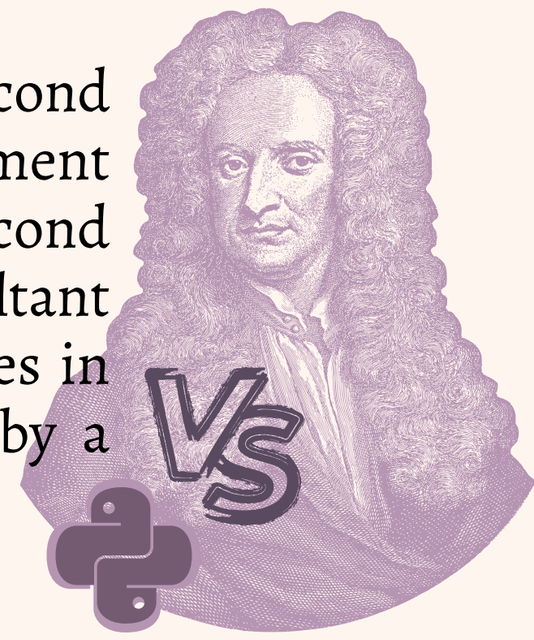
جامعة الإمام عبد الرحمن بن فيصل
IMAM ABDULRAHMAN BIN FAISAL UNIVERSITY
كلية العلوم والدراسات الإنسانية - قسم الفيزياء
College of Science and Humanities - Physics Department

NEWTON'S SECOND LAW EXPERIMENT

Nouf Alqahtani | 2200002231@iau.edu.sa - Lama Alqarni | 2190001225@iau.edu.sa- Fatimah Alshwikhat | 2190002250@iau.edu.sa - Arwa Alabdulhadi |2190001858@iau.edu.sa
Subject :Scientific Computing &Modeling. Supervised by : Dr. Ghada ameereh & Ms. Hind Al-ssay

Introduction



Using Anaconda's Jupyter Notebook (coding in Python) to simulate Newton's second law experiment, many libraries were used to do multiple tasks, define the experiment data, plot the graph, and calculate all the problems using only codes. Newton's second law answers the question of what happens to an object that has a nonzero resultant force acting on it. The experiment in real life provides a demonstration of "forces in motion" acting on a dynamic cart moving on a flat track that is accelerated by a hanging mass.[1]

Objectives

1. Verifying Newton's Second Law of motion using python.
2. Calculating the theoretical acceleration value, acceleration experimental value, and then compare the results in percentage error using python.

Methodology

Part 1: Tables

```
import pandas as pd # library for defining data and tabels
data1= pd.DataFrame({ "M(gm)":[168.28] ,"m(gm)":[50]})
data = pd.DataFrame( {"X(cm)":[20,40,60,80,100,120],
                      "t1(msec)":[214,416.63,572.65,755.37,959.59,1076],
                      "t2(msec)":[173,436.8,542.16,726.26,994.79,971.33],
                      "t(msec)":[193.5,426.72,557.41,740.82,977.19,1023.67],
                      "t(sec)":[0.19,0.43,0.56,0.74,0.98,1.02],
                      't^2(sec^2)':[0.04,0.18,0.31,0.55,0.96,1.04]})
table1=pd.DataFrame(data1)
table=pd.DataFrame(data)
```

	M(gm)	m(gm)
0	168.28	50

Table 1: The masses

Pandas library were imported as pd to define the datas for table 1 called data 1, which includes one row and two columns (mass in grams) M for the heavy mass and m for the lighter mass. Then the table were created by calling the Pandas library code pd.DataFrame(data1) and then new codes were used to design the table and give the heads, indexes and background colors. [2]

	X(cm)	t1(msec)	t2(msec)	t(msec)	t(sec)	t^2(sec^2)
0	20	214.00	173.00	193.50	0.19	0.04
1	40	416.63	436.80	426.72	0.43	0.18
2	60	572.65	542.16	557.41	0.56	0.31
3	80	755.37	726.26	740.82	0.74	0.55
4	100	959.59	994.79	977.19	0.98	0.96
5	120	1076.00	971.33	1023.67	1.02	1.04

Table 2: The different distances with the time in different units and it's conversion

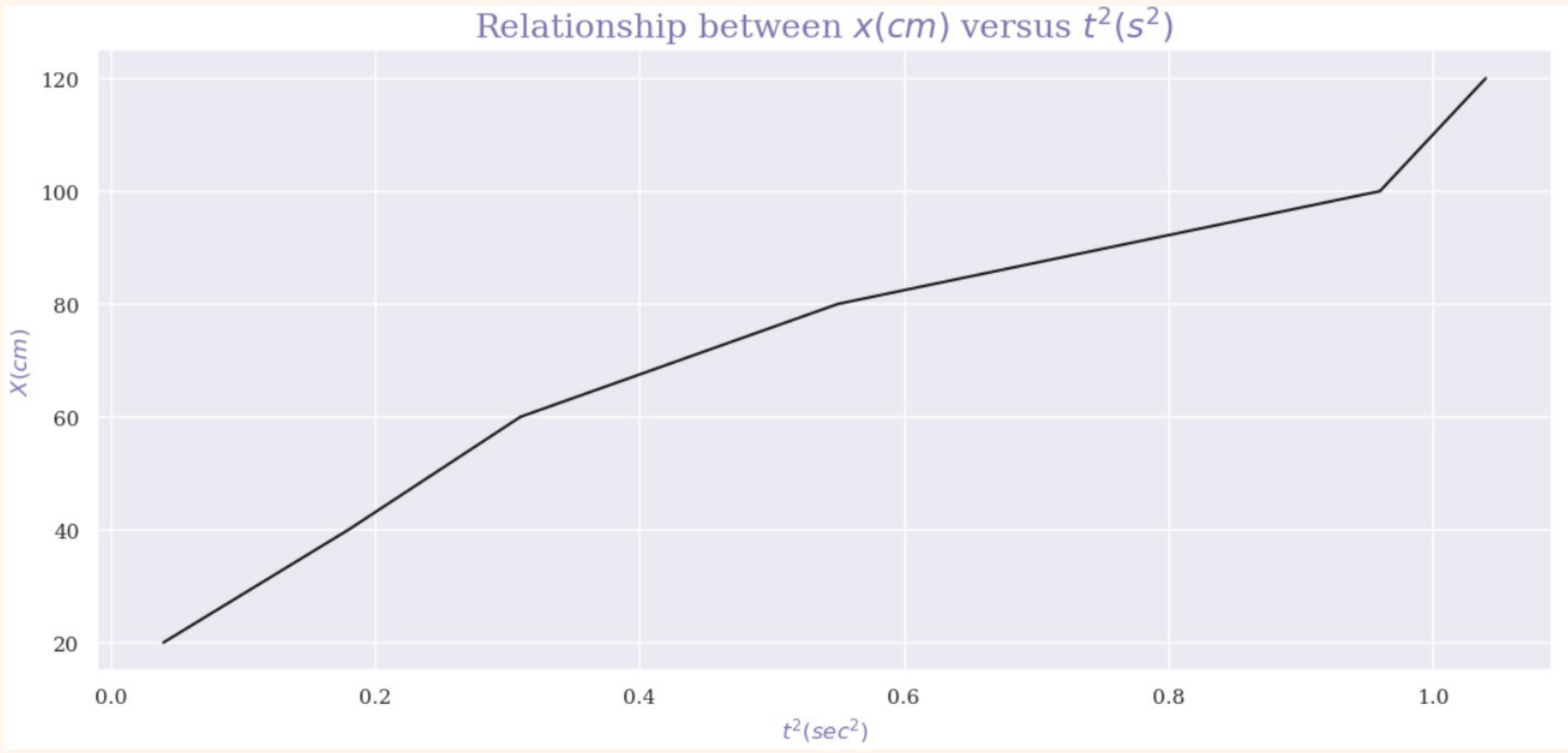
- The same code for table 1 was applied to table 2 (pd.DataFrame(data2)).
- Table 2 data: X represents the distance in cm that was measured in time 1 then in time 2 in milliseconds, the average for time 1 and time 2 was taken and then converted to seconds, and lastly it was squared. The same procedure was applied to different distances.
- Table's text was modified to be at the center. [2]

Part 2: Graph

```python
import matplotlib.pyplot as plt # library for coding grapgh
import seaborn as sns           # library for making the bg of graph blue squares
plt.rc('font', family='serif') # NEW CODE "SELF LEARNING"
plt.figure(figsize=(14,6)) #must be at first to exceed right     # NEW CODE "SELF LEARNING"
x=table['t^2(sec^2)']
y=table["X(cm)"]
plt.grid(True)
plt.plot(x,y,color='k')
sns.set()
plt.xlabel('$t^2(sec^2)$',fontsize=12, color='m')     # NEW CODE "SELF LEARNING"
plt.ylabel("$X(cm)$",fontsize=12 , color='m')         # NEW CODE "SELF LEARNING"
plt.title("Relationship between $x (cm)$ versus $t^2(s^2)$",fontsize=18 , color='m')  # NEW CODE "SELF LEARNING"

plt.show()
```

In the graph, the matplotlib.pyplot library has been imported to plot the graph, and then the Seaborn library has been imported to make the background as shown in the graph. The x-axis was defined as time squared and the y-axis as distances. New codes were used to design the graph titles, labels, fonts, and colors. [1] [3]



Graph: The relationship between the distance and the time square resulted in a linear proportional relationship.

Results & Discussion

1. $\frac{gm}{M+m}$ `a_th = 224.48 cm/s^2` — Theoretical acceleration and g is the gravitational force = 980 cm/s^2
2. `slop= 88.68 cm/s^2` — The slop from the graph using polyfit() [4]
3. $2slop$ `a_exp = 177.37 cm/s^2` — Experimental acceleration result
4. $100\left|\frac{a_{exp}-a_{th}}{a_{th}}\right|$ `P.E = 21.0 %` — Experimental error percentage

- In tables design, we had difficulty changing the color for only heads and indexes, but we finally found the right codes.
- The slope code was hard to find because we didn't find a specific code to determine the slope of the graph itself. However, we found the polyfit and linegress, but we used polyfit so we could round the result and use index to only determine the slope.

Conclusion

In conclusion, we successfully combined physics, mathematics and programming in python language to simulate Newton's second law experiment and calculate its problems.

References & codes

Acknowledgements

Thanks to D.Ghada Ameereh and to ms. Hind Baaqeel for their teaching and guidance to make this project