

# Appending column based ascii data to CDL format file

This text is a user manual to python script, that is capable to transform column based ascii data into CDL format for further conversion into netcdf (for example with ncgen)

To append data into cdl file, following steps are required:

- 1) create column based **ASCII datafile**
- 2) create a **CDL** syntax file, describing netcdf file to be created
- 3) create a **metafile**, with description of data
- 4) place **ASCII datafile** and **metafile** together in one folder
- 5) run the **script** “make\_cdl.py” or “make\_cdl\_gui.py”

## Contents:

1. ASCII datafile
2. Metafile
3. CDL format file
4. Script
5. Example of using script

# 1. ASCII datafile

User should consider using this script with gridded data (because behind the code, data is stored within an array, created by multiplication of “dimension” vectors). Nevertheless script will work with non-gridded data too, inserting it into non-uniform rectangular grid (in example 1 below data is stored in non-uniform 2D grid). The data should be 1D, 2D or 3D (see example 2 below)

Data is expected to be stored in columns. Columns order is not important. There is no need to sort values in columns anyhow. Each row of data, representing one measurement, should be placed on a single line. Values within the line can be separated with any symbol (except “=” and “\*”) or whitespace. Commented lines are not allowed within data body. Any additional information may be placed in multiline header, that will be later ignored.

**Example 1:** ascii file consist of 3 columns data. Created vectors <X>, <Y>. Data <Z> is stored in 2d array with shape (x,y)=(2,3)

x,y;z		
-----		
147000;6194000;52.8703	=>	X [147000 147250]
147000;6194250;52.9353	z(x,y)	
147250;6194000;52.803	=>	Y [6194000 6194250 6195000]
147250;6194250;52.8681		Z [52.8703 52.803 52.9353 52.8681 61.0903 68.5771]
147000;6195000;61.0903		
147250;6195000;68.5771		

**Example 2:** ascii file consist of 5 columns data. Created vectors <X>, <Y>, <class>. Data is stored in 3d array of shape (x,y,class)=(2,2,3). Dataclass1, Dataclass2, Dataclass3 is (:,:,1), (:,:,2), (:,:,3) respectively

X,Y,class1,class2,class3		
300250,5920500,49.1,11.59,14.19	=>	X [300250 300500]
300250,5920750,50.96,12,14.01	(x,y,class)	Y [5920500 5920750]
300500,5920500,52.83,12.46,13.85	=>	class [class1 class2 class3]
300500,5920750,53.88,12.75,13.78		Dataclass1 [49.1 52.83 50.96 53.88]
		Dataclass2 [11.59 12.46 12 12.75]
		Dataclass3 [14.19 13.85 14.01 13.78]

## 2. Metafile

Metafile is an ASCII file, containing information to interpret data correctly. File structure: Each parameter should be written on its own string. Spaces, tabs, and newlines can be used freely for readability. Comments may follow character '#' on any line. Symbol "=" is considered to be special symbol and may only be used once per line to set parameter value.

Parameters:

***dataFname*** - a string, containing name of the data-file which is described by current metafile. Only name, omitting path should be inserted. Later on, path will be taken from this Metafile (there comes the requirement to put metafile and datafile in the same folder).

**Example:** *dataFname* = *my\_data.csv*

***skiprows*** - integer, which forces to ignore first *skiprows* number of lines in data-file, useful to ignore header.

**Example :** *skiprows* = 2

***delimiter*** - string used to separate values. If *None* is set - delimiter is any whitespace. Delimiter cannot be "=" or "#".

**Example 1:** *delimiter* = *None*

**Example 2:** *delimiter* = ;

***columnDataNames*** - list of comma separated strings defining names of the columns declared in file *dataFname*. The number of string in given list should be equal to number of columns in *dataFname*, because each item in *columnDataNames* list describes column in *dataFname* having same index. Names, declared in this list should match corresponding names of the variables in CDL syntax file.

**Example:** *columnDataNames* = *GridX, GridY, bathymetry\_Z*

Additionally, when a column in ASCII datafile contains values of array with known index for one (!) dimension, following syntax may be used: *<spm(GridX; GridY; 0)>*. Currently this feature is implemented only for 3D and 3D-alike (see section CDL) arrays. Dimensions in brackets should match corresponding dimensions declared in CDL file, but in contrast to CDL are separated with semicolon ";". One-size or unlimited dimensions must be omitted and not included into brackets in Metafile. It is not necessary to preserve order: name with higher index may stand in front name with lower index. Below is an example of a namelist describing spm data for 3 different sediment classes stored in a X, Y grid, that will be stored in three dimensional array, where first two dimensions are grid, and the third (has size of 3) is a spm class.

**Example :** *columnDataNames* = *X, Y, spm(X; Y; 0), spm( X;Y;2), spm( X;Y;1)*

### 3. CDL format file

The script is capable of reading of netcdf3 CDL syntax files (reading of any groups or compound data types is not supported). User should create a standard CDL file, and mark information which is unknown with star symbol “\*”. Examples are given below.

#### Marking unknown size of dimensions and unknown data.

Unknown sizes of dimensions and data should be specified with star symbol “\*”, pointing script, which data should be read from other file. Only one star symbol is allowed per line. Please note, that unknown information can only be located within *<dimensions:>* or *<data:>* section.

#### Dimension limitations

Whenever user specifies an unknown variable, he should remember that only 1D, 2D and 3D data arrays are supported (array of shape (100, 100, 1, 1, 1, 1, 1) or for example (1, 10, 1, 10) are treated as 2D arrays; (200, 300, 12) and (1, 200, 300, 12) are both 3D arrays).

#### Unlimited dimensions

Unlimited dimensions are treated as dimensions of size one (for filling data), and data will be stored at first index of dimension time. In example below, variable *<z>* has a 3D shape, and its first dimension *<time>* is a unlimited dimension. In the code , this array will be treated as array of shape (1, 1000, 1500)

#### Example:

```
netcdf foo {
dimensions:
    time = unlimited;
    GridX = 1000;
    GridY = 1500;
variables:
    float z(time, GridX, GridY);
    float GridX(GridX);
    float GridY(GridY);

    time:units = "seconds";
    z:units = "meters";
    z:valid_range = 0., 5000.;
    z:testing_param = 434;
data:
    GridX = *;
    GridY = * ;
    z = *;
}
```

Make sure to check correct dimensioning of your variable with unknown data. Let's look at the CDL example below. A variable `<bathymetry(GridX, GridY)>` is initialized, which has 2 dimensions. Therefore this input is valid. Both dimensions `<GridX>`, `<GridY>` are unknown. Data points are unknown as well for this variable, and marked with `"**"`.

**Example:**

```
netcdf foo {  
  
    dimensions:  
        GridX = * ;  
        GridY = * ;  
  
    variables:  
        float bathymetry(GridX, GridY);  
        float GridX(GridX);  
        float GridY(GridY);  
  
        GridX: units = "meters";  
        GridX: units = "meters";  
        bathymetry:long_name = "bed elevation in meters below MSL";  
        bathymetry:units = "meters";  
        bathymetry:_FillValue = -1.;  
        :global_attr = "cdl with unknown dimensions" ;  
  
    data:  
        GridX = *;  
        GridY = * ;  
        bathymetry = *;  
  
}
```

## 4. Script

Script copies passed CDL file line by line, looking for undefined data marked with “\*” (see CDL section). If undefined data has been found, reads parameters from Metafile, based on them reads information from ascii file into memory and finally writes it into the file. Resulting file is created within script directory, and has old name with an underscore at the beginning (*myfile.cdl* >>> *\_myfile.cdl*). Metafile should be placed in same directory as ascii datafile.

### Usage

Python 2.7 is required.

Copy files “make\_cdl.py”, “make\_cdl\_gui.py” and directory “lib” with all content. Run one of the files. Two versions of scripts are available: “make\_cdl.py” and “make\_cdl\_gui.py”. First one has to be run via command line passing following input arguments:

-h	for help (optional)
-i <input CDL filename>	path to source CDL file
-m <input meta file>	path to source Metafile
-l	enable console log (optional)

Example Linux:

```
$ python make_cdl.py -i data/myfile.cdl -m metafile.txt -l
```

Example Windows:

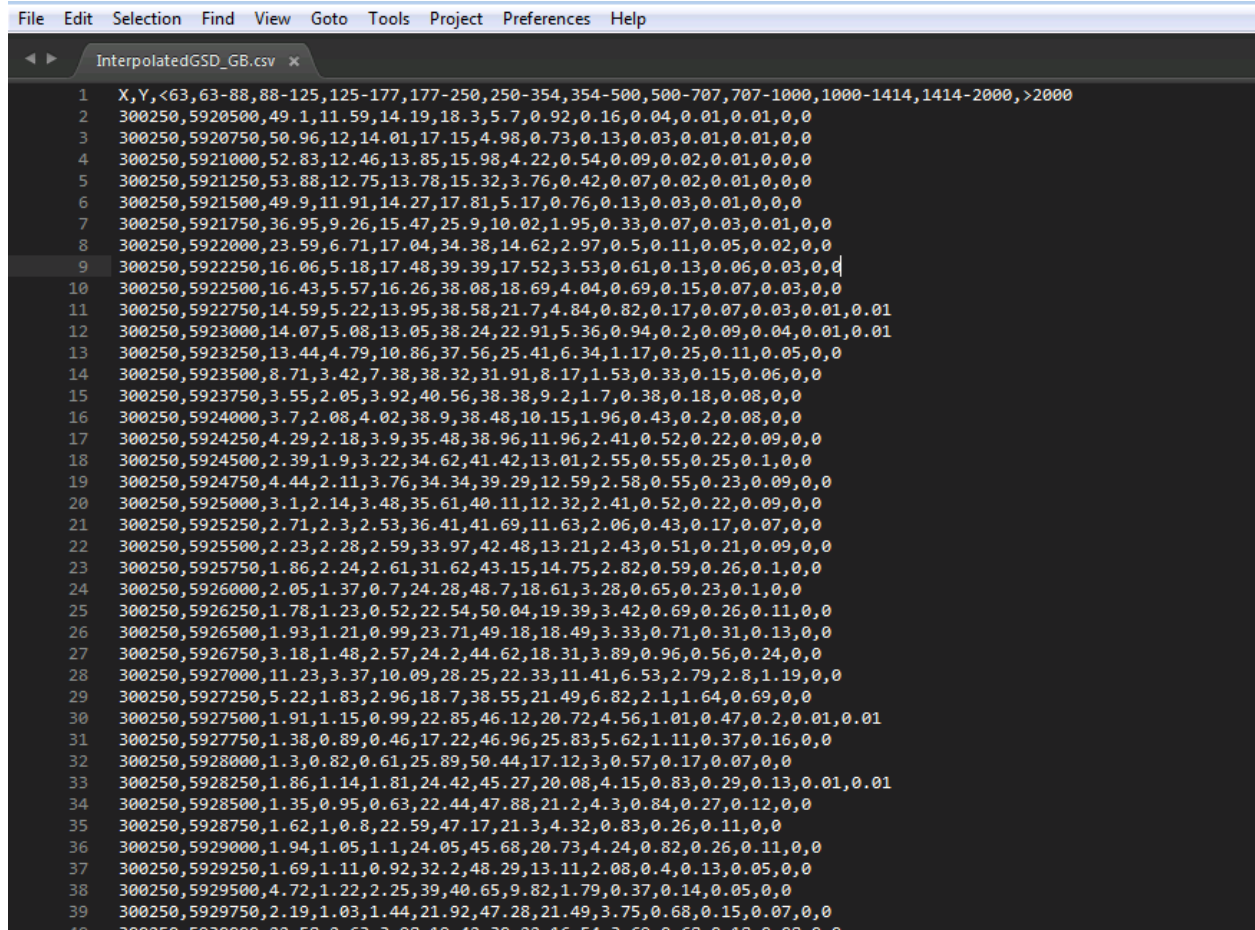
```
> python make_cdl.py -i data/myfile.cdl -m metafile.txt -l
```

A pointer to environmental variable “python” is specified for Linux platforms as */usr/bin/python* at line 1 of script “make\_cdl.py”

For running a gui version, execute “make\_cdl\_gui.py”, select paths for CDL and Metafile, click “Create New CDL” button. If the script finishes successfully a pop-up message box appears. Error messages are printed in console window.

## 5. Example of using script

### Step 1) Open ASCII data file



```
File Edit Selection Find View Goto Tools Project Preferences Help
InterpolatedGSD_GB.csv x
1 X,Y,<63,63-88,88-125,125-177,177-250,250-354,354-500,500-707,707-1000,1000-1414,1414-2000,>2000
2 300250,5920500,49.1,11.59,14.19,18.3,5.7,0.92,0.16,0.04,0.01,0.01,0,0
3 300250,5920750,50.96,12,14.01,17.15,4.98,0.73,0.13,0.03,0.01,0.01,0,0
4 300250,5921000,52.83,12.46,13.85,15.98,4.22,0.54,0.09,0.02,0.01,0,0,0
5 300250,5921250,53.88,12.75,13.78,15.32,3.76,0.42,0.07,0.02,0.01,0,0,0
6 300250,5921500,49.9,11.91,14.27,17.81,5.17,0.76,0.13,0.03,0.01,0,0,0
7 300250,5921750,36.95,9.26,15.47,25.9,10.02,1.95,0.33,0.07,0.03,0.01,0,0
8 300250,5922000,23.59,6.71,17.04,34.38,14.62,2.97,0.5,0.11,0.05,0.02,0,0
9 300250,5922250,16.06,5.18,17.48,39.39,17.52,3.53,0.61,0.13,0.06,0.03,0,0
10 300250,5922500,16.43,5.57,16.26,38.08,18.69,4.04,0.69,0.15,0.07,0.03,0,0
11 300250,5922750,14.59,5.22,13.95,38.58,21.7,4.84,0.82,0.17,0.07,0.03,0.01,0.01
12 300250,5923000,14.07,5.08,13.05,38.24,22.91,5.36,0.94,0.2,0.09,0.04,0.01,0.01
13 300250,5923250,13.44,4.79,10.86,37.56,25.41,6.34,1.17,0.25,0.11,0.05,0,0
14 300250,5923500,8.71,3.42,7.38,38.32,31.91,8.17,1.53,0.33,0.15,0.06,0,0
15 300250,5923750,3.55,2.05,3.92,40.56,38.38,9.2,1.7,0.38,0.18,0.08,0,0
16 300250,5924000,3.7,2.08,4.02,38.9,38.48,10.15,1.96,0.43,0.2,0.08,0,0
17 300250,5924250,4.29,2.18,3.9,35.48,38.96,11.96,2.41,0.52,0.22,0.09,0,0
18 300250,5924500,2.39,1.9,3.22,34.62,41.42,13.01,2.55,0.55,0.25,0.1,0,0
19 300250,5924750,4.44,2.11,3.76,34.34,39.29,12.59,2.58,0.55,0.23,0.09,0,0
20 300250,5925000,3.1,2.14,3.48,35.61,40.11,12.32,2.41,0.52,0.22,0.09,0,0
21 300250,5925250,2.71,2.3,2.53,36.41,41.69,11.63,2.06,0.43,0.17,0.07,0,0
22 300250,5925500,2.23,2.28,2.59,33.97,42.48,13.21,2.43,0.51,0.21,0.09,0,0
23 300250,5925750,1.86,2.24,2.61,31.62,43.15,14.75,2.82,0.59,0.26,0.1,0,0
24 300250,5926000,2.05,1.37,0.7,24.28,48.7,18.61,3.28,0.65,0.23,0.1,0,0
25 300250,5926250,1.78,1.23,0.52,22.54,50.04,19.39,3.42,0.69,0.26,0.11,0,0
26 300250,5926500,1.93,1.21,0.99,23.71,49.18,18.49,3.33,0.71,0.31,0.13,0,0
27 300250,5926750,3.18,1.48,2.57,24.2,44.62,18.31,3.89,0.96,0.56,0.24,0,0
28 300250,5927000,11.23,3.37,10.09,28.25,22.33,11.41,6.53,2.79,2.8,1.19,0,0
29 300250,5927250,5.22,1.83,2.96,18.7,38.55,21.49,6.82,2.1,1.64,0.69,0,0
30 300250,5927500,1.91,1.15,0.99,22.85,46.12,20.72,4.56,1.01,0.47,0.2,0.01,0.01
31 300250,5927750,1.38,0.89,0.46,17.22,46.96,25.83,5.62,1.11,0.37,0.16,0,0
32 300250,5928000,1.3,0.82,0.61,25.89,50.44,17.12,3,0.57,0.17,0.07,0,0
33 300250,5928250,1.86,1.14,1.81,24.42,45.27,20.08,4.15,0.83,0.29,0.13,0.01,0.01
34 300250,5928500,1.35,0.95,0.63,22.44,47.88,21.2,4.3,0.84,0.27,0.12,0,0
35 300250,5928750,1.62,1,0.8,22.59,47.17,21.3,4.32,0.83,0.26,0.11,0,0
36 300250,5929000,1.94,1.05,1.1,24.05,45.68,20.73,4.24,0.82,0.26,0.11,0,0
37 300250,5929250,1.69,1.11,0.92,32.2,48.29,13.11,2.08,0.4,0.13,0.05,0,0
38 300250,5929500,4.72,1.22,2.25,39,40.65,9.82,1.79,0.37,0.14,0.05,0,0
39 300250,5929750,2.19,1.03,1.44,21.92,47.28,21.49,3.75,0.68,0.15,0.07,0,0
40 300250,5930000,2.53,2.63,2.08,10.42,30.22,15.54,2.60,0.68,0.18,0.08,0,0
```

This file (*InterpolatedGSD\_GB.csv*) consist of 14 columns of data, values are separated with commas. First two columns are X and Y coordinates, rest - are values of Grain Size distribution for 12 sediment fractions. There are two possibilities: put this data into twelve 2d arrays of (x,y) shape (option1) or into one 3d array of (x,y,12) shape (option2). Lets make example of both these options. To show how to work with unlimited dimensions, for option 2 we will add <time> dimension, storing our data at 0 timestep.

### Step 2) Create CDL file

Creating, a usual CDL file, and marking unknown dimensions and data with “\*” symbol. In examples below we are omitting most attributes for shortage of report. Of course any attributes can be added as well. Files are saved as <cdl1.cdl> and <cdl2.cdl> for option 1 and 2 respectively.

Option 1.	Option 2.
<pre> netcdf foo {  dimensions:     GridX = * ;     GridY = * ;  variables:     float  GridX(GridX);     float  GridY(GridY);     float  gsd_class1 (GridX, GridY);     float  gsd_class2 (GridX, GridY);     float  gsd_class3 (GridX, GridY);     float  gsd_class4 (GridX, GridY);     float  gsd_class5 (GridX, GridY);     float  gsd_class6 (GridX, GridY);     float  gsd_class7 (GridX, GridY);     float  gsd_class8 (GridX, GridY);     float  gsd_class9 (GridX, GridY);     float  gsd_class10 (GridX, GridY);     float  gsd_class11 (GridX, GridY);     float  gsd_class12 (GridX, GridY);      GridX: units = "meters";     GridY: units = "meters";      gsd_class1 : _FillValue = -999.;     gsd_class2 : _FillValue = -999.;     gsd_class3 : _FillValue = -999.;     gsd_class4 : _FillValue = -999.;     gsd_class5 : _FillValue = -999.;     gsd_class6 : _FillValue = -999.;     gsd_class7 : _FillValue = -999.;     gsd_class8 : _FillValue = -999.;     gsd_class9 : _FillValue = -999.;     gsd_class10: _FillValue = -999.;     gsd_class11: _FillValue = -999.;     gsd_class12: _FillValue = -999.;  // global attributes :info = "cdl with unknown dimensions option 1 (2d arrays)" ;  data:     GridX = *;     GridY = * ;     gsd_class1 = * ;     gsd_class2 = * ;     gsd_class3 = * ;     gsd_class4 = * ;     gsd_class5 = * ;     gsd_class6 = * ;     gsd_class7 = * ;     gsd_class8 = * ;     gsd_class9 = * ;     gsd_class10 = * ;     gsd_class11 = * ;     gsd_class12 = * ; } </pre>	<pre> netcdf foo {  dimensions:     GridX = * ;     GridY = * ;     time = unlimited ;     twelve = 12 ;  variables:     float  gsd(time, twelve , GridX, GridY);     float  GridX(GridX);     float  GridY(GridY);      GridX: units = "meters";     GridX: units = "meters";     gsd:long_name = "grain size distribution. 12 classes &lt;63,63-88,88-125,125-177,177-250,250-354,354-500,500-707,707-1000,1000-14 14,1414-2000,&gt;2000";     gsd:units = "%";     gsd: _FillValue = -999.;  :info = "cdl with unknown dimensions option 2 (3d array)" ;  data:     GridX = *;     GridY = * ;     gsd = * ; } </pre>



### Step 3) Create Metafile

Creating a text file with for keywords: *dataFName* , *skiprows* ,*delimiter* ,*columnDataNames*.

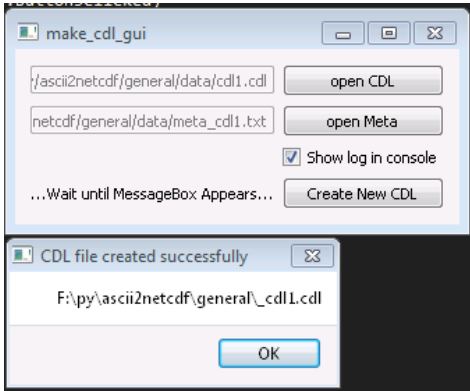
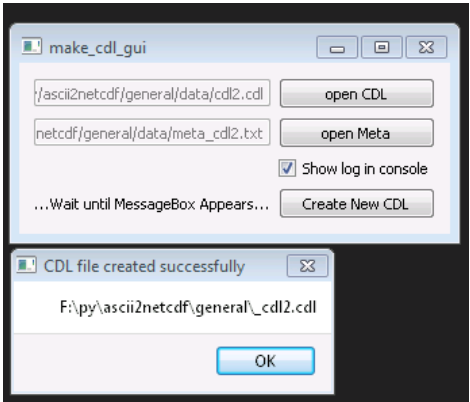
<i>dataFName</i> = InterpolatedGSD_GB.csv	>>> name of our data file.
<i>skiprows</i> = 1	>>> ascii data-file has 1n line header
<i>delimiter</i> = ,	>>> data is separated with comma
<i>columnDataNames</i> = ...	>>> see description in section Metafile

Each keyword should be within one line(!). Thus, files below consist only out of 4 lines. Dont forget to use semicolons for separating dimensions not comma. In Option2 file, variable <gsd> has only three dimensions and not four as in CDL (time is an unlimited dimension, therefore is not included into brackets). Details can be found in metafile section. For better visualization you can copy examples to your favourite text-editor. First file is saved as <meta\_cdl1.txt>, second as <meta\_cdl2.txt>.

Option 1.	Option 2.
<pre>dataFName = InterpolatedGSD_GB.csv skiprows = 1 delimiter = , columnDataNames = GridX, GridY, gsd_class1, gsd_class2 ,gsd_class3 ,gsd_class4, gsd_class5 , gsd_class6 ,gsd_class7 ,gsd_class8 ,gsd_class9 ,gsd_class10 ,gsd_class11 ,gsd_class12</pre>	<pre>dataFName = InterpolatedGSD_GB.csv skiprows = 1 delimiter = , columnDataNames = GridX, GridY, gsd(0; GridX; GridY), gsd(1; GridX; GridY), gsd(2; GridX; GridY), gsd(3; GridX; GridY), gsd(4; GridX; GridY), gsd(5; GridX; GridY), gsd(6; GridX; GridY), gsd(7; GridX; GridY), gsd(8; GridX; GridY), gsd(9; GridX; GridY), gsd(10; GridX; GridY), gsd(11; GridX; GridY)</pre>

### Step 4) Place metafiles together with ascii data <InterpolatedGSD\_GB.csv>

### Step 5) Running the script

Option 1.	Option 2.
	

After running script, two files are created within script directory. It took almost 400 seconds to create a Option1 file (with 12 2d arrays), and 120 seconds for Option2 file (3d array). Despite, larger amount of data in memory (3d array VS. one 2d array) option 2 works faster, because it uses sorting and write-to-file routines only once