Opacplot2 Documentation

Release 1.0

Jordan Laune

January 10, 2017

Contents

Contents:

CHAPTER 1

opacplot2

Python package for manipulating Equation of State (EoS) and Opacity data.

`Opacplot2` comes with an EoS Table conversion tool named `opac-convert`. It also comes with an EoS Table comparison tool named `opac-error`. Both can be found in the *Command Line Tools*

1.1 Dependencies

`opacplot2`'s dependencies include:

- numpy
- six
- tables
- matplotlib
- scipy
- periodictable
- hedp (https://github.com/luli/hedp)

They can be installed as follows:

```
pip install numpy six tables matplotlib scipy periodictable
pip install git+https://github.com/luli/hedp
```

1.2 Installation

This module requires Python 2.7 or 3.5. The latest version can be installed with:

```
pip install git+https://github.com/flash-center/opacplot2
```

If you have the Propaceos Python reader, in order to include it in the installation, you must install `opacplot2` as follows:

```
git clone https://github.com/flash-center/opacplot2
cp /path/to/opg_propaceos.py opacplot2/opacplot2/
cd opacplot2
python setup.py install
```

1.3 Configuring Matplotlib for `opac-error`

The plotting capabilities of `opacplot2` rely on the `matplotlib` python module. It can be installed with `pip`. It can also be installed with `conda` if you are using Anaconda Python. For information on how to install `matplotlib`, see their website.

1.3.1 Configuring `matplotlib` to display inline

In order to get plots to display inline with the python interpreter, it is recommended to use `matplotlib` in conjunction with Project Jupyter's QtConsole.

Steps:

1. Install `jupyter` using your preferred Python package manager.
2. Start the console with `jupyter qtconsole`.
3. Set the mode to suit inline `matplotlib` plots using:

```
%matplotlib inline
```

At any point during your Python session on `qtconsole`, you can view what your current plots look like by typing the name of your figure.

1.3.2 Troubleshooting `matplotlib` on OSX

In order to display your plots, `matplotlib` uses a variety of backends. If you have your plots displaying inline on the `qtconsole`, it is using a backend specific to the `qtconsole` that is separate from other standalone backends. Thus, for those of you using `qtconsole`, you can ignore this section. The recommended backend to use for OSX is `macosx`. This can be set in `matplotlib`'s cofiguration file `~/.matplotlib/matplotlibrc` with the line:

```
backend: macosx
```

It can also be set in an interactive session **before** you import `matplotlib.pyplot` with:

```
matplotlib.rcParams['backend'] = 'macosx'
```

For more information on the `matplotlibrc` file, see `matplotlib`'s website.

For Double Implementation Errors

If you are receiving double implementation errors in Python, it is probably due to the backend chosen for `matplotlib`. This error is confirmed for OSX users using the `Tk` backends. To fix this, switch your backend to `macosx` as shown above.

For Framework Errors

From `matplotlib`'s website:

> ''On OSX, two different types of Python Builds exist: a regular build and a framework build. In order to interact correctly with OSX through some GUI frameworks you need a framework build of Python. At the time of writing the macosx, WX and WXAgg backends require a framework build to function correctly. Unfortunately virtualenv creates a non framework build even if created from a framework build of Python. Conda environments are framework builds. From Matplotlib 1.5 onwards the macosx backend checks that

a framework build is available and fails if a non framework build is found. WX has a similar check build in.' '

—Matplotlib's documentation

CHAPTER 2

File Format Functions and Classes

2.1 IONMIX (.cn4)

class opacplot2.**OpacIonmix**(*fn*, *mpi*, *twot=False*, *man=False*, *hassele=False*, *verbose=False*)

Class to read in IONMIX EOS and Opacity Files.

The `OpacIonmix` class is used to read in an IONMIX file and translate its information into object attributes. All energies in this file are in Joules and must be converted to ergs. Unlike other file classes, `OpacIonmix` does not store its data as a dictionary. Instead, it stores its data in class attributes.

Parameters

- **fn** (*str*) – The name of the file to open.
- **mpi** (*str*) – The mass per ion **in grams**.
- **twot** (*bool*) – Flag for two-temperature data.
- **man** (*bool*) – Flag for manual temperature/density points.
- **hassele** (*bool*) – lag for electron entropy data.

fn

str

Filename.

mpi

float

Mass per ion.

twot

bool

Two-temperature data.

man

bool

Manual temp/dens points.

hassele

bool

Has electron entropy data.

verb
bool

Verbose.

ntemp
int

Number of temperature points.

ndens
int

Number of density points.

numDens
numpy.ndarray

Number densitites.

temps
numpy.ndarray

Temperatures.

ngroups
numpy.ndarray

Number of groups

data
str

Data at the end of the IONMIX file.

dens
numpy.ndarray

Densities.

ngroups
int

Number of groups.

zbar
numpy.ndarray

Average ionizations.

etot
numpy.ndarray

Total energy. Only included in single-temperature data.

cvtot
numpy.ndarray

Total C_v. Only included in single-temperature data.

dedn
numpy.ndarray

de/dn. Only included in single-temperature data.

dzdt
numpy.ndarray

dz/dt. Only included in two-temperature data.

pion
numpy.ndarray

Ion pressure. Only included in two-temperature data.

pele
numpy.ndarray

Electron pressure. Only included in two-temperature data.

dpidt
numpy.ndarray

dp_i/dt. Only included in two-temperature data.

dpedt
numpy.ndarray

dp_e/dt. Only included in two-temperature data.

eion
numpy.ndarray

Ion energy. Only included in two-temperature data.

eele
numpy.ndarray

Electron energy. Only included in two-temperature data.

cvion
numpy.ndarray

C_v for ions. Only included in two-temperature data.

cvele
numpy.ndarray

C_v for electrons. Only included in two-temperature data.

deidn
numpy.ndarray

de_i/dn. Only included in two-temperature data.

deedn
numpy.ndarray

de_e/dn. Only included in two-temperature data.

opac_bounds
numpy.ndarray

Opacity boundaries.

rosseland
numpy.ndarray

Rosseland opacity.

planck_absorb
numpy.ndarray

Planck absorption.

planck_emiss
numpy.ndarray

Planck emissivity.

Examples

For a directory with the IONMIX file `imx.cn4` for Aluminum:

```
>>> import opacplot2 as opp
>>> op = opp.Opac_Ionmix('imx.cn4', 4.4803895e-23) # Al mass in grams
>>> print(op.zbar)
array([...]) # Array of average ionizations for dens/temp points.
```

Notes

If you receive a `ValueError: invalid literal for int() with base 10` error, setting `man=True` may help to fix this.

extendToZero()
This routine adds another temperature point at zero.

write(*fn*, *zvals*, *fracs*, *twot=None*, *man=None*)
This method writes to an IONMIX file.

Parameters

- **fn** (*str*) – Name of output file.
- **zvals** (*tuple*) – Atomic numbers.
- **fracs** (*tuple*) – Element fractions.
- **twot** (*bool*) – Flag for two-temperature data.
- **man** (*bool*) – Flag for manual temp/dens points.

Examples

In order to extend `imx.cn4` for Al to zero:

```
>>> import opacplot2 as opp
>>> op = opp.OpacIonmix('imx.cn4', (13,), (1,))
>>> op.extendToZero() # Add temperature point at zero.
>>> op.write('imx-0.cn4', (13,), (1,))
```

opacplot2.**writeIonmixFile**(*fn*, *zvals*, *fracs*, *numDens*, *temps*, *zbar=None*, *dzdt=None*, *pion=None*, *pele=None*, *dpidt=None*, *dpedt=None*, *eion=None*, *eele=None*, *cvion=None*, *cvele=None*, *deidn=None*, *deedn=None*, *ngroups=None*, *opac_bounds=None*, *rosseland=None*, *planck_absorb=None*, *planck_emiss=None*, *sele=None*)
`opacplot2.writeIonmixFile()` provides an explicit and flexible way to write IONMIX files.

Parameters

- **fn** (*str*) – Name of the file to write.
- **zvals** (*tuple*) – Atomic numbers of elements to write to file.
- **fracs** (*tuple*) – Element fractions.
- **numdens** (*numpy.ndarray*) – Number densities.
- **temps** (*numpy.ndarray*) – Temperature array.
- **zbar=None** (*numpy.ndarray*) – Average ionization. Only used for tabulated EoS in *FLASH*.
- **dzdt=None** (*numpy.ndarray*) – Temperature derivative of average ionization. Ignored by *FLASH*.
- **pion=None** (*numpy.ndarray*) – Ion pressure Only used for tabulated EoS in *FLASH*.
- **pele=None** (*numpy.ndarray*) – Electron pressure. Only used for tabulated EoS in *FLASH*.
- **dpidt=None** (*numpy.ndarray*) – Temperature derivative of ion pressure. Ignored by *FLASH*.
- **dpedt=None** (*numpy.ndarray*) – Temperature derivative of electron pressure. Ignored by *FLASH*.
- **eion=None** (*numpy.ndarray*) – Ion specific internal energy. Only used for tabulated EoS in *FLASH*.
- **eele=None** (*numpy.ndarray*) – Electron specific internal energy. Only used for tabulated EoS in *FLASH*.
- **cvion=None** (*numpy.ndarray*) – Ion heat capacity at constant volume. Ignored by *FLASH*.
- **cvele=None** (*numpy.ndarray*) – Electron heat capacity at constant volume. Ignored by *FLASH*.
- **deidn=None** (*numpy.ndarray*) – Number derivative of ion energy. Ignored by *FLASH*.
- **deedn=None** (*numpy.ndarray*) – Number derivative of electron energy. Ignored by *FLASH*.
- **ngroups=None** (*int*) – Number of energy groups.
- **opac_bounds=None** (*numpy.ndarray*) – Energy group boundaries.
- **rosseland=None** (*numpy.ndarray*) – Rosseland opacities. Only used for tabulated EoS in *FLASH*.
- **orb=None** (*planck_abs*) – Planck absorption opacity. Only used for tabulated EoS in *FLASH*.
- **planck_emiss=None** (*numpy.ndarray*) – Planck emission opacity. Only used for tabulated EoS in *FLASH*.
- **sele=None** (*numpy.ndarray*) – Electron entropy.

Examples

Here we open an HDF5 file containing EoS and Opacity data and write it to an IONMIX file:

```
>>> import opacplot2 as opp
>>> op = opp.OpgHdf5.open_file('/path/to/infile.h5')
>>> opp.writeIonmixFile('outfile.cn4',
                        op['Znum'], op['Xnum'],
                        numDens=op['idens'][:], temps=op['temp'][:],
                        ngroups=op.Ng,
                        opac_bounds=op['groups'][:],
                        planck_absorb=op['opp_mg'][:],
                        rosseland=op['opr_mg'][:],
                        planck_emiss=op['emp_mg'][:])
```

2.2 MULTI (.opp, .opr, .eps)

class opacplot2.**OpgMulti**(**cargs*, ***vargs*)

Can be used either to parse or to write MULTIv5 tables.

OpgMulti is a subclass of dict. Through open_file(), it can read opacity data (only) from MULTI files. Then, the data can be accessed through the key:value pairs of the OpgMulti instance.

Examples

If we are in a directory with the files He_snp.eps.gz, He_snp.opp.gz, He_snp.opr.gz, and He_snp.opz.gz, then the following would read their data into an OpgMulti object:

```
>>> import opacplot2 as opp
>>> op = opp.OpgMulti.open_file('/path/to/current/dir', 'He_snp')
```

classmethod open_file(*folder*, *base_name*, *verbose=True*)

Parse MULTI format from a file.

Parameters

- **str** (*folder*) – Name of directory containing MULTI files.
- **base_name** (*str*) – Base name of MULTI files.
- **verbose** (*bool*) – Verbose option.

Returns Dictionary containing EoS and/or opacity data.

Return type *OpgMulti*

toEosDict(*Znum=None*, *Anum=None*, *Xnum=None*, *log=None*)

This method creates a dictionary with keys that are common among the various EoS table formats.

write(*prefix*, *fmin=None*, *fmax=None*)

Write multigroup opacities to files specified by a prefix.

Parameters

- **prefix** (*str*) – Prefix to append to files that are written.
- **fmin** (*float*) – Minimum value for opacities to write.
- **fmax** (*float*) – Maximum value for opacities to write.

Examples

After filling an instance of `OpgMulti` with EoS/opacity data, one could call:

```
>>> op_multi.write('multi_')
```

to write data files containing multigroup opacities specified by the 'multi_' prefix.

write2hdf (*filename*, *Znum=None*, *Anum=None*, *Xnum=None*)
Convert to HDF5 parameters.

Parameters

- **filename** (*str*) – Output filename.
- **Znum** (*tuple*) – Atomic numbers of elements.
- **Anum** (*tuple*) – Atomic masses of elements.
- **Xnum** (*tuple*) – Fractions of elements.

Examples

The atomic number and relative fractions must be given if they are not already parsed into an instance of `OpgMulti`. If they are not, `write2hdf` will raise a `ValueError`.

If we were in the same directory as the previous example, the following would write an HDF5 file with data from our MULTI files:

```
>>> import opacplot2 as opp
>>> op = opp.OpgMulti.open_file('/path/to/current/dir', 'He_snp')
>>> op.write2hdf('outfile.h5')
```

opacplot2.**get_related_multi_tables** (*folder*, *base_name*, *verbose=False*)
Get all related multi Tables defined by a folder and a base name.

Parameters

- **folder** (*str*) – Folder containing the tables.
- **base_name** (*str*) – Base name of the table.
- **verbose** (*bool*) – Flag for verbose option.

2.3 SESAME ASCII (.ses)

class opacplot2.**OpgSesame** (*filename*, *precision*, *verbose=False*)
This class is responsible for loading all SESAME formatted data files.

`OpgSesame` reads in a SESAME file. Each key:value pair of the `data` attribute corresponds to a table ID and its data from the file, respectively.

Parameters

- **fn** (*str*) – Name of file to open.
- **precision** (*int*) – `opacplot2.OpgSesame.SINGLE` for entry lengths of 15 or `opacplot2.OpgSesame.Double` for entry lengths of 22.
- **verbose** (*bool*) – Verbose option.

data
dict

Dictionary of material IDs included in the SESAME file.

Examples

The `opacplot2.OpgSesame.data` dictionary will hold the EoS data for the table referenced by `table_id`. For example, if we are in a directory with the file `sesame.ses`:

```
>>> import opacplot2 as opp
>>> op = opp.OpgSesame('sesame.ses', opp.OpgSesame.SINGLE)
>>> print(op.data.keys())
dict_keys([..., 13719]) # Table ID numbers; Aluminum.
>>> data = op.data[13719]
>>> print(sorted(data.keys()))
dict_keys(['abar',...,'zmax']) # Dictionary containing EoS data.
```

Note: There are only handling functions for 300 series entries in the SESAME tables.

2.3.1 Data Prefixes

From Los Alamos National Laboratory, each SESAME table has five different parts of their EoS tables:

Table #	Data Type	`opacplot2` prefix
Table 301	TotalEOS(304+305+306)	`total_`
Table 303	Ion EOS Plus Cold Curve (305 + 306)	`ioncc_`
Table 304	Electron EOS	`ele_`
Table 305	Ion EOS (Including Zero Point)	`ion_`
Table 306	Cold Curve (No Zero Point)	`cc_`

If we wanted to print out all of the EoS data for the electrons:

```
>>> import opacplot2 as opp
>>> op = opp.OpgSesame('sesame.ses', opp.OpgSesame.SINGLE)
>>> data = op.data[13719]
>>> for key in data.keys(): # Table ID for aluminum.
...     if 'cc_' == key[:3]:
...         print(key+':')
...         print(data[key])
```

2.3.2 Data Points

There are several top-level data points that are not included in a specific curve of the SESAME tables:

`opacplot2` Abbr.	Phys. Meaning
`abar`	Mean Atomic Mass
`bulkmod`	Bulk Modulus
`excoef`	Exchange Coefficient
`rho0`	Normal Density
`zmax`	Mean Atomic number

Furthermore, each table (301, ..., 305) prefixes these data points:

opacplot2 Abbr.	Phys. Meaning
dens	Density
eint	Energy
ndens	Number of Densities
ntemp	Number of Temperatures
pres	Pressures
temps	Temperatures

From the example in the previous section, if we wanted to print out the density array of electrons, we would simply type:

```
>>> print(data['ele_dens'])
array([...]) # Electron density array.
```

2.4 PROPACEOS ASCII (.prp)

Warning: Handling for Propaceos EoS tables is not publicly distributed with opacplot2 and so its documentation will not be presented here.

2.5 HDF5

class opacplot2.**OpgHdf5**

force_eval()
Load the whole table into memory.

classmethod open_file(*filename*, *explicit_load=False*)
Open an HDF5 file containing opacity data.

Parameters

- **filename** (*str*) – Name of file to open.
- **explicit_load** (*bool*) – Option to load the whole file to memory.

Examples

To open a file:

```
>>> import opacplot2 as opp
>>> op = opp.OpgHdf5.open_file('infile.h5')
>>> print(op.keys())
dict_keys(['Anum', ..., 'Znum']) # OpgHdf5 is a dictionary.
>>> print(op['Zf_DT'])
array([...]) # Array for average ionization.
```

Notes

Loading the entire file into memory can be potentially dangerous. Use explicit_load with caution.

write2file (*filename*, ***args*)
Write to an HDF5 output file.

Parameters

- **filename** (*str*) – Name of output file.
- **args** (*dict*) – Dictionary of data to write.

2.5.1 Data Points

Depending on what kind of data has been written to an HDF5 file, each HDF5 file may vary widely. Despite this fact, listed below are the naming conventions `opacplot2` uses for HDF5 data points that are relevant to IONMIX. The abbreviations are keys unless otherwise stated to be an attribute to the `OpgHdf5` object.

`opacplot2` Abbr	Physical Meaning
`Znum`	Atomic numbers
`Xnum`	Element fractions
`idens`	Number densities
`temp`	Temperature array
`Ng` (attr)	Number of energy groups
`groups`	Energy group boundaries
`opp_mg`	Planck absorption
`opr_mg`	Rosseland opacities
`emp_mg`	Planck emissivity

For example, if we wanted to open up an HDF5 file named `input.h5` and write it to an IONMIX file named `output.cn4`, we could write:

```
>>> import opacplot2 as opp
>>> op = OpgHdf5.open_file('input.h5')
>>> opp.writeIonmixFile(outfile,
                        op['Znum'], op['Xnum'],
                        numDens=op['idens'][:], temps=op['temp'][:],
                        ngroups=op.Ng,
                        opac_bounds=op['groups'][:],
                        planck_absorb=op['opp_mg'][:],
                        rosseland=op['opr_mg'][:],
                        planck_emiss=op['emp_mg'][:])
```

CHAPTER 3

Command Line Tools

3.1 opac-convert

Command line tool for converting EoS Table formats into the IONMIX format that comes with `opacplot2`.

Supported input file formats:

- Propaceos (not distributed, contact jtlaune at uchicago dot edu.)
- SESAME (.ses)
- MULTI (.opp, .opr, .opz, .eps)

The only supported output format is IONMIX.

3.1.1 Usage

```
opac-convert [options] myfile.ext
```

`opac-convert` will attempt to read your file extension and convert it to IONMIX accordingly. If it is unable to read the extension, you can use the input flag to specify your filetype. Some files need additional information to write to IONMIX, such as atomic numbers. These you must specify with the command line options shown below.

3.1.2 Options

Option	Action
-i, –input	Specify the input filetype (`propaceos`, `sesame`, `multi`)
–Znum	Comma separated list of atomic numbers.
–Xfracs	Comma separated list of element fractions.
–outname	Specify the output filename.
–log	Comma separated list of logarithmic data.
–tabnum	SESAME table number (defaults to last).

3.1.3 Example

To specify the files atomic numbers, one may use `--Znum` with a comma separated list of integers. If more than one atomic number is given, one must also specify the element fractions with `--Xfracs`. For example, take a SESAME table for CH named `myfile.ses`:

```
opac-convert --Znum 1, 6 --Xfracs .5, .5 myfile.ses
```

This will convert `myfile.ses` to an IONMIX file named `myfile.cn4`.

3.1.4 Logarithmic Data

If you would like to take the log of the data before you write it to the IONMIX file, use `--log` with a comma separated list of the data keys as shown below. Each key specified will be written to IONMIX after the base 10 logarithm has been applied.

Data	Key
Ion number density	`idens`
Temperatures	`temps`
Average ionization	`Zf_DT`
Ion pressure	`Pi_DT`
Electron pressure	`Pec_DT`
Ion internal energy	`Ui_DT`
Electron internal energy	`Uec_DT`
Opacity bounds	`groups`
Rosseland mean opacity	`opr_mg`
absorption Planck mean opacity	`opp_mg`
emission Planck mean opacity	`emp_emg`

For example, in order to specify that the emission Planck Mean Opacity be written logarithmically:

```
opac-convert --log emp_mg my-file.ext
```

3.1.5 Troubleshooting

Invalid Literal for `int()`

The `--log` flag may be used to fix the following error:

```
ValueError: invalid literal for int() with base 10
```

This error arises when the exponent for the data is more than 2 digits long, which IONMIX does not support. What that usually means is that the data was originally stored logarithmically and must be written back to IONMIX as logarithmic data.

3.2 opac-error

Command line tool for comparing two EoS table data. Unlike *opac-convert*, this tool will only compare equation of state data (not opacity). It is particularly useful in checking the consistency of *opac-convert* by comparing the original file with the converted IONMIX output. Supported input file formats:

- Propaceos (not distributed, contact jtlaune at uchicago dot edu.)
- SESAME (.ses)
- IONMIX

The output will consist of an error report with maximum absolute % error RMS % error. All % errors are calculated with respect to the first file listed. The user may also opt to create % error plots for the EoS data stored in the file (ion/electron pressure and energy and average ionization). These consist of three plots with resolutions of 10%, 1%, and .01%.

3.2.1 Usage

:: opac-convert [options] myfile_1.ext myfile_2.ext

Much like `opac-convert`, this tool will first attempt to read the extensions from your EoS tables in order to open them up. However, some file types require additional information, as in `opac-convert`, which can be specified in the `[options]`. The options have many of the same names as `opac-convert` but suffixed by `_1` for file 1 or `_2` for file 2.

Then, `opac-error` will create error reports for the following data:

- Average ionization
- Electron Ppessure
- Ion pressure
- Electron energy
- Ion energy

Included in the error report, we have

- Root mean squared % error
- Maximum absolute % error

If the `--plot` flag is called, `opac-convert` will also make error plots and save them as images to the current directory.

3.2.2 Options

Option	Action
–filetypes	Comma separated list of file types.
–mpi_#	Mass per ion (g) for file 1 or 2.
–Znum_#	Comma separated list of atomic numbers for file 1 or 2.
–Xfrac_#	Comma separated list of number fractions for file 1 or 2.
–filters_#	`dens_filter, temp_filter` for file 1 or 2.
–tabnum_#	SESAME table number for file 1 or 2.
–plot	Create % error plots for data.
–writelog	Write log file with % errors for data.
–lin_grid	Plot using linear axes.

3.2.3 Example

See the wiki on GitHub.

3.2.4 How It Works

First, *opac-convert* takes a conservative intersection of the two dens/temp grids from each file. Then it linearly interpolates the data from both files onto the intersection dens/temp grids. Using the interpolated data, it is able to create an error report.

CHAPTER 4

Utilities

4.1 Interpolation

opacplot2.utils.**interpDT**(*arr*, *dens*, *temps*, *bcdmin=0*, *bctmin=0*, *lookup=0*)

Depending on the choice for lookup, this function returns an interpolation function for values in arr, the density derivative of arr, or the temperature derivative of arr.

What `interpDT()` returns is dependent upon the `lookup` and `bcdmin`/`bctmin` arguments:

`INTERP_FUNC` will return an interpolation function for any density/temperature point within the range.

`INTERP_DFDD` will return a function for the density derivative at any density/temperature point within the range.

`INTERP_DFDT` will return a function for the temperature derivative at any density/temperature point within the range.

`BC_BOUND` is the default setting. If an input density/temp is smaller than the minimum value in the `dens` (or `temps`) array, then it will automatically be set to this minimum value.

`BC_EXTRAP_ZERO` will insert zero points into the `arr` and either `dens` or `temps` for `bcdmin`, `bctmin` respectively.

Parameters

- **arr** (*numpy.ndarray*) – Data array.
- **dens** (*numpy.ndarray*) – Density array.
- **temps** (*numpy.ndarray*) – Temperature array.
- **bcdmin** (*int*) – Boundary conditions for density.
- **bctmin** (*int*) – Boundary conditions for temperature.
- **lookup** (*int*) – Type of function to return.

Examples

In order to find a function that interpolates between dens/temp points for an IONMIX file that we had previously opened as `imx`, we could use:

```
>>> import opacplot2 as opp
>>> f = opp.utils.interpDT(imx.pion, imx.dens, imx.temps,
...                        bcdmin=BC_EXTRAP_ZERO,
...                        bctmin=BC_EXTRAP_ZERO,
...                        lookup=INTERP_FUNC)
>>> print(f(0,0)) # We added points at zero.
0
>>> print(f(123, 456)) # Density of 123 and temperature of 456.
1234.5678 # Resulting ion pressure at this dens/temp.
```

opacplot2.utils.**intersect_1D_sorted_arr**(*arr_1*, *arr_2*)

Function to return the venn diagram of two sorted 1D arrays.

In other words, this function will return the union of all values from both arrays that are within the intersection of their ranges. This may be used for interpolation schemes.

Parameters

- **arr_1** (*numpy.ndarray*) – 1D sorted array number 1.
- **arr_2** (*numpy.ndarray*) – 1D sorted array number 2.

Returns An array that is the venn diagram of the two input arrays.

Return type numpy.ndarray

Examples

```
>>> import numpy as np
>>> import opacplot2 as opp
>>> a = np.array([1,2,3,4,5])
>>> b = np.array([3.5,4.5,5.5,6])
>>> c = opp.utils.intersect_1D_sorted_arr(a,b)
>>> print(c)
[3.5 4 4.5 5]
```

4.2 Grid Structure

class opacplot2.utils.**EosMergeGrids**(*eos_data, filter_dens=<function <lambda>>, filter_temps=<function <lambda>>, intersect=['ele', 'ioncc'], thresh=[], qeos=False*)

This class provides filtering capabilities for the EoS temperature and density grids.

For instance, SESAME tables may have some additional points in the ion EoS table, compared to the electron EoS table, and as FLASH requires the same density and temperature grid for all species, the simplest solution is to remove those extra points.

Parameters

- **eos_data** (*dict*) – Dictionary contraining the EoS data.
- **intersect** (*list*) – The resulting temperature [eV] and density [g/cm^(-3)] grids will be computed as an intersection of grids of all the species given in this list. Default: ['ele', 'ioncc']
- **filter_dens** (*function*) – A function that takes a grid of densities and returns a mask of points we don't wont to keep. Defaut: (lamdba x: x>0.) i.e. don't remove anything.

- **filter_temps** (*function*) – A function that takes a grid of temperatures and returns a mask of points we don't wont to keep. Defaut: (lamdba x: x>0.) i.e. don't remove anything.
- **thresh** (*list*) – Zero threshold on following keys

Returns **out** – A dictionary with the same keys as eos_data. The species specified by `intersect` will have equal temperature and density grids.

Return type dict

Examples

```
>>> eos_sesame = opp.OpgSesame("../sesame/xsesame_ascii",
                               opp.OpgSesame.SINGLE,verbose=False)
>>> eos_data  = eos_sesame.data[3720]  # Aluminum
>>> eos_data_filtered = EosMergeGrids(eos_data,
        intersect=['ele', 'ioncc'],   # Merge ele and ioncc grids
        filter_temps=lamda x: x>1.) # Remove temperatures below 1eV
```

4.3 Miscellaneous

`opacplot2.utils.`**randomize_ionmix**(*filename*, *outfilename*)

Randomizes the data from an existing ionmix file and rewrites it to the outfile.

Parameters

- **filename** (*str*) – Name of file to randomize.
- **outfilename** (*str*) – Name of output file.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Z