eDoping

Release 0.5.1

Jianbo Zhu, Jingyu Li, Peng-Fei Liu

Jul 26, 2026

Explore More

1 Feature List

- Streamlined calculation of formation energies for charged point defects
- Entirely command-line based, requiring no programming or complex scripts
- Supports estimation of elemental chemical potentials in multi-component systems
- Quickly access competing phase structures and formation energies from OQMD/MaterialsProject.
- Calculation of correction terms for defect formation energies
- Determination of the self-consistent Fermi energy

2 Installation

The eDoping package is based on Python3, so ensure that it is correctly installed. If the network is available, the most efficient way to install (or update) the eDoping package is via `pip` :

```
$ pip install -U eDoping
```

If you do not have internet access or are interested in the source code, you can download the package from GitHub or use `git` :

```
$ git clone https://github.com/JianboHIT/eDoping.git
```

For users in mainland China, the source code can also be obtained from Gitee, similar to the above:

```
$ git clone https://gitee.com/joulehit/eDoping.git
```

After downloading the source code, enter the folder (if it's a compressed file, extract it first), and ensure a good network connection. We can automatically install the program and all dependent libraries (it actually only depends on numpy and scipy) using the `pip` (or `pip3`) tool:

```
$ pip install .
```

After installation is complete, we can use the eDoping package with the `edp` command. The `-h` (or `--help`) option can print help information and also check if the installation was successful:

```
$ edp -h
usage: edp [-h] [-v | -q] Subcommand ...
```

(continues on next page)

(continued from previous page)

```
Point Defect Formation Energy Calculation - v0.5.1

optional arguments:
  -h, --help       show this help message and exit
  -v, --verbosity  increase output verbosity
  -q, --quiet      only show key output

Tips:
  Subcommand       Description
    cal            Calculate defect formation energy
    energy         Read final energy from OUTCAR
    ewald          Read Ewald from OUTCAR
    volume         Read volume from OUTCAR
    epsilon        Read epsilon from OUTCAR
    evbm           Read VBM from EIGENVAL
    fixchg         Produce charge-fixed inputs
    boxhyd         Place a single hydrogen atom in the box
    refine         Refine the unit cell
    replace        Replace atoms X by Y
    groupby        Group atoms by radial distribution function
    diff           Show difference between two POSCAR
    query          Fetch data from OQMD website
    chempot        Calculate chemical potential
    react          Evaluate reactions between two compounds
    trlevel        Calculate transition levels
    scfermi        Calculate sc-fermi level
    fzfermi        Calculate fz-fermi level

>>>>>>>>>> Citation of EDOPING <<<<<<<<<<
If you have used EDOPING, please cite the following article:
 J. Zhu, J. Li, P. Liu, et al, eDoping: A high-throughput software
 package for evaluating point defect doping limits in semiconductor
 and insulator materials, Materials Today Physics, 55 (2025) 101754.
 DOI: 10.1016/j.mtphys.2025.101754
```

We can further check the help information of subcommands:

```
$ edp replace -h
usage: edp replace [-h] [-p fa fb fc] [-m] [-s] [-i FILENAME] [-o FILENAME] X Y

positional arguments:
  X                     Label of the previous atom, e.g., 'Sb12' represents the 12th Sb␣
↪atom
  Y                     Label of the present atom, e.g., 'Bi' (the index is typically␣
↪omitted)

optional arguments:
  -h, --help            show this help message and exit
  -p fa fb fc, --position fa fb fc
                        Fractional coordinates of new interstitial atom
  -m, --multiple        Enable multiple atomic substitutions for creating alloy␣
↪structures
```

(continues on next page)

(continued from previous page)

```
  -s, --shuffle         Similar to the -m option, but additionally shuffles the positions
  -i FILENAME, --input FILENAME
                        Input filename(default: POSCAR)
  -o FILENAME, --output FILENAME
                        Output filename(default: POSCAR)
```

At this point, we have successfully installed the eDoping package.

Optionally: Python, being an interpreted language, requires loading the entire environment or virtual environment every time it is run. This is very convenient for personal devices, but it becomes inconvenient for large public computing platforms. One solution is to package the program into a standalone executable, making it independent of the Python environment like regular programs. We considered this from the start of program development, carefully controlling dependencies on third-party libraries and implementing as much as possible using Python's standard library. In the program source package, we included a standalone folder containing a compile_for_linux.sh script, which assists in creating standalone executables. Here, we need to prepare a clean Python virtual environment and install pyinstaller and other eDoping dependencies, then run the following command:

```
$ cd standalone
$ bash compile_for_linux.sh
```

Once the script runs successfully, an executable program `edp` is created in standalone/dist, which can be moved to any desired location for convenient daily use.

3 Quick Start

Ensure the eDoping package is correctly installed (refer to the *Installation* section for details), and you can print help information using `edp -h`. Density functional calculations use `VASP` software as an example; theoretically, other computational software could be used, but current interfaces do not fully support them yet and require further improvement.

3.1 Directory Structure

Taking NbFeSb with Mn or Ni interstitials as an example (see `examples/` in the Github/Gitee repository), we recommend organizing files in the following directory structure (prefix numbers make it convenient for tab completion):

```
NbFeSb_Interstitials
├── 1.perfect
├── 2-1.defect-Mn_i
│   ├── charge_+1
│   ├── charge_+2
│   ├── charge_+3
│   ├── charge_-1
│   ├── charge_-2
│   ├── charge_-3
│   ├── charge_0
│   └── relax
├── 2-2.defect-Ni_i
│   ├── charge_+1
│   ├── charge_+2
│   ├── charge_+3
│   ├── charge_-1
│   │
```

(continues on next page)

(continued from previous page)

```
│   ├── charge_-2
│   ├── charge_-3
│   ├── charge_0
│   └── relax
├── 3.phases
│   ├── NbFeSb_with_Mn
│   ├── NbFeSb_with_Ni
│   ├── elemental_Fe
│   ├── elemental_Mn
│   ├── elemental_Nb
│   ├── elemental_Ni
│   └── elemental_Sb
├── 4-1.corr-dielectric
├── 4-2.corr-hydrogen
├── EDOPING.Mn_i.in
└── EDOPING.Ni_i.in
```

3.2 Energy Calculation

This is the core time-consuming part of defect calculations. VASP software needs to be called to calculate the energy of the perfect supercell and defect supercells in all charge states (ensure all structures are reasonably relaxed to convergence), specifically the subfolders under `1.perfect` and `2-X.defect-XX` directories.

To simulate defects with varying charge states, the total electron number of the system must be specified via the NELECT parameter in the INCAR file. To streamline this process, the *edp fixchg* command can automatically generate charged defect calculation folders (e.g., `charge_+1`, `charge_-1`) from the neutral calculation directory:

```
$ edp fixchg -i charge_0 +1 -1 +2 -2 +3 -3
```

The `charge_0` directory contains files required for self-consistent calculations of the neutral system. The command will duplicate this directory into `charge_+1`, `charge_-1`, etc., while adjusting the NELECT parameter in their INCAR files to set the system's net charge to the specified value.

3.3 Chemical Potential Calculation

In defect formation energy calculations, the atomic chemical potential term is used to express the energy change due to the non-conservation of atom types and numbers between the defect supercell and the perfect supercell. The chemical potential of an atom consists of two parts, $\mu_i = \mu_i^\Theta + \Delta\mu_i$, where μ_i^Θ represents the average energy per atom in the elemental bulk, obtained through theoretical calculations or experimental methods; $\Delta\mu_i$ is determined within a range by thermodynamic stability conditions. Strictly speaking, we need to calculate the formation energies of all potential competing phases, which are obtained by querying databases such as OQMD, MaterialsProject, AFLOW, etc.

Here, we provide a query command *edp query*, which can directly retrieve the formation energies of all competing phases from the OQMD database (with support for the *Materials Project* database starting from v0.4, see *edp query* for details). For example, for NbFeSb with Mn atom defects, you can use the following command to obtain the formation energies of all competing phases with Ehull < 0.01 eV/atom (run in the `3.phases/NbFeSb_with_Mn` directory):

```
$ edp query NbFeSb -x Mn --ehull 0.01
```

The results are stored in the *EDOPING.cmpot* file. When the `--ehull` option is omitted, the command retrieves energy data for both stable and metastable competing phases. Additionally, structural files (in POSCAR format) for all competing phases can be downloaded via the `-s/--structure` option to enable more accurate energy calculations:

```
$ edp query NbFeSb -x Mn --ehull 0.01 --structure
```

Then, manually prepare the *EDOPING.cmpot* file based on the calculation results. Using the *EDOPING.cmpot* file, the elemental chemical potential can be determined according to the chemical environment using the `edp chempot` command:

```
$ edp chempot -n
```

Note that the `-n` (or `--norm`) option indicates that the formation energy in the file is in units of eV/atom. If the formation energy reflects the total energy of the supercell for the respective composition, then this option is not needed. The *EDOPING.cmpot* file will be automatically read, and the chemical potentials ($\Delta\mu_i$) under different environments will be printed on the screen.

3.4 Correction Terms

In point defect calculations, due to finite size limitations, the obtained formation energies often require corrections, represented by the term E_{corr} in the formula. Among various correction terms, image charge correction requires additional input of dielectric constants and Madelung constants. To apply this correction mechanism, you can follow these steps to calculate the dielectric constants and Madelung constants using VASP.

In VASP, for the original primitive supercell of the perfect structure, you can use the following INCAR parameters to obtain the dielectric constants (refer to `4-1.corr-dielectric/INCAR`):

```
Global Parameters
ISTART =  0            (Read existing wavefunction; if there)
ISPIN  =  1            (Non-Spin polarised DFT)
LREAL  = .FALSE.       (Projection operators: automatic)
ENCUT  =  500          (Cut-off energy for plane wave basis set, in eV)
PREC   =  Accurate     (Precision level)
LWAVE  = .FALSE.       (Write WAVECAR or not)
LCHARG = .FALSE.       (Write CHGCAR or not)

Static Calculation
NSW    = 1
IBRION =  8
ISMEAR =  0            (gaussian smearing method)
SIGMA  =  0.01         (please check the width of the smearing)
NELM   =  60           (Max electronic SCF steps)
EDIFF  =  1E-08        (SCF energy convergence; in eV)

Macroscopic Dielectric Tensor
LEPSILON = .TRUE.
LPEAD = .TRUE.
```

Upon calculation completion, the dielectric tensor data can be extracted from OUTCAR files using the `edp epsilon` command:

```
$ edp epsilon -f 4-1.corr-dielectric/OUTCAR
HEAD OF MICROSCOPIC STATIC DIELECTRIC TENSOR (INDEPENDENT PARTICLE, excluding Hartree␣
↪and local field effects)
------------------------------------------------------
25.438394     0.000000    -0.000000
0.000000    25.438394     0.000000
```

(continues on next page)

(continued from previous page)

```
-0.000000    -0.000000    25.438394
------------------------------------------------------

MACROSCOPIC STATIC DIELECTRIC TENSOR (including local field effects in DFT)
------------------------------------------------------
24.482055     0.000000    -0.000000
0.000000    24.482055    -0.000000
-0.000000     0.000000    24.482055
------------------------------------------------------

MACROSCOPIC STATIC DIELECTRIC TENSOR (including local field effects in DFT)
------------------------------------------------------
24.482055     0.000000    -0.000000
0.000000    24.482055    -0.000000
-0.000000     0.000000    24.482055
------------------------------------------------------

MACROSCOPIC STATIC DIELECTRIC TENSOR IONIC CONTRIBUTION
------------------------------------------------------
19.549608    -0.000000    -0.000000
-0.000000    19.549608     0.000000
-0.000000     0.000000    19.549608
------------------------------------------------------
```

From the displayed results, the ionic contribution to the dielectric constant is 19.55 (the last tensor), the electronic contribution is 24.48 (the second-to-last tensor), so the total dielectric constant is 43.93.

To determine the Madelung constant, place a single hydrogen atom in a supercell-equivalent simulation box. After performing a VASP self-consistent field (SCF) calculation, the corresponding Madelung constant will be recorded in the OUTCAR file. The provided *`edp boxhyd`* command generates a single-hydrogen POSCAR file with identical dimensions to the supercell POSCAR (execute within the `4-2.corr-hydrogen` directory containing the original supercell POSCAR):

```
$ edp boxhyd
```

The POSCAR.H file will be generated upon execution. Perform a self-consistent field (SCF) calculation using this file, then extract the Madelung constant with the *`edp ewald`* command:

```
$ edp ewald -f 4-2.corr-hydrogen/OUTCAR
Final (absolute) Ewald: 1.7152
```

The Madelung constant for this supercell is 1.7152.

3.5 Post-processing

Prepare the *EDOPING.in* file based on the previous information as follows:

```
DPERFECT = 1.perfect
DDEFECT  = 2-1.defect-Mn_i
CMPOT    = 0 -9.0147
VALENCE  = -3 -2 -1 0 1 2 3
# PREFIX   = charge_
```

(continues on next page)

```
# DDNAME   = auto
EVBM      = inf
ECBM      = inf
PENERGY   = inf
PVOLUME   = inf
EWALD     = 1.7152
EPSILON   = 44.03
BFTYPE    = 2
EMIN      = -1
EMAX      = 2
NPTS      = 3001
```

Then invoke the *edp cal* command to perform the computation:

```
$ edp cal -i EDOPING.in
```

Upon completion, the *EDOPING.log* and *EDOPING.dat* files will be generated, which record the program's execution log and the calculation results, respectively.

4 Point Defect Formation Energy Calculation

Within the framework of first-principles calculations, all calculations here are centered around energy (also referred to as enthalpy) calculations. For a defect D with charge q, its formation energy is defined as:

$$\Delta H_D^q(E_F) = E_D^q - E_{perfect} - \sum_i n_i \mu_i + qE_F + E_{corr}$$

Here, E_D^q represents the energy of the supercell with defect D carrying charge q, $E_{perfect}$ denotes the energy of the corresponding perfect supercell, μ_i stands for the chemical potential of atoms lost (when $n_i < 0$) or added (when $n_i > 0$) during defect formation, n_i is the number of corresponding atoms, E_F is the Fermi level of the actual defect system, and E_{corr} includes energy corrections such as those from periodic boundary conditions and electrostatic potential changes. Generally, we cannot precisely determine the position of the system's Fermi level, but we know it lies near the band gap. Therefore, we typically provide the ΔH_D^q - E_F relationship curve, expressing the formation energy as a function of the Fermi level. Next, we will explain the calculation of each term and the final data processing steps in detail.

4.1 Defect Supercell Construction and System Energy Calculation

The energy of the perfect supercell is relatively easy to obtain, so we'll start with energy calculations for defect structures. We first consider the supercell structure construction for single-point defects, including vacancies, substitutions, and interstitials. For vacancy and interstitial defects, typically, we can directly manually modify the POSCAR file to obtain the defect structure, as we do not need to change the order of the atom position list during this process. For substitution defects, you can construct structures using the *edp replace* command from a POSCAR file. You can also use crystallography visualization tools to help generate defect structures, such as the free VESTA software.

When considering more complex defects, the number of possible supercell structure configurations can increase significantly. Utilizing structural symmetry can effectively reduce the required computational effort. For relatively simple cases, we can use structural visualization programs to observe and analyze, excluding symmetrically equivalent structures, but dealing with complex structures becomes difficult. Additionally, software and programs specifically for handling this aspect are quite limited. In our software, we have integrated a *edp groupby* command to assist in filtering non-equivalent structures. When we need to introduce an additional defect into a structure that already contains a defect, we abandon considering equivalence based on traditional symmetry, and instead analyze based on the neighboring environment, grouping atoms with similar surroundings to identify representative structures. Due to the local properties of point defects, neighbor analysis may provide a more direct and effective means to determine candidate composite defect configurations.

Once the defect structures are constructed, we can use the `edp diff` command to compare the differences between the original supercell and the current supercell to ensure that the constructed configuration is as desired.

See also:

- `replace` - Generate Atomic Substitution Structure
- `groupby` - Non-equivalent Atom Position Analysis
- `diff` - Crystal Structure Comparison and Analysis

Following defect structure construction, structural relaxation of the supercell is required to achieve converged energy values. Furthermore, varying the electron count in each defect system (via the NELECT parameter in VASP's INCAR file) is necessary to simulate different charge states, ultimately acquiring their corresponding energy values.

See also:

- `fixchg` - Prepare Calculation Files with Different Charge Numbers

For VASP software, if the structural optimization/self-consistent calculation ends successfully, we can use the `grep` command combined with `tail` to read the energy from the OUTCAR file:

```
$ grep 'energy  without entropy' OUTCAR | tail -n 1
  energy  without entropy=     -755.64631647  energy(sigma->0) =     -755.65114440
```

Here, the system energy is -755.651 eV. You can also extract it from the OUTCAR file using the `edp energy` command.

See also:

- `energy` - Read System Energy Value from OUTCAR.

4.2 Chemical Potential Calculation and Database Usage

After we complete the construction and calculations for defect structures, an important point should be noted: it is quite challenging to maintain atomic number conservation between defect structures and the corresponding perfect structures. To evaluate the formation energy of defects, we must eliminate the energy difference of the atoms themselves, which we refer to as the chemical potential here. A straightforward idea is that we can use calculations from corresponding elemental materials to evaluate the energy of a single atom. However, this proves to be a very rough estimate, accompanied by significant systematic errors. We can imagine that the energy of atoms in our target compound must be lower than that in the elemental state; otherwise, our target compound would decompose into elemental substances to lower the system's energy. Here, the energy of atoms in the compound is generally called the chemical potential μ_i, the energy of atoms in the element is called the standard chemical potential $\mu_i^\ominus$, so we have $\mu_i = \mu_i^\ominus + \Delta\mu_i$. Our goal here is to determine the magnitude of $\Delta\mu_i$. Unfortunately, there is currently no way to provide an exact value of $\Delta\mu_i$; instead, we can make a range estimation and further determine its value based on specific experimental conditions.

According to our previous discussion, we can clearly know that there must be

$$\Delta\mu_i < 0$$

On the other hand, according to energy conservation, we know that the change in internal energy of all elements in the compound is the formation enthalpy of the compound ΔH_{comp}, that is,

$$\sum_i c_i \cdot \Delta\mu_i = \Delta H_{comp}$$

Here, assuming $c_1 + c_2 + \ldots + c_N = 1$, and ΔH_{comp} is the formation enthalpy per average atom. We can thus determine the lower bound of $\Delta\mu_i$:

$$c_i \cdot \Delta\mu_i > \Delta H_{comp}$$

In experiments, μ_i is referred to as "rich" when $\Delta\mu_i = 0$, and "poor" when $\Delta\mu_i = \Delta H_{comp}/c_i$.

For binary compounds, it is easy to notice that when one element's chemical potential is "rich", the other element's chemical potential will inevitably be "poor". Therefore, we usually present the formation energy of defects under conditions where each of the two different elements is "rich", reflecting the scenario where the chemical environment transitions from one extreme to the other, with real experimental conditions likely falling between these two extremes.

As the number of elements increases to three, the concepts of "poor" and "rich" become more complex, since when one atom is "rich", the states of the other two atoms are not easily determined. We have to conduct a detailed classification discussion to approximate the experimental environment as closely as possible.

Nevertheless, this range is still too coarse. Currently, the most effective way to further narrow the range of chemical potentials is to consider the inclusion of competing phases. Based on our previous analysis, it is easy to understand that the sum of the chemical potentials of atoms in the target compound must be less than the formation enthalpy of competing phases; otherwise, more "stable" competing phases would form instead of our target phase in experiments. Thus, we can introduce a series of inequality constraints:

$$\sum_i c_{j,i} \cdot \Delta\mu_{j,i} \leq \Delta H_{comp,j}$$

Here, the subscript j stands for the j-th competing phase. Under this series of inequality constraints, the range of chemical potentials becomes more refined, and the shape of the feasible region becomes more complex.

In our program design, we abandon discussions about the shape of the feasible region and instead focus directly on the range of chemical potentials for each element. Particularly for multi-component compounds, when the number of elements is N, the dimension of the feasible region is N-1. Trimming the feasible region with competing phases makes its shape extremely complex. At this point, we are not concerned with all the vertex cases but instead wish to directly know the range of chemical potentials for certain elements of interest. To this end, our program has been designed with the `edp chempot` command to directly obtain the chemical potential ranges for different elements.

Manually handling a large number of competing phases is a labor-intensive and time-consuming task. Therefore, we provide the `edp query` command, which allows direct retrieval of all competing phase structure files from the database. Additionally, it enables the simultaneous extraction of formation enthalpies for competing phases from the database, facilitating verification of one's own calculation results. On the other hand, within the framework of first-principles calculations, the energy values of the system depend on the pseudopotentials and calculation software, but the formation enthalpy of a material has relatively good stability. When high precision is not required or for initial explorations, the formation enthalpy of competing phases from the database can be used to determine the element's chemical potential range, thereby accelerating our workflow.

See also:

- `chempot` - Estimate Atomic Chemical Potentials Based on Compound and Competing Phase Formation Enthalpy
- `query` - Retrieve Competing Phase Structures and Formation Enthalpies from Database

> **Warning:** Due to the nature of high-throughput calculations in the database, the precision of the formation enthalpies is very limited. Therefore, it is recommended for preliminary exploration only. We cannot guarantee the reliability of the data obtained from the database. Furthermore, this feature was developed primarily to facilitate communication and learning. If any infringement occurs, we will immediately disable this feature.

5 Command Line Usage Reference

You can view all supported commands using `edp -h`. The general format for using commands is:

```
$ edp [-v|-q] <command> --option1 --option2 [inputfile]
```

The `-v` option increases the verbosity of the screen output, while the `-q` option suppresses the output as much as possible. You can use the `-h` option with subcommands to see the supported operations, for example, to check the options supported by the *edp chempot* command:

```
$ edp chempot -h
```

Next, we will introduce the supported subcommands (listed in alphabetical order):

boxhyd

Generate a same-sized POSCAR file containing only a single hydrogen atom.

cal

Calculate the defect formation energy vs. Fermi level based on the configuration file (specified by `-i`/`--input`, default is *EDOPING.in*). To get started, generate a template input file using the `--template` option:

```
$ edp cal --template
```

This generates a default *EDOPING.in* file without triggering a calculation. (Custom filenames are unsupported; beware of overwriting existing files). You can modify it as needed before running the actual calculation.

New in version 0.5: The `--template` option.

chempot

Solve the range of elemental chemical potentials

Here we need to prepare an input file (default filename is *EDOPING.cmpot*), where the first line starting with '#' contains element names separated by spaces. Following this, list the elemental ratios of all compounds under consideration, along with their respective energy values. The first compound listed (i.e., the second line in the file) is recognized by the program as the target compound, our base phase material.

Important Note: When handling element ratios and energies, due to personal habits and differences in database format specifications, we need to be very careful about normalization issues:

- Element Ratio Format: (1) Number of each atom in the unit cell (2) Simplest atomic ratio (3) Normalized ratio
- Compound Enthalpy Representation: (A) Total enthalpy of the unit cell (B) Average enthalpy per atom
- Enthalpy Reference: (I) Absolute enthalpy, as given by the computation program (II) Formation enthalpy, i.e., the difference in enthalpy relative to its elemental state

Internally, the program is handling expressions like the one below:

$$\frac{1}{C}\sum_i c_i \cdot \mu_i \leq \mu$$

Here, i refers to different compounds, c_i is the element ratio from the input file, μ is the compound enthalpy from the input file; if the `-n` (`--norm`) option is used, then $C = \sum_i c_i$, otherwise $C = 1$. Simply put, to obtain correct results, you need to specify the `-n` (`--norm`) option for scenarios (1+B) and (2+B), but avoid it for scenarios (1+A) and (3+B). Due to lack of necessary information, we cannot handle scenarios (2+A) and (3+A), requiring the user to perform the necessary data processing.

As for the reference of the enthalpy, the basic principle is: the reference for solving the chemical potentials is the initial reference given for compound enthalpy. If all provided values are (I) absolute enthalpy, then the given potentials are absolute chemical potentials; if all provided values are (II) formation enthalpy, then the given chemical potentials are differences from the respective elemental states.

Since the *EDOPING.in* file requires specifying absolute chemical potentials, the `--refs` option can be used to define standard chemical potentials for each elemental species, thereby obtaining the corresponding absolute chemical potentials. Internally, the program adds these reference values to the final calculated chemical potential values.

New in version 0.3: The `--refs` option.

diff

Compares atom differences in two POSCAR files with the same lattice vectors to identify point defects. For example, check Mn interstitials in a NbFeSb supercell:

```
$ cd examples/NbFeSb_Interstitials
$ edp diff 1.perfect/POSCAR 2-1.defect-Mn_i/relax/POSCAR
  No.    f_a     f_b     f_c     previous    present
 i 1     0.1250  0.1250  0.1250   Vac1         Mn1
```

Here, `i` stands for interstitial defect (`v` stands for vacancy, `s` for substitution).

energy

Read the final energy from the OUTCAR file. Available options:

- `--ave/--average`: Output energy per atom instead of the total cell energy.
- `-c/--count`: Count specified atoms in the cell. For instance, in a system with 32 Nb, 32 Fe, and 32 Sb, you can count the elements of interest (here: Ti, Co, Fe, Sb) like this:

  ```
  $ edp energy -c "Ti Co Fe Sb"
  Final energy ( 0  0  32  32 ): -757.0157 eV/cell
  ```

  The numbers in parentheses represent the respective atom counts for Ti, Co, Fe, and Sb. This is highly useful for preparing the *EDOPING.cmpot* file (especially when combined with the `-q` flag):

  ```
  $ edp -q energy -c "Ti Co Fe Sb"
   0  0  32  32  -757.0157
  ```

New in version 0.5: The `-c/--count` option.

epsilon

Read and print the dielectric constants from the OUTCAR file.

ewald

Read and print the Madelung constant from the OUTCAR file.

fixchg

Automatically generate calculation folders for charged defects by specifying the neutral structure's self-consistent calculation folder using the `-i/--inputdir` option (default is `charge_0`). The program calculates the total number of electrons from the POTCAR file and then automatically determines the number of electrons (NELECT parameter) based on the given system charge. It is recommended to prepare the neutral structure's calculation folder first, run this command to generate the charged defect folders, and then submit them in batches.

groupby

Group atoms in a POSCAR using the radial distribution function to identify complex defects at inequivalent sites. For example, introducing an Fe vacancy into an NbFeSb supercell with an Mn interstitial defect can be time-consuming if you calculate each Fe site individually. A practical approach is to group Fe atoms by their local environments. Within the same group, Fe atoms should exhibit similar defect behavior. Assume the following POSCAR file is of a NbFeSb supercell containing an Mn interstitial:

```
$ cd examples/NbFeSb_Interstitials/2-1.defect-Mn_i/relax/
$ edp groupby -f POSCAR Fe
Group #1: Fe1, Fe2, Fe9, Fe11, Fe17, Fe21
Group #2: Fe3, Fe4, Fe5, Fe6, Fe10, Fe12, Fe13, Fe15, Fe18, Fe19, Fe22, Fe23
Group #3: Fe7, Fe8, Fe14, Fe16, Fe20, Fe24
Group #4: Fe25, Fe26, Fe27, Fe28, Fe29, Fe30, Fe31, Fe32

===============================================================================
No.|     Group #1      |     Group #2      |     Group #3      |     Group #4
---+-------------------+-------------------+-------------------+-------------------
 0 |  (0.0, 'Fe', 1)   |  (0.0, 'Fe', 1)   |  (0.0, 'Fe', 1)   |  (0.0, 'Fe', 1)
 1 |  (2.6, 'Nb', 4)   |  (2.6, 'Nb', 4)   |  (2.6, 'Nb', 4)   |  (2.6, 'Nb', 4)
 2 |  (2.6, 'Sb', 4)   |  (2.6, 'Sb', 4)   |  (2.6, 'Sb', 4)   |  (2.6, 'Sb', 4)
 3 |  (3.0, 'Mn', 1)   | (4.2, 'Fe', 12)   | (4.2, 'Fe', 12)   | (4.2, 'Fe', 12)
 4 | (4.2, 'Fe', 12)   | (4.9, 'Nb', 12)   | (4.9, 'Nb', 12)   | (4.9, 'Nb', 12)
 5 | (4.9, 'Nb', 12)   | (4.9, 'Sb', 12)   | (4.9, 'Sb', 12)   | (4.9, 'Sb', 12)
 6 | (4.9, 'Sb', 12)   |  (6.0, 'Fe', 6)   |  (6.0, 'Fe', 6)   |  (5.2, 'Mn', 1)
 7 |  (6.0, 'Fe', 6)   | (6.5, 'Nb', 12)   | (6.5, 'Nb', 12)   |  (6.0, 'Fe', 6)
 8 | (6.5, 'Nb', 12)   | (6.5, 'Sb', 12)   | (6.5, 'Sb', 12)   | (6.5, 'Nb', 12)
 9 | (6.5, 'Sb', 12)   |  (6.7, 'Mn', 2)   | (7.3, 'Fe', 24)   | (6.5, 'Sb', 12)
10 | (7.3, 'Fe', 24)   | (7.3, 'Fe', 24)   | (7.7, 'Nb', 16)   | (7.3, 'Fe', 24)
11 | (7.7, 'Nb', 16)   | (7.7, 'Nb', 16)   | (7.7, 'Sb', 16)   | (7.7, 'Nb', 16)
12 | (7.7, 'Sb', 16)   | (7.7, 'Sb', 16)   | (8.4, 'Fe', 12)   | (7.7, 'Sb', 16)
13 | (8.4, 'Fe', 12)   | (8.4, 'Fe', 12)   | (8.8, 'Nb', 24)   | (8.4, 'Fe', 12)
14 | (8.8, 'Nb', 24)   | (8.8, 'Nb', 24)   | (8.8, 'Sb', 24)   | (8.8, 'Nb', 24)
15 | (8.8, 'Sb', 24)   | (8.8, 'Sb', 24)   |  (8.9, 'Mn', 4)   | (8.8, 'Sb', 24)
16 |  (8.9, 'Mn', 1)   | (9.4, 'Fe', 24)   | (9.4, 'Fe', 24)   | (9.4, 'Fe', 24)
17 | (9.4, 'Fe', 24)   | (9.8, 'Nb', 12)   | (9.8, 'Nb', 12)   | (9.8, 'Nb', 12)
18 | (9.8, 'Nb', 12)   | (9.8, 'Sb', 12)   | (9.8, 'Sb', 12)   | (9.8, 'Sb', 12)
19 | (9.8, 'Sb', 12)   | (10.3, 'Fe', 8)   | (10.3, 'Fe', 8)   |  (9.9, 'Mn', 3)
20 | (10.3, 'Fe', 8)   | (10.6, 'Nb', 24)  | (10.6, 'Nb', 24)  | (10.3, 'Fe', 8)
21 | (10.6, 'Nb', 24)  | (10.6, 'Sb', 24)  | (10.6, 'Sb', 24)  | (10.6, 'Nb', 24)
22 | (10.6, 'Sb', 24)  | (10.7, 'Mn', 2)   | (11.1, 'Fe', 48)  | (10.6, 'Sb', 24)
23 | (11.1, 'Fe', 48)  | (11.1, 'Fe', 48)  | (11.4, 'Nb', 36)  | (11.1, 'Fe', 48)
24 | (11.4, 'Nb', 36)  | (11.4, 'Nb', 36)  | (11.4, 'Sb', 36)  | (11.4, 'Nb', 36)
25 | (11.4, 'Sb', 36)  | (11.4, 'Sb', 36)  | (11.9, 'Fe', 6)   | (11.4, 'Sb', 36)
26 | (11.9, 'Fe', 6)   | (11.9, 'Fe', 6)   | (12.2, 'Nb', 12)  | (11.9, 'Fe', 6)
27 | (12.2, 'Nb', 12)  | (12.2, 'Nb', 12)  | (12.2, 'Sb', 12)  | (12.2, 'Nb', 12)
28 | (12.2, 'Sb', 12)  | (12.2, 'Sb', 12)  | (12.3, 'Mn', 4)   | (12.2, 'Sb', 12)
29 | (12.3, 'Mn', 4)   | (12.6, 'Fe', 36)  | (12.6, 'Fe', 36)  | (12.6, 'Fe', 36)
30 | (12.6, 'Fe', 36)  | (12.9, 'Nb', 28)  | (12.9, 'Nb', 28)  | (12.9, 'Nb', 28)
===============================================================================
```

You can see that the 32 Fe atoms are divided into 4 groups. Each cell contains three entries: distance, atom type, and count. For instance, Fe1, Fe2, Fe9, Fe11, Fe17, and Fe21 all belong to group #1, sharing identical neighboring atoms within 12.6 angstroms. Specifically, they are 3.0 angstroms away from the nearest Mn atom and 8.9 angstroms from the second-nearest Mn atom. To focus on Mn atoms clearly, you can use the `--grep Mn` option to keep only lines with Mn atoms:

```
$ edp groupby -f POSCAR Fe --grep Mn
Group #1: Fe1, Fe2, Fe9, Fe11, Fe17, Fe21
Group #2: Fe3, Fe4, Fe5, Fe6, Fe10, Fe12, Fe13, Fe15, Fe18, Fe19, Fe22, Fe23
Group #3: Fe7, Fe8, Fe14, Fe16, Fe20, Fe24
```

(continues on next page)

(continued from previous page)

```
Group #4: Fe25, Fe26, Fe27, Fe28, Fe29, Fe30, Fe31, Fe32

===============================================================================
No.|     Group #1      |     Group #2      |     Group #3      |     Group #4
---+-------------------+-------------------+-------------------+------------------
 3 |  (3.0, 'Mn', 1)   | (4.2, 'Fe', 12)   | (4.2, 'Fe', 12)   | (4.2, 'Fe', 12)
 6 | (4.9, 'Sb', 12)   |  (6.0, 'Fe', 6)   |  (6.0, 'Fe', 6)   |  (5.2, 'Mn', 1)
 9 | (6.5, 'Sb', 12)   |  (6.7, 'Mn', 2)   | (7.3, 'Fe', 24)   | (6.5, 'Sb', 12)
15 | (8.8, 'Sb', 24)   | (8.8, 'Sb', 24)   |  (8.9, 'Mn', 4)   | (8.8, 'Sb', 24)
16 |  (8.9, 'Mn', 1)   | (9.4, 'Fe', 24)   | (9.4, 'Fe', 24)   | (9.4, 'Fe', 24)
19 | (9.8, 'Sb', 12)   | (10.3, 'Fe', 8)   | (10.3, 'Fe', 8)   |  (9.9, 'Mn', 3)
22 | (10.6, 'Sb', 24)  | (10.7, 'Mn', 2)   | (11.1, 'Fe', 48)  | (10.6, 'Sb', 24)
28 | (12.2, 'Sb', 12)  | (12.2, 'Sb', 12)  | (12.3, 'Mn', 4)   | (12.2, 'Sb', 12)
29 | (12.3, 'Mn', 4)   | (12.6, 'Fe', 36)  | (12.6, 'Fe', 36)  | (12.6, 'Fe', 36)
===============================================================================
```

query

Retrieve competing phase information from open materials databases. Specify the database name using the `--backend` option:

- `OQMD` (default): The Open Quantum Materials Database (https://www.oqmd.org)
- `MP` : The Materials Project (https://next-gen.materialsproject.org)

This command generates the input data file *EDOPING.cmpot* used for estimating chemical potentials, with the simplified atomic ratios and the average formation enthalpy per atom of the compounds. Therefore, to correctly calculate the chemical potential $\Delta\mu_i$, you need to add the `-n` (`--norm`) option when using `edp chempot`.

Hint: Before using the *Materials Project* database, the `MP_API_KEY` environment variable must be properly configured (register and obtain an API key on the Materials Project - API page). Replace <your_api_key> in the following command with your API key:

```
$ export MP_API_KEY=<your_api_key>
```

For private platforms, you can add this to the `~/.bashrc` file to avoid repeated configuration.

Tip: Ensure a stable network connection when using this command, and be mindful of database access frequency limits. It is not recommended to use this command repeatedly within a short period. Typically, you can first check the database website for better visualization results, and then use this command to retrieve the data.

New in version 0.4: Support for the *Materials Project* backend database.

react

Evaluate reaction energies for all possible pathways between two compounds based on their formation enthalpies.

This requires the *EDOPING.cmpot* file, which contains the formation enthalpies of all potential compounds formed by the reactant atoms. For example, to predict reactions between Ni and Bi_2Te_3, the *EDOPING.cmpot* file must include enthalpies for all stable Ni-Bi-Te compounds (obtainable from open databases via `edp query` or calculated manually):

The four columns represent the Bi, Te, Ni content, and formation enthalpy (eV/atom) of each compound. The 1st and 10th compounds correspond to Bi_2Te_3 and Ni, respectively. To predict their reaction energies, run:

```
$ edp react -n 1 10
# Reaction Equation: x Bi2Te3 + y Ni -> Products (5x + y = 1)
# Column 1: Total Atomic Fraction of Bi2Te3
# Column 2: Reaction Equation
# Column 3: Reaction Energy[eV/atom]
0.000   Ni -> Ni                                        0.0000
0.012   Bi2Te3 + 402 Ni -> 3 TeNi124 + 2 BiNi15         0.0089
0.013   Bi2Te3 + 392 Ni -> 2 BiNi10 + 3 TeNi124         0.0070
0.013   Bi2Te3 + 388 Ni -> 3 TeNi124 + 2 BiNi8          0.0070
0.013   Bi2Te3 + 374 Ni -> 2 BiNi + 3 TeNi124           0.0060
0.013   2 Bi2Te3 + 747 Ni -> Bi4Ni3 + 6 TeNi124         0.0061
```

The `-n` option indicates that enthalpies in *EDOPING.cmpot* are given per atom (see `edp chempot` for details). Other options include:

- `--with-hull`: Also output the reaction energy relative to the convex hull. Smaller values indicate more favorable reactions.
- `unit`: Specify the unit for the output reaction energy. Options: eV/atom (default), eV, kJ/mol, kJ.

New in version 0.5: The `react` command.

refine

Perform refinement to the unit cell. The currently supported operations are as follows:

- `-t/--transform`: Create the supercell using a given transformation matrix. The transformation matrix is typically a 3x3 integer matrix, input as a flattened row of 9 elements separated by spaces, such as `-t "2 0 0 0 2 0 0 0 2"`. Alternatively, you can use 3 elements or a single value to represent a diagonal matrix or a scaling factor for the identity matrix, respectively. The following three commands are equivalent:

```
$ edp refine -t "2 0 0 0 2 0 0 0 2"
$ edp refine -t "2 2 2"
$ edp refine -t 2
```

Additionally, if no value is provided, the software will attempt to read the transformation matrix from the `TRANSMAT.in` file (compatible with the format used by vaspkit) in the current directory:

```
$ edp refine -t
```

Important: The eDoping program internally uses **row-wise** matrix representation for basis vectors, aligning with *VASP* , *Quantum Espresso* , and *vaspkit* . This convention differs from the **column-wise** matrix representation used by *VESTA* and *phonopy*, requiring transpose operations for matrix conversions between these formats.

- `-s/--strain`: Apply a strain tensor to the unit cell. See the `-t/--transform` option for the tensor format. Example:

```
$ edp refine -s "1.05 1 1"
```

This applies a +5% strain along the x-axis only.

- `-c/--cubicize`: Find an almost cubic supercell near the given atom count. For example:

```
$ edp refine -c 100
```

This will create a (almost) cubic supercell with a total number of atoms close to 100.

- `-d/--displace`: Displace atoms within the unit cell. This typically requires four parameters: the target atom index and the displacement values along the Cartesian x, y, and z directions (in Angstrom, **NOT** fractional coordinates). For example:

  ```
  $ edp refine -d Sb3 0 0 0.1
  ```

  This will move the *3rd* Sb atom by 0.1 Angstrom along the z-axis.

- `-r/--sort`: Sort atoms in the output cell by fractional coordinates. Works well with options like `-c` or `-d`.

New in version 0.4: The `refine` command.

New in version 0.5: The `-s/--strain` and `-r/--sort` options.

`replace`

Replace atoms in the POSCAR file. Two positional arguments are typically needed for the old and new atoms. Specifically, setting the new atom to `Vac` creates a vacancy by deleting the target. Conversely, setting the old atom to `Vac` inserts an atom to create an interstitial defect, which requires specifying its coordinates via the `-p/--position` option. Beyond single-atom substitution, you can replace an element with a combination of elements, optionally shuffling their arrangement.

Example 1: Replace Fe2 in the POSCAR with Co, saving the new structure as POSCAR-sub:

```
$ edp replace -o POSCAR-sub Fe2 Co
```

You can use *`edp diff`* to check the changes in the POSCAR file:

```
$ edp diff POSCAR POSCAR-int
```

Example 2: Create a vacancy at the Fe2 site in the POSCAR, saving the new structure as POSCAR-vac:

```
$ edp replace -o POSCAR-vac Fe2 Vac
```

Example 3: Insert a Co interstitial at (0.125, 0.125, 0.125) in the POSCAR, saving the new structure as POSCAR-int:

```
$ edp replace -o POSCAR-int Vac Co -p 0.125 0.125 0.125
```

Example 4: Replace all Nb atoms (32 in total) with 16 V and 16 Ta, saving the new structure as POSCAR-split (the first 16 Nb become V, and the rest become Ta):

```
$ edp replace -o POSCAR-split -m Nb V16Ta16
```

Example 5: Randomly replace Nb atoms (32 in total) with 16 V and 16 Ta, saving the new structure as POSCAR-random:

```
$ edp replace -o POSCAR-random -s Nb V16Ta16
```

Hint: When performing multi-atom substitutions via the `-m` or `-s` options, the total number of provided atoms must exactly match the original count to avoid an error.

New in version 0.3: Support the `Vac` keyword for creating vacancy/interstitial defects.

New in version 0.5: The `-m` and `-s` options for element-level multi-atom substitution.

`scfermi`

Calculate the self-consistent Fermi level based on defect formation energies and the charge neutrality condition.

The total charge provided by all defects as a function of the Fermi level E_F is:

$$Q_1 = \sum_{q,D} q \cdot g_D^q \cdot \exp\left[-\Delta H_D^q(E_F)/k_B T\right]$$

Meanwhile, the net electron count varies with E_F as:

$$Q_2 = n - p = \int_{\text{CBM}}^{+\infty} f \cdot g(E) dE - \int_{-\infty}^{\text{VBM}} (1-f) \cdot g(E) dE$$

Here, g_D^q is the degeneracy factor of defect D in charge state q, $g(E)$ is the density of states (DOS), and f is the Fermi-Dirac distribution:

$$f = \frac{1}{1 + \exp\left[(E - E_F)/(k_B T)\right]}$$

Solving $Q_1 = Q_2$ yields a specific E_F, which is the system's self-consistent Fermi level.

To evaluate Q_1, we need q, ΔH_D^q, and g_D^q, which are standardly stored in *EDOPING.qform*. To evaluate Q_2, we need the DOS $g(E)$, typically obtained from first-principles calculations. By default, the program reads this from the VASP DOSCAR (generic text files are also supported: 1st column as energy, 2nd as DOS).

With these files, calculate the self-consistent Fermi level like this:

```
$ edp scfermi 145.91 EDOPING.qform
```

Here, 145.91 is the cell volume in Å^3. The program automatically reads the DOS from the local DOSCAR, and defect data from *EDOPING.qform*, to compute both the self-consistent Fermi level and carrier concentration at 1000 K.

Additionally, the following options are available:

- `-t`: Specify the temperature in Kelvin (default: 1000).
- `--dos`: Path to the DOS file (default: DOSCAR).
- `--vbm`: Manually set the VBM energy (default: read from DOSCAR).
- `--use-idos`: Use the integrated DOS (instead of regular DOS) to calculate Q_2. It can be mathematically shown that:

$$Q_2 = \int_{-\infty}^{+\infty} f \cdot g(E) dE - \int_{-\infty}^{\text{VBM}} g(E) dE = \int_{-\infty}^{+\infty} \left(-\frac{\partial f}{\partial E}\right) \cdot G(E) dE - N_{\text{VEC}}$$

In some cases, this provides numerically more stable results.

6 Input/Output Files

6.1 EDOPING.in

This is the input file for the `edp cal` command, used to specify configurations for implementing defect calculations. It is recommended to use uppercase letters for keywords (case insensitivity is still experimental). Lines starting with # are comments and will be ignored by the program. Content after # within a line will also be ignored.

DPERFECT

Directory path for the self-consistent calculation of the perfect substrate supercell.

DDEFECT

Top-level directory path for self-consistent calculations of defect supercells, containing subdirectories for different charge conditions.

CMPOT

The chemical potential of added or removed atoms μ_i $(= \mu_i^{\ominus} + \Delta\mu_i)$, separated by spaces, is a sequence containing an even number of values, alternating to represent the chemical potential of removed and added atoms. For example, for Nb substituted by Ta (i.e., removing Nb atom and adding Ta atom), it is set to:

```
CMPOT = mu_Nb mu_Ta
```

Specifically, for interstitials and vacancies, it can be assumed that a zero-chemical-potential atom is simultaneously removed or added. For example, for Nb vacancy defects:

```
CMPOT = mu_Nb 0
```

For Nb interstitial defects:

```
CMPOT = 0 mu_Nb
```

VALENCE

The charge values of defect atoms, separated by spaces. Note that electrons themselves are negatively charged, which means in VASP's INCAR file, increasing the NELECT value corresponds to a more negative charge value.

DDNAME

The subdirectory naming convention for self-consistent calculations of each charge state (*VALENCE*) defaults to `auto`. This auto-generation combines *PREFIX* with *VALENCE* while preserving the +/- sign prefix. For instance, given *VALENCE = -1 0 1* and *PREFIX = charge_*, setting *DDNAME = auto* produces subdirectories equivalent to specifying *DDNAME = charge_-1 charge_0 charge_+1*. Alternatively, explicit subdirectory names can be defined with space separation (note: spaces within names are currently unsupported).

DREFER

Specify the directory containing the defective supercell, primarily used for comparing with the perfect crystal to confirm the location of the defect. The default value is `auto`, where the program automatically reads the defective supercell structure from the neutral defect directory. If provided, the program will read the POSCAR file of the defective supercell under the *DDNAME* / *DREFER* directories.

New in version 0.3: The *DREFER* keyword.

PREFIX

The prefix for subdirectories of self-consistent calculations with different charge values, default is `charge_`.

EVBM

Valence band maximum energy. Default is `inf`, and the program automatically reads it from the EIGENVAL file in the *DDEFECT* directory.

ECBM

Conduction band minimum energy. Default is `inf`, and the program automatically reads it from the EIGENVAL file in the *DDEFECT* directory.

PENERGY

Total energy of the perfect supercell. Default is `inf`, and the program automatically reads it from the OUTCAR file in the *DPERFECT* directory.

PVOLUME

Volume of the perfect supercell. Default is `inf`, and the program automatically reads it from the OUTCAR file in the *DPERFECT* directory.

EWALD

Madelung constant. Default is `0`, indicating that the image charge correction term is disabled.

EPSILON

Dielectric constant. Default is `inf`, indicating that the image charge correction term is disabled.

ICCOEF

The image charge correction term can be expressed as $E_{\mathrm{IC}} = C_{\mathrm{IC}} \cdot q^2$, where the coefficient C_{IC} is specified via ICCOEF. The default value `inf` indicates automatic calculation based on *EPSILON* and *EWALD*, given by:

$$C_{\mathrm{IC}} = \left[1 - \frac{1}{3}\left(1 - \frac{1}{\varepsilon}\right)\right]\frac{E_{wald}}{2\varepsilon}$$

PADIFF

The potential alignment correction term can be formulated as $E_{\mathrm{PA}} = q \cdot \Delta V$, where the potential difference ΔV is specified through PADIFF (a list with length matching *VALENCE*). The default `[inf, ...]` triggers automatic extraction of the electrostatic potential difference at the farthest point from the defect in OUTCAR files.

BFTYPE

The type of band filling correction mechanism. Default is `0`, indicating that the band filling correction term is disabled. `1` means correcting only the conduction band, `-1` means correcting only the valence band, and `2` means correcting both the conduction and valence bands.

EMIN

Lower bound of the Fermi level (relative to *EVBM*), default is `-1`.

EMAX

Upper bound of the Fermi level (relative to *EVBM*), default is `2`.

NPTS

Number of sampling points for the Fermi level, default is `1001`.

6.2 EDOPING.log

Execution log for the *edp cal* command, same as the screen output.

6.3 EDOPING.dat

The *edp cal* command result file contains the formation energies of defects with different charge values. It has `Nq + 3` columns separated by spaces. The first column is the Fermi level (relative to *EVBM*), the second column is the defect formation energy, the third column is the corresponding charge value, and the subsequent columns are the defect formation energies for different charge values. The first row is a comment row containing column names, and the last two values represent the supercell volume and degeneracy factor. An example is as follows:

```
# Ef, Eformation, q , q_-3, q_-2, q_-1, q_+0, q_+1, q_+2, q_+3;    1689.3500    1
-1.0000 -1.6816 3.0000 5.8115 4.3378 2.8985 1.4866 0.0987 -0.8285 -1.6816
-0.9990 -1.6786 3.0000 5.8085 4.3358 2.8975 1.4866 0.0997 -0.8265 -1.6786
-0.9980 -1.6756 3.0000 5.8055 4.3338 2.8965 1.4866 0.1007 -0.8245 -1.6756
-0.9970 -1.6726 3.0000 5.8025 4.3318 2.8955 1.4866 0.1017 -0.8225 -1.6726
```

(continues on next page)

(continued from previous page)

```
-0.9960 -1.6696 3.0000 5.7995 4.3298 2.8945 1.4866 0.1027 -0.8205 -1.6696
-0.9950 -1.6666 3.0000 5.7965 4.3278 2.8935 1.4866 0.1037 -0.8185 -1.6666
-0.9940 -1.6636 3.0000 5.7935 4.3258 2.8925 1.4866 0.1047 -0.8165 -1.6636
-0.9930 -1.6606 3.0000 5.7905 4.3238 2.8915 1.4866 0.1057 -0.8145 -1.6606
-0.9920 -1.6576 3.0000 5.7875 4.3218 2.8905 1.4866 0.1067 -0.8125 -1.6576
...
```

6.4 EDOPING.qform

The output file from the `edp cal` command, containing the zero-point formation energies for different defect charge states. The relation between the defect formation energy ΔH_D^q and the Fermi level E_F is given by:

$$\Delta H_D^q(E_F) = q \cdot E_F + H_0^q$$

Here, H_0^q is the zero-point formation energy (the energy when $E_F = 0$). eDoping typically references the Fermi level to the valence band maximum, `EVBM`. These two parameters, H_0^q and q, analytically define the relation between ΔH_D^q and E_F, making them the core data of this file. It generally contains two columns: charge state q (column 1) and H_0^q (column 2). An optional third column can specify the defect's degeneracy factor (default: 1).

```
#   q        H0
   -3      2.6960
   -2      2.2865
   -1      1.8856
   +0      1.4866
   +1      1.0858
   +2      1.1202
   +3      1.2029
```

New in version 0.5: Support for the `.qform` file format.

6.5 EDOPING.cmpot

The `edp chempot` command input file is used to specify the ratios and formation energies of compounds. Take the Mn-doped NbFeSb system as an example to explain the file format: the first line starts with # and contains the element names of the system. The second line is the ratio and formation energy of the target compound NbFeSb, followed by the elemental ratios and formation energies of all considered competing phases. For more details on the normalization of elemental ratios and formation energies, see the `edp chempot` command.

```
# Nb   Fe   Sb   Mn
 1  1  1  0  -0.350468735
 0  1  0  0  0.0
 0  0  1  0  -1.9464166666871563e-05
 0  1  2  0  -0.03650248166666733
 3  0  1  0  -0.28675903625
 1  0  2  0  -0.279502903333333
 1  0  0  2  -0.1441571941954
 ...
```

7 How to Cite

If this software and documentation have helped you in your work, please cite our article. This is very important to us. Thank you very much!

[1] J. Zhu, J. Li, Z. Ti, L. Wang, Y. Shen, L. Wei, X. Liu, X. Chen, P. Liu, J. Sui, Y. Zhang, eDoping: A high-throughput software package for evaluating point defect doping limits in semiconductor and insulator materials, *Materials Today Physics*, 55 (2025) 101754, https://doi.org/10.1016/j.mtphys.2025.101754.

[2] J. Li, J. Zhu, Z. Ti, W. Zhai, L. Wei, C. Zhang, P. Liu, Y. Zhang, Synergistic defect engineering for improving n-type NbFeSb thermoelectric performance through high-throughput computations, *Journal of Materials Chemistry A*, 10 (46) (2022) 24598-24610, https://doi.org/10.1039/d2ta07142h.

Index

Q

R

S

V