pycxl
Python Wrapper for Computing the Convex Hull

Samad Hajinazar *

1 Introduction

`pycxl` (version 1.0) is a Python script that:

- computes the convex hull for data points of a *system of arbitrary dimensions* (e.g., binary, ternary, quaternary, etc.),
- calculates and outputs the "distance above hull" for all data points,
- produces output files suitable for creating convex hull plots for binary and ternary systems,
- creates a 2-D plot output for binaries and ternaries if the `matplotlib` module exists on the system.

1.1 Dependencies

The `pycxl` script requires `scipy` and `numpy` modules to calculate and analyze the convex hull, and -optionally- uses `matplotlib` to produce convex hull plots for binary and ternary system.

1.2 Usage

The script can be used as:

```
pycxl.py input_file
```

where the "input_file" is a text file with the input data. If the input file is not specified, the script will look for the `points.txt` file in the working directory to read the input data.

2 Input

2.1 Input file format and content

The input file should be a text file that contains the composition-energy information. Each line in the input file should contain the fractional composition of a configuration (with a total sum of 1.0 for elemental contributions), while the last column should contain the corresponding energy. An example of a binary system is as follows:

```
0.3  0.7 -1.2
1.0  0.0  0.0
0.0  1.0  0.1
0.5  0.5  0.1
1.0  0.0 -0.1
0.3  0.7  0.8
```

*Department of Chemistry, State University of New York at Buffalo — samadh˜at˜buffalo.edu

In the input file:

1. the order of the entries is not important,
2. empty lines and those starting with "#" (i.e., comment lines) are ignored.

2.2 About reference structures

Technically, the "reference" structures don't need to be explicitly identified in the input file. However, at least one of each elemental entries for the system "must" be given, e.g., the following lines in the above-mentioned example of a binary system:

```
...
0.0  1.0  0.1
...
1.0  0.0 -0.1
...
```

To make sure that the elemental formation energies are always zero, the script re-calculates a new set of energies by simply subtracting the weighted lowest elemental energies from the input energy values. This set of calculated "formation energies" will be used to compute the distance above hull values, and will be reported in the output.

Since the script internally calculates the formation energies, in principle, it is possible to use the raw total energy per atom as the input energy values. However, this will lead to acceptable results only if the elemental phases are correct references for the formation energies of the system. In such a case, the calculated formation energies should be checked before using the results.

3 Output

The main output of the script is the text file `out_distances.txt` which lists all configurations, their original energy (as given in the input file), the calculated formation energies (which might or might not be different from the original energies as described above), and the distances above the hull.

The output file for the example binary system at the beginning of this document is:

```
#       elem1       elem2       orig_ene      form_ene      distance
    0.300000    0.700000     -1.200000     -1.240000      0.000000
    1.000000    0.000000      0.000000      0.100000      0.100000
    0.000000    1.000000      0.100000      0.000000      0.000000
    0.500000    0.500000      0.100000      0.100000      0.985714
    1.000000    0.000000     -0.100000      0.000000      0.000000
    0.300000    0.700000      0.800000      0.760000      2.000000
```

It should be noted that the above output file -and any other output of this script- is essentially a result of application of a series of numerical algorithms and analyses. Hence, the user must carefully verify the results before any use of them.

3.1 Extra output files for binary and ternary systems

For binary and ternary systems, a set of extra `out_plot*` files are also produced where, generally, each input entry (as well as the convex hull facets' vertices) are represented by a set of x-y coordinates which are adjusted to result in a "proper" 2-D convex hull plot. Further, the formation energy and distance above the hull of configurations is provided in data files just as an extra column which can be used for producing convex hull plots with color-coded symbols.

The produced files (plus, example output for the above binary system) are as follows:

- `out_plot_hull_points.txt` : adjusted coordinates of the input data for the points on the convex hull

```
#           x             y        form_ene       distance
      0.700000      -1.240000      -1.240000       0.000000
      1.000000       0.000000       0.000000       0.000000
      0.000000       0.000000       0.000000       0.000000
```

- `out_plot_points.txt` : adjusted coordinates of the input data for all points (except those on the hull)

```
#           x             y        form_ene       distance
      0.000000       0.100000       0.100000       0.100000
      0.500000       0.100000       0.100000       0.985714
      0.700000       0.760000       0.760000       2.000000
```

- `out_plot_lines.txt` : adjusted coordinates of the vertices of the convex hull facets (i.e., lines for binary and triangles for ternary systems)

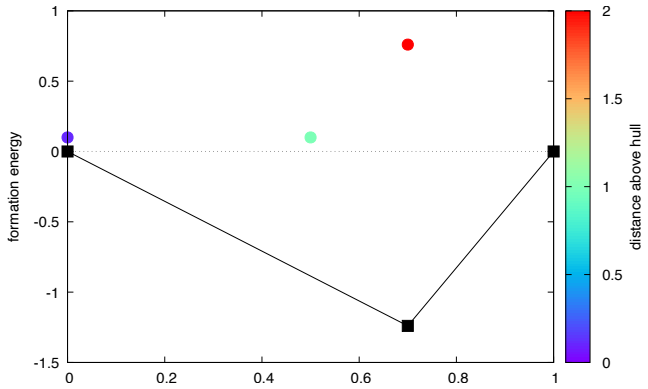
```
#           x             y
      0.700000      -1.240000
      1.000000       0.000000

      0.000000       0.000000
      0.700000      -1.240000
```

These files are formatted such that they can be directly used in common plotting software to create convex hull plots. For demonstration purposes, the simple `gnuplot` script

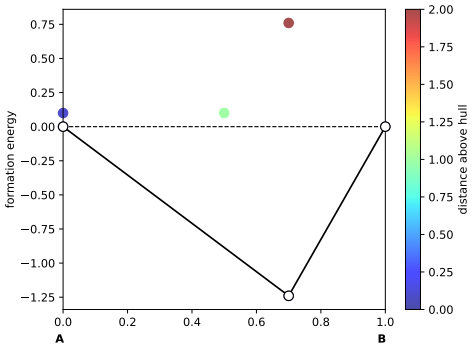
```
set terminal pdf; set output 'out_gnuplot.pdf'; set key off
set palette rgb 33,13,10; set cbrange [0:]; set cblabel "distance above hull"
# === Uncomment next line only for binaries
set xzeroaxis; set ylabel "formation energy"
# === Uncomment next line only for ternaries
#unset border; unset ytics; unset xtics
plot "out_plot_points.txt"   w p palette pt 7, \
     "out_plot_lines.txt"    w l lc 0, \
     "out_plot_hull_points.txt" w p pt 20 lc 0
```

produces the following figure for the example binary system:



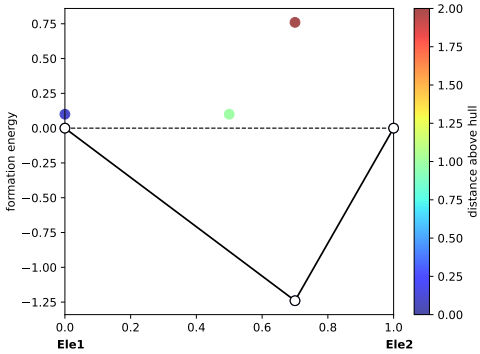
3.2 Plot outputs for binary and ternary systems (requires `matplotlib`)

If `matplotlib` is present on the system, the script will automatically produce an `out_distances.pdf` file for binary and ternary systems. This file contains the convex hull and data points colored according to their distance from the hull. For the simple binary system of this document, the automatically generated plot is:

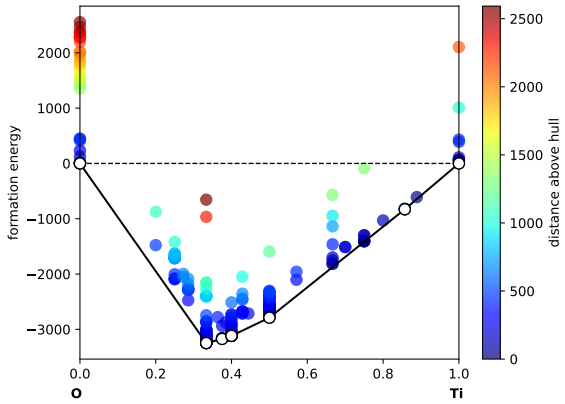


By default, the element names in these plots will be generic letters of "A-B" and "A-B-C" for binary and ternary systems, respectively. If the user wants the script-produced plots to contain the actual symbols for the elements, a line with symbols (without a "#" at the beginning) should be added to the input file, e.g.,

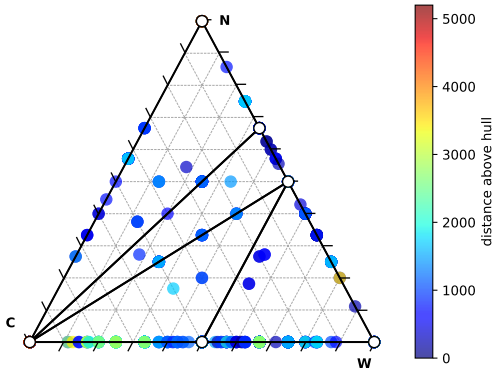
```
0.3   0.7  -1.2
1.0   0.0   0.0
0.0   1.0   0.1
0.5   0.5   0.1
Ele1 Ele2
1.0   0.0  -0.1
0.3   0.7   0.8
```



More examples of the script-produced figures (using the data from AFLOW[1]) are as follows:



O-Ti system



C-W-N system

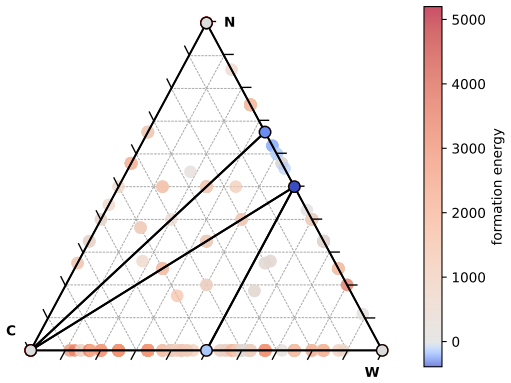
[1] C. Oses *et al.*, J. Chem. Inf. Model. 58(12), 2477-2490 (2018)

3.2.1 Additional options for automatic plots

By default, the script-produced plots include all input data points which are colored according to the distance above the hull of configurations, and with a grid mesh for ternary plots. These specifications can be changed using one or more of the following input flags.

Color coding the points using formation energies

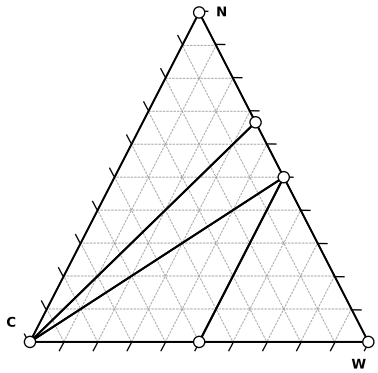
If the command line option of "`-f`" is added, the data points will be colored according to the formation energy of configurations. Then, for example, the output of the script for the C-W-N system (shown above) will be:



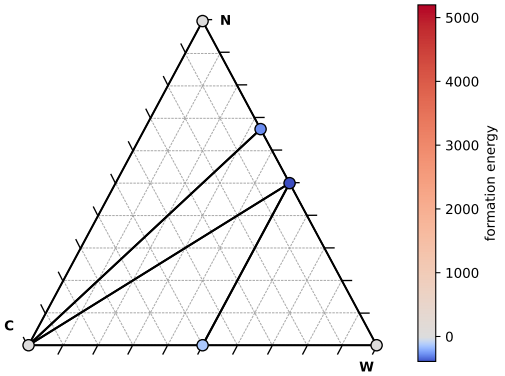
Output of "`pycxtl.py -f`" command

Plots with points on the hull only

In order to only show the points on the convex hull (instead of all data points), the command line option of "`-h`" can be used. This also can be combined with "`-f`" option to obtain a convex hull plot in which only the points on the hull are shown and they are colored according to their formation energies. Examples of these output plots for the C-W-N system are as follows:



Output of "`pycxl.py -h`" command



Output of "`pycxl.py -h -f`" command

Ternary plots with no grid mesh

Finally, the dashed-line grid mesh in the ternary plots can be turned off using the input flag of "`-p`", i.e., to produce a "plain" convex hull plot.

4 Optional identifiers for configurations

In dealing with large datasets, especially those that include several configurations of the same composition, it is challenging to keep track of the desired configurations in the output file.

In the input file, optionally, additional field(s) can be added after the energy, after a "#" character, for all or a subset of configurations. Configuration identifiers will appear in the `out_distances.txt` and `out_plot*points.txt` files for the corresponding entries.

These identifiers can be used to search the output files (e.g., for distance above the hull of a desired configuration) or to label symbols in the convex hull plots.

As an example of identifiers, the input file for the test binary system:

```
0.3  0.7 -1.2 # cfg1
1.0  0.0  0.0
0.0  1.0  0.1 # ref1
0.5  0.5  0.1 # test cfg 1
Ele1 Ele2
1.0  0.0 -0.1 #ref2
0.3  0.7  0.8 #a new cfg2
```

results in the following output `out_distances.txt` file:

```
#       elem1       elem2       orig_ene      form_ene      distance
    0.300000    0.700000     -1.200000     -1.240000      0.000000 cfg1
    1.000000    0.000000      0.000000      0.100000      0.100000
    0.000000    1.000000      0.100000      0.000000      0.000000 ref1
    0.500000    0.500000      0.100000      0.100000      0.985714 test cfg 1
    1.000000    0.000000     -0.100000      0.000000      0.000000 ref2
    0.300000    0.700000      0.800000      0.760000      2.000000 a new cfg2
```

With the command line option "`-t`", the script adds a tag of the form `inp#` to the entries that don't have an identifier where the `#` is the configuration number (in the order they appear in the input file).