## B1 Scientific Coding Practical – Technical Report

### Overview

This report covers the updated version of the UUV Mission simulation codebase ([link](https://github.com/tdavies-OX/b1-coding-practical-mt24)). The changes were implemented to extend the functionality of the existing codebase, allowing for fast simulation of missions, and testing of controller gains, ensuring missions can be completed safely with a given controller. Key changes are:

1. Adding .csv reading to dynamic.py – this lets the user import cave missions efficiently
2. Implementing a PD controller in pdcontroller.py, used by the Mission class in dynamic.py – this allows the user to quickly test different PD controller gains, to see how different values affect the UUV’s ability to reject disturbances, follow the trajectory, and avoid crashing into the cave walls.

### Revisions to Codebase

1. In dynamic.py, the Mission class now has a method called from\_csv. The only input parameter is the file path to the desired mission CSV path, whose data should be in columns of [reference, cave\_height, cave\_depth]. This method requires the pandas library v2.2.3, as noted in requirements.txt, and returns each column of values as an array stored in the Mission class.
2. The new file pdcontroller.py introduces a PDController class, used by ClosedLoop’s simulate method in dynamic.py. The class accepts the controller P and D gains upon initialisation. It also has a method called compute, which calculates the controller output, given the time step length, current error (calculated in and passed from the simulate method), and previous time step’s error (stored within the class).

The derivative term is calculated according to

and the controller output is calculated as

The controller was implemented as a class, as this means it need only be passed the controller gains once, upon initialisation. It was made into a separate file, such that future developers will be able to find the controller implementation more easily, and can adapt it into e.g. a PID controller. The controller gains are specified in demo.ipynb, as the mission simulation is visualised in this notebook. Therefore, controller designers can rapidly test different gains, to see how this affects the UUV’s ability to follow the trajectory.

### Reflection and Next Steps

Development (assisted by GitHub copilot) was relatively straightforward once I had set up version management, although I did struggle initially with cross-file referencing and relative pathing. Additionally, there was a brief hiccup with controller behaviour until I realised the controller gain order in demo.ipynb and pdcontroller.py were opposite to each other.

Possible opportunities for further development include additions to the controller (e.g. PID functionality for optimised performance) and making use of terrain.py to generate missions other than that in the csv for additional controller robustness testing.