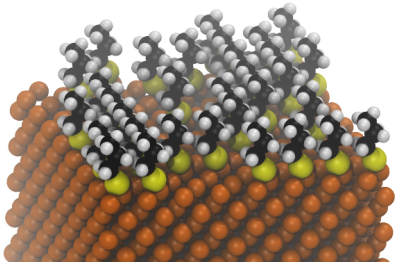
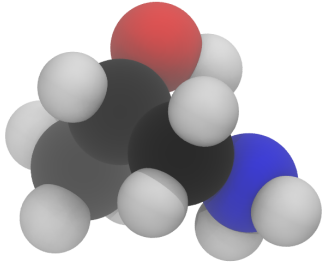
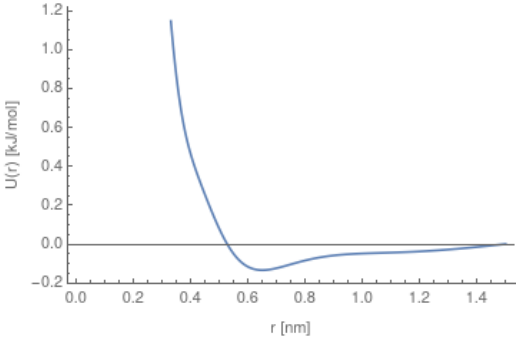
PMFPredictor Tutorial

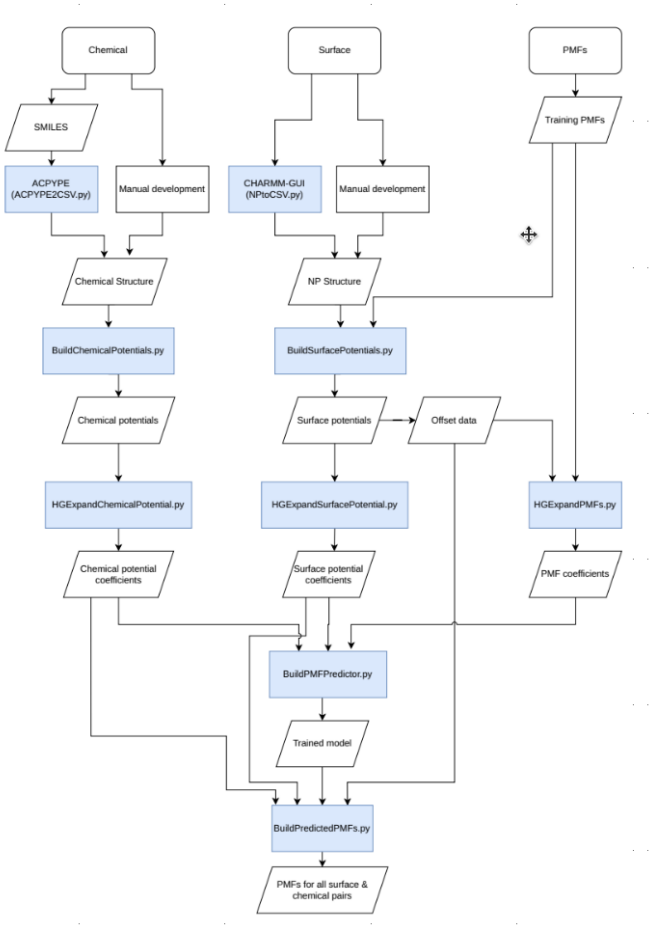


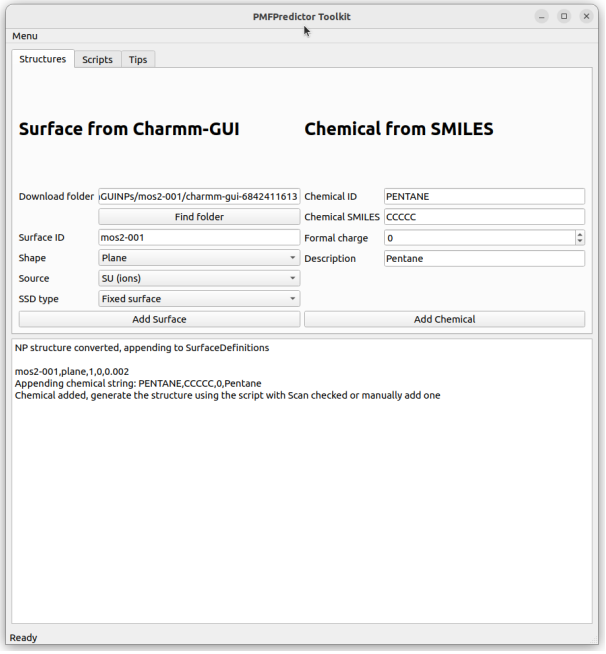


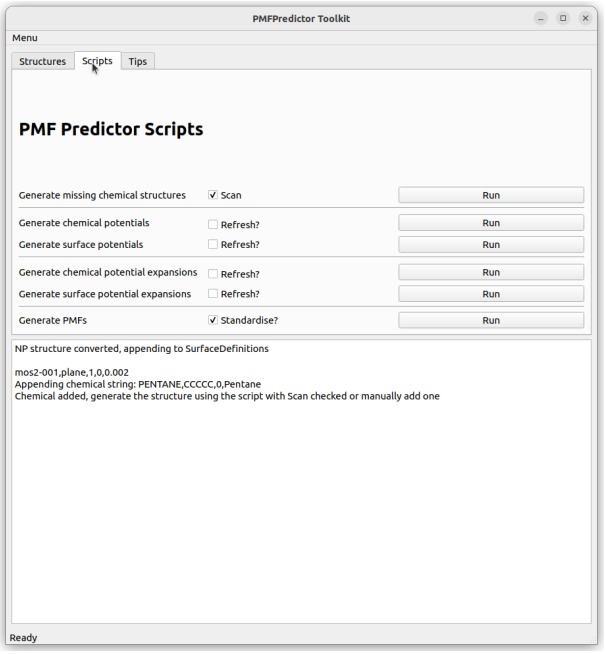


Ian Rouse, 15/11/2022
ian.rouse@ucd.ie

PMF Prediction overview







Left: The overall workflow with scripts shown as blue boxes.
Right: Screenshots of the GUI (PMFPredictor Toolkit) for running the scripts for new input.

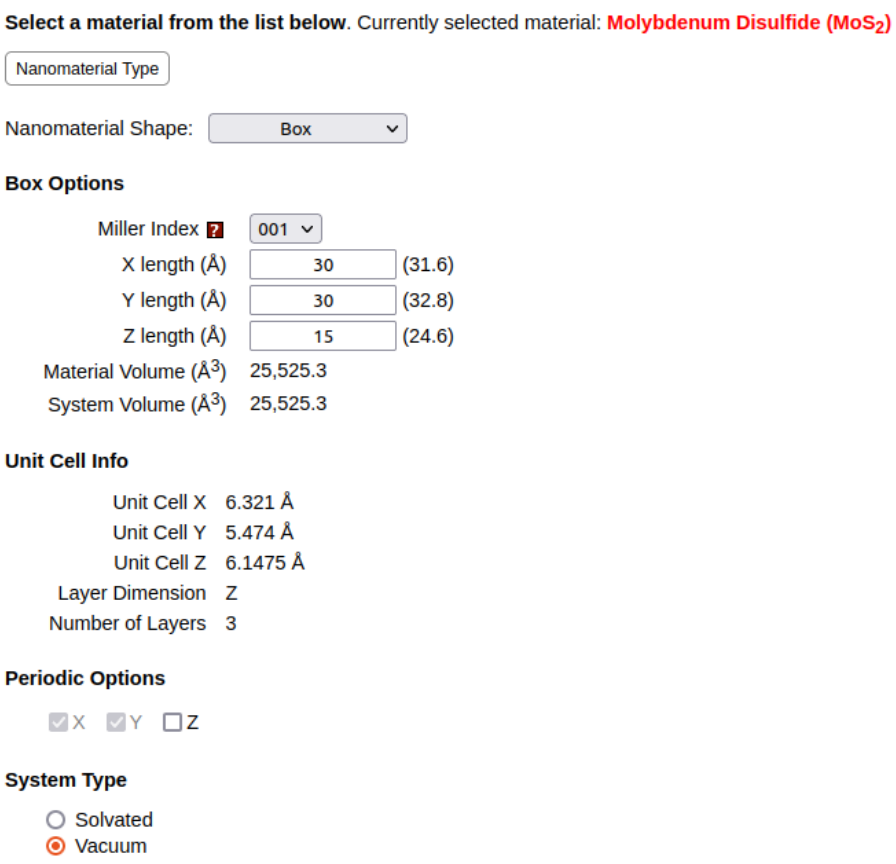
PMFPredictor installation (1)

- Installation of required library packages:
  - At least: pip3 install numpy scipy tensorflow tensorflow_probability pandas argparse matplotlib PySide6
  - Optional for re-training: pip3 install scikit_learn keras_nlp
  - Optional for generating structures: OpenBabel https://pypi.org/project/openbabel/ and AcPype https://pypi.org/project/acpype/
  - Everything:
  - pip3 install keras_nlp matplotlib numpy pandas PySide6 scikit_learn scipy tensorflow tensorflow_probability openbabel acpype

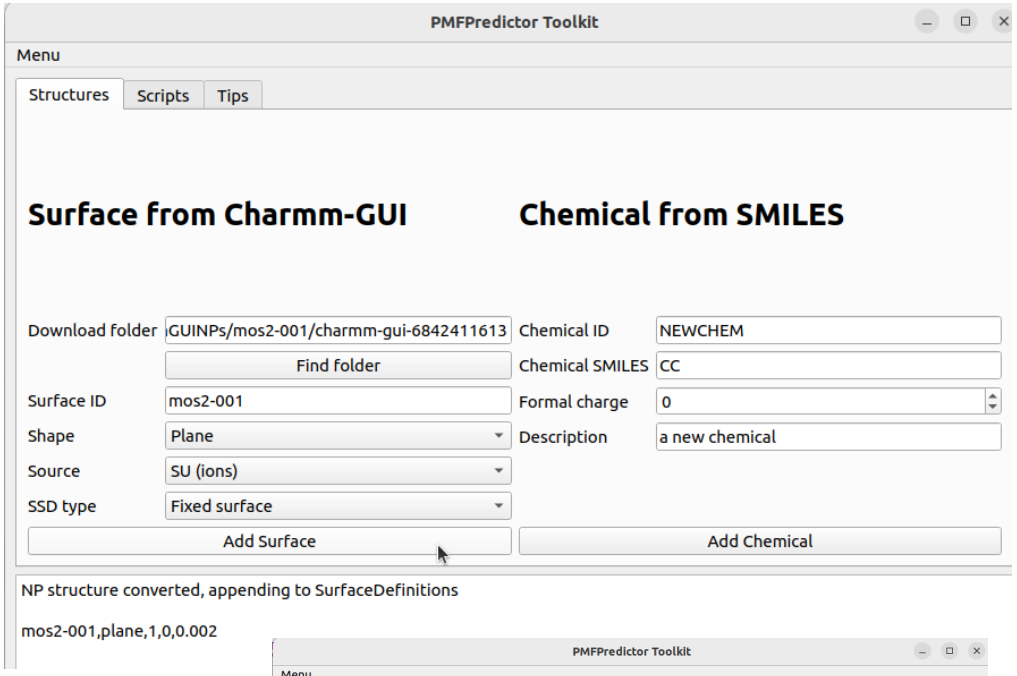
PMFPredictor installation (2)

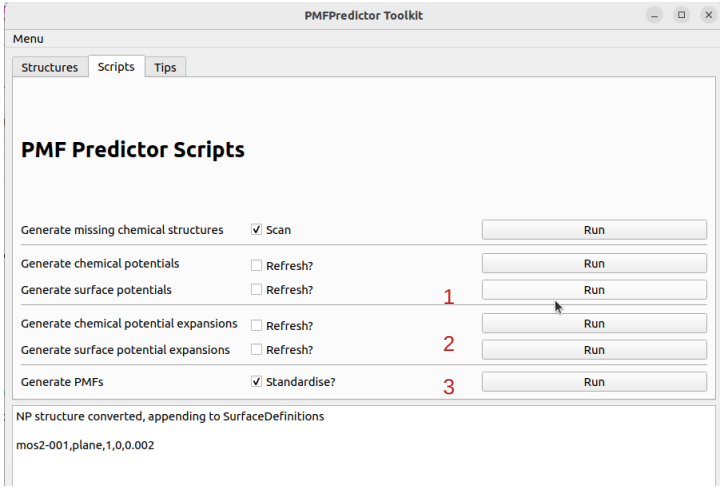
- First time:
  - git clone https://github.com/ijrouse/PMFPredictor-Toolkit.git
  - Afterwards use “git pull” to update
- Download trained models from latest Zenodo release of PMFPredictor-Repository.zip, currently doi:10.5281/zenodo.7303624 (https://zenodo.org/record/7303624), copy the trained models into the “models” folder.
- Download training data if required – not needed for generation of PMFs using trained models.
- Run “python3 PMFPredictorToolkit.py”
- Or “chmod 755 PMFPredictorToolkit.py” and then it can be run with ./PMFPredictorToolkit.py

Adding a CHARMM-GUI material



- Choose the material and Miller index. Recommended settings are Box, 30x30x15, periodic x and y, vacuum system type.
- Screen 2: Fit box size to material.
- Screen 3: Generate automatically
- Screen 4: **Enable GROMACS output**, keep all other defaults.
- Screen 5: Download .tar.gz



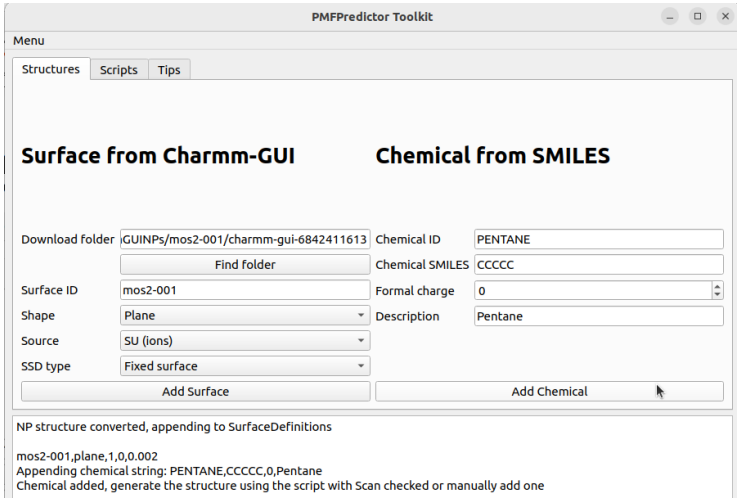


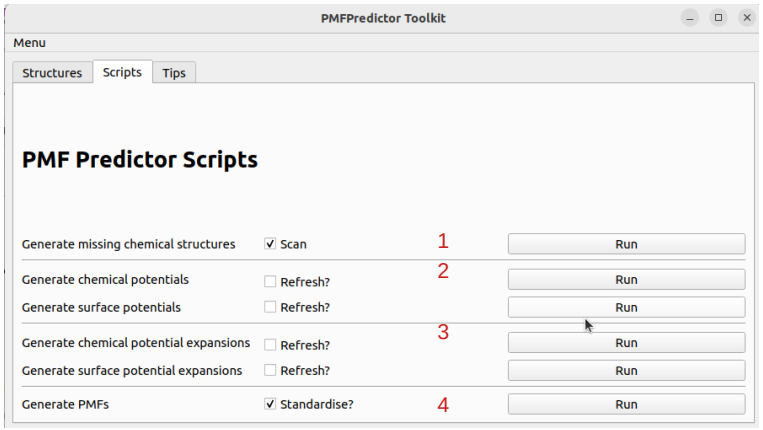
- Unzip the CHARMM-GUI download and find it in PMFPredictor toolkit
  - Make sure you select the base folder, charmm-gui-XXXXXXXXX and that GROMACS output is present
- Name the surface, generally leave the defaults alone.
- Click “Add Surface”
- Then run the scripts to build a potential model, HGExpand, and generate PMFs for your new surface.

Non CHARMM-GUI structures

- The Toolkit only needs these files to be located in the specified folder:
  - basedir/gromacs/toppar/forcefield.itp
  - basedir/gromacs/step3_input.psf
  - basedir/gromacs/step3_input.gro
- General GROMACS input should be acceptable if you rename the files appropriately and they have the same formatting as CHARMM-GUI output.
  - PSF must have one line for each individual input atom to ensure correct charges are assigned
  - ITP must have the same general formatting and have [atomtypes] for all atom types to get LJ parameters.
  - Note that the file parsers may give unexpected results or crash if the formatting is different to CHARMM-GUI output, even if its valid GROMACS input.
- For more complex cases, see NPtoCSV.py

Adding a chemical

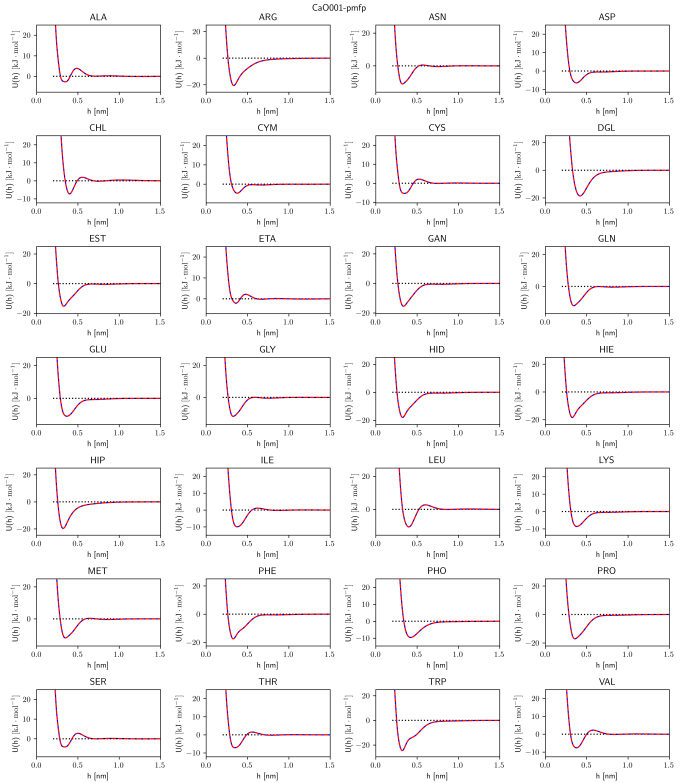




- Assuming you have ACPYPE installed and want to use GAFF, add the chemical and then run scripts 1 - 4.
- If not, add the chemical, manually add the structure file to the Structures/Chemicals folder, run scripts 2 – 4. Check pre-existing structures to see the expected formatting.

Getting PMFs

- Run BuildPredictedPMFs.py either through the toolkit or command line. This generates PMFs for all material – chemical pairs.
- Optional: Run CalcEAdsSet.py to generate a table of adsorption energies.
- Optional: Run UASetFigures.py to generate a set of output folders for UnitedAtom with PMFs renamed to match the UA convention and also generate the material summary figures.



Material summary figure for calcium oxide (001) showing PMFs required for UA

I don’t want to/can’t use the GUI

- ChemicalFromACPYPE.py scans Structures/ChemicalDefinitions.csv and runs acpype to generate any missing structures.
- NPtoCSV.py can convert CHARMM-GUI structures.
- GenerateChemicalPotentials.py generates potentials for chemicals
- GenerateSurfacePotentials.py generates potentials for surfaces.
  - If you’re adding a lot of surfaces at once, install screen and set multiple running in separate sessions:
    - screen -dm python3 GenerateSurfacePotentials.py -i 0 -s 5
    - screen -dm python3 GenerateSurfacePotentials.py -i 1 -s 5
    - …
    - screen -dm python3 GenerateSurfacePotentials.py -i 4 -s 5
  - Then run GenerateSurfacePotentials.py one final time with no arguments to rebuild the SurfaceOffsetData file that the above multi-threading destroys.
- Conversion of the potentials to expansion coefficients is done with HGExpandChemicalPotential.py and HGExpandSurfacePotential.py
- Generation of PMFs for all pairs is done with BuildPredictedPMFs.py , default arguments are recommended.
- All of the above have various command line options that can be configured, try python3 <scriptname> --help to see them.
- Running as ipython notebooks is untested, unsupported, and likely to lead to errors due to variables not getting cleared if cells are run in the wrong order. It is strongly recommended to run everything through the GUI where possible, except for the addition of surfaces which should be done on the command line.