Scripts for XR postprocessing with USPEX

Michele Galasso

August 22, 2021

1 split_CIFs.py

It splits the structures with lowest enthalpy of a variable composition USPEX run into multiple CIF files. The script takes the following arguments from command line:

1. the output file `extended_convex_hull` from USPEX
2. the output file `extended_convex_hull_POSCARS` from USPEX
3. the value of the external pressure used for the USPEX run

The script performs the following operations:

1. for each structure, it reads the parameters *enthalpy* and *fitness* from `extended_convex_hull` and the geometry from `extended_convex_hull_POSCARS`
2. it selects, for each reduced formula, the 5 structures with lowest enthalpy
3. it outputs the selected structures as CIF files in a new folder *results*

The name of each CIF file has the format `i_ID_fitness_enthalpy_iupacformula_pressure_symmetry.cif`, where:

i is a natural number which orders the output with increasing *fitness*

ID is the structure ID from the USPEX run

fitness is the *fitness* of the structure

enthalpy is the *enthalpy* of the structure

iupacformula is the *IUPAC formula* of the structure

pressure is the pressure used for the USPEX run

symmetry is the space group number, determined with tolerance 0.2

Example: `python split_CIFs.py extended_convex_hull extended_convex_hull_POSCARS 50GPa`

2 sublattice_split_CIFs.py

It reads the results of a variable composition USPEX run, it removes all hydrogen atoms, and then it splits the structures with lowest enthalpy into multiple CIF files. The script takes the following arguments from command line:

1. the output file `extended_convex_hull` from USPEX
2. the output file `extended_convex_hull_POSCARS` from USPEX
3. the value of the external pressure used for the USPEX run

The script performs the following operations:

1. for each structure, it reads the parameters *enthalpy* and *fitness* from `extended_convex_hull` and the geometry from `extended_convex_hull_POSCARS`
2. it deletes all hydrogen atoms
3. it selects, for each reduced formula, the 5 structures with lowest enthalpy
4. it outputs the selected structures as CIF files in a new folder *results*

The name of each CIF file has the format `i_ID_fitness_enthalpy_iupacformula_pressure_symmetry.cif`, where:

i is a natural number which orders the output with increasing *fitness*

ID is the structure ID from the USPEX run

fitness is the *fitness* of the structure

enthalpy is the *enthalpy* of the structure

iupacformula is the *IUPAC formula* of the structure, with hydrogens

pressure is the pressure used for the USPEX run

symmetry is the space group number, determined with a tolerance of 0.2 and without hydrogens

Example: `python sublattice_split_CIFs.py extended_convex_hull extended_convex_hull_POSCARS 50GPa`

3 fixcomp_split_CIFs.py

It splits all the structures of a fixed composition USPEX run into multiple CIF files. The script takes the following arguments from command line:

1. the output file `Individuals` from USPEX
2. the output file `gatheredPOSCARS` from USPEX

3. the value of the external pressure used for the USPEX run

The script performs the following operations:

1. for each structure, it reads the parameter *enthalpy* from `Individuals` and the geometry from `gatheredPOSCARS`
2. it computes `real_fitness` = `enthalpy` / `total_number_of_atoms`
3. it outputs the structures as CIF files in a new folder *results*

The name of each CIF file has the format `i_ID_fitness_enthalpy_iupacformula_pressure_symmetry.cif`, where:

i is a natural number which orders the output with increasing `real_fitness`

ID is the structure ID from the USPEX run

fitness is the `real_fitness` of the structure

enthalpy is the *enthalpy* of the structure

iupacformula is the *IUPAC formula* of the structure

pressure is the pressure used for the USPEX run

symmetry is the space group number, determined with a tolerance of 0.2

Example: `python fixcomp_split_CIFs.py Individuals gatheredPOSCARS 50GPa`

4 powder_xrd_screening.py

It performs a screening of USPEX results, looking for the structures that best match an experimental powder X-ray spectrum. For the theory behind this script, see appendix A. In a few words, given an input experimental spectrum, the theoretical spectrum is calculated for each structure in the USPEX run and a value F is computed. The smaller is F, the better is the agreement between theoretical and experimental spectra. The script contains the following input parameters:

1. the experimental pressure
2. the pressure of the USPEX run
3. the start and end angles for the computation of theoretical spectra
4. the experimental wavelength
5. the value of σ for the gaussian smearing of peaks, used for generating output pictures

6. the name of the *spectrum file*, containing angles and intensities of the experimental spectrum
7. the name of the file `extended_convex_hull` from USPEX
8. the name of the file `extended_convex_hull_POSCARS` from USPEX
9. the parameter *match_tol*, that is the tolerance for matching experimental peaks with theoretical peaks, in degrees

The script performs the following operations:

1. for each structure, it reads the parameter *fitness* from `extended_convex_hull` and the geometry from `extended_convex_hull_POSCARS`
2. it computes the theoretical X-ray spectrum
3. it computes the agreement F between the spectra
4. it outputs a CIF file with the symmetrized structure (tolerance 0.2)
5. it outputs a PNG graph with the theoretical and experimental spectra superimposed for comparison

The name of each CIF and PNG file have the format `F_ID_fitness_iupacformula_pressure_symmetry`, where:

F is the agreement F between theoretical and experimental spectra

ID is the structure ID from the USPEX run

fitness is the `fitness` of the structure from the USPEX run

iupacformula is the *IUPAC formula* of the structure

pressure is the pressure used for the USPEX run

symmetry is the space group number, determined with a tolerance of 0.2

Example: `python xr_screening.py`

5 exclusion.py

It allows to quickly filter a multitude of CIF files, by removing those which have significant peaks in a user-defined exclusion region of the X-ray spectrum. The script takes the following arguments from command line:

1. the wavelength of the incident radiation in Å
2. the peak cut-off, in % of the maximum intensity
3. a number of intervals in degrees, expressed as two angles separated by a hyphen (-), defining the exclusion region

The script works in a folder with many CIF files, and performs the following:

1. it opens, one by one, all CIF files and it predicts the XRD pattern of the structure according to the given wavelength
2. if the predicted pattern contains any peak in the exclusion regions that is bigger than the given cut-off, it deletes the CIF file

Example: `python exclusion.py 0.6199 25 25-28 31-32`

6 find_peak.py

It allows to quickly filter a multitude of CIF files, by removing those which do not have significant peaks in all user-defined search intervals of the X-ray spectrum. The script takes the following arguments from command line:

1. the wavelength of the incident radiation in Å
2. the peak cut-off, in % of the maximum intensity
3. a number of search intervals in degrees, expressed as two angles separated by a hyphen (-)

The script works in a folder with many CIF files, and performs the following:

1. it opens, one by one, all CIF files and it predicts the XRD pattern of the structure according to the given wavelength
2. if there is at least one search region which does not contain any peak bigger than the given cut-off, it deletes the CIF file

Example: `python find_peak.py 0.6199 15 25-28 31-32`

7 change_pressure.py

It translates the structures contained in many CIF files to a different pressure, by deforming the lattice parameters using a second order Taylor expansion of the Birch-Murnaghan equation. More details about the underlying theory can be found in appendix **??**. The script takes the following arguments from command line:

1. the initial pressure
2. the final pressure

The script works in a folder with many CIF files, and performs the following:

1. it reads the structures contained in all CIF files
2. it deforms the lattice parameters according to the new pressure

3. it creates new CIF files containing the deformed structures

The name of the new CIF files have the format `OLDNAME_toNEWPRESSURE`, where:

OLDNAME is the name of the file with the structure at the initial pressure

NEWPRESSURE is the final pressure

Example: `python change_pressure.py 50 58`

8 relax_new_pressure.py

It allows an accurate translation of the structures contained in many CIF files to a different pressure, by relaxing them with VASP. The script works in a folder with many CIF files and it is meant to be run on a cluster. Before running it, add the following lines to your `.bashrc` file:

1. export VASP_SCRIPT=PATH/TO/relax_new_pressure/run_vasp.py
2. export VASP_PP_PATH=PATH/TO/YOUR/pp/FOLDER

Then tune the `relax_new_pressure.py` file with the desired new pressure and the appropriate VASP parameters for your calculation and run the python script on a computing node. By default, VASP relaxations are performed sequentially and every relaxation takes 4 computing cores. For more details about the implementation, have a look at the ASE VASP calculator.

9 pareto_picture.py

This script organizes and visualizes the results of a USPEX calculation in which multi-objective optimization of stability together with the agreement with an experimental powder XRD spectrum has been optimized. Lines 19-21 of the script contain the input parameters, which need to be tuned by the user:

1. the wavelength of the incident radiation in Å
2. the name of the working directory, which must contain the files `goodStructures` and `goodStructures_POSCARS` from the USPEX output, as well as the spectrum file
3. the name of the spectrum file

The output of the script is constituted by a picture, named `pareto.png`, displaying each structure on a plane where stability lies on the Y axis and the disagreement with the given spectrum lies on the X axis (see Fig. 1). On the picture, the first three Pareto fronts are shown. It is possible to focus on a specific area of the graph by specifying the intervals to display on the X and Y axes at lines 76-77 and by rerunning the script. In addition, the scripts creates a folder named `pareto_fronts` which contains, for each structure on the first

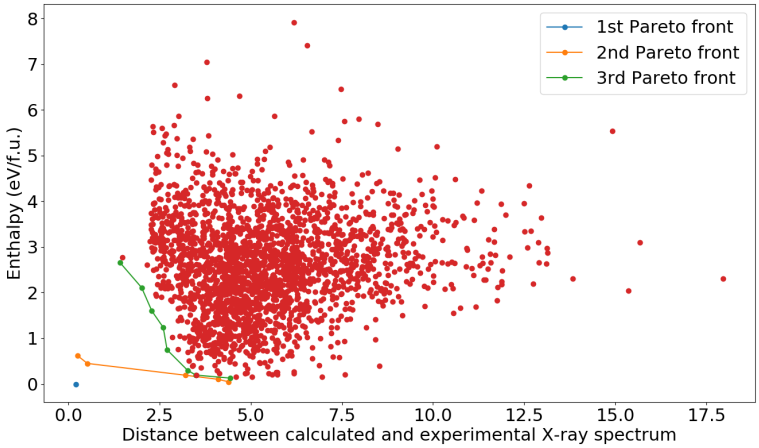

Figure 1: Graph produced by the script `pareto_picture.png` from the results of an USPEX run in which stability of the system $K_4Ta_4W_4O_{24}$ has been optimized together with the agreement with an experimental powder XRD spectrum.

three Pareto fronts, a structure file in the POSCAR format and a picture which visually compares the predicted powder XRD spectrum of the structure with the given spectrum contained in the spectrum file, as in Fig. 2. The name of these files have the format `rank_F_enthalpy_ID`, where:

rank is the progressive number of the Pareto front

F is the agreement F between simulated and given spectra

enthalpy is the enthalpy of the structure

ID is the structure ID from the USPEX run

Example: `python pareto_picture.py`

10 single_plot.py

This simple script is used to obtain a plot of the predicted powder XRD spectrum of a single structure against the experimental spectrum. It takes in input, at lines 12-14, the following three parameters:

1. the wavelength of the incident radiation in Å
2. the name of the file containing the geometry of the structure in POSCAR format

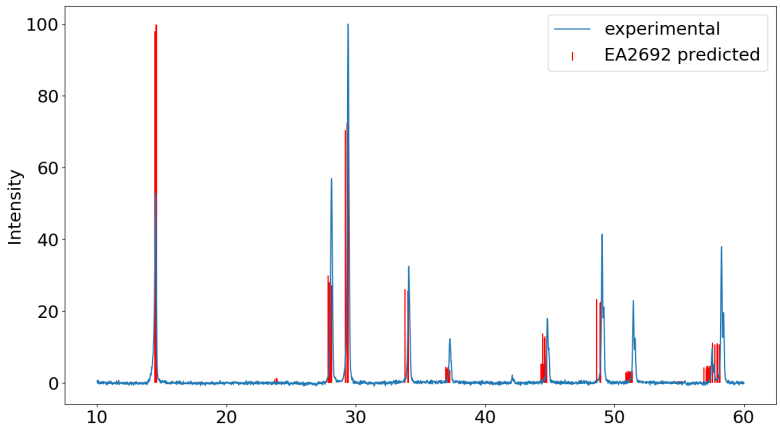

Figure 2: Graph produced by the script `pareto_picture.png` from the results of an USPEX run in which stability of the system $K_4Ta_4W_4O_{24}$ has been optimized together with the agreement with an experimental powder XRD spectrum.

3. the name of the file containing the experimental spectrum

Example: `python single_plot.py`

11 scxrd_screening

This script performs a screening of USPEX results, looking for the structures that best match an experimental single crystal X-ray spectrum. The fitness function for evaluating the agreement between theoretical and experimental spectra is the weighted R-factor (see appendix B). Notice that the USPEX run must have been performed at fixed cell parameters. The script contains the following input parameters:

1. the name of the `gatheredPOSCARS` file from the USPEX run
2. the name of the HKL file containing experimental reflections

The script outputs a file `results.txt` which contains a list of ordered structure IDs with their relative weighted R-factor, from the most fit to the least fit.

Example: `python scxrd_screening.py`

Appendices

A Powder spectra comparison

We developed a code which computes, from the experimental powder XRD spectrum and the USPEX output, the degree of disagreement (fitness) of each relaxed structure with the experimental data. The USPEX calculation and the experimental spectrum do not need to be exactly at the same pressure, but the two pressures need to be *close*, that is, no more than 20 GPa apart.

Since we have a pressure difference, we first translate each calculated structure to the experimental pressure by using the Birch-Murnaghan equation

$$\Delta P = \frac{3B_0}{2}\left[\left(\frac{V_0}{V}\right)^{\frac{7}{3}} - \left(\frac{V_0}{V}\right)^{\frac{5}{3}}\right]\left\{1 + \frac{3}{4}\left(B_0' - 4\right)\left[\left(\frac{V_0}{V}\right)^{\frac{2}{3}} - 1\right]\right\}$$

where ΔP is the pressure difference, V_0 is the volume of the unit cell at the calculated pressure, V is the volume at the experimental pressure, B_0 is the bulk modulus and B_0' is the derivative of the bulk modulus with respect to pressure. For B_0 and B_0' we take the average values $B_0 = 300$ and $B_0' = 3$.

Assuming that, for small pressure variations, also V_0/V will be small, we approximate the Birch-Murnaghan equation to a second order Taylor expansion in V_0/V and we get the volume V at the experimental pressure:

$$V = \frac{300}{150 + \sqrt{22500 + 300\Delta P}} V_0$$

Then we define the following scaling factor

$$k = \sqrt[3]{\frac{300}{150 + \sqrt{22500 + 300\Delta P}}}$$

that will be used to rescale the lattice parameters of all the calculated structures. After this rescaling, the relaxed structures are symmetrized with a tolerance of 0.2 Å and the theoretical XRD spectra are computed. Both the theoretical and the experimental spectra are in the form of a series of peaks.

In the case of powder XRD spectra, for each peak we know the diffraction angle and the relative intensity, while the intensity of the highest peak in each spectrum has been conventionally given the value 100. We define a *match* between a theoretical and an experimental peak the case when the two peaks are less than *match_tol* degrees apart, regardless of their intensities. The fitness between a calculated powder XRD spectrum and the experimental spectrum is defined by the following fitness function

$$F = \sum_{i,j}^{match}\left(\frac{h_i^{exp} - h_j^{th}}{100}\right)^2\left(\frac{h_i^{exp}}{100}\right)^2 + \sum_{i}^{exp\,rest}\left(\frac{h_i^{exp}}{100}\right)^2 + \sum_{i}^{th\,rest}\left(\frac{h_i^{th}}{100}\right)^2$$

where the h_i are diffraction intensities. The first term of the fitness is a sum over the matched peaks, and each addend of this sum will be smaller the more the intensities of the two matched peaks are close to each other. The second term is a sum over the experimental peaks that are left after our matching, we call these peaks *experimental rest*. The third term, analogously, is a sum over the *theoretical rest*. It is clear that a low value of F gives a good agreement between calculated and experimental powder XRD spectra, allowing a quick identification of promising candidates in the USPEX output.

B Single crystal spectra comparison

In the case of single crystal XRD spectra, for each reflection we know the Miller indices (h, k, l), its intensity I and its uncertainty of measure σ. As fitness we use the weighted R-factor, given by

$$wR = \sqrt{\frac{\sum \frac{1}{\sigma^2}(I_{exp} - I_{th})^2}{\sum \frac{1}{\sigma^2} I_{exp}^2}}$$

where the sums run over all the experimental reflections and theoretical reflections at the same Miller indices (h, k, l). If an experimental reflection is not matched by any theoretical reflection an error is returned, while in the opposite case the unmatched theoretical reflection is simply ignored.