Capriqorn Documentation

Release 1.0.1

Juergen Koefinger, Klaus Reuter

Sep 17, 2020

Contents

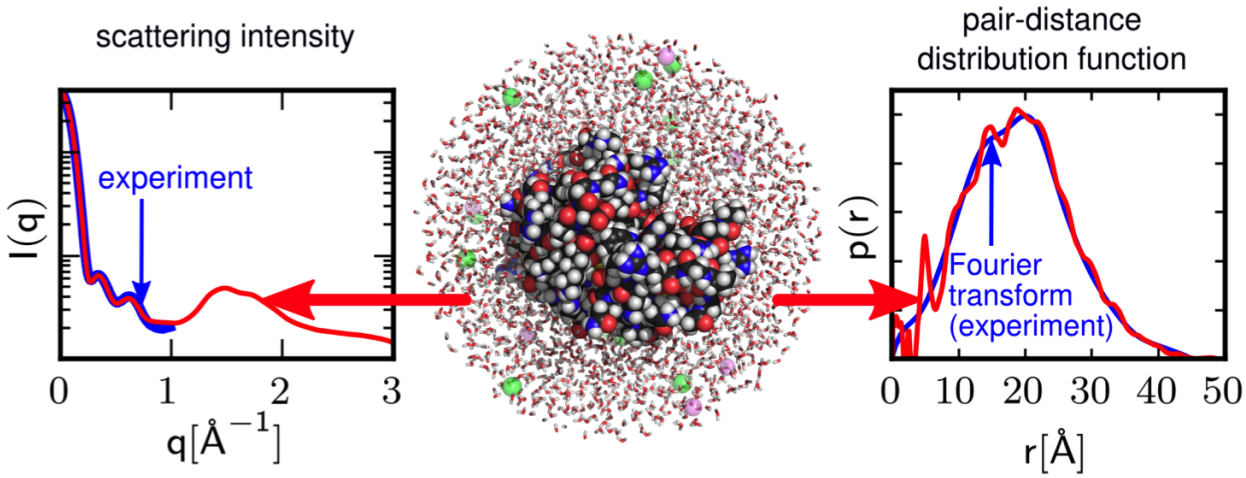

CAPRIQORN

CHAPTER 1

Introduction

CAPRIQORN: **CA**lculation of **P**(**R**) and **I**(**Q**) **O**f mac**R**omolecules in solutio**N**.

Capriqorn is a software suite designed to facilitate the calculation of quantities such as **pair-distance distribution functions** and **scattering intensities** directly from the output trajectories of **molecular dynamics simulations**.



In particular, Capriqorn implements the methods published in the following publication:

Atomic-resolution structural information from scattering experiments on macromolecules in solution
Jürgen Köfinger and Gerhard Hummer
Phys. Rev. E 87, 052712 (2013)

For your convenience, we provide citations in `bibtex` format and `endnote` format.

Additionally, a novel method for **non-spherical observation volumes** using **virtual particles** has been implemented (manuscript in preparation).

Capriqorn is mostly implemented in Python with time-critical kernels accelerated using compiled Cython code. The distance histogram calculation is done via the Cadishi package which offers high-performance parallel implementations for the CPU and the GPU.

CHAPTER 2

Requirements

Capriqorn runs on Linux and OS X platforms. In particular, the following software packages are required.

- Capriqorn requires a Python 2.7 installation including the NumPy, SciPy, Cython, h5py, and PyYAML modules. We have successfully used the Anaconda Python Distribution which provides all these modules out of the box. Moreover, to compile the Cython kernels, a recent GCC installation is required.
- Capriqorn requires the Cadishi Python package to perform fast distance histogram calculations on CPUs and GPUs.
- Capriqorn requires the MDAnalysis Python library, in particular its data reader which supports various file formats from molecular dynamics simulation codes.
- Optionally and for convenience, a HDF5 viewer (such as HDFView or HDF Compass) is useful to be able to quickly inspect the HDF5 files generated by Capriqorn.

CHAPTER 3

Installation

The package comes with a standard Python setup.py installer. It is installed e.g. as follows into the user's homedirectory:

```
python setup.py install --user
```

In this case, setup.py copies the Cadishi files into the directory `~/.local`. Make sure that your `PATH` environment variable contains the directory `~/.local/bin`. To perform a system-wide installation omit the `--user` flag (and prefix the command with `sudo` to gain administrative privileges on a typical Linux system).

3.1 Quick Start

To run the example analysis pipeline that comes with Capriqorn please issue the following sequence of commands:

```
capriq example  # generate example parameter files
capriq preproc  # run preprocessor
capriq histo  # run distance histogram calculation
capriq postproc  # run postprocessor
```

3.2 Basic workflow

Capriqorn implements the concept of a data processing pipeline. A typical workflow consists of three stages.

1. preprocessor run: Read MD trajectory, apply methods, write to single HDF5 file.
2. distance histogram calculation: Read frames from HDF5 file, compute histograms, write to HDF5 file.
3. postprocessor run: Read histograms from HDF5 file, perform computations (PDDF), write to HDF5 file.

The data processing pipeline is implemented using Python generators. Methods (computations, geometry and data modifications, etc.) are considered 'filters'.

CHAPTER 4

Documentation

4.1 Background

Macromolecules in solution

To calculate the **scattering intensities** and **pair-distance distribution functions** of macromolecules in solution from a molecular dynamics trajectory using explicit solvent, we have to account for the finite size and shape of the simulation box to avoid artifacts in the scattering intensity. To do so, we cut out an **observation volume** containing the macromolecule and a sufficiently thick layer of solvent.

Pure solvent simulations

In experiments, the **difference intensity** between the macromolecule in solution and pure solvent is determined. We therefore also have to simulate pure solvent. In our method, we only need the **particle densities** and **partial radial distribution functions** calculated from these pure solvent simulations. This approach is in contrast to other methods, where the same observation volume as it is used for the macromolecule in solution, has to be cut out from the solvent trajectory.

Self-consistent solvent matching

Small differences in the solvent densities and structure at the border of the observation volume can lead to artifacts in the small angle regime. To avoid such artifacts, we match solvent properties of the system containing the macromolecule and pure solvent in a layer at the boundary of the observation volume by properly scaling particle densities and partial radial distribution functions.

4.2 Method Details

Pure solvent

We describe pure solvent by its **particle densities** and **partial radial distribution functions**. These quantities are calculated using **Capriqorn** from a pure solvent simulation in a separate calculation and then provided as input to Capriqorn when analyzing the trajectory of the macromolecule in solution.

Macromolecule in solution

We we can choose various **geometries (observation volumes)** to cut the macromolecule and sufficient solvent out of the simulation box.

Choosing the **geometry** we want to

- increase the signal-to-noise ratio and increase performance by minimizing the amount of bulk solvent while
- including sufficient bulk solvent to avoid finite size effects and to facilitate solvent matching.

Each of the geometries listed below has its own merits. A sphere is the most efficient geometry for globular macromolecules. For all other geometries, we have to use **virtual particles** ("ideal gas" particles) to account for the geometry of the observation volume which comes with additional computational costs.

Overview of **observation volume geometries**:

- **Sphere** A sphere centered at the origin is cut out. No virtual particles are needed.
- **Cuboid and Ellipsoid** Cuboids and ellipsoids are centered at the origin. Their faces/axes are aligned with the coordinate system. The box should be rotated such that the macromolecule is aligned correspondingly (tcl script `orient.tcl` used with VMD).
- **Single reference structure** A single reference structure is used for all frames to cut out particles within a distance of this reference structure. Minimum number of solvent particles are added, resulting in a better signal to noise ratio. Only a subset of the atoms has to be selected in the reference structure, e.g., select only carbon atoms for a protein. Uses virtual particles. Reference structure should be RMSD aligned with trajectory.
- **Multiple reference structures (i.e., a reference trajectory)** A trajectory is providing reference structures for each frame individually to cut out particles within a distance of the reference structures. Usually, the trajectory used for reference is the same trajectory we want to calculate scattering intensities for. Only a subset of the atoms has to be selected in the reference structure, e.g., select only carbon atoms for a protein. Uses virtual particles. No alignment with coordinate system necessary.

4.3 How to use Capriqorn

Example: Hen egg-white lysozyme

We provide example input files and plots of the results at http://ftp.biophys.mpg.de/tbhummer/Capriqorn. We suggest to use it to get started. You can pick the parameter files for the geometry of your choice and adapt them accordingly to your problem at hand. Additionally, you can use *capriq example* to generate commented parameter files.

1. **Prerequisites**
   - **Input trajectories**
     - PDB file (provides atom names)
     - Trajectory file
   - **Mapping of atom names to element names** Different force fields use different atom names. We have to map these names onto the corresponding element names, which determine the form factors for each element. This information is saved in a file we usually call *alias.dat*. This file contains the atom name provided by the force field in the first column and the element name in the second column. The bash script `get_aliases_initial_guess_from_pdb.sh` extracts atom names from a pdb and provides a first guess of the element names. **!!!YOU HAVE TO EDIT THIS FILE BY HAND AND MAKE CORRECTIONS!!!**
2. **Pure solvent**
   - We suggest to use orthorhombic (cubic) boxes of similar size (or larger) as the simulation box used for the macromolecule in solution.

- The force field and composition of the pure solvent should be the same as in the simulation of the macromolecule.

3. **Macromolecule in solution**
   - Preparation of the trajectory

     The macromolecule has to be centered in the box, ideally maximizing the solvent thickness around the macromolecule, i.e., maximizing the minimum distance of atoms of the macromolecule to the box borders:

     – Sphere: Center macromolecule at origin.

     – Cuboid: Center macromolecule at origin and align principal axis with VMD (tcl script `orient.tcl`)

     – Ellipsoid: Center macromolecule at origin and align principal axis with VMD (tcl script `orient.tcl`)

     – Reference: RMSD alignment of the macromolecule with chosen reference structure.

     – MultiReference: When using the same trajectories as input and reference, no alignment is necessary.

     Trajectories can be prepared with VMD (wrapping of the box: http://www.ks.uiuc.edu/Research/vmd/plugins/pbctools/) or if you use Gromacs using *trjconv* (*gmx trjconv* in newer versions Gromacs).

   - **Preprocessing: capriq preproc -f preprocessor.yaml**

     – Run the preprocessor for each trajectory separately. The preprocessor can be run in parallel over a single node. Also note that splitting the trajectory in multiple files facilitates further trivial parallelization of the preprocessor.

   - **Histogram calculation: capriq histo -f histograms.yaml**

     – Multiple trajectory h5-files (preprocessor output) can be read in. We use Cadishi to efficiently calculate histograms on CPUs and/or GPUs.

   - **Postprocessing: capriq postproc -f postprocessor.yaml**

     – Multiple histogram h5-files can be read in at once for postprocessing.

     – The output is stored in an hdf5 file, which can be unpacked using "capriq unpack" such that the output files are available in ASCII format.

4. **Analysis**
   - Reading in hdf5 files with python (template is coming soon!)

4.4 Tips and tricks

- **Use VMD (http://www.ks.uiuc.edu/Research/vmd/) to choose and check geometry of observation volume.**

  – Using selection strings, you can choose representation in VMD which visualize various geometries. Note that the selection string syntax in VMD is different to the one used in Capriqorn (Capriqorn using MD Analysis which uses CHARMM syntax).

  – The preprocessor can write out xyz files which you can visualize using VMD to check that the macromolecule has been cut out correctly.

  – To cite VMD, please visit http://www.ks.uiuc.edu/Research/vmd/allversions/cite.html.

- Capriqorn offers a plethora of methods and modules. See the example parameter files for an overview. The files can be written via the command capriq example [–expert] The *–expert* switch adds additional options which allow to override some default values. Some hints on the parameter choices, the general usage, and the file handling are given in the following.
  - For various reasons Capriqorn uses HDF5 files. To inspect a HDF5 file, use a viewer software or extract the HDF5 file using the Capriqorn command `capriq unpack`.
  - Compression of the HDF5 output datasets using the LZF algorithm is usually beneficial regarding performance and file size. LZF comes with h5py by default. Other installations and tools may lack LZF, so use no compression or gzip compression in case you need to interact with such software. You can use the `capriq merge` tool to change the compression of a file.
- An essential part of the Capriqorn pipeline consists of the distance histogram calculation performed by the Cadishi package. Cadishi offers many parameters which allow to tune and optimize the performance. As a quick start one may try the following configuration via the parameter file:
  - adapt the number of CPU workers to the number of CPU sockets you have in your system;
  - adapt the number of threads per CPU worker to the number of cores you have per socket, however, consider the following point:
  - when choosing the thread numbers reserve one core each for the input and output processes and for the GPU processes (if applicable);
  - pinning the processes to NUMA domains is usually a good idea;
  - example: On a dual socket system with 8 cores per socket and two GPUs one may start with the following configuration: 2 CPU workers, 6 threads per CPU worker, 2 GPU workers.

  By default Cadishi uses a reasonable process and thread configuration.

4.5 Notes

- Efficiency:
  - In the current version of the code, the histogram calculation in Cadishi has been highly optimized. Compared to the histogram calculation, the preprocessor, however, can take a significant amount of time as it has not been fully optimized yet.
  - The preprocessor pipeline can be parallelized using the ParallelFork() and ParallelJoin() filters.
- Capriqorn uses MDAnalysis (http://www.mdanalysis.org) for reading in trajectories.
  - From their website: “MDAnalysis is an object-oriented Python library to analyze trajectories from molecular dynamics (MD) simulations in many popular formats. It can write most of these formats, too, together with atom selections suitable for visualization or native analysis tools.”
  - To cite MDAnalysis, please visit http://www.mdanalysis.org/pages/citations/.

CHAPTER 5

Technical Features

Capriqorn provides – but is not restricted to – the following functionalities:

- versatile read-in of MD trajectories in various formats
- trajectory preprocessing
- high-performance parallel distance histogram calculation on CPUs and GPUs using the Cadishi package
- histogram postprocessing
- efficient data handling using HDF5 files

CHAPTER 6

Source documentation

To be done via docstrings after re-structuring of the code base.

6.1 Capriqorn modules

The following documentation was automatically generated from the docstrings present in the source code files.

TODO: include files after Capriqorn modularization

6.2 Capriqorn executables

The following documentation was automatically generated from the docstrings present in the source code files.

TODO: include files after Capriqorn modularization

CHAPTER 7

License and Citation

Capriqorn is released under the GPLv2 license. See the file `LICENSE.txt` for details.

- Jürgen Köfinger, Max Planck Institute of Biophysics, Department of Theoretical Biophysics, Max-von-Laue-Straße 3, 60438 Frankfurt am Main, Germany, juergen.koefinger@biophys.mpg.de
- Klaus Reuter, Max Planck Computing and Data Facility, Gießenbachstraße 2, 85748 Garching, Germany, klaus.reuter@mpcdf.mpg.de
- Max Linke, Max Planck Institute of Biophysics, Department of Theoretical Biophysics, Max-von-Laue-Straße 3, 60438 Frankfurt am Main, Germany, max.linke@biophys.mpg.de

CHAPTER 8

Indices and tables

- genindex
- modindex
- search