## Citation:

Please cite the following papers if you perform simulations with PACE:

1) Han, W.; Schulten, K. *J. Chem. Theory Comput.* **2012**, *8*, 4413.
2) Han, W.; Wan, C.-K.; Jiang, F.; Wu, Y.-D. *J. Chem. Theory Comput.* **2010**, *6*, 3373.
3) Marrink, S. J.; Risselada, H. J.; Yemov, S.; Tieleman, D. P.; de Vries, A. H. *J. Phys. Chem. B* **2007**, *111*, 7812.

If you perform membrane protein simulations, please also cite:

Wan, C.-K.; Han, W.; Wu, Y.-D. *J. Chem. Theory Comput.* **2012**, *8*, 300.

## Prerequisites:

The following programs are needed for using PACE:

1) A modified version of NAMD 2.9 is needed for PACE simulations. Its source codes are provided within the package (`src_ModNAMD/`).

2) Python 2.6 or later is needed. To install Python, please visit `www.python.org`;

3) A C compiler is required. The GNU C complier will be used throughout this tutorial;

4) VMD is needed to visualize systems and construct topology files that are used for simulations with NAMD. To install VMD, please visit `http://www.ks.uiuc.edu/Research/vmd/`.

5) We assume that users have basic knowledge about how to build a simulation system for NAMD. For the users new to NAMD, please look for more information about NAMD via the following website: `http://www.ks.uiuc.edu/Training/Tutorials/`.

## Installation:

1) Installation of modified NAMD:
   To install modified NAMD, you first need to obtain the installation package for standard NAMD 2.9. A copy of the package, named `NAMD_2.9_Source.tar.gz`, should be available in the present directory. Decompress the package as follows:

   ```
   tar -zxf NAMD_2.9_Source.tar.gz
   ```

   A new directory, namely `NAMD_2.9_Source/`, will be generated and all installation files are located within this directory, including

NAMD_2.9_Source/src/ which contains the source codes for the standard NAMD. Then please unzip SOURCE_PACE.tar and replace NAMD_2.9_Source/src/ by the resulting directory src/ generated from SOURCE_PACE.tar. After that, you should follow exactly the same procedure as is done to install NAMD. For details about how to compile and install NAMD, please check http://www.ks.uiuc.edu/Research/namd/2.9/notes.html. To simplify the installation, a pre-compiled executable, namd2pace, for AMD/Intel multicore machines with 64-bit Linux system is provided.

2) Compile a program named genNAMDPair-lip which will be used to modify topology files for PACE:

   ```
   cd programs/
   gcc genNAMDPair-lip.c -o genNAMDPair-lip
   ```

   From now on, we assume that the path to programs/ is PROGPATH/.

3) Suppose that your working directory is, e.g., at WORKDIR/. Please copy the following files into the working directory:

   ```
   cp FF/* WORKDIR/
   cp CONF/* WORKDIR/
   ```

   FF/ contains all the topology files (.rtf) and parameter files (.prm) and CONF/ contains simulation configuration files (.conf).

## Getting started for simulations:

In this section, we will illustrate how to prepare systems for NAMD simulations. Two types of systems will be discussed, one with proteins solvated in CG water, and the other with proteins embedded in membrane. We assume here that the users are currently at their working directory WORKDIR/.

### Building systems with proteins in CG water

1) Obtain a protein coordinates in pdb format, either from www.rcsb.org or from your own simulations. An example of pdb files (3gb1.pdb) is provided in the package of PACE. Save the pdb file to your working directory WORKDIR/.

2) Simulations with NAMD rely on two types of files, a psf file having topological information of systems and a pdb file containing starting atomic coordinates. A program called psfgen, which can be a standalone program or a plugin associated with VMD, can generally used to yield, based on the pdb file from step 1), the psf and pdb files with corrected formats needed by NAMD. Such a procedure follows a series of instructions specified in tcl script files provided by users. We provide programs to generate necessary scripts for users to build up the

files for simulations with PACE in NAMD so that there is no need for the users to involve the details of writing scripts.

The first program is `PROGPATH/pdb2tcl.py.` It generates the required script file based on the raw `pdb` file. Note that in a `pdb` file from simulations, histidine residues may have a non-standard name like "`HSD`", "`HSE`" or "`HSH`". Please change any of such into "`HIS`" before use the `pdb` file as input for the program. An example of using the script is shown as the following:

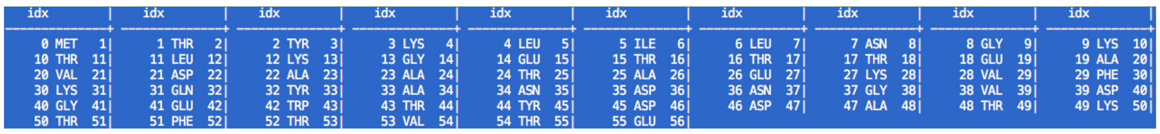
```
python PROGPATH/pdb2tcl.py 3gb1.pdb pace-reopt.rtf
[HIS]
```

In this example, `pace-reopt.rtf` is a topology database of PACE which should be present in `WORKDIR/`. `HIS`, which is optinal, tells the program to let users to determine protonation states of each histidine residue. Otherwise, the program will assume that all histidine has its $N_\delta$ protonated. Finally, the program calculates distance matrix for all Cys pair. The users can determine the presence of disulfide bonds according to the calculated distances.

One needs to group the whole system into segments for NAMD simulations. Each individual protein chain is required to be in a separated segment with a unique segment name. The program will first check if the `pdb` file that users provide contains proper segments. The following question like below may appear:

```
Chain 1 starts at :MET1
Want to re-split chain structures?(Y/N)
```

Here the program found that there are only one chain in the provided file, e.g., `3gb1.pdb`, but there is no segment name for this chain (Otherwise you would see "`xxx:MET1`" instead of "`:MET1`"). In some cases, your system may contain more than one protein chain but the program recognizes all of them as one. In such a situation, you need to answer "`y`" to let the program help you to re-partition the system. If you choose "`y`", the program will proceed and summarize index information about all protein residues in the provided `pdb` file. You should see information like below:



```
   idx        |   idx        |   idx        |   idx        |   idx        |   idx        |   idx        |   idx        |   idx        |   idx        |
--------------+ -------------+ -------------+ -------------+ -------------+ -------------+ -------------+ -------------+ -------------+ -------------+
     0 MET   1|     1 THR   2|     2 TYR   3|     3 LYS   4|     4 LEU   5|     5 ILE   6|     6 LEU   7|     7 ASN   8|     8 GLY   9|     9 LYS  10|
    10 THR  11|    11 LEU  12|    12 LYS  13|    13 GLY  14|    14 GLU  15|    15 THR  16|    16 THR  17|    17 THR  18|    18 GLU  19|    19 ALA  20|
    20 VAL  21|    21 ASP  22|    22 ALA  23|    23 ALA  24|    24 THR  25|    25 ALA  26|    26 GLU  27|    27 LYS  28|    28 VAL  29|    29 PHE  30|
    30 LYS  31|    31 GLN  32|    32 TYR  33|    33 ALA  34|    34 ASN  35|    35 ASP  36|    36 ASN  37|    37 GLY  38|    38 VAL  39|    39 ASP  40|
    40 GLY  41|    41 GLU  42|    42 TRP  43|    43 THR  44|    44 TYR  45|    45 ASP  46|    46 ASP  47|    47 ALA  48|    48 THR  49|    49 LYS  50|
    50 THR  51|    51 PHE  52|    52 THR  53|    53 VAL  54|    54 THR  55|    55 GLU  56|
```

```
Please input starting residue indices for new chains
```

In each entry shown are absolute residue index, residue name and residue index as indicated in the `pdb` file. Only the ABSOLUTE residue indices are used to re-partition the system. Because there is only one chain in `3GB1`, you should type "0" which means the chain starts at the first residue and ends at the last one. However, if you have, e.g., a dimer, each monomer having 50 residues, you should type "`0 50`", which means the first monomer span residues with absolute indices [0, 49] and the second spans [50, 99]. Once the chains are selected, each of them will be assigned a segment name, with "`P`" as initials. If you don't like the assigned segment name, you can rename them in the next step:

```
Want to reset segname?(Y/N)
```

However we recommend avoiding using names starting with "`S`" that will be used by the programs for other purposes. The program will eventually generate a `tcl` file for `psfgen` and the following summary information will show up:

```
+------------+
|Please note:|
+------------+
3gb1_auto.tcl has been generated.
Try to use psfgen to generate psf with this tcl
file.
You should get 3gb1_auto.pdb and 3gb1_auto.psf for
proteins
```

3) Generate `psf` and `pdb` for NAMD from `3gb1_auto.tcl` as follow (assuming your have correct `PATH` to VMD):

```
vmd -dispdev text -e 3gb1_auto.tcl
```

In case you have the standalone psfgen, you may also try the following:

```
psfgen 3gb1_auto.tcl
```

For either approach, you should obtain `3gb1_auto.psf` and `3gb1_auto.pdb`. At this point, only `3gb1_auto.pdb` is needed for the next step.

4) Thus far we already have a NAMD-recognizable `pdb` file for proteins only. Another program called `pdb2tcl_combine.py` will be used to include environment into your system. It basically combines different `pdb` files, such as the ones for proteins and environment, respectively, and generates a combined system for `psfgen`. An example is shown as follows:

```
python  PROGPATH/pdb2tcl_combine.py 3gb1_auto.pdb
watnamd.pdb pace-reopt.rtf box  65  65  65 exclusion 4
```

Notes:

(a) All the `pdb` files should be already processed by `psfgen/VMD` by following steps 2) and 3).

(b) PACE toplogy database `pace-reopt.rtf` is needed.

(c) "`box`" means size of simulation box. In order to determine the size of box, we can first determine the size of protein by:

```
python PROGPATH/pdb_minmax.py 3gb1_auto.pdb

          x          y          z
min     -16.81     -15.03     -11.22
max     18.11      13.12      10.79
```

Suppose that we attempt to make a cubic box with a minimum clearance of 15 Å between proteins and box edges, the size should be 65×65×65 $Å^3$.

(d) The order of the `pdb` files as shown in the command line is important. If structures from two `pdb` files have overlapping parts, the program will remove the part from the `pdb` file that appears later. The parameters following "`exclusion`" specify a cutoff distance to remove the overlapping part, which is 4 Å in the present case.

(e) One useful function of `pdb2tcl_combine.tcl` is to solvate proteins. Actually, the example show how to solvate proteins with CG water in PACE. Here `watnamd.pdb` is a pre-processed CG water box with a size of 200×200×200 $Å^3$.

Once the program finishes the calculation, it should generate the following information:

```
+------------+
|Please note:|
+------------+
3gb1_auto_combined.tcl has been generated.
Try to use psfgen to generate psf with this tcl file.
You should get 3gb1_auto_combined.pdb and
3gb1_auto_combined.psf for combined structures
```

5) Run `psfgen/VMD` again:

```
vmd -dispdev text -e 3gb1_auto_combined.tcl
```

Then you should get `3gb1_auto_combined.pdb` and `3gb1_auto_combined.psf`.

6) We are almost there. `3gb1_auto_combined.pdb` and `3gb1_auto_combined.psf`, generated in the last step, can be by themselves used in NAMD simulations. However, we need to modify `3gb1_auto_combined.psf` to further include a few exclusion lists as required by PACE. This is done through the complied program `genNAMDPair-lip` as follows:

   ```
   PROGPATH/genNAMDPair-lip 3gb1_auto_combined.psf
   3gb1_sim.psf
   ```

   `3gb1_sim.psf`, instead of `3gb1_auto_combined.psf`, is exactly what you need in simulations.

7) For NAMD, configuration files (`.conf`) specify simulation conditions and what files to be used. We provide three `.conf` files as templates:

   `min.conf` ; minization
   `pr.conf` ; pre-equilibrium
   `prod.conf` ; for production run

   For all `conf` files, you need to set file entries for `structures`, `coordinates` and `parameters` as `3gb1_sim.psf`, `3gb1_auto_combined.pdb` and `pace-reopt.prm`, respectively. The equilibrium simulations include minimization (`min.conf`) and pre-equilibration (`pr.conf`). Don't forget to set the correct box size in `min.conf`. After that, you could carry out production runs. Note that when you perform minimization with multiple CPUs, NAMD may stop to report that the box shrinks too much to fit into the grids set up at the beginning of simulations. This is normal and you can continue pre-equilibration from the checkpoint files.

**Building systems with proteins embedded in membrane**

The procedure of building a membrane system is very similar to that of building a solvated protein. The basic idea is to start with an all-atom system and convert it into CG models. There are several useful tools to construct all-atom membrane proteins for simulations. Here are the links to those tools:

VMD:
`http://www.ks.uiuc.edu/Research/vmd/`
CHARMM-GUI
`http://www.charmm-gui.org/`

`pdb2tcl.py` will be used for the conversion. The program first decomposes the system into two parts. The first part contains proteins only and the proteins are converted in PACE models; the second part contains lipids, ions etc that are converted in MARTINI CG models. For each part, a separated script file will be generated for `psfgen/VMD`. We provide here an all-atom system (`all.pdb`) of LeuT in POPC bilayer as an example. Run `pdb2tcl.py` as follows:

```
python PROGPATH/pdb2tcl.py all.pdb pace-reopt.rtf
martini_v2.0_lipids.rtf martini_v2.0_ions.rtf [HIS]
```

Note:
1) as `all.pdb` contains not only proteins but also ions and lipids, the corresponding `.rtf` files are needed

2) if the `pdb` files contains residues that cannot be recognized by the program, you may see the warning like the following:

```
TIP3 1 is not chosen for any class of segments
We have following groups of residue names for
different classes:
0 : ['ALA', 'GLY', 'VAL', 'ILE', 'LEU', 'PHE', 'TYR',
'TRP', 'ASN', 'GLN', 'ASP', 'GLU', 'LYS', 'ARG',
'ACE', 'NMR', 'PRO', 'DPR', 'SER', 'CYS', 'MET',
'HIS', 'HSD', 'HSE', 'HSH', 'ASPH', 'GLUH', 'THR',
'CYSH', 'NLE']
1 : ['SOL', 'TIP', 'SOD', 'CLA', 'NA', 'CL', 'MG',
'POPC', 'DPPC', 'POPE', 'DPPE', 'RA', 'RU', 'RC',
'RG']
Please choose one or this type of residue is all
discarded
```

In the above case, residue `1` in all.pdb is TIP3P water. As the explicit water like TIP3P will not be used for building the model, it can be ignored and you could type “`n`”. Note that once you choose to discard TIP3P for residue `1`, all TIP3P residues in the `pdb` file will be ignored. On the other hand, if you want to consider an unrecognized residue type, you need to let the program know whether such type should be considered as protein part or not (class “`0`” or “`1`”).

3) The script for lipid and ion parts will be generated. You should see the following information:

```
+---------------------+
|Solvent, lipid and ion|
+---------------------+
+------------+
|Please note:|
+------------+
all-small_auto.tcl has been generated.
Try to use psfgen to generate psf with this tcl file.
You should get all-small_auto.pdb and all-
small_auto.psf for lipids, solvent, ions etc
```

4) The script file for proteins part will be generated in the same way as discussed in the previous section.

Now you should have script files `all_auto.tcl` for PACE proteins and `all-small_auto.tcl` for CG lipid and ions. Run `psfgen/VMD` for each script to obtain `all_auto.pdb` and `all-small_auto.pdb`. Then you need to combine the parts:

```
python PROGPATH/pdb2tcl_combine.py all_auto.pdb all-
small_auto.pdb watnamd.pdb pace-reopt.rtf
martini_v2.0_lipids.rtf martini_v2.0_ions.rtf exclusion 0 4
Z-exclusion SOL -15 15 box 105 110 100
```

Note:
1) `pdb2tcl_combine.py` can combine more than two `pdb` files. The `pdb` files are combined in the order of their appearance in the command line as discussed earlier, i.e., that the program combines the first two `pdb` files into one and continues to include another `pdb` file into the combined `pdb` iteratively until no more `pdb` files are left.

2) In the above case, the program will combine the protein part (`all_auto.pdb`) and the lipid and ion part (`all-small_auto.pdb`) first. Then it will merge the combined `pdb` file with a CG water box, namely `watnamd.pdb`, to solvate the whole system.

3) “`exclusion 0 4`” tells the program that when combining the first two parts, the cutoff distance for overlapping parts are zero, i.e., that it removes nothing. This is because the first two parts come from an all-atom system and we assume that the clash check has already been dealt with in that system. When

combining the merged pdb of the first two parts with the water box, the program will apply a 4 Å of cutoff for clash check.

4) Box sizes are determined according to the size of lipid and ion part (all-small_auto.pdb), similar to what we showed earlier.

5) "Z-exclusion SOL -15 15" means that the program will remove from the resulting system all residues named SOL and positioned between -15 to 15 Å in z direction. This is equivalent to remove CG water particles that are placed in the lipid tail region.

You should obtained all_auto_combined.tcl which can in turn be used with psfgen/VMD to generate all_auto_combined.pdb/psf. Again, you need to modify all_auto_combined.psf using genNAMDpair-lip, as discussed earlier.

Suppose that eventually you obtain all_auto_combined.pdb and all_sim.psf, you need to perform a series of pre-equilibrium simulations with NAMD so that initial setup does not distort your protein structures. We have provided here two extra configuration files for such a purpose:

```
min-mem.conf
pr-mem.conf
```

Note:
1) Make sure that in all .conf files you provide correct names for protein coordinates and structures, i.e., that you need to set file entries for structures, coordinates as all_sim.psf and all_auto_combined.pdb, respectively. For the simulations involving membrane like the current case, please include pace-reopt.prm, martini_v2.0_lipids.prm and martini_v2.0_ions.prm as parameters.

2) To keep proteins from being distorted by initial setup, harmonic constraints are applied to initial positions of proteins. This is done by a section in the above .conf files which are:

```
constraints on
consexp 2
consref    all_auto_combined.pdb
conskfile  all_auto_combined.cnst
conskcol B
```

Here all_auto_combined.pdb contains the coordinates for reference structures and all_auto_combined.cnst tells NAMD which part of systems to be constrained. To generate the .cnst file, do the following:

```
vmd -dispdev text -e PROGPATH/pickcons.tcl
```

Before using `pickcons.tcl`, please make sure that you have made the following change to the script:

```
mol new all_sim.psf
mol addfile all_auto_combined.pdb
…
$a writepdb all_auto_combined.cnst
…
```

3) Please follow the order (in terms of `.conf` files used) below to perform pre-equilibrium simulations:

| | |
|---|---|
| `min-mem.conf` | ; minimization with protein constrained |
| `pr-mem.conf` | ; simulation with protein constrained |
| `min-nocons.conf` | ; minimization |
| `prod.conf` | ; production run |