AMBER Trajectory NetCDF Convention
Version 1.0, Revision A

John Mongan (jmongan@mccammon.ucsd.edu)

9th February 2006

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://amber.scripps.edu).

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)
- Compatability 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 files ("creators") shall adhere strictly to the requirements of this document. Programs reading trajectory 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 compatability 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 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 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. 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.

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, size 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).

spatial (required, size 3)

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

atom (required, size set as appropriate)

Coordinates along this dimension are the indices of particles for which data is stored in the file. The size 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.

6 Variables

Variables are described below as <type> <name>(<dimension> [,<dimension>..])

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 are required whenever their corresponding dimension is present. Currently, all dimensions are required, so all label variables are required.

char spatial(spatial)

Creators shall write the string “xyz” to this variable, indicating the labels for coordinates along the spatial dimension. This variable is for self-description purposes, so readers may generally ignore it.

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.

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.

float time(frame) units = ”picosecond”

When coordinates on the frame dimension have a temporal sequence (e.g. they form a molecular dynamics trajectory), creators shall define this dimension 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.

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

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

float cell_lengths(frame, 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 of the unit cell for each frame.

float cell_angles(frame, spatial) units = ”degree”

Creators shall include this variable if and only if they include the *cell_lengths* variable. This variable shall represent the angles defining the unit cell for each frame.

float velocities(frame, 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.

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 ;
variables:
        char spatial(spatial) ;
        float time(frame) ;
                time:units = "picosecond" ;
        float coordinates(frame, atom, spatial) ;
                coordinates:units = "angstrom" ;
        float cell_lengths(frame, spatial) ;
                cell_lengths:units = "angstrom" ;
        float cell_angles(frame, spatial) ;
                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 the author (jmongan@mccammon.ucsd.edu) for inclusion of these elements into a future revision of this document.