

---

# **h5cube File Specification**

***Release All Versions***

**Brian Skinn**

**21 Jul 2017**



---

## Contents

---

<b>1</b>	<b>Contents</b>	<b>3</b>
1.1	Gaussian CUBE File Format . . . . .	3
1.2	h5cube File Specifications . . . . .	6
1.3	References . . . . .	9
	<b>Bibliography</b>	<b>11</b>



Gaussian **CUBE** files are a common format for storing molecular geometric and volumetric field data from quantum/computational chemical calculations. The data in these files is stored as plain text, and thus their compressibility by standard tools such as `gzip` and `bzip2` is limited, even in cases where the numerical structure of the dataset itself might be well suited for greater compression by other means.

It is the purpose of this document to define a file specification for storing the data contained in Gaussian **CUBE** files in the binary **HDF5** format. HDF5 was chosen due to its acceptance across numerous disciplines as a standardized, cross-platform data storage format, and for the free, cross-platform compression algorithms integrated into it (see [here](#)). In circumstances where ‘semi-lossy’ compression (e.g., truncation of precision and/or data thresholding) is acceptable, particularly large reductions in file size are feasible. Documentation at the companion Python project `h5cube` ([ReadTheDocs](#) | [GitHub](#)) will eventually illustrate representative compression factors achievable in the **h5cube** file format; preliminary data can be found at this [Google Spreadsheet](#).

Definition of an explicit specification for the **h5cube** format is anticipated to facilitate direct, cross-platform binary read and write of **CUBE** data. Particular advantages should be observed by applications aware of the HDF5 format and of this specification when reading data, gained from rapid retrieval of individual data points or subsets directly from **h5cube** files without the need to load the full dataset. Even in instances where the full dataset must be loaded into memory, the disk usage will still be significantly reduced in most cases, since it should be unnecessary to recreate an intermediate uncompressed **CUBE** file.

The Gaussian **CUBE** file format itself is also delineated [here](#), using a “field”-style syntax for convenient cross-reference from the various version(s) of the **h5cube** specification.

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC 2119](#). The specification versioning follows in the spirit of [Semantic Versioning](#); see the [root specifications page](#) for more information.



## Gaussian CUBE File Format

### Disclaimer

The CUBE file format as described here is **NOT** an official specification, sanctioned by Gaussian, Inc. It is instead a best effort to define the contents of a representative subset of CUBE files in circulation. **FILES FORMATTED TO THIS SPECIFICATION MAY NOT BE COMPATIBLE WITH ALL SOFTWARE SUPPORTING CUBE FILE INPUT.**

### Overview

The CUBE file format is described on the Gaussian webpage as part of the documentation of the `cubegen` utility [Gau16]. As noted there, **all data** in CUBE files **MUST** be stored in atomic units (electrons and Bohrs, and units derived from these).

The format specification on the webpage of the VMD visualization program [UIUC16] provides a cleaner layout of one possible arrangement of CUBE file contents. In particular, the Gaussian specification is ambiguous about whitespace requirements, so parsing of CUBE files **SHOULD** accommodate some variation in the format, including (i) variable amounts/types of whitespace between the values on a given line, and (ii) the presence of leading and/or trailing whitespace on a given line.

The CUBE file format as laid out below uses tagged fields (`{FIELD (type)}`) to indicate the types of the various data elements and where they are located in the file. Descriptions of the fields are provided below the *field layout*. Lowercase algebraic symbols ( $x, y, z$ ) indicate coordinates in the frame of the molecular geometry, whereas uppercase algebraic symbols ( $X, Y, Z$ ) indicate coordinates in the voxel grid defined by `{XAXIS}`, `{YAXIS}`, and `{ZAXIS}`.

All fields except for `{DSET_IDS}` and `{NVAL}` **MUST** be present in all files.

`{DSET_IDS}` **MUST** be present if `{NATOMS}` is negative; it **MUST NOT** be present if `{NATOMS}` is positive.

`{NVAL}` **MAY** be omitted if its value would be equal to one; it **MUST** be absent or have a value of one if `{NATOMS}` is negative.

## Field Layout

```
{COMMENT1 (str)}
{COMMENT2 (str)}
{NATOMS (int)} {ORIGIN (3x float)} {NVAL (int)}
{XAXIS (int) (3x float)}
{YAXIS (int) (3x float)}
{ZAXIS (int) (3x float)}
{GEOM (int) (float) (3x float)}
.
.
{DSET_IDS (#x int)}
.
.
{DATA (#x scinot)}
.
.
```

## Table of Contents

*{COMMENT1}* and *{COMMENT2}* *{NATOMS}* *{ORIGIN}* *{NVAL}* *{XAXIS}* *{YAXIS}* *{ZAXIS}* *{GEOM}*  
*{DSET\_IDS}* *{DATA}*

## Field Descriptions

### {COMMENT1 (str)} and {COMMENT2 (str)}

Two lines of text at the head of the file. Per VMD [\[UIUC16\]](#), by convention *{COMMENT1}* is typically the title of the system and *{COMMENT2}* is a description of the property/content stored in the file, but they MAY be anything. For robustness, both of these fields SHOULD NOT be zero-length. As well, while there is no defined maximum length for either of these fields, both SHOULD NOT exceed 80 characters in length.

### {NATOMS (int)}

The absolute value of this first field on the third line indicates the number of atoms  $N_A$  present in the system. A negative value indicates the CUBE file MUST contain the *{DSET\_IDS}* line(s); a positive value indicates the file MUST NOT contain this/these lines.

The value of  $N_A$  also specifies the number of rows of molecular geometry data that MUST be present in *{GEOM}*.

The CUBE specification is silent as to whether a zero value is permitted for *{NATOMS}*; regardless, it is probable that many applications **do not** support CUBE files with no atoms. Accordingly, this specification hereby declares that *{NATOMS}* MUST be nonzero.

### {ORIGIN (3x float)}

This set of three fields defines the displacement vector from the geometric origin of the system (0, 0, 0) to the reference point  $(x_0, y_0, z_0)$  for the spanning vectors defined in *{XAXIS}*, *{YAXIS}*, and *{ZAXIS}*.

### {NVAL (int)}

If *{NATOMS}* is positive, this field indicates the number of data values  $N_V$  that are recorded at each point in the voxel grid; it MAY be omitted, in which case a value of one is assumed.

If *{NATOMS}* is negative, this field MUST be either absent or have a value of one.



**{XAXIS (int) (3x float)}**

The first field on this line is an integer indicating the number of voxels  $N_X$  present along the  $X$ -axis of the volumetric region represented by the **CUBE** file. This value **SHOULD** always be positive; whereas the *input* to the `cubegen` [Gau16] utility allows a negative value here as a flag for the units of the axis dimensions, in a **CUBE** file distance units **MUST always** be in Bohrs, and thus the ‘units flag’ function of a negative sign is superfluous. It is prudent to design applications to handle gracefully (viz., disregard the sign of) a negative value here, however.

The second through fourth values on this line are the components of the vector  $\vec{X}$  defining the voxel  $X$ -axis. As noted in the Gaussian documentation [Gau16], the voxel axes need neither be orthogonal nor aligned with the geometry axes. However, many tools only support voxel axes that **are** aligned with the geometry axes (and thus are also orthogonal). In this case, the first `float` value ( $X_x$ ) will be positive and the other two ( $X_y$  and  $X_z$ ) will be identically zero.

**{YAXIS (int) (3x float)}**

This line defines the  $Y$ -axis of the volumetric region of the **CUBE** file, in nearly identical fashion as for **{XAXIS}**. The key differences are: (1) the first integer field  $N_Y$  **MUST** always be positive; and (2) in the situation where the voxel axes aligned with the geometry axes, the second `float` field ( $Y_y$ ) will be positive and the first and third `float` fields ( $Y_x$  and  $Y_z$ ) will be identically zero.

**{ZAXIS (int) (3x float)}**

This line defines the  $Z$ -axis of the volumetric region of the **CUBE** file, in nearly identical fashion as for **{YAXIS}**. The key difference is that in the situation where the voxel axes are aligned with the geometry axes, the third `float` field ( $Z_z$ ) will be positive and the first and second `float` fields ( $Z_x$  and  $Z_y$ ) will be identically zero.

**{GEOM (int) (float) (3x float)}**

*This field **MUST** have  $N_A$  rows of the below composition.*

Each row of this field provides atom identity and position information for an atom in the molecular system of the **CUBE** file:

- (`int`) - Atomic number of atom  $a$
- (`float`) - Nuclear charge of atom  $a$  (will deviate from the atomic number when an ECP is used)
- (`3x float`) - Position of the atom in the geometric frame of reference ( $x_a, y_a, z_a$ )

**{DSET\_IDS (#x int)}**

*This field is only present if **{NATOMS}** is negative*

This field comprises one or more rows of integers, representing identifiers associated with multiple **{DATA}** values at each voxel, with a total of  $m + 1$  values present. The most common meaning of these identifiers is orbital indices, in **CUBE** files containing wavefunction data. The first value **MUST** be positive and equal to  $m$ , to indicate the length of the rest of the list. Each of these  $m$  values may be any integer, with the constraint that all values **SHOULD** be unique. Further, all  $m$  values **SHOULD** be non-negative, as unpredictable behavior may result in some applications if negative integers are provided.

**{DATA (#x scinot)}**

This field encompasses the remainder of the **CUBE** file. Typical formatted **CUBE** output has up to six values on each line, in whitespace-separated scientific notation. Non-numeric data values are **not** supported and **MUST NOT** be present.

If **{NATOMS}** is positive, a total of  $N_X N_Y N_Z N_V$  values should be present, flattened as follows (in the below Python pseudocode the for-loop variables are iterated starting from zero):

```
for i in range(NX):
    for j in range(NY):
        for k in range(NZ):
            for l in range(NV):

                write(data_array[i, j, k, l])
                if (k*NV + l) mod 6 == 5:
                    write('\n')

            write('\n')
```

If  $\{NATOMS\}$  is negative and  $m$  datasets are present (see  $\{DSET\_IDS\}$  above), a total of  $N_X N_Y N_Z m$  values should be present, flattened as follows:

```
for i in range(NX):
    for j in range(NY):
        for k in range(NZ):
            for l in range(m):

                write(data_array[i, j, k, l])
                if (k*m + l) mod 6 == 5:
                    write('\n')

            write('\n')
```

The sequence of the data values along the last ( $l$ ) dimension of the data array for each  $i, j, k$  MUST match the sequence of the identifiers provided in  $\{DSET\_IDS\}$  in order for the dataset to be interpreted properly.

Regardless of the sign of  $\{NATOMS\}$ , as illustrated above a newline is typically inserted after the block of data corresponding to each  $(X_i, Y_j)$  pair.

## h5cube File Specifications

This page collates all existing versions of the **h5cube** file specification for convenient access. The versioning scheme used here follows the spirit of [Semantic Versioning](#) but in a revised syntax:

- Specification version numbers are of the form  $vx.yrev$ , where  $x$  and  $y$  indicate *major* and *minor* categories of changes, and  $rev$  is a *revision number*.
- An increment in the *revision number* indicates minor editorial fix(es), such as correcting typos or introducing clarification(s) that do not affect the semantic content of the specification.
- An increment in the *minor* version level indicates that new field(s) or other semantic content have been added to the specification, but existing fields/content are unchanged.
- An increment in the *major* version level indicates that existing field(s)/content have been removed/changed. New field(s)/content may also have been added.

Based upon the above, it is expected that applications built against the specific version  $vX.Y$  should be compatible with the versions  $vx.y$  where  $x = X$  and  $y \geq Y$ . Applications *may or may not* be compatible with versions  $x > X$ , depending on the particular changes introduced with that *major* version increment.

## Specification Versions

## h5cube Specification v1.0 - Description

Only scalar volumetric data... *and other commentary*

---

### Todo

Complete this description.

---

Test substitution of autogen spec dataset reference: *[GEOM]<sub>v1.0r1</sub>*

## h5cube Specification v1.0 rev1

### Datasets

*[VERSION] [COMMENT1] [COMMENT2] [NATOMS] [ORIGIN] [XAXIS] [YAXIS] [ZAXIS] [GEOM]  
[NUM\_DSETS] [DSET\_IDS] [SIGNS] [LOGDATA]*

### Dataset Descriptions

#### [VERSION]

*Integer2,*

**h5cube** specification version met by the file, where the first and second elements are the major and minor version numbers, respectively:

$$vx.y \rightarrow (x, y)$$

This dataset MAY be absent for specification v1.0 **only**.

#### [COMMENT1]

*String*

First comment line of the **CUBE** file. Corresponds to *{COMMENT1}*.

#### [COMMENT2]

*String*

Second comment line of the **CUBE** file. Corresponds to *{COMMENT2}*.

#### [NATOMS]

*Integer*

Corresponds directly to *{NATOMS}*, where the absolute value equals  $N_A$ , the number of atoms in the system. The value here may be negative, with the same semantic implications as in the case of *{NATOMS}*. See *[NUM\_DSETS]* and *[DSET\_IDS]* for more details.

This value **MUST** be nonzero.

#### [ORIGIN]

*Float3,*

Vector pointing from the origin of the system geometry frame to the reference point  $(x_0, y_0, z_0)$  of the vectors spanning the **CUBE** voxel grid. Corresponds directly to *{ORIGIN}*.

#### [XAXIS]

*Float4*,

Corresponds directly to *{XAXIS}*. The first element is the number of voxels along the  $X$ -axis of the volumetric grid,  $N_X$ , and MUST be a positive integer value, despite the *Float* type of the dataset. The remaining three elements are the vector  $\vec{X}$  defining the voxel  $X$ -axis. See *{XAXIS}* for more information about the semantics of these values.

#### [YAXIS]

*Float4*,

Corresponds directly to *{YAXIS}*. The first element is the number of voxels along the  $Y$ -axis of the volumetric grid,  $N_Y$ , and MUST be a positive integer value, despite the *Float* type of the dataset. The remaining three elements are the vector  $\vec{Y}$  defining the voxel  $Y$ -axis. See *{YAXIS}* for more information about the semantics of these values.

#### [ZAXIS]

*Float4*,

Corresponds directly to *{ZAXIS}*. The first element is the number of voxels along the  $Z$ -axis of the volumetric grid,  $N_Z$ , and MUST be a positive integer value, despite the *Float* type of the dataset. The remaining three elements are the vector  $\vec{Z}$  defining the voxel  $Z$ -axis. See *{ZAXIS}* for more information about the semantics of these values.

#### [GEOM]

*Float* $N_A$ , 5

Corresponds directly to *{GEOM}*. The first element of each of the  $N_A$  lines (see *{NATOMS}*) indicates the atomic number of atom  $a$ , and MUST be an *Integer* value. The second element of each line indicates the nuclear charge of atom  $a$ , and will generally be (i) equal to the atomic number and (ii) an integer quantity. This value *will* deviate from the atomic number when an ECP *[WP\_PP]* is used on atom  $a$ .

The remaining three *Float* elements of each line provide the coordinates  $(x_a, y_a, z_a)$  of atom  $a$  in the geometric frame of reference.

#### [NUM\_DSETS]

*Integer*

Corresponds to the first value in *{DSET\_IDS}*, indicating the number  $m$  of dataset identifiers provided in the remainder of *{DSET\_IDS}*. This value  $m$  also specifies the required size of *[DSET\_IDS]*.

#### [DSET\_IDS]

*Integer* $m$ ,

An array of  $m$  values indicating the identifiers associated with the multiple data values provided at each voxel in *[SIGNS]* and *[LOGDATA]*.

As noted in *{DSET\_IDS}*, each of these values SHOULD be unique within the array, and SHOULD be non-negative. Otherwise, they can be any *Integer* value.

#### [SIGNS]

*[NATOMS]* > 0 *Integer* $N_X, N_Y, N_Z$  *[NATOMS]* < 0 *Integer* $N_X, N_Y, N_Z, m$

This dataset combines with *[LOGDATA]* to define the value of the volumetric data at each voxel  $(X, Y, Z)$ . This dataset contains the arithmetic signs of the data values, as per the standard mathematical *signum* () function *[WP\_Sign]*. Thus, if *[NATOMS]* > 0:

$$[SIGNS]_{X,Y,Z} = [\Phi(X, Y, Z)]$$

and if *[NATOMS]* < 0:

$$[SIGNS]_{X,Y,Z,i} = [\Phi_i(X, Y, Z)]$$

where  $\Phi_i$  is the  $i^{\text{th}}$  dataset included in the **CUBE** file.

#### [LOGDATA]

$$[NATOMS] > 0 \text{Float} N_X, N_Y, N_Z \quad [NATOMS] < 0 \text{Float} N_X, N_Y, N_Z, m$$

This dataset combines with *[SIGNS]* to define the value of the volumetric data at each voxel  $(X, Y, Z)$ .

This dataset contains the common logarithms *[WP\_log10]* of the data values. Thus, if *[NATOMS]* > 0:

$$[LOGDATA]_{X,Y,Z} = \log_{10} [\Phi(X, Y, Z)]$$

and if *[NATOMS]* < 0:

$$[LOGDATA]_{X,Y,Z,i} = \log_{10} [\Phi_i(X, Y, Z)]$$

where  $\Phi_i$  is the  $i^{\text{th}}$  dataset included in the **CUBE** file.

## Syntax and Conventions

The HDF5 dataset names defined by these specifications are formatted in fixed-font, with surrounding square brackets (e.g., `[GEOM]`). This distinguishes them from the fields in the **CUBE file specification**, which use curly braces (e.g., `{GEOM}`). All of the dataset names are strings, and thus to access the `[GEOM]` dataset in Python, use code like the following:

```
>>> import h5py as h5
>>> hf = h5.File('file.h5cube')
>>> hf['GEOM'].value
array([ ... ])
```

#### Todo

- Description of the syntax/formatting of the dataset descriptions residing in the subpages here. In particular, a brief overview of the HDF5 datatypes and the ‘lumped’ approach taken in the spec layouts. (Fig. 6-4, Tbl. 6-1).
- Python tuple syntax for dataset dimensions.
- The data type provided on the first line under a given dataset name is the data type it must be present as in an **h5cube** file.
- The formula for generating the non-logscale data from the *[SIGNS]* and *[LOGDATA]*.

## References



---

## Bibliography

---

- [Bou03] Bourke, P. “Gaussian Cube Files.” Dec 2003. Online resource: <http://paulbourke.net/dataformats/cube/>. Accessed 11 Dec 2016.
- [Gau16] “cubegen.” Website of the Gaussian program package. <http://gaussian.com/cubegen/>. Accessed 8 Feb 2017.
- [HDF5-6] “6. HDF5 Datatypes.” Web version of the HDF5 User’s Guide. <https://support.hdfgroup.org/HDF5/....> Accessed 15 Mar 2017.
- [UIUC16] “Cube Plugin, Version 1.1.” Website of the Theoretical and Computational Biophysics Group, University of Illinois at Urbana-Champaign. <http://www.ks.uiuc.edu/Research/vmd/....> Accessed 11 Dec 2016.
- [WP\_log10] “Common logarithm.” Wikipedia, The Free Encyclopedia. <https://en.wikipedia.org/w/....> Link is to article version as of 19 Jun 2017, accessed 20 Jul 2017.
- [WP\_PP] “Pseudopotential.” Wikipedia, The Free Encyclopedia. <https://en.wikipedia.org/w/....> Link is to article version as of 26 Jun 2017, accessed 20 Jul 2017.
- [WP\_Sign] “Sign function.” Wikipedia, The Free Encyclopedia. <https://en.wikipedia.org/w/....> Link is to article version as of 28 Apr 2017, accessed 10 May 2017.





## R

RFC

RFC 2119, 1