Proposal for NetCDF (re)Implementation for use with Planar Velocimetry Data

C. Willert

Institute of Propulsion Technology, German Aerospace Center (DLR), 51170 Koeln, Germany; (email: chris.willert@dlr.de)

Abstract

In order to efficiently share PIV data between research groups in Europe it was decided at a very early stage (1994) to use a self-describing, platform-independent data format. The netCDF software library was chosen to implement this data format. This library was originally developed at the UNIDATA Program Center (UCAR, [1]) in Boulder, Colorado, to suit exactly these types of data exchange requirements. Its source code and associated utilities is freely available.

Since the first PIV-specific netCDF implementation the PIV technique has undergone significant changes that make it necessary to revise the original format. The open structure of the netCDF format makes it possible to retain older variable names and data structures (for backward compatibility) while adding new ones. In principle it would be possible to generate data files that suit both the original as well as the proposed implementation. Feedback collected from up-coming PIV-related workshops as well as experiences gained by current users guarantees the success of this (re)implementation. Once the revised implementation is accepted, both a C/C++ interface and ASCII-to-netCDF conversion tools can be provided on a suited internet-based server (e.g. [2]).

1 Introduction - Why yet another implementation ?

The current netCDF implementation for PIV was defined in the framework of GARTEUR (1993-1995) and then finalized/implemented within the EuroPIV BRITE-EURAM contract (1995-1998, see Chapter 3.2 in [3]). Back then the PIV technique was still undergoing rapid developments in the process of maturing from the laboratory stage to today's wide-spread research tool for both industry and academia. Fully digital PIV was still emerging while the photographic techniques used in the mid 90's have essentially vanished from the field (with the exception of holographic PIV). Multiple-plane or stereoscopic PIV were only scarcely used at the time. This means that a number of parameters of the original netCDF implementation are no longer required, while new ones should be added to suit new developments. Also there are a few inconsistencies within the original implementation.

Finally a variety of other planar techniques could take profit of a more generalized implementation of the netCDF data format for PIV. Planar Doppler velocimetry (=DGV) and background oriented schlieren (BOS) both are methods with similar requirements on data storage. In the context of the PIV-Challenge held in 2001 the original netCDF implementation was already modified for purposes specific to the challenge.

1.1 What is required ?

The revised data format should fulfill the following requirements

- capable of handling more than two dimensions (volume data!),
- capable of handling time-resolved data set (addition of time-axis),
- provide a choice of data grids: evenly gridded, irregularly gridded, randomly spaced,
- support for multiple components,
- support for multiple data sets (e.g. velocity, raw displacement, vorticity, etc. all in the same file),
- capable of combining data from differing (planar) measurement techniques (transparency of data),
- capable of seamlessly accepting format extensions in the future,
- easy (electronic) access to third parties, providing source code, simple utilities and documentation.
- form a basis for standardization and ISO9000 certification procedures (i.e. Work package 4.2 in PIVNET-2 project)

1.2 What is available ?

Among the data formats available the easiest to handle are ASC-II formatted text file as these can be read and modified by essentially any text editor. However, problems start arising when these files have to be read by another application. For instance, floating point numbers can be represented with different significant figures; decimal points are locale specific (comma '**,**' in. Germany vs. period '**.**' in the United Kingdom). At the same time the numeric entries may be separated by spaces, tabs or any other special character (even commas!). Further, ASC-II formatted files require roughly four times the storage space as their binary-formatted counterpart. The biggest disadvantage of ASC-II formatted files is that their content cannot be accessed randomly, that is, they must be read in their entirety (line-by-line) because they lack an internal (descriptive) directory structure which could be queried during loading. This in turn makes loading and writing of ASC-II files slow and resource consuming.

The first step toward a more suitable data format is to depart from ASC-II formatted files in favor of a binary format which is both memory-conserving and more efficiently accessed. This of course requires that the structure of the file's content is precisely defined. In this context it makes sense to utilize exiting data

format specifications rather than 're-inventing the wheel'. Several such implementations, offering –but not limited to– low-level data handling, are available:

- **netCDF –** *network Common Data Form is an interface for array-oriented data access and a library that provides an implementation of the interface. The netCDF library also defines a machine-independent format for representing scientific data. Together, the interface, library, and format support the creation, access, and sharing of scientific data* [1].The netCDF software was developed at the Unidata Program Center in Boulder, Colorado, and is freely available. A more detailed description of netCDF will be given in the following sections.
- **HDF –** Hierachical Data Format was developed by the National Center for Supercomputing Applications (NCSA, Illinois, U.S.A.) with the intention of *developing, deploying and supporting open and free technologies that facilitate scientific data exchange, access, analysis, archiving and discovery* [4]. HDF is very well suited to handle experimental data generated by image-based techniques and also is capable of handling netCDF formatted files. This implies, that a conversion between netCDF and HDF is feasible in the future.
- **CGNS** – CFD General Notation System *consists of a collection of conventions, and software implementing those conventions, for the storage and retrieval of CFD (computational fluid dynamics) data. CGNS is designed to facilitate the exchange of data between sites and applications. The data are stored in a compact, binary format and are accessible through a complete and extensible library of functions* [5]. It is freely available through terms of the GNU Lesser General Public License [6]. In principle CGNS is capable of fulfilling the requirements stated earlier and is especially attractive in the context of sharing experimental data with the CFD community. However, proper use of the CGNS format requires a thorough knowledge of CFD-specific terminology. Also, there is no obvious way to store descriptive information such as processing variables in a self-descriptive manner. Nevertheless, utilities capable of extracting the fluid dynamically relevant data from experiment-specific data files, such as the netCDF-implementation proposed below, for the generation of CGNS data sets may become necessary.

All three of these data formats have in common that they are machine-independent, that is, insensitive to hardware-specific byte-ordering (endianess) and memory alignment issues. Although similar capabilities are provided by file formats based on commercial software packages, such as the TECPLOT binary format (Amtec, Inc.) or the MAT-File format for MATLAB (Mathworks, Inc.), issues concerning licensing and redistribution make them unsuitable in the present context.

1.3 Brief overview of netCDF

The netCDF package including source code can be obtained via FTP directly from UCAR or through several mirror sites. The library can be compiled on nearly any

platform. A variety of programming interfaces (e.g C, C++, Fortran, Java, Perl, Tcl/Tk, Python) as well as numerous utilities for access and manipulation of netCDF data are available. Commercial packages such as MATLAB (Mathworks, Inc.) and IDL (Research Systems Inc.) provide utilities to access netCDF files.

Although not directly readable with a text editor[1], the data within a netCDF file is arranged in a self-documenting structure, that is, it is grouped into *variables* with specified (and named) *dimensions*. *Attributes* specific to each variable give descriptive information of the stored data., such as units, titles or even processing parameters. In general NetCDF data sets are structured as follows:

1. **Dimension** fields are used to store the dimensions of a data set and are identified through user specific dimension names (e.g. `xdim=64`). The maximum number of dimensions may exceed 100 which is far beyond the space-time requirements of common experimental data.
2. Freely named **variables** contain the actual multidimensional data. This could be anything from 3-C velocity data, 1-D grid coordinates to scalar image data. Information about the data set dimensions is provided through the associated dimension fields. Here several data sets within the file may share the same dimension fields. Currently, netCDF offers a limited number of external numeric data types: character strings, 8-, 16-, 32-bit signed integers, and 32- or 64-bit floating-point numbers. Other data types have to be converted to/from either of available types. There are no 'real' variable size limitation as file sizes beyond 2 GByte are possible.
3. Named **attributes** are used to provide information specific to a given variable (e.g. data set) or about the file in general (e.g. **global attributes**). Examples of attributes are data units, descriptive titles, interrogation parameters or conversion factors.

While netCDF is intended for self-documenting data, it is often necessary for a given group of users to agree upon a common **naming convention**. Although Unidata recommends to adopt one of the exiting netCDF naming conventions, these are strongly related to applications in geophysical research and only have minor commonality to planar velocimetry data (e.g. PIV or DGV) which makes a new formulation necessary. In the following this paper outlines some of the PIV-specific aspects in formulating a new naming convention.

2 NetCDF formulation for PIV and similar data

2.1 Description of file contents

In order to distinguish the new data format from its predecessors or other netCDF files it is suggested to use the `Conventions` global attribute:

[1] The *ncdump* utility can be used to convert a netCDF-file to a human-readable ASC-II format.

```
:Conventions="PIVNET2"
:file_format_version=1.0f
```

Additional, but not mandatory, attributes are used to describe the file content :

`title`	a short description of what is in the data set
`history`	provides a summary of steps in the modification of the file
`institution`	specifies where the original data was produced
`references`	published references that describe the data or methods used to produce it
`source`	method by which the data in the file was generated (e.g. PIV, DGV,...)
`comment`	additional information about the data
`software`	information about the software used to create the file
`Type`	Regular grid, irregular grid, grid-less data
`dimensions`	2D, 3D, or 2D + time, or 3D + time (maximum 4)

2.2 Making use of variable-specific attributes

NetCDF allows the storage of attributes in a global sense or specific to a certain variable. The previous netCDF-PIV implementation only made use of global variables. Here it is suggested to also use the non-global variables because this makes organization and data retrieval much easier. Examples for attribute names that can be associated with each variable are:

`units`	unit of variable (data set). Where possible the units should follow the Unidata UDUNITS standard [7].
`long_name`	human readable descriptive name for a variable's content. This could also be used for labeling plots.
`title`	optional title, for instance for axis labeling (text string)
`C_format`	formatting specification for use with ncdump, e.g.: `C_format=%10.5f;`
`valid_range`	expected "reasonable" range for variable
`valid_min`, `valid_max`	minimum, maximum value in given data set (alternative to `valid_range`)
`actual_range`	actual data range for variable
`Fill_Value`	useful for indicating missing data (has default values in the netCDF-library)
`add_offset`	if present, this number is to be added to the data after it is read by the application that accesses the data
`scale_factor,`	if present, the data are to be multiplied by this factor after being read.

The attributes `scale_factor` and `add_offset` can be used together to provide simple data compression to store low resolution floating-point data as small (short) integers in a netCDF file. The unpacking algorithm is:

```
unpacked_value = add_offset + ( packed_value * scale_factor )
```

2.3 Some comments on coordinate systems

Although any desired coordinate system could be implemented using the NetCDF interface the Cartesian (orthogonal) system chosen in the initial implementation should be retained. The origin is always located in the lower left even though traditional image processing places it in the upper left. The X-axis will be parallel to the lower image rows, its values increasing from left to right. The Y-axis is aligned along the image columns, with y-values monotonically increasing from bottom to top. The Z-axis will then point normal to the plane spanned by the X- and Y-Axis. Starting from the lower left corner the data itself is stored line-by-line, that is, the X-index (horizontal dimension) varies the fastest, followed by the Y-index (vertical dimension), etc.

All data stored in the file should adhere to this convention. Transformations used to map these coordinates into some desired object space are optional and should be defined by specifying an appropriate transformation matrix.

2.4 Multivariate variable handling

The previous netCDF-PIV implementation stored PIV vector data in one set, that is, both components were given the same variable name `piv_data`. The dimension `vec_dims` specified the number of vector components which works fine as long as all components are available. In some cases some vector components may be missing. Also vorticity data is usually restricted to the out-of-plane component only. To simplify data access it is suggested to store the available components in separate (scalar) variables. The data reading routine then has the responsibility of checking which of the desired components are available.

Furthermore the variable `piv_data` was used to interchangeably handle either raw or converted (velocity) data which can lead to serious trouble when reading the data. Here it is proposed to use distinct names for each data type, storing each component separately, e.g. scalar data only. This does not make sense at first sight (e.g. for displacement or velocity) but is of importance for the storage of differential quantities for which not all components are usually available. Some examples of the proposed variable names are:

`disp_x`, `disp_y`, `disp_z`	displacement data, X-, Y-, Z-components
`vel_x`, `vel_y`, `vel_z`	velocity data, X-, Y-, Z-components

`shift_x, shift_y, shift_z`	physical shift data, X-, Y-, Z-components (i.e. for BOS data or speckle interferometry)
`vort_x, vort_y, vort_z`	vorticity data, X-, Y-, Z-components
`strain_xx, strain_xy,…`	components of the strain tensor
`dvx_dx, dvx_dy, …`	differential estimates for velocity
`coords_x, …`	coordinates of data for randomly spaced (ungridded) data (see below)
`flags`	formerly `piv_flags`, signaling the status of each data point

Each data set should have its unit specified as one of its own attributes (e.g. non-global!). Also all data sets specified within one file should have the same dimensions, that is, the grids have to match between the variables.

Within the data file the `flags` variable plays a special role in that it indicates missing, invalid or disabled data within a given data set. If necessary, a flag array specific to a given attribute should be used (e.g. `vort_flags` for vorticity data). The flag variable is an 8-bit valued array for which the combination of different bits gives information about the status of the data:

Bit	Value	Description
1	1	if set, then **value is active** (only bit active if the raw data s retained, see bits 5 to 7 otherwise) if not set, then data value is not active (see bits 2 to 4 for explanation),
2	2	if set, then the value was **not measured** or calculated (i.e. a data point lying inside a mask or outside a boundary)
3	4	if set, then the measurement technique gave **no result**
4	8	if set, then the data value was automatically disabled by a post-processing filter (e.g. **outlier**) and not replaced.
5	16	if set, then the data value was replaced through a **secondary measurement** (e.g. 2nd or 3rd peak in PIV correlation plane)
6	32	if set, then the data value was calculated by some **filter** (i.e. smoothing with neighbors, interpolation)
7	64	if set, then the data value was **manually edited**
8	128	currently not used

2.5 Gridding of data

As depicted in Fig. 1, it is proposed to provide support for three types of data grids:

1. **regular-spaced grid data**: All data is evenly spaced from node to node. The grid is defined by specifying the coordinates along the axes using the variables `grid_x`, `grid_y`, `grid_z`, and `time`.
2. **irregular-spaced, grid data**: The data is spaced unevenly on any of the available axes. The grid is defined by specifying the coordinates along the axes using the variables `grid_x`, `grid_y`, `grid_z`, and `time`.
3. **grid-less data**: The data is completely randomly spaced in which case the coordinates have to be specified for every point using separate variables (ie. `coords_x`, `coords_y`, `coords_z and time`.). This is common with tracking data.

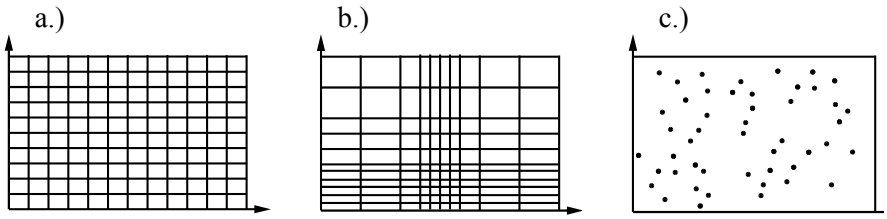

Fig. 1. Data grid formats: a) evenly spaced grid; b) irregularly spaced grid, c) grid-less data (randomly positioned data)

2.6 Image origin, magnification factors and conversion

Unless specified otherwise axis and coordinate data are specified in image space with pixel units with the origin in the lower left corner of the lower left pixel (this pixel has coordinates (0.5,0.5) in the image. Conversion into physical space requires both a magnification factor and the coordinates of the reference point in image space. Delay times are required for the conversion of displacement data into velocity. All of these should be specified as global variables:

`scale_x`	horizontal magnification factor in [pixel/physical unit]
`scale_y`	optional vertical magnification factor (if xmag ≠ ymag)
`scale_z`	optional depth magnification factor for volumetric image data
`scale_unit`	optional unit of conversion – default: `image_unit/phys_unit`
`image_origin_x`	x, y and z coordinates of reference point in image

image_origin_y image_origin_z image_unit	space. The z-coordinate may be required for multi-plane (volumetric) data required unit
phys_origin_x phys_origin_y phys_origin_z phys_unit	physical x, y and z coordinates of reference point in physical units of length required unit
time_delay time_delay_unit	time delay between exposures of PIV records unit of time delay, required
time_origin time_separation time_unit	Origin of time for time series Time separation of the records for time series Unit for time origin and separation, required

In case of non uniform magnification (angular viewing, stereoscopic PIV…), the equation of the transformation function should be provided using the scale_function (or scale_function_1 and scale_function_2 for stereo PIV) attributes.

2.7 Parameters unique to measurement technique

In cases where the netCDF file is to contain parameters unique to a specific measurement technique these should be stored with a unique prefix in the form of global attributes, for example: `piv_sample_size_x` , `ptv_max_displacement` or `dgv_ref_transmission`.

In principle a valid netCDF file may even lack variable data containing only global attributes, which makes it very suitable as a parameter file for processing software. For instance, the PIV software package of the German Aerospace Center (DLR) has been making extensive use of this practice since 1995, providing transparent data access through platform independent access (Both PC's and SUN workstations are used for PIV processing and data analysis).

2.8 Further recommendations

Since netCDF variable and attribute names are case sensitive it is recommended to keep them lower case were possible. Also netCDF names should only make use of underscores in place of hyphens. Leading underscores should be avoided as these are frequently reserved for system use. All of this makes the implementation of reading and writing routines easier and more robust. (Please note: Many of the variable and attribute names presented in the previous sections serve as descriptors only as the final format has not been finalized as of this writing.)

3 Outlook

The primary intention of this paper was to outline a strategy toward defining a data format capable of handling time-resolved, volumetric data from a variety of today's whole-field measurement techniques. To suit the needs imposed by future developments an extensible, self-describing data format based on netCDF was chosen, which allows for a straightforward extension to other dimensions such as the frequency domain or phase space. The ultimate aim of any such implementation is to provide a common data format that facilitates unambiguous transfer of data between researchers. While this paper was being written, the format was undergoing final revisions prior to its release. The final definition of the data format along with sample data sets, utilities and associated source code will be made freely available to the public through the PIVNET site [2].

Acknowledgements

Existing netCDF conventions were highly valued in the formulation of this proposed format. Especially helpful were the suggestions made by the *Climate and Forcast (CF) Metadata Convention* [8] although it was too specialized to meteorological research to qualify for use in experimental fluid dynamics.

References and Links

[1] NetCDF Software package, from the UNIDATA Program Center of the University Corporation for Atmospheric Research, http://www.unidata.ucar.edu/packages/netcdf/

[2] PIVNET info base : http://pivnet.dlr.de/

[3] J. Kompenhans (2000): "Test and comparison of various methods of analysis and post-processing on a database of PIV records" in *Particle Image Velocimetry – Progress toward industrial application,* edited by: M. Stanislas, J. Kompenhans and J. Westerweel, Kluwer Academic Publishers, Dordrecht, NL:

[4] NCSA Hierachical data Format (HDF): http://hdf.ncsa.uiuc.edu/

[5] CFD General Notation System (CGNS): http://www.cgns.org/

[6] GNU Lesser General Public License, http://www.gnu.org/

[7] UDUNITS Software Package, from the UNIDATA Program Center of the University Corporation for Atmospheric Research, http://www.unidata.ucar.edu/packages/udunits/

[8] NetCDF Climate and Forecast (CF) Metadata Conventions, http://www.cgd.ucar.edu/cms/eaton/cf-metadata/