AMBER NetCDF Trajectory/Restart Convention
Version 1.0, Revision C

Original Trajectory Implementation: John Mongan (jmongan@mccammon.ucsd.edu)
REMD Trajectory, Restart Implementation: Daniel R. Roe (daniel.r.roe@gmail.com)

Last Revised: 2014-12-08

1 Introduction

The file format described in this document was developed for storing data generated by molecular dynamics simulations. It was introduced in version 9 of the AMBER suite of programs (http://ambermd.org/).

The primary design goals of this format are:

- Efficient input and output
- Compact, high-precision representation of data
- Portability of data files across different machine architectures
- Extensibility of the format (ability to add additional data without re-writing parsers)
- Compatibility with existing tools and formats

The file format is based on the NetCDF (Network Common Data Form) developed by Unidata (http://www.unidata.ucar.edu/software/netcdf/). NetCDF is designed for representation of arbitrary array-based data. Unidata provides libraries with bindings in C, C++, Fortran (F77 and F90), Java, Python, Perl, Ruby and MATLAB for reading and writing NetCDF files. The design goals above are largely met by NetCDF and the libraries that implement it. It is expected that all I/O of the format described here will occur through these libraries; this specification describes the file format at a high level in terms of the API implemented by version 3.6 of these libraries. In NetCDF terms, this document is a "Convention," describing the names, dimensions and attributes of the arrays that may be present in the file.

2 Program behavior

Programs creating trajectory or restart files ("creators") shall adhere strictly to the requirements of this document. Programs reading trajectory or restart files ("readers") shall be as permissive as possible in applying the requirements of this document. Readers may emit warnings if out-of-spec files are encountered; these warnings should include information about the program that originally created the file (see Global attributes, section 4). Readers shall not fail to read a file unless the required information cannot be located or interpreted. In particular, to ensure forward compatibility with later extension of the format, readers shall not fail or emit warnings if elements not described in this document are present in the file.

3 NetCDF file encoding

Trajectory/restart files shall be encoded in the manner employed by NetCDF version 3.x.

Those using NetCDF versions 4 or later should take care to ensure that files are read and written using this encoding, and not the HDF5 encoding.

Trajectory/restart files shall use 64 bit offsets

This can be accomplished by setting a flag during file creation; refer to API docs for details.

4 Global attributes

Global attributes shall have type character string. Spelling and capitalization of attribute names shall be exactly as appears below. Creators shall include all attributes marked required and may include attributes marked optional. Creators shall not write an attribute string having a length greater than 80 characters. Readers may warn about missing required attributes, but shall not fail, except in the case of a missing or unexpected *Conventions* or *ConventionVersion* attributes.

Conventions (required)

Contents of this attribute are a comma or space delimited list of tokens representing all of the conventions to which the file conforms. For trajectories, creators shall include the string *AMBER* as one of the tokens in this list. In the usual case, where the file conforms only to this convention, the value of the attribute will simply be

"AMBER". Readers may fail if this attribute is not present or none of the tokens in the list are *AMBER*. Optionally, if the reader does not expect NetCDF files other than those conforming to the AMBER convention, it may emit a warning and attempt to read the file even when the Conventions attribute is missing.

For restarts, creators shall include the string AMBERRESTART as one of the tokens in this list. Otherwise, the same above guidelines for trajectories shall be followed.

ConventionVersion (required)

Contents are a string representation of the version number of this convention. Future revisions of this convention having the same version number may include definitions of additional variables, dimensions or attributes, but are guaranteed to have no incompatible changes to variables, dimensions or attributes specified in previous revisions. Creators shall set this attribute to "1.0". If this attribute is present and has a value other than "1.0", readers may fail or may emit a warning and continue. It is expected that the version of this convention will change rarely, if ever.

application (optional)

If the creator is part of a suite of programs or modules, this attribute shall be set to the name of the suite.

program (required)

Creators shall set this attribute to the name of the creating program or module.

programVersion (required)

Creators shall set this attribute to the preferred textual formatting of the current version number of the creating program or module.

title (optional)

Creators may set use this attribute to represent a user-defined title for the data represented in the file. Absence of a title may be indicated by omitting the attribute or by including it with an empty string value.

5 Dimensions

frame (required for trajectories only, length unlimited)

Coordinates along the frame dimension will generally represent data taken from different time steps, but may represent arbitrary conformation numbers when the trajectory file does not represent a true trajectory but rather a collection of conformations (e.g. from clustering).

remd_dimension (required for Multi-dimension REMD, length set as appropriate)

When running replica exchange in multiple dimensions this will be set to the number of replica dimensions.

spatial (required, length 3)

This dimension represents the three spatial dimensions (X,Y,Z), in that order.

atom (required, length set as appropriate)

Coordinates along this dimension are the indices of particles for which data is stored in the file. The length of this dimension may be different (generally smaller) than the actual number of particles in the simulation if the user chooses to store data for only a subset of particles.

cell_spatial (optional, length 3)

This dimension represents the three lengths (a,b,c) that define the size of the unit cell.

cell_angular (optional, length 3)

This dimension represents the three angles (alpha,beta,gamma) that define the shape of the unit cell.

label (optional, length set as appropriate)

This dimension is used for character strings in label variables where the label is longer than a single character. The length of this dimension is equal to the length of the longest label string.

6 Variables

Variables are described below as "<type> <name>(<dimension> [,<dimension>..])". If dimension is not present the variable is a scalar. In the case where the type and/or dimensions differ between the trajectory and restart formats, they are preceded with the heading "Trajectory:" or "Restart:" respectively. If they do not differ they are preceded with the heading "Both:".

Note that the order of dimensions corresponds to the CDL and C APIs. When using the Fortran APIs, the order of dimensions should be reversed.

6.1 Label variables

Label variables shall be written by creators whenever their corresponding dimension is present. These variables are for self-description purposes, so readers may generally ignore them. Labels requiring more than one character per coordinate shall use the label dimension. Individual coordinate labels that are shorter than the length of the label dimension shall be space padded to the length of the label dimension.

char spatial(spatial)

Creators shall write the string "xyz" to this variable, indicating the labels for coordinates along the spatial dimension.

char cell_spatial(cell_spatial)

Creators shall write the string "abc" to this variable, indicating the labels for the three lengths defining the size of the unit cell.

char cell_angular(cell_angular, label)

Creators shall write the strings "alpha", "beta", "gamma" to this variable, naming the angles defining the shape of the unit cell.

6.2 Data variables

All data variables are optional. Some data variables have dependencies on other data variables, as described below. Creators shall define a *units* attribute of type character string for each variable as described below. Creators may define a *scale_factor* attribute of type float for each variable. Creators shall ensure that the units of data values, after being multiplied by the value of *scale_factor* (if it exists) are equal

to that described by the *units* attribute. If a *scale_factor* attribute exists for a variable, readers shall multiply data values by the value of the *scale_factor* attribute before interpreting the data. This scaling burden is placed on the reader rather than the creator, as writing data is expected to be a more time-sensitive operation than reading it.

For trajectories, it is left as an implementation detail whether creators create a separate file for each variable grouping (e.g. coordinates and velocities) or a single file containing all variables. Some creators may allow the user to select the approach. Readers should support reading both styles, that is, combining data from multiple files or reading it all from a single file.

For restarts, any variable present should be written to one file.

time

Trajectory: float time(frame) units = “picosecond”

Restart: double time units = ”picosecond”

When coordinates on the frame dimension have a temporal sequence (e.g. they form a molecular dynamics trajectory), creators shall define this variable and write a float for each frame coordinate representing the number of picoseconds of simulated time elapsed since the start of the trajectory. When the file stores a collection of conformations having no temporal sequence, creators shall omit this variable.

For restarts, if the coordinates are from molecular dynamics, creators shall define this variable and write the time (with double precision) that corresponds to the coordinates. Otherwise (if e.g. from a minimization) creators shall omit this variable.

coordinates

Trajectory: float coordinates(frame, atom, spatial) units = ”angstrom”

Restart: double coordinates(atom, spatial) units = “angstrom”

This variable shall contain the Cartesian coordinates of the specified particle for the specified frame.

cell_lengths

Trajectory: float cell_lengths(frame, cell_spatial) units = ”angstrom”

Restart: double cell_lengths(cell_spatial) units = “angstrom”

When the *coordinates* variable is included *and* the data in the coordinates variable come from a simulation with periodic boundaries, creators shall include this variable. This variable shall represent the lengths (a,b,c) of the unit cell for each frame. When each of the angles in *cell_angles* is 90, a, b and c are parallel to the x, y and z axes, respectively. If the simulation has one or two dimensional periodicity, then the length(s) corresponding to spatial dimensions in which there is no periodicity shall be set to zero.

cell_angles

Trajectory: float cell_angles(frame, cell_angular) units = ”degree”

Restart: double cell_angles(cell_angular) units = “degree”

Creators shall include this variable if and only if they include the *cell_lengths* variable. This variable shall represent the angles (α, β, γ) defining the unit cell for each frame. α defines the angle between the a-b and a-c planes, β defines the angle between the a-b and b-c planes and γ defines the angle between the a-c and b-c planes. Angles that are undefined due to less than three dimensional periodicity shall be set to zero.

velocities

Trajectory: float velocities(frame, atom, spatial) units = ”angstrom/picosecond”

Restart: double velocities(atom, spatial) units = “angstrom/picosecond”

When the *velocities* variable is present, it shall represent the Cartesian components of the velocity for the specified particle and frame. It is recognized that due to the nature of commonly used integrators in molecular dynamics, it may not be possible for the creator to write a set of velocities corresponding to exactly the same point in time as defined by the *time* variable and represented in the *coordinates* variable. In such cases, the creator shall write a set of velocities from the nearest point in time to that represented by the specified frame.

forces

Trajectory: float forces(frame, atom, spatial) units = “amu*angstrom/picosecond^2”

Restart: double forces(atom, spatial) units = “amu*angstrom/picosecond^2”

When the *forces* variable is present, it shall represent the components of the force for the specified particle and frame.

6.3 Replica Exchange Variables

temp0

Trajectory: double temp0(frame) units = "kelvin"

Restart: double temp0 units = "kelvin"

For use with replica exchange simulations, the *temp0* variable will define the current temperature that the thermostat is set to maintain, **NOT** the actual temperature of the coordinates.

> Note: For replica exchange self-guided Langevin (RXSGLD) simulations the stage ID is stored in temp0 instead of thermostat temperature. This allows existing analysis framework to properly sort these trajectories.

remd_dimtype

Both: integer remd_dimtype(remd_dimension)

For use with multi-dimensional replica exchange simulations, the *remd_dimtype* variable will define the type of each dimension using an integer value corresponding to the following table:

Value	Description
1	Temperature
3	Hamiltonian

remd_indices

Trajectory: integer remd_indices(frame, remd_dimension)

Restart: integer remd_indices(remd_dimension)

For use with multi-dimensional replica exchange simulations, the *remd_indices* variable indicates the position in all dimensions that each frame is in.

7 Example

The following is an example of the CDL for a trajectory file conforming to the preceding specification and containing most of the elements described in this document. This CDL was generated using `ncdump -h <trajectory file>`.

```
netcdf mdtrj {
dimensions:
        frame = UNLIMITED ; // (10 currently)
        spatial = 3 ;
        atom = 28 ;
        cell_spatial = 3 ;
        cell_angular = 3 ;
        label = 5 ;
variables:
        char spatial(spatial) ;
        char cell_spatial(cell_spatial) ;
        char cell_angular(cell_angular, label) ;
        float time(frame) ;
                time:units = "picosecond" ;
        float coordinates(frame, atom, spatial) ;
                coordinates:units = "angstrom" ;
        float cell_lengths(frame, cell_spatial) ;
                cell_lengths:units = "angstrom" ;
        float cell_angles(frame, cell_angular) ;
                cell_angles:units = "degree" ;
        float velocities(frame, atom, spatial) ;
                velocities:units = "angstrom/picosecond" ;
                velocities:scale_factor = 20.455f ;
// global attributes:
                :title = "netCDF output test" ;
                :application = "AMBER" ;
                :program = "sander" ;
                :programVersion = "9.0" ;
                :Conventions = "AMBER" ;
                :ConventionVersion = "1.0" ;
}
```

8 Extensions and modifications

Standards and formats are most useful when they are supported widely, and become less useful and more burdensome if they fragment into multiple dialects. If you plan to support additional variables, dimensions or attributes beyond those described here in a publicly released creator or reader program, please contact one of the Amber developers (daniel.r.roe@gmail.com, jason.swails@gmail.com,

case@biomaps.rutgers.edu, or jmongan@mccammon.ucsd.edu) for inclusion of these elements into a future revision of this document.

9 Revision history

- Revision A, February 9, 2006: Initial document.
- Revision B, February 15, 2006: Better self-description for unit cells in periodic simulations; standards for indicating one and two dimensional periodicity.
- Revision C, November 11, 2013: Add dimensions/variables for REMD; add NetCDF restart description.
- Revision D, December 8, 2014: Add brief note regarding storage of stage ID in temp0 variable for RXSGLD.