7 SCIENTIFIC HIGHLIGHT OF THE MONTH: An extensible and portable file format for electronic structure and crystallographic data

An extensible and portable file format for electronic structure and crystallographic data

X. Gonze[1,2], C.-O. Almbladh[1,3], A. Cucca[1,4], D. Caliste[1,2,5],
C. Freysoldt[1,6], M. Marques[1,7,8], V. Olevano[1,4,9], Y. Pouillon[1,2,10], M.J. Verstraete[1,11]

[1]European Theoretical Spectroscopy Facility (ETSF)
[2]Université Catholique de Louvain, Louvain-la-Neuve (Belgium)
[3]University of Lund, Lund (Sweden)
[4]LSI, CNRS-CEA, Ecole Polytechnique, Palaiseau (France)
[5]C.E.A. Grenoble, Grenoble (France)
[6]Fritz-Haber-Institut, Berlin (Germany)
[7]F. U. Berlin, Berlin (Germany)
[8]U. Coimbra, Coimbra (Portugal)
[8]Fritz-Haber-Institut, Berlin (Germany)
[9]Institut NEEL, CNRS and U. Joseph Fourier, Grenoble (France)
[10]Universidad del País Vasco UPV/EHU, Donostia-San Sebastián (Spain)
[11]U. York, York (United Kingdom)

In order to allow software applications to interact and exchange data, file format specifications are mandatory. Widely agreed file format specifications are still lacking in the field of first-principles calculations of material properties. One of the (numerous) objectives of the European Network of Excellence “NANOQUANTA” (that is about to launch the “European Theoretical Spectroscopy Facility”) is precisely to specify file formats, for the contents that are relevant to the scientific activity of its constituent nodes. The present article gives an overview of the agreed specifications, relevant for selected content (crystallographic/density/potential/wavefunctions). The specification relies on the NetCDF library, widely used in many different scientific communities. It is a binary format, that provides complete portability of the files accross languages (C/C++/Fortran/...) and platforms (big/little-endian is irrelevant). The files are addressed by content, bringing in the additional advantages of extensibility and automatic backward compatibility. Different software applications already implement this specification, for which a specific I/O library has been developed. It is hoped that it will be implemented in other software projects, or (at least) will be the basis of even better file format specifications.

1 Introduction

In a domain of application where different software projects coexist, user's demand often drives file standardization. Different standards might be used concurrently, as, for example, JPEG, PNG, GIF, EPS, ... , that encode 2D images. The development of associated conversion software quickly follows the definition of the specification.

However, in a field which is in constant mutation, such as first-principles calculations of materials properties (electronic-structure based), every software project usually develops its own type of files, matching its capabilities at one moment in time, without paying much attention to file standardization. For reasons of ease of development and economical use of storage space, the large files are naturally based on the binary representation provided by the native programming language (e.g. F77/90, C/ C++). This has several drawbacks: (i) the produced files might not be portable between a big-endian platform to a little-endian platform (and vice-versa), or between a 32 bit and a 64 bit platform; (ii) files written in F77/90 cannot be easily read by C/C++ codes (and vice-versa); (iii) these files are not extensible, and one file produced for one particular version of the software might not be readable by a past/forthcoming version.

These problems were addressed by software scientists and several other scientific communities already a long time ago. In particular, the meteorology/climatology community developed a library, called NetCDF [1, 2], that provides tools for generating/reading files that are portable accross platforms (independent of the native binary encoding -big/little endian- but also of some platform specificity, like the 8-byte encoding of single precision on Cray machines, instead of the usual 4 bytes), independent of the programming language (C/C++, F77/90, Java, Perl, Python, ...), and addressed by content (usually solving the problem of backward compatibility). This appears in retrospect as a crucial step in the cross-validation of simulations performed by different groups, working with different platforms and codes.

The idea of standardization of file formats is not new in the electronic-structure community [3]. However, it proved difficult to achieve without a formal organization, gathering code developers with a sufficient incentive to realize effective file exchange between software. In the EU Nanoquanta Network of Excellence (see Appendix 1, where the connection with the future "European Theoretical Spectroscopy Facility" -ETSF- is outlined), the standardization of file formats appears as an explicit goal of the project. For the sake of brevity, we will refer to this project as NQ/ETSF. After several preliminary steps needed to achieve this goal, involving discussions on coding rules, selection of contact persons for the different NQ/ETSF software, and an agreement to use NetCDF for large files (instead of native binary files), a specification for NQ/ETSF file formats was gradually written, by way of "mini-workshops" and e-mail exchange.

The outcome of this procedure is a document available on the NQ/ETSF site [4], whose main ideas will be presented here, together with several detailed excerpts. This specification has already been implemented in different software, either on the basis of a full-fledged library, ETSF_IO [5, 6] or using an early exploratory interface [7] . Additional information can be found in Refs. [8, 9].

Actually, NetCDF is not the only possible standard in the above-mentioned context. HDF [10] also implements a software solution that addresses the portability and extensibility problems.

NetCDF being already used by several NQ/ETSF software projects at the start of our work, without major drawbacks (considering our target use), we simply stuck to this choice. As a matter of fact, NetCDF and HDF have now announced that they would merge within a few years.

Section II presents a brief account of the major characteristics of NetCDF, then outlines our generic design decisions for the NQ/ETSF file format specification, and lists the few global attributes of NQ/ETSF NetCDF files. The reader will not find there a full description of the use of NetCDF, but only sufficient NetCDF background to understand our design decisions and the NQ/ETSF specifications themselves. Then we present (Section III) the specification for files that contain (at least) a charge density, represented on a regular grid in real space. This is the simplest part of the NQ/ETSF specification, thus providing an excellent example of our methodology. In section IV, we briefly describe the specification for the potentials, crystallographic data and wavefunction parts of the NQ/ETSF files. Section V summarizes the present status of the specification, and planned improvements.

2 Generic design considerations

In a NetCDF (Network Common Data Form) file, a series of numerical arrays can be stored, each under the name of a variable, that might possess one or several attributes and possibly one or several dimensions. One finds also the global attributes of the file (not associated to one particular variable).

The NetCDF library provides functions to initialize a NetCDF file, to create variable names and their dimensions in this file, to associate attributes to them, to define their dimensions, and to store the associated numerical data. It provides also functions to inquire about the content of a file (names of variables, associated dimensions and attributes), to access the information associated to a variable name (in full or by segments), to copy it, to rename attributes or variables, or to delete some of its content. Thanks to the powerful inquiry functions, it is possible to get full information from a NetCDF file without *a priori* knowledge of its content. In most cases, however, the user knows the name of the variables, the content he/she would like to retrieve, and the associated dimensions.

The ability of NetCDF to retrieve the information, irrespective of the actual physical layout of the file, is a key characteristic allowing exchange of data between different software (and also different versions of the same software), that contrasts with the rigidity of the usual binary representations.

The first step in the NQ/ETSF specification consists in the definition of variable names, associated dimensions and attributes, and accurate definition of the corresponding physical quantities.

Our first design decision, following from the flexibility of NetCDF, dealt with the different types of variables/attributes and associated numerical information that might be stored in a NQ/ETSF NetCDF file. We distinguished four different types of variables:

(A) The actual numerical data (which defines whether a file contains wavefunctions, a density, etc), for which a name must have been agreed in the specification document [4] (and properly described in this specification document).

(B) The auxiliary data that is mandatory to make proper usage of the actual numerical data of A-type. The name and description of this auxiliary information is also agreed in the specification document [4].

(C) The auxiliary data that is not mandatory to make proper usage of the A-type numerical data, but for which a name and description has been agreed in the specification document [4].

(D) Other data, typically code-dependent, whose availability might help the use of the file for a specific code. The name of these variables should be different from the names chosen for agreed variables of A-C types. Such other data might even be redundant with them.

The four types are compatible with a file being sufficiently complete for use by many different codes, though adapted to the specific usage by each. The NQ/ETSF file descriptions to be provided below (and fully in the specification document [4]) are based on this generic classification: for three sets of numerical data (A-type : density/potential ; crystallographic ; wavefunctions), we define the auxiliary data that is mandatory to make proper usage of it (B-type variables). In addition, we provide names for variables that can be either mandatory or not (in the context of a file containing a density/potential, or a wavefunction, or crystallographic data, or other large numerical data not yet taken into account), but for which a NetCDF description has been agreed.

Some technical details concerning the use of NetCDF files will apply to all specifications in the NQ/ETSF framework:

(1) Concerning the names of variables, attributes, and dimensions, long names have been chosen, as close as possible to natural language (so inherently self-descriptive, e.g. "number_of_atoms").

(2) All names are lower case, except the global attribute "Conventions" - a name agreed by the NetCDF community.

(3) Underscores are used to replace blanks separating words in names.

(4) In the specification of the dimensions of the variables, the slow indices are left-most, and the fast indices are right-most, like in C, so that the order of indices has to be reversed in FORTRAN.

We come now to the description of global attributes, and of two attributes used for many different variables.

Global attributes are used for the general description of the file (mainly the file format convention). Important data is not contained in attributes, but rather in variables. There are five global attributes in NQ/ETSF files: "file_format", "file_format_version", "Conventions", "history", and "title". The first three are mandatory, while the fourth and fifth are optional. "file_format" is a character string, always equal to "ETSF Nanoquanta" . The real number "file_format_version" gives the version of the specification (2.1 at present). "Convention" is a NetCDF recommended attribute specifying where the conventions for the file can be found on the Internet, in our case "`http://www.etsf.eu/fileformats`". The string "history" is a NetCDF-recommended attribute: each code modifying/writing this file is encouraged to add a line about itself in the history attribute. The string "title" is a short description of the content of the file.

The problems associated with the definition of units have also been addressed in a global way, although the corresponding attributes will all be associated to a particular variable. "units" is one of the NetCDF recommended attributes. It applies to several variables in our case, although many of them (e.g. integer type variables) do not have a physical dimension (for variables with a physical dimension, this attribute is required). The use of atomic units (aka Hartree, the "units" variable attribute having the string value "atomic units") is advised throughout for portability. If other units are used, the definition of an appropriate scaling factor to atomic units ("scale_to_atomic_units" attribute) is mandatory. Actually, the definition of the name "units" in the NQ/ETSF files is only informative: the "scale_to_atomic_units" attribute should be the only one used for machine reading of the file. Indeed, if "units" is something other than the character string "atomic units" (based on Hartree for energies, Bohr for lengths) we request the definition of an appropriate scaling factor. The appropriate value in atomic units is obtained by multiplying the number found in the variable by the scaling factor. Examples:

units="eV" → scale_to_atomic_units = 0.036749326

units="angstrom" → scale_to_atomic_units = 1.8897261

units="parsec" → scale_to_atomic_units = 5.8310856e+26

This can be used to deal with unknown units. Note that the recommended values for the fundamental constants can be found in Ref. [11].

Dimensions (in the C/C++/Fortran sense) are used for one- or multi-dimensional variables. It is very important to remember that the NetCDF interface adapts the dimension ordering to the programming language used. The notation presented in the next sections is C-like, i.e. the last index varies fastest. In Fortran, the order is reversed. When implementing new reading interfaces, the dimension names can be used to check the dimension ordering. The dimension names also help to identify the meaning of certain dimensions in cases where the number alone is not sufficient.

NetCDF files that respect the NQ/ETSF specifications should be easily recognized, irrespective of whether they contain a density, wavefunctions, or crystallographic data. We suggest to append the string "-etsf.nc" to their names. The ".nc" extension is a standard convention for naming NetCDF files [12]. Some filesystems are case-insensitive, and this motivates the lower-case choice. Finally, a dash is to be preferred to an underscore to allow the files to be referenced by a Web search engine.

3 Specifying a density

We now specify the content needed for a file to be declared a NQ/ETSF density file (the density is represented on a real space grid). Of course, it should follow the general rules indicated above.

Such a file should contain (at least) three global attributes, two variables, and seven dimensions. The three global attributes ("file_format", "file_format_version" and "Conventions") have been mentioned in Section II.

The most important variable is "density". It is a type-A variable for a file to be declared NQ/ETSF density file (and the only type-A variable needed for a file to be declared a density file). Its structure is the following (under C/C++ convention for ordering – Fortran is just the reverse):

double [number_of_components] [number_of_grid_points_vector3] [number_of_grid_points_vector2] [number_of_grid_points_vector1] [real_or_complex_density]

In the present status of the specification, such a "density" is suitable to represent densities obtained on a homogeneous 3D grid, typical of norm-conserving pseudopotential calculations, i.e. pseudodensities. In case of ultrasoft pseudopotentials, or Projector-Augmented Waves, the augmentation contribution is missing. Similarly, a specific treatment to represent the density inside an atomic sphere (as in Augmented Wave formalisms) is not yet defined. Later work will include the extension of the specification to such other basis methodologies.

The "units" attribute is mandatory. By default, the density is given in atomic units, that is, number of electrons per cubic Bohr. If another unit is used, the attribute "scale_to_atomic_units" is mandatory.

Eight other type-B variables or dimensions must also be specified in a NQ/ETSF density file, for the purpose of being able to interpret correctly the information contained in the "density" variable : "number_of_components", "number_of_grid_points_vector1", "number_of_grid_points_vector2", "number_of_grid_points_vector3", "real_or_complex_density", "primitive_vectors", "number_of_vectors", and "number_of_cartesian_directions". All of them, except "primitive_vectors", are NetCDF dimensions.

"number_of_components" is a dimension, with value 1, 2 or 4, related to the spin characteristics of the density: 1 if the density is a spin-scalar, 2 if the density is spin-collinear (spin up and spin down), 4 if the density is spin-non-collinear (average density, then magnetization vector in cartesian coordinates x, y and z).

"number_of_grid_points_vector1" gives the number of grid points along direction 1 in the unit cell in real space. A similar meaning applies to "number_of_grid_points_vector2" and "number_of_grid_points_vector3". The real space grid starts at the origin in real space, with coordinates (0 0 0).

"real_or_complex_density" is a dimension, with value 1 or 2, respectively, depending whether the density is real or complex, respectively. Complex densities in real space might happen in the framework of responses in perturbation theory.

The primitive vectors are now defined thanks to the variable "primitive_vectors", with structure (under C/C++ convention for ordering – Fortran is just the reverse):

double [number_of_vectors] [number_of_cartesian_directions]

The primitive vectors are specified in cartesian coordinates. The two additional dimensons "number_of_vectors" and "number_of_cartesian_directions" are always equal to 3. Despite the fixed value ("3") attributed to both of these dimensions, we choose to define them, in order to waive any ambiguity in the ordering of the dimensions for the "primitive_vectors" variable.

4 Outline of the full specification

The full specification document [4] proceeds from the simplest concepts to the most complex ones, at variance to the order by which we presented the specification of the density in Section III. After global attributes and the other widely used attributes, dimensions are defined, then several optional (typically type-C) variables, then finally the type-A and type-B variables.

Some of the dimensions have already been described in Section III : “number_of_vectors”, “number_of_cartesian_directions”, “number_of_components”, “number_of_grid_points_vector1”, “number_of_grid_points_vector2”, “number_of_grid_points_vector3”, “real_or_complex_density”.

Additional dimensions include (the list below is not exhaustive): “character_string_length”, “real_or_complex_wavefunctions”, “number_of_symmetry_operations”, “number_of_atoms”, “number_of_atom_species”, “number_of_kpoints”, “number_of_spins”, “number_of_spinor_components”, etc.

We decided to address from the very start the possibility to distribute the data on different processors, each accessing one of a series of files, which together contain the full information. In this context, we have associated to some of the dimension variables an additional pair of variables describing the distribution (or splitting) of data.

As an example, suppose that one deals with seven wavevectors (nicknamed kpoints) in the Brillouin Zone, with a wavefunction file to be distributed on two processors (hence two wavefunction files, one containing e.g. information related to wavevectors number 1, 3, 5 and 7, and the other containing information related to wavevectors number 2, 4 and 6). The value of the variable “number_of_kpoints” will be 7 in both files (as if the file had not been split), the value of the variable dimension “my_number_of_kpoints” (not defined if the file were not split), will be four for the first processor file and three for the second processor file. The value of the variable “my_kpoints” (not defined if the file were not split) will be (1,3,5,7) for the first processor, and (2,4,6) for the second processor.

In the category of optional variables (type-C), whose names were agreed upon in order to avoid divergence of the format for the additional data, one finds (the list below is not exhaustive): “valence_charges”, “number_of_electrons”, “kpoint_grid_vectors”, “exchange_functional”, etc.

At this stage, the specification document [4] presents the type-A and type-B variables for a file with crystallographic information. Such a file should contain (at least) three global attributes, seven variables and five dimensions. The type-A variables are “primitive_vectors”, “reduced_symmetry_matrices”, “reduced_symmetry_translations”, “space_group”, “atom_species”,“reduced_atom
and one among “atomic_numbers”, “atom_species_names”, “chemical_symbols”

After presenting the case of files with crystallographic information, the specification goes on with density and/or potential information. The density specification was fully described in Section III of the present article. The specification for different types of potentials (Hartree, Kohn-Sham, exchange-correlation) is very similar.

The last type of content of a NQ/ETSF NetCDF file concerns wavefunctions. The specification considers that such a file should contain enough information for one to be able to construct a density from it. Also, since the eigenvalues are intimately linked to eigenfunctions, it is expected that such a file contains eigenvalues. Of course, files might contain less information than that required above but still follow the naming conventions of NQ/ETSF. At least the above-mentioned considerations lead to a consistent choice of mandatory (type-A) variables. Such a file should contain at least three global attributes, twelve variables, and eleven dimensions. We will not describe in detail this last type of file in this overview.

5 Summary and discussion

A full specification for files with selected content related to electronic structure and crystallographic data has been defined, that relies on the NetCDF library. This specification inherits naturally of all the interesting properties of NetCDF-based files, in particular portability and extensibility. It is designed for both serial and parallel usage.

Several software in the Nanoquanta context can produce or read this file format. In order to further encourage its use, a library of Fortran routines [5] has been set up, and is available under the GNU LGPL licence. This library introduces an API with three levels of access. The lowest level is a wrapper around NetCDF calls to be able to call with one routine the commonly associated NetCDF calls (including dimension checks during read or write access). This level is independent of the NQ/ETSF specifications. Based on this low-level API, the group level is defined to give access to groups of variables matching the NQ/ETSF specifications. This group level gives an easy-to-use API to NQ/ETSF files with transparent handling of unit conversions... It is also designed to avoid as much memory copy as possible with a mapping between NQ/ETSF variable definitions and internal variables of user programs. Finally a third level API is available with incorporated routines such as validity checkers.

These API handle all the specifications (in their current version 2.1), including attributes and file splitting for parallel runs. An implementation in Fortran90 of these API has been done and is available on the Web [5]. A concurrent implementation in C is also scheduled.

Although the present version 2.1 of the specification allows already effective communication between different software, there is ample room for generalizations, along at least two different axes: (1) some other heavy numerical quantities might be specified, e.g. dielectric matrices that appear in the GW approximation, (2) basis sets beyond a regular 3D grid or a sphere of planewaves, naturally associated with norm-conserving pseudopotentials, might be supported. Concerning the latter axis, we are considering the definition of a support for wavelets in version 2.2 of the specification.

6 Acknowledgments

We thank the participants to the Louvain-la-neuve mini-workshop held in November 2005, at which a first instance of the density specification was outlined, especially P. Giannozzi, A. Marini, G. Bussi, A. Castro, M. Giantomassi, A. Incze, F. Jollet, J. J. Mortensen, M. Oliveira, S. Pesant, L. E. Ramos, G.-M. Rignanese, R. Shaltaf, F. Sottile, M. Stankovski. We also acknowledge the help, interest and encouragements of several Nanoquanta members including R.W. Godby, L. Reining, A. Rubio, T. Patman, G. Bruant, C. Hogan. This work was carried out in the Nanoquanta EU Network of Excellence (NMP4-CT-2004-500198). MJV acknowledges support from Marie Curie fellowship MANET (MEIF-CT-2005-024152).

7 Appendix

Nanoquanta is a Network of Excellence funded by the European Commission's Sixth Framework Programme (FP6). Operating from June 1st, 2004 to May 31st, 2008 and consisting in 10 nodes and over 100 researchers, Nanoquanta integrates and develops the research capabilities of ten European teams in the field of the fundamental science of nanoscale systems and advanced materials (especially related to electronic spectroscopy and/or involving many-body perturbation theory), exploiting the powerful combination of quantum-mechanical theory and computer simulation to make contact with experimental studies in nanoscience and also directly with technologically relevant electronic, dynamic and optical processes. The Network's final result is the European Theoretical Spectroscopy Facility (ETSF), which will have strong links with a wide range of research groups and will collaborate with users from across science and industry in projects related to the spectroscopy of nanoscale systems and advanced materials. Further information can be obtained from the `http://www.etsf.eu/` Web site .

References

[1] R.K. Rew and G. P. Davis, "NetCDF: An Interface for Scientific Data Access", IEEE Computer Graphics and Applications, Vol. 10, No. 4, pp. 76-82, July 1990.

[2] URL: `http://www.unidata.ucar.edu/software/netcdf`.

[3] X. Gonze, G. Zerah, K.W. Jakobsen, and K. Hinsen. Psi-k Newsletter 55, February 2003, pp 129-134 (URL: `http://psi-k.dl.ac.uk`).

[4] URL: `http://www.etsf.eu/fileformats`.

[5] URL : `http://www.etsf.eu/etsf\_io`.

[6] The ETSF_IO library is also available in the ABINIT software, URL : `http://www.abinit.org`. Dowload the ABINIT package, the ETSF_IO library is located in lib/etsf_io .

[7] URL: `http://lepes.grenoble.cnrs.fr/theorie/olevano/dp/license/download/ioetsf.f90`

[8] X. Gonze, C.-O. Almbladh, A. Cucca, D. Caliste, C. Freysoldt, M. Marques, V. Olevano, Y. Pouillon, and M.J. Verstraete, unpublished .

[9] D. Caliste, Y. Pouillon, M. J. Verstraete, V. Olevano, and X. Gonze, unpublished.

[10] URL: `http://hdf.ncsa.uiuc.edu`.

[11] URL: `http://physics.nist.gov/cuu/Constants/index.html`.

[12] See URL: http://www.unidata.ucar.edu/software/netcdf/docs/faq.html\#filename.