

# Automating the GROMACS analysis tools on HPC systems

Olivier Fisette [olivier.fisette@usask.ca](mailto:olivier.fisette@usask.ca)  
Advanced Research Computing, ICT  
University of Saskatchewan

WestDRI Webinar  
2023-03-14  
CC BY 4.0

# Introduction

- The GROMACS software package includes many programs.
  - An MD engine: `gmx mdrun`
  - Tools to prepare simulations: `pdb2gmx`, `grompp`, ...
  - Post-processing and analysis tools: `trjconv`, `rms`, `energy`, ...
- The MD engine is very well suited to HPC environments.
  - Highly optimised for parallel computations
  - Easy to use in scripts submitted to the job scheduler
- The GROMACS analysis tools are more challenging on HPC systems.
  - They ask for input interactively.
  - They typically do not work in parallel.
  - They are hard to script (no interface for languages such as Python).

# Introduction

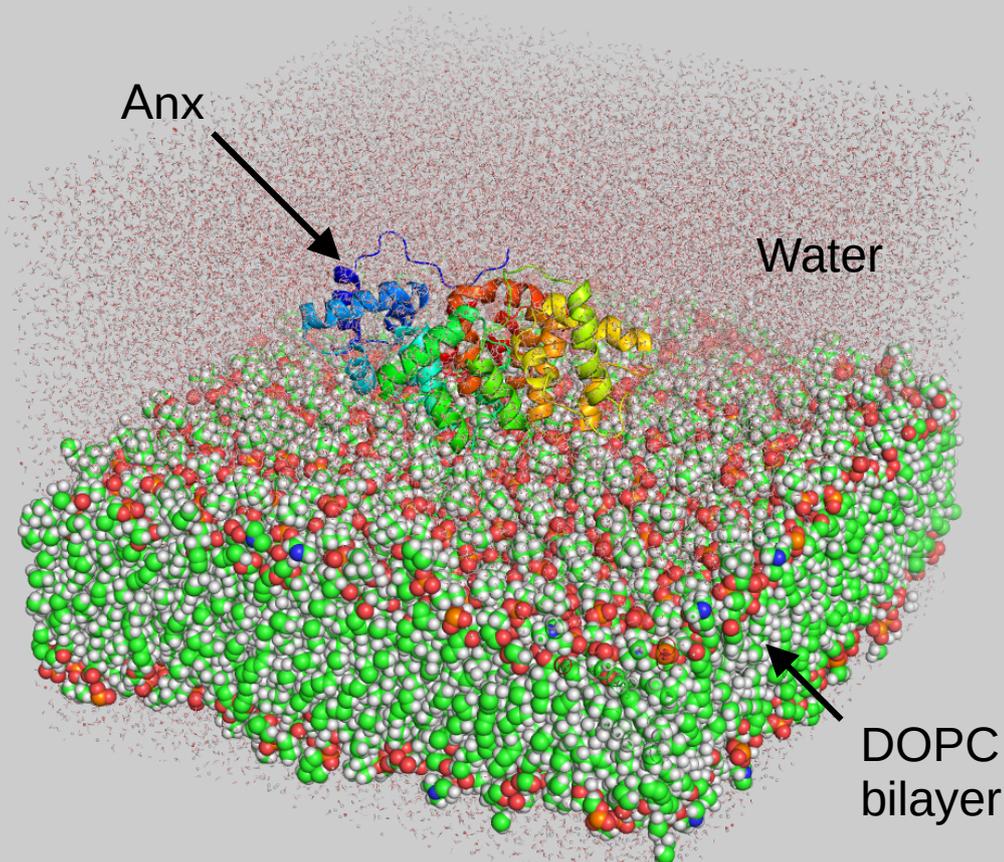
- The GROMACS analysis tools work well on a workstation.
  - Output files can be copied from the cluster to your PC.
  - This might be acceptable for small projects.
  - For large or numerous trajectories, this is limiting.
- Running the GROMACS analysis tools on the cluster is better.
  - No need to copy files back and forth
  - Access to much more processing power
- There are techniques to work around these limitations.

# Outline

1. Avoiding interactivity using input redirection
2. Running analyses in parallel with job arrays
3. Using job arrays with multi-dimensional input
4. Multiple short analyses in a single job with GNU parallel
5. Speeding up analyses using trajectory post-processing
6. MDAnalysis for GROMACS structures and trajectories
7. Generating GROMACS index groups with MDAnalysis

# Molecular system used for all examples

- O. Fiset, C. Pöslack, R. Barnes, J. M. Isas, R. Langen, M. Heyden, S. Han, L. V. Schäfer. *Hydration Dynamics of a Peripheral Membrane Protein*. *JACS*, 2016, 138, 36, 11526–11535. doi: [10.1021/jacs.6b07005](https://doi.org/10.1021/jacs.6b07005).
- Water diffusion is slowed down in the vicinity of a peripheral membrane protein, annexin B12.
- 14 annexin residues were used as probes in MD and NMR to measure water retardation.



100 ns, 1 trajectory, 10 ps / saved frame



1 ns, 10 trajectories, 0.1 ps / saved frame

# Following and repeating these examples

- All examples will be done on a cluster (Plato at USask) similar to Alliance clusters.
- All interactive commands are given in the slides.
- Pre-written scripts will be discussed extensively.
- All scripts are available in an archive.
  - WestDRI webinar page
  - <https://asteria.feralhosting.com/ribosome/gmxtools.tar.gz>
- The trajectories are not included in the script archive.
- These slides are available as a PDF.
  - WestDRI webinar page
  - <https://asteria.feralhosting.com/ribosome/gmxtools.pdf>

# Avoiding interactivity with input redirection

- We compute backbone RMSD, after a minimising fit to alpha carbons, for a short portion of an Anx trajectory.
- Using the GROMACS tools in an interactive session

```
ssh plato
salloc
cd ~/Anx_on_DOPC/simulations
ls
cd 2_MD
ls
module spider gromacs
module spider gromacs/2022.3
module load StdEnv/2020 gcc/9.3.0 openmpi/4.0.3
module load gromacs/2022.3
gmx rms -f traj_comp.xtc \
        -o Anx_backbone_RMSD_fit_to_Ca.xvg -e 5 -tu ns
3
4
...
exit
```

# Avoiding interactivity with input redirection

- The problem with scripts and interactivity

```
mkdir -p ~/Anx_on_DOPC/analysis/RMSD
cd ~/Anx_on_DOPC/analysis/RMSD
cp ~/gmxtools/01_RMSD.sh RMSD.sh
less RMSD.sh
sbatch --wait RMSD.sh
less slurm_RMSD.out
```

- The `gmx msd` program requested input interactively and stopped with an error because that input was not provided.

# Avoiding interactivity with input redirection

- We can use a separate text file to select the index groups.
- The text file is passed to `gmx rms` using the `<` redirection operator.

```
cp ~/gmxtools/02_selected_groups.txt selected_groups.txt
cp ~/gmxtools/03_RMSD.sh RMSD.sh
cat selected_groups.txt
less RMSD.sh
sbatch --wait RMSD.sh
less slurm_RMSD.out
(head; tail) < Anx_backbone_RMSD_fit_to_Ca.xvg
```

- The `<` redirection operator copies the contents of a file to the standard input channel of a program.

# Avoiding interactivity with input redirection

- Using numbers to select the index groups is not clear.
- Fortunately, the GROMACS tools also accept index group names.

```
cp ~/gmxttools/04_selected_groups.txt selected_groups.txt
cat selected_groups.txt
sbatch --wait RMSD.sh
less slurm_RMSD.out
```

# Avoiding interactivity with input redirection

- Having a separate file to select the index groups is a little confusing.
- We can use a Bash “here-document” to inline the index groups selection in the script.

```
cp ~/gmxtools/05_RMSD.sh RMSD.sh
less RMSD.sh
sbatch --wait RMSD.sh
less slurm_RMSD.out
```

- The << and <<- redirection operators start a here-document.
  - They are followed by an arbitrary token (e.g. EOF) that is repeated at the end of the here-document.
  - The text between the two tokens is the here-document. It is copied to the standard input channel of a program.
  - The <<- operator ignores tab characters at the start of lines. This allows indenting the text to make it more readable.

# Avoiding interactivity with input redirection

- Here-documents are the best way to select index groups in scripts.
- Some GROMACS tools have command-line options for selections.

```
gmx msd -sel [...]
gmx sasa -surface [...] -output [...]
```

- Most tools have no such options!

# Running analyses in parallel using job arrays

- We compute minimum distance time series between 14 Anx residues and the DOPC bilayer. (These residues were used as probes in NMR.)
- We have not one but 14 calculations to perform.
- We do these calculations in parallel.
- The first step is to prepare index groups for the 14 residues and the DOPC bilayer.
  - This step can be done on the login node.
  - We use a here-document with `gmx select`.

```
mkdir -p ~/Anx_on_DOPC/analysis/bilayer_aa_mindist
cd ~/Anx_on_DOPC/analysis/bilayer_aa_mindist
cp ~/gmxtools/06_make_mindist_index.sh make_mindist_index.sh
less make_mindist_index.sh
bash make_mindist_index.sh
less groups.ndx
```

# Running analyses in parallel using job arrays

- We start with a naive serial script.

```
cp ~/gmxttools/07_mindist.sh mindist.sh
less mindist.sh
sbatch mindist.sh
...
ls
...
scancel ...
```

- Analyses are done one at a time.
- This approach is much too slow.

# Running analyses in parallel using job arrays

- Our second attempt is a loop that submits multiple jobs.
  - This is an example of what not to do!
- We have two scripts.
  - One is run on the login node to loop over the residues and submit a job for each.
  - One job script to perform a mindist calculation for a given residue.

```
rm *.{sh,out,xvg}
cp ~/gmxttools/08_submit_mindist.sh submit_mindist.sh
cp ~/gmxttools/09_mindist.sh mindist.sh
less submit_mindist.sh
less mindist.sh
bash submit_mindist.sh
...
```

# Running analyses in parallel using job arrays

- Using a loop to submit jobs sounds like a good idea at first.
- It is problematic on clusters.
  - Each call to sbatch sends a remote call to the job scheduler.
  - A large number of calls can make the scheduler unresponsive (denial of service).
- It also has other disadvantages.
  - There is no match between the input and the SLURM output filenames.
  - Rerunning one crashed calculation is easy, but what about a subset of the calculations?
- **Do not use loops to submit multiple jobs.**

# Running analyses in parallel using job arrays

- Job arrays are a SLURM feature to submit multiple similar jobs efficiently.
- In a job array, all jobs use the same batch script.

```
rm *.{out,sh,xvg}  
cp ~/gmxttools/10_mindist.sh mindist.sh  
less mindist.sh  
sbatch mindist.sh; sq  
...
```

- The `--array` option submits a job array.
- The `SLURM_ARRAY_TASK_ID` variable is set to the array step by the job scheduler.

# Running analyses in parallel using job arrays

- Job arrays can span ranges.

```
--array=1-10,15
```

- A maximum number of concurrent running jobs can be set.

```
--array=1-100%10
```

- Useful to avoid overloading a filesystem

- You can run new array steps or rerun past steps.

```
sbatch --array=12,252 mindist.sh
```

- Useful to rerun crashed calculation

- Output filenames match array steps.

```
ls slurm-*.out
```

# Using job arrays with multi-dimensional input

- We compute the mean-square displacement (MSD) of water molecules in the vicinity of 14 Anx residues (NMR probes) in 10 trajectories.
- We have 140 ( $14 \times 10$ ) calculations to perform.
  - Residues: 12, 16, 104, 112, 117, 121, 124, 128, 137, 141, 144, 162, 180, 260
  - Trajectories: 01, 02, 03, 04, 05, 06, 07, 08, 09, 10
- SLURM job arrays are one-dimensional.
  - Each step is a positive integer: 0, 1, 2, ...
  - This does not match our two input variables.
  - How do we convert between multiple input variables and a linear job step?

# Using job arrays with multi-dimensional input

- One way to convert between multiple input variables and a linear job step is to put all variable combinations in a file.

```
12 01
12 02
...
12 10
16 01
16 02
...
260 09
260 10
```

- The line number of a given variable combination becomes its array step.
- The total number of lines is the number of array steps.

# Using job arrays with multi-dimensional input

- We start by generating the file for the array step variables.

```
mkdir -p ~/Anx_on_DOPC/analysis/MSD
cd ~/Anx_on_DOPC/analysis/MSD
cp ~/gmxttools/11_make_MSD_vars.sh make_MSD_vars.sh
less make_MSD_vars.sh
bash make_MSD_vars.sh
(head; tail) < MSD_vars.txt
wc -l MSD_vars.txt
```

- We then use a job array.

```
cp ~/gmxttools/12_MSD.sh MSD.sh
less MSD.sh
sbatch MSD.sh; sq
...
```

# Multiple short analyses in a single job with GNU parallel

- Some calculations require only a minute or even a few seconds.
- In such cases, a job array is not an appropriate approach.
  - Each array step is a full-fledged job.
  - It takes (at least) a few seconds to set up when a step starts.
  - It takes (at least) a few seconds to clean up after a step completes.
  - When steps very are short, this overhead becomes significant.
- Very short calculations should be packed together in a single job.
- However, we want to retain the ability to run calculations in parallel.

# Multiple short analyses in a single job with GNU parallel

- GNU parallel is a great tool to run multiple calculations in parallel inside a single SLURM job.
- GNU parallel dispatches calculations efficiently on one or more CPU cores.
- Example: You have 800 calculations to perform and submit a job requesting one full 40-core node.
  - GNU parallel starts 40 calculations when the job starts.
  - As soon as a calculation ends, GNU parallel starts another.
  - The 40 cores stay busy until there are no more calculations to start.
  - On average, each core performs 20 calculations ( $800 / 40$ ).

# Multiple short analyses in a single job with GNU parallel

- We repeat the previous example, computing the MSD of water around 14 Anx residues in 10 trajectories.
- We use two scripts.
  - One is a batch job script that runs GNU parallel to repeat an MSD calculation over 14 residues and 10 trajectories.
  - The other performs an MSD calculation for the water in the vicinity of a given residue, for a given trajectory.

```
cd ~/Anx_on_DOPC/analysis/MSD
rm *
cp ~/gmxtools/13_gnupar_MSD.sh gnupar_MSD.sh
cp ~/gmxtools/14_MSD.sh MSD.sh
less gnupar_MSD.sh
less MSD.sh
sbatch gnupar_MSD.sh
...
less slurm_MSD.out
```

# Multiple short analyses in a single job with GNU parallel

- In the previous example, the output from all calculations is collated in one SLURM output file.
- We can separate the output for each calculation using `--results`.

```
rm *.{ndx,out,xvg}
cp ~/gmxttools/15_gnupar_MSD.sh gnupar_MSD.sh
less gnupar_MSD.sh
sbatch --wait gnupar_MSD.sh
tree output | less
less output/i/12/j/01/stderr
```

# Speeding up analyses using trajectory post-processing

- Some trajectory analyses are IO-limited (input-output).
  - Usually by the rate at which the trajectory can be read from disk
  - This is frequent for simple analyses such as extracting positions.
- Trajectory compression (e.g. XTC format) slows down analyses.
  - Frames must be decompressed to read coordinates
- Appropriate trajectory post-processing can speed up analyses
  - Removing atoms not used during analysis (typically the solvent)
  - Using an uncompressed format (e.g. TRR)
  - Post-processed trajectories can be stored in `/scratch`.
    - Better for intensive read/write operations on large files
    - These are temporary files for which no backup is needed.
    - Remember that inactive files are regularly purged from `/scratch`.
  - Post-processing compressed trajectories and outputting another compressed trajectory can lead to loss of precision (e.g. XTC → XTC).

# Speeding up analyses using trajectory post-processing

- We perform the same backbone RMSD calculation on two trajectories.
  - The original XTC resulting from the simulation.
  - A post-processed TRR containing only the protein.
- We compare calculation times.

```
mkdir -p ~/Anx_on_DOPC/analysis/traj_post_process
cd ~/Anx_on_DOPC/analysis/traj_post_process
cp ~/gmxttools/16_extract_protein.sh extract_protein.sh
less extract_protein.sh
sbatch --wait extract_protein.sh
ls -l ~/Anx_on_DOPC/simulations/2_MD/traj_comp.xtc \
    traj_protein.trr
cp ~/gmxttools/17_timed_RMSD.sh timed_RMSD.sh
less timed_RMSD.sh
sbatch --wait timed_RMSD.sh
grep real slurm_RMSD.out
```

# MDAnalysis for GROMACS structures and trajectories

- The GROMACS tools are command-line programs and can be automated using Bash scripts.
- They support a variety of options that make them very powerful.
- However, they are not flexible beyond their built-in options.
- There is no programming language interface (e.g. Python) to control or customise the tools.
- If you want to program something that the GROMACS tools cannot do, one solution is to a library for a programming language you are familiar with.
- Examples (this list is not exhaustive):
  - bio3D (R)
  - MDAnalysis (Python, MD trajectory analysis)
  - MDtraj (Python, MD trajectory analysis)
  - pytraj (Python, front-end for cpptraj from the Amber Tools)

# MDAnalysis for GROMACS structures and trajectories

- MDAnalysis is a Python library for the analysis of MD trajectories.
- It supports reading and writing structures and trajectories in the file formats typically used with GROMACS.
  - PDB, GRO, XTC, TRR
- MDAnalysis gives you direct access to atom coordinates and other properties through NumPy arrays.
- It comes with an extensive analysis submodule.
  - Distances, contacts, H-bonds, membranes, nucleic acids
- This provides a very flexible analysis framework.
- You can develop your own analysis routines by accessing atom properties and reusing existing analysis routines.

# MDAnalysis for GROMACS structures and trajectories

- We repeat the previous distance calculations between 14 Anx residues (NMR probes) and the DOPC bilayer, using MDAnalysis instead of the GROMACS tools.
- This time, instead of minimum distance, we want time series of  $\Delta Z$  between the centre of mass of the residues and the average position of the phosphates in the upper leaflet of the bilayer.
- We start by installing MDAnalysis in a Python virtual environment.

```
module purge
module load python/3.8
virtualenv ~/MDA_env
source ~/MDA_env/bin/activate
avail_wheels MDAnalysis --all-versions
pip install --no-index MDAnalysis
deactivate
```

# MDAnalysis for GROMACS structures and trajectories

- We use two scripts.
  - One is a batch job script that activates our virtual environment and calls the other script.
  - The other is a Python script that uses MDAnalysis to compute all 14 timeseries in a single pass over the trajectory.

```
mkdir -p ~/Anx_on_DOPC/analysis/bilayer_aa_comdist
cd ~/Anx_on_DOPC/analysis/bilayer_aa_comdist
cp ~/gmxttools/18_comdist_job.sh comdist_job.sh
cp ~/gmxttools/19_comdist.py comdist.py
less comdist_job.sh
less comdist.py
sbatch --wait comdist_job.sh
...
```

# Generating GROMACS index groups with MDAnalysis

- MDAnalysis can be used to generate index groups for usage with the GROMACS tools.
- We repeat the atom selections done in the previous example, but save them to an NDX file instead of calculating distances in MDAnalysis.
- This script can be run on the login node.

```
cp ~/gmxtools/20_make_comdist_index.py make_comdist_index.py
less make_comdist_index.py
source ~/MDA_env/bin/activate
python make_comdist_index.py
deactivate
less groups.ndx
```

# References

- GROMACS: <https://www.gromacs.org/>
  - Manual: <https://manual.gromacs.org/>
- Anx article in JACS: <https://pubs.acs.org/doi/10.1021/jacs.6b07005>
- SLURM: <https://slurm.schedmd.com/>
  - Job arrays: [https://slurm.schedmd.com/job\\_array.html](https://slurm.schedmd.com/job_array.html)
- GNU parallel: <https://www.gnu.org/software/parallel/>
  - Manual: <https://www.gnu.org/software/parallel/man.html>
- MDAnalysis: <https://www.mdanalysis.org/>
  - Documentation: <https://docs.mdanalysis.org/>