**Documentation**

The code requires the ULYSSES package, which is available in the repository: <https://gitlab.com/siriius/ulysses>

To have in-pocket (or any other ULYSSES module) integrated into a code, all that is required is to include the respective header files. Taking the example of the executable for in-pocket, this means including the GFN2 method (GFN.hpp), the mixing of the QM Hamiltonian with the bias potential (in MolecularDynamics.hpp), and the modules for energy minimization (math/SolverPackage.hpp). To compile the code, you can use a standard C++ compiler. All that is needed, in these cases, is compatibility with C++99. I tested so far gcc in Ubuntu machines, and clang with Apple (replacement of gcc). The compilation is simple and can be done using

$gcc gfn2xtb\_in\_pocket\_opt.cpp -O3 -o gfn2xtb\_in\_pocket\_opt.exe

This will generate the executable needed to run InPocket.py. The program that you compiled is a simple interface to a series of functions available in ULYSSES.

The InPocket.py script is a very primitive and simple interface to the cpp executable. I tried documenting what each variable is and what it does. Due to the peculiarities of the previous MacOS I was using, I could not have Pybel properly installed, so I made a simple call to Open Babel within the script. The purpose of Open Babel is just to add protons based on the sdf/pdb format. It may therefore be replaced by any other user-preferred software. The minimum, unambiguous call to the script is something like

$python3 InPocket.py pdbfile=4wa9B.pdb ligandname=AXI charge=0 activeres=248,253,256,269,271,315,316,318,319,321,368,370,380,382

which generates the geometries and outputs that are included in the folder Test. All calculations and optimizations of the structures (residue-ligand pairs) took 25 seconds on my laptop (just to give a rough estimate).

The command above reads the PDB file 4wa9B.pdb, identifies the ligand as the molecule named AXI and assigns it a charge of 0. It will also consider the residues 248, 253, 256, 269, 271, 315, 316, 318, 319, 321, 368, 370, 380, and 382 for the calculations. If you leave the ligand name implicit, then the code will fetch everything that is not protein (HETATM). The charge is only important in case Open Babel gives a wrong protonation state, and then it allows the user to manually fix this issue (and even correct the charge). If you leave the flag activeresidues out from the argument list, then all residues are considered for the calculation. Therefore, the only thing that is important to define is the PDB code, everything else has default values.

The capping of residues is the default on the N-terminus only. For the C terminus, there is an aldehyde generated. The electronic density can be calculated using calcdens=1 additionally. This will generate the density based on a grid that is 0.1 A fine (it can be made rougher, and I can add options to the script). The output format for densities is cube, which can be converted to mrc. By default, implicit water is used, though calculations may be run in a vacuum or other solvents. There is a list of available solvents in the script. Wet octanol is one example that could be interesting in biological applications. I would still like to mention the Telec parameter. This is a specific thing of GFN2-xTB, which defines the electronic temperature that allows the systems to have a multiconfigurational character (non-integer occupations of the orbitals). This is potentially important for transition metals.

You may want to replace the semi-empirical method used for the calculation. Imagine you want AM1. Then replace all references to GFN or GFN2 with AM1 and the code should compile with AM1 (header, basis set, and method). The same for other NDDO methods available (RM1, PM6, PM3, PM3-PDDG, …). There are also dispersion and H-bond corrections for most of these NDDO methods. This requires a change in the initialization of the method that identifies the correction. For AM1 and most other NDDO methods, there is only the D3H4 correction. For PM6 this also includes the X correction for halogen bonds (D3H4X). There is also D3H+ for PM6. It is claimed in the original publication that GFN2-xTB does not require corrections for H-bonds nor halogen bonds, that this is straight out of the method, and usually this seems good. In case someone wants to try these other methods and has a hard time or would like to discuss something, an email to [filipe.menezes@helmholtz-munich.de](mailto:filipe.menezes@helmholtz-munich.de) will do the trick. Unfortunately, I still do not have implicit solvation models for NDDO methods. However, I invested some time in methodologies to favor convergence, even for “weird” ligands. This gives robustness and speed to the optimization of the wavefunction.

Additional information about the ULYSSES program is available in <https://pubs.acs.org/doi/full/10.1021/acs.jcim.2c00757>. The SI of the paper lists atom parametrizations available and the respective references. There are atomic or molecular properties that could be of interest too. Charges, polarizabilities, bond orders. Some of these are available in the output files generated in the “output” folder. All of this should be documented in this paper.