

EXCOGITO

Generated by Doxygen 1.8.5

Fri Aug 4 2023 11:28:49



# Contents



## **Chapter 1**

# **EXCOGITO**



## Chapter 2

# python scripts

This folder contains a minimal conda environment that allows the user to:

- convert a GROMACS xtc file to an XYZ file (script `sample_convert_xtc_to_xyz.py`)
- generate a custom .ini parameter file (script `setup_parfile.py`)
- test the software (more info inside the *tests* folder)

### Installation

In order to install the software you must have `conda` installed.

Then, it is sufficient to run the following command:

```
"" conda env create --file conda_env_excogito.yml ""
```

to create the **excogito** environment. Once the packages are downloaded, **excogito** can be activated via:

```
"" conda activate excogito ""
```

### XTC to XYZ conversion

In order to convert a GROMACS XTC to XYZ you just need to run:

```
"" python3 sample_convert_xtc_to_xyz.py ""
```

Once you provide GROMACS XTC and GRO files and a reasonable name for your output, the script will perform the conversion making use of the `MDTraj 1.9.5` software

### Parameter file setup

#### Running

```
"" python3 setup_parfile.py ""
```

will help you with the setup of the *ini* parameter file needed by EXCOGITO.

### Contacts

Marco Giulini ([mr CGIULINI@gmail.com](mailto:mr CGIULINI@gmail.com))





## Chapter 3

# README

EXCOGITO is the program to investigate the mapping problem in coarse-grained modelling of biomolecules.

If you use EXCOGITO in your research please cite:

**EXCOGITO, an EXtensible COarse-Graining TOol**, M Giulini, R Fiorentini, L Tubiana, R Potestio, *in preparation*

**\*\*An Information-Theory-Based Approach for Optimal Model Reduction of Biomolecules\*\***, M Giulini, R Menichetti, MS Shell, R Potestio, *Journal of chemical theory and computation* 16 (11), 6795-6813

**\*\*A journey through mapping space: characterising the statistical and metric properties of reduced representations of macromolecules\*\***, R Menichetti, M Giulini, R Potestio, *The European Physical Journal B* 94 (10), 1-26

## 1. Installation

### 1.1 General requirements on Linux systems

The only requirement is to have **Openmp** installed on your machine.

### 1.2 Additional requirements on MAC OS

- Install **argp** by using homebrew. At the terminal, run this command: `“ brew install argp-standalone “`
- Install **xcode** if your version is higher than MacOS 10.7. You are not required to install the Xcode App from AppStore. At the terminal, just run this command (about 15 Gb are required free on your disk even though, at the end of installation, only 2 Gb will be consumed) `“ xcode-select –install “`

In order to have access to OpenMP libraries you can install **libomp** by using homebrew. At the terminal, run this command: `“ brew install libomp “`

### 1.3 Compiling

The code can be compiled using **CMake**. A minimal installation is obtained following these steps:

1. create a directory in *excogito*, such as *build* `“bash mkdir build cd build “`
2. run cmake from *build*, calling the outer directory `“bash cmake .. “`
3. run make `“bash make “`

### 1.3.2 Compilation options

Cmake allows to specify several options, such as the C compiler, compilation links and compilation flags. For instance, if the optimized Intel C compiler (icc) is available, step 2 may be substituted by: “`bash cmake .. -DCMAKE_C_COMPILER=icc -DCMAKE_C_FLAGS="-Ofast -fopenmp -I./include -mkl -xSSE4.2 -parallel -ipo -mcpu=native"`”

On MacOS, the C compiler identification should be AppleClang (check the first line printed on terminal after launching the command `cmake . .`).

## 2. Running

The typical usage of the program consists in a call to *excogito* with one of the following options:

- **optimize**: to optimize the coarse-grained mapping by minimising its mapping entropy;
- **random**: to randomly generate coarse-grained representations and measure the associated mapping entropies;
- **measure**: to measure the mapping entropy of a mapping provided by the user (in the form of a .txt file);
- **norm**: to calculate the norm of a mapping (provided by the user) throughout a trajectory;
- **cosine**: to calculate pairwise distance and cosine between a pair of mappings (provided by the user) throughout a trajectory;
- **distance**: to calculate the distance matrix between a data set of mappings (provided by the user) over a single conformation;
- **optimize\_kl**: to optimize the coarse-grained mapping by minimising its mapping entropy, calculated using the original Kullback-Leibler divergence;
- **random\_kl**: to randomly generate coarse-grained representations and measure the associated mapping entropies, calculated using the original Kullback-Leibler divergence;
- **measure\_kl**: to measure the mapping entropy of a mapping provided by the user (in the form of a .txt file), calculated using the original Kullback-Leibler divergence.

Each task can require different input files, which are provided to the program in the form of command-line options.

For further information, please type on terminal `./excogito --help` or `./excogito -h`

Alternatively, for printing a short usage message, please type: `./excogito --usage` or `./excogito -u`

After selecting which task is suitable for your purposes, read carefully the documentation below according to your choice.

### 2.1. Optimize Task

The **optimize** task requires the *protein code* string and three input files: *parameter*, *trajectory*, and *energy*.

In order to launch the **optimize** task follow this syntax:

```
“bash ./excogito optimize -p $parameter_file.ini -t $trajectory_file.xyz -e $energy_file.txt -c $prot_code
```

or

```
./excogito optimize -p $parameter_file.ini -t $trajectory_file.xyz -e $energy_file.txt -code $prot_code “
```

For further information, please type on terminal `./excogito optimize`

## 2.2. Random Task

The **random** task requires the *protein code* string and three input files: *parameter*, *trajectory*, and *energy*.

In order to launch the **random** task follow this syntax:

```
“bash ./excogito random -p $parameter_file.ini -t $trajectory_file.xyz -e $energy_file.txt -c $prot_code
```

or

```
./excogito random -p $parameter_file.ini -t $trajectory_file.xyz -e $energy_file.txt -code $prot_code “
```

For further information, please type on terminal `./excogito random`

## 2.3. Measure Task

The **measure** task requires the *protein code* string and four input files: *parameter*, *trajectory*, *energy*, and *mapping*.

In order to launch the **measure** task follow this syntax:

```
“bash ./excogito measure -p $parameter_file.ini -t $trajectory_file.xyz -e $energy_file.txt -c $prot_code -m $mapping_file.txt
```

or

```
./excogito measure -p $parameter_file.ini -t $trajectory_file.xyz -e $energy_file.txt -prot_code $prot_code -m1 $mapping_file.txt “
```

For further information, please type on terminal `./excogito measure`

## 2.4. Norm Task

The **norm** task requires the *protein code* string and three input files: *parameter*, *trajectory*, and *mapping*.

In order to launch the **norm** task follow this syntax:

```
“bash ./excogito norm -p $parameter_file.ini -t $trajectory_file.xyz -c $prot_code -m $mapping_file.txt
```

or

```
./excogito norm -p $parameter_file.ini -t $trajectory_file.xyz -prot_code $prot_code -m1 $mapping_file.txt “
```

For further information, please type on terminal `./excogito norm`

## 2.5. Cosine Task

The **cosine** task requires the *protein code* string and four input files: *parameter*, *trajectory*, *1st mapping*, and *2nd mapping*.

In order to launch the **cosine** task follow this syntax:

```
“bash ./excogito cosine -p $parameter_file.ini -t $trajectory_file.xyz -c $prot_code -m $mapping_file.txt -n $mapping_file2.txt
```

or

```
./excogito cosine -p $parameter_file.ini -t $trajectory_file.xyz -prot_code $prot_code -m1 $mapping_file.txt -m2 $mapping_file2.txt “
```

For further information, please type on terminal `./excogito cosine`

## 2.6. Distance Task

The **distance** task requires the *protein code* string and three input files: *parameter*, *trajectory*, *mapping matrix*.

In order to launch the **distance** task follow this syntax:

```
““bash ./excogito distance -p $parameter_file.ini -t $trajectory_file.xyz -c $prot_code -x $mapping_matrix_file.txt
or
./excogito distance -p $parameter_file.ini -t $trajectory_file.xyz -prot_code $prot_code -matrix $mapping_matrix-
_file.txt
““
```

For further information, please type on terminal `./excogito distance`

## 2.7. Optimize\_kl Task

The **optimize\_kl** task requires the *protein code* string and three input files: *parameter*, *trajectory*, and *probability*.

In order to launch the **optimize\_kl** task follow this syntax:

```
““bash ./excogito optimize -p $parameter_file.ini -t $trajectory_file.xyz -r $probability_file.txt -c $prot_code
or
./excogito optimize -p $parameter_file.ini -t $trajectory_file.xyz -probs $probability_file.txt -code $prot_code ““
```

For further information, please type on terminal `./excogito optimize_kl`

## 2.8. Random\_kl Task

The **random\_kl** task requires the *protein code* string and three input files: *parameter*, *trajectory*, and *probability*.

In order to launch the **random\_kl** task follow this syntax:

```
““bash ./excogito random_kl -p $parameter_file.ini -t $trajectory_file.xyz -r $probability_file.txt -c $prot_code
or
./excogito random_kl -p $parameter_file.ini -t $trajectory_file.xyz -probs $probability_file.txt -code $prot_code ““
```

For further information, please type on terminal `./excogito random_kl`

## 2.9. Measure\_kl Task

The **measure\_kl** task requires the *protein code* string and four input files: *parameter*, *trajectory*, *probability*, and *mapping*.

In order to launch the **measure\_kl** task follow this syntax:

```
““bash ./excogito measure_kl -p $parameter_file.ini -t $trajectory_file.xyz -r $probability_file.txt -c $prot_code -m
$mapping_file.txt
or
./excogito measure_kl -p $parameter_file.ini -t $trajectory_file.xyz -probs $probability_file.txt -prot_code $prot_
code -m1 $mapping_file.txt ““
```

For further information, please type on terminal `./excogito measure_kl`

## 2.10. Optimize\_spins Task

The **optimize\_spins** task requires the *protein code* string and three input files: *parameter*, *trajectory*, and *probability*.

In order to launch the **optimize\_spins** task follow this syntax:

```
““bash ./excogito optimize_spins -p $parameter_file.ini -t $trajectory_file.xyz -r $probability_file.txt -c $prot_code
or
```

```
./excogito measure_kl -p $parameter_file.ini -t $trajectory_file.xyz -probs $probability_file.txt -prot_code $prot_code ""
```

Here the trajectory is simply given by the states of a discrete system (like [this](#)).

For further information, please type on terminal `./excogito optimize_spins`.

### 3. Which arguments are mandatory? A short explanation

As shown in **Section 2.x**, the *protein code* string and two files are always mandatory, namely the *parameter file* and the *xyz trajectory file*. The other files can be mandatory, depending on the chosen task.

What are these files?

- **\*\*\$parameter\_file.ini\*\*** → Set of parameters in *ini* format for the algorithm (see 3.1). Examples are present in *\*/examples/parameters\**;
- **\*\*\$trajectory\_file.xyz\*\*** → Trajectory in *xyz* format (see the Section 3.2). An example is present in *\*/examples/trajectories\**;
- **\*\*\$energy\_file.txt\*\*** → File with the energies corresponding to each configuration in the trajectory (see the Section 3.3). An example is present in *\*/examples/energies\**;
- **\*\*\$prot\_code\*\*** → Unique string that identifies the structure (see 3.4). It will be used to generate the output files;
- **\*\*\$mapping\_file.txt\*\*** → Mapping file, containing the indices of the retained atoms (see 3.5). An example is present in *\*/examples/mappings\**;
- **\*\*\$mapping\_file2.txt\*\*** → 2<sup>nd</sup> Mapping file, containing the indices of the retained atoms (see 3.5). An example is present in *\*/examples/mappings\**;
- **\*\*\$mapping\_matrix\_file.txt\*\*** → Matrix with *n\_mappings* CG mappings (see 3.6).
- **\*\*\$probability\_file.txt\*\*** → File with the probabilities corresponding to each configuration in the trajectory (see 3.7). They must sum to 1.0. An example is present in *\*/examples/probabilities\**;

#### 3.1. Parameter FILE

The core element of EXCOGITO is the parameter file, which is employed to define the constants used in the different tasks.

A sample parameter file for each task can be found in *\*/examples/parameters*.

There exist 16 parameters, but only few of them are mandatory for the selected task. They are illustrated in the following table:

Parameter	Description	Type	Mandatory	Suggested value
atomnum	number of atoms in the system	int	all	
frames	number of frames in the trajectory	int	all	< 5000 on laptops, < 15000 if criterion != 1
cgnum	number of CG sites	int	all	between atomnum/20 and atomnum/2

criterion	criterion for clustering	int	O-R-M	0, 1, 2, 3
nclust	number of CG macrostates	int	C0 - C3	between frames/500, and frames/100
n_mappings	number of mappings in tasks <b>random</b> and <b>distance</b>	int	R-D	
MC_steps	number of MC step in task <b>optimize</b>	int	O	> 5000
rotmats_period	MC steps between two full alignments in task <b>optimize</b>	int	O	
t_zero	starting temperature in task <b>optimize</b>	double	O	
distance	cophenetic distance threshold	double	C3	
max_nclust	upper number of clusters	int	C2	between frames/100 and frames/50
min_nclust	lower number of clusters	int	C2	between frames/1000 and frames/500 (must be < max_nclust)
Ncores	number of cores	int	no	
decay_time	governs temperature decay in task <b>optimize</b>	double	O	
rsd	use rsd (if 1) instead of rmsd (if 0)	int	no	
stride	number of structures between two pivot configurations	int	C1	~ 10 if frames between 1000 and 10000

O-R-M-D refer to the tasks (optimize/optimize\_kl, random/random\_kl, measure/measure\_kl, distance) in which the parameter is mandatory. C0 .. C3 indicates that the parameter is mandatory if the clustering criterion is equal to 0 .. 3, respectively.

## Clustering

Four criteria for hierarchical clustering:

- **0** *Maxclust* clustering: configurations are lumped into *Nclust* macrostates;
- **1** *Fast clustering*: as in criterion **0**, but applied to a set of pivot configurations. Labels of intermediate structures are assigned to the closer pivot;
- **2** *Multiple maxclust*: as described in *Giulini et al.* (JCTC, 2020);
- **3** *Maxdist* clustering: clustering with the cophenetic distance;

## 3.2. Trajectory FILE

The trajectory should be provided in the xyz format. The first line of each frame indicates the number of atoms, while the second can contain an arbitrary string. As an example, a trajectory with 2 frames and 3 atoms should

resemble the following string:

```
“ 3
X 2.53 2.09 3.55 X 2.57 1.95 3.51 X 2.45 1.87 3.46 3
X 2.69 1.96 3.40 X 2.80 1.91 3.43 X 2.67 2.03 3.28 “
```

In the *python* subdirectory there is a script that helps with the conversion from GROMACS XTC to the XYZ format.

### 3.3. Energy FILE

Energy files, mandatory for tasks **optimize**, **random**, and **measure**, should contain one value for each frame in the trajectory.

### 3.4 Protein Code

The protein code is a string that is used to create output files. Don't insert spaces or special characters in this string

### 3.5 Mapping FILES

A mapping file, mandatory for tasks **measure**, **norm**, and **cosine** is a file with an integer per line. The value correspond to the index of the atom in the xyz trajectory. As an example, a mapping with 8 sites on a peptide of 50 sites should respect the following format:

```
“ 3 7 19 21 26 34 40 47 “
```

### 3.6. Mapping Matrix FILES

A mapping matrix is mandatory for task **distance**. It is simply a series of transposed mappings. If we aim at computing the distance matrix between three mappings with 8 sites on a peptide of 50 sites, we must respect the following syntax:

```
“ 3 7 19 21 26 34 40 47 2 8 19 24 25 38 41 44 0 10 12 20 29 31 35 49 “
```

### 3.7. Probability FILE

Probability files, mandatory for tasks **optimize\_kl**, **random\_kl**, and **measure\_kl**, must contain one value for each frame in the trajectory and should be properly normalized to 1. For a trajectory of 5 frames, the following file is acceptable:

```
“ 0.1 0.15 0.6 0.05 0.1 “
```

## 4. Examples

Inside the directory examples there are example files for the *6d93* protein, allowing the user to try all the different tasks:

- **optimize:** `./build/excogito optimize -p examples/parameters/parameters_optimize_6d93_N31_small.ini -t examples/trajectories/6d93_100frames.xyz -e examples/energies/6d93_energies_100frames.txt -c 6d93`
- **random:** `./build/excogito random -p examples/parameters/parameters_random_6d93_N31_small.ini -t examples/trajectories/6d93_100frames.xyz -e examples/energies_100frames.txt -c 6d93`

- **measure:** `./build/excogito measure -p examples/parameters/parameters-_loadca_6d93_N31.ini -t examples/trajectories/6d93_1000frames.xyz -e examples/energies/6d93_energies_1000frames.txt -c 6d93 -m examples/mappings/tamapin_ca_mapping.txt`
- **norm:** `./build/excogito norm -p examples/parameters/parameters_norm_6d93_N31.ini -t examples/trajectories/6d93_1000frames.xyz -e examples/energies/6d93_energies_1000frames.txt -c 6d93 -m examples/mappings/tamapin_ca_mapping.txt`
- **cosine:** `./build/excogito cosine -p ./examples/parameters/parameters_cosine_6d93_N31.ini -t ./examples/trajectories/6d93_1000frames.xyz -e ./examples/energies/6d93_energies_1000frames.txt -c 6d93 -m ./examples/mappings/tamapin_nextca_mapping.txt --m2 ./examples/mappings/tamapin_nextca_mapping.txt`
- **distance:** `./build/excogito distance -p examples/parameters/parameters_distance_6d93_N31.ini -t ./examples/trajectories/6d93_1frame.xyz -x examples/mappings/6d93_mapping_matrix.txt -c 6d93`
- **optimize:** `./build/excogito optimize_kl -p examples/parameters/parameters_optimizekl_6d93_N31_notemp.ini -t examples/trajectories/6d93_100frames.xyz -r examples/probabilities/6d93_probs_100frames.txt -c 6d93`
- **random\_kl:** `./build/excogito random_kl -p examples/parameters/parameters_randomkl_6d93_N31.ini -t examples/trajectories/6d93_100frames.xyz -r examples/probabilities/6d93_probs_100frames.txt -c 6d93`
- **measure\_kl:** `./build/excogito measure_kl -p examples/parameters/parameters_measurekl_6d93_N31.ini -t examples/trajectories/6d93_100frames.xyz -r examples/probabilities/6d93_probs_100frames.txt -c 6d93 -m examples/mappings/tamapin_ca_mapping.txt`
- **optimize\_spins** `./excogito optimize_spins -p ../examples/parameters/parameters_spins_m1.ini -t ../examples/trajectories/m1_spins_grouped.csv -r ../examples/probabilities/probs.txt -c m1`

## 5. Scaling values

The approximated mapping entropy is calculated (tasks **optimize**, **random** and **measure**) without the scaling factor (see. [Giulini et al.](#)). This factor should be computed by the user according to the temperature employed to simulate the system.

## 6. Documentation

File `refman.pdf` in the `docs` directory contains detailed documentation automatically generated with doxygen version 1.8.5.

A custom documentation can be generated in `html` and `tex` format by running `doxygen excogito_doxygen.conf`.

## 7. Contacts

Marco Giulini ([mr CGIULINI@gmail.com](mailto:mr CGIULINI@gmail.com)) Raffaello Potestio ([raffaello.potestio@unitn.it](mailto:raffaello.potestio@unitn.it))



## Chapter 4

# How to test the software?

We employ python `unittest` class to test our code. The file `test_suite.py` contains some unittest Test Cases that should be run in order to be sure that the compilation went successfully.

```
“bash python3 test_suite.py “
```

Or, for verbose output: “bash python3 `test_suite.py` -v “

If everything went smoothly the output should look like the following:

```
“bash ... Ran 20 tests in 0.140s
```

```
OK “
```

`test_suite.py` makes use of the following packages:

- unittest
- pathlib
- os
- subprocess



## Chapter 5

# Namespace Index

### 5.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">sample_convert_xtc_to_xyz</a>	. . . . .	??
<a href="#">setup_parfile</a>	. . . . .	??
<a href="#">test_suite</a>	. . . . .	??



## Chapter 6

# Hierarchical Index

### 6.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

alignment	??
alignments	??
arguments	??
cg_mapping	??
cg_mapping_lib	??
clust_params	??
geometry	??
hierarchical_clustering	??
ini_parse_string_ctx	??
io	??
MC_params	??
observables	??
parameters	??
spin_traj	??
TestCase	
test_suite.test0	??
test_suite.test1	??
test_suite.test10	??
test_suite.test11	??
test_suite.test12	??
test_suite.test13	??
test_suite.test14	??
test_suite.test15	??
test_suite.test16	??
test_suite.test17	??
test_suite.test18	??
test_suite.test19	??
test_suite.test2	??
test_suite.test20	??
test_suite.test21	??
test_suite.test22	??
test_suite.test23	??
test_suite.test24	??
test_suite.test25	??
test_suite.test26	??
test_suite.test27	??
test_suite.test3	??
test_suite.test4	??

test_suite.test5	. . . . .	??
test_suite.test6	. . . . .	??
test_suite.test7	. . . . .	??
test_suite.test8	. . . . .	??
test_suite.test9	. . . . .	??
traj	. . . . .	??

## Chapter 7

# Class Index

### 7.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">alignment</a>	Library of functions that perform alignments of pairs of structures . . . . .	??
<a href="#">alignments</a>	Structure that defines the current alignments stored in memory . . . . .	??
<a href="#">arguments</a>	. . . . .	??
<a href="#">cg_mapping</a>	Structure that defines a cg mapping . . . . .	??
<a href="#">cg_mapping_lib</a>	Library of functions that perform simple operations on CG mappings . . . . .	??
<a href="#">clust_params</a>	Structure that defines the parameters for hierarchical clustering . . . . .	??
<a href="#">geometry</a>	Library of functions that perform simple geometrical calculations . . . . .	??
<a href="#">hierarchical_clustering</a>	Library of functions that perform hierarchical clustering . . . . .	??
<a href="#">ini_parse_string_ctx</a>	. . . . .	??
<a href="#">io</a>	Library of functions for all input-output operations . . . . .	??
<a href="#">MC_params</a>	Structure that defines a the parameters of Monte Carlo sampling . . . . .	??
<a href="#">observables</a>	Library of functions for the calculation of several observables . . . . .	??
<a href="#">parameters</a>	. . . . .	??
<a href="#">spin_traj</a>	. . . . .	??
<a href="#">test_suite.test0</a>	. . . . .	??
<a href="#">test_suite.test1</a>	. . . . .	??
<a href="#">test_suite.test10</a>	. . . . .	??
<a href="#">test_suite.test11</a>	. . . . .	??
<a href="#">test_suite.test12</a>	. . . . .	??
<a href="#">test_suite.test13</a>	. . . . .	??
<a href="#">test_suite.test14</a>	. . . . .	??
<a href="#">test_suite.test15</a>	. . . . .	??
<a href="#">test_suite.test16</a>	. . . . .	??
<a href="#">test_suite.test17</a>	. . . . .	??
<a href="#">test_suite.test18</a>	. . . . .	??
<a href="#">test_suite.test19</a>	. . . . .	??
<a href="#">test_suite.test2</a>	. . . . .	??
<a href="#">test_suite.test20</a>	. . . . .	??

<a href="#">test_suite.test21</a>	. . . . .	??
<a href="#">test_suite.test22</a>	. . . . .	??
<a href="#">test_suite.test23</a>	. . . . .	??
<a href="#">test_suite.test24</a>	. . . . .	??
<a href="#">test_suite.test25</a>	. . . . .	??
<a href="#">test_suite.test26</a>	. . . . .	??
<a href="#">test_suite.test27</a>	. . . . .	??
<a href="#">test_suite.test3</a>	. . . . .	??
<a href="#">test_suite.test4</a>	. . . . .	??
<a href="#">test_suite.test5</a>	. . . . .	??
<a href="#">test_suite.test6</a>	. . . . .	??
<a href="#">test_suite.test7</a>	. . . . .	??
<a href="#">test_suite.test8</a>	. . . . .	??
<a href="#">test_suite.test9</a>	. . . . .	??
<a href="#">traj</a>		
Structure that defines a MD trajectory	. . . . .	??



## Chapter 8

# File Index

### 8.1 File List

Here is a list of all files with brief descriptions:

<a href="#">excogito.c</a>	??
include/ <a href="#">alignment.h</a>	??
include/ <a href="#">cosine.h</a>	??
include/ <a href="#">distance.h</a>	??
include/ <a href="#">geometry.h</a>	??
include/ <a href="#">hierarchical_clustering.h</a>	??
include/ <a href="#">ini.h</a>	??
include/ <a href="#">io.h</a>	??
include/ <a href="#">mapping.h</a>	??
include/ <a href="#">measure.h</a>	??
include/ <a href="#">measure_kl.h</a>	??
include/ <a href="#">my_malloc.h</a>	??
include/ <a href="#">norm.h</a>	??
include/ <a href="#">observables.h</a>	??
include/ <a href="#">optimize.h</a>	??
include/ <a href="#">optimize_kl.h</a>	??
include/ <a href="#">optimize_spins.h</a>	??
include/ <a href="#">random_mapping.h</a>	??
include/ <a href="#">random_mapping_kl.h</a>	??
include/ <a href="#">sampling.h</a>	??
include/ <a href="#">traj.h</a>	??
lib/ <a href="#">alignment.c</a>	??
lib/ <a href="#">cosine.c</a>	??
lib/ <a href="#">distance.c</a>	??
lib/ <a href="#">geometry.c</a>	??
lib/ <a href="#">hierarchical_clustering.c</a>	??
lib/ <a href="#">ini.c</a>	??
lib/ <a href="#">io.c</a>	??
lib/ <a href="#">mapping.c</a>	??
lib/ <a href="#">measure.c</a>	??
lib/ <a href="#">measure_kl.c</a>	??
lib/ <a href="#">my_malloc.c</a>	??
lib/ <a href="#">norm.c</a>	??
lib/ <a href="#">observables.c</a>	??
lib/ <a href="#">optimize.c</a>	??
lib/ <a href="#">optimize_kl.c</a>	??
lib/ <a href="#">optimize_spins.c</a>	??
lib/ <a href="#">random_mapping.c</a>	??

---

lib/ <a href="#">random_mapping_kl.c</a>	??
lib/ <a href="#">sampling.c</a>	??
lib/ <a href="#">traj.c</a>	??
python/ <a href="#">sample_convert_xtc_to_xyz.py</a>	??
python/ <a href="#">setup_parfile.py</a>	??
tests/ <a href="#">test_suite.py</a>	??

## Chapter 9

# Namespace Documentation

### 9.1 sample\_convert\_xtc\_to\_xyz Namespace Reference

#### Variables

- tuple `xtc_path` = input("insert path to XTC file\n")
- tuple `gro_path` = input("insert path to GRO file\n")
- tuple `xyz_filename` = input("insert path to output XYZ file\n")
- tuple `full_traj` = mdtraj.load\_xtc(xtc\_path.strip(),top=gro\_path.strip())
- tuple `full_traj_topology` = full\_traj.topology
- tuple `no_h` = full\_traj\_topology.select('type != H')
- tuple `n_heavy_traj` = len(`no_h`)
- tuple `mdt_tr_heavy` = mdtraj.load\_xtc(`xtc_path`, top=`gro_path`, atom\_indices = list(`no_h`))

#### 9.1.1 Variable Documentation

9.1.1.1 tuple `sample_convert_xtc_to_xyz.xtc_path` = input("insert path to XTC file\n")

9.1.1.2 tuple `sample_convert_xtc_to_xyz.gro_path` = input("insert path to GRO file\n")

9.1.1.3 tuple `sample_convert_xtc_to_xyz.xyz_filename` = input("insert path to output XYZ file\n")

9.1.1.4 tuple `sample_convert_xtc_to_xyz.full_traj` = mdtraj.load\_xtc(xtc\_path.strip(),top=gro\_path.strip())

9.1.1.5 `sample_convert_xtc_to_xyz.full_traj_topology` = full\_traj.topology

9.1.1.6 tuple `sample_convert_xtc_to_xyz.no_h` = full\_traj\_topology.select('type != H')

9.1.1.7 tuple `sample_convert_xtc_to_xyz.n_heavy_traj` = len(`no_h`)

9.1.1.8 tuple `sample_convert_xtc_to_xyz.mdt_tr_heavy` = mdtraj.load\_xtc(`xtc_path`, top=`gro_path`, atom\_indices = list(`no_h`))

### 9.2 setup\_parfile Namespace Reference

#### Functions

- def `retrieve_parameter`
- def `get_mandatory_parameters`

- def [get\\_optional\\_parameters](#)
- def [write\\_parameters](#)

## Variables

- list [tasks](#) = ["optimize", "random", "measure", "norm", "cosine", "distance", "optimize\_kl", "measure\_kl"]
- dictionary [mandatory\\_pars](#)
- dictionary [optional\\_pars](#)
- dictionary [pars\\_description](#)
- dictionary [pars\\_type](#)
- dictionary [clustering\\_pars](#)
- tuple [task](#) = input(f"Insert the task you would like to perform among the following: {str(tasks)}{os.linesep}")
- dictionary [my\\_pars](#) = {}
- tuple [opt](#) = input("Insert optional [parameters](#)? (y/n)")

## 9.2.1 Function Documentation

9.2.1.1 def [setup\\_parfile.retrieve\\_parameter](#) ( *par\_name*, *par\_type*, *par\_desc* )

9.2.1.2 def [setup\\_parfile.get\\_mandatory\\_parameters](#) ( *task* )

9.2.1.3 def [setup\\_parfile.get\\_optional\\_parameters](#) ( *task* )

9.2.1.4 def [setup\\_parfile.write\\_parameters](#) ( *task*, *pars\_dict* )

## 9.2.2 Variable Documentation

9.2.2.1 list [setup\\_parfile.tasks](#) = ["optimize", "random", "measure", "norm", "cosine", "distance", "optimize\_kl", "measure\_kl"]

9.2.2.2 dictionary [setup\\_parfile.mandatory\\_pars](#)

### Initial value:

```
1 = {
2     "optimize" : [ "atomnum", "frames", "cgnum", "MC_steps", "rotmats_period", "Ncores" ],
3     "random" : [ "atomnum", "frames", "cgnum", "n_mappings" ],
4     "measure" : [ "atomnum", "frames", "cgnum" ],
5     "norm": [ "atomnum", "frames", "cgnum" ],
6     "cosine" : [ "atomnum", "frames", "cgnum" ],
7     "distance" : [ "atomnum", "frames", "cgnum", "n_mappings" ],
8     "optimize_kl" : [ "atomnum", "frames", "cgnum", "MC_steps", "Ncores" ],
9     "measure_kl" : [ "atomnum", "frames", "cgnum" ]
10 }
```

9.2.2.3 dictionary [setup\\_parfile.optional\\_pars](#)

### Initial value:

```
1 = {
2     "optimize" : [ "criterion", "t_zero", "decay_time" ],
3     "random" : [ "criterion" ],
4     "measure" : [ "criterion" ],
5     "norm": [],
6     "cosine" : [],
7     "distance" : [],
8     "optimize_kl" : [ "criterion", "t_zero", "decay_time" ],
9     "measure_kl" : [ "criterion" ]
10 }
```

## 9.2.2.4 dictionary setup\_parfile.pars\_description

**Initial value:**

```

1 = {
2   "atomnum" : "number of atoms in your structure",
3   "frames" : "number of frames in your trajectory",
4   "cgnum" : "number of coarse-grained sites",
5   "n_mappings": "number of coarse-grained mappings",
6   "MC_steps" : "number of Monte Carlo steps",
7   "rotmats_period" : "steps between two calculations of rotation matrices ",
8   "criterion" : ""criterion for hierarchical clustering      0 : fixed number of clusters      1 :
fast clustering on a subset of configurations (only for continuous trajectories)      2 : multiple numbers
of clusters      3 : cophenetic distance"",
9   "Ncores" : "number of cores to use",
10  "t_zero" : "starting temperature for Simulated Annealing",
11  "nclust" : "number of clusters",
12  "distance" : "cophenetic distance",
13  "max_nclust" : "upper limit to the number of clusters",
14  "min_nclust" : "lower limit to the number of clusters",
15  "decay_time" : "temperature decay for Simulated Annealing",
16  "rsd" : "use RSD instead of RMSD",
17  "stride" : "number of structures between two pivot configurations"
18 }

```

## 9.2.2.5 dictionary setup\_parfile.pars\_type

**Initial value:**

```

1 = {
2   "atomnum" : int,
3   "frames" : int,
4   "cgnum" : int,
5   "n_mappings": int,
6   "MC_steps" : int,
7   "rotmats_period" : int,
8   "criterion" : int,
9   "Ncores" : int,
10  "t_zero" : float,
11  "criterion" : int,
12  "nclust" : int,
13  "distance" : float,
14  "max_nclust" : int,
15  "min_nclust" : int,
16  "Ncores" : int,
17  "decay_time" : float,
18  "rsd" : int,
19  "stride" : int
20 }

```

## 9.2.2.6 dictionary setup\_parfile.clustering\_pars

**Initial value:**

```

1 = {
2   0: [ "nclust" ],
3   1: [ "stride", "nclust"],
4   2: [ "min_nclust", "max_nclust" ],
5   3: [ "distance" ]
6 }

```

9.2.2.7 tuple setup\_parfile.task = input(f"Insert the task you would like to perform among the following:  
{str(tasks)}{os.linesep}")

9.2.2.8 dictionary setup\_parfile.my\_pars = {}

9.2.2.9 tuple setup\_parfile.opt = input("Insert optional parameters? (y/n)")

## 9.3 test\_suite Namespace Reference

### Classes

- class [test0](#)
- class [test1](#)
- class [test2](#)
- class [test3](#)
- class [test4](#)
- class [test5](#)
- class [test6](#)
- class [test7](#)
- class [test8](#)
- class [test9](#)
- class [test10](#)
- class [test11](#)
- class [test12](#)
- class [test13](#)
- class [test14](#)
- class [test15](#)
- class [test16](#)
- class [test17](#)
- class [test18](#)
- class [test19](#)
- class [test20](#)
- class [test21](#)
- class [test22](#)
- class [test23](#)
- class [test24](#)
- class [test25](#)
- class [test26](#)
- class [test27](#)

### Variables

- tuple [t\\_start](#) = dt.datetime.now()
- tuple [bash\\_script](#) = subprocess.Popen("./test\_suite.sh", shell=True, stdout=subprocess.PIPE)

#### 9.3.1 Detailed Description

```
\class test_suite
```

The file contains several python tests to check the correct installation of METool package.

#### 9.3.2 Variable Documentation

9.3.2.1 tuple `test_suite.t_start = dt.datetime.now()`

9.3.2.2 tuple `test_suite.bash_script = subprocess.Popen("./test_suite.sh", shell=True, stdout=subprocess.PIPE)`

# Chapter 10

## Class Documentation

### 10.1 alignment Class Reference

library of functions that perform alignments of pairs of structures

#### 10.1.1 Detailed Description

library of functions that perform alignments of pairs of structures

The documentation for this class was generated from the following file:

- [lib/alignment.c](#)

### 10.2 alignments Class Reference

structure that defines the current alignments stored in memory

```
#include <alignment.h>
```

#### Public Attributes

- double \* [rmsd\\_mat](#)
- double \*\* [rotation\\_matrices](#)
- double \*\* [coms](#)
- int [rsd](#)
- double \* [rmsd\\_vector](#)
- double \*\* [rotation\\_matrices\\_vector](#)

#### 10.2.1 Detailed Description

structure that defines the current alignments stored in memory

#### 10.2.2 Member Data Documentation

##### 10.2.2.1 double\* alignments::rmsd\_mat

condensed pairwise RMSD matrix

**10.2.2.2 double\*\* alignments::rotation\_matrices**

condensed matrix of pairwise rotation matrices

**10.2.2.3 double\*\* alignments::coms**

array of centers of mass

**10.2.2.4 int alignments::rsd**

RSD parameter. {0: use the RMSD, 1: use the RSD}

**10.2.2.5 double\* alignments::rmsd\_vector**

RMSD vector for fast, 1D, clustering

**10.2.2.6 double\*\* alignments::rotation\_matrices\_vector**

vector of pairwise rotation matrices

The documentation for this class was generated from the following file:

- include/[alignment.h](#)

**10.3 arguments Struct Reference**

```
#include <io.h>
```

**Public Attributes**

- int [silent](#)
- int [verbose](#)
- char \* [parameter\\_file](#)
- char \* [energy\\_file](#)
- char \* [mapping\\_file](#)
- char \* [mapping\\_file2](#)
- char \* [trajectory\\_file](#)
- char \* [prot\\_code](#)
- char \* [task](#)
- char \* [mapping\\_matrix](#)
- char \* [probability\\_file](#)

**10.3.1 Member Data Documentation****10.3.1.1 int arguments::silent****10.3.1.2 int arguments::verbose****10.3.1.3 char\* arguments::parameter\_file**

input parameter file



10.3.1.4 `char* arguments::energy_file`

input energy file

10.3.1.5 `char* arguments::mapping_file`

input first mapping file

10.3.1.6 `char* arguments::mapping_file2`

input second mapping file

10.3.1.7 `char* arguments::trajectory_file`

input trajectory file

10.3.1.8 `char* arguments::prot_code`

protein code

10.3.1.9 `char* arguments::task`

task: (optimize, random, measure, norm, cosine, distance)

10.3.1.10 `char* arguments::mapping_matrix`

input mapping matrix

10.3.1.11 `char* arguments::probability_file`

input probability file

The documentation for this struct was generated from the following file:

- [include/io.h](#)

## 10.4 cg\_mapping Class Reference

structure that defines a cg mapping

```
#include <mapping.h>
```

### Public Attributes

- `int n_at`
- `int n_cg`
- `int * mapping`
- `double smap`
- `int * clusters`
- `int * size`
- `double * norms`

### 10.4.1 Detailed Description

structure that defines a cg mapping

### 10.4.2 Member Data Documentation

#### 10.4.2.1 `int cg_mapping::n_at`

number of atoms in the atomistic structure

#### 10.4.2.2 `int cg_mapping::n_cg`

number of CG sites

#### 10.4.2.3 `int* cg_mapping::mapping`

binary array defining the CG mapping

#### 10.4.2.4 `double cg_mapping::smap`

value of mapping entropy

#### 10.4.2.5 `int* cg_mapping::clusters`

array CG macrostates

#### 10.4.2.6 `int* cg_mapping::size`

sizes of CG macrostates

#### 10.4.2.7 `double* cg_mapping::norms`

moduli of CG mapping over the trajectory

The documentation for this class was generated from the following file:

- `include/mapping.h`

## 10.5 `cg_mapping_lib` Class Reference

library of functions that perform simple operations on CG mappings

### 10.5.1 Detailed Description

library of functions that perform simple operations on CG mappings

The documentation for this class was generated from the following file:

- `lib/mapping.c`

## 10.6 clust\_params Class Reference

structure that defines the parameters for hierarchical clustering

```
#include <hierarchical_clustering.h>
```

### Public Attributes

- int [crit](#)
- int [ncl](#)
- int [max\\_ncl](#)
- int [min\\_ncl](#)
- double [c\\_distance](#)

### 10.6.1 Detailed Description

structure that defines the parameters for hierarchical clustering

### 10.6.2 Member Data Documentation

#### 10.6.2.1 int clust\_params::crit

criterion for clustering structures. {0: single nclust, 1: distance-based, 2: multiple nclust, 3: fast clustering}

#### 10.6.2.2 int clust\_params::ncl

number of clusters (if crit is 0)

#### 10.6.2.3 int clust\_params::max\_ncl

maximum number of clusters (if crit is 2)

#### 10.6.2.4 int clust\_params::min\_ncl

minimum number of clusters (if crit is 2)

#### 10.6.2.5 double clust\_params::c\_distance

maximum cophenetic distance (if crit is 1)

The documentation for this class was generated from the following file:

- include/[hierarchical\\_clustering.h](#)

## 10.7 geometry Class Reference

library of functions that perform simple geometrical calculations

### 10.7.1 Detailed Description

library of functions that perform simple geometrical calculations

The documentation for this class was generated from the following file:

- [lib/geometry.c](#)

## 10.8 hierarchical\_clustering Class Reference

library of functions that perform hierarchical clustering

### 10.8.1 Detailed Description

library of functions that perform hierarchical clustering

Credits to scipy authors:

Copyright (C) Damian Eads, 2007-2008. New BSD License.

hierarchy.py (derived from cluster.py, <http://scipy-cluster.googlecode.com>)

Author: Damian Eads Date: September 22, 2007

Copyright (c) 2007, 2008, Damian Eads

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the author nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The documentation for this class was generated from the following file:

- [lib/hierarchical\\_clustering.c](#)

## 10.9 ini\_parse\_string\_ctx Struct Reference

### Public Attributes

- const char \* [ptr](#)
- size\_t [num\\_left](#)

### 10.9.1 Member Data Documentation

10.9.1.1 `const char* ini_parse_string_ctx::ptr`

10.9.1.2 `size_t ini_parse_string_ctx::num_left`

The documentation for this struct was generated from the following file:

- lib/[ini.c](#)

## 10.10 io Class Reference

library of functions for all input-output operations

### 10.10.1 Detailed Description

library of functions for all input-output operations

The documentation for this class was generated from the following file:

- lib/[io.c](#)

## 10.11 MC\_params Class Reference

structure that defines a the parameters of Monte Carlo sampling

```
#include <sampling.h>
```

### Public Attributes

- double [t\\_zero](#)
- double [decay\\_time](#)
- int [rotmats\\_period](#)
- int [MC\\_steps](#)

### 10.11.1 Detailed Description

structure that defines a the parameters of Monte Carlo sampling

### 10.11.2 Member Data Documentation

10.11.2.1 `double MC_params::t_zero`

starting effective temperature

10.11.2.2 `double MC_params::decay_time`

decay parameter

### 10.11.2.3 int MC\_params::rotmats\_period

Simulated Annealing steps between two updates of the alignments

### 10.11.2.4 int MC\_params::MC\_steps

number of MC steps

The documentation for this class was generated from the following file:

- include/[sampling.h](#)

## 10.12 observables Class Reference

library of functions for the calculation of several observables

### 10.12.1 Detailed Description

library of functions for the calculation of several observables

The documentation for this class was generated from the following file:

- lib/[observables.c](#)

## 10.13 parameters Struct Reference

```
#include <io.h>
```

### Public Attributes

- int [atomnum](#)
- int [frames](#)
- int [cgnum](#)
- int [nclust](#)
- int [n\\_mappings](#)
- int [MC\\_steps](#)
- int [rotmats\\_period](#)
- float [t\\_zero](#)
- float [distance](#)
- int [criterion](#)
- int [max\\_nclust](#)
- int [min\\_nclust](#)
- int [Ncores](#)
- float [decay\\_time](#)
- int [rsd](#)
- int [stride](#)
- int [Flag\\_atomnum](#)
- int [Flag\\_frames](#)
- int [Flag\\_cgnum](#)
- int [Flag\\_nclust](#)
- int [Flag\\_n\\_mappings](#)
- int [Flag\\_MC\\_steps](#)

- int [Flag\\_rotmats\\_period](#)
- int [Flag\\_t\\_zero](#)
- int [Flag\\_distance](#)
- int [Flag\\_criterion](#)
- int [Flag\\_max\\_nclust](#)
- int [Flag\\_min\\_nclust](#)
- int [Flag\\_Ncores](#)
- int [Flag\\_decay\\_time](#)
- int [Flag\\_rsd](#)
- int [Flag\\_stride](#)

### 10.13.1 Member Data Documentation

10.13.1.1 int parameters::atomnum

10.13.1.2 int parameters::frames

10.13.1.3 int parameters::cgnum

10.13.1.4 int parameters::nclust

10.13.1.5 int parameters::n\_mappings

10.13.1.6 int parameters::MC\_steps

10.13.1.7 int parameters::rotmats\_period

10.13.1.8 float parameters::t\_zero

10.13.1.9 float parameters::distance

10.13.1.10 int parameters::criterion

10.13.1.11 int parameters::max\_nclust

10.13.1.12 int parameters::min\_nclust

10.13.1.13 int parameters::Ncores

10.13.1.14 float parameters::decay\_time

10.13.1.15 int parameters::rsd

10.13.1.16 int parameters::stride

10.13.1.17 int parameters::Flag\_atomnum

10.13.1.18 int parameters::Flag\_frames

10.13.1.19 int parameters::Flag\_cgnum

10.13.1.20 int parameters::Flag\_nclust

10.13.1.21 int parameters::Flag\_n\_mappings

10.13.1.22 int parameters::Flag\_MC\_steps

10.13.1.23 int parameters::Flag\_rotmats\_period

10.13.1.24 int parameters::Flag\_t\_zero

10.13.1.25 int parameters::Flag\_distance

10.13.1.26 int parameters::Flag\_criterion

10.13.1.27 int parameters::Flag\_max\_nclust

10.13.1.28 int parameters::Flag\_min\_nclust

10.13.1.29 int parameters::Flag\_Ncores

10.13.1.30 int parameters::Flag\_decay\_time

10.13.1.31 int parameters::Flag\_rsd

10.13.1.32 int parameters::Flag\_stride

The documentation for this struct was generated from the following file:

- [include/io.h](#)

## 10.14 spin\_traj Struct Reference

```
#include <traj.h>
```

### Public Attributes

- int [frames](#)
- int \*\* [traj\\_coords](#)
- double \* [energies](#)
- int [n\\_at](#)
- int [pairs](#)
- int \* [strides](#)
- int [stride](#)
- int [eff\\_frames](#)

### 10.14.1 Member Data Documentation

10.14.1.1 int spin\_traj::frames

number of frames in the trajectory

10.14.1.2 int\*\* spin\_traj::traj\_coords

2D array of xyz coordinates



## 10.14.1.3 double\* spin\_traj::energies

1D array of energies. One value per frame.

## 10.14.1.4 int spin\_traj::n\_at

number of atoms in the atomistic structure

## 10.14.1.5 int spin\_traj::pairs

number of possible pairs of structures

## 10.14.1.6 int\* spin\_traj::strides

vector of configurations to consider (criterion 3)

## 10.14.1.7 int spin\_traj::stride

number of configurations between each pivot for clustering (criterion 3)

## 10.14.1.8 int spin\_traj::eff\_frames

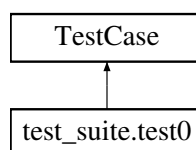
number of effective frames in the trajectory (criterion 3)

The documentation for this struct was generated from the following file:

- include/[traj.h](#)

## 10.15 test\_suite.test0 Class Reference

Inheritance diagram for test\_suite.test0:



### Public Member Functions

- def [test0\\_exist](#)
- def [test0\\_SA](#)
- def [test0\\_log\\_exist](#)
- def [test0\\_head\\_tail](#)

### 10.15.1 Detailed Description

class that checks a three-cores Simulated Annealing run

## 10.15.2 Member Function Documentation

### 10.15.2.1 `def test_suite.test0.test0_exist ( self )`

check existence of .dat files

### 10.15.2.2 `def test_suite.test0.test0_SA ( self )`

open files and check that the optimizations correctly finished

### 10.15.2.3 `def test_suite.test0.test0_log_exist ( self )`

check existence of log file

### 10.15.2.4 `def test_suite.test0.test0_head_tail ( self )`

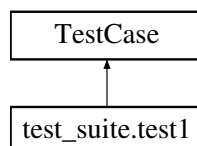
check log's head, tail and size

The documentation for this class was generated from the following file:

- tests/[test\\_suite.py](#)

## 10.16 `test_suite.test1` Class Reference

Inheritance diagram for `test_suite.test1`:



### Public Member Functions

- `def test1_exist`
- `def test1_exist_dat`
- `def test1_count_mapping`
- `def test1_head_tail`

### 10.16.1 Detailed Description

class that checks the correct generation of random mappings and the measurement of their mapping entropy

## 10.16.2 Member Function Documentation

### 10.16.2.1 `def test_suite.test1.test1_exist ( self )`

check existence of log file

#### 10.16.2.2 def test\_suite.test1.test1\_exist\_dat ( self )

check existence of output file

#### 10.16.2.3 def test\_suite.test1.test1\_count\_mapping ( self )

check for consistency between the declared and effective number of random mappings (and the number of calculations)

#### 10.16.2.4 def test\_suite.test1.test1\_head\_tail ( self )

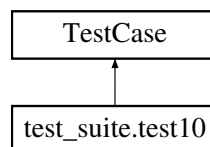
check log's head, tail and size

The documentation for this class was generated from the following file:

- tests/test\_suite.py

## 10.17 test\_suite.test10 Class Reference

Inheritance diagram for test\_suite.test10:



### Public Member Functions

- def test10\_log\_exist
- def test10\_err\_exist
- def test10\_err\_correct

#### 10.17.1 Detailed Description

class that checks the error output if the mapping file is not complete (longer than n\_cg beads)

#### 10.17.2 Member Function Documentation

##### 10.17.2.1 def test\_suite.test10.test10\_log\_exist ( self )

check existence of log file

##### 10.17.2.2 def test\_suite.test10.test10\_err\_exist ( self )

check existence of error file

10.17.2.3 `def test_suite.test10.test10_err_correct ( self )`

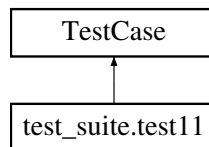
check error file

The documentation for this class was generated from the following file:

- tests/[test\\_suite.py](#)

## 10.18 test\_suite.test11 Class Reference

Inheritance diagram for test\_suite.test11:



### Public Member Functions

- def [test11\\_log\\_exist](#)
- def [test11\\_err\\_exist](#)
- def [test11\\_err\\_correct](#)

#### 10.18.1 Detailed Description

class that checks the error output if the mapping file is not complete (value not between [0 ;n\_at) )

#### 10.18.2 Member Function Documentation

10.18.2.1 `def test_suite.test11.test11_log_exist ( self )`

check existence of log file

10.18.2.2 `def test_suite.test11.test11_err_exist ( self )`

check existence of error file

10.18.2.3 `def test_suite.test11.test11_err_correct ( self )`

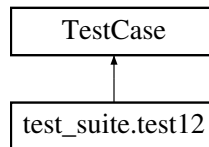
check error file

The documentation for this class was generated from the following file:

- tests/[test\\_suite.py](#)

## 10.19 test\_suite.test12 Class Reference

Inheritance diagram for test\_suite.test12:



### Public Member Functions

- def [test12\\_log\\_exist](#)
- def [test12\\_err\\_exist](#)
- def [test12\\_err\\_correct](#)

### 10.19.1 Detailed Description

class that checks the error output if the mapping file is not complete (each value must be INT)

### 10.19.2 Member Function Documentation

#### 10.19.2.1 def test\_suite.test12.test12\_log\_exist ( self )

check existence of log file

#### 10.19.2.2 def test\_suite.test12.test12\_err\_exist ( self )

check existence of error file

#### 10.19.2.3 def test\_suite.test12.test12\_err\_correct ( self )

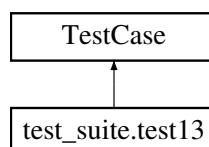
check error file

The documentation for this class was generated from the following file:

- tests/[test\\_suite.py](#)

## 10.20 test\_suite.test13 Class Reference

Inheritance diagram for test\_suite.test13:



## Public Member Functions

- def [test13\\_log\\_exist](#)
- def [test13\\_err\\_exist](#)
- def [test13\\_err\\_correct](#)

### 10.20.1 Detailed Description

class that checks the error output if the mapping file is not complete (it contains duplicates)

### 10.20.2 Member Function Documentation

#### 10.20.2.1 def test\_suite.test13.test13\_log\_exist ( self )

check existence of log file

#### 10.20.2.2 def test\_suite.test13.test13\_err\_exist ( self )

check existence of error file

#### 10.20.2.3 def test\_suite.test13.test13\_err\_correct ( self )

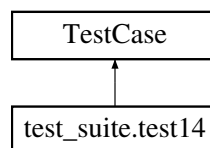
check error file

The documentation for this class was generated from the following file:

- tests/[test\\_suite.py](#)

## 10.21 test\_suite.test14 Class Reference

Inheritance diagram for test\_suite.test14:



## Public Member Functions

- def [test14\\_log\\_exist](#)
- def [test14\\_err\\_exist](#)
- def [test14\\_err\\_correct](#)

### 10.21.1 Detailed Description

class that checks the error output if the parameter file contains, at least, a string VALUE instead of integer

### 10.21.2 Member Function Documentation

#### 10.21.2.1 def test\_suite.test14.test14\_log\_exist ( self )

check existence of log file

#### 10.21.2.2 def test\_suite.test14.test14\_err\_exist ( self )

check existence of error file

#### 10.21.2.3 def test\_suite.test14.test14\_err\_correct ( self )

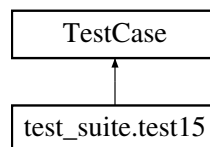
check error file

The documentation for this class was generated from the following file:

- tests/[test\\_suite.py](#)

## 10.22 test\_suite.test15 Class Reference

Inheritance diagram for test\_suite.test15:



### Public Member Functions

- def [test15\\_log\\_exist](#)
- def [test15\\_err\\_exist](#)
- def [test15\\_err\\_correct](#)

### 10.22.1 Detailed Description

class that checks the error output if the energy file contains, at least, an integer value instead of float

### 10.22.2 Member Function Documentation

#### 10.22.2.1 def test\_suite.test15.test15\_log\_exist ( self )

check existence of log file

#### 10.22.2.2 def test\_suite.test15.test15\_err\_exist ( self )

check existence of error file

10.22.2.3 `def test_suite.test15.test15_err_correct ( self )`

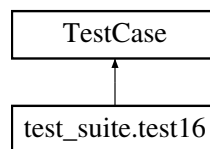
check error file

The documentation for this class was generated from the following file:

- tests/[test\\_suite.py](#)

## 10.23 test\_suite.test16 Class Reference

Inheritance diagram for test\_suite.test16:



### Public Member Functions

- def [test16\\_log\\_exist](#)
- def [test16\\_err\\_exist](#)
- def [test16\\_err\\_correct](#)

### 10.23.1 Detailed Description

class that checks the error output if the energy file contains, at least, one row containing more than one col

### 10.23.2 Member Function Documentation

10.23.2.1 `def test_suite.test16.test16_log_exist ( self )`

check existence of log file

10.23.2.2 `def test_suite.test16.test16_err_exist ( self )`

check existence of error file

10.23.2.3 `def test_suite.test16.test16_err_correct ( self )`

check error file

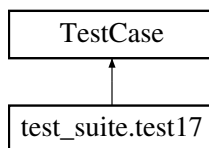
The documentation for this class was generated from the following file:

- tests/[test\\_suite.py](#)



## 10.24 test\_suite.test17 Class Reference

Inheritance diagram for test\_suite.test17:



### Public Member Functions

- def [test17\\_log\\_exist](#)
- def [test17\\_err\\_exist](#)
- def [test17\\_err\\_correct](#)

#### 10.24.1 Detailed Description

class that checks the error output if the trajectory file contains an integer number != n\_atoms when n\_column

#### 10.24.2 Member Function Documentation

##### 10.24.2.1 def test\_suite.test17.test17\_log\_exist ( self )

check existence of log file

##### 10.24.2.2 def test\_suite.test17.test17\_err\_exist ( self )

check existence of error file

##### 10.24.2.3 def test\_suite.test17.test17\_err\_correct ( self )

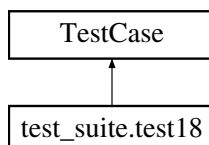
check error file

The documentation for this class was generated from the following file:

- tests/[test\\_suite.py](#)

## 10.25 test\_suite.test18 Class Reference

Inheritance diagram for test\_suite.test18:



## Public Member Functions

- def [test18\\_log\\_exist](#)
- def [test18\\_err\\_exist](#)
- def [test18\\_err\\_correct](#)

### 10.25.1 Detailed Description

class that checks the error output if the trajectory file contains, at least, one letter, instead of float at

### 10.25.2 Member Function Documentation

#### 10.25.2.1 def test\_suite.test18.test18\_log\_exist ( self )

check existence of log file

#### 10.25.2.2 def test\_suite.test18.test18\_err\_exist ( self )

check existence of error file

#### 10.25.2.3 def test\_suite.test18.test18\_err\_correct ( self )

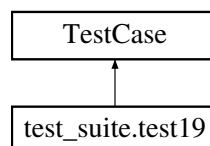
check error file

The documentation for this class was generated from the following file:

- tests/[test\\_suite.py](#)

## 10.26 test\_suite.test19 Class Reference

Inheritance diagram for test\_suite.test19:



## Public Member Functions

- def [test19\\_output\\_exist](#)
- def [test19\\_correct\\_coord\\_number](#)

### 10.26.1 Detailed Description

class that checks the correct calculation of the norm of the mapping for 4AKE

## 10.26.2 Member Function Documentation

### 10.26.2.1 def test\_suite.test19.test19\_output\_exist ( self )

check existence of output file

### 10.26.2.2 def test\_suite.test19.test19\_correct\_coord\_number ( self )

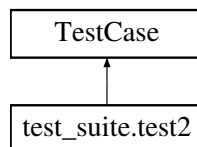
check correct atomistic coordination number

The documentation for this class was generated from the following file:

- tests/test\_suite.py

## 10.27 test\_suite.test2 Class Reference

Inheritance diagram for test\_suite.test2:



### Public Member Functions

- def test2\_log\_exist
- def test2\_output\_exist
- def test2\_head\_tail
- def test2\_correct\_smap

### 10.27.1 Detailed Description

class that checks loading mapping task

## 10.27.2 Member Function Documentation

### 10.27.2.1 def test\_suite.test2.test2\_log\_exist ( self )

check existence of log file

### 10.27.2.2 def test\_suite.test2.test2\_output\_exist ( self )

check existence of output file

### 10.27.2.3 def test\_suite.test2.test2\_head\_tail ( self )

check parameter file is correct and that file length is not excessive

10.27.2.4 `def test_suite.test2.test2_correct_smap ( self )`

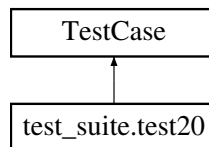
check that the mapping entropy is correct

The documentation for this class was generated from the following file:

- tests/[test\\_suite.py](#)

## 10.28 test\_suite.test20 Class Reference

Inheritance diagram for test\_suite.test20:



### Public Member Functions

- def [test20\\_output\\_exist](#)
- def [test20\\_cosines](#)
- def [test20\\_distances](#)

#### 10.28.1 Detailed Description

class that checks calculation of cosines and distances between two identical mappings

#### 10.28.2 Member Function Documentation

10.28.2.1 `def test_suite.test20.test20_output_exist ( self )`

check existence of output file

10.28.2.2 `def test_suite.test20.test20_cosines ( self )`

check that all cosines are calculated and equal to one

10.28.2.3 `def test_suite.test20.test20_distances ( self )`

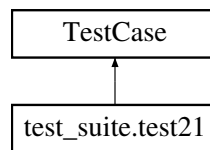
check that all distances are calculated and equal to zero

The documentation for this class was generated from the following file:

- tests/[test\\_suite.py](#)

## 10.29 test\_suite.test21 Class Reference

Inheritance diagram for test\_suite.test21:



### Public Member Functions

- def [test21\\_output\\_exist](#)
- def [test21\\_consistent\\_norms](#)

### 10.29.1 Detailed Description

class that checks the correct calculation of the norm of the Calpha mapping for 6D93

### 10.29.2 Member Function Documentation

#### 10.29.2.1 def test\_suite.test21.test21\_output\_exist ( self )

check existence of output file

#### 10.29.2.2 def test\_suite.test21.test21\_consistent\_norms ( self )

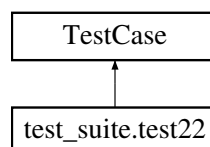
check consistency with calculated norms

The documentation for this class was generated from the following file:

- tests/[test\\_suite.py](#)

## 10.30 test\_suite.test22 Class Reference

Inheritance diagram for test\_suite.test22:



### Public Member Functions

- def [test22\\_log\\_exist](#)
- def [test22\\_output\\_exist](#)
- def [test22\\_distmat\\_exist](#)
- def [test22\\_distmat\\_shape](#)

### 10.30.1 Detailed Description

class that checks the correct calculation of distance matrix between mappings

### 10.30.2 Member Function Documentation

#### 10.30.2.1 def test\_suite.test22.test22\_log\_exist ( self )

check existence of log file

#### 10.30.2.2 def test\_suite.test22.test22\_output\_exist ( self )

check existence of output file

#### 10.30.2.3 def test\_suite.test22.test22\_distmat\_exist ( self )

check existence of distance matrix file

#### 10.30.2.4 def test\_suite.test22.test22\_distmat\_shape ( self )

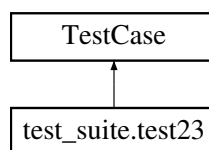
check shape of distance matrix

The documentation for this class was generated from the following file:

- tests/[test\\_suite.py](#)

## 10.31 test\_suite.test23 Class Reference

Inheritance diagram for test\_suite.test23:



### Public Member Functions

- def [test23\\_output\\_exist](#)
- def [test23\\_correct\\_smap](#)
- def [test23\\_check\\_pairs](#)

### 10.31.1 Detailed Description

class that checks the correct functioning of criterion 3 (fast clustering)

### 10.31.2 Member Function Documentation

#### 10.31.2.1 def test\_suite.test23.test23\_output\_exist ( self )

check existence of output file

#### 10.31.2.2 def test\_suite.test23.test23\_correct\_smap ( self )

check that the mapping entropy is correct

#### 10.31.2.3 def test\_suite.test23.test23\_check\_pairs ( self )

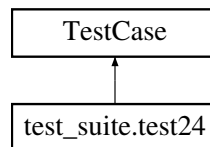
check existence of log file and that the number of pairs matches its expected value

The documentation for this class was generated from the following file:

- tests/[test\\_suite.py](#)

## 10.32 test\_suite.test24 Class Reference

Inheritance diagram for test\_suite.test24:



### Public Member Functions

- def [test24\\_output\\_exist](#)
- def [test24\\_check\\_probabilities](#)
- def [test24\\_use\\_probabilities](#)

### 10.32.1 Detailed Description

class that checks the correct functioning of task optimize\_kl

### 10.32.2 Member Function Documentation

#### 10.32.2.1 def test\_suite.test24.test24\_output\_exist ( self )

check existence of output file

#### 10.32.2.2 def test\_suite.test24.test24\_check\_probabilities ( self )

check that the program is checking probabilities

### 10.32.2.3 `def test_suite.test24.test24_use_probabilities ( self )`

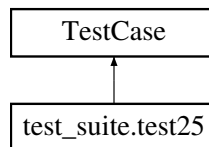
check that the program is using probabilities

The documentation for this class was generated from the following file:

- tests/[test\\_suite.py](#)

## 10.33 `test_suite.test25` Class Reference

Inheritance diagram for `test_suite.test25`:



### Public Member Functions

- `def test25_output_exist`
- `def test25_check_probabilities`
- `def test25_use_probabilities_correct_smap`

### 10.33.1 Detailed Description

class that checks the correct functioning of task `measure_kl`

### 10.33.2 Member Function Documentation

#### 10.33.2.1 `def test_suite.test25.test25_output_exist ( self )`

check existence of output file

#### 10.33.2.2 `def test_suite.test25.test25_check_probabilities ( self )`

check that the program is checking probabilities

#### 10.33.2.3 `def test_suite.test25.test25_use_probabilities_correct_smap ( self )`

check that the program is using probabilities and that the mapping entropy is correct

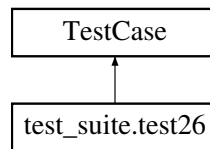
The documentation for this class was generated from the following file:

- tests/[test\\_suite.py](#)



## 10.34 test\_suite.test26 Class Reference

Inheritance diagram for test\_suite.test26:



### Public Member Functions

- def [test26\\_output\\_exist](#)
- def [test26\\_use\\_probabilities](#)

#### 10.34.1 Detailed Description

class that checks the correct functioning of task random\_kl

#### 10.34.2 Member Function Documentation

10.34.2.1 def test\_suite.test26.test26\_output\_exist ( self )

check existence of output file

10.34.2.2 def test\_suite.test26.test26\_use\_probabilities ( self )

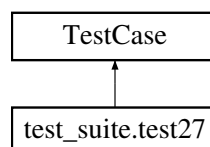
check that the program is using probabilities

The documentation for this class was generated from the following file:

- tests/[test\\_suite.py](#)

## 10.35 test\_suite.test27 Class Reference

Inheritance diagram for test\_suite.test27:



### Public Member Functions

- def [test27\\_output\\_exist](#)
- def [test27\\_smeps](#)

### 10.35.1 Detailed Description

class that checks the correct functioning of task `optimize_spins`

### 10.35.2 Member Function Documentation

#### 10.35.2.1 `def test_suite.test27.test27_output_exist ( self )`

check existence of output files

#### 10.35.2.2 `def test_suite.test27.test27_smags ( self )`

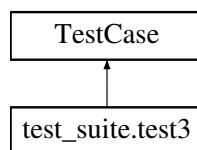
check that all 3 possible values of `smag` are calculated

The documentation for this class was generated from the following file:

- [tests/test\\_suite.py](#)

## 10.36 test\_suite.test3 Class Reference

Inheritance diagram for `test_suite.test3`:



### Public Member Functions

- `def test3_log_exist`
- `def test3_head_tail`
- `def test3_deltas`

### 10.36.1 Detailed Description

class that checks the estimation of `T_start` inside optimisation

### 10.36.2 Member Function Documentation

#### 10.36.2.1 `def test_suite.test3.test3_log_exist ( self )`

check existence of log file

#### 10.36.2.2 `def test_suite.test3.test3_head_tail ( self )`

check log's head and tail

10.36.2.3 `def test_suite.test3.test3_deltas ( self )`

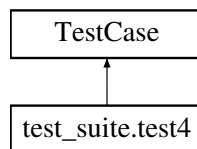
check delta dat files

The documentation for this class was generated from the following file:

- tests/[test\\_suite.py](#)

## 10.37 test\_suite.test4 Class Reference

Inheritance diagram for test\_suite.test4:



### Public Member Functions

- def [test4\\_log\\_exist](#)
- def [test4\\_err\\_exist](#)
- def [test4\\_err\\_correct](#)

### 10.37.1 Detailed Description

class that checks an invalid task ID (ex. optimizer)

### 10.37.2 Member Function Documentation

10.37.2.1 `def test_suite.test4.test4_log_exist ( self )`

check existence of log file

10.37.2.2 `def test_suite.test4.test4_err_exist ( self )`

check existence of error.dat

10.37.2.3 `def test_suite.test4.test4_err_correct ( self )`

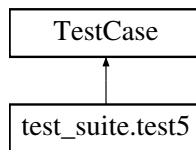
check error file

The documentation for this class was generated from the following file:

- tests/[test\\_suite.py](#)

## 10.38 test\_suite.test5 Class Reference

Inheritance diagram for test\_suite.test5:



### Public Member Functions

- def [test5\\_log\\_exist](#)
- def [test5\\_count\\_alignments](#)

### 10.38.1 Detailed Description

class that checks the expected number of alignments

### 10.38.2 Member Function Documentation

#### 10.38.2.1 def test\_suite.test5.test5\_log\_exist ( self )

check existence of log file

#### 10.38.2.2 def test\_suite.test5.test5\_count\_alignments ( self )

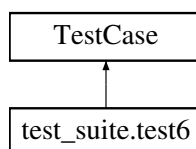
count the effective (from dat file) and expected (from log/parameter file) number of alignments. See if they m

The documentation for this class was generated from the following file:

- tests/[test\\_suite.py](#)

## 10.39 test\_suite.test6 Class Reference

Inheritance diagram for test\_suite.test6:



### Public Member Functions

- def [test6\\_log\\_exist](#)
- def [test6\\_err\\_exist](#)
- def [test6\\_err\\_correct](#)

### 10.39.1 Detailed Description

class that checks the error output if the trajectory does not respect the declared number of frames

### 10.39.2 Member Function Documentation

#### 10.39.2.1 def test\_suite.test6.test6\_log\_exist ( self )

check existence of log file

#### 10.39.2.2 def test\_suite.test6.test6\_err\_exist ( self )

check existence of error.dat

#### 10.39.2.3 def test\_suite.test6.test6\_err\_correct ( self )

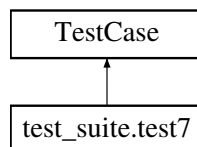
check error file

The documentation for this class was generated from the following file:

- tests/[test\\_suite.py](#)

## 10.40 test\_suite.test7 Class Reference

Inheritance diagram for test\_suite.test7:



### Public Member Functions

- def [test7\\_log\\_exist](#)
- def [test7\\_err\\_exist](#)
- def [test7\\_err\\_correct](#)

### 10.40.1 Detailed Description

class that checks the error output if the trajectory is cut at the last frame

### 10.40.2 Member Function Documentation

#### 10.40.2.1 def test\_suite.test7.test7\_log\_exist ( self )

check existence of log file

#### 10.40.2.2 `def test_suite.test7.test7_err_exist ( self )`

check existence of error.dat

#### 10.40.2.3 `def test_suite.test7.test7_err_correct ( self )`

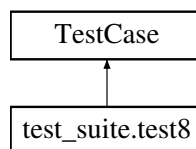
check error file

The documentation for this class was generated from the following file:

- tests/[test\\_suite.py](#)

## 10.41 `test_suite.test8` Class Reference

Inheritance diagram for `test_suite.test8`:



### Public Member Functions

- `def test8\_log\_exist`
- `def test8\_err\_exist`
- `def test8\_err\_correct`

#### 10.41.1 Detailed Description

class that checks the error output if the energy file is not complete

#### 10.41.2 Member Function Documentation

##### 10.41.2.1 `def test_suite.test8.test8_log_exist ( self )`

check existence of log file

##### 10.41.2.2 `def test_suite.test8.test8_err_exist ( self )`

check existence of error.dat

##### 10.41.2.3 `def test_suite.test8.test8_err_correct ( self )`

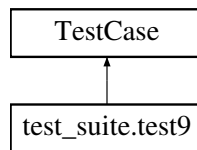
check error file

The documentation for this class was generated from the following file:

- tests/[test\\_suite.py](#)

## 10.42 test\_suite.test9 Class Reference

Inheritance diagram for test\_suite.test9:



### Public Member Functions

- def [test9\\_log\\_exist](#)
- def [test9\\_err\\_exist](#)
- def [test9\\_err\\_correct](#)

### 10.42.1 Detailed Description

class that checks the error output if the mapping file is not complete (shorter than n\_cg beads)

### 10.42.2 Member Function Documentation

#### 10.42.2.1 def test\_suite.test9.test9\_log\_exist ( self )

check existence of log file

#### 10.42.2.2 def test\_suite.test9.test9\_err\_exist ( self )

check existence of error file

#### 10.42.2.3 def test\_suite.test9.test9\_err\_correct ( self )

check error file

The documentation for this class was generated from the following file:

- [tests/test\\_suite.py](#)

## 10.43 traj Class Reference

structure that defines a MD trajectory

```
#include <traj.h>
```

### Public Attributes

- int [frames](#)
- double \*\* [traj\\_coords](#)
- double \* [energies](#)

- int [n\\_at](#)
- int [pairs](#)
- int \* [strides](#)
- int [stride](#)
- int [eff\\_frames](#)

### 10.43.1 Detailed Description

structure that defines a MD trajectory

### 10.43.2 Member Data Documentation

#### 10.43.2.1 int traj::frames

number of frames in the trajectory

#### 10.43.2.2 double\*\* traj::traj\_coords

2D array of xyz coordinates

#### 10.43.2.3 double\* traj::energies

1D array of energies. One value per frame.

#### 10.43.2.4 int traj::n\_at

number of atoms in the atomistic structure

#### 10.43.2.5 int traj::pairs

number of possible pairs of structures

#### 10.43.2.6 int\* traj::strides

vector of configurations to consider (criterion 3)

#### 10.43.2.7 int traj::stride

number of configurations between each pivot for clustering (criterion 3)

#### 10.43.2.8 int traj::eff\_frames

number of effective frames in the trajectory (criterion 3)

The documentation for this class was generated from the following file:

- include/[traj.h](#)



# Chapter 11

## File Documentation

### 11.1 excogito.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <my_malloc.h>
#include <time.h>
#include <omp.h>
#include <sys/types.h>
#include <unistd.h>
#include <limits.h>
#include <stdbool.h>
#include <sampling.h>
#include <io.h>
#include <geometry.h>
#include <mapping.h>
#include <observables.h>
#include <alignment.h>
#include <ini.h>
#include <traj.h>
#include <optimize.h>
#include <random_mapping.h>
#include <measure.h>
#include <norm.h>
#include <cosine.h>
#include <distance.h>
#include <argp.h>
#include <measure_kl.h>
#include <optimize_kl.h>
#include <random_mapping_kl.h>
#include <optimize_spins.h>
```

#### Functions

- int [main](#) (int argc, char \*argv[])

#### 11.1.1 Function Documentation

11.1.1.1 `int main ( int argc, char * argv[] )`

main file of the program

## 11.2 include/alignment.h File Reference

```
#include <mapping.h>
#include <traj.h>
```

### Classes

- class [alignments](#)  
*structure that defines the current alignments stored in memory*

### Typedefs

- typedef struct [alignments](#) [alignments](#)

### Functions

- void [free\\_new\\_alignment](#) ([alignments](#) \*new\_align)
- void [free\\_alignment](#) ([alignments](#) \*align)
- void [align\\_two\\_frames](#) (double \*frame\_ref, double \*frame\_middle, int ref\_id, int middle\_id, [cg\\_mapping](#) \*mapping, [alignments](#) \*align)
- double [optimal\\_alignment](#) (double \*\*x, double \*\*y, int mapping\_length, double u[][3])
- void [correct\\_rmsd](#) ([alignments](#) \*new\_align, [traj](#) \*Trajectory, [alignments](#) \*prev\_align, int cignum, int removed, int added)
- void [cycle\\_alignment](#) ([traj](#) \*Trajectory, [alignments](#) \*align, [cg\\_mapping](#) \*mapping)
- void [cycle\\_alignment\\_fastclust](#) ([traj](#) \*Trajectory, [alignments](#) \*align, [cg\\_mapping](#) \*mapping)
- void [correct\\_rmsd\\_fastclust](#) ([alignments](#) \*new\_align, [traj](#) \*Trajectory, [alignments](#) \*prev\_align, int cignum, int removed, int added)
- void [cycle\\_alignment\\_stride](#) ([traj](#) \*Trajectory, [alignments](#) \*align, [cg\\_mapping](#) \*mapping)
- void [align\\_traj\\_to\\_reference](#) ([traj](#) \*Trajectory, int ref\_id)

#### 11.2.1 Typedef Documentation

11.2.1.1 typedef struct [alignments](#) [alignments](#)

#### 11.2.2 Function Documentation

11.2.2.1 void [free\\_new\\_alignment](#) ( [alignments](#) \* *new\_align* )

routine that frees an alignments object used in criterion 1

#### Parameters

*new\_align*: alignments object

**11.2.2.2 void free\_alignment ( alignments \* align )**

routine that frees an alignments object

**Parameters**

align: alignments object

**11.2.2.3 void align\_two\_frames ( double \* frame\_ref, double \* frame\_middle, int ref\_id, int middle\_id, cg\_mapping \* mapping, alignments \* align )**

routine that aligns a pair of frames in a trajectory, calling optimal\_alignment

**Parameters**

frame\_ref: reference frame

frame\_middle: frame in between two pivot clusters

ref\_id: id (index) of frame\_ref in the trajectory

middle\_id: id (index) of frame\_middle in the trajectory

mapping: [cg\\_mapping](#) object

align: alignments object

**11.2.2.4 double optimal\_alignment ( double \*\* x, double \*\* y, int mapping\_length, double u[3][3] )**

routine that computes the Kabsch alignment and the rmsd between two configurations

**Parameters**

x, y: CG structures

cgnum: length of CG mapping

u: rotation matrix

**11.2.2.5 void correct\_rmsd ( alignments \* new\_align, traj \* Trajectory, alignments \* prev\_align, int cgnum, int removed, int added )**

routine that computes the rmsd matrix without aligning frames over frames

**Parameters**

new\_align: trial alignments object

Trajectory: traj object

align: alignments object

cgnum: number of CG sites (useful to normalize)

removed: index of removed atom

added: index of added atom

**11.2.2.6 void cycle\_alignment ( traj \* *Trajectory*, alignments \* *align*, cg\_mapping \* *mapping* )**

routine that cycles over all pairs of frames in a trajectory, calling `optimal_alignment`

**Parameters**

`Trajectory` : traj object

`align` : alignments object

`mapping` : [cg\\_mapping](#) object

**11.2.2.7 void cycle\_alignment\_fastclust ( traj \* *Trajectory*, alignments \* *align*, cg\_mapping \* *mapping* )**

routine that computes the alignments if clustering must be fast

**Parameters**

`Trajectory` : traj object

`align` : alignments object

`mapping` : [cg\\_mapping](#) object

**11.2.2.8 void correct\_rmsd\_fastclust ( alignments \* *new\_align*, traj \* *Trajectory*, alignments \* *prev\_align*, int *cgnum*, int *removed*, int *added* )**

routine that computes the rmsd matrix without aligning frames over frames

**Parameters**

`new_rmsd_mat` : new condensed pairwise RMSD matrix

`Trajectory` : traj object

`align` : alignments object

`cgnum` : number of CG sites (useful to normalize)

`removed` : index of removed atom

`added` : index of added atom

**11.2.2.9 void cycle\_alignment\_stride ( traj \* *Trajectory*, alignments \* *align*, cg\_mapping \* *mapping* )**

routine that cycles over all pairs of frames in a trajectory, calling `optimal_alignment`

**Parameters**

`Trajectory` : traj object

`align` : alignments object

`mapping` : [cg\\_mapping](#) object

**11.2.2.10 void align\_traj\_to\_reference ( traj \* *Trajectory*, int *ref\_id* )**

routine that aligns the trajectory to a reference frame

#### Parameters

Trajectory : traj object

ref\_id : reference frame

## 11.3 include/cosine.h File Reference

```
#include <stdio.h>
```

```
#include <io.h>
```

#### Functions

- void `cosine` (`arguments` \*arguments, `parameters` \*cc)

#### 11.3.1 Function Documentation

##### 11.3.1.1 void cosine ( arguments \* arguments, parameters \* cc )

subprogram to calculate pairwise distance and cosine between a pair of mappings (provided by the user) throughout a trajectory

#### Parameters

arguments : arguments object (command line arguments)

parameters : parameters object

## 11.4 include/distance.h File Reference

```
#include <stdio.h>
```

```
#include <io.h>
```

#### Functions

- void `distance` (`arguments` \*arguments, `parameters` \*cc)

#### 11.4.1 Function Documentation

##### 11.4.1.1 void distance ( arguments \* arguments, parameters \* cc )

subprogram to calculate the distance matrix between a data set of mappings (provided by the user) over a single conformation

#### Parameters

arguments : arguments object (command line arguments)

parameters : parameters object

## 11.5 include/geometry.h File Reference

### Functions

- void [vecprod\\_d](#) (double \*a, double \*b, double \*c)
- double [scal\\_d](#) (double \*a, double \*b, int dim)
- double [coseno](#) (double \*vec1, double \*vec2, int dim)
- double [norm\\_d](#) (double \*a, int dim)
- void [normalize\\_d](#) (double \*a, int dim)
- double [dist\\_d](#) (double \*a, double \*b, int dim)
- double [det](#) (double a1, double a2, double a3, double b1, double b2, double b3, double c1, double c2, double c3)
- void [vec\\_sum\\_d](#) (double \*a, double \*b, double \*c, double d, int dim)
- void [print\\_vec\\_d](#) (double \*a, int dim)
- void [zero\\_vec\\_d](#) (double \*a, int dim)
- void [zero\\_vec\\_i](#) (int \*a, int dim)
- void [zero\\_matrix\\_d](#) (double \*\*a, int dim1, int dim2)
- void [myjacobi](#) (double a[][3], int n, double \*d, double v[][3], int \*nrot)

### 11.5.1 Function Documentation

11.5.1.1 void [vecprod\\_d](#) ( double \* a, double \* b, double \* c )

11.5.1.2 double [scal\\_d](#) ( double \* a, double \* b, int dim )

11.5.1.3 double [coseno](#) ( double \* vec1, double \* vec2, int dim )

11.5.1.4 double [norm\\_d](#) ( double \* a, int dim )

11.5.1.5 void [normalize\\_d](#) ( double \* a, int dim )

11.5.1.6 double [dist\\_d](#) ( double \* a, double \* b, int dim )

11.5.1.7 double [det](#) ( double a1, double a2, double a3, double b1, double b2, double b3, double c1, double c2, double c3 )

11.5.1.8 void [vec\\_sum\\_d](#) ( double \* a, double \* b, double \* c, double d, int dim )

11.5.1.9 void [print\\_vec\\_d](#) ( double \* a, int dim )

11.5.1.10 void [zero\\_vec\\_d](#) ( double \* a, int dim )

11.5.1.11 void [zero\\_vec\\_i](#) ( int \* a, int dim )

11.5.1.12 void [zero\\_matrix\\_d](#) ( double \*\* a, int dim1, int dim2 )

11.5.1.13 void [myjacobi](#) ( double a[][3], int n, double \* d, double v[][3], int \* nrot )

## 11.6 include/hierarchical\_clustering.h File Reference

### Classes

- class [clust\\_params](#)  
*structure that defines the parameters for hierarchical clustering*

## Typedefs

- typedef struct [clust\\_params](#) [clust\\_params](#)

## Functions

- void [mergesort\\_merge](#) (double \*\*arr, int l, int m, int r, int dim, int dims)
- void [my\\_mergesort](#) (double \*\*arr, int l, int r, int dim, int dims)
- int [condensed\\_index](#) (int frames, int i, int j)
- double [new\\_dist](#) (double d\_xi, double d\_yi, double d\_xy, int size\_x, int size\_y, int size\_i)
- int [is\\_visited](#) (unsigned char \*bitset, int i)
- void [set\\_visited](#) (unsigned char \*bitset, int i)
- void [get\\_max\\_dist\\_for\\_each\\_cluster](#) (double \*\*Z, double \*MD, int frames)
- void [cluster\\_monocrit](#) (double \*\*Z, double \*MC, int \*T, double cutoff, int frames)
- void [cluster\\_maxclust\\_monocrit](#) (double \*\*Z, double \*MC, int \*T, int n, int max\_nc)
- void [cluster\\_maxclust\\_dist](#) (double \*\*Z, int \*T, int frames, int Nclust)
- void [cluster\\_dist](#) (double \*\*Z, int \*T, double cutoff, int frames)
- int [find](#) (int x, int \*self\_parent)
- int [merge](#) (int \*self\_parent, int \*self\_size, int next\_label, int x, int y)
- void [label](#) (double \*\*Z, int frames)
- void [hierarchical\\_clustering](#) (double \*rmsd\_mat, int n, int couples, int \*size, double \*\*Z)
- void [compute\\_clusters\\_list](#) (int \*clusters, int \*cluster\_list, int \*cluster\_list\_idx, int frames, int Nclust)

### 11.6.1 Typedef Documentation

#### 11.6.1.1 typedef struct [clust\\_params](#) [clust\\_params](#)

### 11.6.2 Function Documentation

#### 11.6.2.1 void [mergesort\\_merge](#) ( double \*\* *arr*, int *l*, int *m*, int *r*, int *dim*, int *dims* )

#### 11.6.2.2 void [my\\_mergesort](#) ( double \*\* *arr*, int *l*, int *r*, int *dim*, int *dims* )

#### 11.6.2.3 int [condensed\\_index](#) ( int *frames*, int *i*, int *j* )

*frames* : number of observations

*i* : node

*j* : node

#### 11.6.2.4 double [new\\_dist](#) ( double *d\_xi*, double *d\_yi*, double *d\_xy*, int *size\_x*, int *size\_y*, int *size\_i* )

#### 11.6.2.5 int [is\\_visited](#) ( unsigned char \* *bitset*, int *i* )

routine that checks if node *i* was visited.

### Parameters

*bitset* : char defining visits

*i* : node

### 11.6.2.6 void set\_visited ( unsigned char \* *bitset*, int *i* )

routine that marks node *i* as visited.

#### Parameters

*bitset* : char defining visits

*i* : node

### 11.6.2.7 void get\_max\_dist\_for\_each\_cluster ( double \*\* *Z*, double \* *MD*, int *frames* )

Get the maximum inconsistency coefficient for each non-singleton cluster.

#### Parameters

*Z* : linkage matrix.

*MD* : array to store the result.

*frames* : number of observations.

### 11.6.2.8 void cluster\_monocrit ( double \*\* *Z*, double \* *MC*, int \* *T*, double *cutoff*, int *frames* )

Form flat clusters by monocrit criterion.

#### Parameters

*Z* : linkage matrix.

*MC* : monotonic criterion array.

*T* : array to store the cluster numbers. The *i*'th observation belongs to cluster *T*[*i*].

*cutoff* : Clusters are formed when the MC values are less than or equal to *cutoff*.

*frames* : number of observations

### 11.6.2.9 void cluster\_maxclust\_monocrit ( double \*\* *Z*, double \* *MC*, int \* *T*, int *n*, int *max\_nc* )

Form flat clusters by maxclust\_monocrit criterion.

#### Parameters

*Z* : linkage matrix

*MC* : monotonic criterion array

*T* : array to store the cluster numbers. The *i*'th observation belongs to cluster *T*[*i*]

*frames* : number of observations

*max\_nc* : The maximum number of clusters

### 11.6.2.10 void cluster\_maxclust\_dist ( double \*\* *Z*, int \* *T*, int *frames*, int *Nclust* )

routine that converts the dendrogram into nclust clusters



**Parameters**

*Z* : linkage matrix.

*T* : array to store the cluster numbers. The *i*'th observation belongs to cluster *T*[*i*].

*frames* : number of observations.

*nclust* : number of desired clusters.

**11.6.2.11** void cluster\_dist ( double \*\* *Z*, int \* *T*, double *cutoff*, int *frames* )

**11.6.2.12** int find ( int *x*, int \* *self\_parent* )

**11.6.2.13** int merge ( int \* *self\_parent*, int \* *self\_size*, int *next\_label*, int *x*, int *y* )

**11.6.2.14** void label ( double \*\* *Z*, int *frames* )

routine that correctly labels clusters in the unsorted dendrogram

**Parameters**

*Z* : linkage matrix

*frames* : number of observations

**11.6.2.15** void hierarchical\_clustering ( double \* *rmsd\_mat*, int *n*, int *couples*, int \* *size*, double \*\* *Z* )

overall routine for hierarchical clustering

**Parameters**

*rmsd\_mat* : condensed pairwise RMSD matrix

*frames* : number of observations

*pairs* : possible pairs of structures

*size* : size of the clusters (it is *nclust* long)

*Z* : linkage matrix

**11.6.2.16** void compute\_clusters\_list ( int \* *clusters*, int \* *cluster\_list*, int \* *cluster\_list\_idx*, int *frames*, int *Nclust* )

routine that computes the list of cluster IDs

**Parameters**

*clusters* : list of labels (one for each frame)

*cluster\_list* : ordered list of labels

*cluster\_list\_idx* : is an index vector that stores the sum of populations up to each index

*frames* : number of observations

*nclust* : number of clusters

< array that stores the population of each clusters

## 11.7 include/ini.h File Reference

```
#include <stdio.h>
```

### Macros

- `#define INI_HANDLER_LINENO 0`
- `#define INI_ALLOW_MULTILINE 1`
- `#define INI_ALLOW_BOM 1`
- `#define INI_START_COMMENT_PREFIXES ";"`
- `#define INI_ALLOW_INLINE_COMMENTS 1`
- `#define INI_INLINE_COMMENT_PREFIXES ";`
- `#define INI_USE_STACK 1`
- `#define INI_MAX_LINE 200`
- `#define INI_ALLOW_REALLOC 0`
- `#define INI_INITIAL_ALLOC 200`
- `#define INI_STOP_ON_FIRST_ERROR 0`
- `#define INI_CALL_HANDLER_ON_NEW_SECTION 0`
- `#define INI_ALLOW_NO_VALUE 0`
- `#define INI_CUSTOM_ALLOCATOR 0`

### Typedefs

- `typedef int(* ini\_handler)(void *user, const char *section, const char *name, const char *value)`
- `typedef char *(* ini\_reader)(char *str, int num, void *stream)`

### Functions

- `int ini\_parse(const char *filename, ini\_handler handler, void *user)`
- `int ini\_parse\_file(FILE *file, ini\_handler handler, void *user)`
- `int ini\_parse\_stream(ini\_reader reader, void *stream, ini\_handler handler, void *user)`
- `int ini\_parse\_string(const char *string, ini\_handler handler, void *user)`

#### 11.7.1 Macro Definition Documentation

11.7.1.1 `#define INI_HANDLER_LINENO 0`

11.7.1.2 `#define INI_ALLOW_MULTILINE 1`

11.7.1.3 `#define INI_ALLOW_BOM 1`

11.7.1.4 `#define INI_START_COMMENT_PREFIXES ";"`

11.7.1.5 `#define INI_ALLOW_INLINE_COMMENTS 1`

11.7.1.6 `#define INI_INLINE_COMMENT_PREFIXES ";`

11.7.1.7 `#define INI_USE_STACK 1`

11.7.1.8 `#define INI_MAX_LINE 200`

11.7.1.9 `#define INI_ALLOW_REALLOC 0`

11.7.1.10 `#define INI_INITIAL_ALLOC 200`

11.7.1.11 `#define INI_STOP_ON_FIRST_ERROR 0`

11.7.1.12 `#define INI_CALL_HANDLER_ON_NEW_SECTION 0`

11.7.1.13 `#define INI_ALLOW_NO_VALUE 0`

11.7.1.14 `#define INI_CUSTOM_ALLOCATOR 0`

## 11.7.2 Typedef Documentation

11.7.2.1 `typedef int(* ini_handler)(void *user, const char *section, const char *name, const char *value)`

11.7.2.2 `typedef char*( * ini_reader)(char *str, int num, void *stream)`

## 11.7.3 Function Documentation

11.7.3.1 `int ini_parse ( const char * filename, ini_handler handler, void * user )`

11.7.3.2 `int ini_parse_file ( FILE * file, ini_handler handler, void * user )`

11.7.3.3 `int ini_parse_stream ( ini_reader reader, void * stream, ini_handler handler, void * user )`

11.7.3.4 `int ini_parse_string ( const char * string, ini_handler handler, void * user )`

## 11.8 include/io.h File Reference

```
#include <stdio.h>
#include <string.h>
#include <argp.h>
#include <ini.h>
#include <stdlib.h>
```

## Classes

- struct [arguments](#)
- struct [parameters](#)

## Functions

- FILE \* [open\\_file\\_w](#) (char \*filename)
- FILE \* [open\\_file\\_r](#) (char \*filename)
- FILE \* [open\\_file\\_a](#) (char \*filename)
- void [close\\_file](#) (FILE \*fp)
- void [print\\_help\\_main](#) (char \*argv[])
- void [print\\_help](#) (char \*argv[])
- static error\_t [parse\\_opt](#) (int key, char \*arg, struct argp\_state \*state)
- int [handler](#) (void \*config, const char \*section, const char \*name, const char \*value)
- [parameters pp\\_config](#) ([parameters](#) config)
- void [print\\_usage\\_main](#) (char \*argv[])

- void [check\\_files](#) (char \*\*pars, char \*\*pars\_names, int n\_pars, char \*argv[])
- void [check\\_empty\\_file](#) (FILE \*f, char \*filename)
- int [n\\_rows](#) (FILE \*f)
- void [check\\_empty\\_rows](#) (char \*str)
- void [check\\_int\\_string](#) (const char \*str, int row, char \*fname)
- void [check\\_int\\_string\\_iniFile](#) (const char \*str, char \*fname, char \*name)
- void [check\\_argv\\_errors](#) (char \*argv[], int argc)
- void [check\\_float\\_string](#) (char \*str, int row, char \*fname)
- void [check\\_float\\_string\\_iniFile](#) (const char \*str, char \*fname, char \*name)
- int [columns](#) (char \*string)
- void [mandatory\\_files\\_present](#) (arguments \*arguments, char \*argv[])
- void [read\\_ParameterFile](#) (arguments \*arguments, parameters \*cc)
- void [check\\_optional\\_parameters](#) (parameters \*cc)
- void [check\\_parameters](#) (int \*pars, char \*\*pars\_names, int n\_pars)
- void [read\\_mapping\\_matrix](#) (char \*mappings\_filename, FILE \*f\_out\_l, struct [cg\\_mapping](#) \*mapping\_matrix[], int nmaps)

## Variables

- static char [doc\\_main](#) []
- static char [args\\_doc\\_main](#) [] = "random\noptimize\nmeasure\nnorm\ncosine\ndistance"
- static struct argp\_option [options\\_main](#) []
- static struct argp [argp](#) = { [options\\_main](#), [parse\\_opt](#), [args\\_doc\\_main](#), [doc\\_main](#) }

## 11.8.1 Function Documentation

### 11.8.1.1 FILE\* open\_file\_w ( char \* filename )

routine that opens a file in write mode

### 11.8.1.2 FILE\* open\_file\_r ( char \* filename )

routine that opens a file in read mode

### 11.8.1.3 FILE\* open\_file\_a ( char \* filename )

routine that opens a file in append mode

### 11.8.1.4 void close\_file ( FILE \* fp )

routine that closes a file

### 11.8.1.5 void print\_help\_main ( char \* argv[] )

routine that prints detailed information about the program

## Parameter

[argv\[\]](#) : array of command line arguments

**11.8.1.6 void print\_help ( char \* argv[] )**

routine that prints some help

**Parameter**

argv[] : array of command line arguments

**11.8.1.7 static error\_t parse\_opt ( int key, char \* arg, struct argp\_state \* state ) [static]****11.8.1.8 int handler ( void \* config, const char \* section, const char \* name, const char \* value )****11.8.1.9 parameters pp\_config ( parameters config )****11.8.1.10 void print\_usage\_main ( char \* argv[] )**

routine that prints the usage of the program

**Parameter**

argv[] : array of command line arguments

**11.8.1.11 void check\_files ( char \*\* pars, char \*\* pars\_names, int n\_pars, char \* argv[] )**

routine that checks if all command line arguments are correctly provided

**Parameter**

pars : parameters

pars\_names : names of parameters

n\_pars : number of parameters

argv[] : array of command line arguments

**11.8.1.12 void check\_empty\_file ( FILE \* f, char \* filename )**

routine that checks if the file required exists. If it is the case, check if it is empty or not.

**Parameters**

f : FILE structure that represents the file opened.

filename : filename read

**11.8.1.13 int n\_rows ( FILE \* f )**

routine that returns the number of rows in a file. It counts correctly this number even if the last row does not present

**Parameter**

f : FILE structure that represents the file opened.

**11.8.1.14 void check\_empty\_rows ( char \* *str* )**

routine that checks if a generic line is empty or not

**Parameter**

*str* : string token in account

**11.8.1.15 void check\_int\_string ( const char \* *str*, int *row*, char \* *fname* )**

routine that checks if the string token in account reading a generic FILE is an INTEGER number

**Parameters**

*str* : string token in account

*row* : number of row where the string is found.

*fname* : filename read

**11.8.1.16 void check\_int\_string\_iniFile ( const char \* *str*, char \* *fname*, char \* *name* )**

routine that checks if the string token in account is an INTEGER number. It works only for ini Files

**Parameters**

*str* : string token in account

*fname* : parameter filename

*name* : name of each parameter in the file

**11.8.1.17 void check\_argv\_errors ( char \* *argv*[], int *argc* )**

routine that checks the correctness of command line arguments

**Parameter**

*argv*[] : array of command line arguments

*argc* : number of command line arguments

**11.8.1.18 void check\_float\_string ( char \* *str*, int *row*, char \* *fname* )**

routine that checks if the string token in account reading a generic FILE is an Float number

**Parameters**

*str* : string token in account

*row* : number of row where the string is found.

*fname* : filename read

**11.8.1.19** void check\_float\_string\_iniFile ( const char \* *str*, char \* *fname*, char \* *name* )

routine that checks if the string token in account is an Float number. It works only for ini Files

#### Parameters

*str* : string token in account

*fname* : parameter filename

*name* : name of each parameter in the file

**11.8.1.20** int columns ( char \* *string* )

routine that returns the number of columns for each row inside the file chosen.

#### Parameter

*string* : string token in account

**11.8.1.21** void mandatory\_files\_present ( arguments \* *arguments*, char \* *argv*[] )

routine that checks if the mandatory files are present

#### Parameters

*arguments* : command line arguments

*argv*[] : array of command line arguments

**11.8.1.22** void read\_ParameterFile ( arguments \* *arguments*, parameters \* *cc* )

routine that reads the input parameter file

#### Parameter

*ParameterFileName* : parameter filename

**11.8.1.23** void check\_optional\_parameters ( parameters \* *cc* )

routine that checks optional parameters for the tasks that need them

#### Parameters

*cc* : parameters

**11.8.1.24** void check\_parameters ( int \* *pars*, char \*\* *pars\_names*, int *n\_pars* )

routine that checks if all mandatory parameters are correctly provided

**Parameter**

`pars` : parameters

`pars_names` : names of parameters

`n_pars` : number of parameters

**11.8.1.25** `void read_mapping_matrix ( char * mappings_filename, FILE * f_out_l, cg_mapping * mapping_matrix[], int nmaps )`

routine that reads the input mapping matrix

**Parameters**

`filename` : mapping filename

`f_out_l` : output filename

`cg_mapping` : `cg_mapping` object routine that reads the input mapping matrix

**Parameters**

`filename` : mapping filename

`f_out_l` : output filename

`cg_mapping` : `cg_mapping` object

`nmaps` : number of mappings defined in parameter file

**11.8.2 Variable Documentation**

**11.8.2.1** `char doc_main[]` `[static]`

**Initial value:**

```
=
"\n-----\n\
Please, choose one of the following tasks:\n\n\
  *random*      To randomly generate coarse-grained representations\n\
                  and measure the associated mapping entropies;\n\n\
  *optimize*    To optimize the coarse-grained mapping by minimising\n\
                  its mapping entropy\n\n\
  *measure*     To measure the mapping entropy of a mapping\n\
                  provided by the user (in the form of a .txt file)\n\n\
  *norm*        To calculate the norm of a mapping (provided by the user)\n\
                  throughout a trajectory\n\n\
  *cosine*      To calculate pairwise distance and cosine between a pair\n\
                  of mappings (provided by the user) throughout a trajectory\n\n\
  *distance*    To calculate the distance matrix between a data set\n\
                  of mappings (provided by the user) over a single conformation\n\
                  of mappings\n\n\
-----\n\
Hereafter the list of OPTIONS:"
```

**11.8.2.2** `char args_doc_main[] = "random\noptimize\nmeasure\nnorm\ncosine\ndistance"` `[static]`

**11.8.2.3** `struct argp_option options_main[]` `[static]`

**Initial value:**

```
= {
  { "verbose",      'v',      0, 0, "Produce verbose output" },
```



```

{"quiet",      'q',      0, OPTION_HIDDEN, "Don't produce any output" },
{"help",      'h',      0, 0, "Give this help list"},
{"p",        'p',      "FILE", OPTION_HIDDEN, "Parameter file in ini.format (mandatory)"},
{"e",        'e',      "FILE", OPTION_HIDDEN, "Energy file (mandatory for tasks 0, 1, 2, 3)"},
{"m1",       'm',      "FILE", OPTION_HIDDEN, "Mapping file1 (mandatory for tasks 2, 4, 5)" },
{"m2",       'n',      "FILE", OPTION_HIDDEN, "Mapping file2 (mandatory for task 5)" },
{"t",        't',      "FILE", OPTION_HIDDEN, "Trajectory file in .xyz format (mandatory)"},
{"code",     'c',      "STR",  OPTION_HIDDEN, "String that identifies your structure (mandatory)"},
{"matrix",   'x',      "STR",  OPTION_HIDDEN, "mapping_matrix"},
{"prob",     'r',      "FILE", OPTION_HIDDEN, "Probability file"},
{ 0 }
}

```

11.8.2.4 struct argp argp = { options\_main, parse\_opt, args\_doc\_main, doc\_main } [static]

## 11.9 include/mapping.h File Reference

```
#include <stdio.h>
```

### Classes

- class [cg\\_mapping](#)  
*structure that defines a cg mapping*

### Typedefs

- typedef struct [cg\\_mapping](#) [cg\\_mapping](#)

### Functions

- void [free\\_mapping](#) ([cg\\_mapping](#) \*mapping)
- void [convert\\_mapping](#) ([cg\\_mapping](#) \*mapping, FILE \*f\_out)
- void [generate\\_random\\_mapping](#) ([cg\\_mapping](#) \*mapping, FILE \*f\_out)
- void [update\\_mapping](#) ([cg\\_mapping](#) \*curr\_mapping, [cg\\_mapping](#) \*old\_mapping, int frames)
- void [read\\_MappingFile](#) (char \*MappingFileName, FILE \*f\_out\_l, [cg\\_mapping](#) \*mapping)
- void [read\\_mapping\\_matrix](#) (char \*mappings\_filename, FILE \*f\_out\_l, [cg\\_mapping](#) \*mapping\_matrix[], int nmaps)

### 11.9.1 Typedef Documentation

#### 11.9.1.1 typedef struct [cg\\_mapping](#) [cg\\_mapping](#)

### 11.9.2 Function Documentation

#### 11.9.2.1 void [free\\_mapping](#) ( [cg\\_mapping](#) \* *mapping* )

routine that frees the mapping

#### Parameters

*mapping*: [cg\\_mapping](#) object

#### 11.9.2.2 void convert\_mapping ( cg\_mapping \* mapping, FILE \* f\_out )

routine that prints out the mapping

##### Parameters

mapping : [cg\\_mapping](#) object

f\_out : file to write on

#### 11.9.2.3 void generate\_random\_mapping ( cg\_mapping \* mapping, FILE \* f\_out )

routine that generates a random mapping

##### Parameters

mapping : [cg\\_mapping](#) object

f\_out : file to write on

#### 11.9.2.4 void update\_mapping ( cg\_mapping \* curr\_mapping, cg\_mapping \* old\_mapping, int frames )

routine that updates old\_mapping with the data contained in curr\_mapping

##### Parameters

curr\_mapping : current [cg\\_mapping](#) object

old\_mapping : [cg\\_mapping](#) object to be updated

frames : length of the MD trajectory

#### 11.9.2.5 void read\_MappingFile ( char \* MappingFileName, FILE \* f\_out\_l, cg\_mapping \* mapping )

routine that reads the input mapping file

##### Parameters

MappingFileName : mapping filename

f\_out\_l : output filename

cg\_mapping : [cg\\_mapping](#) object

#### 11.9.2.6 void read\_mapping\_matrix ( char \* mappings\_filename, FILE \* f\_out\_l, cg\_mapping \* mapping\_matrix[], int nmaps )

routine that reads the input mapping matrix

##### Parameters

filename : mapping filename

f\_out\_l : output filename

cg\_mapping : [cg\\_mapping](#) object routine that reads the input mapping matrix

**Parameters**

filename : mapping filename

f\_out\_l : output filename

cg\_mapping : [cg\\_mapping](#) object

nmaps : number of mappings defined in parameter file

**11.10 include/measure.h File Reference**

```
#include <stdio.h>
```

```
#include <io.h>
```

**Functions**

- void [measure](#) (arguments \*arguments, parameters \*cc)

**11.10.1 Function Documentation**

11.10.1.1 void [measure](#) ( arguments \* arguments, parameters \* cc )

subprogram to measure the mapping entropy of a mapping provided by the user

**Parameters**

arguments : arguments object (command line arguments)

parameters : parameters object

**11.11 include/measure\_kl.h File Reference**

```
#include <stdio.h>
```

```
#include <io.h>
```

**Functions**

- void [measure\\_kl](#) (arguments \*arguments, parameters \*cc)

**11.11.1 Function Documentation**

11.11.1.1 void [measure\\_kl](#) ( arguments \* arguments, parameters \* cc )

subprogram to measure the KL divergence version of the mapping entropy for a mapping provided by the user

**Parameters**

arguments : arguments object (command line arguments)

parameters : parameters object

## 11.12 include/my\_malloc.h File Reference

```
#include "stdio.h"
```

### Functions

- void [readeol](#) (FILE \*fp)
- FILE \*\* [F2t](#) (int n1)
- FILE \*\*\* [F3t](#) (int n1, int n2)
- char \* [c1t](#) (int n1)
- char \*\* [c2t](#) (int n1, int n2)
- char \*\*\* [c3t](#) (int n1, int n2, int n3)
- char \*\*\*\* [c4t](#) (int n1, int n2, int n3, int n4)
- void [free\\_c4t](#) (char \*\*\*\*p)
- void [free\\_c3t](#) (char \*\*\*p)
- void [free\\_c2t](#) (char \*\*p)
- void [free\\_c1t](#) (char \*p)
- short \* [s1t](#) (int n1)
- short \*\* [s2t](#) (int n1, int n2)
- short \*\*\* [s3t](#) (int n1, int n2, int n3)
- short \*\*\*\* [s4t](#) (int n1, int n2, int n3, int n4)
- void [free\\_s4t](#) (short \*\*\*\*p)
- void [free\\_s3t](#) (short \*\*\*p)
- void [free\\_s2t](#) (short \*\*p)
- void [free\\_s1t](#) (short \*p)
- int \* [i1t](#) (int n1)
- int \*\* [i2t](#) (int n1, int n2)
- int \*\*\* [i3t](#) (int n1, int n2, int n3)
- int \*\*\*\* [i4t](#) (int n1, int n2, int n3, int n4)
- void [free\\_i4t](#) (int \*\*\*\*p)
- void [free\\_i3t](#) (int \*\*\*p)
- void [free\\_i2t](#) (int \*\*p)
- void [free\\_i1t](#) (int \*p)
- float \* [f1t](#) (int n1)
- float \*\* [f2t](#) (int n1, int n2)
- float \*\*\* [f3t](#) (int n1, int n2, int n3)
- float \*\*\*\* [f4t](#) (int n1, int n2, int n3, int n4)
- float \*\*\*\*\* [f5t](#) (int n1, int n2, int n3, int n4, int n5)
- void [free\\_f5t](#) (float \*\*\*\*\*p)
- void [free\\_f4t](#) (float \*\*\*\*p)
- void [free\\_f3t](#) (float \*\*\*p)
- void [free\\_f2t](#) (float \*\*p)
- void [free\\_f1t](#) (float \*p)
- double \* [d1t](#) (int n1)
- double \*\* [d2t](#) (int n1, int n2)
- double \*\*\* [d3t](#) (int n1, int n2, int n3)
- double \*\*\*\* [d4t](#) (int n1, int n2, int n3, int n4)
- void [free\\_d4t](#) (double \*\*\*\*p)
- void [free\\_d3t](#) (double \*\*\*p)
- void [free\\_d2t](#) (double \*\*p)
- void [free\\_d1t](#) (double \*p)
- void [pdarray](#) (int n, int m, double \*\*a)
- void [pdvector](#) (int n, double \*a)

- void [pfarray](#) (int n, int m, float \*\*a)
- void [pfvector](#) (int n, float \*a)
- void [piarray](#) (int n, int m, int \*\*a)
- void [pivector](#) (int n, int \*a)
- void [zdarray](#) (int n, int m, double \*\*a)
- void [zdvector](#) (int n, double \*a)
- void [zfarray](#) (int n, int m, float \*\*a)
- void [zfvector](#) (int n, float \*a)
- void [ziarray](#) (int n, int m, int \*\*a)
- void [zivector](#) (int n, int \*a)
- void [failed](#) (char msg[])

## 11.12.1 Function Documentation

11.12.1.1 void [readeol](#) ( FILE \* *fp* )

11.12.1.2 FILE\*\* [F2t](#) ( int *n1* )

11.12.1.3 FILE\*\*\* [F3t](#) ( int *n1*, int *n2* )

11.12.1.4 char\* [c1t](#) ( int *n1* )

11.12.1.5 char\*\* [c2t](#) ( int *n1*, int *n2* )

11.12.1.6 char\*\*\* [c3t](#) ( int *n1*, int *n2*, int *n3* )

11.12.1.7 char\*\*\*\* [c4t](#) ( int *n1*, int *n2*, int *n3*, int *n4* )

11.12.1.8 void [free\\_c4t](#) ( char \*\*\*\* *p* )

11.12.1.9 void [free\\_c3t](#) ( char \*\*\* *p* )

11.12.1.10 void [free\\_c2t](#) ( char \*\* *p* )

11.12.1.11 void [free\\_c1t](#) ( char \* *p* )

11.12.1.12 short\* [s1t](#) ( int *n1* )

11.12.1.13 short\*\* [s2t](#) ( int *n1*, int *n2* )

11.12.1.14 short\*\*\* [s3t](#) ( int *n1*, int *n2*, int *n3* )

11.12.1.15 short\*\*\*\* [s4t](#) ( int *n1*, int *n2*, int *n3*, int *n4* )

11.12.1.16 void [free\\_s4t](#) ( short \*\*\*\* *p* )

11.12.1.17 void [free\\_s3t](#) ( short \*\*\* *p* )

11.12.1.18 void [free\\_s2t](#) ( short \*\* *p* )

11.12.1.19 void [free\\_s1t](#) ( short \* *p* )

11.12.1.20 int\* [i1t](#) ( int *n1* )

11.12.1.21 int\*\* [i2t](#) ( int *n1*, int *n2* )

11.12.1.22 `int*** i3t ( int n1, int n2, int n3 )`

11.12.1.23 `int**** i4t ( int n1, int n2, int n3, int n4 )`

11.12.1.24 `void free_i4t ( int **** p )`

11.12.1.25 `void free_i3t ( int *** p )`

11.12.1.26 `void free_i2t ( int ** p )`

11.12.1.27 `void free_i1t ( int * p )`

11.12.1.28 `float* f1t ( int n1 )`

11.12.1.29 `float** f2t ( int n1, int n2 )`

11.12.1.30 `float*** f3t ( int n1, int n2, int n3 )`

11.12.1.31 `float**** f4t ( int n1, int n2, int n3, int n4 )`

11.12.1.32 `float***** f5t ( int n1, int n2, int n3, int n4, int n5 )`

11.12.1.33 `void free_f5t ( float ***** p )`

11.12.1.34 `void free_f4t ( float ***** p )`

11.12.1.35 `void free_f3t ( float **** p )`

11.12.1.36 `void free_f2t ( float *** p )`

11.12.1.37 `void free_f1t ( float * p )`

11.12.1.38 `double* d1t ( int n1 )`

11.12.1.39 `double** d2t ( int n1, int n2 )`

11.12.1.40 `double*** d3t ( int n1, int n2, int n3 )`

11.12.1.41 `double**** d4t ( int n1, int n2, int n3, int n4 )`

11.12.1.42 `void free_d4t ( double **** p )`

11.12.1.43 `void free_d3t ( double *** p )`

11.12.1.44 `void free_d2t ( double ** p )`

11.12.1.45 `void free_d1t ( double * p )`

11.12.1.46 `void pdarray ( int n, int m, double ** a )`

11.12.1.47 `void pdvector ( int n, double * a )`

11.12.1.48 `void pffarray ( int n, int m, float ** a )`

11.12.1.49 `void pfvector ( int n, float * a )`

11.12.1.50 void piarray ( int *n*, int *m*, int \*\* *a* )

11.12.1.51 void pivector ( int *n*, int \* *a* )

11.12.1.52 void zdarray ( int *n*, int *m*, double \*\* *a* )

11.12.1.53 void zdvector ( int *n*, double \* *a* )

11.12.1.54 void zfarray ( int *n*, int *m*, float \*\* *a* )

11.12.1.55 void zfvector ( int *n*, float \* *a* )

11.12.1.56 void ziarray ( int *n*, int *m*, int \*\* *a* )

11.12.1.57 void zivector ( int *n*, int \* *a* )

11.12.1.58 void failed ( char *msg*[] )

## 11.13 include/norm.h File Reference

```
#include <stdio.h>
#include <io.h>
```

### Functions

- void [norm](#) (arguments \*arguments, parameters \*cc)

#### 11.13.1 Function Documentation

11.13.1.1 void norm ( arguments \* arguments, parameters \* cc )

subprogram to To calculate the norm of a mapping (provided by the user) throughout a trajectory

#### Parameters

arguments : arguments object (command line arguments)

parameters : parameters object

## 11.14 include/observables.h File Reference

```
#include <stdio.h>
#include <mapping.h>
#include <hierarchical_clustering.h>
#include <traj.h>
#include <alignment.h>
```

### Functions

- void [compute\\_coupling\\_matrix](#) (double \*coupling\_mat, [traj](#) \*Trajectory, int fr\_id, float sigma)

- double `compute_atomistic_coord_number` (double \*coupling\_mat, `traj` \*Trajectory, FILE \*f\_out\_l)
- void `compute_norm` (`cg_mapping` \*mapping, double \*coupling\_mat, double n\_coord\_at, int fr\_id, FILE \*f\_out\_l)
- double `compute_distance` (`cg_mapping` \*mapping, `cg_mapping` \*mapping\_prime, double \*coupling\_mat, double n\_coord\_at, int fr\_id, FILE \*f\_out\_l)
- void `compute_mapping_norms` (`traj` \*Trajectory, `cg_mapping` \*mapping, FILE \*f\_out\_l)
- void `compute_mapping_distances` (`traj` \*Trajectory, `cg_mapping` \*mapping, `cg_mapping` \*mapping\_prime, FILE \*f\_out\_l)
- void `compute_mapping_distmat` (`traj` \*Trajectory, `cg_mapping` \*mapping\_matrix[], int nmaps, FILE \*f\_out\_l, char \*distmat\_filename)
- void `compute_variances` (int Nclust, double \*variances, int \*cluster\_list, int \*cluster\_list\_idx, double \*energies)
- double `get_smap` (int frames, int curr\_nclust, int \*clusters, double \*energies)
- void `overall_compute_smap` (`alignments` \*align, `clust_params` \*clustering, `traj` \*Trajectory, `cg_mapping` \*mapping, int verbose, int kl\_flag)
- void `compute_smap_spins` (`spin_traj` \*Trajectory, `cg_mapping` \*mapping)

### 11.14.1 Function Documentation

#### 11.14.1.1 void compute\_coupling\_matrix ( double \* coupling\_mat, traj \* Trajectory, int fr\_id, float sigma )

routine that computes the coupling matrix over a frame

`coupling_mat` : coupling matrix

`Trajectory` : traj object

`fr_id` : frame ID

`sigma` : sigma parameter

#### 11.14.1.2 double compute\_atomistic\_coord\_number ( double \* coupling\_mat, traj \* Trajectory, FILE \* f\_out\_l )

routine that computes the atomistic coordination number for a certain coupling matrix. Double counting is necessary to ensure proper normalisation to norm and scalar product

`coupling_mat` : coupling matrix

`Trajectory` : traj object

`f_out_l` : output filename

#### 11.14.1.3 void compute\_norm ( cg\_mapping \* mapping, double \* coupling\_mat, double n\_coord\_at, int fr\_id, FILE \* f\_out\_l )

routine that computes the norm of a mapping over a frame of a trajectory

#### Parameters

`mapping` : `cg_mapping` object

`coupling_mat` : coupling matrix

`n_coord_at` : atomistic coordination number

`fr_id` : frame index

`f_out_l` : output filename



**11.14.1.4** `double compute_distance ( cg_mapping * mapping, cg_mapping * mapping_prime, double * coupling_mat, double n_coord_at, int fr_id, FILE * f_out_l )`

routine that computes the distance and cosine between a pair of cg mappings

#### Parameters

mapping, mapping\_prime : [cg\\_mapping](#) objects

coupling\_mat : coupling matrix

n\_coord\_at : atomistic coordination number

fr\_id : frame index

f\_out\_l : output filename

**11.14.1.5** `void compute_mapping_norms ( traj * Trajectory, cg_mapping * mapping, FILE * f_out_l )`

routine that computes the norm of a mapping over a MD trajectory

#### Parameters

Trajectory : traj object

mapping : [cg\\_mapping](#) object

f\_out\_l : output filename

**11.14.1.6** `void compute_mapping_distances ( traj * Trajectory, cg_mapping * mapping, cg_mapping * mapping_prime, FILE * f_out_l )`

routine that computes the distances and cosines between two mappings provided by the user over a MD trajectory

#### Parameters

Trajectory : traj object

mapping, mapping\_prime : [cg\\_mapping](#) objects

f\_out\_l : output filename

**11.14.1.7** `void compute_mapping_distmat ( traj * Trajectory, cg_mapping * mapping_matrix[], int nmaps, FILE * f_out_l, char * distmat_filename )`

routine that computes the distance matrix between a set of mappings over a single structure

#### Parameters

Trajectory : traj object

mappings\_filename : filename with the chosen mappings

namps : number of mappings

f\_out\_l : output filename

**11.14.1.8** `void compute_variances ( int Nclust, double * variances, int * cluster_list, int * cluster_list_idx, double * energies )`

routine that computes the variance of the energies

#### Parameters

`nclust` : number of macrostates

`variances` : vector of variances

`cluster_list` : list of cluster IDs

`cluster_list_idx` : list of cluster indices

`energies` : array of energies

**11.14.1.9** `double get_smap ( int frames, int curr_nclust, int * clusters, double * energies )`

routine that computes the observable given the current `nclust` and the current `clusters`

#### Parameters

`frames` : number of frames

`curr_nclust` : current index of CG macrostate

`clusters` : list of cluster IDs

`energies` : array of energies

**11.14.1.10** `void overall_compute_smap ( alignments * align, clust_params * clustering, traj * Trajectory, cg_mapping * mapping, int verbose, int kl_flag )`

routine that calls `get_smap` with the correct parameters

#### Parameters

`rmsd_mat` : condensed matrix of pairwise RMSDs

`clustering` : [clust\\_params](#) object

`Trajectory` : traj object

`mapping` : [cg\\_mapping](#) object

`verbose` : tunes the level of verbosity

`f_out` : output filename

**11.14.1.11** `void compute_smap_spins ( spin_traj * Trajectory, cg_mapping * mapping )`

## 11.15 include/optimize.h File Reference

```
#include <stdio.h>
```

```
#include <io.h>
```

## Functions

- void `optimize` (`arguments *arguments`, `parameters *cc`)

### 11.15.1 Function Documentation

11.15.1.1 void `optimize` ( `arguments * arguments`, `parameters * cc` )

subprogram to optimize the coarse-grained mapping by minimising its mapping entropy

#### Parameters

`arguments`: `arguments` object (command line arguments)

`parameters`: `parameters` object

## 11.16 include/optimize\_kl.h File Reference

```
#include <stdio.h>
```

```
#include <io.h>
```

## Functions

- void `optimize_kl` (`arguments *arguments`, `parameters *cc`)

### 11.16.1 Function Documentation

11.16.1.1 void `optimize_kl` ( `arguments * arguments`, `parameters * cc` )

subprogram to optimize the coarse-grained mapping by minimising the KL divergence version of its mapping entropy

#### Parameters

`arguments`: `arguments` object (command line arguments)

`parameters`: `parameters` object

## 11.17 include/optimize\_spins.h File Reference

```
#include <stdio.h>
```

```
#include <io.h>
```

## Functions

- int `spin_clustering` (int `**traj_coords`, int `*mapping`, int `atomnum`, int `frames`, int `*mapping_clusters`, int `cgnum`)
- void `optimize_spins` (`arguments *arguments`, `parameters *cc`)

### 11.17.1 Function Documentation

11.17.1.1 `int spin_clustering ( int ** traj_coords, int * mapping, int atomnum, int frames, int * mapping_clusters, int cgnum )`

spin clustering

11.17.1.2 `void optimize_spins ( arguments * arguments, parameters * cc )`

subprogram to optimize the coarse-grained mapping by minimising the KL divergence version of its mapping entropy

#### Parameters

`arguments` : `arguments` object (command line arguments)

`parameters` : `parameters` object

## 11.18 include/random\_mapping.h File Reference

```
#include <stdio.h>
```

```
#include <io.h>
```

### Functions

- void `random_mapping` (`arguments` \*`arguments`, `parameters` \*`cc`)

### 11.18.1 Function Documentation

11.18.1.1 `void random_mapping ( arguments * arguments, parameters * cc )`

subprogram to randomly generate coarse-grained representations and measure the associated mapping entropies

#### Parameters

`arguments` : `arguments` object (command line arguments)

`parameters` : `parameters` object

## 11.19 include/random\_mapping\_kl.h File Reference

```
#include <stdio.h>
```

```
#include <io.h>
```

### Functions

- void `random_mapping_kl` (`arguments` \*`arguments`, `parameters` \*`cc`)

### 11.19.1 Function Documentation

#### 11.19.1.1 void random\_mapping\_kl ( arguments \* arguments, parameters \* cc )

subprogram to randomly generate coarse-grained representations and measure the KL divergence version of their mapping entropy

#### Parameters

arguments : arguments object (command line arguments)

parameters : parameters object

## 11.20 include/sampling.h File Reference

```
#include <mapping.h>
#include <alignment.h>
#include <hierarchical_clustering.h>
```

### Classes

- class [MC\\_params](#)

*structure that defines a the parameters of Monte Carlo sampling*

### Typedefs

- typedef struct [MC\\_params](#) [MC\\_params](#)

### Functions

- void [my\\_make\\_a\\_move](#) ([cg\\_mapping](#) \*old\_mapping, [cg\\_mapping](#) \*new\_mapping, int rem\_add[2])
- void [simulated\\_annealing](#) ([traj](#) \*Trajectory, [clust\\_params](#) \*clustering, [MC\\_params](#) \*SA\_params, int cgnum, int rsd, int verbose, int kl\_flag, FILE \*f\_out\_l)
- double [tzero\\_estimation](#) ([traj](#) \*Trajectory, [clust\\_params](#) \*clustering, int cgnum, int rsd, int verbose, int kl\_flag, FILE \*f\_out\_l)
- double [tzero\\_estimation\\_spins](#) ([spin\\_traj](#) \*Trajectory, int cgnum, FILE \*f\_out\_l)

### 11.20.1 Typedef Documentation

#### 11.20.1.1 typedef struct MC\_params MC\_params

### 11.20.2 Function Documentation

#### 11.20.2.1 void my\_make\_a\_move ( cg\_mapping \* old\_mapping, cg\_mapping \* new\_mapping, int rem\_add[2] )

function that swaps two atoms inside a CG mapping

**Parameters**

old\_mapping : [cg\\_mapping](#) object

new\_mapping : [cg\\_mapping](#) object

rem\_add : vector of length 2 containing the removed and added atom index

**11.20.2.2** void simulated\_annealing ( traj \* *Trajectory*, clust\_params \* *clustering*, MC\_params \* *SA\_params*, int *cgnum*, int *rsd*, int *verbose*, int *kl\_flag*, FILE \* *f\_out\_l* )

simulated annealing optimisation

**Parameters**

Trajectory : traj object

alignments : align object

mapping : [cg\\_mapping](#) object

SA\_params : set of Monte Carlo parameters

verbose : tunes the level of verbosity

f\_out\_l : output filename

**11.20.2.3** double tzero\_estimation ( traj \* *Trajectory*, clust\_params \* *clustering*, int *cgnum*, int *rsd*, int *verbose*, int *kl\_flag*, FILE \* *f\_out\_l* )

routine that makes *nrun* unbiased moves. It is used to estimate the starting temperature for Simulated Annealing.

**Parameters**

Trajectory : traj object

alignments : align object

mapping : [cg\\_mapping](#) object

verbose : tunes the level of verbosity

**11.20.2.4** double tzero\_estimation\_spins ( spin\_traj \* *Trajectory*, int *cgnum*, FILE \* *f\_out\_l* )

routine that makes *nrun* unbiased moves. It is used to estimate the starting temperature for Simulated Annealing.

**Parameters**

Trajectory : traj object

alignments : align object

mapping : [cg\\_mapping](#) object

verbose : tunes the level of verbosity

## 11.21 include/traj.h File Reference

```
#include <stdio.h>
#include <string.h>
#include <argp.h>
#include <ini.h>
#include <io.h>
```

### Classes

- class [traj](#)  
*structure that defines a MD trajectory*
- struct [spin\\_traj](#)

### Typedefs

- typedef struct [traj](#) [traj](#)
- typedef struct [spin\\_traj](#) [spin\\_traj](#)

### Functions

- int [check\\_probabilities](#) (double \*probabilities, int prob\_length)
- void [read\\_EnergyFile](#) (char \*EnergyFileName, [traj](#) \*Trajectory)
- void [read\\_TrajectoryFile](#) (char \*TrajFileName, [traj](#) \*Trajectory)
- void [read\\_spinFile](#) (char \*TrajFileName, [spin\\_traj](#) \*Trajectory)

#### 11.21.1 Typedef Documentation

11.21.1.1 typedef struct [traj](#) [traj](#)

11.21.1.2 typedef struct [spin\\_traj](#) [spin\\_traj](#)

#### 11.21.2 Function Documentation

11.21.2.1 int [check\\_probabilities](#) ( double \* *probabilities*, int *prob\_length* )

routine that checks that input probabilities sum to 1

##### Parameters

*probabilities* : array of probabilities

*prob\_length* : array length

11.21.2.2 void [read\\_EnergyFile](#) ( char \* *EnergyFileName*, [traj](#) \* *Trajectory* )

routine that reads the input energy file

**Parameters**

EnergyFileName : energies filename

Trajectory : traj object

**11.21.2.3 void read\_TrajectoryFile ( char \* TrajFileName, traj \* Trajectory )**

routine that reads the input xyz coordinate file

**Parameters**

TrajFileName : trajectory filename

Trajectory : traj object

**11.21.2.4 void read\_spinFile ( char \* TrajFileName, spin\_traj \* Trajectory )**

routine that reads the input xyz coordinate file

**Parameters**

TrajFileName : trajectory filename

Trajectory : traj object

**11.22 lib/alignment.c File Reference**

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <my_malloc.h>
#include <alignment.h>
#include <geometry.h>
```

**Functions**

- void [free\\_new\\_alignment](#) (alignments \*new\_align)
- void [free\\_alignment](#) (alignments \*align)
- double [optimal\\_alignment](#) (double \*\*x, double \*\*y, int cgnum, double u[][3])
- void [align\\_two\\_frames](#) (double \*frame\_ref, double \*frame\_middle, int ref\_id, int middle\_id, [cg\\_mapping](#) \*mapping, alignments \*align)
- void [cycle\\_alignment\\_stride](#) (traj \*Trajectory, alignments \*align, [cg\\_mapping](#) \*mapping)
- void [cycle\\_alignment\\_fastclust](#) (traj \*Trajectory, alignments \*align, [cg\\_mapping](#) \*mapping)
- void [cycle\\_alignment](#) (traj \*Trajectory, alignments \*align, [cg\\_mapping](#) \*mapping)
- void [correct\\_rmsd](#) (alignments \*new\_align, traj \*Trajectory, alignments \*align, int cgnum, int removed, int added)
- double [correct\\_rmsd\\_two\\_frames](#) (traj \*Trajectory, double u[9], double com\_ref[3], double com\_other[3], int cgnum, int removed, int added, int ref\_id, int other\_id, double prev\_rmsd)
- void [correct\\_rmsd\\_fastclust](#) (alignments \*new\_align, traj \*Trajectory, alignments \*prev\_align, int cgnum, int removed, int added)
- void [align\\_traj\\_to\\_reference](#) (traj \*Trajectory, int ref\_id)



### 11.22.1 Function Documentation

#### 11.22.1.1 void free\_new\_alignment ( alignments \* new\_align )

routine that frees an alignments object used in criterion 1

##### Parameters

new\_align: alignments object

#### 11.22.1.2 void free\_alignment ( alignments \* align )

routine that frees an alignments object

##### Parameters

align: alignments object

#### 11.22.1.3 double optimal\_alignment ( double \*\* x, double \*\* y, int cnum, double u[][3] )

routine that computes the Kabsch alignment and the rmsd between two configurations

##### Parameters

x, y : CG structures

cnum : length of CG mapping

u : rotation matrix

#### 11.22.1.4 void align\_two\_frames ( double \* frame\_ref, double \* frame\_middle, int ref\_id, int middle\_id, cg\_mapping \* mapping, alignments \* align )

routine that aligns a pair of frames in a trajectory, calling optimal\_alignment

##### Parameters

frame\_ref : reference frame

frame\_middle : frame in between two pivot clusters

ref\_id : id (index) of frame\_ref in the trajectory

middle\_id : id (index) of frame\_middle in the trajectory

mapping : [cg\\_mapping](#) object

align : alignments object

#### 11.22.1.5 void cycle\_alignment\_stride ( traj \* Trajectory, alignments \* align, cg\_mapping \* mapping )

routine that cycles over all pairs of frames in a trajectory, calling optimal\_alignment

**Parameters**

Trajectory : traj object  
align : alignments object  
mapping : [cg\\_mapping](#) object

**11.22.1.6 void cycle\_alignment\_fastclust ( traj \* *Trajectory*, alignments \* *align*, cg\_mapping \* *mapping* )**

routine that computes the alignments if clustering must be fast

**Parameters**

Trajectory : traj object  
align : alignments object  
mapping : [cg\\_mapping](#) object

**11.22.1.7 void cycle\_alignment ( traj \* *Trajectory*, alignments \* *align*, cg\_mapping \* *mapping* )**

routine that cycles over all pairs of frames in a trajectory, calling `optimal_alignment`

**Parameters**

Trajectory : traj object  
align : alignments object  
mapping : [cg\\_mapping](#) object

**11.22.1.8 void correct\_rmsd ( alignments \* *new\_align*, traj \* *Trajectory*, alignments \* *align*, int *cgnum*, int *removed*, int *added* )**

routine that computes the rmsd matrix without aligning frames over frames

**Parameters**

*new\_align* : trial alignments object  
Trajectory : traj object  
align : alignments object  
*cgnum* : number of CG sites (useful to normalize)  
*removed* : index of removed atom  
*added* : index of added atom

**11.22.1.9 double correct\_rmsd\_two\_frames ( traj \* *Trajectory*, double *u*[9], double *com\_ref*[3], double *com\_other*[3], int *cgnum*, int *removed*, int *added*, int *ref\_id*, int *other\_id*, double *prev\_rmsd* )**

routine that corrects the rmsd between two frames

**Parameters**

Trajectory : traj object  
 u : rotation matrix  
 com\_ref : reference center of mass  
 com\_other : other center of mass  
 removed : index of removed atom  
 added : index of added atom  
 ref\_id : index of reference frame  
 other\_id : index of other frame  
 prev\_rmsd : previous rmsd

**11.22.1.10** void correct\_rmsd\_fastclust ( alignments \* new\_align, traj \* Trajectory, alignments \* prev\_align, int cgnum, int removed, int added )

routine that computes the rmsd matrix without aligning frames over frames

**Parameters**

new\_rmsd\_mat : new condensed pairwise RMSD matrix  
 Trajectory : traj object  
 align : alignments object  
 cgnum : number of CG sites (useful to normalize)  
 removed : index of removed atom  
 added : index of added atom

**11.22.1.11** void align\_traj\_to\_reference ( traj \* Trajectory, int ref\_id )

routine that aligns the trajectory to a reference frame

**Parameters**

Trajectory : traj object  
 ref\_id : reference frame

**11.23 lib/cosine.c File Reference**

```
#include <io.h>
#include <stdlib.h>
#include <my_malloc.h>
#include <traj.h>
#include <mapping.h>
#include <math.h>
#include <geometry.h>
#include <observables.h>
```

## Functions

- void `cosine` (`arguments *arguments`, `parameters *cc`)

### 11.23.1 Function Documentation

#### 11.23.1.1 void `cosine` ( `arguments * arguments`, `parameters * cc` )

subprogram to calculate pairwise distance and cosine between a pair of mappings (provided by the user) throughout a trajectory

#### Parameters

`arguments` : `arguments` object (command line arguments)

`parameters` : `parameters` object

## 11.24 lib/distance.c File Reference

```
#include <io.h>
#include <stdlib.h>
#include <my_malloc.h>
#include <traj.h>
#include <mapping.h>
#include <math.h>
#include <geometry.h>
#include <observables.h>
```

## Functions

- void `distance` (`arguments *arguments`, `parameters *cc`)

### 11.24.1 Function Documentation

#### 11.24.1.1 void `distance` ( `arguments * arguments`, `parameters * cc` )

subprogram to calculate the distance matrix between a data set of mappings (provided by the user) over a single conformation

#### Parameters

`arguments` : `arguments` object (command line arguments)

`parameters` : `parameters` object

## 11.25 lib/geometry.c File Reference

```
#include <stdio.h>
#include <math.h>
#include <geometry.h>
```

## Macros

- `#define ROTATE(a, i, j, k, l) g=a[i][j];h=a[k][l];a[i][j]=g-s*(h+g*tau);a[k][l]=h+s*(g-h*tau);`

## Functions

- void `vecprod_d` (double \*a, double \*b, double \*c)
- double `scal_d` (double \*a, double \*b, int dim)
- double `coseno` (double \*vec1, double \*vec2, int dim)
- double `norm_d` (double \*a, int dim)
- void `normalize_d` (double \*a, int dim)
- double `dist_d` (double \*a, double \*b, int dim)
- double `det` (double a1, double a2, double a3, double b1, double b2, double b3, double c1, double c2, double c3)
- void `vec_sum_d` (double \*a, double \*b, double \*c, double d, int dim)
- void `print_vec_d` (double \*a, int dim)
- void `zero_vec_d` (double \*a, int dim)
- void `zero_vec_i` (int \*a, int dim)
- void `myjacobi` (double a[][3], int n, double \*d, double v[][3], int \*nrot)
- void `zero_matrix_d` (double \*\*a, int dim1, int dim2)

### 11.25.1 Macro Definition Documentation

11.25.1.1 `#define ROTATE( a, i, j, k, l ) g=a[i][j];h=a[k][l];a[i][j]=g-s*(h+g*tau);a[k][l]=h+s*(g-h*tau);`

### 11.25.2 Function Documentation

11.25.2.1 void `vecprod_d` ( double \* *a*, double \* *b*, double \* *c* )

11.25.2.2 double `scal_d` ( double \* *a*, double \* *b*, int *dim* )

11.25.2.3 double `coseno` ( double \* *vec1*, double \* *vec2*, int *dim* )

11.25.2.4 double `norm_d` ( double \* *a*, int *dim* )

11.25.2.5 void `normalize_d` ( double \* *a*, int *dim* )

11.25.2.6 double `dist_d` ( double \* *a*, double \* *b*, int *dim* )

11.25.2.7 double `det` ( double *a1*, double *a2*, double *a3*, double *b1*, double *b2*, double *b3*, double *c1*, double *c2*, double *c3* )

11.25.2.8 void `vec_sum_d` ( double \* *a*, double \* *b*, double \* *c*, double *d*, int *dim* )

11.25.2.9 void `print_vec_d` ( double \* *a*, int *dim* )

11.25.2.10 void `zero_vec_d` ( double \* *a*, int *dim* )

11.25.2.11 void `zero_vec_i` ( int \* *a*, int *dim* )

11.25.2.12 void `myjacobi` ( double *a*[][3], int *n*, double \* *d*, double *v*[][3], int \* *nrot* )

11.25.2.13 void `zero_matrix_d` ( double \*\* *a*, int *dim1*, int *dim2* )

## 11.26 lib/hierarchical\_clustering.c File Reference

```
#include <stdlib.h>
#include <math.h>
#include <my_malloc.h>
#include <string.h>
#include <hierarchical_clustering.h>
```

### Functions

- void [mergesort\\_merge](#) (double \*\*arr, int l, int m, int r, int dim, int dims)
- void [my\\_mergesort](#) (double \*\*arr, int l, int r, int dim, int dims)
- int [condensed\\_index](#) (int frames, int i, int j)
- double [new\\_dist](#) (double d\_xi, double d\_yi, double d\_xy, int size\_x, int size\_y, int size\_i)
- int [is\\_visited](#) (unsigned char \*bitset, int i)
- void [set\\_visited](#) (unsigned char \*bitset, int i)
- void [get\\_max\\_dist\\_for\\_each\\_cluster](#) (double \*\*Z, double \*MD, int frames)
- void [cluster\\_monocrit](#) (double \*\*Z, double \*MC, int \*T, double cutoff, int frames)
- void [cluster\\_maxclust\\_monocrit](#) (double \*\*Z, double \*MC, int \*T, int frames, int max\_nc)
- void [cluster\\_maxclust\\_dist](#) (double \*\*Z, int \*T, int frames, int nclust)
- void [cluster\\_dist](#) (double \*\*Z, int \*T, double cutoff, int frames)
- int [find](#) (int x, int \*self\_parent)
- int [merge](#) (int \*self\_parent, int \*self\_size, int next\_label, int x, int y)
- void [label](#) (double \*\*Z, int frames)
- void [hierarchical\\_clustering](#) (double \*rmsd\_mat, int frames, int pairs, int \*size, double \*\*Z)
- void [compute\\_clusters\\_list](#) (int \*clusters, int \*cluster\_list, int \*cluster\_list\_idx, int frames, int nclust)

### 11.26.1 Function Documentation

11.26.1.1 void [mergesort\\_merge](#) ( double \*\* *arr*, int *l*, int *m*, int *r*, int *dim*, int *dims* )

11.26.1.2 void [my\\_mergesort](#) ( double \*\* *arr*, int *l*, int *r*, int *dim*, int *dims* )

11.26.1.3 int [condensed\\_index](#) ( int *frames*, int *i*, int *j* )

*frames* : number of observations

*i* : node

*j* : node

11.26.1.4 double [new\\_dist](#) ( double *d\_xi*, double *d\_yi*, double *d\_xy*, int *size\_x*, int *size\_y*, int *size\_i* )

11.26.1.5 int [is\\_visited](#) ( unsigned char \* *bitset*, int *i* )

routine that checks if node *i* was visited.

#### Parameters

*bitset* : char defining visits

*i* : node

11.26.1.6 void set\_visited ( unsigned char \* *bitset*, int *i* )

routine that marks node *i* as visited.

#### Parameters

*bitset* : char defining visits

*i* : node

11.26.1.7 void get\_max\_dist\_for\_each\_cluster ( double \*\* *Z*, double \* *MD*, int *frames* )

Get the maximum inconsistency coefficient for each non-singleton cluster.

#### Parameters

*Z* : linkage matrix.

*MD* : array to store the result.

*frames* : number of observations.

11.26.1.8 void cluster\_monocrit ( double \*\* *Z*, double \* *MC*, int \* *T*, double *cutoff*, int *frames* )

Form flat clusters by monocrit criterion.

#### Parameters

*Z* : linkage matrix.

*MC* : monotonic criterion array.

*T* : array to store the cluster numbers. The *i*'th observation belongs to cluster *T*[*i*].

*cutoff* : Clusters are formed when the MC values are less than or equal to *cutoff*.

*frames* : number of observations

11.26.1.9 void cluster\_maxclust\_monocrit ( double \*\* *Z*, double \* *MC*, int \* *T*, int *frames*, int *max\_nc* )

Form flat clusters by maxclust\_monocrit criterion.

#### Parameters

*Z* : linkage matrix

*MC* : monotonic criterion array

*T* : array to store the cluster numbers. The *i*'th observation belongs to cluster *T*[*i*]

*frames* : number of observations

*max\_nc* : The maximum number of clusters

11.26.1.10 void cluster\_maxclust\_dist ( double \*\* *Z*, int \* *T*, int *frames*, int *nclust* )

routine that converts the dendrogram into *nclust* clusters

**Parameters**

*Z* : linkage matrix.

*T* : array to store the cluster numbers. The *i*'th observation belongs to cluster *T* [*i*].

*frames* : number of observations.

*nclust* : number of desired clusters.

**11.26.1.11** void cluster\_dist ( double \*\* *Z*, int \* *T*, double *cutoff*, int *frames* )

**11.26.1.12** int find ( int *x*, int \* *self\_parent* )

**11.26.1.13** int merge ( int \* *self\_parent*, int \* *self\_size*, int *next\_label*, int *x*, int *y* )

**11.26.1.14** void label ( double \*\* *Z*, int *frames* )

routine that correctly labels clusters in the unsorted dendrogram

**Parameters**

*Z* : linkage matrix

*frames* : number of observations

**11.26.1.15** void hierarchical\_clustering ( double \* *rmsd\_mat*, int *frames*, int *pairs*, int \* *size*, double \*\* *Z* )

overall routine for hierarchical clustering

**Parameters**

*rmsd\_mat* : condensed pairwise RMSD matrix

*frames* : number of observations

*pairs* : possible pairs of structures

*size* : size of the clusters (it is *nclust* long)

*Z* : linkage matrix

**11.26.1.16** void compute\_clusters\_list ( int \* *clusters*, int \* *cluster\_list*, int \* *cluster\_list\_idx*, int *frames*, int *nclust* )

routine that computes the list of cluster IDs

**Parameters**

*clusters* : list of labels (one for each frame)

*cluster\_list* : ordered list of labels

*cluster\_list\_idx* : is an index vector that stores the sum of populations up to each index

*frames* : number of observations

*nclust* : number of clusters

< array that stores the population of each clusters



## 11.27 lib/ini.c File Reference

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include "ini.h"
#include <stdlib.h>
```

### Classes

- struct [ini\\_parse\\_string\\_ctx](#)

### Macros

- #define [ini\\_malloc](#) malloc
- #define [ini\\_free](#) free
- #define [ini\\_realloc](#) realloc
- #define [MAX\\_SECTION](#) 50
- #define [MAX\\_NAME](#) 50
- #define [HANDLER](#)(u, s, n, v) [handler](#)(u, s, n, v)

### Functions

- static char \* [rstrip](#) (char \*s)
- static char \* [lskip](#) (const char \*s)
- static char \* [find\\_chars\\_or\\_comment](#) (const char \*s, const char \*chars)
- static char \* [strncpy0](#) (char \*dest, const char \*src, size\_t size)
- int [ini\\_parse\\_stream](#) (ini\_reader reader, void \*stream, [ini\\_handler handler](#), void \*user)
- int [ini\\_parse\\_file](#) (FILE \*file, [ini\\_handler handler](#), void \*user)
- int [ini\\_parse](#) (const char \*filename, [ini\\_handler handler](#), void \*user)
- static char \* [ini\\_reader\\_string](#) (char \*str, int num, void \*stream)
- int [ini\\_parse\\_string](#) (const char \*string, [ini\\_handler handler](#), void \*user)

#### 11.27.1 Macro Definition Documentation

11.27.1.1 #define [ini\\_malloc](#) malloc

11.27.1.2 #define [ini\\_free](#) free

11.27.1.3 #define [ini\\_realloc](#) realloc

11.27.1.4 #define [MAX\\_SECTION](#) 50

11.27.1.5 #define [MAX\\_NAME](#) 50

11.27.1.6 #define [HANDLER](#)( u, s, n, v ) [handler](#)(u, s, n, v)

#### 11.27.2 Function Documentation

11.27.2.1 static char\* [rstrip](#) ( char \* s ) [static]

- 11.27.2.2 `static char* lskip ( const char * s ) [static]`
- 11.27.2.3 `static char* find_chars_or_comment ( const char * s, const char * chars ) [static]`
- 11.27.2.4 `static char* strncpy0 ( char * dest, const char * src, size_t size ) [static]`
- 11.27.2.5 `int ini_parse_stream ( ini_reader reader, void * stream, ini_handler handler, void * user )`
- 11.27.2.6 `int ini_parse_file ( FILE * file, ini_handler handler, void * user )`
- 11.27.2.7 `int ini_parse ( const char * filename, ini_handler handler, void * user )`
- 11.27.2.8 `static char* ini_reader_string ( char * str, int num, void * stream ) [static]`
- 11.27.2.9 `int ini_parse_string ( const char * string, ini_handler handler, void * user )`

## 11.28 lib/io.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <io.h>
#include <argp.h>
#include <ini.h>
#include <traj.h>
#include <mapping.h>
```

### Macros

- `#define MATCH(s, n) strcmp(section, s) == 0 && strcmp(name, n) == 0`

### Functions

- void `check_int_string` (const char \*str, int row, char \*fname)
- void `check_int_string_iniFile` (const char \*str, char \*fname, char \*name)
- int `handler` (void \*config, const char \*section, const char \*name, const char \*value)
- `parameters pp_config` (parameters config)
- void `check_empty_file` (FILE \*f, char \*filename)
- int `n_rows` (FILE \*f)
- void `check_empty_rows` (char \*str)
- void `check_arg_errors` (char \*argv[], int argc)
- void `check_float_string` (char \*str, int row, char \*fname)
- void `check_float_string_iniFile` (const char \*str, char \*fname, char \*name)
- int `columns` (char \*string)
- void `print_usage_main` (char \*argv[])
- void `print_help_main` (char \*argv[])
- void `print_help` (char \*argv[])
- void `check_files` (char \*\*pars, char \*\*pars\_names, int n\_pars, char \*argv[])
- void `check_parameters` (int \*pars, char \*\*pars\_names, int n\_pars)
- void `check_optional_parameters` (parameters \*cc)
- void `mandatory_files_present` (arguments \*arguments, char \*argv[])
- void `init_parameters` (parameters \*cc)
- void `read_ParameterFile` (arguments \*arguments, parameters \*cc)

- FILE \* [open\\_file\\_w](#) (char \*filename)
- FILE \* [open\\_file\\_r](#) (char \*filename)
- FILE \* [open\\_file\\_a](#) (char \*filename)
- void [close\\_file](#) (FILE \*fp)

## Variables

- const char \* [argp\\_program\\_bug\\_address](#) = "<raffaele.fiorentini@unitn.it>"

## 11.28.1 Macro Definition Documentation

11.28.1.1 `#define MATCH( s, n ) strcmp(section, s) == 0 && strcmp(name, n) == 0`

## 11.28.2 Function Documentation

11.28.2.1 `void check_int_string ( const char * str, int row, char * fname )`

routine that checks if the string token in account reading a generic FILE is an INTEGER number

### Parameters

*str* : string token in account

*row* : number of row where the string is found.

*fname* : filename read

11.28.2.2 `void check_int_string_iniFile ( const char * str, char * fname, char * name )`

routine that checks if the string token in account is an INTEGER number. It works only for ini Files

### Parameters

*str* : string token in account

*fname* : parameter filename

*name* : name of each parameter in the file

11.28.2.3 `int handler ( void * config, const char * section, const char * name, const char * value )`

11.28.2.4 `parameters pp_config ( parameters config )`

11.28.2.5 `void check_empty_file ( FILE * f, char * filename )`

routine that checks if the file required exists. If it is the case, check if it is empty or not.

### Parameters

*f* : FILE structure that represents the file opened.

*filename* : filename read

**11.28.2.6 int n\_rows ( FILE \* *f* )**

routine that returns the number of rows in a file. It counts correctly this number even if the last row does not present

**Parameter**

*f* : FILE structure that represents the file opened.

**11.28.2.7 void check\_empty\_rows ( char \* *str* )**

routine that checks if a generic line is empty or not

**Parameter**

*str* : string token in account

**11.28.2.8 void check\_argv\_errors ( char \* *argv*[], int *argc* )**

routine that checks the correctness of command line arguments

**Parameter**

*argv*[] : array of command line arguments

*argc* : number of command line arguments

**11.28.2.9 void check\_float\_string ( char \* *str*, int *row*, char \* *fname* )**

routine that checks if the string token in account reading a generic FILE is an Float number

**Parameters**

*str* : string token in account

*row* : number of row where the string is found.

*fname* : filename read

**11.28.2.10 void check\_float\_string\_iniFile ( const char \* *str*, char \* *fname*, char \* *name* )**

routine that checks if the string token in account is an Float number. It works only for ini Files

**Parameters**

*str* : string token in account

*fname* : parameter filename

*name* : name of each parameter in the file

**11.28.2.11 int columns ( char \* *string* )**

routine that returns the number of columns for each row inside the file chosen.

**Parameter**

`string` : string token in account

**11.28.2.12 void print\_usage\_main ( char \* *argv*[] )**

routine that prints the usage of the program

**Parameter**

`argv[]` : array of command line arguments

**11.28.2.13 void print\_help\_main ( char \* *argv*[] )**

routine that prints detailed information about the program

**Parameter**

`argv[]` : array of command line arguments

**11.28.2.14 void print\_help ( char \* *argv*[] )**

routine that prints some help

**Parameter**

`argv[]` : array of command line arguments

**11.28.2.15 void check\_files ( char \*\* *pars*, char \*\* *pars\_names*, int *n\_pars*, char \* *argv*[] )**

routine that checks if all command line arguments are correctly provided

**Parameter**

`pars` : parameters

`pars_names` : names of parameters

`n_pars` : number of parameters

`argv[]` : array of command line arguments

**11.28.2.16 void check\_parameters ( int \* *pars*, char \*\* *pars\_names*, int *n\_pars* )**

routine that checks if all mandatory parameters are correctly provided

**Parameter**

`pars` : parameters

`pars_names` : names of parameters

`n_pars` : number of parameters

**11.28.2.17** void check\_optional\_parameters ( parameters \* cc )

routine that checks optional parameters for the tasks that need them

#### Parameters

cc : parameters

**11.28.2.18** void mandatory\_files\_present ( arguments \* arguments, char \* argv[] )

routine that checks if the mandatory files are present

#### Parameters

arguments : command line arguments

argv[] : array of command line arguments

**11.28.2.19** void init\_parameters ( parameters \* cc )

routine that initialises the parameters

#### Parameters

cc : parameters object

**11.28.2.20** void read\_ParameterFile ( arguments \* arguments, parameters \* cc )

routine that reads the input parameter file

#### Parameter

ParameterFileName : parameter filename

**11.28.2.21** FILE\* open\_file\_w ( char \* filename )

routine that opens a file in write mode

**11.28.2.22** FILE\* open\_file\_r ( char \* filename )

routine that opens a file in read mode

**11.28.2.23** FILE\* open\_file\_a ( char \* filename )

routine that opens a file in append mode

**11.28.2.24** void close\_file ( FILE \* fp )

routine that closes a file

### 11.28.3 Variable Documentation

11.28.3.1 `const char* argp_program_bug_address = "<raffaele.fiorentini@unitn.it>"`

## 11.29 lib/mapping.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <mapping.h>
#include <io.h>
#include <my_malloc.h>
```

### Functions

- void `free_mapping` (`cg_mapping` \*mapping)
- void `convert_mapping` (`cg_mapping` \*mapping, FILE \*f\_out)
- void `generate_random_mapping` (`cg_mapping` \*mapping, FILE \*f\_out)
- void `update_mapping` (`cg_mapping` \*curr\_mapping, `cg_mapping` \*old\_mapping, int frames)
- void `read_MappingFile` (char \*MappingFileName, FILE \*f\_out\_l, `cg_mapping` \*mapping)
- void `read_mapping_matrix` (char \*mappings\_filename, FILE \*f\_out\_l, `cg_mapping` \*mapping\_matrix[], int nmaps)

### 11.29.1 Function Documentation

11.29.1.1 void `free_mapping` ( `cg_mapping` \* *mapping* )

routine that frees the mapping

#### Parameters

*mapping* : `cg_mapping` object

11.29.1.2 void `convert_mapping` ( `cg_mapping` \* *mapping*, FILE \* *f\_out* )

routine that prints out the mapping

#### Parameters

*mapping* : `cg_mapping` object

*f\_out* : file to write on

11.29.1.3 void `generate_random_mapping` ( `cg_mapping` \* *mapping*, FILE \* *f\_out* )

routine that generates a random mapping

#### Parameters

*mapping* : `cg_mapping` object

*f\_out* : file to write on

11.29.1.4 void update\_mapping ( *cg\_mapping* \* *curr\_mapping*, *cg\_mapping* \* *old\_mapping*, int *frames* )

routine that updates *old\_mapping* with the data contained in *curr\_mapping*

#### Parameters

*curr\_mapping* : current *cg\_mapping* object

*old\_mapping* : *cg\_mapping* object to be updated

*frames* : length of the MD trajectory

11.29.1.5 void read\_MappingFile ( char \* *MappingFileName*, FILE \* *f\_out\_l*, *cg\_mapping* \* *mapping* )

routine that reads the input mapping file

#### Parameters

*MappingFileName* : mapping filename

*f\_out\_l* : output filename

*cg\_mapping* : *cg\_mapping* object

11.29.1.6 void read\_mapping\_matrix ( char \* *mappings\_filename*, FILE \* *f\_out\_l*, *cg\_mapping* \* *mapping\_matrix*[], int *nmaps* )

routine that reads the input mapping matrix

#### Parameters

*filename* : mapping filename

*f\_out\_l* : output filename

*cg\_mapping* : *cg\_mapping* object routine that reads the input mapping matrix

#### Parameters

*filename* : mapping filename

*f\_out\_l* : output filename

*cg\_mapping* : *cg\_mapping* object

*nmaps* : number of mappings defined in parameter file

## 11.30 lib/measure.c File Reference

```
#include <io.h>
```



```
#include <stdlib.h>
#include <my_malloc.h>
#include <traj.h>
#include <hierarchical_clustering.h>
#include <alignment.h>
#include <mapping.h>
#include <sampling.h>
#include <math.h>
#include <geometry.h>
#include <observables.h>
```

## Functions

- void `measure` (`arguments` \*arguments, `parameters` \*cc)

### 11.30.1 Function Documentation

11.30.1.1 void `measure` ( `arguments` \* *arguments*, `parameters` \* *cc* )

subprogram to measure the mapping entropy of a mapping provided by the user

#### Parameters

`arguments`: `arguments` object (command line arguments)

`parameters`: `parameters` object

## 11.31 lib/measure\_kl.c File Reference

```
#include <io.h>
#include <stdlib.h>
#include <my_malloc.h>
#include <traj.h>
#include <hierarchical_clustering.h>
#include <alignment.h>
#include <mapping.h>
#include <sampling.h>
#include <math.h>
#include <geometry.h>
#include <observables.h>
```

## Functions

- void `measure_kl` (`arguments` \*arguments, `parameters` \*cc)

### 11.31.1 Function Documentation

11.31.1.1 void `measure_kl` ( `arguments` \* *arguments*, `parameters` \* *cc* )

subprogram to measure the KL divergence version of the mapping entropy for a mapping provided by the user

**Parameters**

`arguments`: `arguments` object (command line arguments)

`parameters`: `parameters` object

**11.32 lib/my\_malloc.c File Reference**

```
#include <stdio.h>
#include <stdlib.h>
```

**Functions**

- void [failed](#) (char message[])
- FILE \*\* [F2t](#) (int n1)
- FILE \*\*\* [F3t](#) (int n1, int n2)
- char \* [c1t](#) (int n1)
- char \*\* [c2t](#) (int n1, int n2)
- char \*\*\* [c3t](#) (int n1, int n2, int n3)
- char \*\*\*\* [c4t](#) (int n1, int n2, int n3, int n4)
- short \* [s1t](#) (int n1)
- short \*\* [s2t](#) (int n1, int n2)
- short \*\*\* [s3t](#) (int n1, int n2, int n3)
- short \*\*\*\* [s4t](#) (int n1, int n2, int n3, int n4)
- int \* [i1t](#) (int n1)
- int \*\* [i2t](#) (int n1, int n2)
- int \*\*\* [i3t](#) (int n1, int n2, int n3)
- int \*\*\*\* [i4t](#) (int n1, int n2, int n3, int n4)
- float \* [f1t](#) (int n1)
- float \*\* [f2t](#) (int n1, int n2)
- float \*\*\* [f3t](#) (int n1, int n2, int n3)
- float \*\*\*\* [f4t](#) (int n1, int n2, int n3, int n4)
- float \*\*\*\*\* [f5t](#) (int n1, int n2, int n3, int n4, int n5)
- double \* [d1t](#) (int n1)
- double \*\* [d2t](#) (int n1, int n2)
- double \*\*\* [d3t](#) (int n1, int n2, int n3)
- double \*\*\*\* [d4t](#) (int n1, int n2, int n3, int n4)
- void [readeol](#) (FILE \*fp)
- void [pdarray](#) (int n, int m, double \*\*a)
- void [pdvector](#) (int n, double \*a)
- void [pfarray](#) (int n, int m, float \*\*a)
- void [pfvector](#) (int n, float \*a)
- void [piarray](#) (int n, int m, int \*\*a)
- void [pivector](#) (int n, int \*a)
- void [zdarray](#) (int n, int m, double \*\*a)
- void [zdvector](#) (int n, double \*a)
- void [zfarray](#) (int n, int m, float \*\*a)
- void [zfvector](#) (int n, float \*a)
- void [ziarray](#) (int n, int m, int \*\*a)
- void [zivector](#) (int n, int \*a)
- void [free\\_c4t](#) (char \*\*\*\*p)
- void [free\\_c3t](#) (char \*\*\*p)

- void [free\\_c2t](#) (char \*\*p)
- void [free\\_c1t](#) (char \*p)
- void [free\\_s4t](#) (short \*\*\*\*p)
- void [free\\_s3t](#) (short \*\*\*p)
- void [free\\_s2t](#) (short \*\*p)
- void [free\\_s1t](#) (short \*p)
- void [free\\_i4t](#) (int \*\*\*\*p)
- void [free\\_i3t](#) (int \*\*\*p)
- void [free\\_i2t](#) (int \*\*p)
- void [free\\_i1t](#) (int \*p)
- void [free\\_f5t](#) (float \*\*\*\*\*p)
- void [free\\_f4t](#) (float \*\*\*\*p)
- void [free\\_f3t](#) (float \*\*\*p)
- void [free\\_f2t](#) (float \*\*p)
- void [free\\_f1t](#) (float \*p)
- void [free\\_d4t](#) (double \*\*\*\*p)
- void [free\\_d3t](#) (double \*\*\*p)
- void [free\\_d2t](#) (double \*\*p)
- void [free\\_d1t](#) (double \*p)

### 11.32.1 Function Documentation

11.32.1.1 void [failed](#) ( char *message*[ ] )

11.32.1.2 FILE\*\* [F2t](#) ( int *n1* )

11.32.1.3 FILE\*\*\* [F3t](#) ( int *n1*, int *n2* )

11.32.1.4 char\* [c1t](#) ( int *n1* )

11.32.1.5 char\*\* [c2t](#) ( int *n1*, int *n2* )

11.32.1.6 char\*\*\* [c3t](#) ( int *n1*, int *n2*, int *n3* )

11.32.1.7 char\*\*\*\* [c4t](#) ( int *n1*, int *n2*, int *n3*, int *n4* )

11.32.1.8 short\* [s1t](#) ( int *n1* )

11.32.1.9 short\*\* [s2t](#) ( int *n1*, int *n2* )

11.32.1.10 short\*\*\* [s3t](#) ( int *n1*, int *n2*, int *n3* )

11.32.1.11 short\*\*\*\* [s4t](#) ( int *n1*, int *n2*, int *n3*, int *n4* )

11.32.1.12 int\* [i1t](#) ( int *n1* )

11.32.1.13 int\*\* [i2t](#) ( int *n1*, int *n2* )

11.32.1.14 int\*\*\* [i3t](#) ( int *n1*, int *n2*, int *n3* )

11.32.1.15 int\*\*\*\* [i4t](#) ( int *n1*, int *n2*, int *n3*, int *n4* )

11.32.1.16 float\* [f1t](#) ( int *n1* )

11.32.1.17 float\*\* [f2t](#) ( int *n1*, int *n2* )

11.32.1.18 `float*** f3t ( int n1, int n2, int n3 )`

11.32.1.19 `float**** f4t ( int n1, int n2, int n3, int n4 )`

11.32.1.20 `float***** f5t ( int n1, int n2, int n3, int n4, int n5 )`

11.32.1.21 `double* d1t ( int n1 )`

11.32.1.22 `double** d2t ( int n1, int n2 )`

11.32.1.23 `double*** d3t ( int n1, int n2, int n3 )`

11.32.1.24 `double**** d4t ( int n1, int n2, int n3, int n4 )`

11.32.1.25 `void readeol ( FILE * fp )`

11.32.1.26 `void pddarray ( int n, int m, double ** a )`

11.32.1.27 `void pdvector ( int n, double * a )`

11.32.1.28 `void pfarray ( int n, int m, float ** a )`

11.32.1.29 `void pfvector ( int n, float * a )`

11.32.1.30 `void piarray ( int n, int m, int ** a )`

11.32.1.31 `void pivector ( int n, int * a )`

11.32.1.32 `void zdarray ( int n, int m, double ** a )`

11.32.1.33 `void zdvector ( int n, double * a )`

11.32.1.34 `void zfarray ( int n, int m, float ** a )`

11.32.1.35 `void zfvector ( int n, float * a )`

11.32.1.36 `void ziarray ( int n, int m, int ** a )`

11.32.1.37 `void zivector ( int n, int * a )`

11.32.1.38 `void free_c4t ( char **** p )`

11.32.1.39 `void free_c3t ( char *** p )`

11.32.1.40 `void free_c2t ( char ** p )`

11.32.1.41 `void free_c1t ( char * p )`

11.32.1.42 `void free_s4t ( short ***** p )`

11.32.1.43 `void free_s3t ( short **** p )`

11.32.1.44 `void free_s2t ( short ** p )`

11.32.1.45 `void free_s1t ( short * p )`

11.32.1.46 void free\_i4t ( int \*\*\*\* *p* )

11.32.1.47 void free\_i3t ( int \*\*\* *p* )

11.32.1.48 void free\_i2t ( int \*\* *p* )

11.32.1.49 void free\_i1t ( int \* *p* )

11.32.1.50 void free\_f5t ( float \*\*\*\*\* *p* )

11.32.1.51 void free\_f4t ( float \*\*\*\* *p* )

11.32.1.52 void free\_f3t ( float \*\*\* *p* )

11.32.1.53 void free\_f2t ( float \*\* *p* )

11.32.1.54 void free\_f1t ( float \* *p* )

11.32.1.55 void free\_d4t ( double \*\*\*\* *p* )

11.32.1.56 void free\_d3t ( double \*\*\* *p* )

11.32.1.57 void free\_d2t ( double \*\* *p* )

11.32.1.58 void free\_d1t ( double \* *p* )

## 11.33 lib/norm.c File Reference

```
#include <io.h>
#include <stdlib.h>
#include <my_malloc.h>
#include <traj.h>
#include <mapping.h>
#include <math.h>
#include <geometry.h>
#include <observables.h>
```

### Functions

- void [norm](#) ([arguments](#) \**arguments*, [parameters](#) \**cc*)

#### 11.33.1 Function Documentation

11.33.1.1 void [norm](#) ( [arguments](#) \* *arguments*, [parameters](#) \* *cc* )

subprogram to calculate the norm of a mapping (provided by the user) throughout a trajectory

#### Parameters

[arguments](#) : [arguments](#) object (command line arguments)

[parameters](#) : [parameters](#) object

## 11.34 lib/observables.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <my_malloc.h>
#include <observables.h>
#include <io.h>
#include <time.h>
#include <geometry.h>
#include <alignment.h>
```

### Functions

- void [compute\\_coupling\\_matrix](#) (double \*coupling\_mat, [traj](#) \*Trajectory, int fr\_id, float sigma)
- double [compute\\_atomistic\\_coord\\_number](#) (double \*coupling\_mat, [traj](#) \*Trajectory, FILE \*f\_out\_l)
- void [compute\\_norm](#) ([cg\\_mapping](#) \*mapping, double \*coupling\_mat, double n\_coord\_at, int fr\_id, FILE \*f\_out\_l)
- double [compute\\_distance](#) ([cg\\_mapping](#) \*mapping, [cg\\_mapping](#) \*mapping\_prime, double \*coupling\_mat, double n\_coord\_at, int fr\_id, FILE \*f\_out\_l)
- void [compute\\_mapping\\_norms](#) ([traj](#) \*Trajectory, [cg\\_mapping](#) \*mapping, FILE \*f\_out\_l)
- void [compute\\_mapping\\_distances](#) ([traj](#) \*Trajectory, [cg\\_mapping](#) \*mapping, [cg\\_mapping](#) \*mapping\_prime, FILE \*f\_out\_l)
- void [compute\\_mapping\\_distmat](#) ([traj](#) \*Trajectory, [cg\\_mapping](#) \*mapping\_matrix[], int nmaps, FILE \*f\_out\_l, char \*distmat\_filename)
- void [compute\\_variances](#) (int nclust, double \*variances, int \*cluster\_list, int \*cluster\_list\_idx, double \*energies)
- void [compute\\_pR](#) (int nclust, double \*p\_R, int \*cluster\_list, int \*cluster\_list\_idx, double \*energies)
- double [get\\_smap](#) (int frames, int curr\_nclust, int \*clusters, double \*energies)
- double [get\\_kl](#) (int frames, int curr\_nclust, int \*clusters, double \*energies)
- void [compute\\_smap\\_spins](#) ([spin\\_traj](#) \*Trajectory, [cg\\_mapping](#) \*mapping)
- void [overall\\_compute\\_smap](#) ([alignments](#) \*align, [clust\\_params](#) \*clustering, [traj](#) \*Trajectory, [cg\\_mapping](#) \*mapping, int verbose, int kl\_flag)

### 11.34.1 Function Documentation

#### 11.34.1.1 void compute\_coupling\_matrix ( double \* coupling\_mat, *traj* \* Trajectory, int fr\_id, float sigma )

routine that computes the coupling matrix over a frame

*coupling\_mat* : coupling matrix

*Trajectory* : *traj* object

*fr\_id* : frame ID

*sigma* : sigma parameter

#### 11.34.1.2 double compute\_atomistic\_coord\_number ( double \* coupling\_mat, *traj* \* Trajectory, FILE \* f\_out\_l )

routine that computes the atomistic coordination number for a certain coupling matrix. Double counting is necessary to ensure proper normalisation to norm and scalar product

*coupling\_mat* : coupling matrix

*Trajectory* : *traj* object

*f\_out\_l* : output filename

**11.34.1.3** void compute\_norm ( cg\_mapping \* mapping, double \* coupling\_mat, double n\_coord\_at, int fr\_id, FILE \* f\_out\_l )

routine that computes the norm of a mapping over a frame of a trajectory

#### Parameters

mapping : [cg\\_mapping](#) object

coupling\_mat : coupling matrix

n\_coord\_at : atomistic coordination number

fr\_id : frame index

f\_out\_l : output filename

**11.34.1.4** double compute\_distance ( cg\_mapping \* mapping, cg\_mapping \* mapping\_prime, double \* coupling\_mat, double n\_coord\_at, int fr\_id, FILE \* f\_out\_l )

routine that computes the distance and cosine between a pair of cg mappings

#### Parameters

mapping, mapping\_prime : [cg\\_mapping](#) objects

coupling\_mat : coupling matrix

n\_coord\_at : atomistic coordination number

fr\_id : frame index

f\_out\_l : output filename

**11.34.1.5** void compute\_mapping\_norms ( traj \* Trajectory, cg\_mapping \* mapping, FILE \* f\_out\_l )

routine that computes the norm of a mapping over a MD trajectory

#### Parameters

Trajectory : traj object

mapping : [cg\\_mapping](#) object

f\_out\_l : output filename

**11.34.1.6** void compute\_mapping\_distances ( traj \* Trajectory, cg\_mapping \* mapping, cg\_mapping \* mapping\_prime, FILE \* f\_out\_l )

routine that computes the distances and cosines between two mappings provided by the user over a MD trajectory

#### Parameters

Trajectory : traj object

mapping, mapping\_prime : [cg\\_mapping](#) objects

f\_out\_l : output filename

**11.34.1.7** `void compute_mapping_distmat ( traj * Trajectory, cg_mapping * mapping_matrix[], int nmaps, FILE * f_out_l, char * distmat_filename )`

routine that computes the distance matrix between a set of mappings over a single structure

#### Parameters

*Trajectory* : traj object  
*mappings\_filename* : filename with the chosen mappings  
*namps* : number of mappings  
*f\_out\_l* : output filename

**11.34.1.8** `void compute_variances ( int nclust, double * variances, int * cluster_list, int * cluster_list_idx, double * energies )`

routine that computes the variance of the energies

#### Parameters

*nclust* : number of macrostates  
*variances* : vector of variances  
*cluster\_list* : list of cluster IDs  
*cluster\_list\_idx* : list of cluster indices  
*energies* : array of energies

**11.34.1.9** `void compute_pR ( int nclust, double * p_R, int * cluster_list, int * cluster_list_idx, double * energies )`

routine that computes the variance of the energies

#### Parameters

*nclust* : number of macrostates  
*variances* : vector of variances  
*cluster\_list* : list of cluster IDs  
*cluster\_list\_idx* : list of cluster indices  
*energies* : array of energies

**11.34.1.10** `double get_smap ( int frames, int curr_nclust, int * clusters, double * energies )`

routine that computes the observable given the current *nclust* and the current *clusters*

#### Parameters

*frames* : number of frames  
*curr\_nclust* : current index of CG macrostate  
*clusters* : list of cluster IDs  
*energies* : array of energies



11.34.1.11 `double get_kl ( int frames, int curr_nclust, int * clusters, double * energies )`

routine that computes the observable given the current `nclust` and the current `clusters`

#### Parameters

`frames` : number of frames

`curr_nclust` : current index of CG macrostate

`clusters` : list of cluster IDs

`energies` : array of energies

11.34.1.12 `void compute_smap_spins ( spin_traj * Trajectory, cg_mapping * mapping )`

11.34.1.13 `void overall_compute_smap ( alignments * align, clust_params * clustering, traj * Trajectory, cg_mapping * mapping, int verbose, int kl_flag )`

routine that calls `get_smap` with the correct parameters

#### Parameters

`rmsd_mat` : condensed matrix of pairwise RMSDs

`clustering` : `clust_params` object

`Trajectory` : traj object

`mapping` : `cg_mapping` object

`verbose` : tunes the level of verbosity

`f_out` : output filename

## 11.35 lib/optimize.c File Reference

```
#include <io.h>
#include <stdlib.h>
#include <my_malloc.h>
#include <traj.h>
#include <hierarchical_clustering.h>
#include <alignment.h>
#include <mapping.h>
#include <sampling.h>
#include <math.h>
#include <omp.h>
```

#### Functions

- void `optimize` (`arguments` \*arguments, `parameters` \*cc)

### 11.35.1 Function Documentation

### 11.35.1.1 void optimize ( arguments \* arguments, parameters \* cc )

subprogram to optimize the coarse-grained mapping by minimising its mapping entropy

#### Parameters

arguments : arguments object (command line arguments)

parameters : parameters object

## 11.36 lib/optimize\_kl.c File Reference

```
#include <io.h>
#include <stdlib.h>
#include <my_malloc.h>
#include <traj.h>
#include <hierarchical_clustering.h>
#include <alignment.h>
#include <mapping.h>
#include <sampling.h>
#include <math.h>
#include <omp.h>
```

#### Functions

- void [optimize\\_kl](#) (arguments \*arguments, parameters \*cc)

### 11.36.1 Function Documentation

#### 11.36.1.1 void optimize\_kl ( arguments \* arguments, parameters \* cc )

subprogram to optimize the coarse-grained mapping by minimising the KL divergence version of its mapping entropy

#### Parameters

arguments : arguments object (command line arguments)

parameters : parameters object

## 11.37 lib/optimize\_spins.c File Reference

```
#include <io.h>
#include <stdio.h>
#include <stdlib.h>
#include <my_malloc.h>
#include <traj.h>
#include <hierarchical_clustering.h>
#include <alignment.h>
#include <mapping.h>
#include <sampling.h>
#include <math.h>
#include <omp.h>
#include <stdbool.h>
```

### Functions

- int [spin\\_clustering](#) (int \*\*traj\_coords, int \*mapping, int atomnum, int frames, int \*mapping\_clusters, int cnum)
- void [optimize\\_spins](#) (arguments \*arguments, parameters \*cc)

### 11.37.1 Function Documentation

11.37.1.1 int [spin\\_clustering](#) ( int \*\* *traj\_coords*, int \* *mapping*, int *atomnum*, int *frames*, int \* *mapping\_clusters*, int *cnum* )

spin clustering

11.37.1.2 void [optimize\\_spins](#) ( arguments \* *arguments*, parameters \* *cc* )

subprogram to optimize the coarse-grained mapping by minimising the KL divergence version of its mapping entropy

### Parameters

arguments : arguments object (command line arguments)

parameters : parameters object

## 11.38 lib/random\_mapping.c File Reference

```
#include <io.h>
#include <stdlib.h>
#include <my_malloc.h>
#include <traj.h>
#include <hierarchical_clustering.h>
#include <alignment.h>
#include <mapping.h>
#include <sampling.h>
#include <math.h>
#include <geometry.h>
#include <observables.h>
```

## Functions

- void `random_mapping` (`arguments *arguments`, `parameters *cc`)

### 11.38.1 Function Documentation

11.38.1.1 void `random_mapping` ( `arguments * arguments`, `parameters * cc` )

subprogram to randomly generate coarse-grained representations and measure the associated mapping entropies

#### Parameters

`arguments` : `arguments` object (command line arguments)

`parameters` : `parameters` object

## 11.39 lib/random\_mapping\_kl.c File Reference

```
#include <io.h>
#include <stdlib.h>
#include <my_malloc.h>
#include <traj.h>
#include <hierarchical_clustering.h>
#include <alignment.h>
#include <mapping.h>
#include <sampling.h>
#include <math.h>
#include <geometry.h>
#include <observables.h>
```

## Functions

- void `random_mapping_kl` (`arguments *arguments`, `parameters *cc`)

### 11.39.1 Function Documentation

11.39.1.1 void `random_mapping_kl` ( `arguments * arguments`, `parameters * cc` )

subprogram to randomly generate coarse-grained representations and measure the KL divergence version of their mapping entropy

#### Parameters

`arguments` : `arguments` object (command line arguments)

`parameters` : `parameters` object

## 11.40 lib/sampling.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <my_malloc.h>
#include <math.h>
#include <sampling.h>
#include <alignment.h>
#include <mapping.h>
#include <observables.h>
```

### Functions

- void [my\\_make\\_a\\_move](#) ([cg\\_mapping](#) \*old\_mapping, [cg\\_mapping](#) \*new\_mapping, int rem\_add[2])
- void [accept\\_move](#) (int rem\_add[2], [cg\\_mapping](#) \*mapping, [cg\\_mapping](#) \*new\_mapping, [alignments](#) \*align, [alignments](#) \*new\_align, [traj](#) \*Trajectory, [clust\\_params](#) \*clustering, double \*\*new\_coefficients\_matrix)
- void [accept\\_move\\_spins](#) (int rem\_add[2], [cg\\_mapping](#) \*mapping, [cg\\_mapping](#) \*new\_mapping)
- double [tzero\\_estimation\\_spins](#) ([spin\\_traj](#) \*Trajectory, int cignum, FILE \*f\_out\_l)
- double [tzero\\_estimation](#) ([traj](#) \*Trajectory, [clust\\_params](#) \*clustering, int cignum, int rsd, int verbose, int kl\_flag, FILE \*f\_out\_l)
- void [simulated\\_annealing](#) ([traj](#) \*Trajectory, [clust\\_params](#) \*clustering, [MC\\_params](#) \*SA\_params, int cignum, int rsd, int verbose, int kl\_flag, FILE \*f\_out\_l)
- void [simulated\\_annealing\\_spins](#) ([spin\\_traj](#) \*Trajectory, [MC\\_params](#) \*SA\_params, int cignum, int verbose, FILE \*f\_out\_l)

### 11.40.1 Function Documentation

11.40.1.1 void [my\\_make\\_a\\_move](#) ( [cg\\_mapping](#) \* *old\_mapping*, [cg\\_mapping](#) \* *new\_mapping*, int *rem\_add*[2] )

function that swaps two atoms inside a CG mapping

#### Parameters

*old\_mapping* : [cg\\_mapping](#) object

*new\_mapping* : [cg\\_mapping](#) object

*rem\_add* : vector of length 2 containing the removed and added atom index

11.40.1.2 void [accept\\_move](#) ( int *rem\_add*[2], [cg\\_mapping](#) \* *mapping*, [cg\\_mapping](#) \* *new\_mapping*, [alignments](#) \* *align*, [alignments](#) \* *new\_align*, [traj](#) \* *Trajectory*, [clust\\_params](#) \* *clustering*, double \*\* *new\_coefficients\_matrix* )

routine that accepts a Simulated Annealing move. It updates all the relevant observables.

#### Parameters

*rem\_add* : vector of length 2 containing the removed and added atom index

*alignments* : align object

*mapping* : [cg\\_mapping](#) object

*verbose* : tunes the level of verbosity

11.40.1.3 `void accept_move_spins ( int rem_add[2], cg_mapping * mapping, cg_mapping * new_mapping )`

routine that accepts a Simulated Annealing Spins move. It updates all the relevant observables.

#### Parameters

rem\_add : vector of length 2 containing the removed and added atom index

mapping : [cg\\_mapping](#) object

new\_mapping : [cg\\_mapping](#) object

11.40.1.4 `double tzero_estimation_spins ( spin_traj * Trajectory, int cgnum, FILE * f_out_l )`

routine that makes *nrun* unbiased moves. It is used to estimate the starting temperature for Simulated Annealing.

#### Parameters

Trajectory : traj object

alignments : align object

mapping : [cg\\_mapping](#) object

verbose : tunes the level of verbosity

11.40.1.5 `double tzero_estimation ( traj * Trajectory, clust_params * clustering, int cgnum, int rsd, int verbose, int kl_flag, FILE * f_out_l )`

routine that makes *nrun* unbiased moves. It is used to estimate the starting temperature for Simulated Annealing.

#### Parameters

Trajectory : traj object

alignments : align object

mapping : [cg\\_mapping](#) object

verbose : tunes the level of verbosity

11.40.1.6 `void simulated_annealing ( traj * Trajectory, clust_params * clustering, MC_params * SA_params, int cgnum, int rsd, int verbose, int kl_flag, FILE * f_out_l )`

simulated annealing optimisation

#### Parameters

Trajectory : traj object

alignments : align object

mapping : [cg\\_mapping](#) object

SA\_params : set of Monte Carlo parameters

verbose : tunes the level of verbosity

f\_out\_l : output filename

11.40.1.7 void simulated\_annealing\_spins ( spin\_traj \* *Trajectory*, MC\_params \* *SA\_params*, int *cgnum*, int *verbose*, FILE \* *f\_out\_l* )

simulated annealing optimisation of a spin system

#### Parameters

Trajectory : traj object

mapping : [cg\\_mapping](#) object

SA\_params : set of Monte Carlo parameters

verbose : tunes the level of verbosity

f\_out\_l : output filename

## 11.41 lib/traj.c File Reference

```
#include <traj.h>
#include <stdio.h>
#include <io.h>
#include <stdlib.h>
#include <ini.h>
```

### Functions

- int [check\\_probabilities](#) (double \*probabilities, int prob\_length)
- void [read\\_EnergyFile](#) (char \*EnergyFileName, [traj](#) \*Trajectory)
- void [read\\_spinFile](#) (char \*TrajFileName, [spin\\_traj](#) \*Trajectory)
- void [read\\_TrajectoryFile](#) (char \*TrajFileName, [traj](#) \*Trajectory)

#### 11.41.1 Function Documentation

11.41.1.1 int check\_probabilities ( double \* *probabilities*, int *prob\_length* )

routine that checks that input probabilities sum to 1

#### Parameters

probabilities : array of probabilities

prob\_length : array length

11.41.1.2 void read\_EnergyFile ( char \* *EnergyFileName*, [traj](#) \* *Trajectory* )

routine that reads the input energy file

#### Parameters

EnergyFileName : energies filename

Trajectory : traj object

11.41.1.3 void read\_spinFile ( char \* *TrajFileName*, spin\_traj \* *Trajectory* )

routine that reads the input xyz coordinate file

#### Parameters

TrajFileName : trajectory filename

Trajectory : traj object

11.41.1.4 void read\_TrajectoryFile ( char \* *TrajFileName*, traj \* *Trajectory* )

routine that reads the input xyz coordinate file

#### Parameters

TrajFileName : trajectory filename

Trajectory : traj object

## 11.42 python/README.md File Reference

## 11.43 README.md File Reference

## 11.44 tests/README.md File Reference

## 11.45 python/sample\_convert\_xtc\_to\_xyz.py File Reference

### Namespaces

- [sample\\_convert\\_xtc\\_to\\_xyz](#)

### Variables

- tuple [sample\\_convert\\_xtc\\_to\\_xyz.xtc\\_path](#) = input("insert path to XTC file\n")
- tuple [sample\\_convert\\_xtc\\_to\\_xyz.gro\\_path](#) = input("insert path to GRO file\n")
- tuple [sample\\_convert\\_xtc\\_to\\_xyz.xyz\\_filename](#) = input("insert path to output XYZ file\n")
- tuple [sample\\_convert\\_xtc\\_to\\_xyz.full\\_traj](#) = mdtraj.load\_xtc(xtc\_path.strip(),top=gro\_path.strip())
- [sample\\_convert\\_xtc\\_to\\_xyz.full\\_traj\\_topology](#) = full\_traj.topology
- tuple [sample\\_convert\\_xtc\\_to\\_xyz.no\\_h](#) = full\_traj\_topology.select('type != H')
- tuple [sample\\_convert\\_xtc\\_to\\_xyz.n\\_heavy\\_traj](#) = len(no\_h)
- tuple [sample\\_convert\\_xtc\\_to\\_xyz.mdt\\_tr\\_heavy](#) = mdtraj.load\_xtc(xtc\_path, top=gro\_path, atom\_indices = list(no\_h))

## 11.46 python/setup\_parfile.py File Reference

### Namespaces

- [setup\\_parfile](#)



## Functions

- def `setup_parfile.retrieve_parameter`
- def `setup_parfile.get_mandatory_parameters`
- def `setup_parfile.get_optional_parameters`
- def `setup_parfile.write_parameters`

## Variables

- list `setup_parfile.tasks` = ["optimize", "random", "measure", "norm", "cosine", "distance", "optimize\_kl", "measure\_kl"]
- dictionary `setup_parfile.mandatory_pars`
- dictionary `setup_parfile.optional_pars`
- dictionary `setup_parfile.pars_description`
- dictionary `setup_parfile.pars_type`
- dictionary `setup_parfile.clustering_pars`
- tuple `setup_parfile.task` = input(f"Insert the task you would like to perform among the following-  
: {str(tasks)}{os.linesep}")
- dictionary `setup_parfile.my_pars` = {}
- tuple `setup_parfile.opt` = input("Insert optional parameters? (y/n)")

## 11.47 tests/test\_suite.py File Reference

### Classes

- class `test_suite.test0`
- class `test_suite.test1`
- class `test_suite.test2`
- class `test_suite.test3`
- class `test_suite.test4`
- class `test_suite.test5`
- class `test_suite.test6`
- class `test_suite.test7`
- class `test_suite.test8`
- class `test_suite.test9`
- class `test_suite.test10`
- class `test_suite.test11`
- class `test_suite.test12`
- class `test_suite.test13`
- class `test_suite.test14`
- class `test_suite.test15`
- class `test_suite.test16`
- class `test_suite.test17`
- class `test_suite.test18`
- class `test_suite.test19`
- class `test_suite.test20`
- class `test_suite.test21`
- class `test_suite.test22`
- class `test_suite.test23`
- class `test_suite.test24`
- class `test_suite.test25`
- class `test_suite.test26`
- class `test_suite.test27`

## Namespaces

- [test\\_suite](#)

## Variables

- tuple [test\\_suite.t\\_start](#) = dt.datetime.now()
- tuple [test\\_suite.bash\\_script](#) = subprocess.Popen("./test\_suite.sh", shell=True, stdout=subprocess.PIPE)