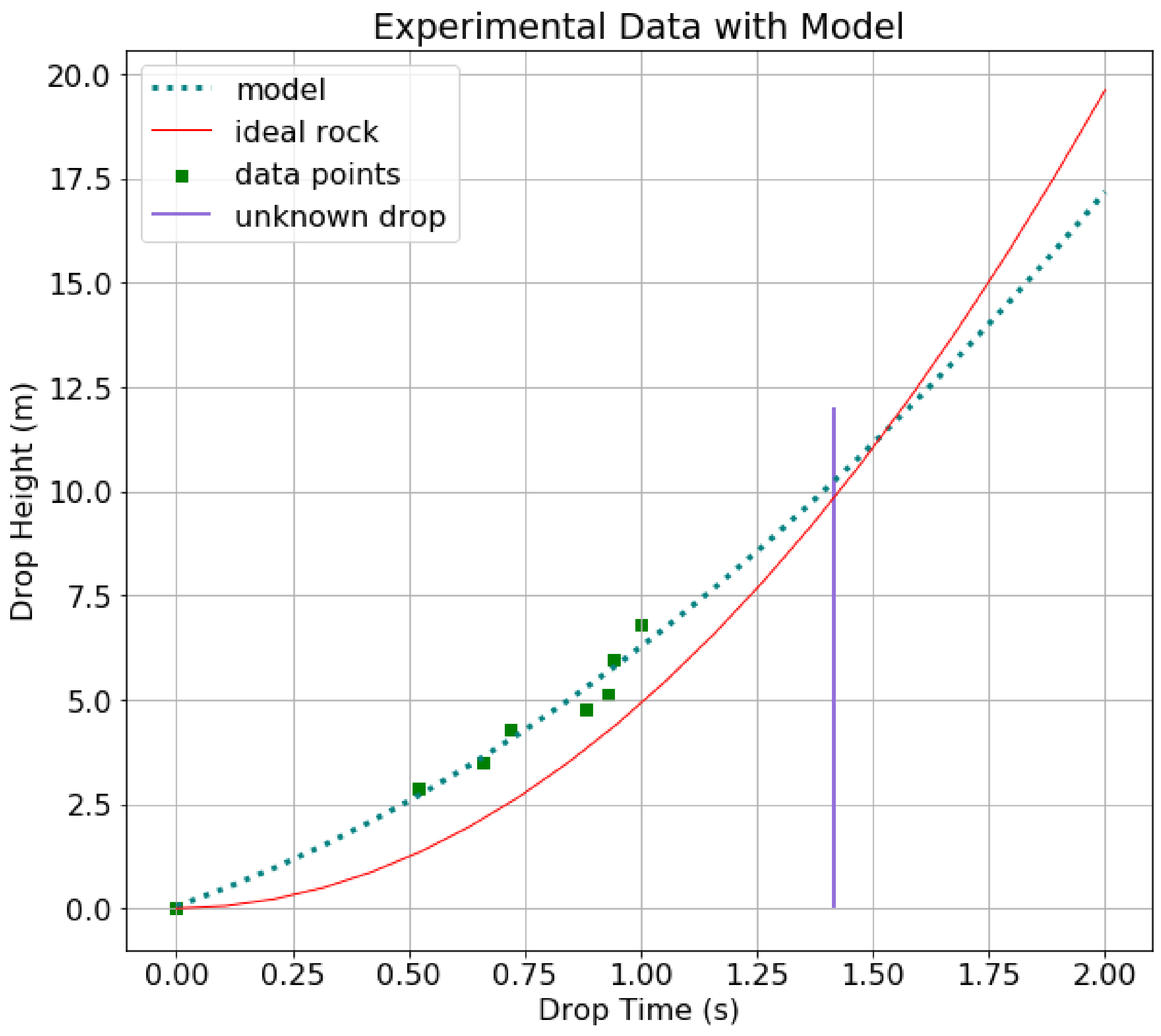
```
In [90]: #enter physics model parameters

         # generate x values for model of data
         maxtime = 2.0
         numpoints = 20
         modeltime = np.linspace(0.,maxtime,numpoints)

         # create a model height list that matches the model time
         modelheight = np.full_like(modeltime,0)
         idealrock = np.full_like(modeltime,0)

         # calculate the heights predicted from the model
         for i in range (0,numpoints):
             modelheight[i] = coefs[0] + coefs[1]*modeltime[i] + \
                 coefs[2]* modeltime[i]**2
             idealrock[i] = 0.5*9.81*modeltime[i]**2

         print("Testing the output of the loop;", modelheight)
```

```
Testing the output of the loop; [ 0.04432741  0.47803147  0.96374795
1.50147684  2.09121813  2.73297184
  3.42673796  4.17251649  4.97030743  5.82011078  6.72192655  7.6757547
2
  8.6815953   9.7394483  10.8493137  12.01119152 13.22508175 14.4909843
9
 15.80889944 17.1788269 ]
```

Deliverable Two:

In the code cells above, we used our data points to generate our mathematical expression model that is plotted below. We first calculated the coefficients of our quadratic equation. Next, we generated the x values of the model, the height list matching the model times, and calculated the heights predicted from the model. All of this is then used in the code cell below to create a plot of our model.

In [98]:
```
#plot physics model (deliverable two)

fig2, ax2 = plt.subplots()
ax2.scatter(timedata, heightdata,
            marker = 's', color = 'green',
            label = "data points")

#ax.scatter(1.42, 10.2, color = 'blue', marker = 's', label = 'unknown')

ax.plot(1.45, 10.2, color = "purple", marker = 's', label = 'unknown')


ax2.plot(modeltime, modelheight,
         color = 'teal', linestyle = ':',
         linewidth = 3., label = "model")
ax2.plot(modeltime, idealrock,
         color = 'red', linestyle = '-',
         linewidth = 1., label = "ideal rock")

unknown_data = 1.415
ax2.vlines(unknown_data, 0, 12,
         color = 'mediumpurple', linestyle = '-',
         linewidth = 2., label = "unknown drop")

plt.rcParams.update({'font.size': 16}) # make labels easier to read
ax2.set(xlabel='Drop Time (s)', ylabel='Drop Height (m)',
       title='Experimental Data with Model')

fig2.set_size_inches(10, 9)
ax2.grid()

plt.legend(loc= 2)
plt.show()
```



Deliverable Three:

When analyzing our data, it appears that there must have been an aspect of our measurement strategy that produced inaccuracies. This is because our model strays from the ideal rock model. Therefore, it is difficult to say whether or not our data supports the idea that the 'rock' reaches terminal velocity. However, as we were taking our measurements we did notice that the time it took for the 'rock' to hit the ground got closer and closer as height increased, indicating it was approaching terminal velocity. When determining the different heights we didn't choose consistent increments, which is most likely the reason it is hard to find an explanation regarding terminal velocity.

In [73]:
```
#challenge drop, measured drop time for unknown height

predicted_height = coefs[0] + coefs[1]*unknown_data + \
        coefs[2]* unknown_data**2

print("Our predicted height of unknown drop is (m):", predicted_height)
```

```
Our predicted height of unknown drop is (m): 10.224148172475992
```

Deliverable Four:

To check our calculated prediction of the unknown height, I plotted the data point on the first scatter plot to be able to visualize where our prediction lies. Unfortunately I could not figure out how to plot a single point on our physics model, but it is still easy to visualize with the unknown drop model. I think that both the output of the calculation and the visual representations on both plots show that this is a reasonable prediction.