GROMACS-LS: Computing the Local Stress Tensor from MD Simulations

Juan M. Vanegas and Alejandro Torres-Sánchez
juan.m.vanegas@gmail.com
torres.sanchez.a@gmail.com
Universitat Politècnica de Catalunya-BarcelonaTech,
Barcelona, Spain.

August 21, 2015

1   Introduction

This is a manual for GROMACS-LS, a custom GROMACS version to compute the local stress tensor in 3D from a molecular dynamics simulation. This program computes the local stress based on the Hardy stress definition (see Ch.8 in Tadmor & Miller (2011) **Modeling Materials: Continuum, Atomistic and Multiscale Techniques**). The atomic virial stress (A. P. Thompson et al., J. Chem. Phys. 131, 154107 (2009)) definition was also recently added. Methodological details of the implementation can be found in:

Importance of Force Decomposition for Local Stress Calculations in Biomembrane Molecular Simulations by J. M. Vanegas, A. Torres-Sanchez, and M. Arroyo. *J. Chem. Theor. Comput.* 10, 691-702 (**2014**).

Examining the Mechanical Equilibrium of Microscopic Stresses in Molecular Simulations by A. Torres-Sánchez, J. M. Vanegas, and M. Arroyo. *Phys. Rev. Lett.* 114, 258102 (**2015**).

Geometric Derivation of the Microscopic Stress: A Covariant Central Force Decomposition by A. Torres-Sánchez, J. M. Vanegas, and M. Arroyo. In preparation.

We have patched the GROMACS source code (v4.5.5) at various locations in the routines that calculate particles forces and velocities. Our patches rely heavily on the code structure and functions implemented in a previous local pressure code (obtained from http://repo.or.cz/w/gromacs.git/shortlog/refs/heads/release-4-5-localpressure and ftp://ftp.gromacs.org/pub/tmp/gromacs-4.0.2_localpressure.tar.gz). This previous code implements the methodology outlined in

3D Pressure Field in Lipid Membranes and Membrane-Protein Complexes by S. Ollila et al. *Phys. Rev. Lett.* 102, 078101 (**2009**).

1.1   Highlights of GROMACS-LS

- Decomposition of multi-body potential forces using:
    1. Covariant central force decomposition (cCFD) (Torres-Sánchez A. Vanegas, J. M. and Arroyo, M. (Phys. Rev. Lett., 2015)).

2. Non-covariant central force decomposition (nCFD) (N. C. Admal and E. B. Tadmor; J. Elast. 100, 63-143, 2010).

3. Goetz-Lipowsky decomposition (GLD) (R. Goetz and R. J. Lipowsky; J. Chem. Phys. 108, 7397-7409, 1998.).

4. Method of planes (Heinz, A. Paul W. and Binder K. Phys. Rev. E. 72 066704 (2005)).

- Ability to output the total or individual contributions to the local stress such as those from vdw, electrostatics, angles, and others.

- Finite discretization over a rectangular grid using tri-linear weight functions, which result in smoother stress fields and also make the discretization exact regardless of the grid size.

- Consistent treatment of forces arising from bond constraints (LINCS, SETTLE and SHAKE algorithms), which produces constant profiles for the normal component (perpendicular to the membrane plane) of the stress. This is necessary from an equilibrium stand-point and was an unresolved problem for stress profiles obtained from atomistic membrane simulations.

- Virial stress per atom as defined in A. P. Thompson et al., J. Chem. Phys. 131, 154107 (2009).

If you publish results using GROMACS-LS, we kindly ask you to cite the papers by Vanegas et al., Torres-Sánchez et al., and Ollila et al., which we recommend you to read before continuing with the rest of the manual. In these papers we show the main differences between the different definitions of the stress. We show that the virial stress per atom and the stress from the method of planes do not satisfy balance of linear momentum, while the stress from cCFD, nCFD and GLD do. We also show that the GLD and the stress from the decomposition on geometric centers lead to non-symmetric stresses, which therefore do not satisfy balance of angular momentum. The two flavors of the CFD do satisfy both balance of linear and angular momentum by construction. However, for potentials beyond 4-body, such as CMAP, the nCFD leads to unphysical stresses. We therefore suggest that the cCFD definition is preferred in any case.

The types of interaction potentials that are included in stress calculations and have been tested are the following:

- Bonds

- Angles

- Dihedrals (proper, improper, and Ryckaert-Bellemans)

- Constraints (SETTLE and LINCS, SHAKE is included but has not been tested)

- Van der Waals

- Coulomb (plain cut-off and reaction-field)

- CMAP

As such, if you use this code to analyze a system that may use other types of potentials, we cannot guarantee that it will be correct. Please thoroughly check the output of the program and compare to known quantities such as the total pressure. For such comparisons you should first obtain the total virial pressure in double precision. To do this, reanalyze the trajectory (i.e. `mdrun_d -rerun`) using a double precision version of GROMACS and recompute the pressure with `g_energy_d`.

1.2 Summary of Recent Changes

The latest version of the code has undergone some significant changes that may affect its usage compared to previous versions:

- The code now requires an external LAPACK library. This is need to decompose the higher order potentials such as CMAP, and it also makes the force decomposition much more robust.

- In order to be more consistent with the naming and avoid confusion, we have changed all references from pressure to stress. This affects the naming of programs

  `mdrun_LP` is now `mdrun_LS`

  and input flags

  `-lpgridx` is now `-lsgridx`

- With the addition of new force decompositions the `-lsfd` has new options: `ccfd` (covariant central force decomposition, default), `ncfd` (non-covariant central force decomposition), `gld` (Goetz-Lipowsky decomposition), or `mop` (method of planes). We recommend to use `ccfd`.

- A new flag (`-lssa`) has been added to select whether to calculate the spatial IKN stress as before (option `spat`, default) or the virial stress per atom (option `atom`)

- The CMAP 5-body potential (used in combination with the CHARMM force-field) is now included in the decomposed potentials.

- We now include a utility called `tensortools` which replaces the individual c utilities `bin2xvg`, `bin2ncdf`, etc.

2 Installation

GROMACS-LS is based on GROMACS version 4.5.5. While the vanilla 4.5.5 version can be compiled using both the autoconf tools and CMake, this custom GROMACS-LS version can only be compiled with CMake.

External requirements: FFTW3, LAPACK

You will need to install a recent version of the FFTW3 library (in double precision and as shared library), CMake, and LAPACK (http://www.netlib.org/lapack). The LAPACK library needs to be installed as a shared library (.so). Compiling LAPACK with the autoconf tools can sometimes be a hassle, but recent versions can be easily compiled and installed with CMake.

Installing LAPACK with CMake

```
$> tar -zxvf lapack-3.5.0.tgz
$> cd lapack-3.5.0
$> mkdir build
$> cd build
$> ccmake ../
```

Once the `ccmake` dialog comes up, press c to begin the initial configuration. Change the `BUILD_SHARED_LIBS` flag to ON and modify the `CMAKE_INSTALL_PREFIX` variable to set the target installation directory. After `ccmake` quits, you can follow the standard linux installation commands of `make` and `make install`.

If you install FFTW3 and LAPACK with a linux distribution such as Ubuntu or Fedora, you will also need to install the development (header) packages. If you have installed the FFTW3 or LAPACK libraries in a non-standard location (i.e. other than /usr/lib or /usr/local/lib), then before running `ccmake` you should export the variables `FFTW3_ROOT_DIR` and `CMAKE_PREFIX_PATH`, e.g.

```
export FFTW3_ROOT_DIR=/path/to/fftw3
```

```
export CMAKE_PREFIX_PATH=/path/to/lapack
```

Now the installation of the custom GROMACS-LS package

Installing LAPACK with CMake

```
$> tar -zxvf gromacs-4.5.5-ls-5.0.tar.gz
$> cd gromacs-4.5.5-ls-5.0
$> mkdir build
$> cd build
$> ccmake ../
```

Once the `ccmake` dialog comes up, press c to begin the initial configuration, and then modify the `CMAKE_INSTALL_PREFIX` variable to set the target installation directory. Use the toggle feature of `ccmake` (press t) and check that the paths to the FFTW3 and LAPACK libraries are correct.

Do not modify other variables unless you KNOW what you are doing. By default, the code is compiled in double precision (this is absolutely necessary), it cannot be compiled with MPI or GPU support, and all programs have the suffix _LS. Press c again and then g to generate the necessary Makefiles.

After `ccmake` quits, you can follow the standard linux installation commands of `make` and `make install`.

In addition to the GROMACS-LS code, we have also included two python programs:

`LStensor.py` and `tensortools`

`LStensor.py` contains a number of modules that can be imported into other python programs for the analysis of the data produced with the GROMACS-LS software. The `tensortools` utility provides an easy to use command line interface to most of the modules available in the `LStensor.py` file. These two files are installed along with the other GROMACS-LS files in the bin folder. We discuss these tools in more detail in Section 3.2.

3 Using the Code

3.1 Analyzing an MD Trajectory

For calculating the stress tensor in a MD simulation, we need a trajectory file (`.trr`) that contains both positions and velocities at the same points in time. In this section we show the steps you need to follow to calculate the stress tensor and illustrate it in the particular case of a POPE membrane.

As individual components of the system may drift over time, e.g. a membrane may drift in the simulation box, it is a good idea to first center a given group of atoms (in this case, the POPE bilayer) in the center of the simulation box. To do so, we use the tool `trjconv_LS` with the original trajectory, `traj.trr`:

Center the molecule into the simulation box before analyzing the stress

```
$ trjconv_LS -f traj.trr -o traj_centered.trr -n index -center -s topol.tpr
```

```
(...)

Select group for centering
Group     0 (          System) has 11600 elements
Group     1 (            POPE) has  2600 elements
Group     2 (           Water) has  9000 elements
Select a group: 1
```

Note that this trjconv_LS is a modified version of the vanilla one that places the center of mass, and not the geometrical center as it is done by the original, of the selected index group at the center of the simulation box. You must always pass a .tpr file as part of the input, or the program will return a segmentation fault without an error message. This new traj_centered.trr is the trajectory we will use to analyze the stress.

The 3D stress tensor is obtained by "rerunning" the trajectory with the mdrun_LS. Since the -rerun option of mdrun_LS outputs new .log, .edr, and .trr files, we suggest you first create a new folder and analyze the trajectory within it:

Computing the stress with mdrun_LS

```
$ mkdir stress
$ cd stress
$ mdrun_LS -s ../topol.tpr -rerun ../traj_centered.trr
```

To save space you can recreate the .tpr file by setting all the output nst* variables to 0 in the .mdp file:

```
nstxout = 0
nstvout = 0
nstlog = 0
nstenergy = 0
nstxtcout = 0
```

If your simulation was performed with PME or another long-range electrostatic method, you will need to create a new .tpr file with the coulombtype set to Cut-off or Reaction-field and a cutoff radius of at least 2.0-2.2 nm (see the Ref. 2 for more details). Note that this does not change your original trajectory but the way forces are calculated when analyzing each frame.

As with other GROMACS tools, mdrun_LS displays a help text running mdrun_LS -h. mdrun_LS has several special options for analyzing the stress:

mdrun_LS special options

```
Option      Filename   Type          Description
----------------------------------------------------------
-ols localstress.dat   Output        Generic data file

Option       Type   Value   Description
---------------------------------------------------
-localsgrid  real   0.1     Spacing for local pressure grid (default = 0.1 nm)
-lsgridx     int    0       Set the local pressure grid size in the x
                            direction (default use box[XX][XX]/localsgrid)
-lsgridy     int    0       Set the local pressure grid size in the y
                            direction (default use box[YY][YY]/localsgrid)
-lsgridz     int    0       Set the local pressure grid size in the z
                            direction (default use box[ZZ][ZZ]/localsgrid)
-lscont      string all     Select which contribution to write to output
                            (default = all): all, vdw, coul, angles, bonds,
                            dihp, dihi, dihrb, lincs, settle, shake, cmap, vel
```

5

```
−lsfd       string ccfd   Select the type of force decomposition to be
                           used: ccfd (covariant central force
                           decomposition, default), ncfd (non-covariant
                           central force decomposition), gld (Goetz-Lipowsky
                           decomposition), or mop (method of planes)
−lssa       string spat   Select the type of stress to calculate: spat
                           (spatial stress from IKN theory, default), atom
                           (stress per atom)
```

With the flag `−ols` one selects the output file where the stress tensor is stored. The size of the discretization grid is given by several parameters. If your system was simulated at constant pressure, the program takes the size of the box (box[XX], box[YY], box[ZZ]) from the `.tpr` along with the desired grid spacing (with `−localsgrid`), and calculates the integer size of the grid and the real grid spacing, e.g. if the box size in the x direction is 5.37 nm and the given grid spacing is 0.1 nm, then the grid size in x is 53 and the real grid spacing becomes 0.1013 nm. The grid size in each dimension can also be overridden with the `−lsgridx`, `−lsgridy`, or `−lsgridz`. So if you are interested in calculating a stress profile along a given dimension, e.g. z, then you can use these different flags together:

> **Setting the grid size**
>
> ```
> $ mdrun_LS −s ../topol.tpr −rerun ../traj_centered.trr −localsgrid 0.1 −lsgridx 1 −lsgridy 1
> ```

At the moment, only systems simulated with rectangular boxes can be analyzed. mdrun_LS can also return the total or individual contributions to the stress with the `−lscont` flag. At the moment the available contributions are van der Waals, Coulomb, angles, bonds, proper dihedrals, improper dihedrals, Ryckaert-Bellemans dihedrals, constraints (LINCS, SHAKE, or SETTLE), cmap, and kinetic.

Finally, the `−lsfd` flag allows to select one of two different decompositions for the forces between particles that enter into the potential part of the stress. We advice you to use the ccfd option (set by default in the program) that always provides a symmetric stress, thereby satisfying the balance of angular momentum from a continuum viewpoint, and physically meaningful regardless of the number of particles involved in each additive potential of the force field. The ncfd is equivalent to the ccfd except for 5-body potentials. On the other hand, the gld option allows you to compute the stress from the decomposition by Goetz and Lipowsky (J. Chem. Phys. (1998) 17, 7397-7409), which has been widely used in biomembrane simulations. Please, take a close look to references 2 and 3 for a more detailed discussion about the differences between both decompositions.

The outputted `.dat0` file is a simple binary file in double precision with the following format:

> **Format of the `.dat0` binary file**
>
> ```
> precission  = sizeof(int)*1
> box         = sizeof(double)*9
> gridx       = sizeof(int)*1
> gridy       = sizeof(int)*1
> gridz       = sizeof(int)*1
> stresstensor = sizeof(double)*9*gridx*gridy*gridz
> ```

3.2 `tensortools`: Python scripts to post-process the stress fields

The command mdrun_LS produces binary `.dat0` files that need to be post-processed for analysis. To make life easier, we have included two tools `tensortools` and `LStensor.py` that can load, transform and save these binary files in useful formats. These python files are directly installed to the binary folder of the GROMACS-LS distribution. These python utilities require recent versions of the Numpy and Scipy packages, and optionally the ProDy package (http://prody.csb.pitt.edu/) when using some of the stress per atom options.

3.2.1 LStensor

The `LStensor.py` file defines what we call the LStensor object and corresponding analysis functions, which can be used to post-process the data from the GROMACS-LS program. This object can load IKN stresses as well as stresses per atom and convert these to a grid. It can also be used to compute the basic invariants of the stress as well as its divergence. It also provides a number of functions to load PDB files (stress per atom only) and to store files in NETCDF, plain txt, and PDB format.

3.2.2 tensortools

The other file, `tensortools`, is a command-line utility that can be used for a rapid post-processing of the binary stress files. It uses the functions defined in `LStensor.py` and allows easy manipulation of the data. The main options are:

Options for tensortools (tensortools -h)	
`-h, --help`	show this help message and exit
`-f [input.bin] [[input.bin] ...]`	Single or multiple binary input files (must all have the same grid size)
`-v []`	verbose
`--irep [spat]`	Input data representation, values: spat (spatial stress/density distributed on a grid, default), atom (stress per atom)
`--pdb [[structure.pdb]]`	PDB file with atom coordinates (FOR STRESS PER ATOM ONLY)
`--itype [stress]`	Type of data such as density, stress tensor, or vector field: dens, stress (default), or vec
`--mif [avg]`	How to process multiple input files, values: avg (default) or sum
`--sym []`	FOR STRESS ONLY. Separates the resultant stress in its symmetric and antisymmetric parts, and creates two outputs with the subscripts _sym and _asym respectively
`--prof z`	Output a profile along a given dimension, values: x, y, or z (default)
`--integ z`	integrate out the stress profile along a given dimension, values: x, y, or z (default)
`--gf 1.0`	Process input through a gaussian filter of a given standard deviation (in units of the grid spacing, default=1.0)
`--gridsp 0.1 0.1 0.1`	(For atom to grid conversion) Grid spacing for atom 2 grid conversion (default 0.1, 0.1, 0.1 nm)
`--orep [spat]`	Output data representation, values: spat (spatial stress/density distributed on a grid, default), atom (stress per atom)
`--oformat [bin]`	Output format, values: bin (i.e. binary .dat0, default), nc (NETCDF), txt (default when using --prof), or pdb (stress per atom only, creates a separate file for each element in the tensor)
`-o output.bin`	Output file

Some examples of basic usage:

Averaging multiple input *.dat0 files
`tensortools -f file0.dat0 file1.dat0 file2.dat0 -o stress.dat0`

Conversion to NETCDF format

```
tensortools -f file.dat0 --oformat nc -o stress.nc
```

Compute a stress profile along the y axis and save it to a text file

```
tensortools -f file.dat0 --prof y -o stress.txt
```

Smooth data using a gaussian kernel with a standard deviation of 8 (in units of the grid spacing)

```
tensortools -f file.dat0 --gf 8.0 -o stress.dat0
```

Advanced usage:

Read the IKN stress from *.dat0 files, integrate out the z direction, smooth the data with a Gaussian filter of 25 (units of the grid size), and write the result in the file stress_gf25.nc

```
tensortools -f *.dat0 --inte z --oformat nc --gf 25 -o stress_gf25.nc -v
```

Read the stress per atom from *.dat0 files, load the PDB structure from struct.pdb, translate the data to a grid with spacing 0.01, 0.01, 10, apply a Gaussian filter of 25 and store the result in stress_atom_gf25.nc

```
tensortools -f *.dat0 --irep atom --pdb struct.pdb  --orep spat --oformat nc -o
     stress_atom_gf25.nc --gridsp 0.01 0.01 10 --gf 25 -v
```

Note that although the main purpose of `LStensor.py` and `tensortools` is to analyze the stress tensor binary files, it can also manipulate other binary files such as those outputted by the `g_density3D_LS` utility.

4 Some technical comments about cCFD and nCFD

In the new version of the code, computing any flavor of the central force decomposition (cCFD and nCFD) for a given potential V_I is done by solving the overdetermined system of equations

$$\boldsymbol{F}_I^\alpha = \sum_{\beta \in \mathcal{I}_I, \beta \neq \alpha} \boldsymbol{f}^{\alpha\beta} = \sum_{\beta \in \mathcal{I}_I, \beta \neq \alpha} \varphi^{\alpha\beta} \frac{\boldsymbol{r}^{\alpha\beta}}{|\boldsymbol{r}^{\alpha\beta}|}, \forall \alpha \in \mathcal{I}_I \tag{1}$$

where $\mathcal{I}_I$ is the set of particles interacting through the potential V_I, $\boldsymbol{f}^{\alpha\beta}$ are the terms of the force decomposition and $\varphi\alpha\beta$ are their respective magnitudes. If n_I, the number of particles involved in the potential, is equal or smaller than 4, this overdetermined system has a single solution and thus cCFD and nCFD coincide. For potentials beyond 4-body, such as the CMAP interaction, this systems becomes underdetermined with an infinite number of possible solutions. We have recently proposed a preferred CFD based on covariance arguments (Torres-Sánchez et al., in preparation), which we call cCFD. To compute this cCFD we first need to compute a solution to Eq. (1). We do this by calling the DGELSD function of the LAPACK library (http://www.netlib.org/lapack/explore-html/db/d6a/dgelsd_8f.html) which solves under- and overdetermined linear systems by computing the solution with minimum norm. The solution to this problem is a CFD, which we call nCFD (non-covariant central force decomposition). To compute the cCFD, nCFD is projected onto the tangent plane to the Shape Space, which is calculated through the derivative of Caley-Menger determinants. While nCFD depends on the parametrization of the potential and give spurious results, we have recently proven that cCFD is independent of the representation of the potential and gives physically relevant results (see Torres-Sánchez et al., for more details).

5   Frequently Asked Questions

Q1. How long does it take to analyze a trajectory with `mdrun_LS`? This depends on the size of your system and the grid size. Analysis time increases with a larger system and a finer grid. For a typical system with approximately 100,000 atoms and a grid spacing of 0.1 nm it can take 5-10 minutes (depending on your hardware) to analyze each frame in the trajectory. The code does not utilize any of the SIMD or GPU optimizations that the vanilla GROMACS uses and therefore is much slower.

Q2. `mdrun_LS` has been stuck on the first frame of the trajectory for hours, what's wrong? If the analysis is taking an unusually long time there may be something wrong with the input files or there may be an unknown problem with the code. You can check this by selecting to output only one of the components such as the kinetic one (using the `-lscont vel` flag), which should be done very quickly even for a large number of particles. If this fails there may be a problem with the trajectory, otherwise there may a particular problem with some of the interactions being calculated. You can try calculating some of the other individual components in order to narrow it down and contact us regarding a particular issue.

Q3. I get a segmentation fault when running `mdrun_LS` Make sure that mdrun_LS is run in a different folder than where the input trajectory is at, otherwise the program may inadvertently overwrite the input trajectory and cause the error.

Q4. Can the program be run in parallel? Not directly. The program must always be run serially. However, each frame in a trajectory can be analyzed completely independent of every other frame. One method to parallelize the analysis is to separate the trajectory into smaller chunks with equal number of frames (using `trjconv_LS -split`), and analyze each chunk separately. You can then average all of the resulting stress files together using the `tensortools` utility. The advantage of this method is that it always scales linearly although it takes a bit of work to set up. CAUTION! When doing this make sure to use the same `.tpr` file and same grid spacing to analyze each chunk of the trajectory, otherwise the grid sizes may differ and the average will fail.

Q5. Are the velocities needed? Absolutely! The kinetic contribution to the stress cannot be calculated without the velocities, and this can be pretty substantial contribution to the total stress.

Q5. Can I use an `.xtc` file instead of a `.trr` file? No. The `.xtc` file only stores the positions and not the velocities.

Q6. Does `mdrun_LS` use the positions and velocities from the `.tpr` file? No, mdrun_LS only uses the positions and velocities stored in the `.trr` trajectory. However, the initial box size, and therefore the grid size are determined from the values in the `.tpr` file.

Q7. Are electrostatic contributions calculated using PME included in the stress tensor? No. At the moment, electrostatic contributions calculated in reciprocal space are not included in the local stress calculation. If you ran your simulation with PME electrostatics, you must change the `coulombtype` to `Cut-off` and select an appropriate cut-off value (`rcoulomb`) in the `.mdp` file. Then, you can create a new `.tpr` with `grompp_LS`. We recommend to use an `rcoulomb` value of at least 2.0 - 2.2 nm. CAUTION! The electrostatic contribution to the stress is very computationally expensive due to the large number of pair interactions in a typical simulation. The number of interactions increases cubically with the cut-off radius. In our experience, using a cut-off beyond 3.0 nm is computationally unfeasible for normal systems. See Vanegas et al. (JCTC 2014) for more details.

Q8. What is the relation between stress and pressure? The pressure tensor, P_{ij}, has the opposite sign as the stress tensor, σ_{ij}, so $P_{ij} = -\sigma_{ij}$

Q9. The total stress obtained with `GROMACS-LS` does not match the pressures obtained with `g_energy` First, if you simulated your system with PME electrostatics and re-analyze with a Cut-off, then the total stress values will never match the virial pressures calculated by GROMACS unless you use an infinite cut-off. See Vanegas et al. (JCTC 2014) for more details. Also, the values obtained from the single precision `.edr` file generated during the simulation run are not accurate. To do a meaningful comparison you need

to reanalyze the trajectory (i.e. `mdrun_d -rerun`) using a double precision version of GROMACS and recompute the pressure with `g_energy_d`.

Q10. Is it normal for the local stress values to be so large? The local stress can easily reach values up to several thousand bars depending on the system and types of interactions (see Figure 1).
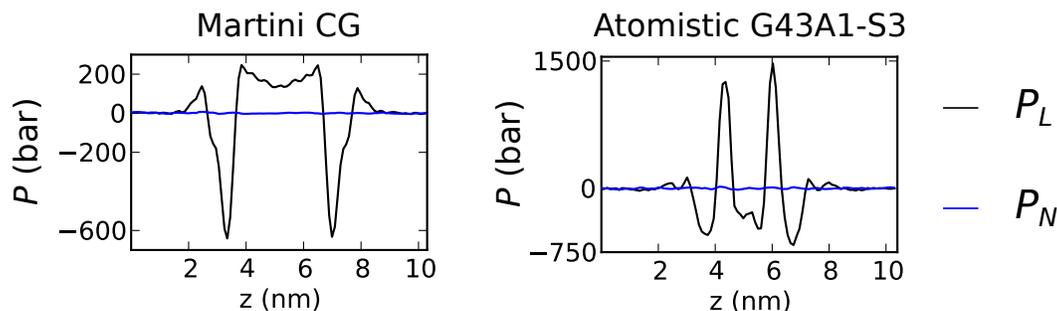
POPE pressure profiles

Figure 1: Comparison of stress profiles ($P_N = -\sigma_{zz}$ and $P_L = -(\sigma_{xx} + \sigma_{yy})/2$) for bilayer systems simulated with coarse grained and atomistic force-fields.

Q11. How often should I save frames to the trajectory? How long does the simulation need to be? The answer to this may differ depending on the system. For lipid bilayer systems we typically save frames to the trajectory every 5 or 10 ps, and analyze a 100 ns trajectory. Figure 2 shows the convergence of the stress profile depending on the number of frames used.

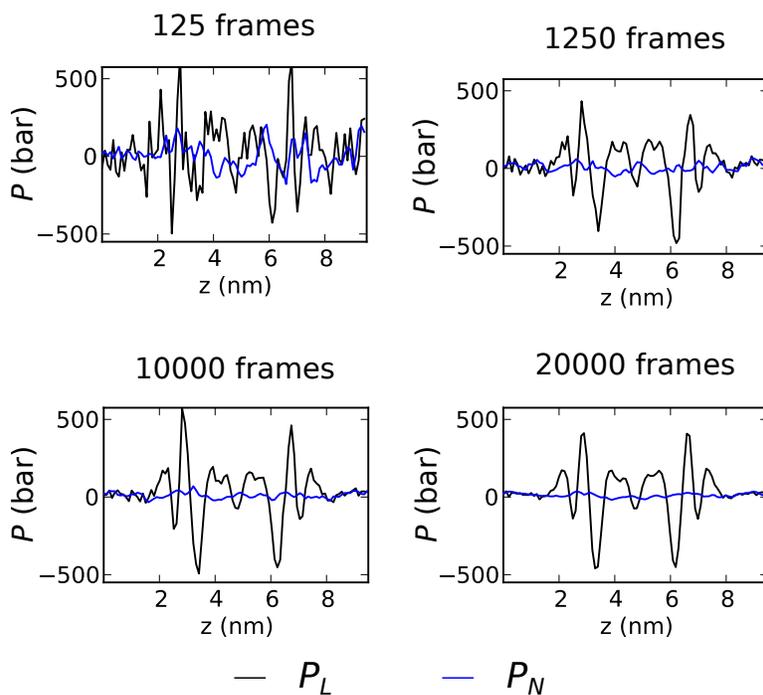## Evolution of a DPPC profile with the number of frames analyzed

Figure 2

Q12. Why is the normal component (e.g. σ_N) of the stress profile of my bilayer system not zero? Mechanical equilibrium dictates that the component of the stress along the direction normal to a lipid bilayer should be constant. The average value of this constant component should match the imposed value of the simulation pressure for the given dimension, which may or may not be zero. The important point is that it should be constant. If σ_N is not constant (beyond reasonable noise) then there could be several problems with the simulation. First, check that the lipid membrane is fully equilibrated. For our systems composed of 200 lipids (100 lipids in each leaflet) in the liquid phase we equlibrate for 400 ns before running the data collection period. Second, check the `comm_grps` option in the `.mdp` file. Bilayer systems are often simulated with center of mass motion removal done on different groups separately, e.g. `comm_grps = BILAYER SOL`. However, doing this will change the internal mechanical behavior of the system. For local stress analysis this `grompp` option should always be set `comm_grps = System` during the simulation run.

Q13. Can the 3D stress tensor be loaded into other programs for visualization? Yes. The easiest way to do this is to first convert the binary stress file to the NETCDF format using the `tensortools` utility. You can then load this generic NETCDF file into programs such as UCSF Chimera or ParaView. Figure 3 shows an example of visualizing a single element of the stress tensor in ParaView.
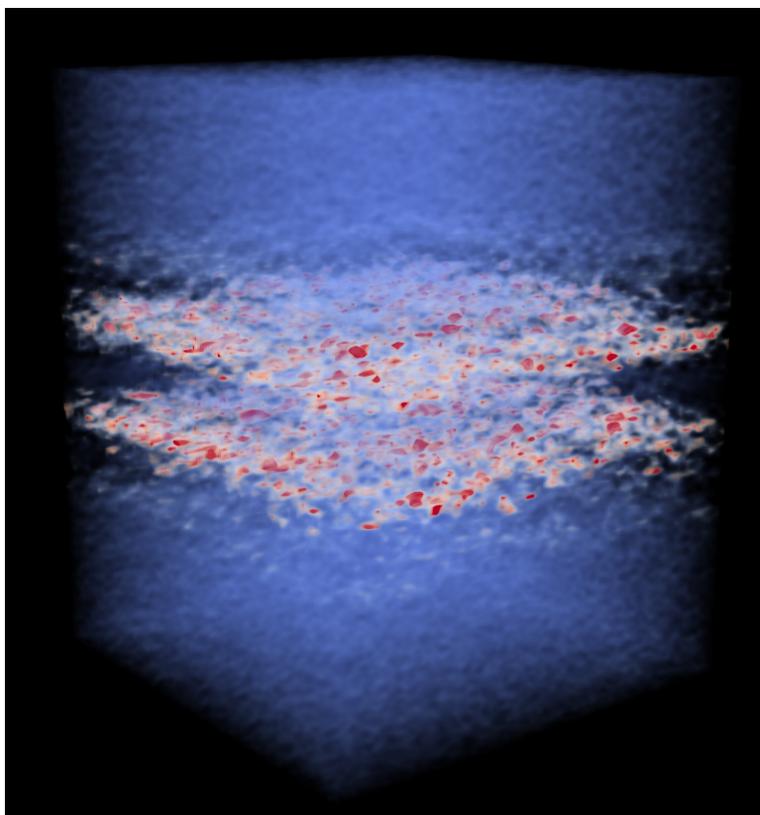

Figure 3: 3D stress tensor for a bilayer system visualized in ParaView

Q14. Can I use GROMACS-LS (v.4.5.5) to analyze a trajectory created with a new version of GROMACS (4.6 or 5.x) Depends. If your simulation uses features (i.e., potentials and such) that are also available in the 4.5.5 version, then it should not be a problem. However, there is a big change in the newer versions of GROMACS with regards to the non-bonded cut-off schemes (see http://manual.gromacs.org/documentation/5.1/user-guide/cutoff-schemes.html). This can affect certain non-bonded interactions and forcefields that use twin-range cut-offs (e.g., GROMOS). To ensure compatibility with GROMACS-LS, you can explicitly use the group cut-off scheme by setting `cutoff-scheme = group` in the `.mdp` file.