Implementing the Radius of Curvature as a collective variable using PYCV

January 18, 2025

Introduction

This repository builds upon the work described in the paper by Toni Giorgino, "PYCV: a PLUMED 2 Module Enabling the Rapid Prototyping of Collective Variables in Python," published in the Journal of Open Source Software (2019). This work integrates Python with PLUMED using the PYCVINTERFACE plugin, enabling

The code provided implements a custom function (`r_f`) which calculates the radius of curvatre of a circle passing through 3 atoms and its gradient calculated using the JAX library. This collective variable is designed to interact with PLUMED via the PyCvInterface Plumed directive. The python module plumedCommunications includes the PythonCVInterface class which gives access to simulation data like atomic positions, simulation step, within the Python script and return CV values and derivatives back to PLUMED.

Implementation Components

- **Python Script (curvature.py)**:
  - It is where the PythonCVInterface is loaded.
  - Defines the radius of curvature (`r_f`) and its derivatives (`r_g`).
  - Exports the functions to the PYCVINTERFACE plugin for use in a PLUMED workflow.
- **PLUMED Input File (rc_input.dat)**
  - Configures the PYCVINTERFACE to import the Python script and calculate the collective variable (CV).

Running the Simulation

- Ensure curvature.py is in the same directory as the PLUMED input file (rc_input.dat) and your trajectory file.

- Confirm the shared library PythonCVInterface.dylib is accessible through your path.
- Run the appropriate Plumed driver command.

```
plumed driver --plumed plumed_input.dat --mf_xtc trajectory.xtc
     --timestep 0.002
```

References

- The PLUMED Consortium
- **PYCV: a PLUMED 2 Module Enabling the Rapid Prototyping of Collective Variables in Python** (2019)