Quick installation and tutorial Guide

This document explains how to install BAT.py and its needed dependencies, and running the BAT tutorial, on a Linux machine with an OS such as Ubuntu. We make use here the Anaconda package manager, which makes it easier to install some of the needed software.

Installation

To make BAT.py operational on your system, follow the steps below. If you already have an Anaconda environment you wish to use to run BAT.py, you can skip to step 2.

1. Install Anaconda: First download and install the Anaconda package manager, from the page:

https://www.anaconda.com/download

Follow the instructions on this page, and download the Linux version of Anaconda to your folder of choice, such as your home directory. For most local machines the Linux x86 version should be chosen.

Install Anaconda from the file you downloaded, typing the command below and following the instructions:

```
you@yourmachine:~$ bash Anaconda3-VERSION-Linux-x86_64.sh
```

After the installation, close all your terminal windows, and open a new window. Create a new Anaconda environment and activate it, or activate a created environment of your choice. Instructions on how to create and activate Anaconda environments can be found here https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html#.

2. Install Ambertools: Still inside your Anaconda environment, you will now install some of the software needed for BAT.py, starting with Ambertools. Type in your command line:

```
(environment)you@yourmachine:~$ conda install
conda-forge::ambertools
```

Recent Anaconda releases (from 2025) showed problems with Ambertools, particularly with the pdb4amber and parmed tools. If you get an error when running the “pdb4amber” or “parmed” commands in your environment after the Ambertools installation, you will have to downgrade your environment’s numpy version with the command:

```
(environment)you@yourmachine:~$ pip install --upgrade numpy==2.2.6
```

After that the problem should be fixed. Alternatively, You can also install Ambertools in other ways, with instructions here https://ambermd.org/GetAmber.php.

3. Install VMD 1.9.3: To install VMD in your Anaconda environment, type:

```
(environment)you@yourmachine:~$ conda install conda-forge::vmd
```

4. Install OpenMM with OpenMMtools: If you want to use OpenMM with OpenMMtools for your simulations (more details in the BAT tutorial), install both of them in your Anaconda environment. Do that by typing these two commands and following the instructions:

```
(environment)you@yourmachine:~$ conda install conda-forge::openmm

(environment)you@yourmachine:~$ conda install
conda-forge::openmmtools
```

5. Install Openbabel 2.4.1: Even though Anaconda provides the 3.1 version, here we want to use the 2.4.1 version instead. Thus, we do not recommend installing the Anaconda version of Openbabel, because it will then execute the 3.1 version.

To install the 2.4.1 version, download the source code zip file from the GitHub page https://github.com/openbabel/openbabel/releases/tag/openbabel-2-4-1 . Unzip it in your folder of choice, typing:

```
you@yourmachine:~$ unzip openbabel-openbabel-2-4-1.zip
```

The Basic Installation instructions in the just created ./openbabel-openbabel-2-4-1/INSTALL file require sudo privileges, which might not always be the case. To overcome that problem, replace the "`cmake ..`" command in the instructions with the command below:

```
you@yourmachine:~$ cmake .. -DCMAKE_INSTALL_PREFIX=~/obabel-inst
```

This will install the needed files in the folder `~/obabel-inst` instead. After that, follow the rest of the basic instructions in the INSTALL file, but replacing the command "`sudo make install`" by just "`make install`".

Once installation is concluded, you might want to create a link to be able to execute openbabel at any folder, which can be done with the command:

```
you@yourmachine:~$ ln -s ~/obabel-inst/bin/obabel ~/bin/obabel
```

The `~/bin` folder can be replaced by any other folder that is in your executable path.

6. Install *pmemd.cuda:* If you wish to use AMBER's *pmemd.cuda* for your simulations, instead of OpenMM with OpenMMtools, you will need to install this software as well. This can be done by downloading Amber24 from the https://ambermd.org/GetAmber.php page, and following the installation instructions.

7. Download BAT.py: If you have not yet done so, download the BAT.py distribution from the GitHub page https://github.com/Gheinzelmann/BAT.py. Unzip the downloaded file at your folder of choice, typing:

```
(environment)you@yourmachine:~$ unzip BAT.py-master.zip
```

You will now see a ./BAT.py-master folder, contaning the program and the needed files for the tutorial. The tutorial will be performed inside the ./BAT.py-master/BAT folder.

8. Run the tutorial: Your Anaconda environment should now have all the needed programs installed. Make sure all of them are are in your path, so they can be executed inside any folder

during the BAT.py routines. Perform the tutorial as explained in the BAT GitHub page, or following the simplified and command-oriented instructions below.

Performing the tutorial using OpenMM

Once you have everything installed as explained above, run the following command on your command line, inside the ./BAT folder:

```
(environment)you@yourmachine:~$ python BAT.py -i input-openmm-rank.in -s equil
```

Now, to run the equilibration simulations using OpenMM, go the ligand folders that were created inside the ./BAT/equil folder, and on each run the command:

```
(environment)you@yourmachine:~$ source run-local.bash
```

You can also use the PBS and SLURMM files provided, whose templates are inside the ./BAT/run_files folder. Once all the equilibration simulations are *finished*, go back to the ./BAT folder and type:

```
(environment)you@yourmachine:~$ python BAT.py -i input-openmm-rank.in -s fe
```

Now go inside each ligand folder in the ./BAT/fe directory, and copy the run-op-express.bash file from the ./BAT/run_files folder to the current one. This bash script is adjusted for SLURMM, edit it if running the simulations locally or using PBS. Now run the script:

```
(environment)you@yourmachine:~$ source run-op-express.bash
```

This will run several simulations at the same time, so make sure your local environment or server has its GPUs properly set. Once all the the simulations are finished, inside the ./BAT folder type:

```
(environment)you@yourmachine:~$ python BAT.py -i input-openmm-rank.in -s analysis
```

Now inside each ligand folder in ./BAT/fe/ directory there should be a Results folder. There, the user can find all the calculated free energies for the blocks and for the whole run.