GROMACS structures & interfaces

Dr Mark Abraham, Intel

Given at SciLifeLab, Sept 10, 2024

Outline

- Workflows when using GROMACS
- Repository structure
- Relevant structures in mdrun
- Internal interfaces in GROMACS
- Other tidbits
- Hands-on exercises

Workflows when using GROMACS

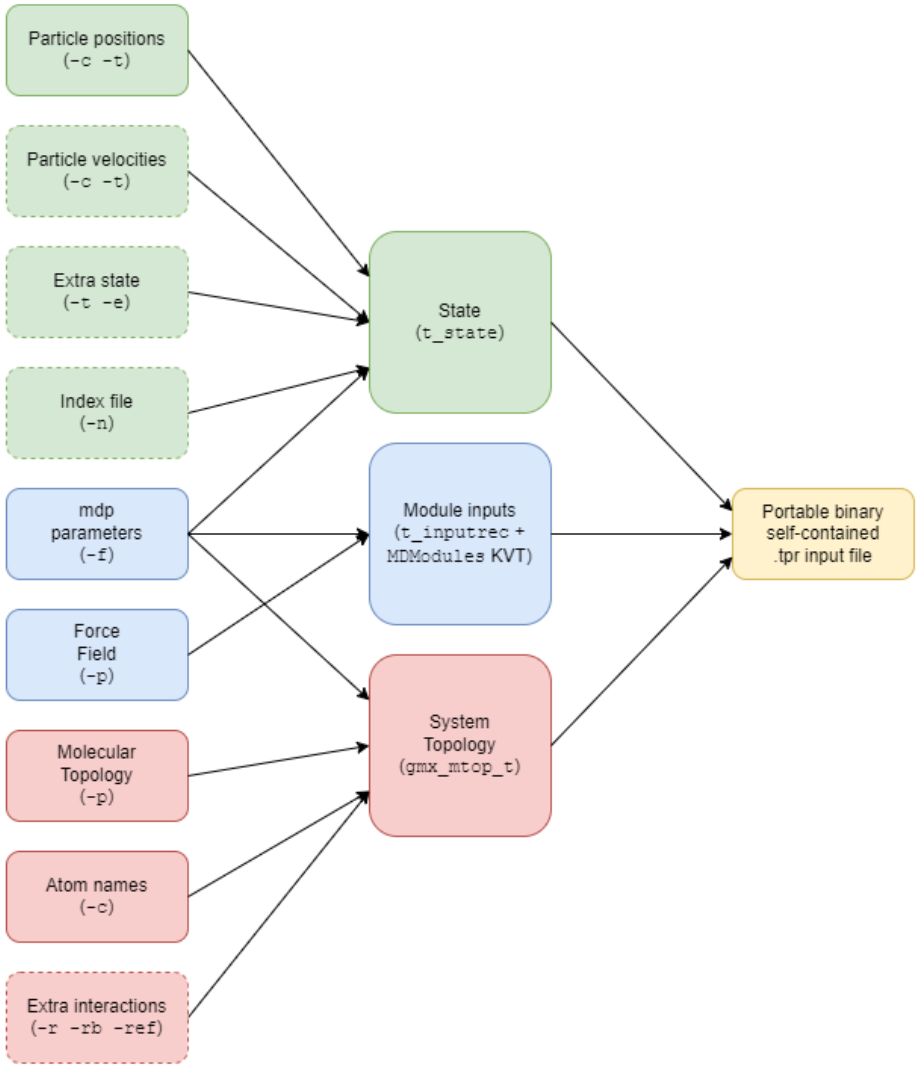
Software exists to be used - so usage should determine structure

We'll look at several usage patterns within GROMACS to demonstrate why several kinds of structures exist

grompp

- The GROMacs Pre-Processor
- Pronounced “grompp” or “grom-p-p”
- Combines various inputs to make an input for `gmx mdrun`
- That input is self-contained and works the same on any computer

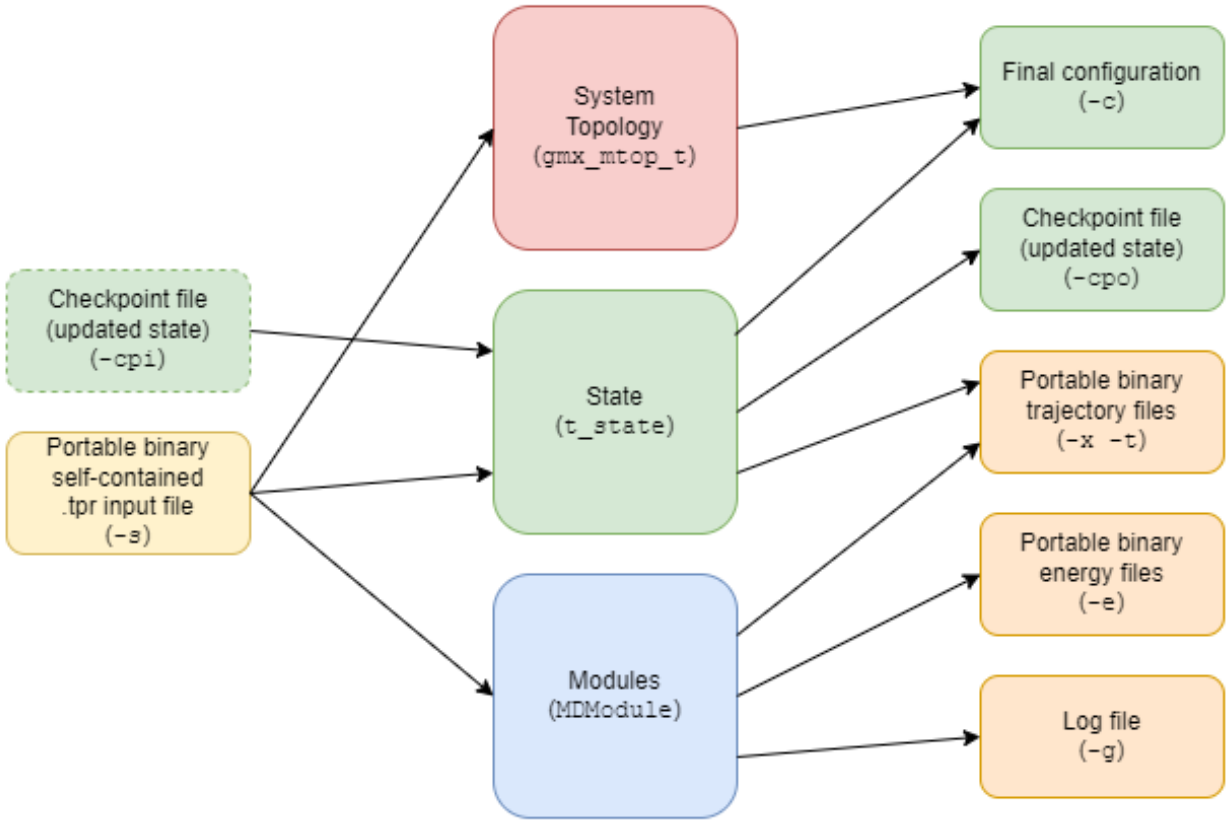
```
gmx grompp -c conf.gro -t state.cpt -n index.ndx -f input.mdp -p topol.top -r restraints.gro
```



mdrun

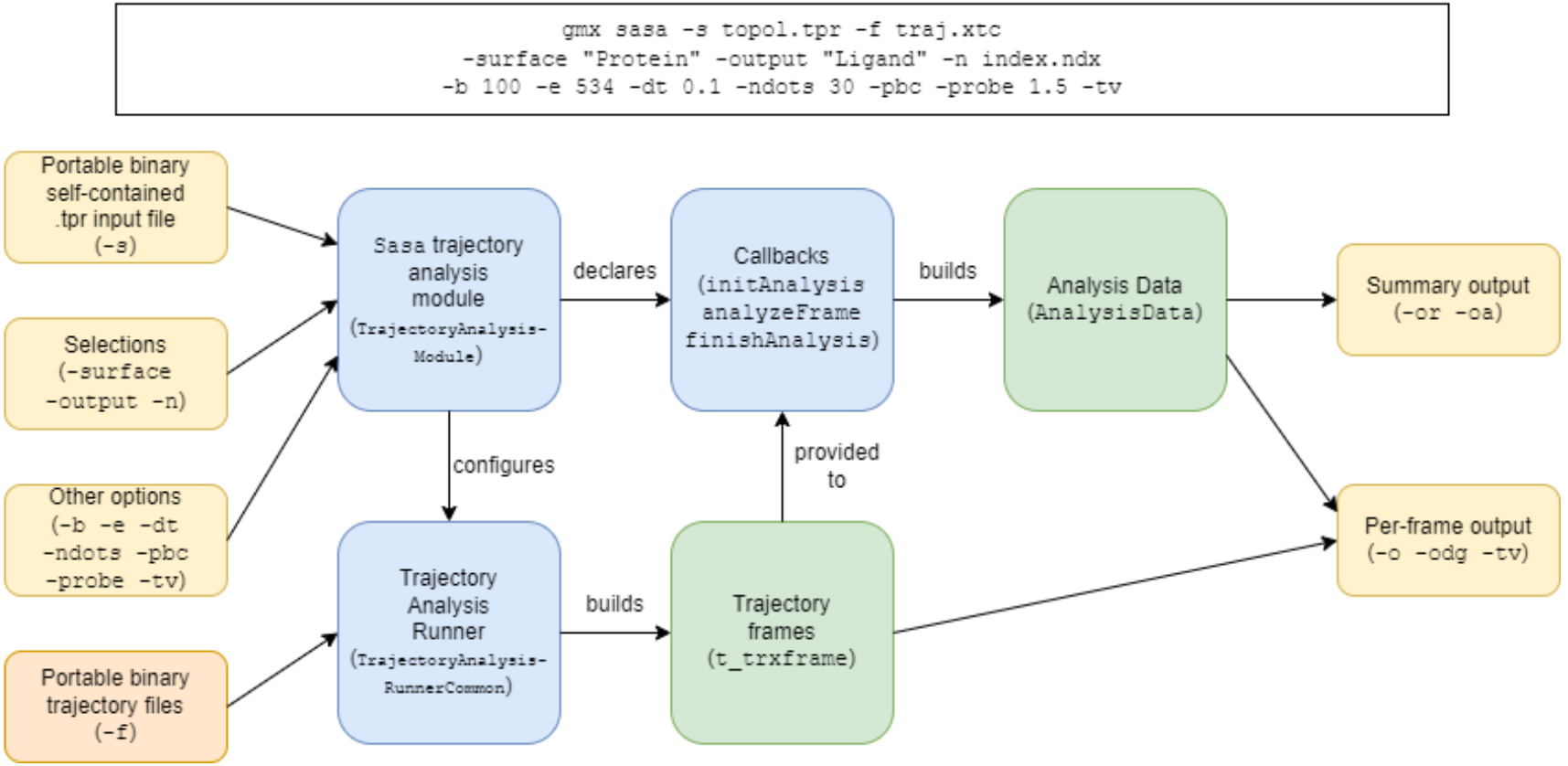
- Runs MD simulations
- Incorporates different engines for doing MD, EM, TPI, rerun, MIMIC

```
gmx mdrun -s topol.tpr -cpi state.cpt
-g md.log -x traj.xtc -e ener.edr -c confout.gro -cpo state.cpt
```



The trajectory analysis framework (TAF)

- Analysis tools are being ported to it
- Framework gets extended to meet new use cases



trjconv – how not to structure

- Docs online
- Code here
- This needs to become a composable toolkit with
  - Input adapters
  - Filters for frames
  - Operations to do on frames
  - Output adapters

  Plus standalone tools for niche functionality.
  Work in progress!

Repository structure

What content is found where?

How do I find things?

Documentation

- /docs includes a lot of high-level ReStructured Text documentation, which is built nightly and appears here https://manual.gromacs.org/nightly/
- Let's go see!
- Source files, class declarations, and function declarations have Doxygen.
- This builds nightly and is found at https://manual.gromacs.org/nightly/doxygen/html-full/index.xhtml Much is not yet documented ☹
- New code contributions must have Doxygen!

Data files

- /share/top has useful static content:
  - Force field definitions
  - Topology building blocks
  - Water boxes
  - Descriptions for fixing broken structures
  - Tables of standard functional forms

Build system

- GROMACS uses CMake (https://cmake.org/)
- Most folders have a CMakeLists.txt file
- Lots of complicated detection of issues and work-arounds so users don’t need to know weird things to get GROMACS installed
- Top-level /cmake folder has some reusable content
- CMake became a totally different (and better!) language since we started using it more than a decade ago, so we’re gradually modernizing it

The code

- Source files - small groups of related code, e.g. the implementation of a class
- Header files - visible interfaces to code in source files, OR performance-sensitive code that needs to be inlined by the compiler
- Modules – medium-sized groups of related code e.g. simd, gmxpreprocess, topology, listed_forces, found two levels under /src, typically with a group of tests. Here's a (scary) dependency map.
- Libraries - large groups of related code, e.g. libgromacs, libgmxapi, libnb

Things found in the code

- Struct - lacks an invariant (typically only public data) and generally no methods e.g. t_forcerec
- Class - has an invariant (typically private data) and methods e.g. PaddedVector
- (Free) functions - frequently found (C heritage) sometimes should be a method on a class that hasn’t grown yet (if so, often first parameter has the type of the class to which it should belong) e.g. wallcycle

Tests

- Several test scopes
  - Unit tests
  - Integration tests
  - End-to-end tests
- Several kinds of test data
  - Correctness test
  - Comparison tests
  - Regression tests
- Two frameworks
  - Based on GoogleTest: found in tests subdirectory of each module
  - Based on perl script in separate repo: https://gitlab.com/gromacs/gromacs-regressiontests – avoid this at all costs

Relevant structures in mdrun

What data types will I keep seeing?

Where should new things go?

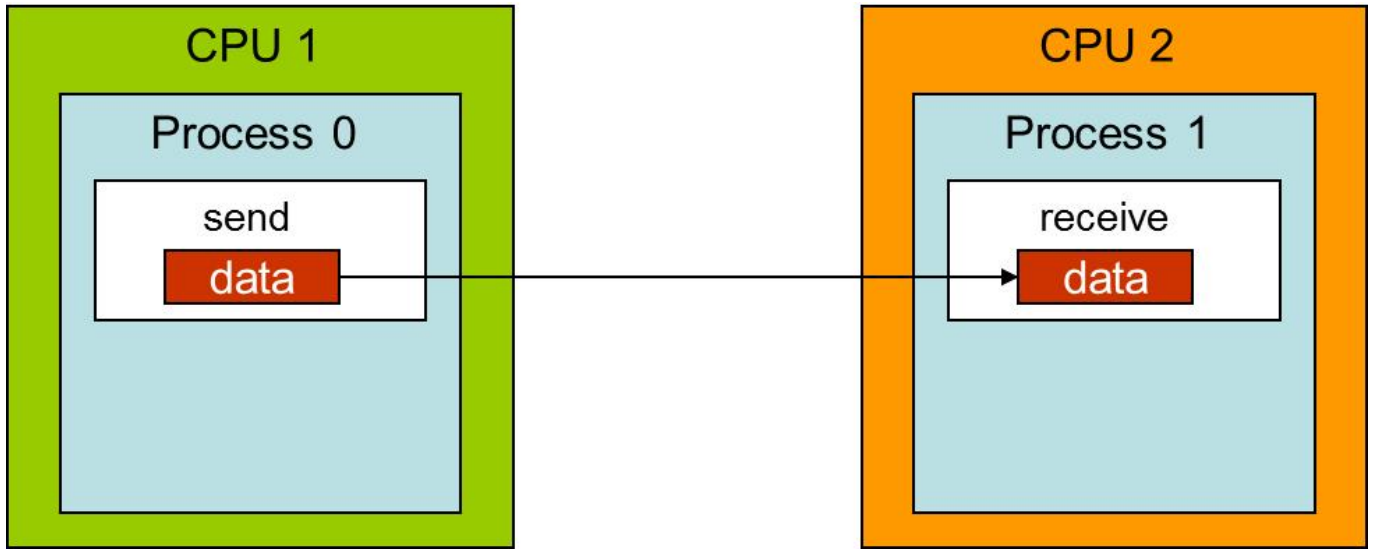
MD parameter input (.mdp)

- Lots of software uses key-value pairs as input
- Historically GROMACS used t_inputrec to contain the values parsed from the .mdp file
- Then every module includes that header (yuck)
- Much better to have each module take care of declaring its own key-value pairs
- Transition is underway, currently the key-value tree (KVT) is owned by t_inputrec. See applied electric field module example.

```
title                    = Yo
cpp                      = /lib/cpp
include                  = -I../top
define                   =
integrator               = md
dt                       = 0.002
nsteps                   = 500000
nstxout                  = 5000
nstvout                  = 5000
nstlog                   = 5000
nstenergy                = 250
nstxout-compressed       = 250
compressed-x-grps        = Protein
energygrps               = Protein  SOL
nstlist                  = 10
ns-type                  = grid
rlist                    = 0.8
coulombtype              = cut-off
rcoulomb                 = 1.4
rvdw                     = 0.8
tcoupl                   = Berendsen
tc-grps                  = Protein       SOL
tau-t                    = 0.1  0.1
ref-t                    = 300  300
Pcoupl                   = Berendsen
tau-p                    = 1.0
compressibility          = 4.5e-5
ref-p                    = 1.0
gen-vel                  = yes
gen-temp                 = 300
gen-seed                 = 173529
constraints              = all-bonds
```

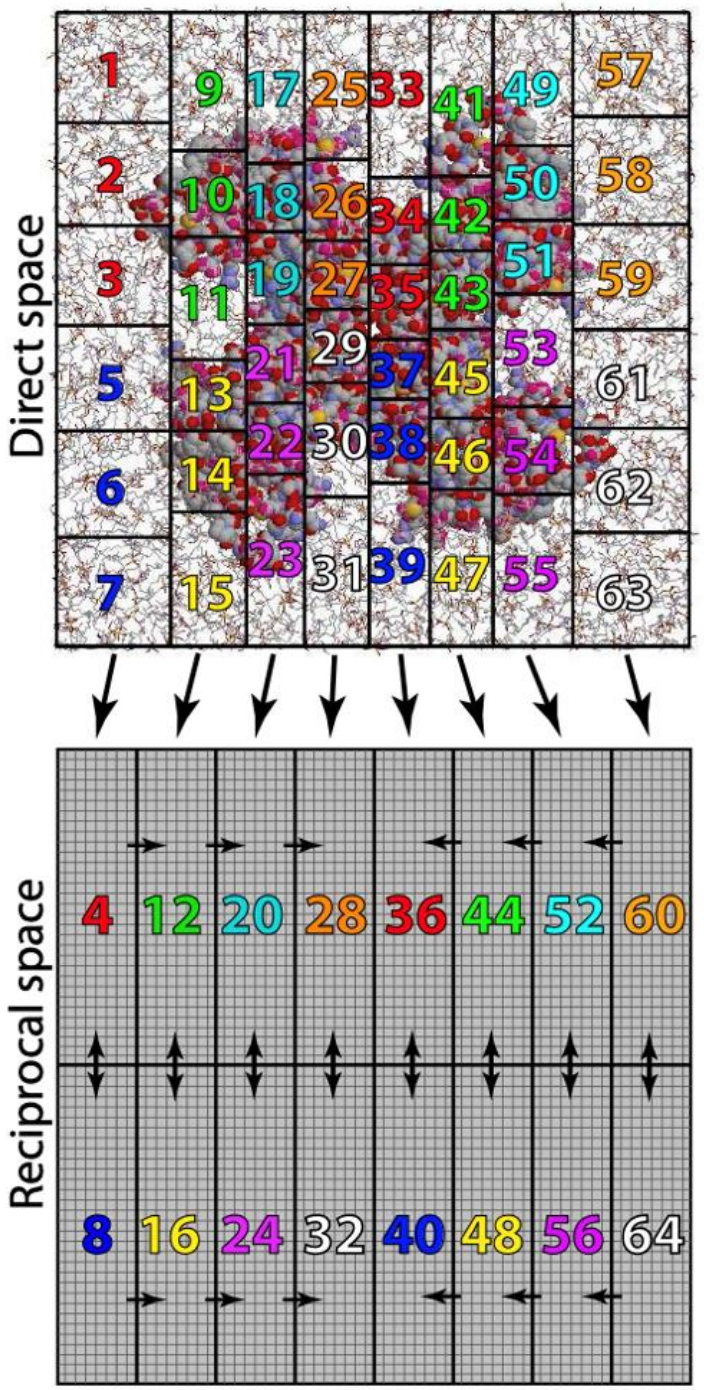
Communication

- mdrun distributes the work to multiple independent workers
- Some form of MPI library starts the workers (called “ranks”)
- The workers stay in a tight collaboration sending messages
- The “communication record” t_commrec helps with that



Domain decomposition

- Pivotal concept behind multi-rank GROMACS
- Each MPI rank maps to a single domain
- A domain is a compact grouping of particles
- Two phases:
  - (Re-)partitioning every 100-200 steps where the domain is made compact again, which triggers rebuilding of short-ranged neighbourlists
  - Halo exchange for x and f every step
- Struct gmx_domdec_t



Molecular topology (mtop)

- Very similar hierarchical structure to `[system]` in `.top` file
- Declaration of gmx_mtop_t
- Like `[system]`, there's plenty of smaller structures that might be reused within the same molecular topology
- To loop over the whole thing to e.g. find all atoms, use the looping functionality
- Some related functionality exists for cases where fully-written out topologies make sense, e.g. within a domain, or in a legacy analysis tool

Other important data structures in mdrun

- Ftype – different kinds of particle-interaction function types
- Options – allows configuring e.g. command-line tools to receive parameters
- t_state – contains all data with the thermodynamic state, plus a bit of algorithmic state that lets GROMACS propagate MD (and *not* forces or energies)
- interaction_const_t – contains parameters for evaluating nonbonded-interations
- t_mdatoms – contains description of domain contents other than the local topology (e.g. mass, charge, group membership)

Internal interfaces in GROMACS

How do I add new functionality?

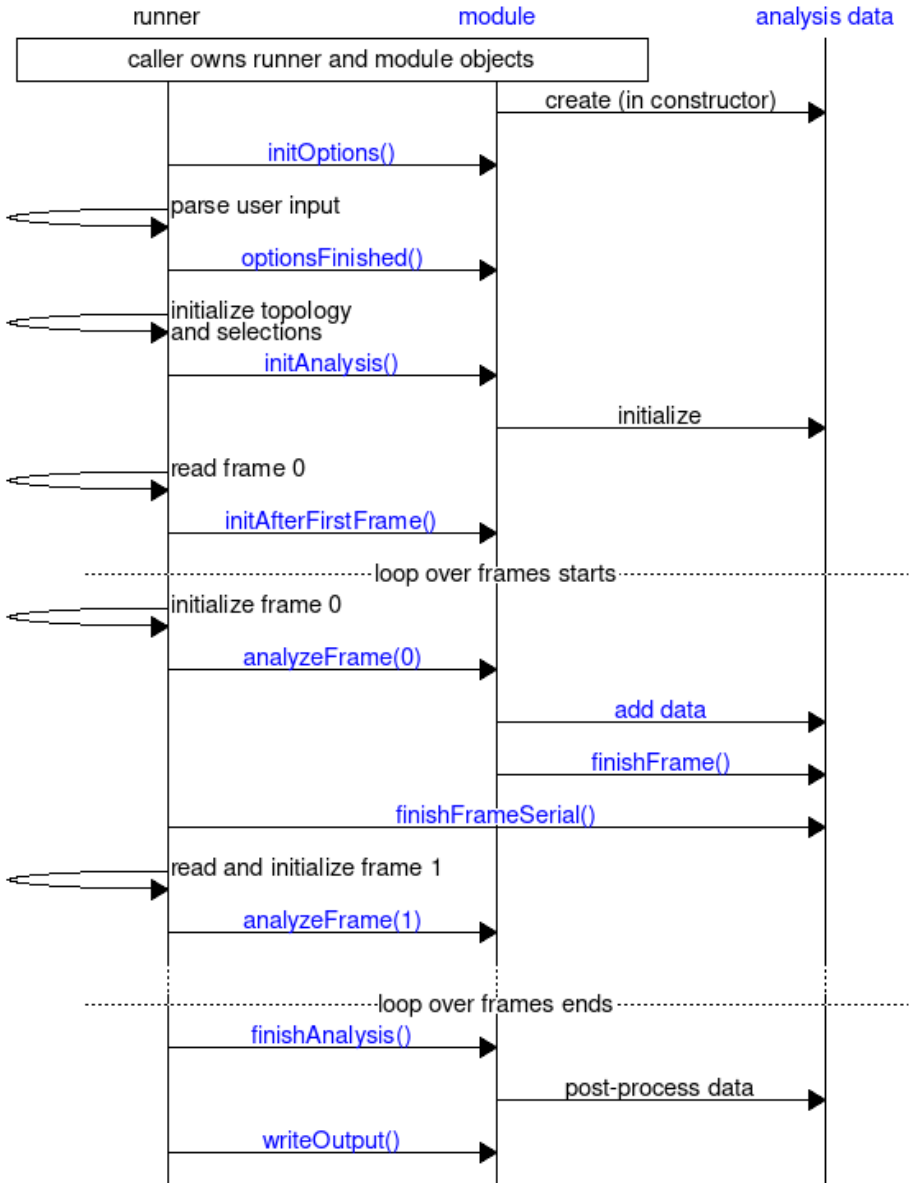
Where do I change existing functionality?

Software interfaces

- Interface is the surface available for use
- Implementation are the details below that which make it work
- If you can get your job done using only the interface, your software is less coupled, so easier to use and maintain
- Example: Google Docs has an API, so with it you could write code to search all your docs for a phrase
- GROMACS has several Application Programming Interfaces (APIs)
- Lower down there are several interfaces that provide a way for new functionality to be incorporated in a modular way

TAF interface

- High-level docs TAF
- Workflow for analysis tools
- Many examples in the framework



MDModules

- Framework for composing GROMACS functionality
- Permits collaboration with lightweight dependencies
- Coordinates common infrastructure needs like getting input values, writing output, computing forces, receiving notifications after events
- High-level docs via mdmodules

IMdpOptions

- Allows modules to receive parameters via the .mdp file passed to `gmx grompp`
- High-level docs via mdmodules
- Doxygen IMdpOptions
- Example QMMM

IForceProvider

- Allows modules to compute forces from given inputs
- Don’t need to know how domain decomposition is done, etc.
- Docs IForceProvider
- Source IForceProvider
- Example restraint module

ISimulator interface

- Used for implementing a different tool like
  - mdrun,
  - minimize,
  - TPI,
  - rerun,
  - MiMiC for QM/MM
- The ISimulator interface
- Example rerun

MDModulesNotifiers

- Allows modules to subscribe to particular events and then be called back and perhaps provided with handles to the new information
- For example:
  - after preprocessing is (nearly) complete,
  - during simulation setup,
  - checkpointing
- High-level docs via MDModulesNotifiers
- Example density fitting
- Currently unclear where the dividing line is between functionality that should go in MDModules and what should be a notifier – ask first!

ObservablesReducer

- Every MD step must be prepared to accumulate values taken from every domain, e.g. the local electrostatic energy has to be added
- But those values come from many modules, and most only need the work done occasionally
- Simple if every module would do global communication itself, but not fast scalable
- Need to aggregate the communication, but efficiently and maintainably
- High-level docs ObservablesReducer
- Source ObservablesReducer

Tidbits

- High-level docs in `*.md` files `/src/docs/doxygen` (but you can also find them with a web search)
- Mdp file has user-defined variables that give you a fast way to get started implementing your module
- To find something in the code e.g.
  `git grep -i mdpoptions`

Hands-on exercises

- Make a new tool in the analysis framework that runs but does nothing!
- Use userint1 .mdp field to pass a value into mdrun which it writes to stdout