pycxl
Python Script to Calculate Convex Hull

Samad Hajinazar *

1 Introduction

pycxl (version 1.8) is a Python script that:

- computes the convex hull for data points of a *system of arbitrary dimensions*,
- 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

A list of options can be obtained by running the script with "-h" flag. One can run the script with:

```
pycxl.py -i 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 either: (1) the fractional composition of a configuration (with a total sum of 1.0 for elemental contributions) while the last column being the corresponding energy per atom; (2) composition (i.e., atom counts) and the total energy. An example of a binary system with fractional composition 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
```

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

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

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 or per atom) energy 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)

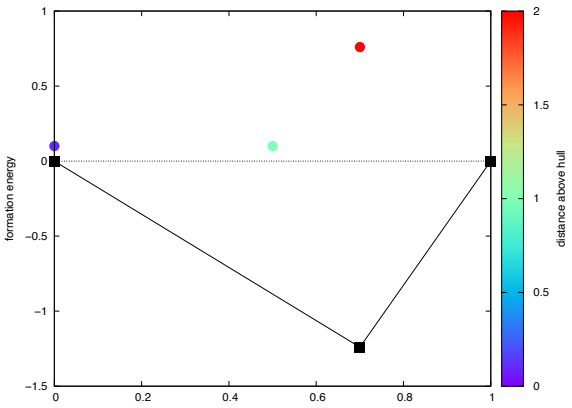
```
#            x              y
       0.700000      -1.240000
       1.000000       0.000000

       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 postscript eps color; set output 'out_gnuplot.eps'; 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" using 1:2:4 w p palette pt 7 ps 2, \
     "out_plot_lines.txt"    w l lc 0, \
     "out_plot_hull_points.txt" w p pt 5 ps 2 lc 0
```

produces the following figure for the example binary system:



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
```

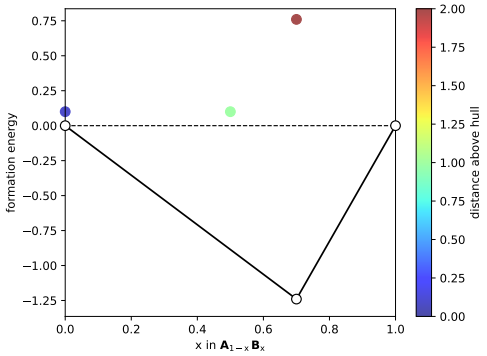
results in the following output `out_distances.txt` file:

```
#       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).

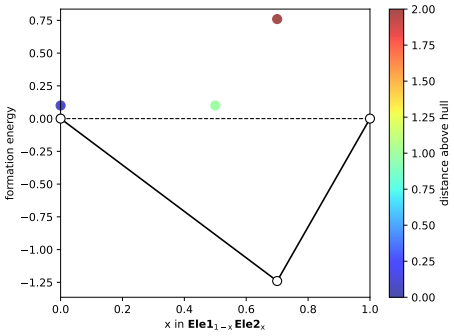
5 Plot outputs for binary and ternary systems (requires `matplotlib`)

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:



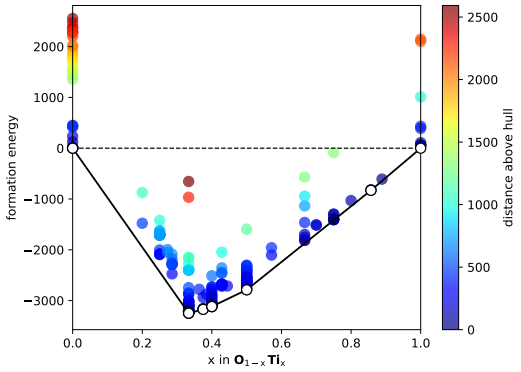
By default, the element names in these plots will be generic letters "A-B" and "A-B-C" for binary and ternary systems, respectively. If the user wants the script-produced plots to contain the actual symbols for the elements, a line with symbols (without a "#" at the beginning) should be added to the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

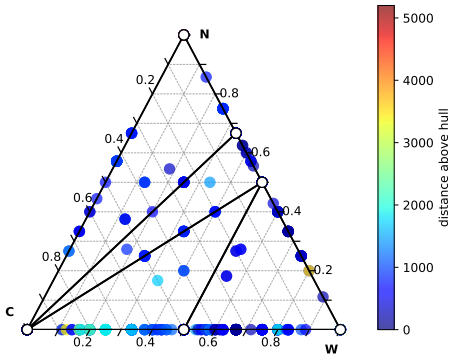


By default, the script-produced plots are in "pdf" format, include all input data points, with hull points highlighted with a edge and filled with white color, and those above the hull colored according to the distance above the hull of configurations. For ternary plots, a grid mesh is also added to highlight the fractional compositions.

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



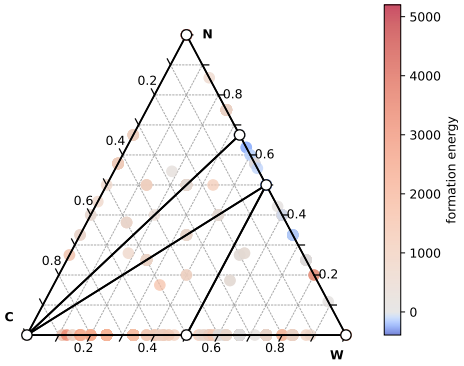
O-Ti system



C-W-N system

5.1 Color coding the points using formation energies

If the command line option "`-f`" is added, the data points will be colored according to the formation energy of configurations. In this case, the hull points are shown with the default settings (edge color highlighting and white inner color).



Output of "`pycxtl.py -f`"

5.2 Plot ranges

By default, the minimum (maximum) of the y-axis in a binary system plot is 10% lower (higher) than the actual values in the plotted data. This can be modified using the flag "`-y`" by specifying a number in the $[0, 1]$ range which will be converted to the offset as a percentage, e.g., the following command will plot the output with 5% offset compared to the actual range of the data:

```
pycxl.py -y .05
```

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

If, on the other hand, this flag is followed by two values, they will be used as the minimum and maximum of the y-axis in the output plot, e.g., the following input will produce a plot with y axis ranging from -0.25 to +0.05:

```
pycxl.py -y -.25 .05
```

These setting only take effect if the specified maximum (minimum) is not smaller (larger) than the actual minimum value of the data points.

Moreover, the maximum value of the color bar (i.e., values that are used to paint the data points in the plot) can be specified using the following input flag:

```
pycxl.py -c 0.25
```

5.3 Customizing the output plot

The specification of the output plot and the file format can be changed using one or a combination of a set of input flags, as follows.

- If the command line option "`-g`" is present, the output image will be a graphics image with "png" format.
- If the option "`-b`" is present, the output image's background will be transparent.
- With the input option "`-e`", the edge color for the hull points and tie lines can be specified (default is black), e.g.,

  ```
  pycxl.py -e red
  ```

- The resolution of the output image can be set using the "`-r`" flag (default is 200 dpi), e.g.,

  ```
  pycxl.py -r 500
  ```

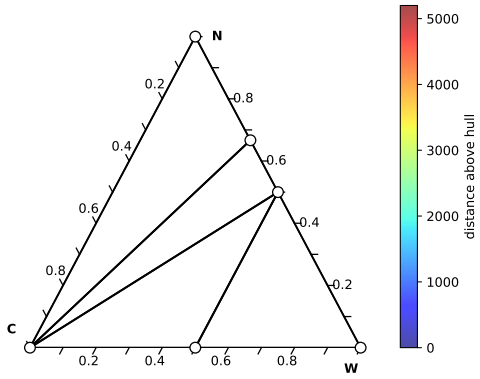
- Using the option "`-p`", results an output plot for ternary systems where the grid lines are removed, i.e., a plain plot.
- With the command line option "`-o`, the output plot will only show the points on the convex hull (instead of all data points). As an example, this can be combined with "`-f -a`" 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.

Starting from version 1.8.0, the following options are also available to further customize the output image:

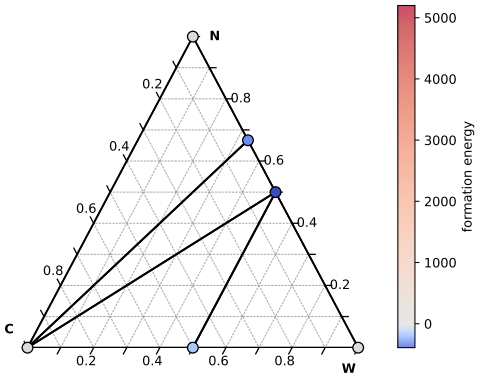
- With option "`-l`", the tie-lines (those connecting hull points) are removed,
- With option "`-s`", the hull points are shown as "square", instead of the default "circle" shape while highlighted with the edge color,
- With option "`-a`", the hull points are colored using the actual color-code value, i.e., the distance above hull (instead of the default white inner color),
- With option "`-n`", all highlighting options for hull points are disabled (i.e, options "`-s`, `-a`, and `-e`").

and, in version 1.8.1, the option "`-u`" with a valid color name will produce a plot where all points are uniformly colored with the specified color. In this case, the colorbar is just a placeholder with a plain color and no meaningful range.

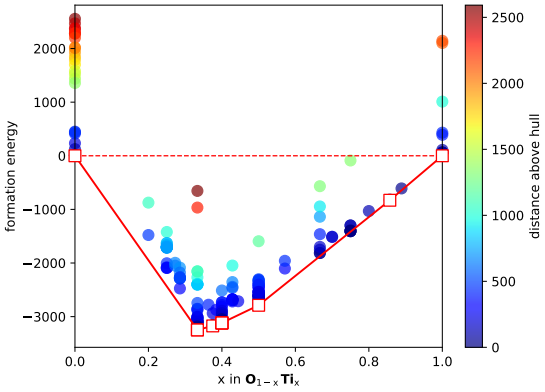
5.4 More examples of output plots with various options



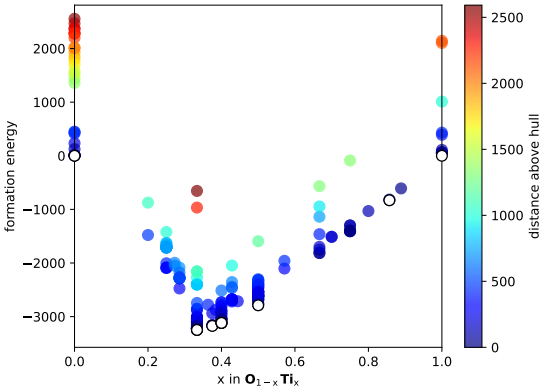
Output of "`pycxl.py -o -p`"



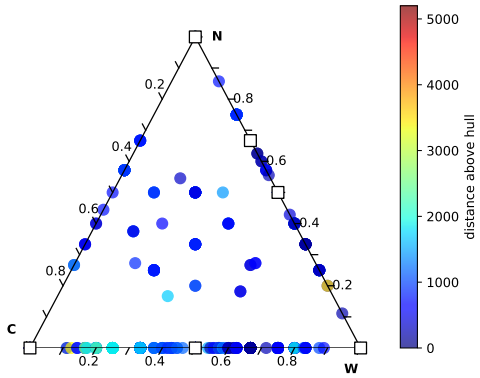
Output of "`pycxl.py -o -f -a`"



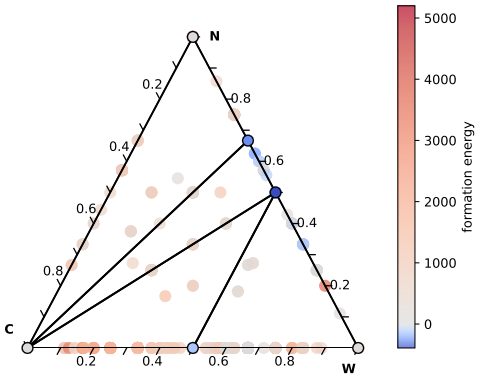
Output of "`pycxl.py -s -e red`"



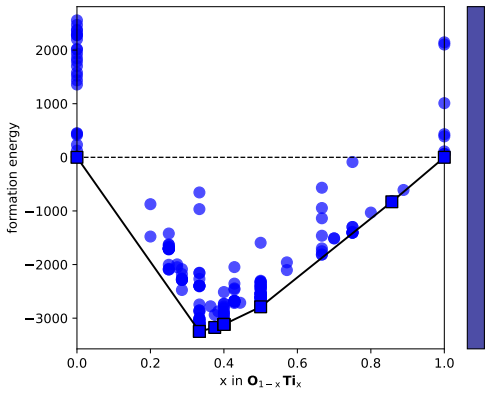
Output of "`pycxl.py -l`"



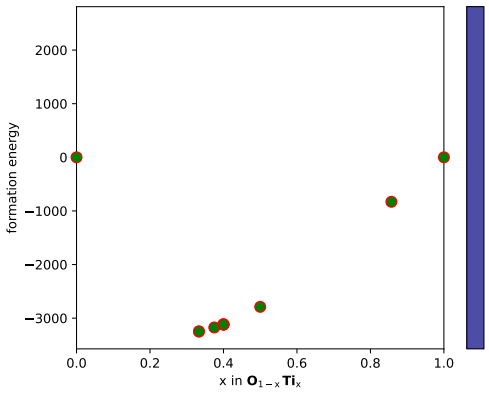
Output of "`pycxl.py -l -s`"



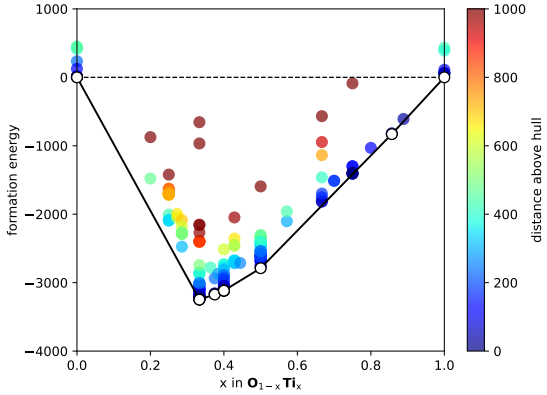
Output of "`pycxl.py -p -f -a`"



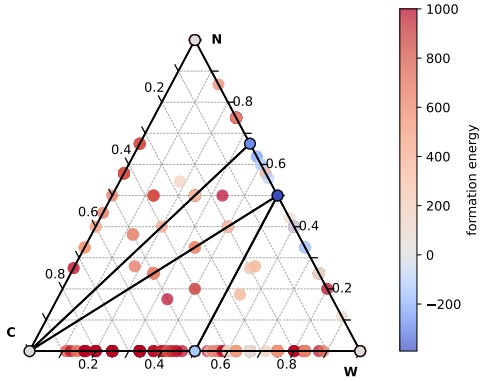
Output of "`pycxl.py -s -u blue`"



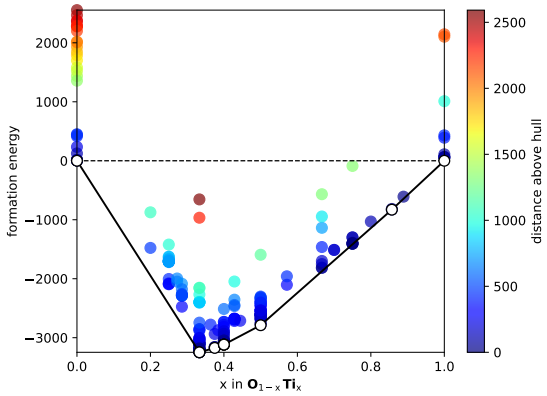
Output of "`pycxl.py -o -u green -e red -l`"



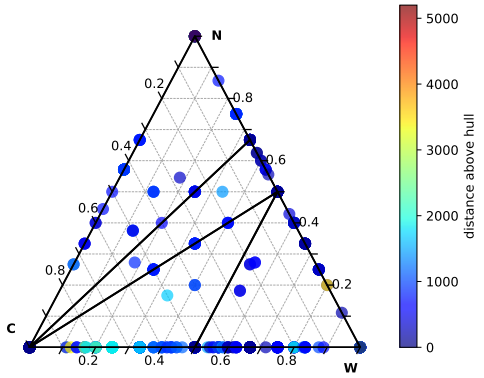
Output of "`pycxl.py -c 1000 -y -4000 1000`"



Output of "`pycxl.py -f -a -c 1000`"



Output of "`pycxl.py -y 0.0`"



Output of "`pycxl.py -n`"