GSoC'23: Python Bindings for d-SEAMS under the PSF

Personal Details

Name: Ruhila S.
Location: India
University: **Redacted**
Short bio: **Redacted**
Resume: https://github.com/RuhiRG/CV/blob/main/Ruhi_Latest-cv.pdf
Email: **Redacted**
GitHub: https://github.com/RuhiRG/
Motivation: **Redacted**
Other Commitments: **Redacted**

Code Contributions

In keeping with the spirit of the GSoC project, which encompasses maintenance and enhancements, I have made the following Pull Requests during the community bonding period, mostly to familiarize myself with the project. In chronological order:

1. BLD: Use meson without lua, add CI :: As Lua bindings are to be removed, I updated the build system to work without it and added meson to the CI
2. MAINT: Update sol2 :: To ensure builds worked in the CI and across MacOS, Windows (WSL) and Linux, I updated the sol2 library and modified the documentation

Beyond these, I have been studying and trying out the tutorials of Pybind11 to better understand how to interface to various C++ codes.

Project Proposal Details

The main goal is to improve the usability of the d-SEAMS core engine, which is written in C++. d-SEAMS currently uses a mixture of YAML files to control top-level workflows and allows limited interaction from the user-perspective by running the selected workflows within the Lua interpreter. In keeping with the sub-org chosen, the goal is to instead switch to Python, which is more friendly to existing scientists, and has a better set of libraries and tooling for working with molecular systems. Additionally, the Lua interpreter makes it difficult to build robustly, as it is pinned to versions which are often not well supported. The goal is therefore split naturally into two milestones as discussed below.

Milestones

I. Replacing Lua Interpreter

Currently, the Lua interpreter is initialized and calls bound C++ functions in a single file. This is a rather repetitive API, and the code is littered with blocks for different workflows. The first goal is to bind these top-level functions within Python instead. Additionally, a cleaner CLI in C++ would be an outcome of this portion as well.

II. Providing Pythonic Access to C++ classes / functions

After the Python based code runs at parity with the earlier Lua approach, this milestone seeks to provide access to the actual underlying API. Instead of interacting only through top-level functions, the goal here is to provide bindings to allow direct interaction within Python. This would enhance the usability of the library significantly.

Weekly Timeline

Note that --- implies that the **lower task** takes the entire week, i.e. W2 and W3 are spent on the same task if W3 has one defined and W2 has ---.

W1. June 7-14: Setting up the Pybind11 infrastructure and tests for expected outcomes

W2. June 14-21: Add pytests and notebooks to interactively test the code
W3. June 21-28: Port the Two-dimensional Ice workflow
W4. June 28-5 July: Port the One-dimensional Ice workflow
W5. July 5-12: Port the Bulk Ice structure determination
W6. [Midway Evaluation] July 12-19: Rework the YAML / Lua CLI interface in main.cpp (Milestone I ends here)
W7. July 19-26: ---
W8. July 26-Aug 2: Port the point cloud structure and other classes (till the end of Milestone 2)
W9. Aug 2-9: Update the tests to run in the CI and add more documentation
W10. Aug 9-16: ---
W11. Aug 16-23: Rework the interface layers to modularise and allow interoperability with other Python objects (ASE Atoms to PointCloud)

Post GSoC Plans

It is my intention to continue to maintain the Python bindings within d-SEAMS as they will be my first open source contribution. If I gain a better understanding of the underlying code and algorithms during the binding process I would like to extend the inputs and generate workflows for use with GROMACS trajectories.