# Introduction to the analysis of a frozen hexadecane-containing system

The analysis of a frozen hexadecane-containing system is a multistep procedure which will be explained in the following text. In general, the first step is to identify all the frozen structures (crystallites) in the system. This includes: (i) ignoring the still too mobile molecules and (ii) in a lot of cases isolating multiple crystallites with different orientations with respect to each other formed in the same system. When all crystallites are identified, the radial distribution functions (RDF) and the deuterium order parameters of each crystallite can be calculated with GROMACS.

After the successful extraction of the crystallites, they will probably appear split because of the projection in the periodic box. For some of the analyses, for example fitting the eigenplane of the crystallite, this is detrimental. Hence, a script has to be used to construct the real-space coordinates of the whole crystallite from the periodic images. With the successful execution of the latter script, the evolution of the tilt angle of the molecules relative to the plane of the crystallite as well as the time-averaged tilt angle for each molecule can be calculated.

As another analysis, 2D Voronoi analysis is performed on a grid of points determined by the intercepts between the vectors constructed from the C3 and C13 atoms in the hexadecane molecules and/or the surfactant alkyl chains and the calculated eigenplane of the crystallite.

# Identification and separation of the different crystallites in the system

The description is prepared on the example of a bulk hexadecane (HEX) system containing 440 molecules and of a hexadecane/surfactant/water (HEX/Surf/water) system containing 494 HEX, 108 C16(EO)2 and 2778 water molecules, respectively. The entire duration of the sample trajectories is 500 ns. C3 and C14(or C13) are chosen as reference atoms spanning the entire molecule on the one hand and being relatively rigid, i.e. away from the mobile molecular termini.

1. Create a trajectory with whole molecules from the part you would like to analyze where the entire system is frozen (last 20 ns of the simulation in the example for steps 1-4 and the last 100 ns for the rest)

gmx trjconv -f *XXX*.trr -s *XXX*\_0ns.tpr -b 480000 (time in ps 20 ns before the end of the trajectory) -pbc whole -o *XXX*\_whole.trr

2. Take the positions (x, y, and z coordinates) of the C3 and C14 atoms of every HEX or Surf molecule.

First create an index file containing only the C3 and C14 atoms of the molecules:

gmx make\_ndx -f *XXX*.gro -o C3-C14.ndx

In the interactive menu delete all default groups with “del 0-N” (N is a number designating the last existing group) and then use the option “a C3 C14” and type q followed by enter.

Then:

gmx traj -f *XXX*\_whole.trr -s *XXX*.tpr -n C3-C14.ndx -ox *C3-C14\_coord\_whole*.xvg

Here you can use any .tpr with the correct topology of the entire system. A .gro file will work as well.

3. All rows with @ and # signs must be deleted from the beginning of the C3-C14\_coord\_whole.xvg file! Otherwise, the script used in the next step will give an error. This can be done effortlessly in the terminal with **sed -i '/@/d; /#/d' \*.xvg** . *The command works directly on all .xvg files present in the folder, so making a copy of them in another folder before executing the command is mandatory!*

Use **slope-script.exe** on a Windows PC with the output file from the previous step to calculate the projections of the C3-C14 vector in the XY (dY/dX), YZ (dZ/dY) and ZX (dX/dZ) planes. (After double-clicking on the program, a graphical window will appear with a “*Browse”* button. Any number of files with coordinates can be selected and the script will create the same number of output files with the appropriate names.) The output is a .csv file(s) with the time-averaged projections and standard deviations, which are converted into slopes of the vector in each plane measured in degree. Each row contains the data for one molecule.

4. Create a 3D graph from the data points in the .csv file and write down the ranges of values along each axis for each cluster formed on the plot, which is disjoint from the rest. For example, the first cluster might have a range of values for the YX slope in the interval 43-50°, for ZY 60-76° and so on. Each crystallite must have a unique combination of slopes ranges for the three projections.

5. Use the row numbers to identify the molecules in each slopes range from the previous step. The easiest way to do it is in Excel. In the case of bulk hexadecane, the row number and the residue ID (resid) coincide. If the system has multiple molecular components, the user has to assign manually the residue number of each HEX molecule or Surf tail from the .gro file to each row in Excel. When all crystallites are identified, create a .txt file where each row contains the resids of the molecules/tails in a single crystallite. Write down the entries for the different crystallites on separate lines. Use **crystallite\_extract.py** to automatically create trajectory and .gro files for each crystallite. The script has 3 variables that have to be changed: **input\_top**: accepts a .gro file, **input\_trr**: accepts a .trr file and **cryst\_sel**: the .txt file with the resids of each crystallite. The .gro and .trr files need to be for the whole system.

6. (may be skipped if only RDFs and order parameters are needed) As mentioned in the Introduction, the molecules in most of the crystallites need to be brought in their real-space positions. This is done with **fix\_layers\_trj.py**. Here, 4 variables need to be changed: **input\_top**: accepts a .gro file, **input\_trr**: accepts a .trr file, **straight\_mol**: a molecule from the provided .gro file that is straight (all-trans conformation) and **vacuum\_width**: the width of the space (in Å) between the groups of molecules that need to be imaged across a direction of the periodic box (a few trial runs may be needed to tune this value, 5.2 Å for the largest crystallite of HEX/Surf/water). For crystallites that are not well ordered yet, e.g., contain mobile molecules, the procedure can fail at this step. This can be fixed by removing these mobile molecules with **non\_frozen\_deleter.py**. An **input\_top** and **input\_trr** should be changed to the .gro and .trr files obtained from step 5. The variable **mobility\_criterion**: controls the criterion for a molecule to be considered mobile. For example, **mobility\_criterion** = 2 means: if at any point in the trajectory a molecule has a distance between its C3 and C14 atoms of 2 Å less than the average C3-C14 distance for the crystallite, it is removed from it.

After removing the mobile molecules, the **fix\_layers\_trj.py** needs to be executed again and should work correctly.

***Intermediate outcome: .gro and .trr files of each crystallite and lists of resids of the molecules therein.***

# Fitting of the eigenplane of a crystallite and calculation of tilt angles

7. Using the trajectory and .gro files obtained in step 6 for a single crystallite, its plane can be calculated and then the evolution of the tilt angle in the crystallite and the time-averaged tilt angle of the molecules in the crystallite can be determined. This can be achieved with **PCA\_plane\_fit\_trj.py**. Here, only the **input\_top** and **input\_trr** variables need to be changed accordingly. After successful completion, the script produces two .csv files, one for the evolution of the tilt angle and the other for the time-averaged tilt angle. The first one, its name ending in *avg\_angle\_over\_time* contains the evolution of the tilt angle (tilt angle averaged over all molecules at a given time + standard deviation), and the other one with name ending in *indiv\_mol\_angle* provides time-averaged values for each molecule with standard deviation.

NB: When calculating the tilt angle for the entire trajectory (starting from the liquid state), a modified script HHHH.py should be used! The only difference there is that the eigenplane of the crystallite is fixed to the one averaged over the last 100 ns (where the system is frozen).

# Preparing files for Voronoi analysis

8. To perform 2D Voronoi tessellation, all key points (one per molecule) must lie in the same plane. Since the molecules move due to thermal fluctuations, they never lie perfectly in the same plane. For this reason, the intercept point between each molecule in the crystallite, represented by the vector C3-C13, and the crystallite plane is calculated with **point\_of\_intercept.py**. Here, only the **input\_top** and **input\_trr** variables should be changed accordingly. The .gro and .trr files are the ones obtained in step 6. The output .csv file is a data array with size (N, 3M+1), where N is the number of trajectory frames and M is the number of molecules in the crystallite. The first column contains the number of the frame (starting from 0) and the subsequent triples of columns contain the x, y, and z coordinates of the key points of each molecule.

# Analysis of the second principal axis

The angle between the second principal axis of each molecule and the x and y axes of the coordinate system gives information about their orientation in space and with respect to each other. This property adds to the identification of the solid-state phase of the crystallite. The **principal\_vectors-trj.py** calculates the evolution of the second principal axis and creates a .mp4 movie. Each frame is a plot with the orientations of all molecules (represented as vectors) in the crystallite. The positions of the molecules are fixed to correspond to the final frame of the analyzed trajectory. This is done since in the part of the simulation where the molecules are in the liquid state, their positions cannot be properly assigned to a plane. The script needs two .gro files (**input\_top** and **input\_top2**) and a .trr file (**input\_trr**). **input\_top** and **input\_trr** are again the files obtained in step 6. The variable **straight\_mol** has the same meaning as described in step 6 while **fig\_legend** is the text that will be displayed in in the legend of the movie. **input\_top2** is a modified version of **input\_top**, created by **rot\_gro\_for\_p2.py**. The latter script will rotate the systems so that the long axes of the molecules are parallel to the z axis of the coordinate system. Here, **input\_top** and **straight\_mol** are the same as **input\_top** and **straight\_mol** in **principal\_vectors-trj.py** .

# RDFs and deuterium order parameters

The RDFs and deuterium order parameters can be calculated using GROMACS with the **gmx rdf** and **gmx order** modules. Example commands are as follows:

gmx rdf -f *crystallite*.trr -s *crystallite*.tpr -n *crystallite\_for\_rdf*.ndx -bin 0.001 -ref 0 -sel 0 -selrpos res\_com -seltype res\_com -o *name\_of\_the\_output\_file.xvg*

gmx gmx order -f *crystallite*.trr -s *crystallite*.tpr -n *crystallite\_for\_order*.ndx -os *name\_of\_the\_output\_file.xvg* -permolecule

The .trr file is that of the isolated crystallite obtained in step 5 and the .tpr file is created with the .gro file created in step 5. To create the .tpr, a force field topology is required that must be the same as the one used for the simulation. The .mdp file may be any (even an empty file!).

gmx pdb2gmx -f *crystallite*.gro -p *crystallite*.top

gmx grompp -f *any*.mdp -c *crystallite*.gro -p *crystallite*.top -o *crystallite*.tpr

For each analysis, a different index file is needed. In case of bulk HEX, an index file is needed only for the order parameter. For the HEX/Surf/water, an index file with only the hexadecanes and the alkyl chains of the surfactants is created.

gmx make\_ndx -f *crystallite.*gro-o *chains\_only.*ndx

Select and merge the numbers of the groups HEX and C16, e.g. 2 | 3, and then type del 0-Gmax; Gmax is whatever number is before the newly created HEX\_C16 group.

The order parameter is calculated for the individual carbon atoms in the chains. So, the indices of each atom: C1, C2, C3, etc., must be in a separate index group.

gmx make\_ndx -f *crystallite.*gro-o *order.*ndx

First, delete all groups in the index file and then create a separate group for each atom, e.g, “a C1”, enter, “a C2”, enter, and so on.

The RDF files can be plotted in a visualization software directly while the order parameter should first be averaged for each atom over all molecules. For example, in hexadecane there are 14 carbon atoms that have an order parameter and in the output files there will be 14 columns. Average the values in each column and calculate the standard deviation. In a relatively ordered system, the average should stay close to constant and the standard deviation should be relatively small, while for more disordered crystallites the average value will vary, especially at the terminal groups, and the standard deviations will be larger.