A close-up of a logo

Description automatically generated

Modelling Of Software

Intensive Systems

Assignment 1: Modelica

1st Master computer science

2024-2025

Liam Leirs

Robbe Teughels

# Plant Model Creation:

We created a model of a plant that has a trolly with a rope attached to it. The goal of this evaluate both the pendulum and the trolly. More specific, the (angular) speed and the displacement. This can be tuned by the Damping factors *Dp* and *Dc*. For controlling the model, we use the control signal *u* that gives the desired displacement.

We tested the model with two intuitive tests: no displacement, set displacement.  
When we require the cart to stay fixed in place, the entire model stays fixed as we would expect. When we set the control system to a desired location, we see the cart move to the exact location, given some time. The pendulum reacts on the movement of the cart as required. Given these test, we assume that the model is representative to the real object for velocity and displacement.

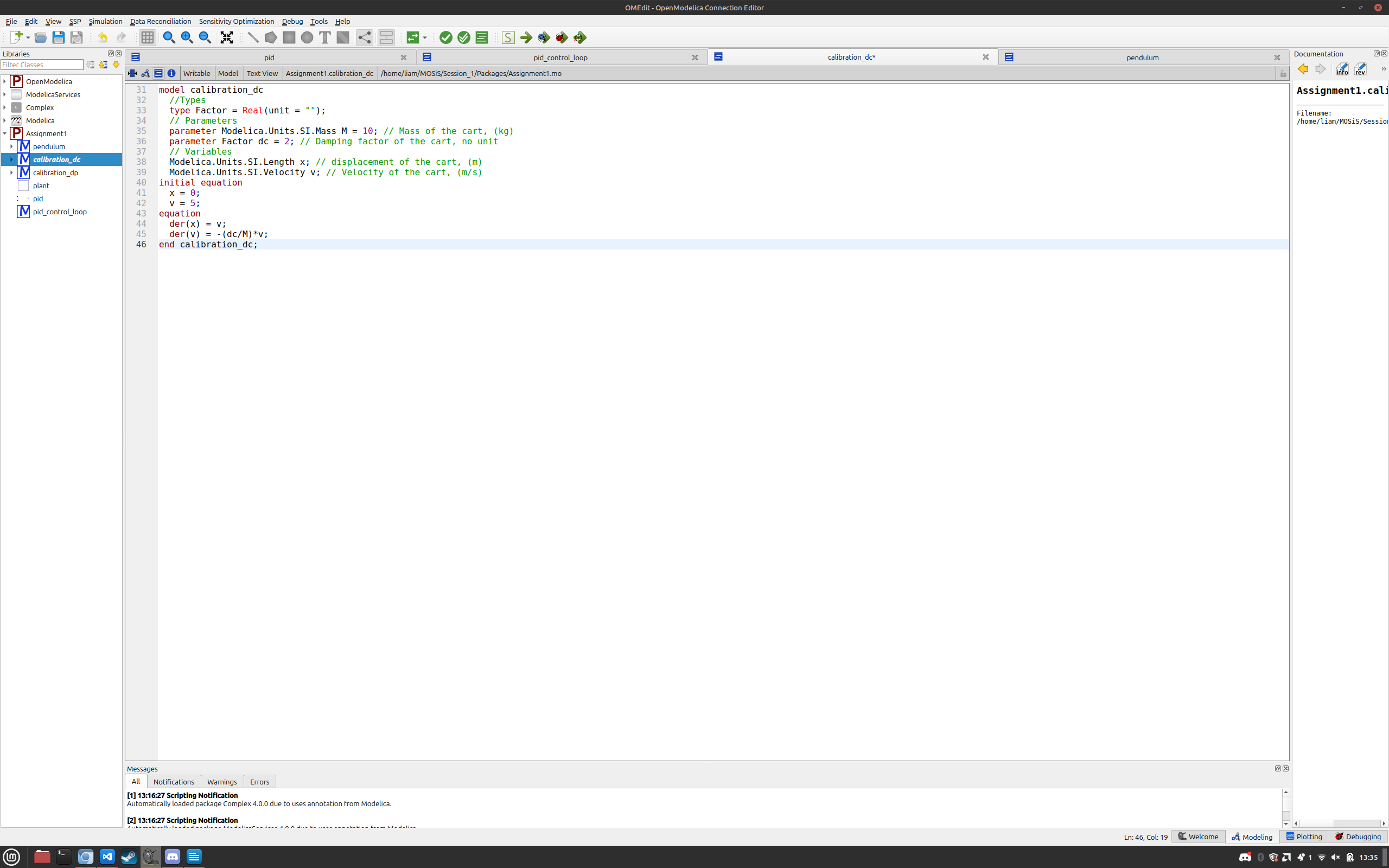


# Plant Model Calibration

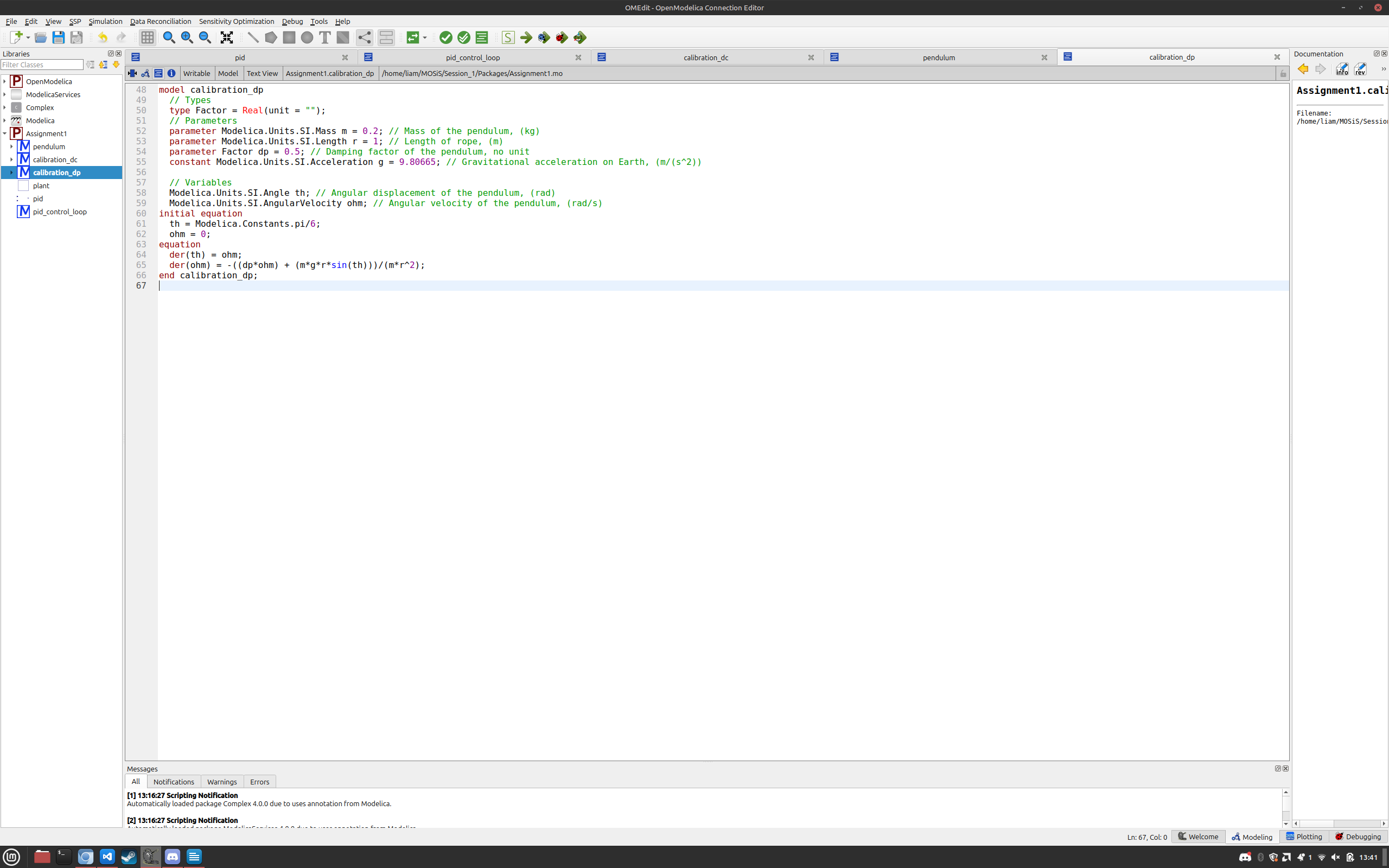
Now that the plant model was created, it still needed to be calibrated (via the damping factors Dp & Dc). To do this, 2 new models had to be created where

the effects of one of these factors is eliminated allowing us to estimate the other one.

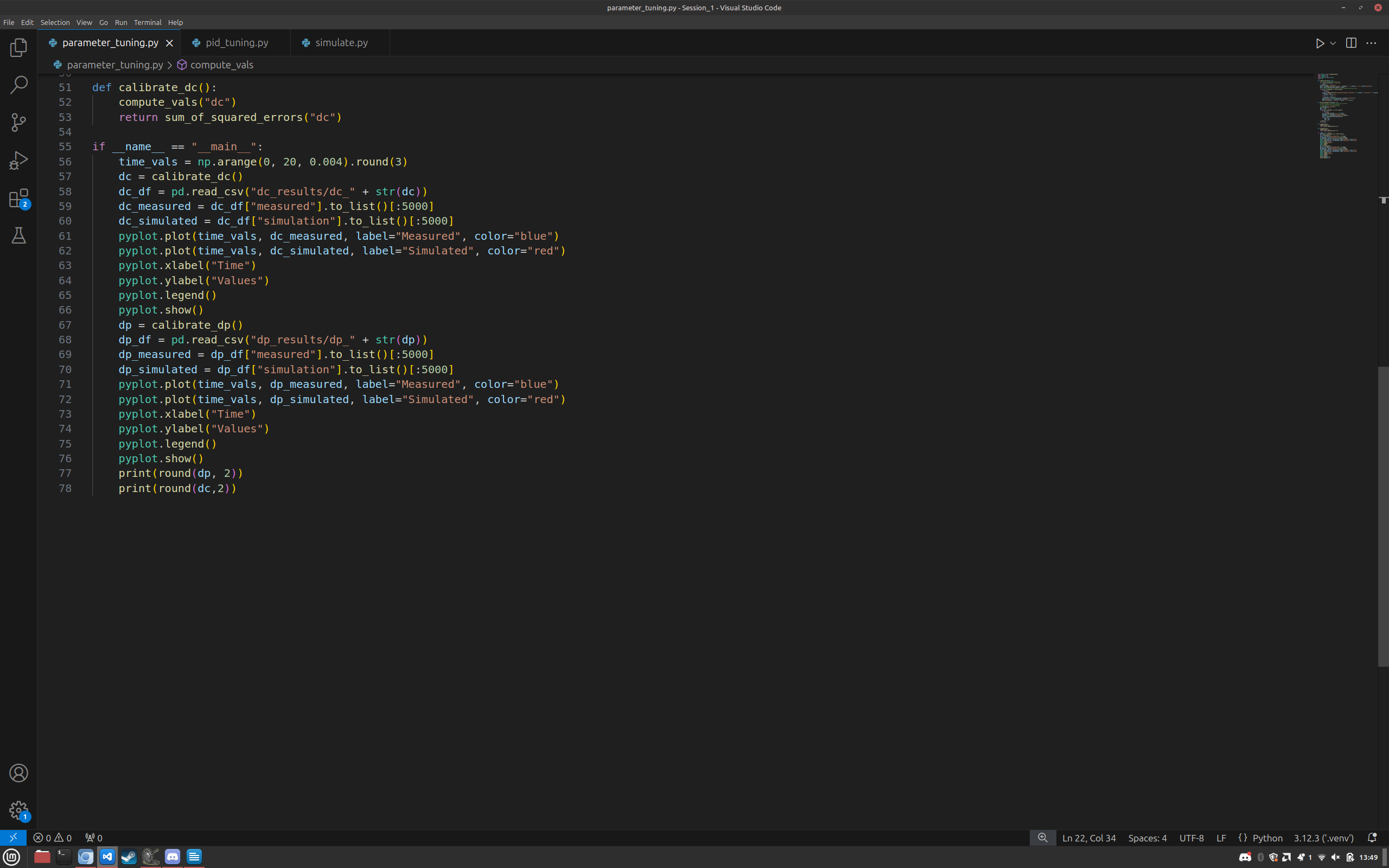
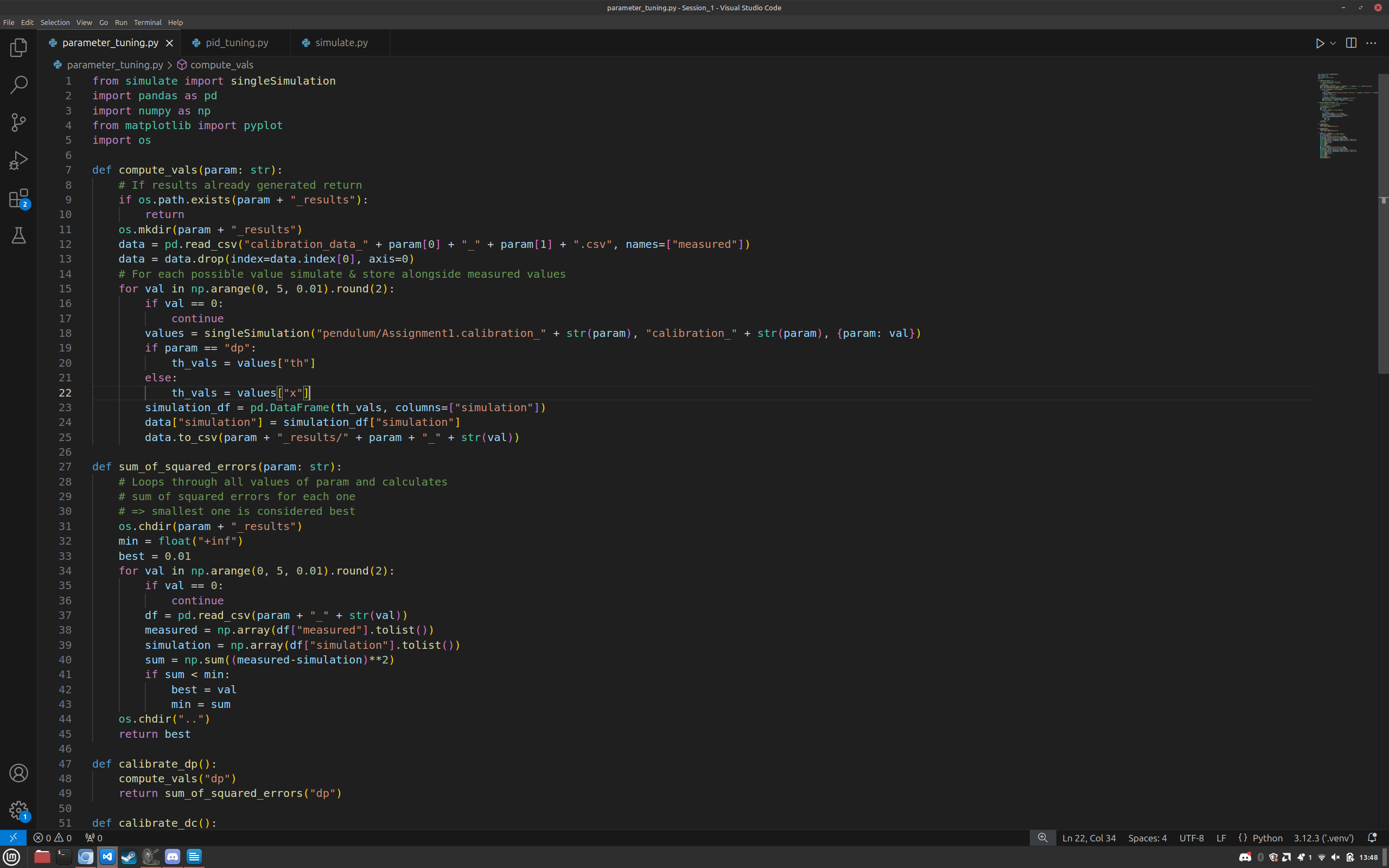
For estimating Dc we will “lock” the pendulum essentially reeling it in:



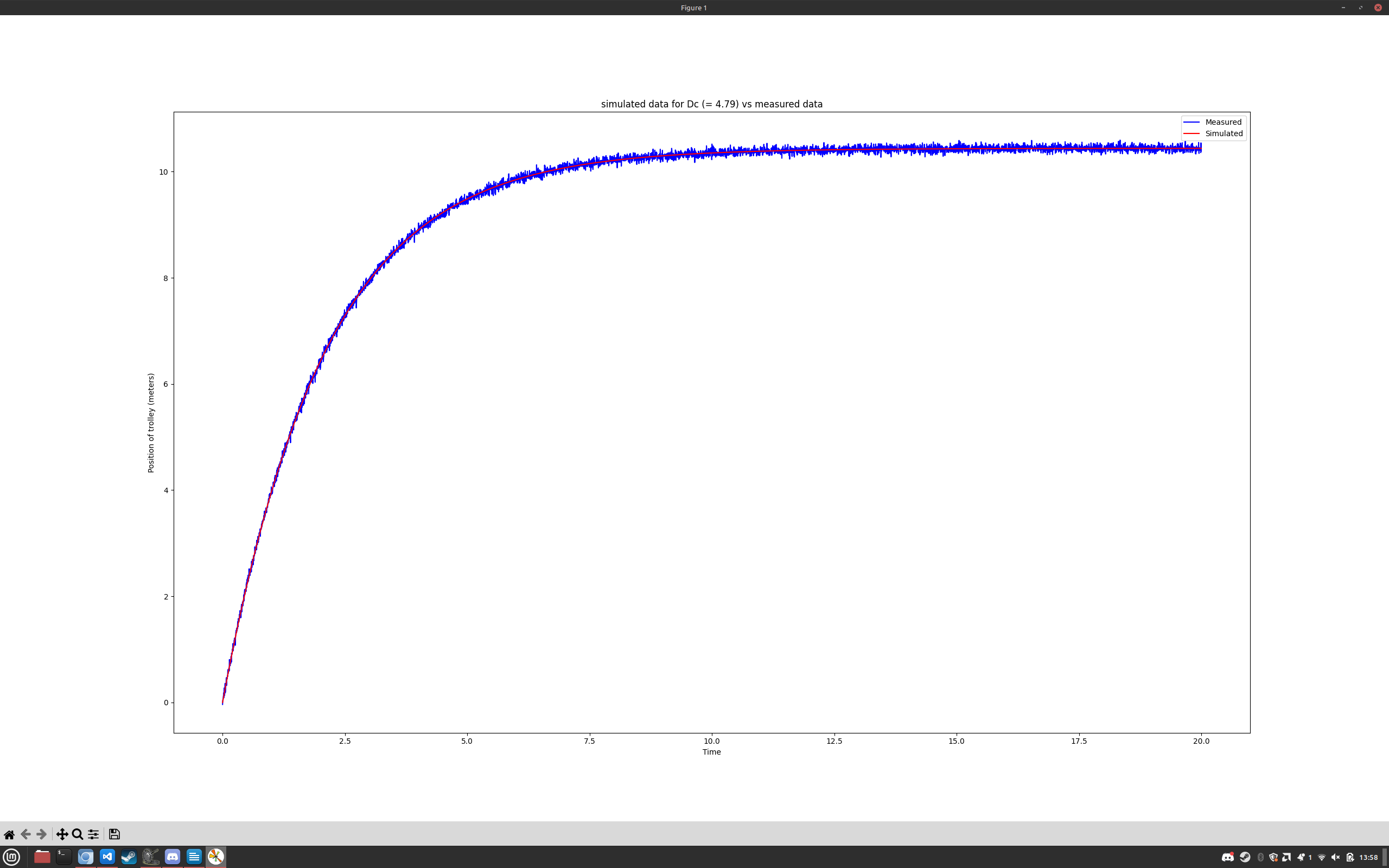
For estimating Dp the trolley’s movement will be locked:

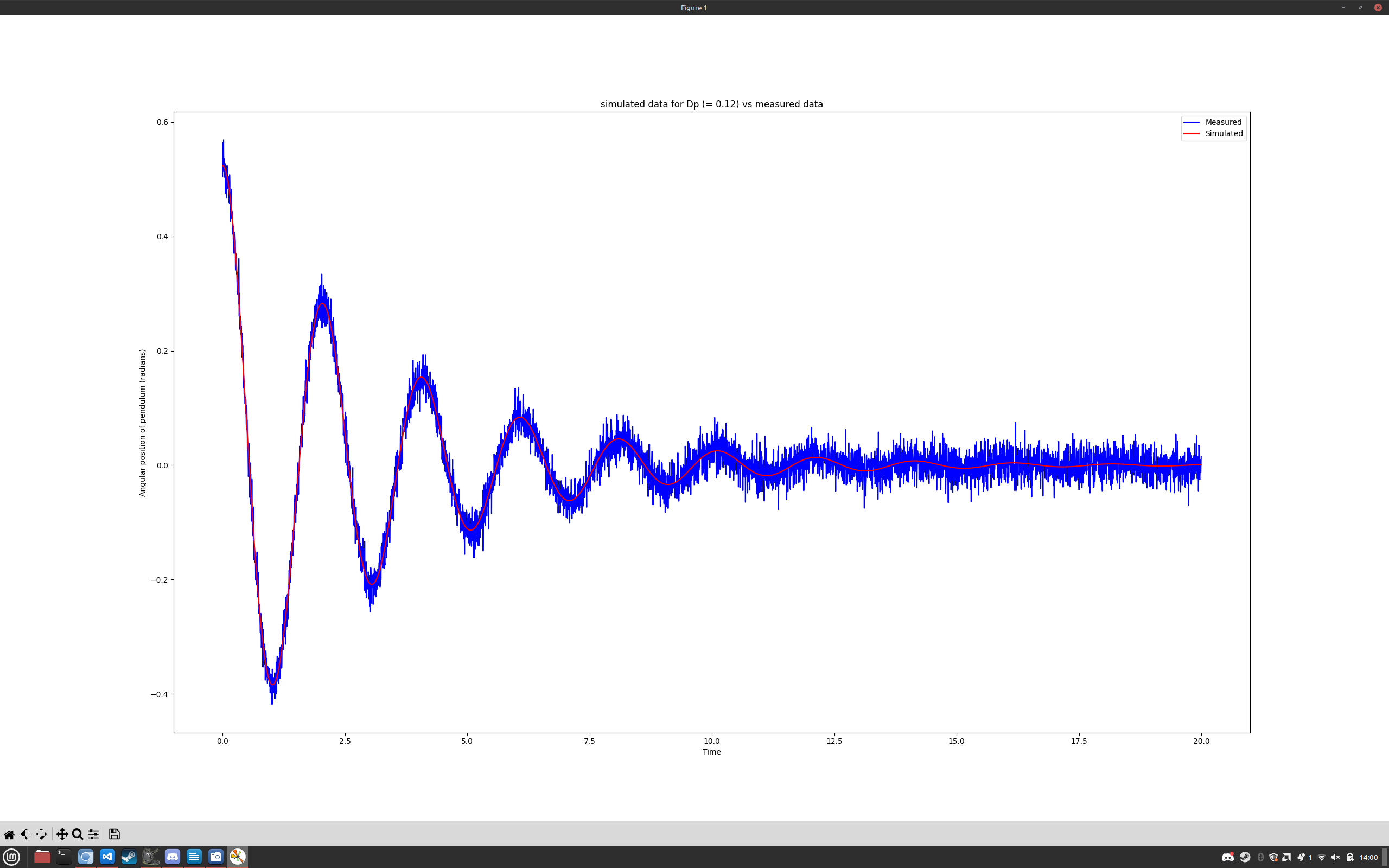


To calibrate these factors, we will use a collection of real life measured data to compare our simulated data to. The damping coefficient Dp & Dc can take on values in the interval (0, 5] (precise up to 2 decimals). For each possible value the model will be simulated compared to the real life data by calculating the sum of squared errors. The simulation with the smallest error will be considered to be the best one.

Code for tuning the parameters (parameter\_tuning.py):  


The above code gave the values 0.12 for Dp and 4.79 for Dc. Plotting the simulation data alongside the measured data for these values generates the following plots:





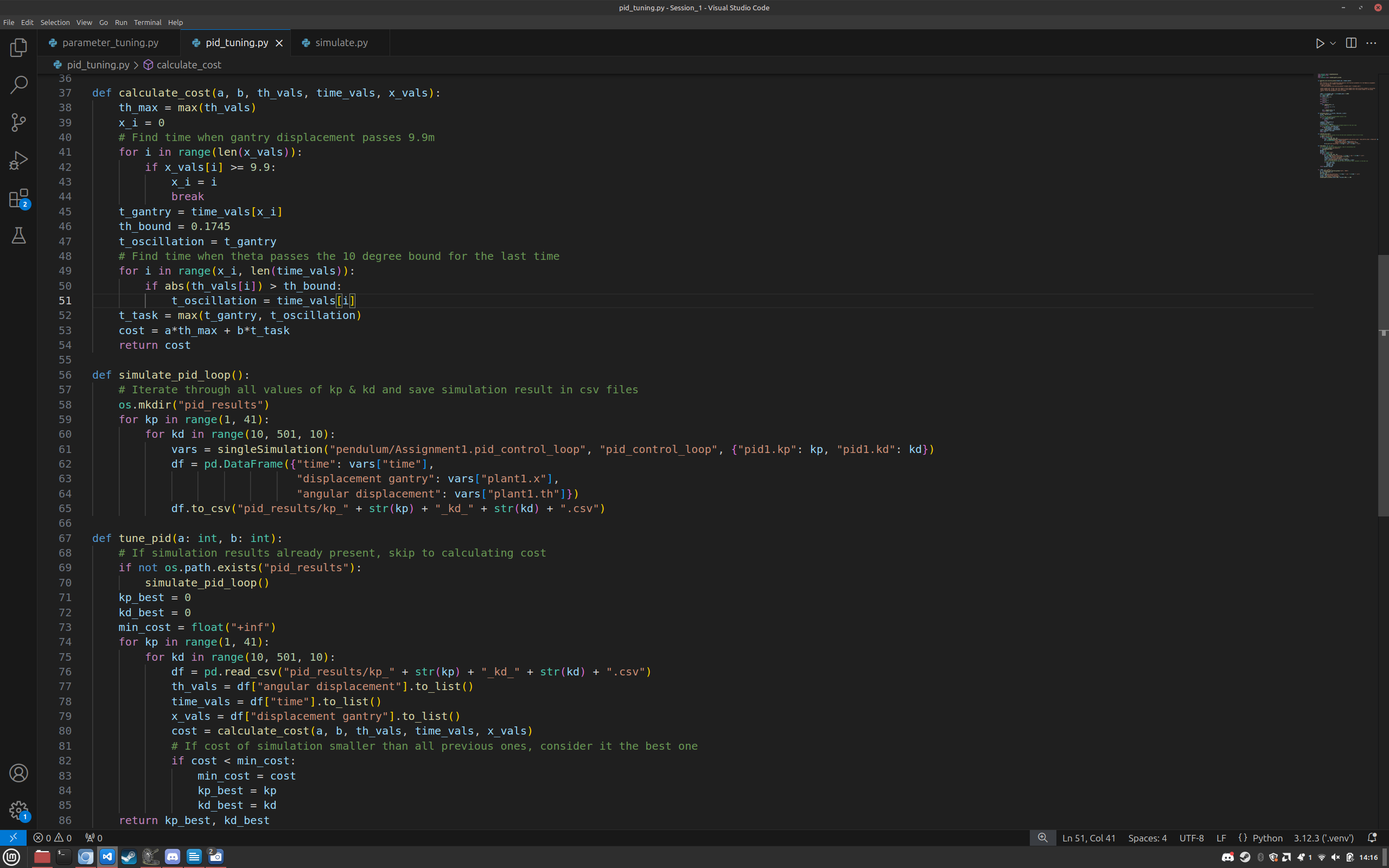
# Controller Model Creation

Text here

# Controller Model Tuning

Now that we have a model for the pid controller and the control loop, it’s time to calibrate it to find the optimal values for Kp, Ki & Kd using the cost function a\*th\_max + b\*t\_task where th\_max is the maximum angular displacement of the pendulum and t\_task is the moment when the gantry reaches the desired set-point of 10 meters (with 10 cm accuracy thus reaching 9.9 meters) and the pendulum's angular displacement remains within the acceptable range of 10 degrees.

Using this cost function it’s clear that a PD controller will suffice thus Ki will be 0. The values for a & b are weight coefficients based on the student ID. Plugging the last four digits of our student ID’s (1127 & 0395) into the provided python script we got the values 15 for a & 22 for b.

Code for finding values with lowest cost:

Executing the script above gave us the optimal values 7 for Kp and 10 for Kd.

Simulating again with these values gives the following plot:

