H02e Video Analysis of Kinematical Processes

2024-11-12

Author(s)

- **Carla Rotzoll**, 50% contribition
- **Mirzokhid Ganiev**, 50% contribution

Group Number: 03

Abstract

In this report, we analysed a provided video of a free falling object and our own video of a cylinder rolling from an incline, utilising a video analysis and modelling tool, **Tracker**, to find the best model for former and use the data from the later to find the gravitational acceleration, g. The former experiment yielded that the model considering the presence of turbulent force (which corresponds to the non-linear model, namely quadratic model) yielded the most accurate value, with a coefficient of determination, R^2, of $0.998 \approx 1$. A nearly perfect fitting. For the later experiment of a rolling cylinder we had **-6.64**% accuracy with the g, where the deviation comes from not accounting for the lose of energy due to friction with the rolling surface and in air.

1 Introduction:

1.1 Kinematic Process 1 (KP1): Free Falling Flower:

In this experiment we will be utilising a provided video of a free falling object and will run it through a video analysis and modelling tool, **Tracker**, to map the position(s), $p_{y,x}$, of the free falling object (which would henceforth be called 'flower' to distinguish the word 'object' for other analyses) against time. The position we will be utilising primarily is the p_y, which will be the main data set to compare our model towards. The 3 model we will be developing are:

- A Frictionless Free Fall; an assumed perfect motion.
- A Free Fall with Viscous Drag; where the free fall's path is influenced by the friction within the fluid (air)
- A Free Fall with Turbulent Friction; where the swirls and eddies mix layers of fluids together

The accuracy of the model will be calculated by comparing the coefficient of determination, R^2, of each model. The R^2 score is a statistical tool that measures how well the model's prediction match the existing data. Furthermore, theoretical analysis of each model will be made use to better compare each model in the finalising of our data in the discussion.

1.2 Kinematic Process 2 (KP2): Rolling Cylinder on an incline

The main aim of this kinematic process is to utilise the retrieved data to measure the gravitational acceleration,g, on earth. Similar to KP1, we will be using the p_y value as a mean to measure the change in height as the body rolls down, and graph it relative to the velocity squared, $\vec{v_y}^2$. In theory, this would yield a graph where the slope is the gravitational acceleration, $\vec{g}$.

Note: From the sections below, we will first be only exploring KP1. The exploration and analysis of KP2 will begin after the discussion section of KP1

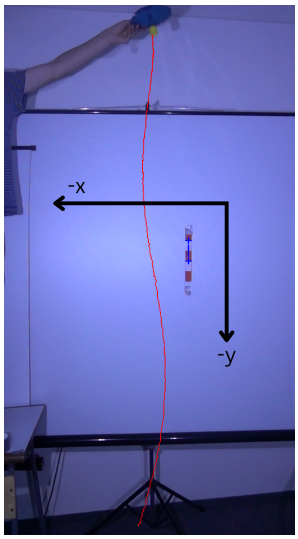
2 Kinematic Process 1

2.1 Theoretical Exploration

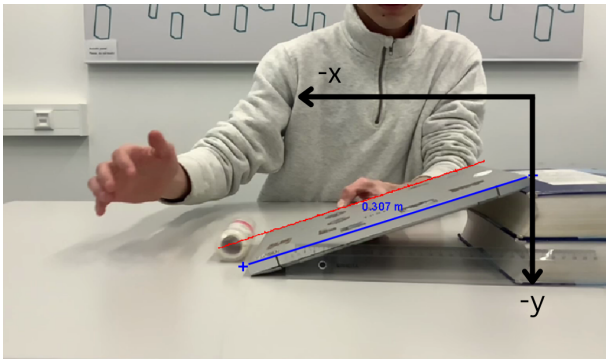
For each model, we will be solving for the p_y in terms of some measurable or time dependent values. Allowing us to graph and compare each p_y to our data set.To construct our models for the free fall, we will start by making use of the most fundamental equation for a kinematic process, the Newton's 2nd Law of Motion:

$$\vec{F} = m\vec{a} \tag{1}$$

This equation is particularly relevant, as it links the force acting on an object to its mass and acceleration. For an object in free fall, the primary force acting is gravity, and so its motion can be described by a constant acceleration downwards. Our axis for this experiment will be as seen in the digram below:



Axis for KP1



Axis for KP2

Image 1: Axis Aligns for both Kinematic Processes

2.1.1 Model 1: A Frictionless Free Fall

In the first model we deal with the idealized case of free fall without any type of friction. In this case the flower's motion can be described by straightforward kinematic equations. Under this assumption, the only force acting on the object is gravity, leading to a constant acceleration, g, which is approximately 9.81 $\frac{m}{s^2}$ near the Earthâs surface. The position p_y of the object as a function of time t is given by:

$$p_y(t) = p_{y0} + v_{y0}t + gt^2 \tag{2}$$

where p_{y0} is the initial position, and v_{y0} is the initial velocity. For our system, we will be taking the v_{y0} and p_{y0} as zero, simplifying equation (2) to:

$$p_y(t) = gt^2 \tag{3}$$

This model assumes that no other forces (such as air resistance) act on the object, resulting in a constant acceleration due to gravity alone. The position of the object is then only represented as equation (3). We will be taking this equation as our position equation.

2.1.2 Model 2: A Free Fall with Viscous Drag

However, different types of friction are factors in the real world that affect the motion of falling objects, especially at higher speeds or with larger surface areas. This resistance opposes the motion, reducing the acceleration, leading to the Viscous Drag Model. In the second model we look at a free fall of an object under the conditions of laminar friction. Laminar friction or viscous drag is used when the flow is smooth and orderly, typically at low velocities. It is proportional to the velocity:

$$\vec{F} = -\alpha m\vec{v} \tag{4}$$

Here α is some proportionality constant, which namely is related to some viscous properties of the fluid (which is air in our care) and the shape of our Flower.

Following through with equation (4), we get:

$$m\vec{a} = -mg - \alpha m\vec{v} \tag{5}$$

which can be written as $\frac{dv_y}{dt} = -g - \alpha\vec{v_y}$, and integrating, we get

$$\vec{v_y} = -v_G + (v_G + v_0)\exp(-\alpha t) \tag{6}$$

We need to integrate again to get the function of the position at time t under the condition that for $t >> \alpha^{-1}$ with $v_G = \frac{g}{\alpha}$ and v_0 set to zero. With this we get:

$$p_y(t) = -v_G t + \frac{v_G}{\alpha}(1 - \exp(-\alpha t)) \tag{7}$$

Where now equation (7) is our equation of position taking into account laminar friction in the fluid. The proportionality constant, α, has the dimensions of one over seconds, $\frac{1}{s}$, as such α $\frac{1}{s}$. Derivation from (Ziese)

2.1.3 Model 3: A Free Fall with Turbulent Friction

The third model of free fall takes the turbulent friction into account. It is also called quadratic drag, due to it's characteristic of being proportional to the square of the velocity. This friction appears, in contrast to laminar friction, in chaotic flows and irregular patterns, mostly at high velocities.

$$\vec{F} = -\beta m \vec{v^2} \tag{8}$$

Where β is another proportionality constant that depends on the properties of our fluid and Flower. Equating with our Newton's 2nd Law, $ma = -mg\beta v^2 (v < 0)$, we get an Ordinary Differential Equation which can be solved using separation of variables, yielding:

$$\vec{v} = v_t \tanh\left(\operatorname{arctanh}\left(\frac{v_0}{v_t}\right) - \frac{g_t}{v_t}\right) \tag{9}$$

,where $v_t = \sqrt{\frac{g}{\beta}}$ is the terminal velocity (the highest velocity the flower will reach). By setting v_{y0} to zero and integrating (9) , we get:

$$p_y(t) = \frac{-1}{\beta} \ln\left(\cosh\left(\frac{gt}{v_t}\right)\right) \tag{10}$$

The proportionality constant, β, has the dimensions of one over meters, $\frac{1}{m}$, as such β $\frac{1}{m}$. Derivation from (Ziese)

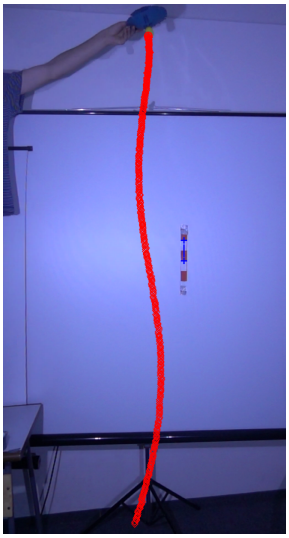
2.2 Hypothesis

We can from the start already make the assumption that Model 1, without friction, will be the least accurate model as it models a perfect environment for energy conservation. Such an assumption is not consistent with real life conditions of energy lose, external forces and minor errors due to factors which are hard to control or eradicate (such as perfect release of the Flower, absence of a fluid surrounding the falling Flower, etc...). However, from the two models which take into account some kind of drag force, we believe the Turbulent model will be the most accurate. As laminar friction is only viable for when the velocity is small, the turbulent model takes into account a more consistent drag influence. The laminar model will most likely be only viable at the start of the experiment. While, as the velocity increases as the time progress and possible reach its terminal velocity, the turbulent model will be more consistent to the increasing velocity.

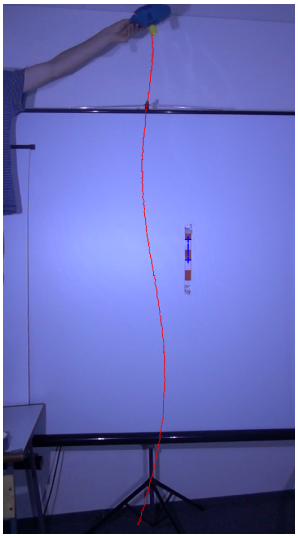
2.3 Results

2.3.1 Data

As previously mentioned, using the software Tracker, with our starting frame being 25, we mapped 526 points along the video. **Figure 1** below present all the mapped points in *(a)* and the general line joining them in *(b)*.



(a) All the Points Mapped



(b) The General Line Connecting the Points

Image 2: Two Images for KP1 side by side

While the positions, p_x and p_y, were mapped over time, the following figure shows the positions $p_{x,y}$ (in meters) plotted against time (in seconds):

```
#the code for plotting is ignored to avoid clutter and the plt method is
↪  standard and understood to already be known.
data = r"data exported.csv"
data_evaluated = pd.read_csv(data)
data_evaluated.columns = ['t', 'y', 'x']
t = data_evaluated.iloc[:, 0]   #time
y = data_evaluated.iloc[:, 1]   #y-axis
x = data_evaluated.iloc[:, 2]   #x-axis
```

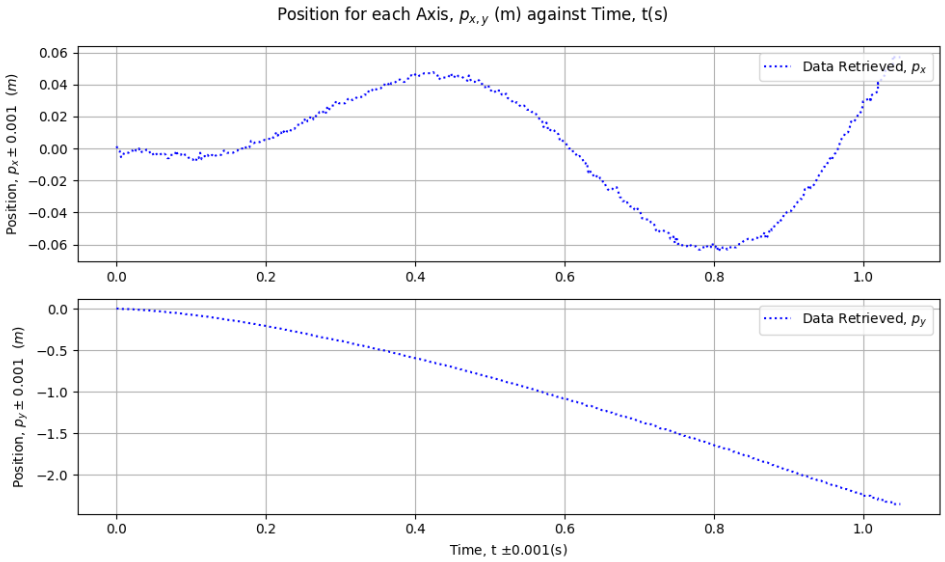


Figure 1: Position, $p_{x,y}$ (m), against Time, t (s)

Running our models using Python's SciPy package's 'curve_fit()' function, the following sections analyse each model against the data and compare their respective R^2 score.

Model 1:

```
g = 9.81
v_0 = 0
y_frictionless_fit = []

for i in range(len(t)):
    y_frictionless = -g*t[i]**2
    y_frictionless_fit.append(y_frictionless)
r2_frictionless = r2_score(y, y_frictionless_fit)
```

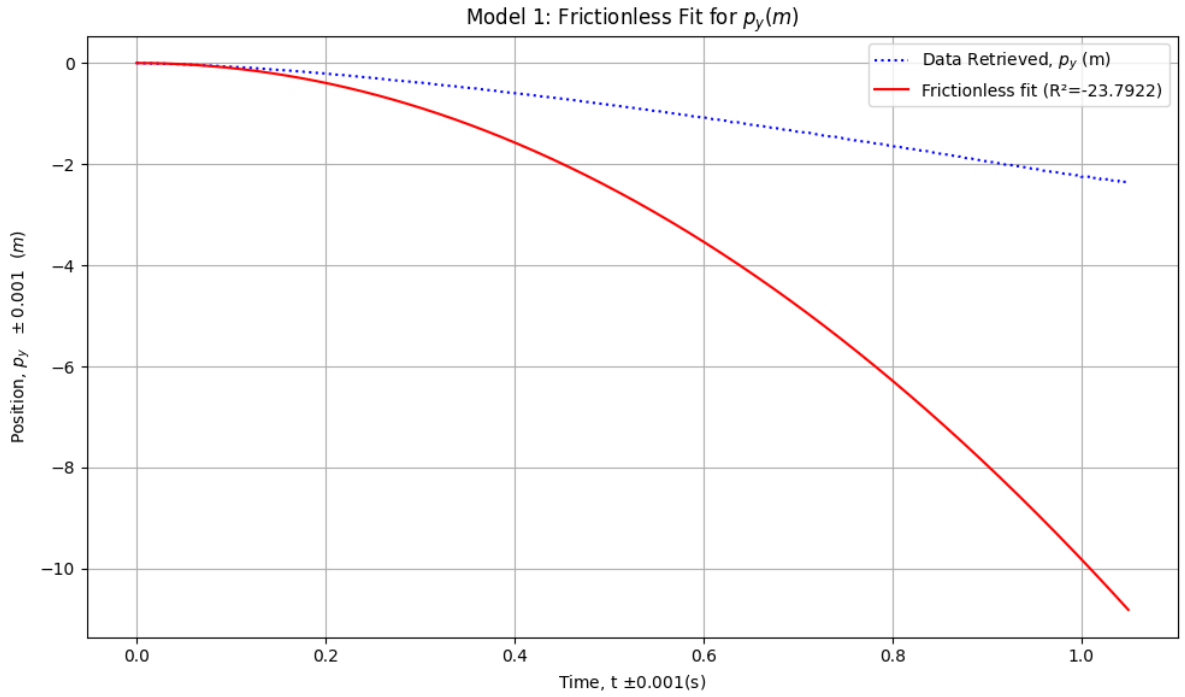


Figure 2: Data Fitting Model 1: Frictionless Free Fall; Position, $p_{x,y}$ (m), against Time, t (s)

Frictionless Fit R^2: [-23.792229851507656]

Model 1 is the only model that was derived without using the function 'curve_fit()', as there are no proportional values to find the best fit values for. Either way, as hypothesised earlier, Model 1 is highly inaccurate, with deviation from the data set starting at already around 0.08 seconds. Meaning, it is close to the data set only for 7.6% of the whole time length of KP1 (1.05 seconds). We can notice how near the end of the time length, the absolute magnitude in the difference between the data set real p_y value and the value from the frictionless data set was $|-8.260496(m)| \approx 8.2(m)$. Which is larger then the distance the Flower was let go at. The R^2 value is in the negatives, which only happens when the model does not follow the data trend, making it **imprecise** but accurate. This model can right away be deemed impractical and ineffective. More details in the Discussion.

Model 2:

```
def laminar_model(t, alpha):
    v_g = g / alpha
    return -1*v_g*t + ((v_g + v_0)/alpha)*(1-np.exp(-alpha*t))
laminar, _ = curve_fit(laminar_model, t, y)
y_laminar_fit = laminar_model(t, *laminar)
r2_laminar = r2_score(y, y_laminar_fit)
```

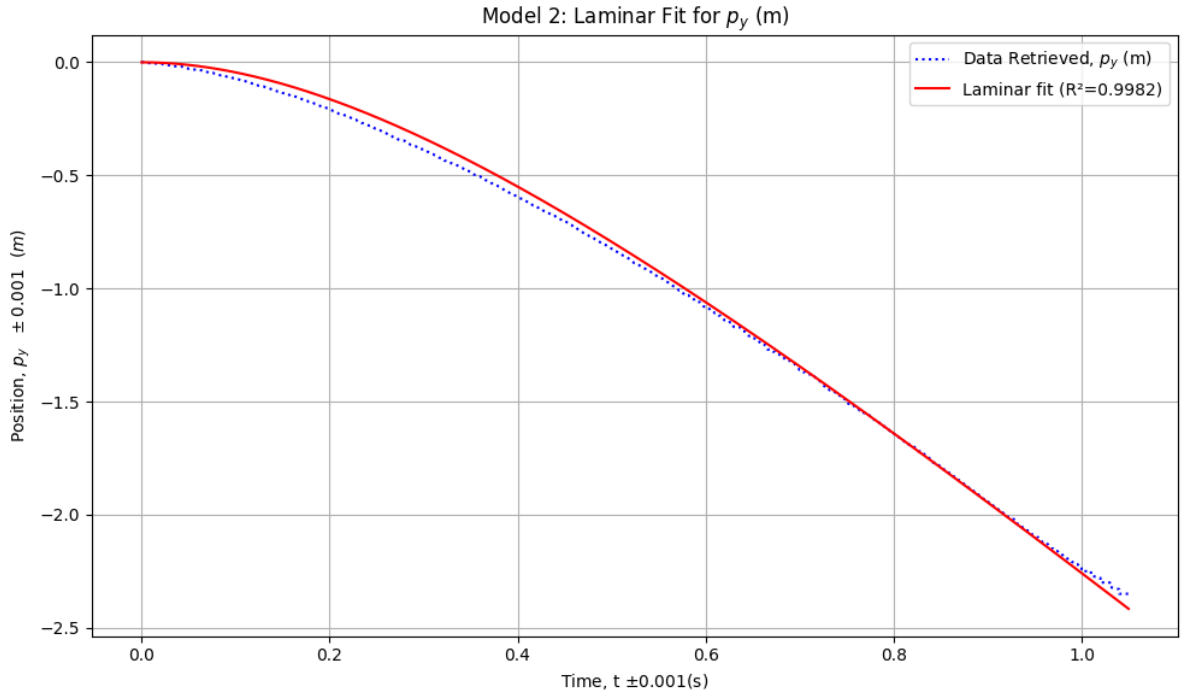


Figure 3: Data Fitting Model 2: Laminar Free Fall; Position, $p_{x,y}$ (m), against Time, t (s)

Laminar Fit Parameters α:2.94805897$\frac{1}{s}$
Laminar Fit R^2: 0.9981752827075149
Standard Error of α: 0.00602233 $\approx$ 0.006

Figure 3 easily shows a more accurate model. The model line produced by the curve_fit algorithm runs nearly smoothly over the data set, with most of the deviations (even though the deviations are very minor relative to the grid size) are right after the start and near the end. The largest peace-wise difference between the Real, p_y, Data set and the Laminar Data Set being only 0.0663271498902227 (m) $\approx$ 0.066 (m). With a R^2 score of 0.9981752827075149 $\approx$ 0.998, the value is nearly 1. A value of one in R^2 means a perfect data fit of the model. Making the data set **accurate** and **precise**

The value for α, from equation (7), was found to be 2.94805897 $\approx$ 2.95 $\frac{1}{s}$, meaning the relative correlation between air viscosity and the shape of the Flower is 2.95 $\frac{1}{s}$. More details in the Discussion

Model 3:

```
def turbalent_model(t, beta):
    v_t = np.sqrt(g/beta)
    return -(1/beta)*np.log(np.cosh(g*t/v_t))
turbalent, _ = curve_fit(turbalent_model, t, y)
y_turbalent_fit = turbalent_model(t, *turbalent)
r2_turbalent = r2_score(y, y_turbalent_fit)
```

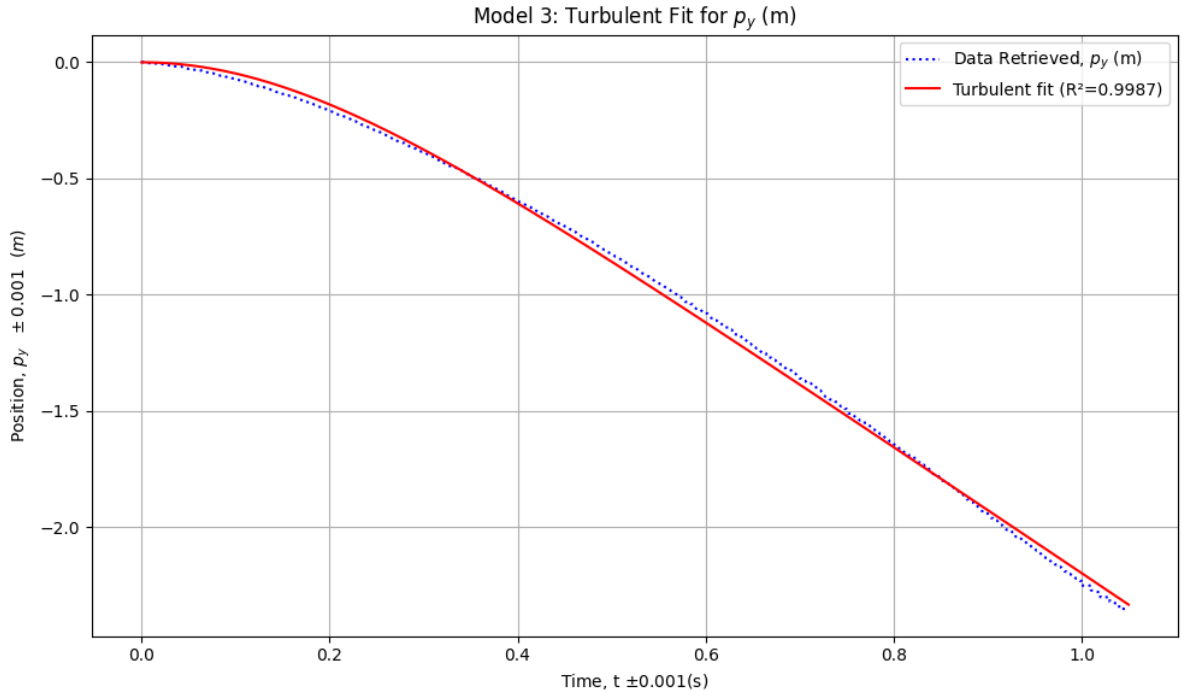


Figure 4: Data Fitting Model 3: Turbulent Free Fall; Position, $p_{x,y}$ (m), against Time, t (s)

Turbulent Fit Parameters β: $1.32827536\frac{1}{s}$
Turbulent Fit R^2: 0.9986828095156135
Standard Error of β: $0.00359422 \approx 0.004$

Figure 4 also provides an accurate data fitting model, with the fitting line being covering nearly all the data points on the graph. Even though the line deviates at the start of the data set, the Turbulent Data fitting is more accurate for after the start. With the graph covering nearly all the data points after the around midway through. The largest peace-wise difference between the Real, p_y, Data set and the Turbulent Data Set being only 0.05365823603765474 (m) $\approx$ 0.054 (m). With a R^2 score of 0.9986828095156135 $\approx$ 0.999, the value is nearly 1. Making the data set **accurate** and **precise**.

The value for β, from equation (10), was found to be $1.32827536 \approx 1.33\ \frac{1}{s}$, meaning the relative correlation between the drag coefficient, surface area and the density of the fluid is $1.33\ \frac{1}{s}$. More details in the Discussion.

2.3.2 Discussion

Note: The code for the maximum difference and the Mean Absolute Error is in the Appendix. Omitted from here to avoid clutter

Reason for Deviations:

For Model 1, the obvious large deviation comes from the absence of friction in the derivation of the formula. The effects of Turbulence and Laminar Friction cannot be under stated, as seen earlier, the absolute magnitude of **8.2** m in difference near the end is largely due to an assumption that the Flower does not reach terminal velocity but keeps on accelerating downwards. The equation utilised for Model 1, equation (3), explicitly has no variable or coefficient which takes into account any slowing down of the Flower.

For Model 2, it is a much a more accurate model. The inclusion of Laminar Friction takes into account the drag force exerted by the fluid and brings in a coefficient which limits the growth of out line, to some degree. The existing deviations, namely between around **0.05** seconds to **0.75** seconds, come from possible initial condition sensitivity. Values such as initial velocity and the force exerted by the person who let go the Flower were all ignored. These factors, even if small, will lead to some difference in the initial behaviour of the system. We furthermore had not taken into account non steady laminar flow and the influence of turbulence on the motion. However, as the object reaches its terminal velocity and stabilises relative to the fluid it is in, as here for the behaviour beyond 0.75 seconds, the model starts to better reflect the real data. However, even with these deviations, the maximum difference between the two data sets were **0.066** m, a much better difference compared to Model 1. The Mean Absolute Error for these two data sets made out to be 0.02544354684383286 (m) $\approx$ **0.025**(m).

For model 3, the fitting line is nearly identical to the one of Model 2. The accuracy of the model is deviates away from the Real Data Set at the start but continuos on to become more precise as the KP1 develops further along time. The reasons for such behaviour would be of similar reasons as stated above, 'not taking into account laminar flow, not precisely negligible initial conditions, and assumption that turbulence starts affecting directly from the start'. Yet, the maximum difference between the two sets is of the best result, with only **0.054 m**. The Mean Absolute Error for these two data sets made out to be 0.02360733087223999 (m) $\approx$ **0.024**(m)

Moreover, as an additional information, we can calculate the terminal velocity, v_t, for our system. As $v_t = \sqrt{\frac{g}{\beta}}$, and considering how β is known because of our most accurate model, we can calculate v_t to its highest possible degree of precision. With this, we get $v_t = 2.71763073472 \approx 2.72\frac{m}{s}$. A value which makes sense in the context of how the Flower is a small object which reached the ground from a height of around 2.5 m within 1.05 seconds.

Most Accurate Model:

From this, we can conclude that **Model 3** is the most accurate model from the 3 we have looked into.

2.3.3 Error Analysis

For KP1, as our data is retrieved utilising a software, the uncertainty in our calculations will be the smallest significant value the software provides. With this in mind, our uncertainty in the Real Data Set for p_y was (± 0.001) ãs also seen in the graphs. While the uncertainty in time, t, will be (± 0.001). Using this same uncertainty for t_0, we can further analyse the propagation of zero of time along the whole data set.

Quantifying the influence of the uncertainty of the zero of time:

The quantifying of the influence of t_0 will be done using the standard uncertainty equation. Where for some time-dependent variable, $p_y(t)$, any uncertainty, ∂t_0, in the zero of time propagates through the whole data set following; (Taylor)

$$\partial p_y(t) \approx \left|\frac{\partial p_y(t)}{\partial t}\right| \partial t_0 \tag{11}$$

where, the partial derivative for each of our models will be:

Frictionless:

$$\partial p_{yf}(t) = -2gt \tag{12}$$

Laminar:

$$\partial p_{yl}(t) = -v_g + -v_g \exp(-\alpha t) \tag{13}$$

Turbulent:

$$\partial p_{yt}(t) = -\frac{g}{\beta v_t} \tanh\left(\frac{gt}{v_t}\right) \tag{14}$$

Where $v_g = \frac{g}{\alpha}$ and $v_t = \sqrt{\frac{g}{\beta}}$. Using these equations, we get the following data:

Models	Mean Error Propagation	Largest Error (Max)
Frictionless	$\pm$ 0.0103	$\pm$ 0.0206
Laminar	$\pm$ 0.0023	$\pm$ 0.0032
Turbulent	$\pm$ 0.0022	$\pm$ 0.0027

Table 1: Error Propagation; Mean and Maximum Values

```
def laminar_model_derivative(t, alpha):
    v_g = g / alpha
    return -v_g + (v_g) * np.exp(-alpha * t)

def turbalent_model_derivative(t, beta):
    v_t = np.sqrt(g / beta)
    return - (g / (beta * v_t)) * np.tanh(g * t / v_t)

def frictionless_model_derivative(t):
```

```
    return -2*g*t

delta_t0 = 0.001

dy_laminar = np.abs(laminar_model_derivative(t, *laminar)) * delta_t0
dy_turbulent = np.abs(turbalent_model_derivative(t, *turbalent)) * delta_t0
dy_fritionless = np.abs(frictionless_model_derivative(t)) * delta_t0

mean_error_laminar = np.mean(dy_laminar)
mean_error_turbulent = np.mean(dy_turbulent)
mean_error_frictionless = np.mean(dy_fritionless)

print("Error Propagation Frictionless: ", mean_error_frictionless, "
↪  Lamniar: ",mean_error_laminar, " Turbulent", mean_error_turbulent)

```

3 Kinematic Process 2: Rolling Cylinder on an incline

3.1 Theoretical Exploration

3.1.1 Energy conservation in Rolling motion on inclined planes

For analysing KP2, we will need to derive an equation that relates change in height to the squared velocity, $v_y^2 \frac{m}{s}$ - while taking into account energy conservation. Our system is made up of a rolling cylinder along a flat surface at an incline. Initially, from rest at the height h, all energy is in the form of gravitational potential energy. At the point where it rolls down through the incline (in our case as it reaches the table), all of the energy is converted to kinetic energy (made up of translational or rotational components). Assuming we treat the table as a position of zero height, and as such zero gravitational potential. Neglecting rolling friction, or any friction due to drag forces in the fluid (air), we get:

$$mgh = \frac{1}{2}m\vec{v^2} + \frac{1}{2}I\vec{\omega^2} \tag{15}$$

The moment of inertia, I, equals $\frac{1}{2}m(R_2^2 + R_1^2)$ (where the R_1^2 is the outer radius and R_1^2 is the inner radius) as our cylinder is made up of two radiuses, with a hole in the middle. In the same manner, our $\vec{\omega^2}$ will be $\frac{\vec{v_y^2}}{R_1^2}$. Furthermore taking into account how as our cylinder and the rolling surface are made up of hard materials, any deformation from rolling (as such assuming our rolling radius is zero, $R' = 0$) can be assumed to be negligible. Now plugging in I and solving for h (which corresponds to the change in height), and we get:

$$gh = \vec{v_y^2}(\frac{1}{2} + \frac{1}{2}\frac{R_1^2 + R_2^2}{R_1^2}) \tag{16}$$

The gradient of a graph of v_y^2 against the change in height, h, (taking into account the constant proportionality of $(\frac{1}{2} + \frac{1}{2}\frac{R_1^2+R_2^2}{R_1^2})$), we should get the gravitational acceleration g

3.2 Physical Exploration

3.2.1 Materials

- 2 books, each each book $\approx$ 8.16 cm $\pm$ 0.005 cm high.
- Laptop
- A Pack of Adhesive Tape Rolls

3.2.2 Set up

The two books are placed on top of each other on a table and the upper edge of the laptop is leaned on the tower of books so that it creates an incline plane. The image of the set up can be seen in **Section 3.3.1** below.

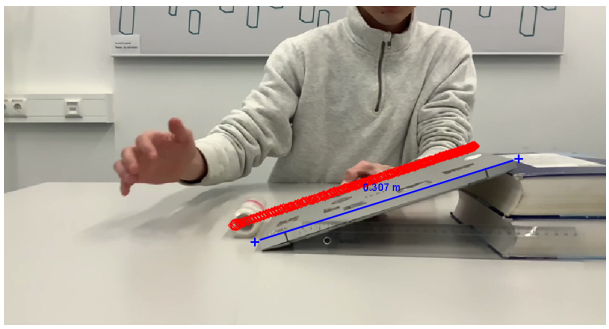
3.2.3 Methodology

At the beginning of the experiment the pack of adhesive tape rolls is held at rest at the top of the inclined plane. Then we stared filming the video. When the adhesive tape roll is released it rolls down the inclined plane and continues rolling but slowing down on the horizontal table. After recording the video we used the software **Da Vinci Records** to slow down the video in order to analyse it. By using the software **Tracker** we were able to map the position and velocity of the adhesive tape roll along the time length of **1.75 seconds** $\pm$0.001.

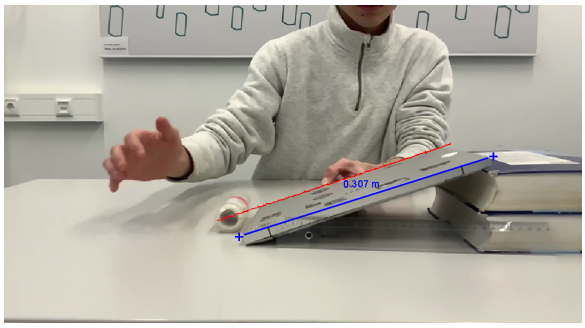
3.3 Results

3.3.1 Data and Analysis

From the mentioned software, Tracker, with our starting frame being 249 and ending frame being 664, we mapped 412 points along the video. The full video can be found in the attached zipped file as supplementary file. The images below present all the mapped points *(a)* and the general line joining them all in *(b)*



(a) All the Points Mapped



(b) The General Line Connecting the Points

Image 3: Two Images for KP2 side by side.

While the positions, p_x and p_y, and the velocity against p_y, $\vec{v_y}$, were mapped over time, the following figures show Position, $p_{x,y}$(m) and Velocity, $\vec{v_y}\frac{m}{s^2}$, against Time, t(s):

```
data_rolling = r"data exported;velocityrolling.csv"
data_evaluated_rolling = pd.read_csv(data_rolling)
data_evaluated_rolling.columns = ['t', 'x', 'y', 'vy']
t_r = data_evaluated_rolling.iloc[:, 0]  #time
x_r = data_evaluated_rolling.iloc[:, 1]  #x-axis
y_r = data_evaluated_rolling.iloc[:, 2]  #y-axis
vy_r = data_evaluated_rolling.iloc[:, 3]  #y-axis
```

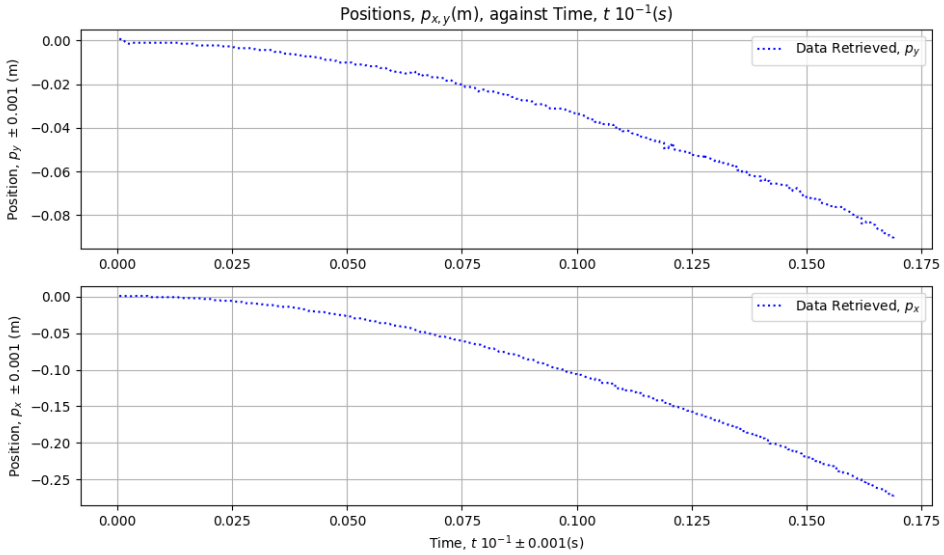


Figure 5: Position, $p_{x,y}$ (m), against Time, t (s)

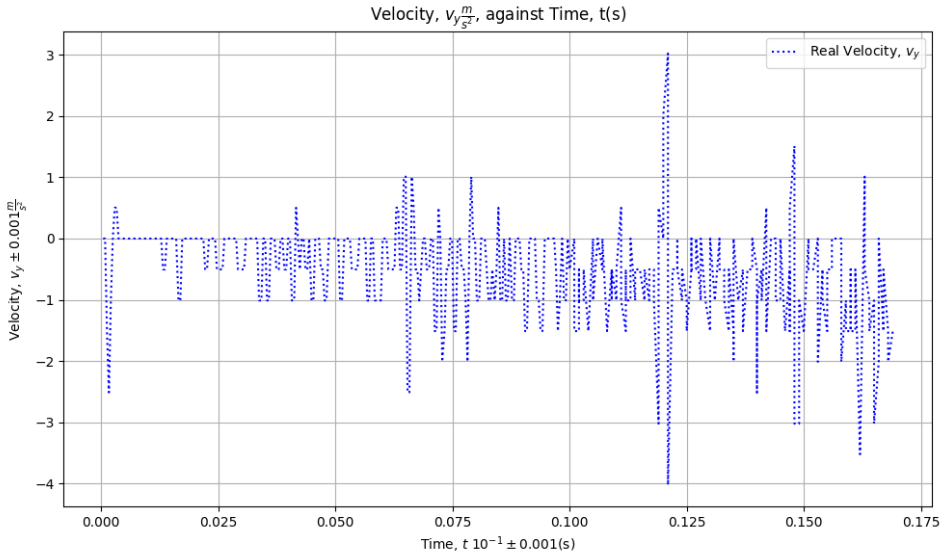


Figure 6: Velocity, $v_y \frac{m}{s}$, against Time, t (s)

An issue you might have noticed right away from the Velocity, $\vec{v_y} \frac{m}{s}$, against Time, t (s) graph is how the velocity is not consistent, and if used such data for graphing velocity squared against the change in height, we would be getting a very disorganised and non-understandable data set. We tried remeasuring the data values 5 times but this irregularity of the velocity data set persisted.

Henceforth, we implemented a best fitting curve to find a line that best finds the overall pattern of the movement. However, to additionally verify that such line would be properly representing the velocity over time, using the p_y(m) position values, we found another velocity curve by taking the derivative of a best fit line for **Figure 5** graph. These two different velocity graphs are presented in the following figures;

```
v_empty = []

def rolling_fitting(t_r, a, b, c):
    return a*t_r**2 + b*t_r + c

rolling, _ = curve_fit(rolling_fitting, t_r, y_r)
rolling_fitting_values = rolling_fitting(t_r, *rolling)
print(rolling)

def function(t_r): #values found using print(rolling)
    return ((-2.71487733)*(t_r)**2 + (-6.95859510e-02)*t_r +
    ↪  1.63049235e-04)

def veloccity(u):
     return derivative(function, u)

def velocity_fitting(t_r, d, e):
    return d*t_r + e

rolling_vel, _ = curve_fit(velocity_fitting, t_r, vy_r)
rolling_vel_fitting = velocity_fitting(t_r, *rolling_vel)
r2_rolling_vel = r2_score(y_r, rolling_vel_fitting)
print("The $R^2$ value is: ", r2_rolling_vel)
```

The R^2 value is: -463.0215107469761

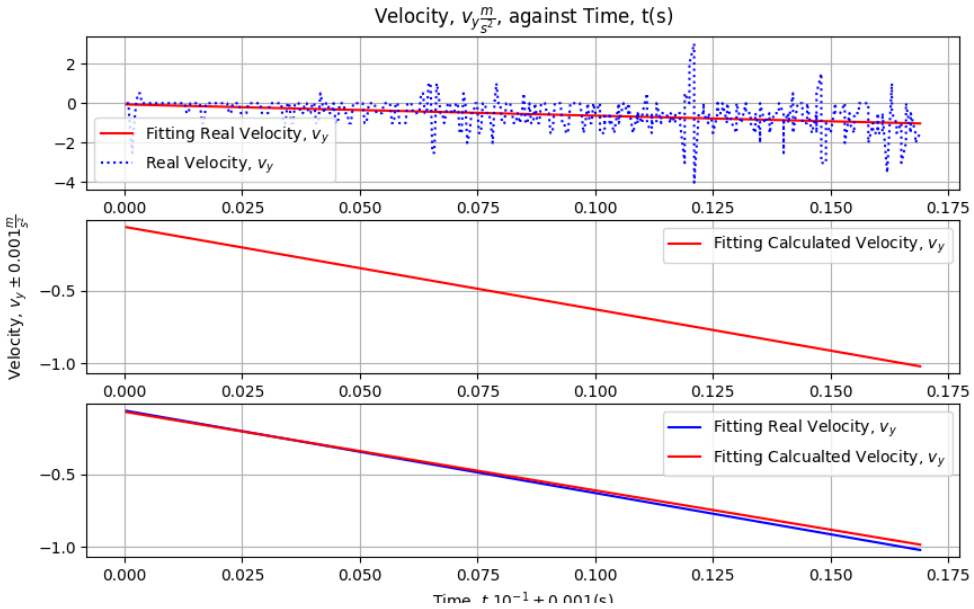


Figure 7: Velocity Fitting Values, $\vec{v_y} \frac{m}{s}$, against Time, t (s)

The graph of the best fit line for p_y against t can be found in the appendix, removed here to avoid clutter. The R^2 value for that graph is 0.9993192636977063 ≈ 0.999

Even though the R^2 value for our best fit line for the Velocity Data Set is -463.0215107469761 ≈ -463 and is heavily unrepresentative of the data set, these irregularities are due to the irregularity of the data set itself. Meaning, it is expected for the R^2 value to be this off. However, the found best fit line describes the behaviour of the data set (i.e the downward decrease of the value).

As seen in Figure 7, the best fit line found for the Velocity Data Set (the *'Fitting Real Velocity'*) is nearly the same with the velocity curve found from calculating the derivative of our previously Figure 5(a) p_y(m) against t(s) graph (the *'Fitting Calculated Velocity'*). The maximum difference between the two curves being 0.03708253061486311 ≈ 0.037$\frac{m}{s}$. Meaning, we can use the the *'Fitting Real Velocity'* data set for our calculation of the gravitational acceleration.

The following figure presents the final curve, the velocity squared, $\vec{v_y^2}\frac{m^2}{s^4}$ against the change in height, $h-p_y(m)$. Following equation *(16)*, we have graphed taking into account the proportionality $(\frac{1}{2}+\frac{1}{2}(R_1^2+R_2^2)(\frac{1}{R_1^2}))$.

```
height_y = (1/2)*veloccity(t_r)**2*(1+((0.00055625)*(1/0.02**2)*(1/2)))
slope, intercept, r_value, p_value, std_err = stats.linregress(y_r,
↪  height_y)
print("The Slope is: ", slope)
```

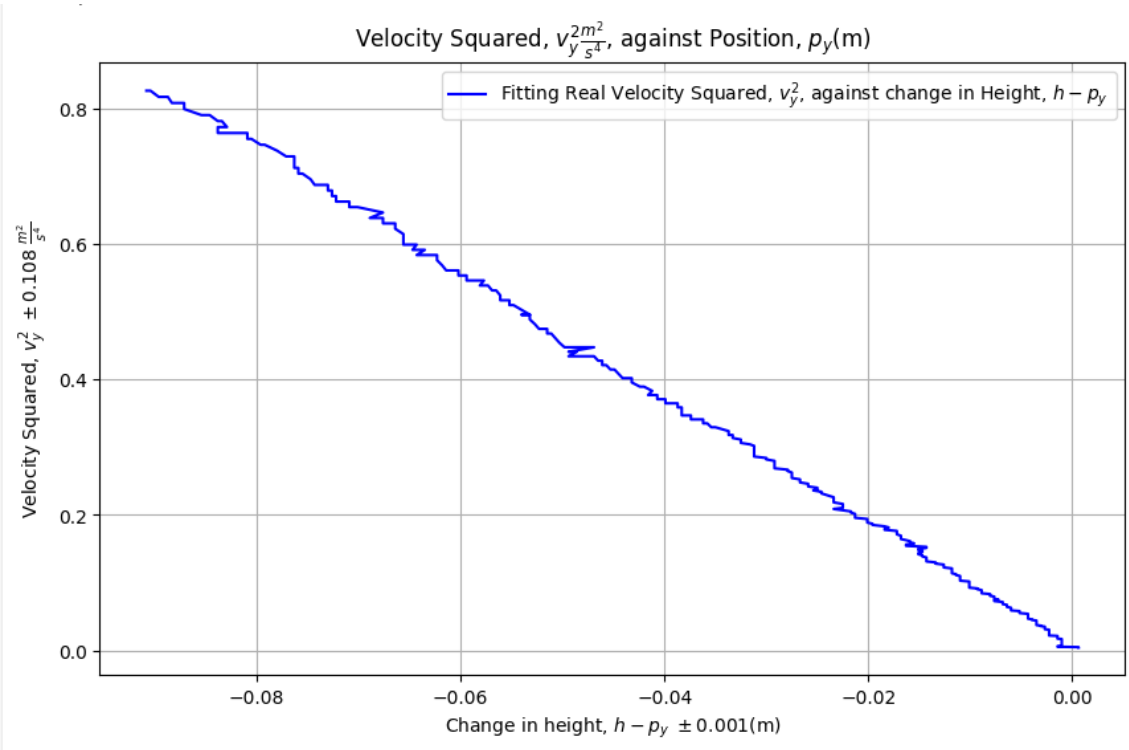


Figure 8: Velocity Squared, $\vec{v_y^2}\frac{m^2}{s^4}$, against Change in Height, $h-p_ym$

The Slope is: -9.198864678036848

Where the slope of this data set came out to be -9.198864678036848 ≈ -9.20 $\frac{m}{s^2}$, which is an error margin of **-6.64%** from the literary value of -9.81 $\frac{m}{s^2}$. Note, the uncertainty of ±0.108 for the velocity is derived from taking into account the constant proportionability. More details in the Discussion.

3.3.2 Discussion

Reason for the error margin of -6.64%:

The clear reason for the error margin, the deviation from the expected -9.81$\frac{m}{s^2}$ is due how our method assumes there is no energy loss over the time length and that the rolling of our cylinder is ideal with no slipping or sliding. Additionally, on a smaller scale, inelastic deformation causes small lose of energy over time. Even though we assumed there is no such deformation (as our cylinder and the surface are both hard surfaces), there could exist very minor deformation on a molecular level. Both the point of the bottom of the cylinder and the ground keep deforming and returning back. The surface of the laptop is further not smooth, as assumed, being a little more rough. Causing small bumping of the cylinder with the surface, further leading to the lose of energy. These factors allow the final magnitude of the acceleration to be less than the literary value.

Moreover, we did not have an ideal velocity data set. As previously mentioned, the tracking software did not provide a data set which is consistent and as such we had to extrapolate a pattern which *describes* the behaviour. And describing is not the same as the actual pattern of a Data Set. A better data set would have lead to a better understanding of the velocities, as such, a more accurate result.

3.3.3 Error Analysis

The uncertainties in our measurements for p_y, similar to KP1, come from the smallest significant value the software provides. We will be multiplying this with the uncertainty found for the additional constant value, as seen in equation (16), (where the uncertainty in R values would be the $\frac{smallest\ measurable\ value}{2}$) yielding our uncertainty in the v_y (additionally with the uncertainty in the v_y^2):

Variable	Uncertainty
p_y	± 0.001
v_y	± 0.0538
v_y^2	± 0.108

Table 2: Uncertainty Calculations for p_y, v_y and v_y^2

4 Conclusion

In conclusion, the first part of this report provided a satisfactory result. With KP1, we were successfully able to implement 3 different models to our existing data and were able to find a model that fits the data pattern, with an R^2 score accuracy of 0.999. Our hypothesis for before processing KP1 was accurate, and there were clear reasons why our model deviated in certain parts of the data fitting. The existing error margins, due to

t_0 or other external factors, were minimal and we were able to present possible future solutions for mitigating these errors. Additional research would combing Model 2 with Model 3, to take into account the initial effects of laminar friction (when the velocity is slow) and also the consistent turbulent frictional effects that affect our Flower throughout the time length.

For KP2, we were able to produce a kinematic process that produced a sensible position against time graph. Even though our needed velocity was not suitable to use, we were able to extrapolate the needed behaviour using additional curve_fitting() algorithms. Resulting a error margin of -6.64%. The relatively small deviation from the expected value of gravitational acceleration were largely due to the assumption that energy is conserved. The acceleration we got in the end was more accurate to a real-life conditions.

A Appendix

Codes for the Maximum Difference and the Mean Absolute Error

```
def largest_pairwise_difference(data_1, data_2):
    differences = [abs(a - b) for a, b in zip(data_1, data_2)]
    max_difference = max(differences)
    return max_difference
resultA2 = largest_pairwise_difference(y_laminar_fit, y)
resultA3 = largest_pairwise_difference(y_turbalent_fit, y)
def quantify_data_similarity(data_1, data_2):
    arr1 = np.array(data_1)
    arr2 = np.array(data_2)
    mae = np.mean(np.abs(arr1 - arr2))
    return mae
resultB2 = quantify_data_similarity(y_laminar_fit, y)
resultB3 = quantify_data_similarity(y_turbalent_fit, y)

print("Max Difference, Laminar: ", resultA2, "Laminar Mean Absolute Error: ", resultB2)
print("Max Difference, Turbulent: ", resultA3, "Turbulent Mean Absolute Error: ", resultB3)
```

Graph for the best line fitting for the calcualted velocity in **Section 3.3.1**:

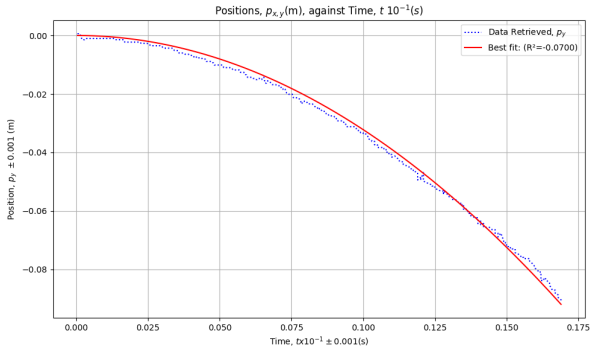


Figure 9: Position Fitting Values, $p_y(m)$, against Time, t (s)

References

Michael Ziese. *H01e Pendulum as an accelerated Frame of Reference*, 2024 Taylor,

John R. *An Introduction to Error Analysis: The Study of Uncertainties in Physical Measurements*. University Science Books, 2022.