Lab report 2: modelling displacement

Written by Jonathan Tybirk, s216136

Purpose

In this report, I will attempt to model the displacement x of a vehicle rolling down a ramp with initial speed v and constant acceleration a for duration t.

$$x = f(t, v, a)$$

By letting gravity accelerate an iOLab down a ramp, I can measure displacement, velocity, and acceleration over a time duration in repeated experiments. As time, velocity, and acceleration can be written as factorizations of just length and time, I can then perform dimensionality reduction on the data to make 2-dimensional regression possible and compute the model parameters.

Experimental design

As sketched in figure 1, an iOLab is placed at the high end of an inclined board to roll down while taking measurements on time, position, velocity, and acceleration. The high end of the board is raised to variable height h, as shown in figure 1. To vary conditions, 3 rolldowns are performed with the high end of the board raised to respectively $10\ \text{cm}, 15\ \text{cm}$, and $20\ \text{cm}$. Two measurements are then taking from each run, respectively the first and second duration of 0.25 seconds. This ensures that the model can account for different displacements and initial velocities. The different heights ensures that the model can account for different constant accelerations.

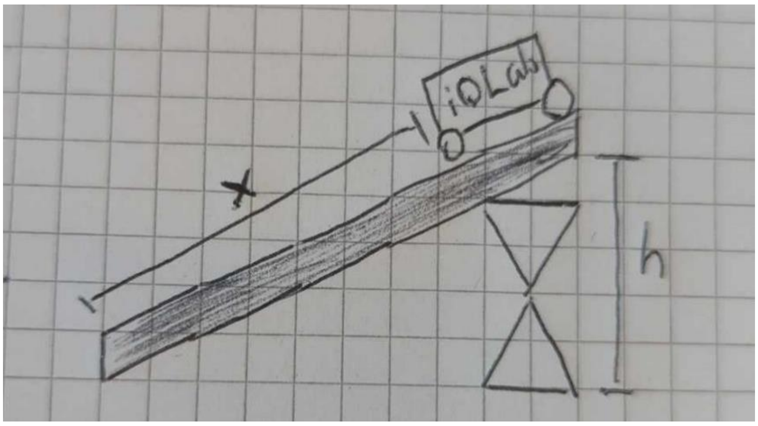

Figure 1: sketch of the experiment setup

```python
h = np.array([10, 10, 15, 15, 20, 20])
x_delta = np.array([
            unc.ufloat(0.045 - 0.002, 0.002), unc.ufloat(0.141 - 0.045, 0.010),
            # h = 15 cm
            unc.ufloat(0.068, 0.002), unc.ufloat(0.227 - 0.068, 0.008),
            # h = 20 cm,
            unc.ufloat(0.101 - 0.002, 0.018), unc.ufloat(0.302 - 0.101, 0.060)])
t_delta = np.array([
            unc.ufloat(0.250,0.030), unc.ufloat(0.250,0.030),
            # h = 15 cm
            unc.ufloat(0.250,0.030), unc.ufloat(0.250,0.030),
            # h = 20 cm
            unc.ufloat(0.250,0.030), unc.ufloat(0.250,0.030)])
v0 = np.array([
      unc.ufloat(0.100,0.020), unc.ufloat(0.260,0.020),
      # h = 15 cm
      unc.ufloat(0.068,0.030), unc.ufloat(0.467,0.030),
      # h = 20 cm,
      unc.ufloat(0.133,0.053), unc.ufloat(0.620,0.080)])
a = np.array([
     unc.ufloat(0.823,0.070), unc.ufloat(0.858,0.099),
     # h = 15 cm
     unc.ufloat(1.690,0.350), unc.ufloat(1.436, 0.094),
     # h = 20 cm
     unc.ufloat(1.932, 0.160), unc.ufloat(1.990, 0.110)])
```

Figure 2: logging of data in Python

Data

v and x were read at timepoints $t = 0$ and $t = 0.250$ respectively from the iOLab-data. No uncertainty was calculated by iOLab for these datapoints, so the value difference between the value at t and the value at $t + 0.030$ was chosen as the uncertainty. 0.030 is the smallest possible timeframe to move in iOLab with the cursor.

a was set to the average acceleration over the periods calculated by iOLab with the standard deviation also calculated by iOLab as the uncertainty.

Finally, no uncertainty was computed for h, as this variable only exists to vary the acceleration by making the slope steeper. The precision of the height does therefore not matter for the experiment outcome and analysis.

Data analysis

With the natural scales $[L] = vt$ and $[T] = t$, the function can be reduced to dimensionally reduced. The variables are reduced by dividing out their units with combinations of the natural scales. This can be done as so:

$$\frac{x}{vt} = f\left(\frac{t}{t}, \frac{v}{\frac{vt}{t}}, \frac{a}{\frac{vt}{t^2}}\right) = f\left(1, 1, \frac{at}{v}\right) = f\left(\frac{at}{v}\right)$$

This gives us the two dimensionless variables $\pi_1 = \frac{at}{v}$ and $\pi_2 = \frac{x}{vt}$ that can be plotted against each other in just two dimensions. Uncertainties for π_1 and π_2 are then calculated with the Uncertainties Python library, which automatically handles error propagation. Plotting these variables together, they look to be correlated by a simple linear relationship. A best linear fit is found with the SciPy Python library and plotted with the datapoints (see figure 3).

This best linear fit is computed to

$$\pi_2 = (0.500 \pm 0.037)\pi_1 + 0.948 \pm 0.115$$

We isolate for x to get that

$$\frac{x}{vt} = A \cdot \frac{at}{v} + B \Leftrightarrow x = A \cdot at^2 + B \cdot vt \Leftrightarrow$$

$$\Leftrightarrow x = (0.500 \pm 0.037)at^2 + (0.948 \pm 0.115)vt$$

where A is the slope, and B is the intercept. x is thus roughly equal to a half times the acceleration times the duration squared plus the initial velocity times the duration. This matches the theoretical formula for displacement in 1D kinematics $x = \frac{1}{2}at^2 + vt$ within the uncertainty range of our regression.

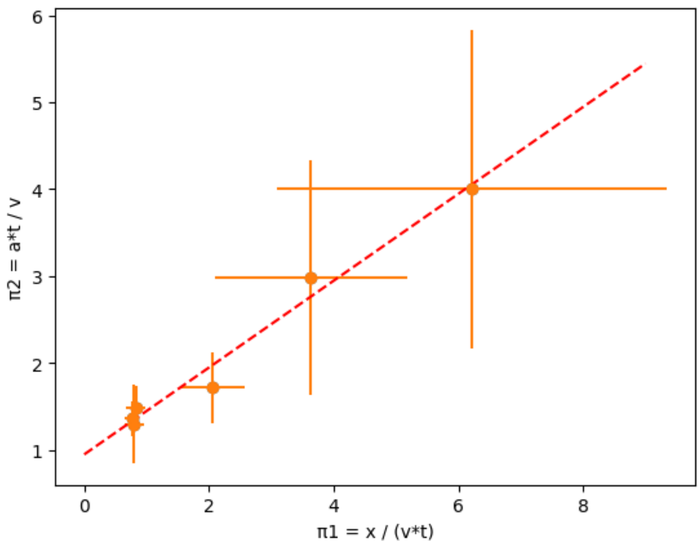

Figure 3: linear regression on the plotted dimensionless variables with error bars

Conclusion

I modelled displacement as a function of acceleration, time, and initial velocity with the relation

$$x = (0.500 \pm 0.037)at^2 + (0.948 \pm 0.115)vt$$

This relationship corresponds well to the theoretical kinematic equation.