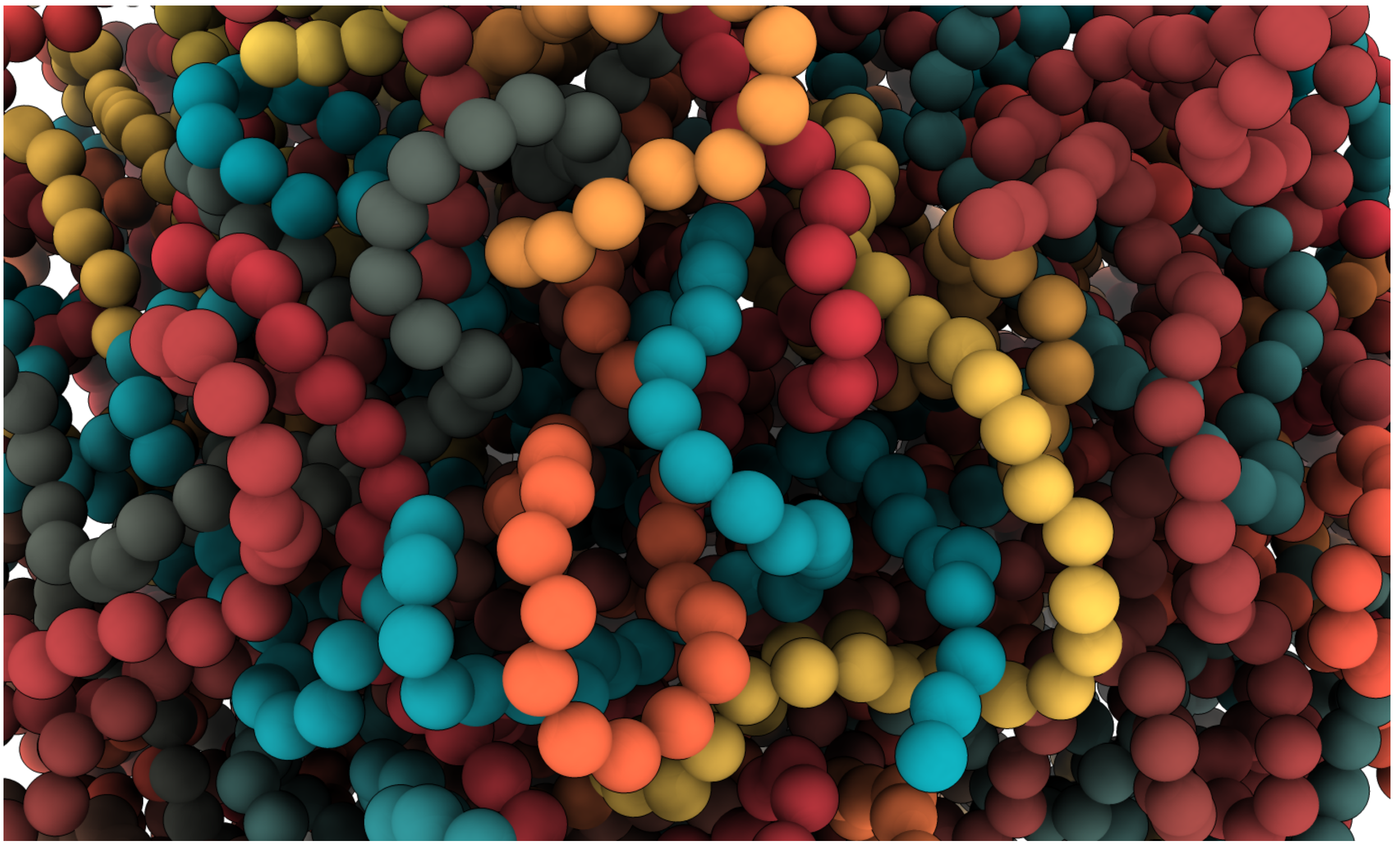
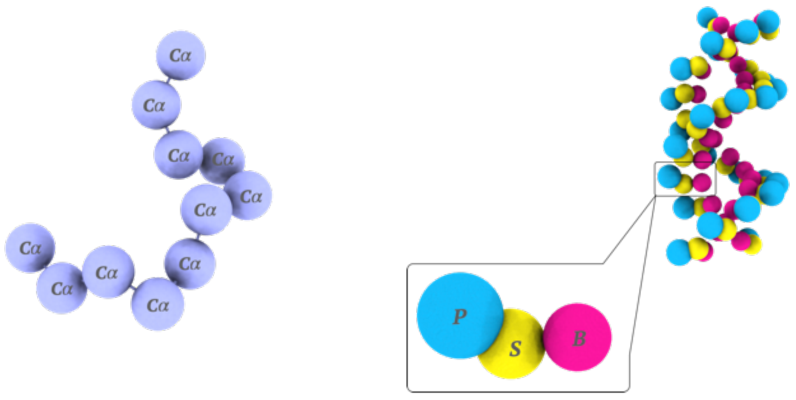
# Coarse-grained Molecular Dynamics Tutorial



## Preparing the topology file

In Coarse-grained Molecular Dynamics (CG-MD), each amino acid in a protein (or a nucleotide in a DNA molecule) is represented in a “concise” manner. The amino acids are represented with one or two beads at the position of the C $\alpha$ atom or at the position of the C $\alpha$ and the C $\beta$ atoms, respectively. The nucleotides of the static DNA are represented with a static version of the 3SPN model.



CG native topology model describes the biomolecule such that the experimentally obtained structure is defined as the global energetic minimum.

The 3D structure of the protein of interest is represented with beads in space, and each bead's position is represented by a position on the x,y, and z-axes (coordinates). During the simulation, the beads are subjected to a force and their trajectory in space is calculated for each timestep and written out in an output file.
To run a simulation of a protein, we first must have the coordinates of the C $\alpha$ or C $\alpha$-C $\beta$ beads. Such coordinates are calculated from experimentally obtained structure coordinates stored in a pdb file.

## 1. Clone the GitHub repository with the MD code

To get access to the MD code, you should clone the repository from the lab's GitHub as follows:

1. Create a user in GitHub using your weizmann e-mail and create the SSH key in your workstation (you can find the setup instructions on GitHub website).
2. Create a `scripts` directory in your home directory.

    ```
    mkdir ~/scripts
    cd ~/scripts
    ```

3. Clone the repository inside the scripts directory .

    ```
    git clone git@github.com:REPOSITORY
    ```

This action will place an "MD" directory inside the scripts folder you created in the previous step. Make sure you have the following subdirectories: `src`, `bin`, `util`, `wrapper`, `generateMDInput`, `t2p` and `tutorial`.

Navigate to the `~/scripts/MD/bin/` folder. Despite having precompiled code, you should still compile it such that it can run on your operating system. Different operating systems have different system libraries and architectures. These libraries and architectures provide the interface between the program and the underlying hardware, and if a program is compiled for one operating system, it will not be able to use the libraries and interfaces of another operating system. Recompiling the code allows the program to use the appropriate libraries and interfaces for that operating system, which in turn allows the code to run correctly.

To compile the MD code execute the following command:

```
./MDmake
```

You might get some warnings similar to this:

```
../src/dynamicRange.f:81.20:

       a2i = a2i * -1
                    1
Warning: Extension: Unary operator following arithmetic operator (use parentheses) at
(1)
```

As long as you do not have any errors in compilation, you're good. Check that compilation created a new MD.exe file and updated the **.o** files.

Navigate into the tutorial folder.

## 2. Download a pdb file

Experimentally obtained structures of biomolecules (proteins, DNA, RNA) are stored in a protein data bank (PDB). Here you can perform a search for the pdbID code of the protein structure or look for a structure by proteins name. One can download the coordinatesrmat when navigating to the structure page file in .pdb through the **Download Files** section.

Download the pdbID:**1srl** to your working directory.

You can read here about the file format to get familiar with it.

## 3. Clean the coordinates from solvent and ligands/ions

Experimental methods that allow us to obtain the native structures of biomolecules, often capture the surrounding of the biomolecule — water molecules, ligands, ions, and other solution components.

The structure of the src SH3 domain was obtained using solution NMR, which does not provide information regarding non-catalytic water molecules. In the future, to prepare your file for the simulation, all of the non-protein\non-DNA molecules must be eliminated from the pdb file. This can be done in at least two ways.

1. Saving the protein atoms only with the following command in the shell:

```
grep -v HOH pdbID.pdb > pdbID_clean.pdb
```

Here using *grep* save all lines as new .pdb file, except the lines that contain the HOH entry in them.

2. Alternatively, you can use *pymol*. Open the application in your workstation in the lab by typing in the shell:

```
pymol pdbID.pdb
```

This will open the pymol window with the pdbID coordinates, where the water molecules and ions can be selected as a group with the *select* command and deleted.

A piece of general advice when handling pdb files and preparing them for simulations is to open the pdb with a text editor (vim, gedit, nedit) and examine it to make sure there are no surprises (missing parts, non-sequential numbering of the residues, etc.). The same exercise should be executed by pymol - opening the structure and looking for missing loops and alternate rotamers is always a good idea. Additional tutorials regarding pymol can be found anywhere on the web, such as this for example.

## 4. Create a .dat topology file

Now that the pdb file is nice and clean you can proceed to the next step - generating the input for our CG simulation. Being said, the CG model contains only a limited set of beads, representing the structure of the protein/DNA molecule. In order to create the beads topology file,  first  look at the `generateMDInput.prefs` file in your tutorial folder. This is the basic set of preferences for generating the CA bead model.

`PDB_FILE=1srl.pdb` The input file.

`NORMALIZE=YES` Normalization of the residue numbering (when normalized, the output would skip missing residues from the count).

`ELECTROSTATICS=YES` Whether to include charge information of the residues in the topology file. Note that for protein - DNA simulations this option should be `YES`. In other cases, the need for electrostatics should be examined.

`MD_INPUT_FILE=1srl.dat` The name of the topology file that would be used as an input of our simulation. Make sure you don't copy the name of the file with the pdb extension, which would result in the deletion of the initial pdb file.

`MODEL_TYPE=CA` What type of model should be generated? Options are CA or CB; in the latter, two kinds of beads represent each amino acid: CA and CB.

`CHIRALITY=NO` If the CB model is chosen, chirality should be checked as `YES`.

`REPULSION_RADIUS_FACTOR=0.7` Parameters of the repulsion terms.

`REPULSION_NEIGHBORS=NORMAL`

`CB_POSITION=CA` This flag allows you to choose where the CB bead is positioned. The options are CA, CB, or the FHA (the furthest heavy atom).

`INTER_CHAIN_CONTACTS=NO` The following flag is relevant in case the structure is comprised of several chains that might/might not interact with each other.

`HAS_STATIC_ATOMS=YES` This option is relevant when parts of the simulated ensemble are static; for example simulation of static DNA with a DNA-binding protein.

`DYNAMIC_ATOM_RANGE=1,?` The range of dynamic beads. To obtain the number of dynamic beads, you can use the previously used *grep* utility of the pdb file. How?

`CONTACTS_COEFFICIENT=1.0` The depth of the energetic well of the non-bonded interactions, which are defined by a Lennard-Jones-type potential.

Now that you have your preference file ready to go, you can generate the dat file; it contains the necessary information regarding the protein structure. Run the following script in the tutorial directory:

```
~/scripts/MD/generateMDInput/generateMDInput.pl
```

This script has read-only permissions, but you are welcome to peek into it to get an idea of what's happening behind the scenes.

This is how the last lines of the output should look:

```
Generating Beads.
Calculating Bonds.
Calculating Angles.
Calculating Dihedrals.
Calculating CSU contacts.
Translating CSU contacts to bead contacts for chain A
Calculating Repulsions.
Calculating electrostatic residues.
Printing to output file 1srl.dat.
```

If you are getting a different output, please refer to the section **Common mistakes**.

## 5. What's inside the dat topology file?

The script generated the dat file; this is a non-binary file that you can view with a text editor, for example, vi or nedit. Open the file and examine it.

The file is divided into sections, each containing different structural information.

### 1. Bonds

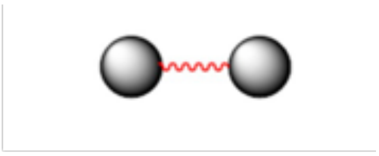
The columns in this section are:

```
      55 Hookean bonded pairs.
  1    1    2   3.822 100.000
  2    2    3   3.823 100.000
```

`index       i       j       distance        spring constant` $\epsilon_r$

The Hookean bonded pairs section lists all of the covalent bonds in the system. In CG-MD the covalent bond is represented with a harmonic potential of the form:

$$\sum_{bonds} \varepsilon_r (r - r_0)^2$$



In this tutorial, you will simulate a C $\alpha$ model, thus the beads are consecutively bonded.

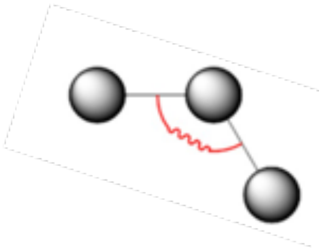
### 2. Angles

```
         54 bond angle trios.
  1    1    2    3   2.295   20.000
  2    2    3    4   2.322   20.000
```

`index       i       j       k       angle       spring constant` $\epsilon_\theta$

The bond angle trios section contains information regarding all the angles that form between every three bonded beads.

$$\sum_{angles} \varepsilon_\theta (\theta - \theta_0)^2$$

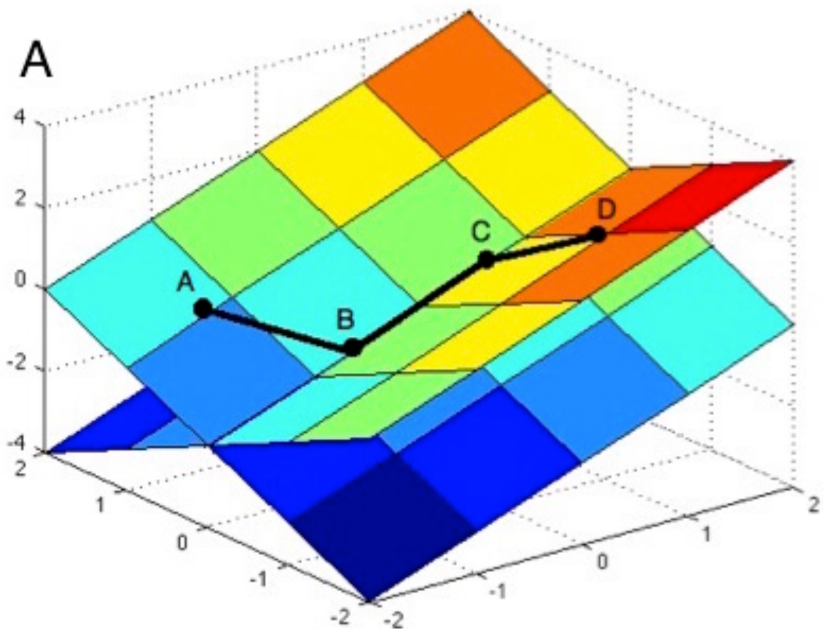


## 3. Dihedrals

```
         53 dihedral quartets.
   1    1    2    3    4    4.074    1.000    0.000    0.000
   2    2    3    4    5    4.183    1.000    0.000    0.000
```

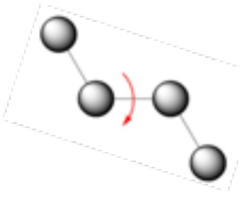
`index        i        j        k        l        angle    kd    -    -` $\mathrm{kd} = \epsilon_{BB}$

Dihedral angle is an angle that is formed between two intersecting planes. In case of proteins every consecutive four atoms from two planes as follows:



In coarse-grained modeling each four beads form such an angle. The dihedral angle potential is of the following form:

$$\sum_{backbone} \varepsilon_{BB} \left[1 - \cos(\phi - \phi_0)\right] + \frac{1}{2}\left[1 - \cos(3(\phi - \phi_0))\right]$$



## 4. Contacts

```
     137 contacts.
   1     1    23     52.771 1.000000
   2     1    24     45.724 1.000000
```

`index       i       j       r^2     epsilon` $\epsilon_C$

The native interactions sections. Created by CSU algorithm based on the pdb file that is used to generate the dat file. Native interactions are defined by Lenard-Johnes-type potential as follows:

$$\sum_{contacts} \epsilon_c \left[5\left(\frac{\sigma_{ij}}{r}\right)^{12} - 6\left(\frac{\sigma_{ij}}{r}\right)^{10}\right]$$

Because some simulated systems require a slightly different potential shape, the power terms might differ. These interactions are responsible for the integrity of the protein structure in the simulations.

## 5. Repulsions

```
       1241 repulsive pairs.
   1     1     5     16.000      1.000
   2     1     6     16.000      1.000
```

`index       i       j       r^2     epsilon` $\epsilon_{NC}$

The repulsive pairs are every pair of beads that do not participate in any native interaction with each other. This prevents non-native interactions or overlapping of the beads.

$$\sum_{non-contacts} \varepsilon_{NC} \left(\frac{\sigma_{ij}}{r}\right)^{12}$$

**Important note**: If you alter the contacts at any time in your project, remember:

1. a pair that is not in contacts should be in the repulsion section; otherwise, you will have two beads overlapping each other.
2. a pair that is in contacts should be removed from repulsion (LJ term includes repulsive term); otherwise, you will have two repulsion terms for this pair of beads.

### 6. Electrostatic residues

```
      11 electrostatic residues.
  1     7  -1.000
  2     9  -1.000
```

`index       i       charge`

Electrostatic residues section list all the residues of the protein (or DNA) that bear some charge. In CG-MD the positively charged residues are Lysine (K) and Arginine (R); the negative charged are Glutamate (E), Aspartate (D), and Phosphate bead in case you're working with DNA. The charge can be positive or negative and its position is chosen in the prefs file (for more information look at the *schema* file for the input generation).

The potential that is mostly used in CG-MD with the structure-based model (in the absence of water and ion molecules) is the Debye-Huckel potential:

$$U_{\text{Debye-Huckel}} = K_{\text{Coulomb}} B(\kappa) \sum_{i,j} \frac{q_i q_j \exp(-\kappa r_{ij})}{\varepsilon r_{ij}}$$

$$\kappa^2 = \frac{8\pi N_A e^2 \rho_A}{1000 \varepsilon k_B T}$$

Debye-Huckel theory produces an effective electrostatic potential in which ions screen the Coulomb interactions; thus, it accounts for the water\ion representation in the CG simulations.

### 7. Coordinates

```
     56 atom positions.
        1 chains.
       56 is the size of chain          1.
  1     1 CA THR  15.308  14.180  -2.955   1.000
  2     2 CA PHE  13.365  10.960  -3.638   1.000
```

`index       i       atom name       residue type       x    y    z    mass`

SH3 is a 56-residue-long domain. Now you have a topology file that describe 56 beads, each positioned at the C $\alpha$ atom of the protein.

## Running the simulation

## 1. Simulation preferences file (MDWrapper.prefs)

The `MDWrapper.prefs` file lists parameters that define your simulation and is a primary input file for your simulation. Here I will go over some of the essential parameters you have to be familiar with; feel free to peek in the `MDWrapper.prefs.schema` file that lists **all** parameters and their input options.

`EXECUTION_LOCATION=LOCAL` You can run the simulation on *your* workstation, or you can run it on the Cluster cluster of our lab. For a start, and it is also true for short simulations of small systems, you will run the SH3 simulation locally.

`OUTPUT_PATH=.` The location where your simulation results will be transfered at the end of the simulation.

`BASE_PATH=.` Location of the input files, relative to the prefs file.
`GO_PATH=/yourHomeDirectory/scripts/MD/bin/MD.exe` This is the path where the simulation binary file is located. **Make sure you update this path to your home directory path.**

`INPUT_FILE_PATH=./1srl.dat` Your input file.

`RUN_POST_EXECUTION_ANALYSIS=NO` Allows a pipeline for the process of simulating-analyzing. At some point in your research, you might be interested in executing some script right after the simulation finishes. This is where you do it.

`HAS_STATIC_ATOMS=YES`

`DYNAMIC_ATOMS_RANGE=1,?` Always verify the dynamic atoms range.

`WRITE_3BODY` At some large systems the calculation of the three-body interactions is impossible due to the large number of beads. If you are dealing with a large number of beads in your system and having troubles with running the simulation, try assigning this flag as NO and see if it helps.

`USE_ELECTROSTATICS=YES` Whether the simulation should run calculations of the electrostatic force.

`USE_DEBYE_HUCKEL=YES` When assigned `NO`, the simulation summons the Coulomb interactions calculation function.

`IONIC_STRENGTH=0.02` Essentially can be regarded as the salt concentration in your simulation (in M).

`MODEL_TYPE=CA` The kind of model you generated in the previously described steps of dat file generation.

`EXECUTION_STEPS=30000` Simulation length. The answer to how many steps you should execute is a complex one. When running the model for the first time, or debugging the modified code, or testing a new analysis code you wrote -- the number of execution steps should not be too large. When simulating for research purposes, the number should be sufficient for the sampling requirements (longer runs for kinetic studies, for example). The sufficient number of steps is somewhere around $10^7 - 10^8$.

`OUTPUT_FREQUENCY=1000` The frequency would directly affect the number of frames you will get in your trajectory, which would be the ratio between the execution steps and the output frequency (as in this example, it's **three** frames).

`TEMPERATURES=0.4` The temperatures list should be passed in the following format: 0.4,0.5,0.6, and so on. Spaces or tabs after each value will cause the execution to run before the simulation start.

`ITERATIONS_PER_TEMPERATURE=1` In how many repeats of each simulation are you interested? In this tutorial, it'll be `1`, but you might want several repeats in the future.

You can find the full `MDWrapper.prefs.schema` file in the following location:

```
~/scripts/MD/wrapper/MDWrapper.prefs.schema
```

## 1. Run a simulation locally

From your working folder, run the following script:

```
~/scripts/MD/wrapper/MDWrapper.pl
```

This script creates all the necessary input files for the simulations and executes the simulation scripts; it might take a while. The larger your system (in the number of beads and interactions), the more time-consuming this step will be. Usually, when running a simulation for research (and not tutorial) purposes, you would prefer to run it on the cluster; there the computational abilities are much higher than in your workstation. Later you will practice doing just that.

At the end of the run, you will have a new folder in your working folder. Inside you have two folders and a log file. If the simulation encounters problems, it is wise to look inside the log file.

The `input` folder contains the dat file in the settings you used for the simulation. The `output` folder is your bread-and-butter folder; inside it are several subdirectories.

`Etotal` The  total energy per time step

`EbyType` for each time step contains several energy terms:

0 = Bonds

1= Angles

2= Dihedrals

3= LJ contacts

4= Replusions

5= Box

6= Electrostatics

7= Chiral

8 = Elipsuid repulsion

9 = dual-basin-gaussian Dihedrals

10 = dual-basin-gaussian contatcs

If your simulation crashes or the system blows up, it's a good idea to visualize the energy terms to indicate where the problem is.

`LastCord` stores the coordinates and velocities of each bead in the last time step. This will be useful when you want to simulate an ending point of another.

`2Body` contains the total number of native contacts in each time step.

`3Body` here you'll find the number of three-body interactions in each time step

`Traj` the trajectories from the simulation.

The files inside these subfolders are zipped to the bz2 file format. To access the files, you can unzip them with the bunzip2 utility. Open the file with any text editor. The trajectory file first lists all the beads of the simulation and after the coordinates of each step.

**Important note:** If you used several dynamic regions and excluded the static beads from the trajectory, you are getting nonsense since this option only works when there is **one** dynamic range of atoms, and it starts from index 1. The reason for that is that this flag was developed for simulating static DNA along with protein.

Let's convert the trajectory file to a pdb file; this will allow us to load the trajectory in pymol and watch it as a movie. Use this utility to convert the file:

```
~/scripts/MD/t2p/TrajToPDB.pl TRAJ_FILENAME.DAT
```

This will generate the pdb file in the same directory; open it with pymol and play the movie.

**Tip**: Get familiar with pymol basic commands that allow you to open files, select chunks of the molecule, show different types of representations (spheres, cartoon, etc.), and color it. You will need it in the future.

Navigate into the EbyType directory and unzip the energy file. You can examine it with a text editor and get familiar with its structure. Plot the columns in the file using gnuplot or xmgrace utility and examine the energy behavior.

## 2. Run a simulation on Cluster

### 1. What is the Cluster?

Cluster is a computer cluster - a group of interconnected computers, or nodes, that work in a single system. There are many reasons to use computer clusters.

Computer clusters can perform complex tasks much faster than a single computer, dividing tasks into smaller subtasks and distributing them across the nodes. Such a system is reliable because the cluster has multiple nodes; if one fails, the others continue to operate; clusters can be easily scaled up by adding new nodes.

This is the general scheme of the Lab's network.

The fast server holds the `/CLUSTERDATA` and the `TRAJECTORIES,` and it can be used to transfer the data from `/CLUSTERDATA` without the need for a secure copy, thus copying it at a faster rate.

The compute cluster is assembled from two types of nodes that create two queues:

1. **GPU Queue** (gpu.q) - four nodes numbered from 2-0 to 2-3 with powerful graphics card processors. These nodes should be used if you run an all-atom simulation that can utilize the graphics processors

for subtasks and calculations.

2. **General Queue** (all.q) - these nodes do not have as powerful graphics cards as the GPU nodes. Thus they are used for simulations that do not require such power or simulations that run on a single processor (as in the case of CG-MD simulations).

## 2. Accessing Cluster remotely

Once you have a USERNAME in the Cluster, you can remotely connect to it as follows:

```
ssh USERNAME@Cluster
Password:
```

Type in the password for the USERNAME, and you're in. It's a good idea to have a shortcut for this action in your ~/.cshrc file: ``alias ssho "ssh -X Cluster". This way, you skip typing your username with each connection.

Upon connection, you land in your home directory on the cluster: /home/USERNAME/. This is your working directory to run a simulation. It's a good idea to create a directory for every project and keep the submission files organized as much as possible. For example, this is a list of all directories the **tutorial** user has under his home directory:

```
[tutorial@Cluster ~]$ ls
earth
[tutorial@Cluster ~]$ ls earth
coastlines
[tutorial@Cluster ~]$ ls earth/coastlines
fjords.dat MDWrapper.prefs
```

## 3. Cluster MDWrapper.prefs file

The only difference between the preference file used locally and on the cluster, is the `EXECUTION_LOCATION` flag that should be set to CLUSTER in case (surprise surprise) you are running the simulation on the cluster. Note that you should change the output path field as well - the output path should be your target directory under cluster data:

```
OUTPUT_PATH=/CLUSTERDATA/USERNAME/TARGET
```

## 4. Running the simulation

To submit your simulation to the Cluster, you first should copy your files to the home directory. That should be done with secure copy to the designated directory; with the following example, you can copy to your home directory in Cluster:

```
scp USERNAME@linklNUMBER:/PATH_TO_DAT_FILE .
scp USERNAME@linklNUMBER:/PATH_TO_MDWrapper.prefs .
```

Once you have your files in place, run the following MDWrapper script to submit the simulation:

```
/home/scripts/MD/wrapper/MDWrapper.pl
```

Now check that your simulation was indeed submitted and is running by typing the following command:

`qstat` to see that your job got submitted. You will get the following output if the job was submitted and running:

```
[USERNAME@Cluster ~]$ qstat
job-ID  prior  name  user state submit/start at   queue  slots ja-task-ID
-----------------------------------------------------------------------------------
 118224 0.50500 _fjords USERNAME  r   11/27/2022 12:20:44 all.q@compute-0-1.local  1
```

Otherwise, you will either get an empty status, or the job state will indicate an error occurred.

If you want to check how much time your simulation is going to run, you can ssh to the relevant compute node and find the `timeleft.dat` file in the simulation folder under scratch. For example:

```
ssh compute-0-1
cd scratch
cd SYM_FOLDER
vi timeleft.dat
```

If you discover some mistake in the job you submitted, you can delete the job by typing the `qdel job-ID` command with the relevant job-ID for your job.

If you want to delete multiple jobs that are numbered sequentially, you can run the following code using bash script:

```
for job_id in {1st_index..last_index}
do
    qdel $job_id
done
```

After the job finishes, it will not appear in the status. The next step is to make sure you copy the target directory from `CLUSTERDATA` to your directory under TRAJECTORIES.

```
scp -r USERNAME@Cluster:TARGET_PATH .
```

## 5. Post-simulation notes

After you make sure your simulation data are copied under `TRAJECTORIES`,**you must delete the data from the** `/CLUSTERDATA`. The issue with leaving the data hanging in the cluster data storage is that at some point, it will get full and new jobs will not be copied to their target directories, and thus **data would get lost**. To check if the storage is full, type the following command:

```
[USERNAME@Cluster]$ df -h .
Filesystem                         Size  Used Avail Use% Mounted on
10.1.1.7:/CLUSTERDATA/USERNAME  1.8T  1.5T  336G  82% /CLUSTERDATA/USERNAME
```

## 3. Run a protein-DNA simulation

In this section, you will run a simulation of a DNA binding protein SAP-1 and a static 50bp DNA molecule. Navigate to the DNA folder in your tutorial directory. Inside you will find the coordinates of the 50bp DNA molecule.

Download the structure with the pdbID of 1bc8 from the PDB site. Open the 1bc8 structure along with the 50bp DNA coordinates.

Align the protein close to DNA (carefully, not too close as there might be clashes) and save the 50bp DNA and the protein as a new pdb file (exclude the DNA molecule that was initially in the structure of 1bc8).

**Important note:** To create input for your protein-DNA simulation, you have to ensure that inside the pdb file, the protein coordinates are written first, and only after it are DNA coordinates. The reason for that is that the script that generates the MD input file creates parameters for the protein and ignores the DNA molecule at the first step. If your DNA molecule coordinates are written first, the script will crash.

Look into the `generateMDInput.prefs` file in the folder. Some preferences differ from the previous run; make sure you understand why.

Run the same script you ran in the previous section for 1srl protein. Ensure your new input file is correct. After the run, you will have only protein parameters in all sections except for electrostatics and coordinates.

Copy the input files to Cluster to a designated folder. Make sure you use MDWrapper.prefs file that has the `EXECUTION_LOCATION` flag set to CLUSTER.

Run the simulation similar to the previous section. When it finishes, copy all the data to the `TRAJECTORIES` and convert the trajectory dat file to a pdb trajectory and open it with pymol to see the movie.

**Important note:** When `EXCLUDE_STATIC_ATOMS` flag is set to YES, the static beads are not written in the trajectory (DNA beads in this case). To see DNA in the movie open the coordinates you used to generate the input file along with the trajectory. You might want to use it for analysis at some point. The flag was developed exclusively for DNA so if you have more than one dynamic region, the simulation will not write it properly!

You can plot the ES energy data to see the change in the energy upon binding. Does the protein move on the DNA molecule? What type of movement are you able to recognize in the movie?

# Common mistakes and ERRORS

*"For a moment, nothing happened. Then, after a second or so, nothing continued to happen."*

**— Douglas Adams, The Hitchhiker's Guide to the Galaxy**

---

## 1. In the generation of dat file

## 1. Invalid TEXT for PDB_FILE: pdbID.pdb .

```
Preferences file was found
Invalid TEXT for PDB_FILE: 1srl.pdb  .
```

If you made sure your input pdb file is in the folder, ensure there are no spaces after the pdb. The script takes the whole string that is designated to be the input pdb file.

## 2. Preferences file was not found

```
Preferences file was not found
Enter PDB_FILE (TEXT):
```

Make sure you did not rename the `generateMDInput.prefs`; if so, the script cannot access the file and extract the preferences. The preferences might be inserted manually in the absence of a prefs file.

## 3. Illegal division by zero

```
Use of uninitialized value in subtraction (-) at ~/scripts/MD/util/MDMath.pm line 164.
Illegal division by zero at ~/scripts/MD/util/MDMath.pm line 170.
```

You might want to check if the range of dynamic atoms matches the range of the C $\alpha$ or C $\alpha$-C $\beta$ beads in the file.

## 4. Undefined value ARRAY

```
Can't use an undefined value as an ARRAY reference at
~/scripts/MD/generateMDInput/Models/CA.pm line 959.
```

You're getting this error message due to the `HAS_PROTEIN_DNA_CONTACTS=YES` flag, while feeding the script with pdb file that lack DNA.

Another possible explanation for this result is that your protein is located too far from DNA, and the program struggles to find any contacts between the protein atoms and DNA atoms.

## 5. Cannot find the number of contacts in the CSU output file

```
Cannot find number of contacts in CSU output file at
~/scripts/MD/generateMDInput/Models/CA.pm line 701.
```

Any error that involves CSU indicates you have some inconsistencies between your preferences and the input PDB file. In this example, we are considering the number of chains, but this could also apply to other elements in the PDB file, such as the number of dynamic atoms, water molecules, ions, or ligands.

### 6. Bonded parameters were not created

You open your dat file and see this:

```
          0 Hookean bonded pairs.
          0 bond angle trios.
          0 dihedral quartets.
        137 contacts.
```

Dear God, what happened?

You might've used the `HAS_STATIC_ATOMS=NO` flag. Yes, I know, counter-intuitive. Try again after assigning `YES` to this flag and keep your dynamic atoms as the dynamic range.

## 2. In simulation execution

### 1. Error in ss.f file

```
READING THE .DAT FILE
At line 276 of file ../src/init-dsbm.f (unit = 30, file = '1srl.dat')
Fortran runtime error: Bad value during integer read
```

### 2. Abort message

```
 READING THE .DAT FILE

Program aborted. Backtrace:
#0  0x149493BE1697
#1  0x149493BE2E72
#2  0x149493CB7518
#3  0x40DA67 in init_
Abort (core dumped)
```

Abort messages usually have to do with the size of the system. Check the following things:

1. Your system is not over 7000 beads.
2. The number of contacts is at most 7000x25.
3. The number of bonds is at most 7000x2

If all these seem fine, your system crosses other size limits that you should trace. Look for clues in MD.com file.

### 3. Permission denied on scratch folder

In case you get any messages regarding permissions to scratch folders, you should check if your username has a folder under the `/scratch` folder. If not, talk to the root user in the lab.

---------- The tutorial was written and modified by Elena Rogoulenko (April 2022) ----------