Physics Laboratory PH1102

Lab Report

Experiment No.: 02

- Verification of Newton's Second Law -

By : Priyanshu Mahato

Roll No. : pm21ms002

March 7, 2022

Contents

1 Aim

To verify Newton's Second law of motion using Air Track method.

2 Apparatus Required

Tracking Camera, Air Track Setup, Slider, Air Pump (to adjust the air flow along the track on which the slider moves and thus reduce friction), weights which can be put on the slider and the pulley to adjust mass and force on the slider. The Videocom Software interfaces between the air-track apparatus and a computer to collect and analyze data.

3 Experimental Setup

The hanging weight is connected (by a string) to a movable setup (slider) on an almost frictionless surface. The friction is minimized using an Air Track. A tracking camera is also set up to record the data about the position of the slider (equipped with a retroreflecting foil) at different instances of time. The schematic diagram is as below:

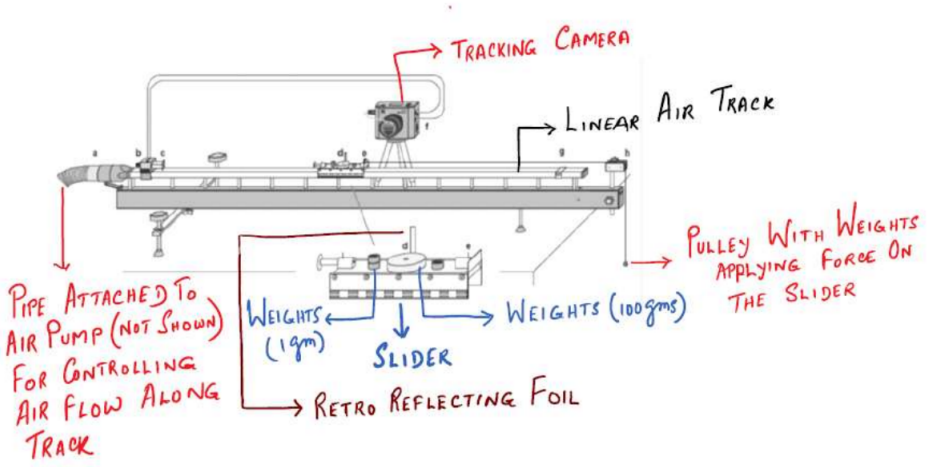

Figure 1: Experimental Setup

4 Objectives

According to Newton's Second Law of Motion, $F = ma$, where F is the force, m is the mass of the object and a is the acceleration of the object. To verify this law using the air-track system, we need to show that when the mass of the object is fixed, the acceleration of the object is proportional to the applied Force. Also, when the applied force is kept constant and the mass of the object is varied, the acceleration varies inversely with mass.

5 Theory

5.1 Newton's Second Law

According to Newton's Second law of motion, the force F required to move a body of given mass with an acceleration a is proportional to the acceleration a i.e.,

$$F \propto a$$

Also, the force F required to move a body of mass m with a given acceleration a is proportional to the mass m, given as,

$$F \propto m$$

Combining the two proportionality relations we get,

$$F \propto ma \Rightarrow F = kma$$

Here K is a constant of proportionality and the unit and dimension for force are chosen such that k has a unit magnitude and is dimensionless . So,

$$F = ma$$

5.2 Working Formulae

Let T be the tension in the string connecting the slider and the hanging weights. This T is the force F acting on the slider system : For the (vertical) motion of the hanging weight m,

$$T = mg - ma$$

For horizontal motion of the system M,

$$T = Ma + \mu Mg$$

where, g =Acceleration due to gravity(=9.8 m/s)

μ =Coefficient of friction

M =Mass of the sliding system

m =Mass of the hanging weights

Combining the above two expressions and rearranging, we get,

$$a = \frac{mg - \mu Mg}{m + M} = \frac{g(m - \mu M)}{m + M}$$

and since we are using an Air Track we may assume $\mu = 0$. Thus, we have,

$$a = \frac{mg}{m + M}$$

and,

$$T = \frac{Mmg}{m + M} + M = F$$

6 Data Plotting and Analysis

6.1 Fixed Mass Case

In this case, the combined mass of the slider and the hanging weights (i.e. $M + m$) has been kept constant and the experiment has been repeated 6 times with different values of hanging weights(m) as:

4.13 g ,3.45 g, 2.76 g, 2.08 g, 1.39 g and 0.70 g

Now we plot position, velocity and acceleration values against time from out experimental data as below:

Displacement vs. Time Graph:-

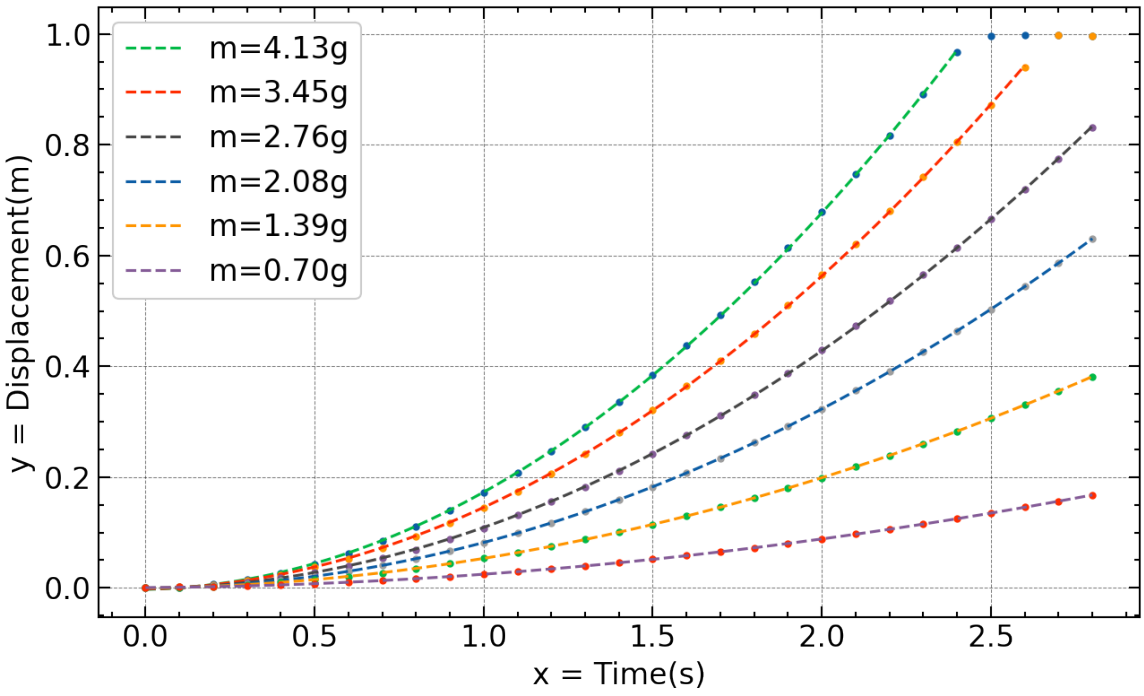

Figure 2:

Mass	Polynomial Generated by Fitting Algorithm
m = 4.13g	$0.1643t^2 + 0.01126t - 0.002624$
m = 3.45g	$0.135t^2 + 0.01245t - 0.002142$
m = 2.76g	$0.1043t^2 + 0.006136t - 0.001242$
m = 2.08g	$0.07948t^2 + 0.002629t - 0.0002204$
m = 1.39g	$0.04593t^2 + 0.007786t - 0.0004171$
m = 0.70g	$0.01974t^2 + 0.004472t + 0.00046$

Table 1: Polynomials generated by the Fitting Algorithm used

The acceleration a can be calculated using the coefficient of t^2 according to the formula $\frac{1}{2}at^2 + vt + const.$

Python Code used for plotting and curve - fitting:

```
import matplotlib.pyplot as plt
import numpy as np
plt.style.use(['science', 'notebook', 'grid'])
from scipy.optimize import curve_fit


plt.figure(figsize=(10,6), dpi=150)


time, disp = np.genfromtxt('xyz.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,1])
plt.plot(time, disp, 'o', ms=3)


time, disp = np.genfromtxt('xyz.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,1],
skip_footer=4)
z = np.polyfit(time, disp, 2)
f = np.poly1d(z)
time_opt = np.linspace(time[0], time[-1], 500)
disp_opt = f(time_opt)


print(f)


plt.plot(time_opt, disp_opt, '--', lw=1.5, label='m=4.13g')


time, disp = np.genfromtxt('xyz.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,2])
plt.plot(time, disp, 'o', ms=3)


time, disp = np.genfromtxt('xyz.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,2],
 skip_footer=2)
z = np.polyfit(time, disp, 2)
f = np.poly1d(z)
time_opt = np.linspace(time[0], time[-1], 500)
disp_opt = f(time_opt)
```

```
print(f)

plt.plot(time_opt, disp_opt, '--', lw=1.5, label='m=3.45g')

time, disp = np.genfromtxt('xyz.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,3])
plt.plot(time, disp, 'o', ms=3)

time, disp = np.genfromtxt('xyz.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,3])
z = np.polyfit(time, disp, 2)
f = np.poly1d(z)
time_opt = np.linspace(time[0], time[-1], 500)
disp_opt = f(time_opt)

print(f)

plt.plot(time_opt, disp_opt, '--', lw=1.5, label='m=2.76g')

time, disp = np.genfromtxt('xyz.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,4])
plt.plot(time, disp, 'o', ms=3)

time, disp = np.genfromtxt('xyz.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,4])
z = np.polyfit(time, disp, 2)
f = np.poly1d(z)
time_opt = np.linspace(time[0], time[-1], 500)
disp_opt = f(time_opt)

print(f)

plt.plot(time_opt, disp_opt, '--', lw=1.5, label='m=2.08g')

time, disp = np.genfromtxt('xyz.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,5])
plt.plot(time, disp, 'o', ms=3)
```

```
time, disp = np.genfromtxt('xyz.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,5])
z = np.polyfit(time, disp, 2)
f = np.poly1d(z)
time_opt = np.linspace(time[0], time[-1], 500)
disp_opt = f(time_opt)

print(f)

plt.plot(time_opt, disp_opt, '--', lw=1.5, label='m=1.39g')

time, disp = np.genfromtxt('xyz.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,6])
plt.plot(time, disp, 'o', ms = 3)

time, disp = np.genfromtxt('xyz.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,6])
z = np.polyfit(time, disp, 2)
f = np.poly1d(z)
time_opt = np.linspace(time[0], time[-1], 500)
disp_opt = f(time_opt)

print(f)

plt.plot(time_opt, disp_opt, '--', lw=1.5, label='m=0.70g')
plt.xlabel('x = Time(s)')
plt.ylabel('y = Displacement(m)')
plt.legend()
plt.savefig('displacement vs. time (fitted).png')
plt.show()
```

After this step, the formula $v(t+\frac{\delta t}{2}) = \frac{[x(t+\delta t)-x(t)]}{\delta t}$ was used to calculate the velocities for the corresponding displacements in time $\delta t = 0.1s$.

The python code used to generate a sample .csv file with velocities is as follows:

```
import csv
import random
time, disp = np.loadtxt('xyz.csv', dtype = 'float', delimiter = ',', unpack = True, usecols = [0,1])
time = time + (0.1/2)
vel = []
vel.append([])
vel.append([])
vel.append([])
vel.append([])
vel.append([])
vel.append([])
vel.append([])
for i in range (0, len(time)-1):
vel[0].append(float((disp[i+1] - disp[i])/0.1))
time, disp = np.loadtxt('xyz.csv', dtype = 'float', delimiter = ',', unpack = True, usecols = [0,2])
for i in range (0, len(time)-1):
vel[1].append(float((disp[i+1] - disp[i])/0.1))
time, disp = np.loadtxt('xyz.csv', dtype = 'float', delimiter = ',', unpack = True, usecols = [0,3])
for i in range (0, len(time)-1):
vel[2].append(float((disp[i+1] - disp[i])/0.1))
time, disp = np.loadtxt('xyz.csv', dtype = 'float', delimiter = ',', unpack = True, usecols = [0,4])
for i in range (0, len(time)-1):
vel[3].append(float((disp[i+1] - disp[i])/0.1))
time, disp = np.loadtxt('xyz.csv', dtype = 'float', delimiter = ',', unpack = True, usecols = [0,5])
for i in range (0, len(time)-1):
vel[4].append(float((disp[i+1] - disp[i])/0.1))
time, disp = np.loadtxt('xyz.csv', dtype = 'float', delimiter = ',', unpack = True, usecols = [0,6])
for i in range (0, len(time)-1):
vel[5].append(float((disp[i+1] - disp[i])/0.1))
```

```
with open('velocity.csv', 'w') as f:
    writer = csv.writer(f)
    writer.writerow(vel)
```

After this step, playing around with MS-Excel helped me get the .csv file in the desired format for plotting the graphs.

Finally, the velocity and time data looked like this:

0.05	0.01	0.01	0.005	0.005	0.005	0.005
0.15	0.052	0.047	0.034	0.026	0.018	0.006
0.25	0.085	0.072	0.054	0.039	0.029	0.012
0.35	0.118	0.103	0.08	0.059	0.038	0.019
0.45	0.155	0.134	0.098	0.072	0.049	0.02
0.55	0.191	0.157	0.121	0.093	0.059	0.026
0.65	0.226	0.188	0.136	0.105	0.07	0.031
0.75	0.258	0.214	0.16	0.121	0.077	0.036
0.85	0.293	0.242	0.185	0.137	0.09	0.041
0.95	0.325	0.268	0.201	0.154	0.098	0.041
1.05	0.358	0.298	0.232	0.17	0.103	0.047
1.15	0.394	0.33	0.247	0.186	0.114	0.051
1.25	0.427	0.353	0.268	0.206	0.123	0.054
1.35	0.461	0.378	0.288	0.216	0.129	0.057
1.45	0.492	0.41	0.312	0.232	0.139	0.062
1.55	0.525	0.435	0.332	0.25	0.149	0.064
1.65	0.557	0.458	0.353	0.265	0.16	0.07
1.75	0.59	0.49	0.373	0.281	0.165	0.072
1.85	0.618	0.512	0.394	0.296	0.18	0.077
1.95	0.646	0.544	0.412	0.314	0.188	0.082
2.05	0.68	0.564	0.436	0.33	0.198	0.088
2.15	0.706	0.589	0.453	0.342	0.204	0.087
2.25	0.733	0.616	0.474	0.361	0.214	0.093
2.35	0.76	0.636	0.494	0.376	0.224	0.098
2.45	0.304	0.662	0.51	0.391	0.234	0.103
2.55	0.005	0.68	0.536	0.407	0.242	0.103
2.65	0	0.579	0.551	0.422	0.25	0.108
2.75	-0.01	-0.005	0.569	0.436	0.26	0.114

Table 2: Time and Velocity

Here, the first column represents time and the rest represent velocities at different instances of time with the different masses.

Velocity vs. Time Graphs:-

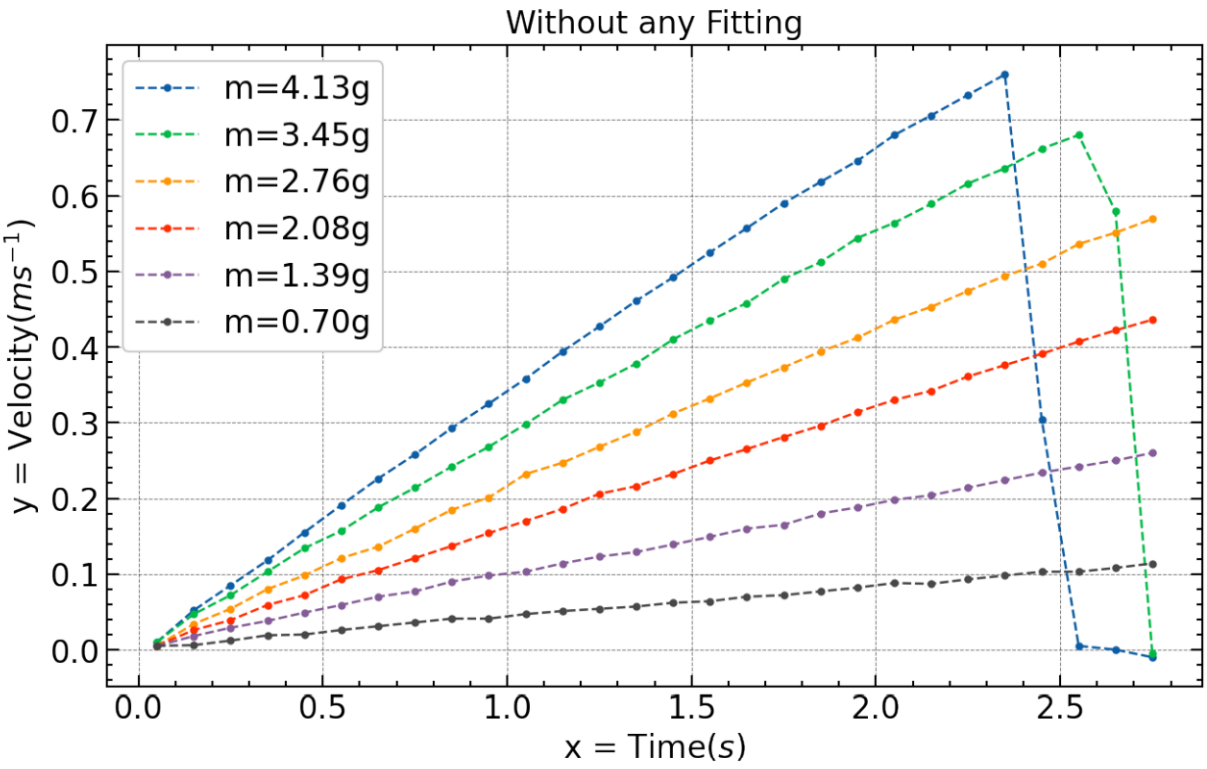

Figure 3: Velocity vs. Time

NOTE: The abrupt enormous decrease in the velocity values towards the right represents the slider hitting the boundary.

Python code used for plotting:

```
import matplotlib.pyplot as plt
import numpy as np
plt.style.use(['science', 'notebook', 'grid'])
from scipy.optimize import curve_fit


plt.figure(figsize=(10,6), dpi=150)


time, velo = np.genfromtxt('velocity.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,1])
plt.plot(time, velo, 'o--',lw=1.2, ms=3, label='m=4.13g')
```

```
time, velo = np.genfromtxt('velocity.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,2])
plt.plot(time, velo, 'o--',lw=1.2, ms=3, label='m=3.45g')


time, velo = np.genfromtxt('velocity.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,3])
plt.plot(time, velo, 'o--',lw=1.2, ms=3, label='m=2.76g')


time, velo = np.genfromtxt('velocity.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,4])
plt.plot(time, velo, 'o--',lw=1.2, ms=3, label='m=2.08g')


time, velo = np.genfromtxt('velocity.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,5])
plt.plot(time, velo, 'o--',lw=1.2, ms=3, label='m=1.39g')


time, velo = np.genfromtxt('velocity.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,6])
plt.plot(time, velo, 'o--',lw=1.2, ms = 3, label='m=0.70g')

plt.title('Without any Fitting')
plt.xlabel('x = Time($s$)')
plt.ylabel(r'y = Velocity($ms^{-1}$)')
plt.legend()
plt.savefig('vel vs. time (no-fit).png')
plt.show()
```

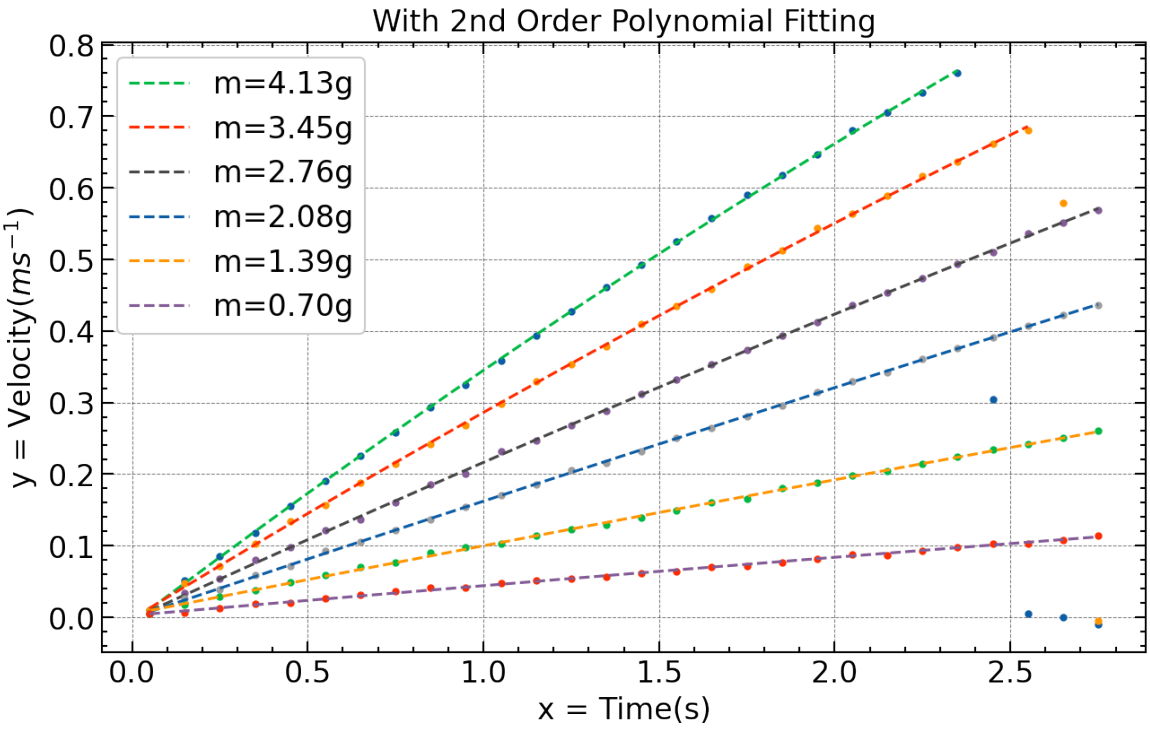

Figure 4: Velocity vs. Time

Python Code used for fitting:

```
import matplotlib.pyplot as plt
import numpy as np
plt.style.use(['science', 'notebook', 'grid'])
from scipy.optimize import curve_fit


plt.figure(figsize=(10,6), dpi=150)


# Here, consider that the variable disp is representing velocity


time, disp = np.genfromtxt('velocity.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,1])
plt.plot(time, disp, 'o', ms=3)


time, disp = np.genfromtxt('velocity.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,1],
skip_footer=4)
z = np.polyfit(time, disp, 2)
```

```
f = np.poly1d(z)
time_opt = np.linspace(time[0], time[-1], 500)
disp_opt = f(time_opt)

plt.plot(time_opt, disp_opt, '--', lw=1.5, label='m=4.13g')

time, disp = np.genfromtxt('velocity.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,2])
plt.plot(time, disp, 'o', ms=3)

time, disp = np.genfromtxt('velocity.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,2],
skip_footer=2)
z = np.polyfit(time, disp, 2)
f = np.poly1d(z)
time_opt = np.linspace(time[0], time[-1], 500)
disp_opt = f(time_opt)

plt.plot(time_opt, disp_opt, '--', lw=1.5, label='m=3.45g')

time, disp = np.genfromtxt('velocity.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,3])
plt.plot(time, disp, 'o', ms=3)

time, disp = np.genfromtxt('velocity.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,3])
z = np.polyfit(time, disp, 2)
f = np.poly1d(z)
time_opt = np.linspace(time[0], time[-1], 500)
disp_opt = f(time_opt)

plt.plot(time_opt, disp_opt, '--', lw=1.5, label='m=2.76g')

time, disp = np.genfromtxt('velocity.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,4])
plt.plot(time, disp, 'o', ms=3)

time, disp = np.genfromtxt('velocity.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,4])
```

```
z = np.polyfit(time, disp, 2)
f = np.poly1d(z)
time_opt = np.linspace(time[0], time[-1], 500)
disp_opt = f(time_opt)

plt.plot(time_opt, disp_opt, '--', lw=1.5, label='m=2.08g')

time, disp = np.genfromtxt('velocity.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,5])
plt.plot(time, disp, 'o', ms=3)

time, disp = np.genfromtxt('velocity.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,5])
z = np.polyfit(time, disp, 2)
f = np.poly1d(z)
time_opt = np.linspace(time[0], time[-1], 500)
disp_opt = f(time_opt)

plt.plot(time_opt, disp_opt, '--', lw=1.5, label='m=1.39g')

time, disp = np.genfromtxt('velocity.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,6])
plt.plot(time, disp, 'o', ms = 3)

time, disp = np.genfromtxt('velocity.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,6])
z = np.polyfit(time, disp, 2)
f = np.poly1d(z)
time_opt = np.linspace(time[0], time[-1], 500)
disp_opt = f(time_opt)

plt.plot(time_opt, disp_opt, '--', lw=1.5, label='m=0.70g')
plt.title('With 2nd Order Polynomial Fitting')
plt.xlabel('x = Time(s)')
plt.ylabel(r'y = Velocity($ms^{-1}$)')
plt.legend()
plt.savefig('vel vs. time (with fit).png')
```

```
plt.show()
```

After this, I calculated acceleration using the formula $a(t) = \frac{[x(t+dt)+x(t-dt)-2x(t)]}{dt^2}$ and wrote it into a .csv file following the same process as was done in the velocity case. The final product looks as follows:

0	0.42	0.37	0.29	0.21	0.13	0.01
0.1	0.33	0.25	0.2	0.13	0.11	0.06
0.2	0.33	0.31	0.26	0.2	0.09	0.07
0.3	0.37	0.31	0.18	0.13	0.11	0.01
0.4	0.36	0.23	0.23	0.21	0.1	0.06
0.5	0.35	0.31	0.15	0.12	0.11	0.05
0.6	0.32	0.26	0.24	0.16	0.07	0.05
0.7	0.35	0.28	0.25	0.16	0.13	0.05
0.8	0.32	0.26	0.16	0.17	0.08	0
0.9	0.33	0.3	0.31	0.16	0.05	0.06
1	0.36	0.32	0.15	0.16	0.11	0.04
1.1	0.33	0.23	0.21	0.2	0.09	0.03
1.2	0.34	0.25	0.2	0.1	0.06	0.03
1.3	0.31	0.32	0.24	0.16	0.1	0.05
1.4	0.33	0.25	0.2	0.18	0.1	0.02
1.5	0.32	0.23	0.21	0.15	0.11	0.06
1.6	0.33	0.32	0.2	0.16	0.05	0.02
1.7	0.28	0.22	0.21	0.15	0.15	0.05
1.8	0.28	0.32	0.18	0.18	0.08	0.05
1.9	0.34	0.2	0.24	0.16	0.1	0.06
2	0.26	0.25	0.17	0.12	0.06	-0.01
2.1	0.27	0.27	0.21	0.19	0.1	0.06
2.2	0.27	0.2	0.2	0.15	0.1	0.05
2.3	-4.56	0.26	0.16	0.15	0.1	0.05
2.4	-2.99	0.18	0.26	0.16	0.08	0
2.5	-0.05	-1.01	0.15	0.15	0.08	0.05
2.6	-0.1	-5.84	0.18	0.14	0.1	0.06

Table 3: Time and Acceleration

Here also, the first column represents time and the rest represent accelerations at different time stamps for the different masses considered.

Acceleration vs. Time Graphs:-

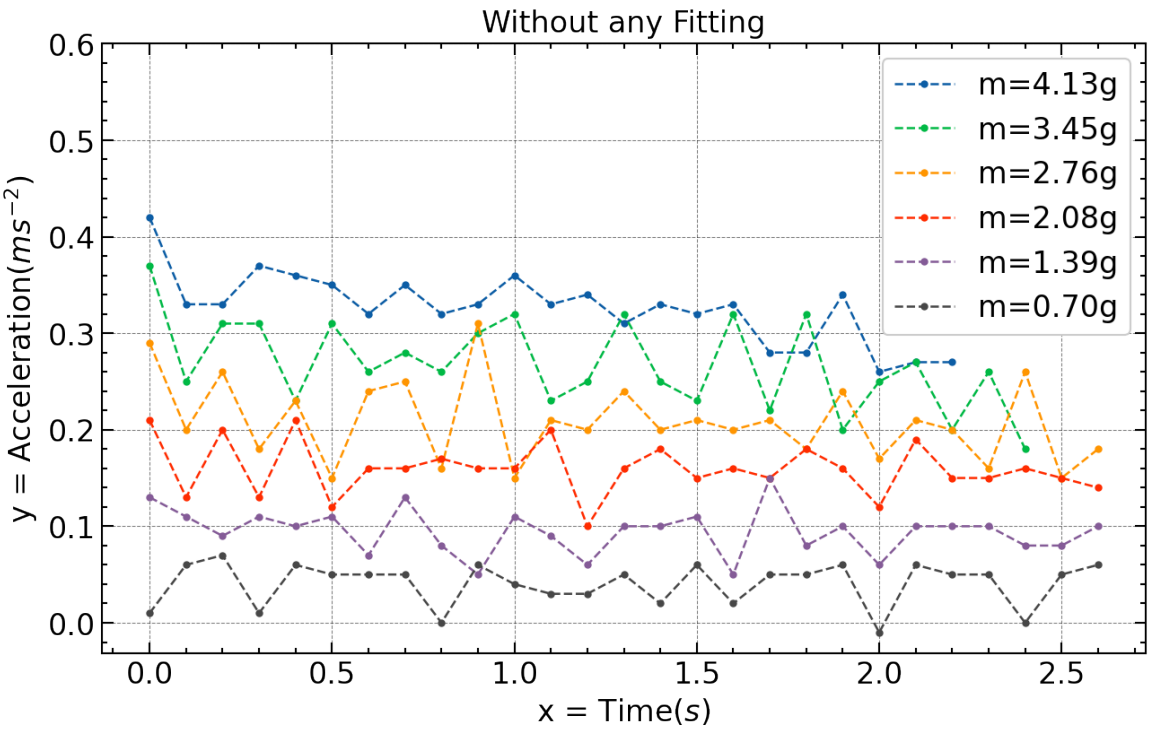

Figure 5: Acceleration vs. Time

Again, python code used for plotting this,

```
import matplotlib.pyplot as plt
import numpy as np
plt.style.use(['science', 'notebook', 'grid'])
from scipy.optimize import curve_fit


plt.figure(figsize=(10,6), dpi=150)


time, velo = np.genfromtxt('acceleration.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,1]
plt.plot(time, velo, 'o--',lw=1.2, ms=3, label='m=4.13g')



time, velo = np.genfromtxt('acceleration.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,2]
plt.plot(time, velo, 'o--',lw=1.2, ms=3, label='m=3.45g')
```

```
time, velo = np.genfromtxt('acceleration.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,3])
plt.plot(time, velo, 'o--',lw=1.2, ms=3, label='m=2.76g')


time, velo = np.genfromtxt('acceleration.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,4])
plt.plot(time, velo, 'o--',lw=1.2, ms=3, label='m=2.08g')


time, velo = np.genfromtxt('acceleration.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,5])
plt.plot(time, velo, 'o--',lw=1.2, ms=3, label='m=1.39g')


time, velo = np.genfromtxt('acceleration.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,6])
plt.plot(time, velo, 'o--',lw=1.2, ms = 3, label='m=0.70g')

plt.ylim(top = 0.6)
plt.title('Without any Fitting')
plt.xlabel('x = Time($s$)')
plt.ylabel(r'y = Acceleration($ms^{-2}$)')
plt.legend()
plt.savefig('acc vs. time (no-fit).png')
plt.show()
```

Force vs. Acceleration Graph:-

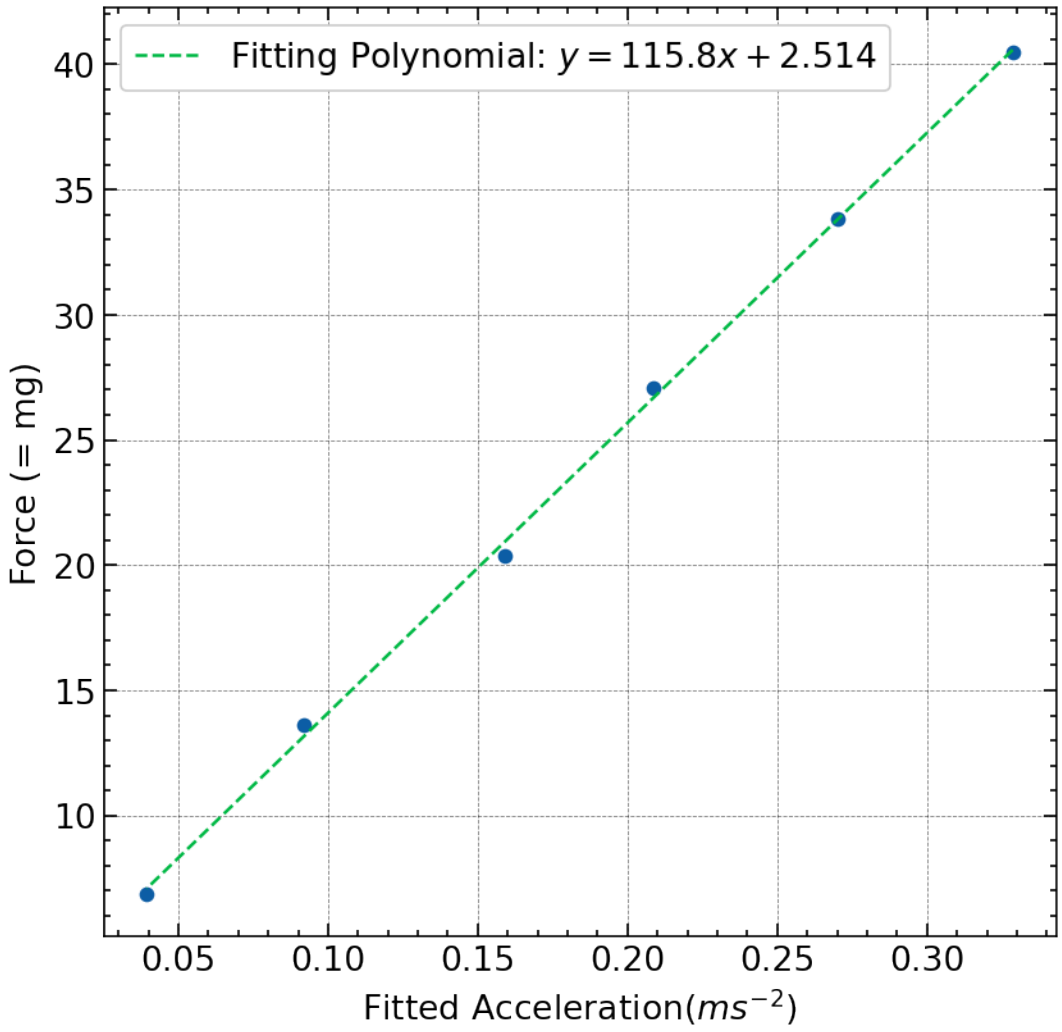

Figure 6: Force vs. Acceleration

The Force vs Acceleration graph is obtained as a Linear fit, thus, Newton's Second Law of Motion is verified for Fixed Mass Case .

The python code used for plotting this graph was:

```
import matplotlib.pyplot as plt
import numpy as np
plt.style.use(['science', 'notebook', 'grid'])
from scipy.optimize import curve_fit


force = [4.13*9.8, 3.45*9.8, 2.76*9.8, 2.08*9.8, 1.39*9.8, 0.70*9.8]
```

```
accln = [0.3286, 0.27, 0.2086, 0.15896, 0.09186, 0.03948]
plt.figure(figsize=(8,8), dpi=120)
plt.plot(accln, force, 'o')
z, res, _, sing, _ = np.polyfit(accln, force, 1, full='True')
f = np.poly1d(z)
print ("Fitting polynomial =",f)
print ('Sum of Residual Values =',res)
print('Singluar Asymptomatic Errors =',sing)
accln_opt = np.linspace(accln[0], accln[-1], 500)
force_opt = f(accln_opt)
plt.xlabel ('Fitted Acceleration($ms^{-2}$)')
plt.ylabel ('Force (= mg)')
plt.plot(accln_opt, force_opt, '--', lw=1.5, label=r'Equation of fitted curve: $y = 115.8x + 2.514$')
plt.legend()
plt.savefig('final.png')
plt.show()
```

This is the final graph for the `Fixed Mass Case`.

6.2 Error Analysis for Fixed Mass Case:

The sum of residual values generated by the above code is:

$$\texttt{Sum of Residual Values} = 0.7136271$$

And the asymptotic errors in each of the parameters of the fitting polynomial is:

Parameters	Singular Asymptotic Error
a (or m)	1.37086951
b	0.34744323

Table 4: Errors in parameters of Fitting Polynomial

Total Mass (Theoretical Value) $= 92.52g + 4.13g = 96.65g(\pm 1.37086951\%)$

Total Mass (Measured Value from Graph) $= 115.8g$

$\Rightarrow$ Relative Error $= \frac{115.8-96.65}{96.65} \times 100\% = 19.8\%$

6.3 Fixed Force Case

In this case, the force F (i.e tension T in string) acting on the slider has been kept constant and the experiment has been repeated 4 times with different values of mass of slider system M (i.e. slider+additional weights on slider) as:

400.61g, 300.47g, 200.29g and 100.01g

And mass of slider $(m_s) = 92.52$ g

Now we plot position, velocity and acceleration values against time from out experimental data as below:

Displacement vs. Time Graph:-

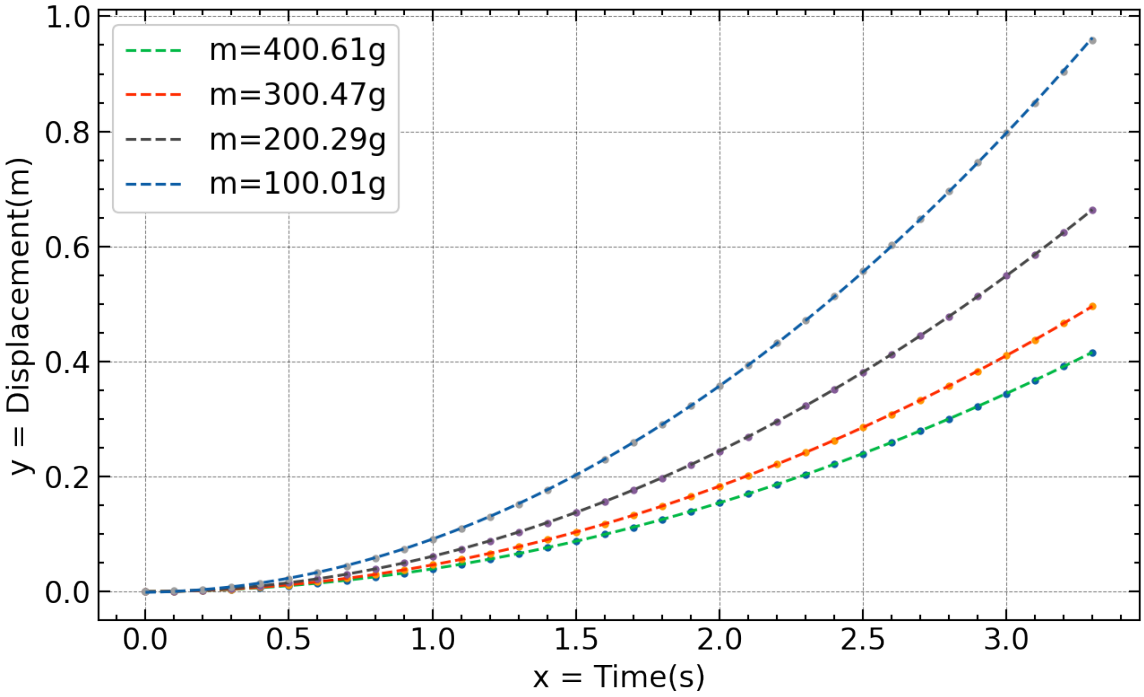

Figure 7: Displacement vs. Time

Mass	Polynomial Generated by Fitting Algorithm
m = 400.61g	$0.03764t^2 + 0.00191t - 0.0002409$
m = 300.47g	$0.04515t^2 + 0.001345t - 0.0003122$
m = 200.29g	$0.06083t^2 + 0.0004967t - 0.000176$
m = 100.01g	$0.08683t^2 + 0.005643t - 0.001379$

Table 5: Polynomials generated by the Fitting Algorithm used

The acceleration a can be calculated using the coefficient of t^2 according to the formula $\frac{1}{2}at^2 + vt + const.$

The python code used for the plotting and data analysis is as follows:

```
import matplotlib.pyplot as plt
import numpy as np
plt.style.use(['science', 'notebook', 'grid'])
from scipy.optimize import curve_fit


plt.figure(figsize=(10,6), dpi=150)


time, disp = np.genfromtxt('displacement.csv', dtype = 'float', delimiter = ',', unpack = True,usecols=[0,1])
plt.plot(time, disp, 'o', ms=3)


time, disp = np.genfromtxt('displacement.csv', dtype = 'float', delimiter = ',', unpack = True,usecols=[0,1])
z = np.polyfit(time, disp, 2)
f = np.poly1d(z)
time_opt = np.linspace(time[0], time[-1], 500)
disp_opt = f(time_opt)


print(f)


plt.plot(time_opt, disp_opt, '--', lw=1.5, label='m=400.61g')


time, disp = np.genfromtxt('displacement.csv', dtype = 'float', delimiter = ',', unpack = True,usecols=[0,2])
plt.plot(time, disp, 'o', ms=3)
```

```
time, disp = np.genfromtxt('displacement.csv', dtype = 'float', delimiter = ',', unpack = True,usecols=[0,2])
z = np.polyfit(time, disp, 2)
f = np.poly1d(z)
time_opt = np.linspace(time[0], time[-1], 500)
disp_opt = f(time_opt)

print(f)

plt.plot(time_opt, disp_opt, '--', lw=1.5, label='m=300.47g')

time, disp = np.genfromtxt('displacement.csv', dtype = 'float', delimiter = ',', unpack = True,usecols=[0,3])
plt.plot(time, disp, 'o', ms=3)

time, disp = np.genfromtxt('displacement.csv', dtype = 'float', delimiter = ',', unpack = True,usecols=[0,3])
z = np.polyfit(time, disp, 2)
f = np.poly1d(z)
time_opt = np.linspace(time[0], time[-1], 500)
disp_opt = f(time_opt)

print(f)

plt.plot(time_opt, disp_opt, '--', lw=1.5, label='m=200.29g')

time, disp = np.genfromtxt('displacement.csv', dtype = 'float', delimiter = ',', unpack = True,usecols=[0,4])
plt.plot(time, disp, 'o', ms=3)

time, disp = np.genfromtxt('displacement.csv', dtype = 'float', delimiter = ',', unpack = True,usecols=[0,4])
z = np.polyfit(time, disp, 2)
f = np.poly1d(z)
time_opt = np.linspace(time[0], time[-1], 500)
disp_opt = f(time_opt)

print(f)
```

```
plt.plot(time_opt, disp_opt, '--', lw=1.5, label='m=100.01g')
plt.xlabel('x = Time(s)')
plt.ylabel('y = Displacement(m)')
plt.legend()
plt.savefig('displacement vs. time (fitter).png')
plt.show()
```

After this step, I used the following code to calculate velocity for each case using the formula $v(t + \frac{dt}{2}) = \frac{[x(t+dt)-x(t)]}{dt}$, and write it into a .csv file:

```
import csv
import random
time, disp = np.loadtxt('displacement.csv', dtype = 'float', delimiter = ',', unpack = True, usecols = [0,1])
time = time + (0.1/2)
vel = []
vel.append([])
vel.append([])
vel.append([])
vel.append([])
vel.append([])
vel.append([])
vel.append([])
for i in range (0, len(time)-1):
vel[0].append(float((disp[i+1] - disp[i])/0.1))
time, disp = np.loadtxt('displacement.csv', dtype = 'float', delimiter = ',', unpack = True, usecols = [0,2])
for i in range (0, len(time)-1):
vel[1].append(float((disp[i+1] - disp[i])/0.1))
time, disp = np.loadtxt('displacement.csv', dtype = 'float', delimiter = ',', unpack = True, usecols = [0,3])
for i in range (0, len(time)-1):
vel[2].append(float((disp[i+1] - disp[i])/0.1))
```

```
time, disp = np.loadtxt('displacement.csv', dtype = 'float', delimiter = ',', unpack = True, usecols = [0,4])
for i in range (0, len(time)-1):
vel[3].append(float((disp[i+1] - disp[i])/0.1))



with open('velocity1.csv', 'w') as f:
writer = csv.writer(f)
writer.writerow(vel)
```

After this, I played around with the data in MS-Excel, and the final data looks like this:

0.05	0.003	0.006	0.011	0.005
0.15	0.015	0.015	0.015	0.026
0.25	0.018	0.021	0.031	0.046
0.35	0.028	0.036	0.044	0.062
0.45	0.036	0.041	0.056	0.079
0.55	0.044	0.051	0.067	0.101
0.65	0.051	0.062	0.08	0.113
0.75	0.06	0.067	0.093	0.136
0.85	0.064	0.08	0.102	0.154
0.95	0.075	0.087	0.116	0.17
1.05	0.079	0.095	0.126	0.191
1.15	0.088	0.106	0.142	0.205
1.25	0.097	0.113	0.149	0.221
1.35	0.103	0.123	0.164	0.242
1.45	0.113	0.134	0.18	0.257
1.55	0.119	0.139	0.186	0.278
1.65	0.123	0.152	0.205	0.293
1.75	0.134	0.159	0.211	0.314
1.85	0.144	0.167	0.227	0.329
1.95	0.149	0.18	0.239	0.345
2.05	0.155	0.185	0.249	0.365
2.15	0.164	0.196	0.265	0.384
2.25	0.17	0.203	0.275	0.398
2.35	0.18	0.214	0.289	0.417
2.45	0.185	0.223	0.298	0.432
2.55	0.196	0.232	0.311	0.448
2.65	0.2	0.242	0.322	0.468
2.75	0.209	0.249	0.334	0.478
2.85	0.216	0.257	0.347	0.496
2.95	0.223	0.268	0.358	0.512
3.05	0.232	0.278	0.37	0.525
3.15	0.239	0.285	0.381	0.54
3.25	0.247	0.296	0.391	0.553

Table 6: Time and Velocity

In this table, the first column represents different time stamps and the rest represent the velocities at those time stamps for the different masses considered.

Velocity vs. Time Graph:-

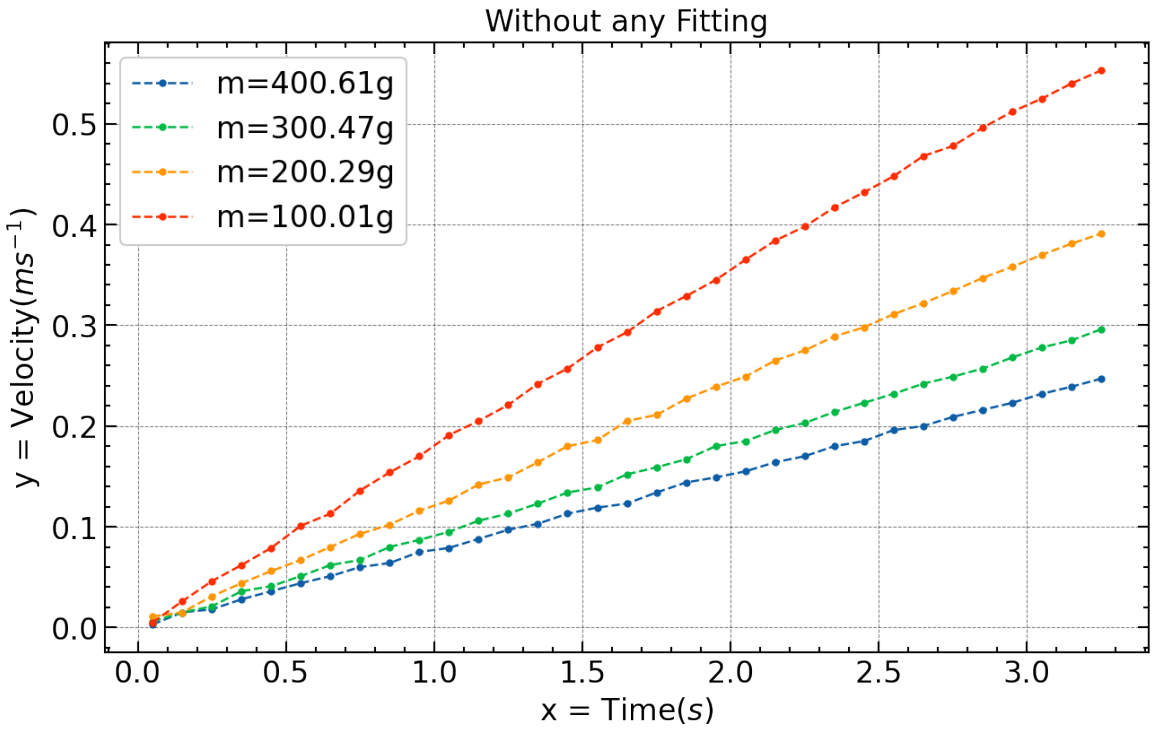

Figure 8: Velocity vs. Time

Python Code used to plot this graph is as follows:

```
import matplotlib.pyplot as plt
import numpy as np
plt.style.use(['science', 'notebook', 'grid'])
from scipy.optimize import curve_fit


plt.figure(figsize=(10,6), dpi=150)


time, velo = np.genfromtxt('velocity1.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,1])
plt.plot(time, velo, 'o--',lw=1.2, ms=3, label='m=400.61g')



time, velo = np.genfromtxt('velocity1.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,2])
```

```
plt.plot(time, velo, 'o--',lw=1.2, ms=3, label='m=300.47g')


time, velo = np.genfromtxt('velocity1.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,3])
plt.plot(time, velo, 'o--',lw=1.2, ms=3, label='m=200.29g')


time, velo = np.genfromtxt('velocity1.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,4])
plt.plot(time, velo, 'o--',lw=1.2, ms=3, label='m=100.01g')

plt.title('Without any Fitting')
plt.xlabel('x = Time($s$)')
plt.ylabel(r'y = Velocity($ms^{-1}$)')
plt.legend()
plt.savefig('vel vs. time (nfit).png')
plt.show()
```

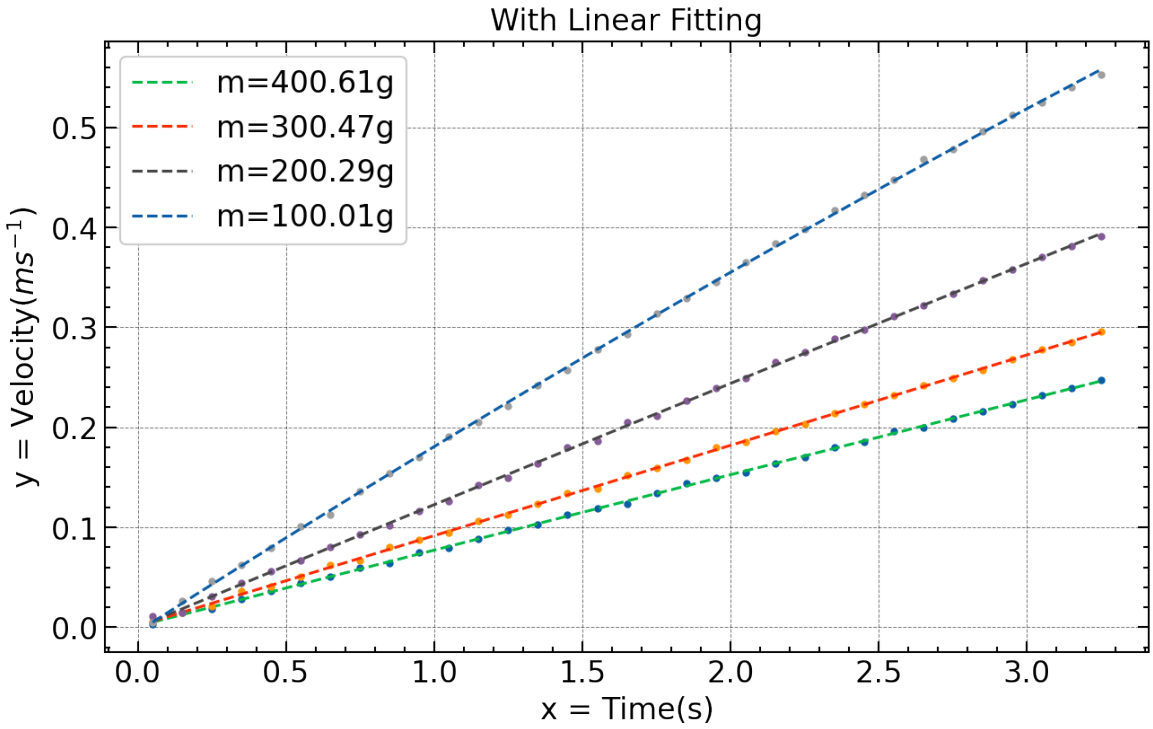

Figure 9: Velocity vs. Time

The Python code used for plotting and fitting the above curve is as follows:

```
import matplotlib.pyplot as plt
import numpy as np
plt.style.use(['science', 'notebook', 'grid'])
from scipy.optimize import curve_fit


plt.figure(figsize=(10,6), dpi=150)


# Here, consider that the variable disp is representing velocity


time, disp = np.genfromtxt('velocity1.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,1])
plt.plot(time, disp, 'o', ms=3)


time, disp = np.genfromtxt('velocity1.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,1])
z = np.polyfit(time, disp, 2)
f = np.poly1d(z)
time_opt = np.linspace(time[0], time[-1], 500)
disp_opt = f(time_opt)


plt.plot(time_opt, disp_opt, '--', lw=1.5, label='m=400.61g')


time, disp = np.genfromtxt('velocity1.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,2])
plt.plot(time, disp, 'o', ms=3)


time, disp = np.genfromtxt('velocity1.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,2])
z = np.polyfit(time, disp, 2)
f = np.poly1d(z)
time_opt = np.linspace(time[0], time[-1], 500)
disp_opt = f(time_opt)


plt.plot(time_opt, disp_opt, '--', lw=1.5, label='m=300.47g')
```

```
time, disp = np.genfromtxt('velocity1.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,3])
plt.plot(time, disp, 'o', ms=3)

time, disp = np.genfromtxt('velocity1.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,3])
z = np.polyfit(time, disp, 2)
f = np.poly1d(z)
time_opt = np.linspace(time[0], time[-1], 500)
disp_opt = f(time_opt)

plt.plot(time_opt, disp_opt, '--', lw=1.5, label='m=200.29g')

time, disp = np.genfromtxt('velocity1.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,4])
plt.plot(time, disp, 'o', ms=3)

time, disp = np.genfromtxt('velocity1.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,4])
z = np.polyfit(time, disp, 2)
f = np.poly1d(z)
time_opt = np.linspace(time[0], time[-1], 500)
disp_opt = f(time_opt)

plt.plot(time_opt, disp_opt, '--', lw=1.5, label='m=100.01g')

plt.title('With Linear Fitting')
plt.xlabel('x = Time(s)')
plt.ylabel(r'y = Velocity($ms^{-1}$)')
plt.legend()
plt.savefig('vel vs. time (ffit).png')
plt.show()
```

Next, I calculated acceleration from the provided data using the formula $a(t) = \frac{[x(t+dt)+x(t-dt)-2x(t)]}{dt^2}$, and wrote it into a.csv file using the following piece of code:

```
import csv
import random
acc = []
acc.append([])
acc.append([])
acc.append([])
acc.append([])
acc.append([])
time, disp = np.loadtxt('displacement.csv', dtype = 'float', delimiter = ',', unpack = True, usecols = [0,1])
for i in range (1, len(disp)-1):
acc[0].append((disp[i+1]+disp[i-1]-2*disp[i])/0.1**2)
time, disp = np.loadtxt('displacement.csv', dtype = 'float', delimiter = ',', unpack = True, usecols = [0,2])
for i in range (1, len(disp)-1):
acc[1].append((disp[i+1]+disp[i-1]-2*disp[i])/0.1**2)
time, disp = np.loadtxt('displacement.csv', dtype = 'float', delimiter = ',', unpack = True, usecols = [0,3])
for i in range (1, len(disp)-1):
acc[2].append((disp[i+1]+disp[i-1]-2*disp[i])/0.1**2)
time, disp = np.loadtxt('displacement.csv', dtype = 'float', delimiter = ',', unpack = True, usecols = [0,4])
for i in range (1, len(disp)-1):
acc[3].append((disp[i+1]+disp[i-1]-2*disp[i])/0.1**2)
with open('acceleration1.csv', 'w') as f:
writer = csv.writer(f)
writer.writerow(acc)
```

After playing around with the data in MS-Excel, the final dataset looks somewhat like this:

0	0.12	0.09	0.04	0.21
0.1	0.03	0.06	0.16	0.2
0.2	0.1	0.15	0.13	0.16
0.3	0.08	0.05	0.12	0.17
0.4	0.08	0.1	0.11	0.22
0.5	0.07	0.11	0.13	0.12
0.6	0.09	0.05	0.13	0.23
0.7	0.04	0.13	0.09	0.18
0.8	0.11	0.07	0.14	0.16
0.9	0.04	0.08	0.1	0.21
1	0.09	0.11	0.16	0.14
1.1	0.09	0.07	0.07	0.16
1.2	0.06	0.1	0.15	0.21
1.3	0.1	0.11	0.16	0.15
1.4	0.06	0.05	0.06	0.21
1.5	0.04	0.13	0.19	0.15
1.6	0.11	0.07	0.06	0.21
1.7	0.1	0.08	0.16	0.15
1.8	0.05	0.13	0.12	0.16
1.9	0.06	0.05	0.1	0.2
2	0.09	0.11	0.16	0.19
2.1	0.06	0.07	0.1	0.14
2.2	0.1	0.11	0.14	0.19
2.3	0.05	0.09	0.09	0.15
2.4	0.11	0.09	0.13	0.16
2.5	0.04	0.1	0.11	0.2
2.6	0.09	0.07	0.12	0.1
2.7	0.07	0.08	0.13	0.18
2.8	0.07	0.11	0.11	0.16
2.9	0.09	0.1	0.12	0.13
3	0.07	0.07	0.11	0.15
3.1	0.08	0.11	0.1	0.13

Table 7: Time and Acceleration

In this dataframe too, the first column represents the different timestamps and the rest represent the acceleration at those timestamps for different masses considered.

Acceleration vs. Time Graph:-

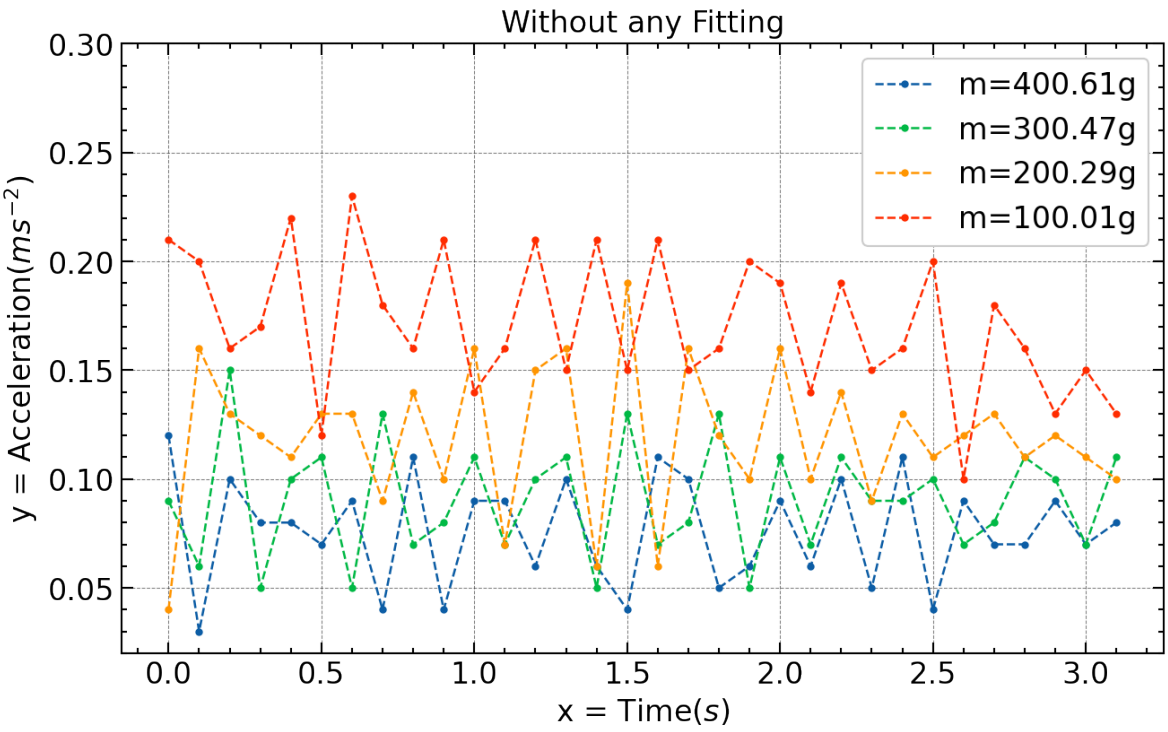

Figure 10: Acceleration vs. Time

The python code used to generate this plot is as follows:

```
import matplotlib.pyplot as plt
import numpy as np
plt.style.use(['science', 'notebook', 'grid'])
from scipy.optimize import curve_fit


plt.figure(figsize=(10,6), dpi=150)


time, velo = np.genfromtxt('acceleration1.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,1])
plt.plot(time, velo, 'o--',lw=1.2, ms=3, label='m=400.61g')



time, velo = np.genfromtxt('acceleration1.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,2])
plt.plot(time, velo, 'o--',lw=1.2, ms=3, label='m=300.47g')
```

```
time, velo = np.genfromtxt('acceleration1.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,3])
plt.plot(time, velo, 'o--',lw=1.2, ms=3, label='m=200.29g')



time, velo = np.genfromtxt('acceleration1.csv', dtype = 'float', delimiter = ',', unpack = True, usecols=[0,4])
plt.plot(time, velo, 'o--',lw=1.2, ms=3, label='m=100.01g')
plt.ylim(top=0.3)
plt.title('Without any Fitting')
plt.xlabel('x = Time($s$)')
plt.ylabel(r'y = Acceleration($ms^{-2}$)')
plt.legend()
plt.savefig('acc vs time (no-fit).png')
plt.show()
```

Mass vs. Inverse Fitted Acceleration Graph:-

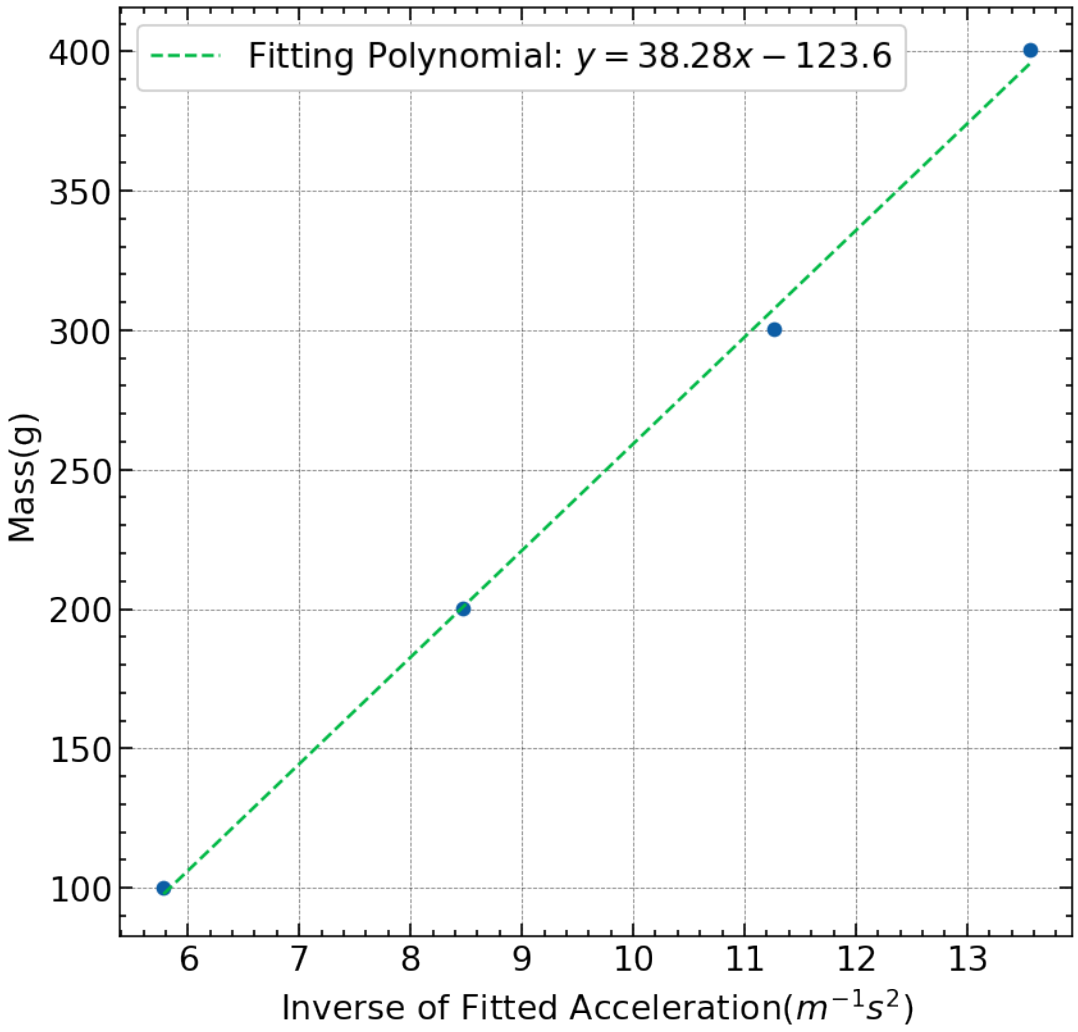

Figure 11: Mass vs. Inverse of Acceleration

Python code used for plotting and fitting is as follows:

```
import matplotlib.pyplot as plt
import numpy as np
plt.style.use(['science', 'notebook', 'grid'])
from scipy.optimize import curve_fit


force = [100.01, 200.29, 300.47, 400.61]
inaccln = [5.78, 8.47, 11.26, 13.56]
plt.figure(figsize=(8,8), dpi=120)
```

```
plt.plot(inaccln, force, 'o')
z, res, _, sing, _ = np.polyfit(inaccln, force, 1, full='True')
f = np.poly1d(z)
print ("Fitting polynomial =",f)
print ('Sum of Residual Values =',res)
print('Singluar Asymptotic Errors =',sing)
inaccln_opt = np.linspace(inaccln[0], inaccln[-1], 500)
force_opt = f(inaccln_opt)
plt.xlabel ('Inverse of Fitted Acceleration($ms^{-2}$)')
plt.ylabel ('Force (= mg)')
plt.plot(inaccln_opt, force_opt, '--', lw=1.5, label=r'Fitting Polynomial: $y = 38.28x - 123.6$')
plt.legend()
plt.savefig('final2.png')
plt.show()
```

6.4 Error Analysis for Fixed Force Case:

The sum of residual values generated by the above code is:

$$\texttt{Sum of Residual Values} = 80.46831428$$

And the asymptotic errors in each of the parameters of the fitting polynomial is:

Parameters	Singular Asymptotic Error
a (or F)	1.39928567
b	0.20493804

Table 8: Errors in parameters of Fitting Polynomial

Accepted Value of Force $= 0.0405N$

Value of Force obtained from the above graph fitting(i.e., slope of m vs. a^{-1} graph) $= 0.03828N$

Absolute Error in Force $= |0.0405N - 0.03828N| = 0.00222N$

Relative Percentage Error $= \dfrac{0.0022}{0.0405} \times 100\% = 5.4321\%$

7 Conclusion

We have verified using experimental data (empirically), that:

1. Force is directly proportional to acceleration.
2. Mass is inversely proportional to acceleration with slope of the graph representing Force.

Thus, we have experimentally proven Newton's 2^{nd} Law of Motion.

8 Acknowledgement

I would like to express my gratitude to:

1. Prof. Ritesh Kumar Singh for allowing us to write this lab report and providing us the data for the experiment.
2. The creators of the NumPy, SciPy, csv, and MatPlotlib python packages which helped me plot the graphs and perform fitting on the provided data.