

Full length article



aflow.org: A web ecosystem of databases, software and tools

Marco Esters^a, Corey Oses^a, Simon Divilov^a, Hagen Eckert^a, Rico Friedrich^{a,b,c}, David Hicks^{a,d}, Michael J. Mehl^a, Frisco Rose^a, Andriy Smolyanyuk^{a,e}, Arrigo Calzolari^f, Xiomara Campilongo^a, Cormac Toher^{a,g}, Stefano Curtarolo^{a,*}

^a Department of Mechanical Engineering and Materials Science and Center for Autonomous Materials Design, Duke University, Durham, NC 27708, USA

^b Institute of Ion Beam Physics and Materials Research, Helmholtz-Zentrum Dresden-Rossendorf, 01328 Dresden, Germany

^c Theoretical Chemistry, Technische Universität Dresden, 01062 Dresden, Germany

^d LIFT, American Lightweight Materials Manufacturing Innovation Institute, Detroit, MI 48216, USA

^e Institute of Solid State Physics, Technische Universität Wien, A-1040 Wien, Austria

^f CNR-NANO Research Center S3, Via Campi 213/a, 41125 Modena, Italy

^g Department of Materials Science and Engineering and Department of Chemistry and Biochemistry, University of Texas at Dallas, Richardson, TX 75080, USA

ARTICLE INFO

Dataset link: <https://aflow.org>

Keywords:

Autonomous materials science
Materials genome initiative
aflow
Computational ecosystems
Online tools
Database
Ab initio

ABSTRACT

To enable materials databases supporting computational and experimental research, it is critical to develop platforms that both facilitate access to the data and provide the tools used to generate/analyze it — all while considering the diversity of users' experience levels and usage needs. The recently formulated FAIR principles (Findable, Accessible, Interoperable, and Reusable) establish a common framework to aid these efforts. This article describes aflow.org, a web ecosystem developed to provide FAIR-compliant access to the AFLOW databases. Graphical and programmatic retrieval methods are offered, ensuring accessibility for all experience levels and data needs. aflow.org goes beyond data-access by providing applications to important features of the AFLOW software [1], assisting users in their own calculations without the need to install the entire high-throughput framework. Outreach commitments to provide AFLOW tutorials and materials science education to a global and diverse audiences will also be presented.

1. Introduction

The advantage of big data has long been recognized in computational materials science, resulting in large databases of experimental and hypothetical materials. The Automatic FLOW Framework for Materials Discovery (AFLOW) is the largest database for computationally investigated materials, with over 3.5 million materials entries [2] and over 1,100 crystallographic prototypes to classify crystals and generate input structures.

The data should not just be publicly available, but also easily accessible to users with varying usage requirements, from casual single-entry access to high-throughput retrieval. To provide a common framework for such an infrastructure, the FAIR principles for data stewardship were developed, where FAIR stands for “Findable”, “Accessible”, “Interoperable”, and “Reusable” [3].

This article introduces the AFLOW web ecosystem, which consists of the AFLOW databases and web interfaces to access features of the AFLOW software. We will start with the aflow.org repository, its data structure, and how it conforms with the FAIR data principles.

Next, it will be shown how to access and retrieve that data using both graphical, low-throughput and programmatic, high-throughput methods. Additionally, the Prototype Encyclopedia – a database over 1,100 crystallographic prototypes – will be discussed along with its tools to create structures for *ab-initio* calculations.

To create the AFLOW database, a variety of tools has been developed and bundled into the open-source AFLOW software suite. These features include structure manipulation and creation [4–6], symmetry and structure analysis [7,8], and post-processing tools [9,10]. While packaging these tools into a monolithic codebase is ideal for generating standardized data, the individual tools are also useful for many applications outside high-throughput calculations. For these purposes, a single large codebase is detrimental: users may only require a subset (thus not using the majority of features), there is a steep learning curve to find and properly use the desired feature, and the lack of a graphical user interface requires that users be comfortable using the command line. To alleviate these issues, AFLOW provides web applications that interface to a large variety of AFLOW features (AFLOW Online), create convex

* Corresponding author.

E-mail address: stefano@duke.edu (S. Curtarolo).

<https://doi.org/10.1016/j.commsci.2022.111808>

Received 19 July 2022; Received in revised form 20 July 2022; Accepted 11 September 2022

Available online 18 October 2022

0927-0256/© 2022 Elsevier B.V. All rights reserved.

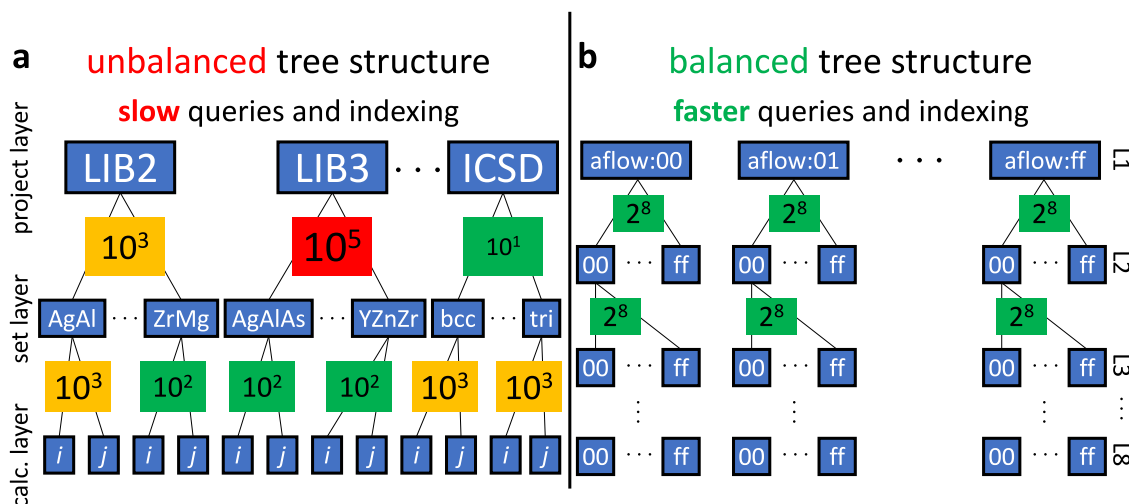


Fig. 1. AFLOW data structure. Illustration of the AFLOW data structure as (a) a human-readable, but unbalanced search tree and (b) a machine-readable, balanced search tree.

hulls for thermodynamic analysis (AFLOW-CHULL), and use AFLOW machine learning applications (AFLOW-ML). These applications will be introduced after the discussion about the AFLOW database.

Lastly, we will present the AFLOW School and seminar series. Making software usable and accessible goes beyond just making it open source. Tutorials are an invaluable way to reduce the barrier to entry for new users. The Schools and seminars represent AFLOW's commitment to providing free education about its software and materials science to a global and diverse audience.

2. The aflow.org repository

2.1. Database content and organization

The full data set generated by the high-throughput AFLOW process [2] is currently over 40 TB. Stored data includes input criteria and other calculation details (to facilitate reproducibility and fulfill the "Reusable" criterion of the FAIR principles), calculated results, and derivative output. To ensure the robust, long-term fulfillment of the FAIR criteria of accessibility and findability for such a large data store, the AFLOW repository [2] is organized in a human-navigable hierarchical directory structure, allowing specific entries to be accessed independently of the database engine. Specifically, the repository is organized into project, set and calculation layers, as illustrated in Fig. 1. At the project layer, the calculations are divided into different catalogs based on the origin and composition of the entries [11,12]. Within each catalog, entries are grouped into sets based on shared lattice type or alloy system. The entries within each set contain the results of density functional theory (DFT) calculated properties for particular structures.

The AFLOW-ICSD catalog contains the DFT-calculated properties for over 60,000 experimentally observed materials listed in the Inorganic Crystal Structure Database (ICSD) [13,14]. Internally, this catalog is organized by lattice type, and then by individual materials entry. The entries in this catalog are grouped by Bravais lattice type into 14 sets: BCC, BCT, CUB, FCC, HEX, MCL, MCLC, ORC, ORCC, ORCF, ORCI, RHL, TET, and TRI. The name of each materials entry is generated using the format `<composition>_ICSD_<ICSD number>`. To aid reproducibility, the species labels in the `composition` part of the entry name are the full VASP pseudopotential (POTCAR) names used for that calculation. Since the materials in this catalog are already known to exist, the primary interest is in accurately calculating electronic structure and thermo-mechanical properties. Therefore, calculations for this catalog are generally performed using the Hubbard U correction to the DFT exchange-correlation functional [15,16] where appropriate, using a set of standardized U values [17].

The entries in the other catalogs, such as LIB1, LIB2, and LIB3, are generated by decorating crystal structure prototypes to predict new hypothetical compounds, and contain unary, binary, and ternary materials, respectively. Within each catalog, the entries are grouped by element and exchange-correlation functional in the case of LIB1, and by alloy system in the cases of LIB2 and LIB3. LIB1 contains a total of 4k entries, while LIB2 currently has 364k entries and LIB3 has about 2.6 million. Within each alloy system, the individual materials entries are named according to the relevant crystal prototype. For example, the directory path for the composition AlCu_2Mg in the Heusler structure is `LIB3_WEB/AlCu_pvMg_pv/T0001.AB2C`, where `LIB3_WEB` is the project layer corresponding to LIB3, `AlCu_pvMg_pv` is the set layer corresponding to the Al-Cu-Mg alloy system (using the Al, Cu_pv and Mg_pv pseudopotential POTCARs), and `T0001.AB2C` is the A2BC decoration of the Heusler prototype T0001. For these catalogs, the emphasis is on the discovery of new thermodynamically stable or metastable materials, and on their use to generate the thermodynamic density of states for the prediction of the formation of disordered materials such as metallic glasses [18] or high entropy alloys [19]. Therefore, calculations in these catalogs are performed using the GGA-PBE exchange-correlation functional [20] without Hubbard U corrections [17] so as to produce consistent energy differences, enabling the calculation of accurate formation enthalpies.

Categorizing data based on the number of species, composition, and crystal prototype is an intuitive, human-readable choice. However, this data structure introduces inefficiencies that make it unsuitable for machine reading. The number of possible element combinations and crystal prototypes increases with the number of species. This results in an unbalanced search tree as illustrated in Fig. 1(a). For instance, the catalog of binary (LIB2) and ternary entries (LIB3) may have 10^3 and 10^5 entries, respectively. In the alloy layer, the number of entries may be unevenly distributed as well. This data structure is inefficient to query and index: walking and searching through LIB3 is significantly slower than through LIB2, resulting in uneven performance and preventing efficient parallelization. Using other materials properties to structure the data would present similar challenges as no property is evenly distributed across all systems.

The AFLOW Unique Identifier (AUID) system eliminates this computational inefficiency. It consists of the `aflow` namespace declaration, followed by a quasi-random 64-bit (8-byte) hash, for example:

```
aflow:d3aa24a7307877b5.
```

This identifier has the capacity for $2^{64} \approx 10^{19}$ entries and can be further extended by using different namespaces. The AUIDs are generated

by hashing the alphabetically ordered output files (OUTCAR) of the AFLOW run using a 64-bit cyclic redundancy check (CRC64) algorithm. Since the content of these files depends on parameters specific to the calculation, such as the options of the VASP calculations or the input and final structure, the resulting AUID is globally unique to the entry. Changing the cut-off energy, for example, would result in a different identifier. Using OUTCAR files as the basis for the algorithm also makes the AUID reproducible and thus persistent, fulfilling the “Findable” criterion of the FAIR data sharing principles [3].

The byte expression of the AUID offers a new way to structure data where each layer is represented by an individual byte of the AUID. This scheme is shown in Fig. 1(b). While this leads to a larger number of branches, they are smaller ($\sim 10^2$ entries) with an even distribution of AUIDs across each byte layer due to the quasi-randomness of the hash. These balanced search trees are fast to query and index and also enable efficient parallelization.

Due to this computational efficiency, AUIDs are used throughout the AFLOW software and web applications when interfacing with the AFLOW database. However, using a hash-based identifier has the major drawback that it is not human-readable as opposed to a material-based identifier such as:

ICSD_WEB/ORCC/Ag5S4Sb1_ICSD_16987.

The data structures presented in this section are thus complementary. The AFLOW database provides both methods of access to its materials entries.

2.2. Accessing and searching the database

2.2.1. The AFLOW REST-API

To facilitate accessibility and interoperability to the 40 TB of data generated by the high-throughput AFLOW process [2], the backing store is exposed via the AFLOW Data REST-API [2] in a hierarchical organization. This direct exposure of our results not only grants the end user a high degree of utility via direct access, but more importantly, guarantees data provenance that promotes reproducibility. The hierarchy of the AFLOW Data REST-API categorizes this abundance of information into meaningful high-level classifications, allowing for exploration of self-similar materials that are related by stoichiometric and/or crystallographic properties. Once a selection of materials has been determined, the full range of available properties and procedural data are retrievable.

The organizational hierarchy of both the underlying data store and the REST-API is project-dependent, as described in Section 2.1. Each project is equivalent to one of the catalogs listed in this section, and in the REST-API is denoted by the project layers ICSD_WEB, LIB1_WEB, LIB2_WEB, and LIB3_WEB. Each project layer contains multiple set layers, which correspond to specific alloy systems in the case of LIB1_WEB, LIB2_WEB, LIB3_WEB, etc. For instance: afwlib.duke.edu/AFLOWDATA/LIB2_RAW exposes the set layer for binary entries, where each set corresponds to a different binary alloy system, allowing for pairwise atomic species examination. Within each set is the entry layer, consisting of decorated structural prototypes which provide stoichiometric and structural variation for each alloy system. Each entry contains the calculated results for a particular structure and composition, organized as keyword-value pairs. The calculated values of thermodynamic, mechanical, electronic, magnetic, chemical and crystallographic properties can be directly accessed by querying a Uniform Resource Identifier (URI) of the form `<server>/<project>/<set>/<entry>/?<keyword>`, where `<server>` is `afwlib.duke.edu/AFLOWDATA`, `<project>` is the appropriate project layer, `<set>` is the alloy system, `<entry>` is the structural prototype, and `<keyword>` corresponds to the materials property of interest. For example, the formation enthalpy per atom of AlCu_2Mg in the Heusler structure can be accessed at https://afwlib.duke.edu/AFLOWDATA/LIB3_WEB/AlCu_pvMg_pv/T0001.AB2C/?enthalpy_formation_atom, where the project is LIB3_WEB, the set is the alloy system AlCu_2Mg , and the keyword is `enthalpy_formation_atom`. A full list with descriptions of the REST-API keywords is provided in the Appendix of this work, combining the original list from Ref. [11] with the additions in the appendices of Refs. [12,21].

Table 1

List of AFLUX operators, their symbols, and their scope. Inter-property operators can connect properties. Intra-property operators apply to the property alone.

Operator	Symbol	Scope
BLOCK	()	inter- & intra-property
NOT	!	intra-property
AND	,	inter- & intra-property
OR	:	inter- & intra-property
LOOSE	*	intra-property
STRING	' '	intra-property
MUTE	\$	intra-property

lib.duke.edu/AFLOWDATA/LIB3_WEB/AlCu_pvMg_pv/T0001.AB2C/?enthalpy_formation_atom, where the project is LIB3_WEB, the set is the alloy system AlCu_2Mg , and the keyword is `enthalpy_formation_atom`. A full list with descriptions of the REST-API keywords is provided in the Appendix of this work, combining the original list from Ref. [11] with the additions in the appendices of Refs. [12,21].

The information required to construct the URI for each entry in the database is available through the AURL (AFLOW Uniform Resource Locator) keyword, which is returned by default for each result in an AFLUX search. The AURL for each entry is similar in form to the first part of the URI: `<server>/<project>/<set>/<entry>`, except that the `server` part has the form `afwlib.duke.edu:AFLOWDATA`. By replacing the “:” with a “/”, the base URI for the corresponding entry can be constructed. The AURL is particularly useful when trying to retrieve data not directly accessible through AFLUX such as calculation input or output files. For example, the file with the relaxed structure for a particular entry, `CONTCAR.relax`, can be downloaded at the URI `<server>/<project>/<set>/<entry>/CONTCAR.relax`, which can be constructed from the AURL. AURLs can also be used to systematically access all of the entries in a particular set.

The ability to explore related entries predicated on a multitude of properties leads directly to novel materials discovery and use. The AFLOW Data REST-API disseminates our methods and results, without restriction, to a global research audience in order to promote scientific and engineering advancement.

2.2.2. AFLUX: The AFLOW search API

The AFLUX API provides programmatic search functionality for the AFLOW database [12]. Queries are submitted through a URI and can thus be conducted with commonly available tools such as `wget`, `curl`, Python’s `urllib` module, or a web browser. Results are returned in easily parsed formats, such as JSON. This makes AFLUX platform-independent and enables integration into computational workflows written in all commonly used languages.

AFLUX is a domain-specific language, i.e., no prior knowledge on the underlying database structure or format is needed. It requires only a very small set of operators, shown in Table 1, which further reduces barriers to using it. An AFLUX call, or a *summons*, is written in the query portion of the URL:

```
afwlib.duke.edu/API/afwlib.duke.edu/AFLOWDATA/LIB3_WEB/AlCu_pvMg_pv/T0001.AB2C/?enthalpy_formation_atom
```

The summons is composed of a *matchbook* and *directives*:

```
afwlib.duke.edu/API/afwlib.duke.edu/AFLOWDATA/LIB3_WEB/AlCu_pvMg_pv/T0001.AB2C/?enthalpy_formation_atom
```

The matchbook consists of the search criteria whereas directives guide the form of the results.

The AFLUX matchbook. To search for and filter by properties, they need to be added to the matchbook. The following summons lists the band gap (Egap of each returned entry:

```
aflow.org/API/aflux/?Egap
```

Or in short:

```
<matchbook> = Egap
```

A full list of properties can be found in the [Appendix](#). Multiple properties can be returned by using the AND (,) operator. For instance,

```
<matchbook> = Egap,spin_atom
```

returns both the band gap and the magnetic moment per atom of each material. Properties can be restricted to a specific value using the BLOCK (parentheses) operator:

```
<matchbook> = species(Cr)
```

This summons will only return entries containing chromium (Cr). These restrictions also work for numerical values. The matchbook

```
<matchbook> = Egap(1)
```

returns all entries with a band gap of exactly 1 eV.

Set logic. AFLUX supports basic logical operators: NOT (!), AND (,), and OR (:). NOT can be used to exclude values from the results. For example, to show entries without Cr, the matchbook should be:

```
<matchbook> = species(!Cr)
```

The AND operator, previously used to retrieve multiple properties, can also be employed as an intra-property operator. The following summons will return all entries containing Cr and O:

```
<matchbook> = species(Cr,O)
```

To get all entries that include Cr or Mn, but not necessarily both, the OR operator can be used:

```
<matchbook> = species(Cr:Mn)
```

When combined with the BLOCK operator, these logical queries can become arbitrarily complex. The following matchbook will return all materials with either Cr or Mn (or both) and at least one chalcogen:

```
<matchbook> = species((Cr:Mn),(O:S:Se:Te))
```

Value ranges and substrings. Exact matches may not always be desired, especially for numeric quantities like the band gap (Egap). The LOOSE operator (*) can be used to build inequalities. It is a positional operator where $N *$ returns values $\geq N$ and $* N$ returns values $\leq N$, for example:

```
<matchbook> = Egap(0.6*)
```

returns entries with band gaps of 0.6 eV and higher. This can be combined with the AND operator to form a tolerance range:

```
<matchbook> = Egap(0.6*,*1.5)
```

This returns entries with $0.6\text{ eV} \leq \text{Egap} \leq 1.5\text{ eV}$. It is also possible to select results that satisfy multiple ranges by using the OR operator:

```
<matchbook> = Egap((0.5*,*0.7):(1.4*,*1.6))
```

To get a strictly greater than 1.6 eV match condition, the NOT operator can be combined with the LOOSE operator:

```
<matchbook> = Egap(!*1.6)
```

Using the LOOSE operator without any value will discard null results, which are returned by default. The matchbook

```
<matchbook> = Egap(*)
```

will only return entries for which Egap has been calculated.

The LOOSE operator can also be used for substring searches. The position of the LOOSE operator determines where the substring ought to be located. The matchbooks

```
<matchbook> = Pearson_symbol_relax('c*')
<matchbook> = Pearson_symbol_relax(*'F*')
<matchbook> = Pearson_symbol_relax(*'8')
```

find entries with substrings at the beginning, in the middle, and at the end, respectively. Single quotes (STRING operator) are required for substring searches. They can also be used to avoid collisions with AFLUX operators as in

```
<matchbook> = auid('aflow:1dcf91da07337d8d
↪')
```

Without STRING, AFLUX would search for `auid='aflow' OR auid='1dcf91da07337d8d'`, resulting in an empty response.

Removing redundancies. Properties in a search result may be ubiquitous and therefore of little use. For example, in the matchbook

```
<matchbook> = Egap(0.6*,*1.5),nspecies(3)
```

each returned material will have three species, bloating the response with redundant information. AFLUX can suppress the output of a match criterion via the MUTE (\$) operator:

```
<matchbook> = Egap(0.6*,*1.5),$nspecies(3)
```

This allows a property to be matched, but not displayed.

Aliases. Complex AFLUX summons can become cumbersome to type. For example, to search for materials containing a transition metal and a chalcogen, the matchbook needs to contain:

```
<matchbook> = species((Sc:Ti:V:...:Hg),(O:S
↪:Se:Te))
```

where ... contains 25 more species. AFLUX contains pre-defined aliases to simplify this summons to:

```
<matchbook> = species(TransitionMetals,
↪ Chalcogens)
```

A list of all supported aliases can be found at aflow.org/API/aflux/?help/aliases.

AFLUX directives. Directives determine the size and format of an AFLUX response and provide documentation. The currently supported directives are as follows:

- format,
- paging,
- help.

The format directive. By default, AFLUX returns search results as minified JSON. The format directive can be used to change the output format:

```
<directive> = format(format)
```

Supported formats are *json*, *aflow*, and *html*. Using json formats the response as a human-readable JSON file. The *aflow* format returns the data in the *afloplib.out* format, i.e., as key-value pairs separated by |. Finally, *html* displays the results as an HTML table.

The paging directive: response size and sorting. The size of the AFLUX repository can lead to very large responses when the search criteria are broad. This can overwhelm systems communicating with AFLUX, especially when a web browser is used. As a result, AFLUX limits the number of entries it returns at once. The paging directive controls this response sizing and enumeration:

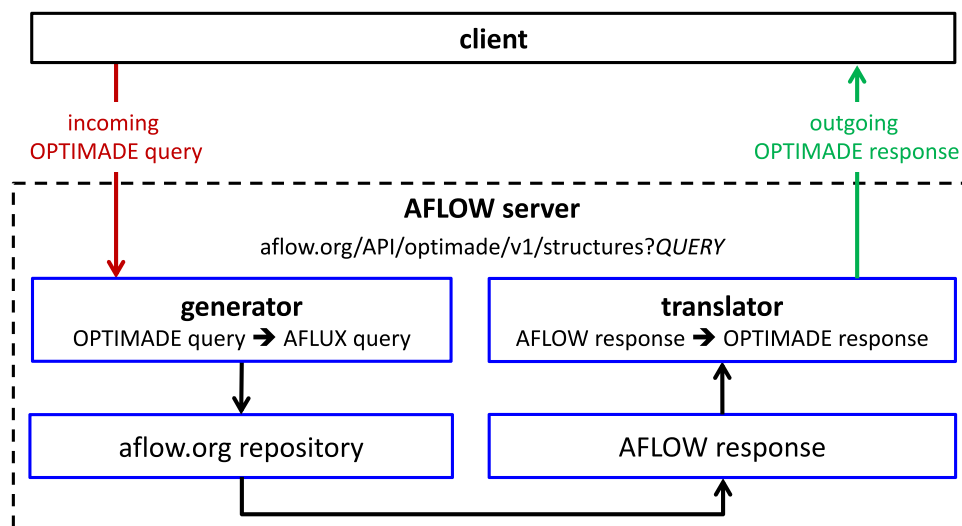


Fig. 2. Infrastructure for responding to OPTIMADE queries. Illustration of the workflow for processing an OPTIMADE query. The query is translated to AFLUX (via the generator) for retrieving the data and the AFLOW response is converted to OPTIMADE (via the translator) before returning to the client.

```
<directive> = paging(J,K)
```

This returns the J^{th} page with K results per page. For example, the directive:

```
<directive> = paging(3,100)
```

can be used to retrieve the results 201–300 (3rd page with 100 entries per page). The default values are $J = 1$ and $K = 64$. If J is larger than the number of available pages, the response will be an empty array or an empty string for json and aflow format, respectively. This can be used as a stop condition in a while-loop for high-throughput retrieval. Setting J or K to zero results in special behavior: $J = 0$ returns all results at once; $K = 0$ returns the number of results, regardless of the value of J .

Paging can also be used to sort results. Positive and negative values for J sort results in ascending and descending order, respectively. This also works for $J = 0$ to return all results in descending order:

```
<directive> = paging(-0)
```

When the output format is JSON, using the `paging` directive returns the results as a dictionary. The MUTE operator can be used to format the data as an array instead:

```
<directive> = $paging(3,100)
```

This is the default behavior when `paging` is not added to the directive. **The help directive: AFLUX documentation.** Usage information on AFLUX can be obtained using `help`. The directive:

```
<directive> = help(general)
```

produces a general help file that serves as the documentation for AFLUX. The same output can be found when no argument to `help` is given or when both the matchbook and the directive are empty. Information on available properties (see [Appendix](#)) can be:

```
<directive> = help(properties)
```

Replacing “properties” with a list of keywords will limit the output to the selected properties. All available help files are listed in the following directive:

```
<directive> = help(help)
```

2.2.3. AFLOW-OPTIMADE integration

To facilitate interoperability with other databases in accordance with the FAIR criteria, as well as to broaden accessibility and findability, the OPTIMADE common API [22] has been implemented within the AFLOW database. OPTIMADE is a common API for searching materials data across a range of repositories, including AFLOW, NOMAD [23], Materials Project [24], Materials Cloud [25], and OQMD [26]. The full specification is available at <https://github.com/Materials-Consortia/OPTIMADE/blob/master/optimade.rst>. Similar to AFLUX, searches are encoded within the query part of URL, and can include logic operations such as AND, OR and NOT; searches for different element combinations using the operators HAS ALL, HAS ANY and HAS ONLY; substring comparisons using CONTAINS, STARTS WITH and ENDS WITH; and searches for property values within specific ranges using $>$, $<$ and $=$. Data is returned in as a JSON object, in a standardized format to maximize interoperability.

The OPTIMADE API for AFLOW can be accessed at aflow.org/API/optimade. Available queryable endpoints include ones for `structures` and `calculations`. The server takes the OPTIMADE query, converts it to an AFLUX query for the AFLOW database, then reads the AFLUX response and translates it into the standardized format, as illustrated in Fig. 2.

AFLOW keywords can be included in Optimade queries of the AFLOW database by appending the prefix “_aflow_” to the front of the keyword. For example, lead-free halide cubic perovskites with a band gap exceeding 3 eV can be searched for in the AFLOW database using `optimade` with the following query: [aflow.org/API/optimade/v1/structures?filter=elements HAS ANY “F”, “Cl”, “Br”, “I” AND NOT elements HAS “Pb” AND _aflow_aflow_prototype_label_relax CONTAINS AB3C_cP5_221_a_c_b](http://aflow.org/API/optimade/v1/structures?filter=elements%20HAS%20ANY%20%22F%22%2C%22Cl%22%2C%22Br%22%2C%22I%22%20AND%20NOT%20elements%20HAS%20%22Pb%22%20AND%20_aflow_aflow_prototype_label_relax%20CONTAINS%20AB3C_cP5_221_a_c_b). Full lists of the keywords available to use with OPTIMADE to query AFLOW are available at the `info` endpoints aflow.org/API/optimade/v1/info/structures and aflow.org/API/optimade/v1/info/calculations.

2.2.4. The AFLOW search GUI

Programmatic access to data is essential for integrating the AFLOW data into workflows, but is not suitable for casual searches or for users unfamiliar with coding or the AFLUX language. To fulfill these usage requirements, `aflow.org` provides a search application, which can be accessed at aflow.org/search/. It consists of four components: the search bar, an element filter, a properties filter, and the search results. The search bar, at the top of the page (see [Figs. 3\(a\)](#) and [\(b\)](#)), contains a text field for the elements of interest, the button to

conduct a search, and the “Display” slider. The text field also contains buttons that can limit the search to the ICSD catalog or extend it to the entire AFLOW database. The “Display” slider toggles between having the element and property filters on separate slides and to have them both displayed on a single page, depending on the preferences of the user.

The element filter is shown in Fig. 3(a) and takes shape of the periodic table. Elements can be added to the query by clicking on its chemical symbol or by directly typing them into the search bar. Groups of elements can also be selected using the element group selector. Hovering over each button reveals which elements belong to each group, as demonstrated for the chalcogens in the figure. Next to the element group selectors is a set of logical operators. Elements or element groups can be excluded using the NOT button or combined using AND, OR, or XOR (exclusive or). AND is automatically added when clicking on two elements successively. Parentheses can be used to make these combinations arbitrarily complex. Using the AFLUX [12] syntax, these logic statements can also be directly entered into the search bar. However, using the buttons can prevent syntax errors because they only accept valid operators, e.g., by disallowing using OR immediately after AND. In either case, the syntax is validated before submitting a search. Lastly, the number of species can be set with “# of species” selector. Both ranges or an exact number are possible.

The property selectors can be displayed by clicking on the arrow next to the “Property Filters” label or by flipping the “Display” slider and is shown in Fig. 3. Available properties are organized into different categories, such as electronic, mechanical, or symmetry properties of the relaxed structure. A property can be added to the search query by clicking the “Add” button. The text field above the categories and descriptions serves as a search field to provide a convenient way to add a property to the list of filters. Selections can be removed via the “x” button at the right-hand side of its card. In the example shown in Fig. 3(b), “Bravais lattice” and “VRH bulk modulus (AEL)” were added to the search query (“space group” and “Pearson symbols” are added by default). The figure also shows that the values for these two properties were set to restricted to retrieve only face-centered cubic compounds with a VRH bulk modulus of 0 – 100 GPa. Additionally, the “Bravais lattice” property has “Display column” unchecked, which will result in it not being displayed in the results table. Since the Bravais lattice will be the same value for every material, this will help focus the search output.

Fig. 3(c) shows the results of this query. The driver of the search is AFLUX [12] — its summons is shown in the field below the “Reset Search” button. The results table is divided into pages and always contains the ENTRY and DATA columns. ENTRY consists of the empirical formula and the AUID of the entry. The link points to the material’s entry page, which will be discussed later. The DATA column contains links to the REST-API page of the entry and its `afloplib.out` and `alowlib.json` files. The remaining columns contain the requested properties for which “Display column” is switched on — column visibility can be toggled after the search as well. The tables can be sorted by each visible column except DATA and array properties.

The entry page contains the calculated properties and downloadable files for the material [27–31]. As for the search page, properties are organized into categories to facilitate navigation. Two example sections — thermal properties and mechanical properties — are shown in Fig. 3(d). Each property has its own card containing its value and unit. The property cards also contain links to their REST-API entries [11]. The “Downloadable Files” section at the bottom of the page is structured the same way and contains various input and output files such as VASP inputs, Bader charges, and outputs of the AFLOW modules used for the entry. The input files and calculation parameter sections fulfill the FAIR “Reusable” criterion and help reproduce the results calculated by AFLOW.

There are also two interactive applets on this page: one for a crystal structure visualization and another one for electronic structures.

Fig. 3(e) shows the structure viewer tool. It is available for both the conventional and primitive cell of the material and is based on JSmol [32,33]. Apart from basic structure viewing options, it offers visualizations for the symmetry elements of the structure calculated by AFLOW-SYM [7]. The “Symmetry” tab shows all available operations and displays their types, symbols in the Schoenflies and Hermann–Mauguin notation, whether they contain a translation, their symmetry plane or axis in fractional coordinates, and their rotation angles. The axes and planes, as well as the inversion points (for inversions and rotoinversions) can be displayed, as shown in Fig. 3(e). The “Apply” buttons start animations that visualize the selected symmetry operation — a supercell of the structure can be displayed as a visual aid. Where available, the applet can also display Bader isosurfaces for each atom in the primitive cell for different cut-off values.

The other interactive applet shows the electronic structure of the material (see Fig. 3(f)). Bands of both spin channels can be traced with a mouse, showing the energy of the band at every step. Zooming in and dragging along the k -path axis is possible as well. The lines of the density of states (DOS) can also be traced, but the value of the DOS is shown instead of the energy. By default, the total DOS and the projections along the orbitals are shown — the projected DOS of each symmetrically-inequivalent atom is accessible using the “Switch DOS” drop-down menu. Static high-quality images of both the total and projected DOS are available through the button underneath the applet.

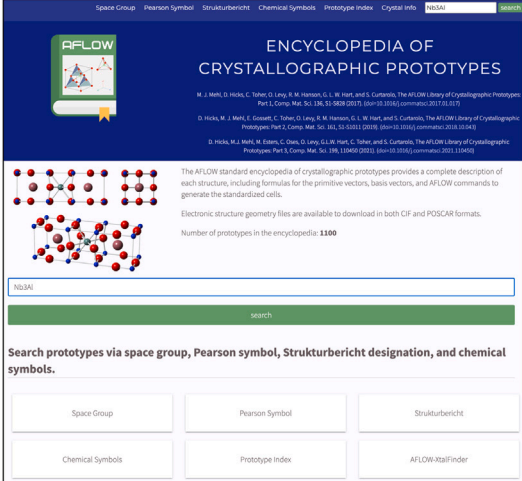
3. The prototype encyclopedia

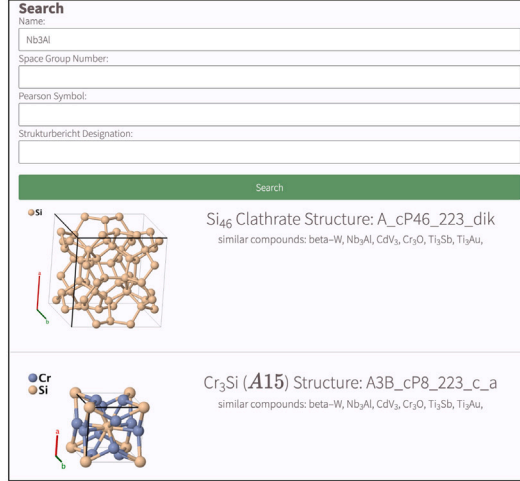
Beginning researchers in computational materials science often find it difficult to determine exactly which structure is to be studied. Take as an example a study of the behavior of the “high-temperature” (as classified in 1969 [34]) superconductor Nb_3Al . Performing computations for this material requires knowledge of its crystal structure, including both the unit cell dimensions and the position of each atom in that cell, but novices in the field may have difficulty finding this information in the literature. The original Nb_3Al article refers to the β -W structure and hints that the structure is similar to that of V_3Sn and Nb_3Sn . Searching further, our hypothetical researcher might find references to Cr_3Si as the “prototype” of this structure, or it might simply be referred to as a mysterious “A15” superconductor. These sources assume that the structure is so well known that it is not necessary to specify the atomic positions or the space group of the structure, and might not even mention that the crystal is cubic. This problem persists across the literature, obfuscating the relevant crystal structure information needed to perform computational studies.

To address this challenge, the AFLOW Encyclopedia of Crystallographic Prototypes (or Prototype Encyclopedia for brevity) serves as an extensive catalog of crystalline prototypes hosted at aflow.org/prototype-encyclopedia/. Based on the “Crystal Lattice Structures” website started by researchers at the U.S. Naval Research Laboratory’s (NRL) Complex Systems Theory Branch (now the Center for Computational Materials Science), the Curtarolo Group at Duke University has continued and extended this effort via the Prototype Encyclopedia. A web page for each structure can be found online, detailing its space group, lattice constants, and atomic (Wyckoff) positions. To assist new researchers and students, each page expanded this basic information by explicitly describing the position of each atom in the structure’s primitive cell given the Wyckoff positions of its site. It also lists compounds with the same structure as the prototype, and included many of the common *Strukturbericht* symbols [35]. A simple search of the website would then reveal the structure of Nb_3Al [36]. In addition to online search capabilities, users can generate geometry files based on these structures with underlying AFLOW prototyping functionality.

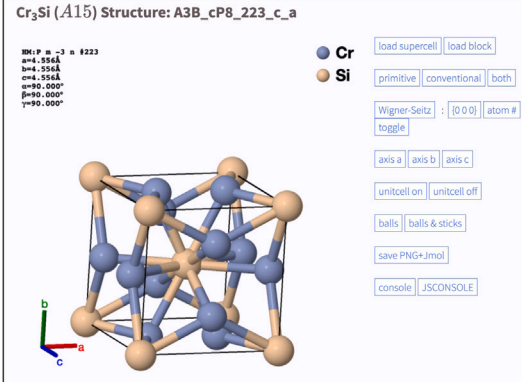
There are currently three parts to the AFLOW Prototype Encyclopedia. Each release resulted in a major update to the website — adding a page for each new prototype — and is accompanied with an article in *Computational Materials Science*. A summary of each part is as follows:

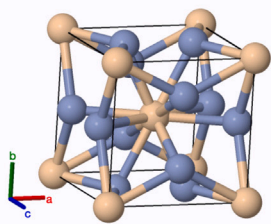
aflow.org/prototype-encyclopedia

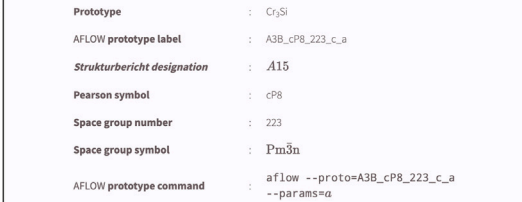
a 

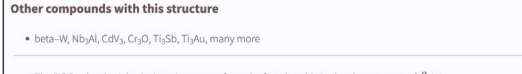
b 

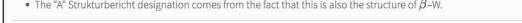
prototype entry page

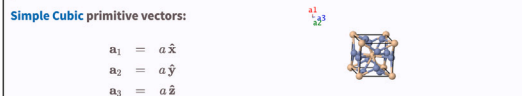
c 

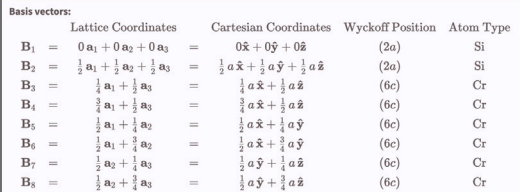
d 

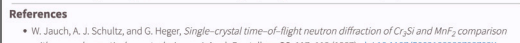
e 


f 

g 

h 

i 

j 

k 

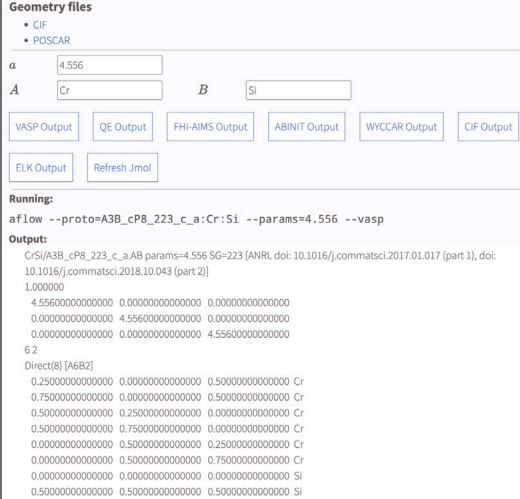
l 

Fig. 4. The AFLOW Prototype Encyclopedia web page. (a) The main web page with a different methods to find specific prototypes, including a search bar. (b) An example response page when searching for “Nb₃Al”. The prototype page for Cr₃Al, the “A15” prototype of Nb₃Al described in the text can be found at aflow.org/prototype-encyclopedia/A3B_cP8_223_c_a.html. Each page contains the following: (c) title, (d) Jmol viewer, (e) prototype designations, (f) a list of compounds/elements exhibiting this structure, (g) notable comments, (h) primitive vectors, (i) basis vectors, (j) references, (k) additional references where the structure was found, and (l) an automatic prototype generator.

- The *Strukturbericht* designation, if any (Fig. 4(e)).
- The Pearson symbol [43], showing the crystal class and the number of atoms in the unit cell (Fig. 4(e)).
- The space group symbol (Fig. 4(e)).
- The format of the AFLOW prototype command which was used to generate the structure (Fig. 4(e)).
- A list of known compounds which have this structure, if any. This list is updated when similar compounds are found in the literature (Fig. 4(f)).

- Any comments about the structure (Fig. 4(g)). This may include links to other structures associated with the prototype or related to the current crystal structure.
- The vectors defining the primitive unit cell of the structure (Fig. 4(h)).
- The basis vectors: the position of each atom in the cell is described in terms of the primitive vectors and the Cartesian coordinates, as a function of the lattice parameters and the Wyckoff positions (Fig. 4(i)). The atoms are grouped by Wyckoff position and atomic species.

- The reference to the structure in the literature, if any, including a link to the publication, if available (Fig. 4(j)).
- If the above reference was found in another publication or website, that reference and its associated link are given (Fig. 4(k)).

Finally, each page contains a link to the AFLOW prototype generator for this structure (Fig. 4(l)). The user can change the chemistry of the prototype, and enter new lattice constants and Wyckoff coordinates appropriate to that structure. Once entered, this information can be used to update the page showing the new structure. Information about the structure can be obtained in a variety of formats, including popular electronic structure programs and the standard Crystallographic Information File (CIF).

Searching structures by crystallographic descriptors. The Prototype Encyclopedia website includes resources which serve as a useful reference for beginning students in crystallography, adapted from published articles [37–39].

All periodic systems in three-dimensional space belong to one of seven crystal systems, each with the same holohedry (rotational symmetry). Different possible translational symmetries in each class lead to the fourteen Bravais lattices. This information is summarized under the “Crystal Info” tab on the home page. This tab provides a brief description of each crystal class, the Bravais lattices included in the class, and the space groups associated with each Bravais lattice. Although each lattice can be described by a large (infinite) variety of primitive vectors, AFLOW makes a standard choice for each lattice [4], which we show here. So, for example, if a structure is known to have a body-centered orthorhombic lattice, clicking the “Orthorhombic” tag under “Crystal Info” and scrolling down to “Body-Centered Orthorhombic” will show:

- the standard primitive vectors of the lattice,
- an image of the primitive unit cell and the associated conventional unit cell,
- and list the nine space groups which are associated with this lattice.

Each crystal structure is associated with a Pearson Symbol, which summarizes the crystal system, the Bravais lattice, and the number of atoms in the conventional unit cell of the structure. The “Pearson Symbol” tab on the home page is an index of all the crystal structures by symbol. If the body-centered orthorhombic system described above has eight atoms in the conventional cell, then the user finds and selects the “Orthorhombic Structures” link on the Pearson Symbol page. The resulting page lists all of the orthorhombic structures currently in the Prototype Encyclopedia. Searching for “oI8” (orthorhombic class “o”, body-centered lattice “I”, and eight atoms for unit cell) leads to the three crystal structures with this Pearson symbol found in the current database.

The “Space Group” tab indexes the structure by space group. To continue with the above example, the oI8 structured ferroelectric NaNO_2 is in space group $Imm2$ #44. This structure can be found under the Orthorhombic Space Group link of the Space Group page, along with the other four structures currently in this space group that are currently in the database.

Similarly, the *Strukturbericht* tab lists all of the structures which have been given a *Strukturbericht* designation. The NaNO_2 structure described above is labeled $F5_5$, and can be found by clicking on the “*Strukturbericht* Type F” link on this web page and scrolling down to this label.

The “Prototype Index” tab lists every structure in the Encyclopedia. This index can be sorted by prototype formula, the number of atomic species, the number of atoms in the primitive cell, the Pearson Symbol, *Strukturbericht* designation (if any), AFLOW prototype label, space group label, space group number, and structure name (if available). The default formula chemical ordering on this page is alphabetical by element, so we can find our sample structure by ordering by prototype structure and scrolling down to NNaO_2 .

Generic search functionality. The prototypes within the Encyclopedia can be searched online via different criteria. In the navigation bar, the users can type the chemical formula, mineral, or common name to search for specific materials/prototypes. This function also searches through the “similar elements/compounds” to consider atom decorations beyond the “prototype material” for a given prototype. Matching prototypes are returned on a separate page, with hyperlinks to navigate the corresponding entries. By selecting “Search” in either the navigation bar or the main web page, users will be redirected to a separate webpage, enabling searches via “Name” (same functionality as with the navigation bar), “Space Group”, “Pearson Symbol”, and/or “*Strukturbericht* Designation”. To further narrow down results, users can type in each search field to perform compound queries.

Comparing to the Encyclopedia with AFLOW-XtalFinder. In the “AFLOW-XtalFinder” tab, users can upload structures to determine if they match to one of the prototypes in the Encyclopedia. Leveraging routines in the AFLOW-XtalFinder module [8], this functionality automatically determines i. the input structure’s ideal prototype designation (i.e., label and degrees of freedom), ii. structurally similar prototypes (isoconfigurational) in the Encyclopedia, and iii. symmetrically similar prototypes (isopointal) in the Encyclopedia. Uploaded geometries can be in a variety of common file formats (auto-detected), namely, VASP [44–46], QUANTUM ESPRESSO [47], FHI-AIMS [48], ABINIT [49], WYCCAR [7], CIF, and ELK [50]. If an uploaded structure does not match to an existing AFLOW prototype, a message will appear on the web page. Users are encouraged to fill out the displayed submission form providing references/citations to the structure, any notable comments about the structure, and user information (optional). Submission of new structures will be considered for possible inclusion into future iterations of the Prototype Encyclopedia.

Future outlook. The Prototype Encyclopedia continues to be expanded, with updates to the database posted periodically. Over 600 new structures are already prepared for future release. We invite users to submit new structures for inclusion in the database. See *The AFLOW Prototype Encyclopedia: Part 3*, [39] and AFLOW-XtalFinder at aflow.org/prototype-encyclopedia/xtal-finder.html for more information.

4. Web applications

4.1. AFLOW Online

Autonomous computational frameworks such as AFLOW are powerful tools with a rich diversity of features. However, the size of the codebase presents a large barrier to entry to users who wish to use only specific functionality, only use these functions occasionally, or are not experienced with using a command line. A graphical user interface, ideally available online, is more suitable for these use cases. AFLOW Online is such a platform and implements many commonly-used features of the AFLOW software. It can be accessed at aflow.org/aflow-online/.

The interface consists of a structure input section, a section containing select AFLOW features, and an output section. The structure input section (Fig. 5(a)) contains the structure that is used to perform the AFLOW calculation. It supports all formats that the AFLOW software supports, which includes VASP [44–46], QUANTUM ESPRESSO [47], ABINIT [49], ELK [50], FHI-AIMS [48], and Crystallographic Information Files (CIF). In a separate tab, a PARTCAR file for the features of AFLOW’s Partial Occupation module (AFLOW-POCC) can be entered [6]. Magnetic moments can be added to the structure by entering the spins as a comma-separated list into the “Magnetic moments” field after activating the “Magnetic Structure” button. There are two other settings in this section: the “Output Type” drop-down menu can toggle between free text and JSON output (where available), and “Symmetry Tol”. sets the symmetry tolerance to either “tight” or “loose”, as defined in the AFLOW-SYM module [7].

a

AFLOW Online

Submit Reset

Input

Input Structure Input PARTCAR

```

EXAMPLE POSCAR
1.000000
0.000000000000000 4.32044824293070 4.32044824293070
4.32044824293070 0.000000000000000 4.32044824293070
4.32044824293070 4.32044824293070 0.000000000000000
Fe 4 0
2 2 4
O 16wets (8) (A2B20C4)
0.375000000000000 0.375000000000000 0.375000000000000
0.375000000000000 0.125000000000000 0.125000000000000
0.425000000000000 0.425000000000000 0.375000000000000
0.375000000000000 0.375000000000000 0.375000000000000
0.125000000000000 0.425000000000000 0.425000000000000
0.000000000000000 0.000000000000000 0.000000000000000
0.500000000000000 0.000000000000000 0.000000000000000
0.000000000000000 0.500000000000000 0.500000000000000
0.500000000000000 0.000000000000000 0.000000000000000

```

Output Type Symmetry Tol. Magnetic Structure Magnetic moments

Main text light

c

Output

Structure conversion (Standard conventional) Lattice information (Real space)

```

REAL LATTICE
Real space lattice
0.0000e+00 4.3204e+00 4.3204e+00
0.0000e+00 0.0000e+00 4.3204e+00
4.3204e+00 4.3204e+00 0.0000e+00
a b c alpha beta gamma 4.11039429 4.11039429 4.11039429 60 60 60
a b c alpha beta gamma 11.8429482 11.8429482 11.8429482 60 60 60 Babar/Depr
Volume 141.7
a/b/c =
BRAGG LATTICE OF THE CRYSTAL (output_real)
Bravais Lattice Primitive = pcc
Lattice Translation = ortho
Lattice System = ortho
Pearson Symbol = oF8c

```

b

AFLOW Online

Submit Reset

Symmetry

More info on AFLOW-SYM

Space group Setting AFLOW standard Wyckoff positions Setting AFLOW standard

Pearson symbol AFLOW prototype

Crystal point group Equivalent atoms

Lattice information Lattice type Real space Crystallographic data Data type

Symmetry group Group Crystal point group

Structure comparison (XtalFinder)

More info on AFLOW_XtalFinder

Compare to library Method Compare to AFLOW database Unique atom decorations

Compare structures Comparison Mode Materials Structures are magnetic Structure 1

Structure 1 Structure 2

```

EXAMPLE POSCAR
1.000000
2.400000000000000 4.15692193816530 0.000000000000000
-2.400000000000000 4.15692193816530 0.000000000000000
0.000000000000000 2.77128129211020 4.320000000000000
Al 0
4 4
O 16wets (16) (A8B8)
0.450000000000000 0.450000000000000 0.050000000000000
0.450000000000000 0.250000000000000 0.450000000000000
0.350000000000000 0.350000000000000 0.350000000000000
0.150000000000000 0.150000000000000 0.150000000000000
0.750000000000000 0.440000000000000 0.750000000000000
0.040000000000000 0.750000000000000 0.750000000000000
0.440000000000000 0.560000000000000 0.750000000000000
0.250000000000000 0.560000000000000 0.250000000000000
0.940000000000000 0.250000000000000 0.250000000000000
0.560000000000000 0.940000000000000 0.250000000000000

```

Fig. 5. The AFLOW Online interface. (a) The structure input section, (b) a representative feature section, (c) the corresponding output section.

Below the structure input section are all AFLOW modules that are available via AFLOW Online. They are categorized based on the type of tasks they perform. Fig. 5(b) shows two example categories: symmetry and structure comparison [7,8]. The symmetry section contains all features provided by AFLOW-SYM: it calculates space groups, Wyckoff positions, Pearson symbols, prototype labels, symmetry groups, information on both direct and reciprocal lattice, and equivalency between atoms.

The structure comparison section is the interface to AFLOW-XtalFinder [8]. It provides the options to compare the input structure to the AFLOW database and the prototype encyclopedia, as well as determining unique atom decorations, in a simple button-drop-down setup. The application for comparing multiple structures is different: the button and drop down select the comparison mode (structure and material), but the structure is not taken from the input structure section. Instead, it has its own section where multiple inputs can be added in separate tabs. Magnetic moments can also be added for each structure using the “Structures are magnetic” button nearby.

To submit a calculation, the button for the feature of interest needs to be clicked — tooltips are available to help users select the appropriate functionality. Some options require additional input, e.g., through a drop-down menu. In the example shown in Fig. 5(b) (“Lattice information”), users can choose between real space, reciprocal space, and a superlattice with equal decorations. Pressing the “Submit” button will start the calculation and the screen is scrolled to the output section where the results will be displayed once available. Fig. 5(c) shows such a submission with the real space lattice information calculated by AFLOW for KFeO_2 . Output is stored for each submitted calculation and can be accessed by clicking on the corresponding tab — in this case, the prior calculation is a conversion of the input structure to the conventional unit cell. Submissions can be removed using the “x” buttons.

AFLOW Online also provides tools to prepare *ab-initio* calculations. Its structure file manipulation tools allow conversion between different structure file formats, coordinate systems (Cartesian and fractional),

and unit cell formats (primitive, AFLOW standard primitive [5], standard conventional, Niggli, and Minkowski). It also has modules for calculating the dimensions of a k-point mesh based on k-points per reciprocal atom or a maximum distance in reciprocal space, and to output the standard k-point path for band structure calculations based on the AFLOW standard [4]. The format for the k-points are in VASP’s KPOINTS file format. Disordered materials can also be set up using AFLOW-POCC [6]. Based on an input PARTCAR, the ideal supercell size can be determined, and the list of representative structures can be generated in POSCAR format. The latter can be converted into the desired format using the structure file manipulation tools.

Finally, the AFLOW Coordination Corrected Enthalpies (AFLOW-CCE) module [9,10] is available as well. It can calculate the corrections to the formation enthalpy for ionic materials calculated from density functional theory and supports the PBE, LDA, and SCAN functionals. It requires knowledge of the oxidation states in the material, which AFLOW can determine automatically. However, users also have the option to provide these numbers directly. The determined oxidation states and the coordination numbers of the cations can also be output separately through the appropriate buttons.

AFLOW Online is thus a user-friendly tool that provides a variety of features implemented in AFLOW for structural analysis, to support *ab-initio* calculations, and to correct energies from density functional theory. It is designed to be used in low-throughput settings without installing the entire AFLOW framework and without requiring a command line. Thus, it is particularly suited for new users and for users with little or no scripting experience.

4.2. AFLOW-CHULL Online

The AFLOW-CHULL web application (aflow.org/aflow-chull) was developed to provide an enhanced, command-line-free platform for the AFLOW-CHULL module and was introduced in Ref. [51]. The application offers an interactive visualization of binary and ternary convex hulls, enabling the selection and investigation of entries of interest. The ternary convex hull allows mouse-enabled pan and zoom, and has



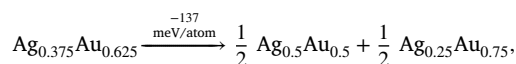
Fig. 6. Distance to the hull and stability criterion visualizations online. Example visualizations are provided of (a and b) the distance to the hull (and corresponding decompositions) and (c and d) stability criterion for binary (AgAu) and ternary (BeOsV) convex hulls. An inset of the original BeOsV hull is included in (b) for reference. (e) A snapshot of the information component populated with non-ground state and ground state entries.

been a useful instructional tool for understanding the stability analysis and grasping higher dimensional hulls with multi-component systems. Hulls can be selected from a responsive periodic table interface that reacts to the input based on the data availability. The number of entries is used as an indicator of the convergence of the analysis [51–54]: **green** (fully reliable, $N_{\text{entries}} \geq 200$), **orange** (potentially reliable, $100 \leq N_{\text{entries}} < 200$), **red** (unreliable, $N_{\text{entries}} < 100$), and **gray** (unavailable, $N_{\text{entries}} = 0$).

To accelerate loading speeds for repeated queries, a caching layer has been added. It saves the results of each request and loads them up automatically. The caching layer is cleared periodically and fresh calculations employing the most up-to-date database are triggered with a new request. Longer wait times for non-cached queries should be expected. Once the results are retrieved, the application loads the visualization viewport, prompting a redirect to the URL endpoint of the selected hull, e.g., /hull/BeOsV. While the URL is ubiquitous, it is dependent on the cached state at the time the query is submitted.

Users can select and highlight points on both binary and ternary hulls. When a point is selected, its name will appear within the sidebar. If the selected point is above the convex hull (meta-stable), the hull visualization will highlight the distance to the hull (vertical line) and tie-line/tie-surface directly below formed by stable phases (Figs. 6(a) and (b)), corresponding to the decomposition reaction in the thermodynamic limit. For example, in Fig. 6(a), the reduced decomposition

reaction is



where the distance to the hull is the energy gained from phase separation. If the selected point is on the convex hull (stable), the hull visualization will show the stability criterion [51,55] (Figs. 6(c) and (d)): the distance of the point from the pseudo-hull constructed without it, illustrated by the dotted lines and blue surfaces on binary and ternary hulls, respectively. This distance quantifies the effect of the phase on the convex hull, as well as its susceptibility to destabilization by a new phase that has yet to be explored. The stability criterion helped find two new magnetic phases, the first ever discovered with computation [55]. Clicking on the name of one of the selected points on the sidebar will bring up the information component (Fig. 6(e)), providing a selection of properties, such as the distance to the hull and stability criterion, with a link to the aflow.org entry page (“More Info”) offering the full set of properties for the entry.

New to the stability analysis is the calculation of the $N+1$ enthalpy gain [56]: the distance to the hull for an N -compound (binaries are 2-compounds) from the pseudo-hull constructed with $\{1, \dots, N-1\}$ -compounds. The descriptor was used to establish the role of disorder in stabilizing multi-component systems [56]. It was shown that, with an increasing number of species, the average enthalpy gained rapidly

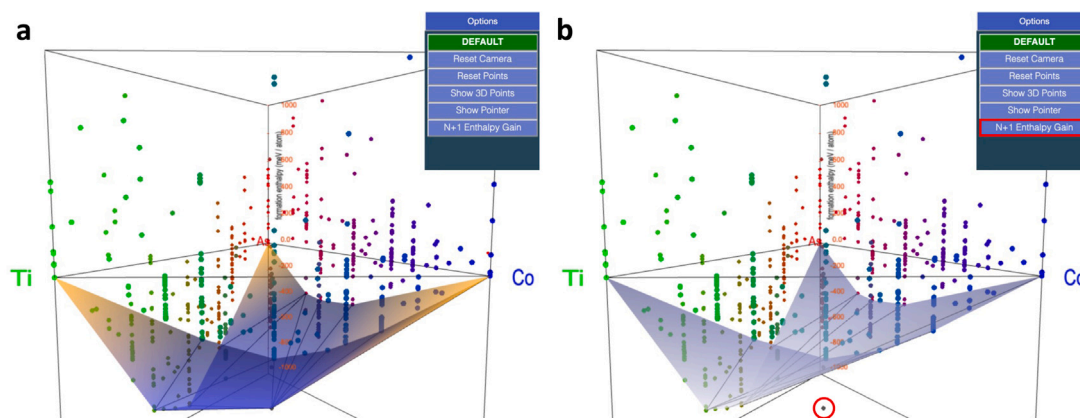


Fig. 7. Visualization of the $N + 1$ enthalpy gain descriptor. The original AsCoTi convex hull is plotted in (a) and the pseudo-hull composed only of unaries and binaries is shown in blue in (b). A stable ternary – not included in the pseudo-hull construction – is highlighted in red.

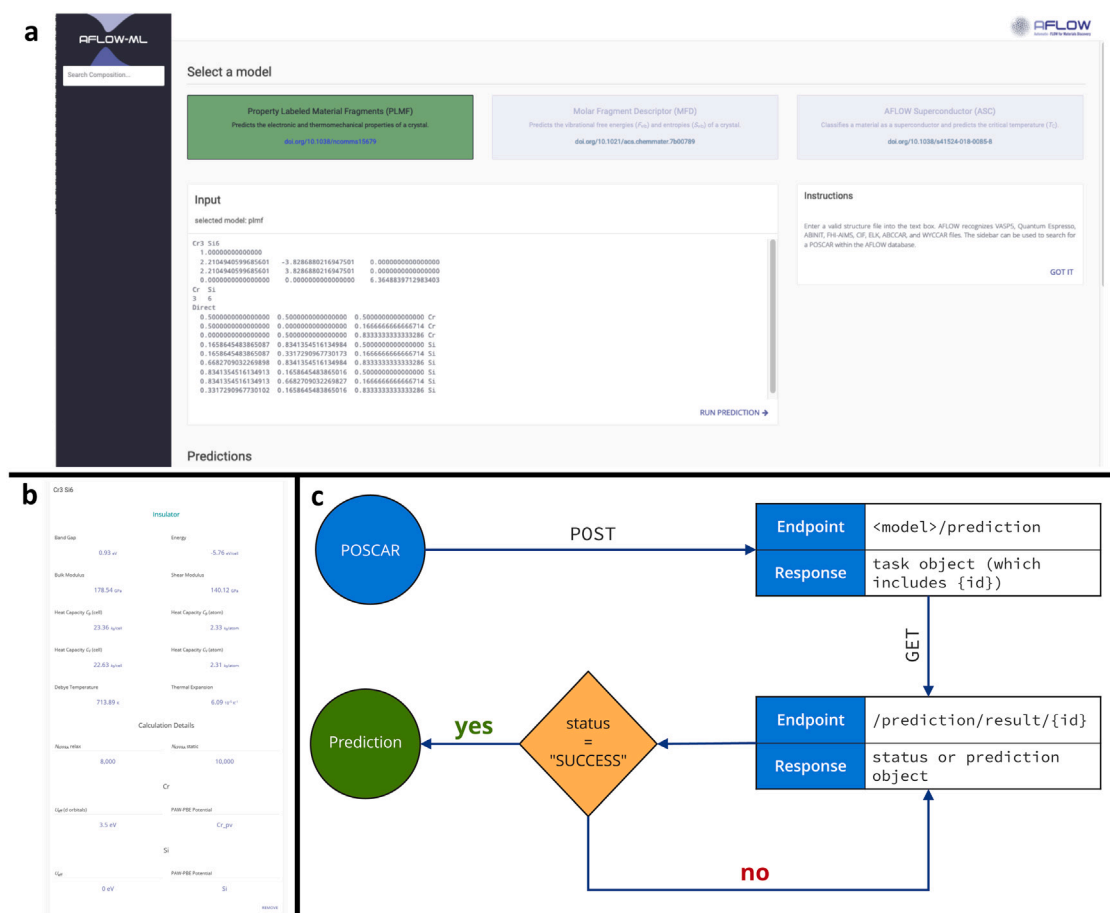


Fig. 8. AFLOW-ML. (a) The AFLOW-ML Online web interface, with three different models available; (b) card with prediction results for AFLOW-ML Online; (c) the AFLOW-ML API enables programmatic access to the models.

diminishes with respect to the configurational entropy gained. The descriptor corresponds to the formation enthalpy for binaries, and for ternaries becomes the distance from the pseudo-hull constructed with the full set of unaries (A , B , and C) and binaries ($A - B$, $B - C$, and $A - C$). An example visualization of the analysis is provided in Fig. 7, where the full AsCoTi convex hull and its $\{1, 2\}$ -pseudo-hull are shown in Figs. 7(a) and (b), respectively. A stable ternary, not included in the pseudo-hull construction, is highlighted in red. The pseudo-hull can be visualized from the Options menu.

4.3. AFLOW-ML Online

The AFLOW machine learning (AFLOW-ML) online application provides a user interface to leverage machine-learning models trained on AFLOW data, as shown in Fig. 8. The application can be accessed through a web browser at aflow.org/aflow-ml, or through the AFLOW-ML API described below. The application takes in structural and/or compositional information as an input and outputs predictions, including electronic, thermomechanical, and superconducting properties.

This application provides an accessible medium to retrieve machine learning predictions without the need to install specialized software libraries or machine learning packages.

Currently, AFLOW-ML supports three different machine-learning models: **i.** The `property_labeled_materials_fragments` model [57], `plmf`, has been trained using data from the `aflow.org` repository, and predicts properties such as the electronic band gap, specific heat capacities, bulk and shear moduli, Debye temperature, and coefficient of thermal expansion. It is based on structural information, using a Voronoi tessellation to identify bonded or “connected” atoms to form a crystal graph, and thus requires a structure file describing the material. Atoms in the graph are labeled with their corresponding elemental properties, and pieces of the graph form the property-labeled materials fragments that form the feature vector for the model. **ii.** The `molar_fraction_descriptor` model [58], `mfd`, predicts vibrational properties such as vibrational free energy and entropy, and is based only on the chemical composition of the material. **iii.** The `AFLOW_Superconductor` model [59], `asc`, predicts the superconducting critical temperature of materials based on their chemical composition, and accepts the chemical formula as input.

Using the online application involves, first, selecting the required model from the three listed and then, posting either a structure file (in the case of the `plmf` and `mfd` models) or a chemical formula (for `asc`) into the “Input” box, as illustrated in Fig. 8(a). Structures can be in any format that AFLOW can read, including VASP POSCAR [44–46], QUANTUM ESPRESSO [47], ABINIT [49], ELK [50], FHI-AIMS [48], and Crystallographic Information Files (CIF) [60]. Additionally, structures within the `aflow.org` repository can be imported via the sidebar. The structure is then submitted and the model is run using the “RUN PREDICTION” button. When the prediction is complete (which usually takes several seconds, depending on the material, model, and the demand on the server), a card will be displayed containing the predicted property values, as shown in Fig. 8(b).

The AFLOW-ML API [61] offers programmatic access to the AFLOW-ML online application, and provides a simplified abstraction that facilitates leveraging powerful machine-learning models. This distills the prediction process down to its essence: from a structure file, return a prediction. Using the API is a two-step process as illustrated in Fig. 8(c): first a structure file is posted (i.e. uploaded) to the endpoint `<server>/<model>/prediction` on the `aflow.org` server using standard HTTP libraries or dedicated programs such `curl` or `wget`, where `<server>` is `aflow.org/API/aflow-ml`, and `<model>` specifies the machine-learning model to use in the prediction (current options: `plmf` and `mfd`). For example, the full endpoint for uploading for the `plmf` model would be `aflow.org/API/aflow-ml/plmf/prediction`. When a prediction is submitted, a JSON response object is returned that includes a task id. Second, the results of the prediction need to be retrieved from the `/prediction/result/` endpoint on the `aflow.org` server by appending the task id to the end of the URL, i.e. `/prediction/result/{id}/`. This endpoint monitors the prediction task and responds with a JSON object that details its status. When complete, the endpoint responds with the results of the prediction, represented as a JSON object containing a key-value pair for each predicted property.

The AFLOW-ML API can also be accessed using the AFLOW-ML Python client, which is available for download at `aflow.org/src/aflow-ml/`. Upon installation, the Python client can be accessed through the command-line interface using the command `aflow-ml`. Predictions tasks can be submitted by specifying the structure file name and the machine-learning model. For example, a structure in the file `test.poscar` can be submitted to the `plmf` machine-learning model using the command `aflowml test.poscar --model=plmf`, where the model is specified using the flags `--model` or `-m`. The results can be saved to a file using the flag `-s` or `--save`, and the output filename can be specified using the flag `--outfile` (the default filename is `prediction.txt`). The output format can be set using the `--output` flag, while the output can be limited to specific predicted fields using the `--fields` flag. Verbose mode can be activated using the flags `-v` or `--verbose`.

5. Education and outreach

In addition to the rational data dissemination discussed above, the AFLOW consortium is engaged in global education and outreach activities to train the next cadre of materials researchers.

AFLOW Schools. We offer tutorials, virtual and in-person, teaching the structure of our repositories and the strategies to retrieve data to a diverse audience. Particular emphasis is put on the discussion of the features of the AFLOW C++ code and python interfaces, so that they can be easily adapted into workflows. These schools cover the AFLOW code [1], DFT, symmetry determination, prototypes, convex hulls, partial occupation, coordination corrected enthalpies, elastic and Gibbs library extensions, database integration, machine learning within AFLOW, prototype crystal finder, and harmonic and quasiharmonic phonon calculations. Videos, slides presentations, and other educational materials are available online at `aflow.org/aflow-school/`.

AFLOW Seminars. We organize biweekly free virtual scientific seminars in the field of materials science and materials physics. Speakers are experts in experimental, theoretical or computational techniques, covering a variety of sub-fields. These seminars attract a global audience from over 50 countries on five continents, providing access to scientific talks that would otherwise require considerable resources to attend. More information can be found online at `aflow.org/seminars/`.

6. Summary

This article describes the `aflow.org` web presence, an ecosystem of databases and online applications for the AFLOW repository and software. Dissemination of the AFLOW data follows the FAIR principles for data stewardship. Access methods include graphical and programmatic as well as low-throughput and high-throughput retrieval methods, covering a wide range of user experience and usage requirements. `aflow.org` also aids the generation of new data through its Prototype Encyclopedia — a collection of over 1,100 crystallographic prototypes — and provides tools to create structure files in a variety of formats. `aflow.org` offers applications for interacting with many AFLOW software modules and machine learning methods, easing the use by students and researchers of all experience levels. Educational efforts such as AFLOW Schools and free seminars further support the effort to make AFLOW and materials science research accessible to a global audience.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data is open and available in <https://aflow.org>.

Acknowledgments

The authors thank Yoav Lederer, Ohad Levy, Eric Gossett, Cheryl Li, Harry Wang, Mana Rose, William Schmitt, and Stuart Ki for fruitful discussions. Research sponsored by DOD-ONR, USA. All the authors acknowledge support by the United States Department of Defense Office of Naval Research, USA (DOD-ONR) (N00014-21-1-2132, N00014-20-1-2525, N00014-20-1-2299) and by the National Science Foundation, USA (NSF) (NRT-HDR DGE-2022040). C.T. acknowledges support from NSF, USA (DMR-2219788). R.F. acknowledges support from the Alexander von Humboldt foundation, Germany under the Feodor Lynen research fellowship.

Appendix. AFLOW keywords

The AFLOW database contains a large variety of materials properties that can be queried through the appropriate keyword using the REST-API or AFLUX. AFLOW goes beyond data and provides rich metadata in multiple ways: *i.* through the keywords, which contain identifiers for the methods and modules used to compute the property (e.g., `ael_bulk_modulus_reuss` describes the bulk modulus calculated with the Reuss method using the AFLOW-AEL module); *ii.* by serving detailed information on how the data was generated with each entry, such as input parameters needed to reproduce results (e.g., cut-off energies and pseudopotentials), software versions, data sources, and more; and *iii.* by providing metadata for the keywords themselves – including references to the methods used – that can be accessed at aflow.org/API/aapi-schema/. This fulfills the “Reusable” criterion of the FAIR principles, which requires describing the context of the data so that users can assess its suitability for their particular purpose.

This Appendix contains all available keywords at the time of this publication. It provides a description for each keyword along with references to the methods used to calculate the property, units (where applicable), the data type of the property, and an example value. The list is also available online at aflow.org/documentation/ where it is continuously updated.

<code>ael_applied_pressure</code> [21]
Returns the applied pressure for the AEL calculations.
Unit: GPa Type: number Example: <code>ael_applied_pressure=0.0</code>
<code>ael_average_external_pressure</code> [21]
Returns the average external pressure for the AEL calculations.
Unit: GPa Type: number Example: <code>ael_average_external_pressure=0.079875</code>
<code>ael_bulk_modulus_reuss</code> [12,21]
Returns the bulk modulus calculated, using the Reuss method, by AEL.
Unit: GPa Type: number Example: <code>ael_bulk_modulus_reuss=105.315</code>
<code>ael_bulk_modulus_voigt</code> [12,21]
Returns the bulk modulus calculated, using the Voigt method, by AEL.
Unit: GPa Type: number Example: <code>ael_bulk_modulus_voigt=105.315</code>
<code>ael_bulk_modulus_vrh</code> [12,21]
Returns the bulk modulus calculated, using the Voigt–Reuss–Hill average, by AEL.
Unit: GPa Type: number Example: <code>ael_bulk_modulus_vrh=105.315</code>
<code>ael_compliance_tensor</code> [21]
Returns the compliance tensor calculated by AEL.
Unit: GPa^{-1} Type: numbers Example: <code>74.2667,49.5167,49.5167,0,0,0;49.5167,74.2667,49.5167,0,0,0;49.5167,49.5167,74.2667,0,0,0;0,0,0,27.2833,0,0;0,0,0,27.2833,0,0;0,0,0,0,27.2833</code>

<code>ael_debye_temperature</code> [21]
Returns the Debye temperature calculated by AEL.
Unit: K Type: number Example: <code>ael_debye_temperature=163.348</code>
<code>ael_elastic_anisotropy</code> [12,21]
Returns the elastic anisotropy calculated by AEL.
Unit: none Type: number Example: <code>ael_elastic_anisotropy=0.0008165</code>
<code>ael_poisson_ratio</code> [12,21]
Returns the isotropic Poisson ratio calculated by AEL.
Unit: none Type: number Example: <code>ael_poisson_ratio=0.216</code>
<code>ael_pughs_modulus_ratio</code> [21]
Returns the Pugh’s modulus ratio calculated by AEL.
Unit: none Type: number Example: <code>ael_pughs_modulus_ratio=0.332265</code>
<code>ael_shear_modulus_reuss</code> [12,21]
Returns the shear modulus calculated, using the Reuss method, by AEL.
Unit: GPa Type: number Example: <code>ael_shear_modulus_reuss=73.787</code>
<code>ael_shear_modulus_voigt</code> [12,21]
Returns the shear modulus calculated, using the Voigt method, by AEL.
Unit: GPa Type: number Example: <code>ael_shear_modulus_voigt=73.799</code>
<code>ael_shear_modulus_vrh</code> [12,21]
Returns the shear modulus calculated, using the Voigt–Reuss–Hill average, by AEL.
Unit: GPa Type: number Example: <code>ael_shear_modulus_vrh=73.793</code>
<code>ael_speed_sound_average</code> [12,21]
Returns the average speed of sound calculated by AEL.
Unit: m/s Type: number Example: <code>ael_speed_sound_average=1540.09</code>
<code>ael_speed_sound_longitudinal</code> [12,21]
Returns the longitudinal speed of sound calculated by AEL.
Unit: m/s Type: number Example: <code>ael_speed_sound_longitudinal=2147.05</code>
<code>ael_speed_sound_transverse</code> [12,21]
Returns the transverse speed of sound calculated by AEL.
Unit: m/s Type: number Example: <code>ael_speed_sound_transverse=1405.57</code>

<u>ael_stiffness_tensor</u> [21]	<u>aflowlib_date</u> [11]
Returns the stiffness tensor calculated by AEL.	Returns the date when the AFLOW post-processor generated the entry in the library.
Unit: GPa Type: numbers	Unit: none Type: string
Example: 0.0288595,-0.0115446,-0.0115446,0,0,0;-0.0115446,0.0288595,-0.0115446,0,0,0;-0.0115446,-0.0115446,0.0288595,0,0,0;0,0,0,0.0366524,0,0,0,0,0,0.0366524,0,0,0,0,0.0366524	Example: aflowlib_date=20140204_13:10:39_GMT-5
<u>ael_youngs_modulus_vrh</u> [21]	<u>aflowlib_version</u> [11]
Returns the Young modulus calculated, using the Voigt–Reuss–Hill average, by AEL.	Returns the version of the AFLOW post-processor which generated the entry in the library.
Unit: GPa Type: number	Unit: none Type: string
Example: ael_youngs_modulus_vrh=31.6223	Example: aflowlib_version=3.1.103
<u>aflow_prototype_label_orig</u> [8]	<u>agl_acoustic_debye</u> [12,21,27]
Returns the AFLOW prototype label for the unrelaxed structure.	Returns the acoustic Debye temperature calculated by AGL.
Unit: none Type: string	Unit: K Type: number
Example: aflow_prototype_label_orig=A2BC4_cF56_227_c_b_e	Example: agl_acoustic_debye=492
<u>aflow_prototype_label_relax</u> [8]	<u>agl_bulk_modulus_isothermal_300K</u> [12,21,27]
Returns the AFLOW prototype label for the relaxed structure.	Returns the isothermal bulk modulus calculated by AGL at 300 K.
Unit: none Type: string	Unit: GPa Type: number
Example: aflow_prototype_label_relax=A2BC4_cF56_227_c_b_e	Example: agl_bulk_modulus_isothermal_300K=96.6
<u>aflow_prototype_params_list_orig</u> [8]	<u>agl_bulk_modulus_static_300K</u> [12,21,27]
Returns the AFLOW prototype parameter labels for the unrelaxed structure.	Returns the static bulk modulus calculated by AGL at 300 K.
Unit: none Type: strings	Unit: GPa Type: number
Example: aflow_prototype_params_list_orig=a,x3	Example: agl_bulk_modulus_static_300K=99.6
<u>aflow_prototype_params_list_relax</u> [8]	<u>agl_debye</u> [12,21,27]
Returns the AFLOW prototype parameter labels for the relaxed structure.	Returns the Debye temperature calculated by AGL.
Unit: none Type: strings	Unit: K Type: number
Example: aflow_prototype_params_list_relax=a,x3	Example: agl_debye=620
<u>aflow_prototype_params_values_orig</u> [8]	<u>agl_gruneisen</u> [12,21,27]
Returns the AFLOW prototype parameter values for the unrelaxed structure.	Returns the Grüneisen parameter calculated by AGL.
Unit: none Type: numbers	Unit: none Type: number
Example: aflow_prototype_params_values_orig=10.336,0.757	Example: agl_gruneisen=2.06
<u>aflow_prototype_params_values_relax</u> [8]	<u>agl_heat_capacity_Cp_300K</u> [12,21,27]
Returns the AFLOW prototype parameter values for the relaxed structure.	Returns the heat capacity per cell, at constant pressure, calculated by AGL at 300 K.
Unit: none Type: numbers	Unit: k_B /cell Type: number
Example: aflow_prototype_params_values_relax=10.336,0.757	Example: agl_heat_capacity_Cp_300K=5.502
<u>aflow_version</u> [2,11]	<u>agl_heat_capacity_Cv_300K</u> [12,21,27]
Returns the version number of AFLOW used to perform the calculation.	Returns the heat capacity per cell, at constant volume, calculated by AGL at 300 K.
Unit: none Type: string	Unit: k_B /cell Type: number
Example: aflow_version=aflow30641	Example: agl_heat_capacity_Cv_300K=4.901
	<u>agl_poisson_ratio_source</u> [12,21,27]
	Returns the source of the Poisson ratio used for AGL calculations.
	Unit: none Type: string
	Example: agl_poisson_ratio_source=ael_poisson_ratio_0.350433

<u>agl_thermal_conductivity_300K</u> [12,21,27]
Returns the thermal conductivity calculated by AGL at 300 K.
Unit: W m ⁻¹ K ⁻¹ Type: number Example: agl_thermal_conductivity_300K=24.41
<u>agl_thermal_expansion_300K</u> [12,21,27]
Returns the thermal expansion coefficient calculated by AGL at 300 K.
Unit: K ⁻¹ Type: number Example: agl_thermal_expansion_300K=4.997e-05
<u>agl_vibrational_entropy_300K_atom</u> [21,27]
Returns the vibrational entropy per atom calculated by AGL at 300 K.
Unit: meV/(K atom) Type: number Example: agl_vibrational_entropy_300K_atom=0.454761
<u>agl_vibrational_entropy_300K_cell</u> [21,27]
Returns the vibrational entropy per cell calculated by AGL at 300 K.
Unit: meV/(K cell) Type: number Example: agl_vibrational_entropy_300K_cell=9.09521
<u>agl_vibrational_free_energy_300K_atom</u> [21,27]
Returns the vibrational free energy per atom calculated by AGL at 300 K.
Unit: meV/atom Type: number Example: agl_vibrational_free_energy_300K_atom=-57.1897
<u>agl_vibrational_free_energy_300K_cell</u> [21,27]
Returns the vibrational free energy per cell calculated by AGL at 300 K.
Unit: meV/cell Type: number Example: agl_vibrational_free_energy_300K_cell=-1143.79
<u>aid</u> [11]
Returns the AFLOWLIB unique identifier (AUID) for the entry.
Unit: none Type: string Example: aid=aflow:e9c6d914c4b8d9ca
<u>aurl</u> [11]
Returns the AFLOWLIB uniform resource locator (AURL) for the entry.
Unit: none Type: string Example: aurl=aflowlib.duke.edu:AFLOWDATA/LIB3_RAW/Bi_dR_h_pvTi_sv/T0003.ABC:LDAU2
<u>bader_atomic_volumes</u> [28]
Returns the volume of each atom calculated by the Atoms in Molecules (AIM) Bader analysis.
Unit: Å ³ Type: numbers Example: bader_atomic_volumes=15.235,12.581,13.009

<u>bader_net_charges</u> [28]
Returns the partial charge of each atom calculated by the Atoms in Molecules (AIM) Bader analysis.
Unit: e ⁻ Type: numbers Example: bader_net_charges=0.125,0.125,-0.25
<u>Bravais_lattice_lattice_system</u> [2,7]
Returns the Bravais lattice of the lattice system for the relaxed structure.
Unit: none Type: string Example: Bravais_lattice_lattice_system=cubic
<u>Bravais_lattice_lattice_system_orig</u> [2,7]
Returns the Bravais lattice of the lattice system for the unrelaxed structure.
Unit: none Type: string Example: Bravais_lattice_lattice_system=cubic
<u>Bravais_lattice_lattice_type</u> [2,7]
Returns the lattice centering type for the relaxed structure.
Unit: none Type: string Example: Bravais_lattice_lattice_type=BCC
<u>Bravais_lattice_lattice_type_orig</u> [2,7]
Returns the lattice centering type for the unrelaxed structure.
Unit: none Type: string Example: Bravais_lattice_lattice_type_orig=BCC
<u>Bravais_lattice_lattice_variation_type</u> [2,7]
Returns the Bravais lattice variation of the lattice system for the relaxed structure.
Unit: none Type: string Example: Bravais_lattice_lattice_variation_type=BCC
<u>Bravais_lattice_lattice_variation_type_orig</u> [2,7]
Returns the Bravais lattice variation of the lattice system for the unrelaxed structure.
Unit: none Type: string Example: Bravais_lattice_lattice_variation_type_orig=BCC
<u>Bravais_lattice_orig</u> [2,4,7,11]
Returns the Bravais lattice of the crystal for the unrelaxed structure.
Unit: none Type: string Example: Bravais_lattice_orig=MCLC
<u>Bravais_lattice_relax</u> [2,4,7,11]
Returns the Bravais lattice of the crystal for the relaxed structure.
Unit: none Type: string Example: Bravais_lattice_relax=MCLC

<p><u>Bravais_superlattice_lattice_system</u> [2,7]</p> <p>Returns the Bravais superlattice of the lattice system for the relaxed structure.</p> <p>Unit: none Type: string Example: Bravais_superlattice_lattice_system=cubic</p>	<p><u>catalog</u> [2]</p> <p>Returns the database name for the calculation.</p> <p>Unit: none Type: string Example: catalog=icsd</p>
<p><u>Bravais_superlattice_lattice_system_orig</u> [2,7]</p> <p>Returns the Bravais superlattice of the lattice system for the unrelaxed structure.</p> <p>Unit: none Type: string Example: Bravais_superlattice_lattice_system_orig=cubic</p>	<p><u>code</u> [11]</p> <p>Returns the software name and version used to perform the calculation.</p> <p>Unit: none Type: string Example: code=vasp.4.6.35</p>
<p><u>Bravais_superlattice_lattice_type</u> [2,7]</p> <p>Returns the Bravais superlattice centering type for the relaxed structure.</p> <p>Unit: none Type: string Example: Bravais_superlattice_lattice_type=BCC</p>	<p><u>composition</u> [11]</p> <p>Returns the number of atoms per type in the simulation cell.</p> <p>Unit: none Type: numbers Example: composition=2,6,6</p>
<p><u>Bravais_superlattice_lattice_type_orig</u> [2,7]</p> <p>Returns the Bravais superlattice centering type for the unrelaxed structure.</p> <p>Unit: none Type: string Example: Bravais_superlattice_lattice_type_orig=BCC</p>	<p><u>compound</u> [11]</p> <p>Returns the chemical formula of the structure.</p> <p>Unit: none Type: string Example: compound=Co2Er6Si6</p>
<p><u>Bravais_superlattice_lattice_variation_type</u> [2,7]</p> <p>Returns the Bravais superlattice variation of the lattice system for the relaxed structure.</p> <p>Unit: none Type: string Example: Bravais_superlattice_lattice_variation_type=BCC</p>	<p><u>crystal_class</u> [7]</p> <p>Returns the crystal class for the relaxed structure.</p> <p>Unit: none Type: string Example: crystal_class=tetrahedral</p>
<p><u>Bravais_superlattice_lattice_variation_type_orig</u> [2,7]</p> <p>Returns the Bravais superlattice variation of the lattice system for the unrelaxed structure.</p> <p>Unit: none Type: string Example: Bravais_superlattice_lattice_variation_type_orig=BCC</p>	<p><u>crystal_class_orig</u> [7]</p> <p>Returns the crystal class for the unrelaxed structure.</p> <p>Unit: none Type: string Example: crystal_class_orig=tetrahedral</p>
<p><u>calculation_cores</u> [11]</p> <p>Returns the number of CPUs used by the calculation.</p> <p>Unit: none Type: number Example: calculation_cores=32</p>	<p><u>crystal_family</u> [7]</p> <p>Returns the crystal family for the relaxed structure.</p> <p>Unit: none Type: string Example: crystal_family=cubic</p>
<p><u>calculation_memory</u> [11]</p> <p>Returns the maximum RAM used by the calculation.</p> <p>Unit: MB Type: number Example: calculation_memory=32</p>	<p><u>crystal_family_orig</u> [7]</p> <p>Returns the crystal family for the unrelaxed structure.</p> <p>Unit: none Type: string Example: crystal_family_orig=cubic</p>
<p><u>calculation_time</u> [11]</p> <p>Returns the total time taken by the calculation.</p> <p>Unit: seconds Type: number Example: calculation_time=32</p>	<p><u>crystal_system</u> [7]</p> <p>Returns the crystal system for the relaxed structure.</p> <p>Unit: none Type: string Example: crystal_system=cubic</p>
	<p><u>crystal_system_orig</u> [7]</p> <p>Returns the crystal system for the unrelaxed structure.</p> <p>Unit: none Type: string Example: crystal_system_orig=cubic</p>

data_api [11] Returns the REST API version for the entry. Unit: none Type: string Example: data_api=aapi1.0	energy_atom [2,11] Returns the total ab-initio energy per atom. Unit: eV/atom Type: number Example: energy_atom=-82.1656
data_source [11] Returns the data source for the entry. Unit: none Type: string Example: data_source=afloplib	energy_cell [2,11] Returns the total ab-initio energy per cell. Unit: eV/cell Type: number Example: energy_cell=-82.1656
delta_electronic_energy_convergence [12,17] Returns the change in total energy from the last step of the self-consistent field (SCF) iteration. Unit: eV Type: number Example: delta_electronic_energy_convergence=6.09588e-05	energy_cutoff [11,17] Return the plane-wave energy cut-off used for the calculation. Unit: eV Type: number Example: energy_cutoff=384.1,384.1,384.1
delta_electronic_energy_threshold [12,17] Returns the threshold for the self-consistent field (SCF) convergence. Unit: eV Type: number Example: delta_electronic_energy_threshold=0.0001	enthalpy_atom [11] Returns the enthalpy per atom. Unit: eV/atom Type: number Example: enthalpy_atom=-82.1656
density [11] Returns the mass density of the unit cell. Unit: g/cm ³ Type: number Example: density=7.76665	enthalpy_cell [11] Returns the enthalpy per cell. Unit: eV/cell Type: number Example: enthalpy_cell=-82.1656
dft_type [11,17] Returns the level of theory used in the calculation, i.e., pseudopotential type, exchange-correlation functional used, and use of GW. Unit: none Type: strings Example: dft_type=PAW_PBE	enthalpy_formation_atom [2,11] Returns the formation enthalpy per atom. Unit: eV/atom Type: number Example: enthalpy_formation_atom=-33.1587
eentropy_atom [11] Returns the electronic entropy per atom used to converge the calculation. Unit: eV/atom Type: number Example: eentropy_atom=0.0011	enthalpy_formation_cell [2,11] Returns the formation enthalpy per cell. Unit: eV/cell Type: number Example: enthalpy_formation_cell=-33.1587
eentropy_cell [11] Returns the electronic entropy per cell used to converge the calculation. Unit: eV/cell Type: number Example: eentropy_cell=0.0011	entropic_temperature [29,53] Returns the entropic temperature. Unit: K Type: number Example: entropic_temperature=1072.1
Egap [2,4,11,17] Returns the electronic band gap. Unit: eV Type: number Example: Egap=2.5	files [11] Returns the input and output files used in the simulation. Unit: none Type: strings Example: files=Bi_dRh_pv.33.cif,Bi_dRh_pv.33.png,CONTCAR.relax,CONTCAR.relax1,
Egap_type [2,4,11,17] Returns the electronic band gap type. Unit: none Type: string Example: Egap_type=insulator_direct	forces [11] Returns the forces on the atoms for the relaxed structure. Unit: eV/Å Type: numbers Example: forces=0,-0.023928,0.000197;0,0.023928,-0.000197;...

<u>geometry</u> [2,4,7,11] Returns the lattice parameters of the relaxed simulation cell. Unit: none Type: numbers Example: geometry=18.82,18.82,18.82,32.41,32.41,32.41	<u>ldau_j</u> [11,17] Returns the J parameters of the DFT+U calculation. Unit: eV Type: numbers Example: ldau_j=0,0,0
<u>geometry_orig</u> [2,4,7,11] Returns the lattice parameters of the unrelaxed simulation cell. Unit: none Type: numbers Example: geometry_orig=18.82,18.82,18.82,32.41,32.41,32.41	<u>ldau_l</u> [11,17] Returns the orbitals of the DFT+U calculation. Unit: none Type: numbers Example: ldau_l=2,0,0
<u>kpoints_bands_nkpts</u> [4,11,12,17] Returns the number of points, between the high-symmetry k-points, used for the band structure calculation. Unit: none Type: number Example: kpoints_bands_nkpts=20	<u>ldau_type</u> [11,17] Returns the type of DFT+U calculation performed. Unit: none Type: number Example: ldau_type=2
<u>kpoints_bands_path</u> [4,11,12,17] Returns the high-symmetry k-point path used for the band structure calculation. Unit: none Type: strings Example: kpoints_bands_path=G-X-W-K-G-L-U-W-K-K-U-X	<u>ldau_u</u> [11,17] Returns the U parameters of the DFT+U calculation. Unit: eV Type: numbers Example: ldau_u=5,0,0
<u>kpoints_relax</u> [4,11,12,17] Returns the k-point grid used for the structural relaxation calculation. Unit: none Type: numbers Example: kpoints_relax=14,14,14	<u>loop</u> [11] Returns information about the type of post-processing that was performed. Unit: none Type: strings Example: loop=thermodynamics,bands,magnetic
<u>kpoints_static</u> [4,11,12,17] Returns the k-point grid used for the static calculation. Unit: none Type: numbers Example: kpoints_static=14,14,14	<u>natoms</u> [11] Returns the number of atoms in the simulation cell. Unit: none Type: number Example: natoms=12
<u>lattice_system_orig</u> [2,4,7,11] Returns the lattice system for the unrelaxed structure. Unit: none Type: string Example: lattice_system_orig=rhombohedral	<u>nbondxx</u> [11] Returns the nearest neighbors distances for the relaxed structure. Unit: Å Type: numbers Example: nbondxx=1.2599,1.0911,1.0911,1.7818,1.2599,1.7818
<u>lattice_system_relax</u> [2,4,7,11] Returns the lattice system for the relaxed structure. Unit: none Type: string Example: lattice_system_relax=rhombohedral	<u>node_CPU_Cores</u> [11] Returns information about the number of CPUs on the node/cluster where the calculation was performed. Unit: none Type: number Example: node_CPU_Cores=12
<u>lattice_variation_orig</u> [2,4,7,11] Returns the lattice variation for the unrelaxed structure. Unit: none Type: string Example: lattice_variation_orig=rhombohedral	<u>node_CPU_MHz</u> [11] Returns information about the speed of CPUs on the node/cluster where the calculation was performed. Unit: MHz Type: number Example: node_CPU_MHz=12
<u>lattice_variation_relax</u> [2,4,7,11] Returns the lattice variation for the relaxed structure. Unit: none Type: string Example: lattice_variation_relax=rhombohedral	<u>node_CPU_Model</u> [11] Returns information about the model of CPUs on the node/cluster where the calculation was performed. Unit: none Type: string Example: node_CPU_Model=12

<u>node_RAM_GB</u> [11]	<u>point_group_order</u> [2,7]
Returns information about the RAM on the node/cluster where the calculation was performed.	Returns the point group order for the relaxed structure.
Unit: Gb Type: number	Unit: none Type: number
Example: node_RAM_GB=12	Example: point_group_order=24
<u>nspecies</u> [2,11]	<u>point_group_order_orig</u> [2,7]
Returns the number of unique species in the structure.	Returns the point group order for the unrelaxed structure.
Unit: none Type: number	Unit: none Type: number
Example: nspecies=3	Example: point_group_order_orig=24
<u>Pearson_symbol_orig</u> [2,4,7,11]	<u>point_group_Schoenflies</u> [2,7]
Returns the Pearson symbol for the unrelaxed structure.	Returns the point group, in Schoenflies notation, for the relaxed structure.
Unit: none Type: string	Unit: none Type: string
Example: Pearson_symbol_orig=mS32	Example: point_group_Schoenflies=T_d
<u>Pearson_symbol_relax</u> [2,4,7,11]	<u>point_group_Schoenflies_orig</u> [2,7]
Returns the Pearson symbol for the relaxed structure.	Returns the point group, in Schoenflies notation, for the unrelaxed structure.
Unit: none Type: string	Unit: none Type: string
Example: Pearson_symbol_relax=mS32	Example: point_group_Schoenflies_orig=T_d
<u>Pearson_symbol_superlattice</u> [2,7]	<u>point_group_structure</u> [2,7]
Returns the Pearson symbol of the superlattice for the relaxed structure.	Returns the point group structure for the relaxed structure.
Unit: none Type: string	Unit: none Type: string
Example: Pearson_symbol_superlattice=cI52	Example: point_group_structure=symmetric
<u>Pearson_symbol_superlattice_orig</u> [2,7]	<u>point_group_structure_orig</u> [2,7]
Returns the Pearson symbol of the superlattice for the unrelaxed structure.	Returns the point group structure for the unrelaxed structure.
Unit: none Type: string	Unit: none Type: string
Example: Pearson_symbol_superlattice_orig=cI52	Example: point_group_structure_orig=symmetric
<u>point_group_Hermann_Mauguin</u> [2,7]	<u>point_group_type</u> [2,7]
Returns the point group, in Hermann–Mauguin notation, for the relaxed structure.	Returns the point group type for the relaxed structure.
Unit: none Type: string	Unit: none Type: string
Example: point_group_Hermann_Mauguin=-43m	Example: point_group_type=non-centrosymmetric,non-enantiomorphic,non-polar
<u>point_group_Hermann_Mauguin_orig</u> [2,7]	<u>point_group_type_orig</u> [2,7]
Returns the point group, in Hermann–Mauguin notation, for the unrelaxed structure.	Returns the point group type for the unrelaxed structure.
Unit: none Type: string	Unit: none Type: string
Example: point_group_Hermann_Mauguin_orig=-43m	Example: point_group_type_orig=non-centrosymmetric,non-enantiomorphic,non-polar
<u>point_group_orbifold</u> [2,7]	<u>positions_cartesian</u> [11]
Returns the point group orbifold for the relaxed structure.	Returns the Cartesian coordinates of the atoms for the relaxed structure.
Unit: none Type: string	Unit: Å Type: numbers
Example: point_group_orbifold=*332	Example: positions_cartesian=0,0,0;18.18438,0,2.85027;...
<u>point_group_orbifold_orig</u> [2,7]	
Returns the point group orbifold for the unrelaxed structure.	
Unit: none Type: string	
Example: point_group_orbifold_orig=*332	

<p><u>positions_fractional</u> [11]</p> <p>Returns the fractional coordinates of the atoms for the relaxed structure.</p> <p>Unit: none Type: numbers Example: positions_fractional=0,0,0;0.25,0.25,0.25;...</p>	<p><u>reciprocal_lattice_type</u> [2,4,7]</p> <p>Returns the reciprocal lattice centering type for the relaxed structure.</p> <p>Unit: none Type: string Example: reciprocal_lattice_type=FCC</p>
<p><u>pressure</u> [11]</p> <p>Returns the hydrostatic pressure on the simulation cell for the unrelaxed structure.</p> <p>Unit: kbar Type: number Example: pressure=10.0</p>	<p><u>reciprocal_lattice_type_orig</u> [2,4,7]</p> <p>Returns the reciprocal lattice centering type for the unrelaxed structure.</p> <p>Unit: none Type: string Example: reciprocal_lattice_type_orig=FCC</p>
<p><u>pressure_residual</u> [12]</p> <p>Returns the hydrostatic pressure, corrected by the Pulay stress, on the simulation cell for the relaxed structure.</p> <p>Unit: kbar Type: number Example: pressure_residual=10.0</p>	<p><u>reciprocal_lattice_variation_type</u> [2,4,7]</p> <p>Returns the reciprocal lattice centering type variation for the relaxed structure.</p> <p>Unit: none Type: string Example: reciprocal_lattice_variation_type=FCC</p>
<p><u>prototype</u> [2,11]</p> <p>Returns the AFLOW prototype for the unrelaxed structure.</p> <p>Unit: none Type: string Example: prototype=T0001.A2BC</p>	<p><u>reciprocal_lattice_variation_type_orig</u> [2,4,7]</p> <p>Returns the reciprocal lattice centering type variation for the unrelaxed structure.</p> <p>Unit: none Type: string Example: reciprocal_lattice_variation_type_orig=FCC</p>
<p><u>Pulay_stress</u> [12,17]</p> <p>Returns the Pulay stress correction for the calculation.</p> <p>Unit: kbar Type: number Example: pulay_stress=10.0</p>	<p><u>reciprocal_volume_cell</u> [2,4,7]</p> <p>Returns the volume of the reciprocal cell for the relaxed structure.</p> <p>Unit: Å⁻³ Type: number Example: reciprocal_volume_cell=0.4733</p>
<p><u>PV_atom</u> [11]</p> <p>Returns the pressure multiplied by volume per atom for the relaxed structure.</p> <p>Unit: eV/atom Type: number Example: PV_atom=12.13</p>	<p><u>reciprocal_volume_cell_orig</u> [2,4,7]</p> <p>Returns the volume of the reciprocal cell for the unrelaxed structure.</p> <p>Unit: Å⁻³ Type: number Example: reciprocal_volume_cell_orig=0.4733</p>
<p><u>PV_cell</u> [11]</p> <p>Returns the pressure multiplied by volume per atom for the relaxed structure.</p> <p>Unit: eV/cell Type: number Example: PV_cell=12.13</p>	<p><u>scintillation_attenuation_length</u> [11,30,31]</p> <p>Returns the scintillation attenuation length.</p> <p>Unit: cm Type: number Example: scintillation_attenuation_length=2.21895</p>
<p><u>reciprocal_geometry_relax</u> [2,4,7]</p> <p>Returns the reciprocal lattice parameters of the relaxed simulation cell.</p> <p>Unit: none Type: numbers Example: reciprocal_geometry_relax=0.8747,0.9747,0.8747,60.0,60.0,60.0</p>	<p><u>sg</u> [2,7,11]</p> <p>Returns the space groups for the structure, before the first relaxation step (unrelaxed), after the first relaxation step and after the last relaxation step (relaxed), using a loose tolerance.</p> <p>Unit: none Type: strings Example: sg=Fm-3m#225,Fm-3m#225,Fm-3m#225</p>
<p><u>reciprocal_geometry_orig</u> [2,4,7]</p> <p>Returns the reciprocal lattice parameters of the unrelaxed simulation cell.</p> <p>Unit: none Type: numbers Example: reciprocal_geometry_orig=0.8747,0.9747,0.8747,60.0,60.0,60.0</p>	<p><u>sg2</u> [2,7,11]</p> <p>Returns the space groups for the structure, before the first relaxation step (unrelaxed), after the first relaxation step and after the last relaxation step (relaxed), using a tight (default) tolerance.</p> <p>Unit: none Type: strings Example: sg2=Fm-3m#225,Fm-3m#225,Fm-3m#225</p>

<p><u>spacegroup_orig</u> [2,7,11]</p> <p>Returns the space group for the unrelaxed structure.</p> <p>Unit: none Type: number Example: spacegroup_orig=225</p>	<p><u>stoichiometry</u> [11]</p> <p>Returns the normalized composition of the structure.</p> <p>Unit: none Type: numbers Example: stoichiometry=0.5,0.25,0.25</p>
<p><u>spacegroup_relax</u> [2,7,11]</p> <p>Returns the space group number for the relaxed structure.</p> <p>Unit: none Type: number Example: spacegroup_relax=225</p>	<p><u>stress_tensor</u> [12]</p> <p>Returns the stress tensor for the relaxed structure.</p> <p>Unit: kbar Type: numbers Example: stress_tensor=-0.96,-0,-0,-0,-0.96,-0,-0,-0,-0.96</p>
<p><u>species</u> [2,11]</p> <p>Returns the unique species.</p> <p>Unit: none Type: strings Example: species=Y,Zn,Zr</p>	<p><u>valence_cell_iupac</u> [11]</p> <p>Returns the sum of the valence electrons, based on IUPAC standards, of the atoms in the simulation cell.</p> <p>Unit: none Type: number Example: valence_cell_iupac=22</p>
<p><u>species_pp</u> [11,17]</p> <p>Returns the pseudopotential of the species.</p> <p>Unit: none Type: strings Example: species_pp=Y,Zn,Zr</p>	<p><u>valence_cell_std</u> [11]</p> <p>Returns the sum of the valence electrons, based on the outermost shell(s), of the atoms in the simulation cell.</p> <p>Unit: none Type: number Example: valence_cell_std=22</p>
<p><u>species_pp_version</u> [11]</p> <p>Returns the pseudopotential version of the species.</p> <p>Unit: none Type: strings Example: species_pp_version=Y,Zn,Zr</p>	<p><u>volume_atom</u> [2,11]</p> <p>Returns the volume per atom of the simulation cell for the relaxed structure.</p> <p>Unit: Å³/atom Type: number Example: volume_atom=100.984</p>
<p><u>species_pp_ZVAL</u> [17,28]</p> <p>Returns the number of valence electrons of the species.</p> <p>Unit: e⁻ Type: numbers Example: species_pp_ZVAL=3</p>	<p><u>volume_cell</u> [2,11]</p> <p>Returns the volume of the simulation cell for the relaxed structure.</p> <p>Unit: Å³ Type: number Example: volume_cell=100.984</p>
<p><u>spin_atom</u> [2,11]</p> <p>Returns the magnetization of the simulation cell per atom.</p> <p>Unit: μ_B/atom Type: number Example: spin_atom=2.16419</p>	<p><u>Wyckoff_letters</u> [7]</p> <p>Returns the Wyckoff letters of each site for the relaxed structure.</p> <p>Unit: none Type: strings Example: Wyckoff_letters=g,c,a</p>
<p><u>spin_cell</u> [2,11]</p> <p>Returns the magnetization of the simulation cell.</p> <p>Unit: μ_B/cell Type: number Example: spin_cell=2.16419</p>	<p><u>Wyckoff_letters_orig</u> [7]</p> <p>Returns the Wyckoff letters of each site for the unrelaxed structure.</p> <p>Unit: none Type: strings Example: Wyckoff_letters_orig=g,c,a</p>
<p><u>spinD</u> [2,11]</p> <p>Returns the magnetic moment on each atom.</p> <p>Unit: μ_B Type: numbers Example: spinD=0.236,0.236,-0.023,1.005</p>	<p><u>Wyckoff_multiplicities</u> [7]</p> <p>Returns the Wyckoff multiplicity of each site for the relaxed structure.</p> <p>Unit: none Type: numbers Example: Wyckoff_multiplicities=24,8,2</p>
<p><u>spinF</u> [2,11]</p> <p>Returns the magnetization of the simulation cell at the Fermi energy.</p> <p>Unit: μ_B/cell Type: number Example: spinF=0.410879</p>	<p><u>Wyckoff_multiplicities_orig</u> [7]</p> <p>Returns the Wyckoff multiplicity of each site for the unrelaxed structure.</p> <p>Unit: none Type: strings Example: Wyckoff_multiplicities_orig=24,8,2</p>

Wyckoff_site_symmetries [7]

Returns the Wyckoff symmetry of each site for the relaxed structure.

Unit: none **Type:** strings**Example:** Wyckoff_site_symmetries=m..., -3., m-3.**Wyckoff_site_symmetries_orig** [7]

Returns the Wyckoff symmetry of each site for the unrelaxed structure.

Unit: none **Type:** strings**Example:** Wyckoff_site_symmetries_orig=m..., -3., m-3.**References**

- [1] C. Oses, M. Esters, S. Divilov, H. Eckert, R. Friedrich, D. Hicks, M.J. Mehl, F. Rose, A. Smolyanyuk, X. Campilongo, C. Toher, S. Curtarolo, *afLOW++: a C++ framework for autonomous materials design*, *Comput. Mater. Sci.* (2022) in press.
- [2] S. Curtarolo, W. Setyawan, S. Wang, J. Xue, K. Yang, R.H. Taylor, L.J. Nelson, G.L.W. Hart, S. Sanvito, M. Buongiorno Nardelli, N. Mingo, O. Levy, *AFLOWLIB.ORG: A distributed materials properties repository from high-throughput ab initio calculations*, *Comput. Mater. Sci.* 58 (2012) 227–235, <http://dx.doi.org/10.1016/j.commatsci.2012.02.002>.
- [3] M.D. Wilkinson, M. Dumontier, I.J.J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L.B. da Silva Santos, P.E. Bourne, J. Bouwman, A.J. Brookes, T. Clark, M. Crosas, I. Dillo, O. Dumon, S. Edmunds, C.T. Evelo, R. Finkers, A. Gonzalez-Beltran, A.J.G. Gray, P. Groth, C. Goble, J.S. Grethe, J. Heringa, P.A.C. 't Hoen, R. Hooft, T. Kuhn, R. Kok, J. Kok, S.J. Lusher, M.E. Martone, A. Mons, A.L. Packer, B. Persson, P. Rocca-Serra, M. Roos, R. van Schaik, S.-A. Sansone, E. Schultes, T. Sengstag, T. Slater, G. Strawn, M.A. Swertz, M. Thompson, J. van der Lei, E. van Mulligen, J. Velterop, A. Waagmeester, P. Wittenburg, K. Wolstencroft, J. Zhao, B. Mons, *The FAIR guiding principles for scientific data management and stewardship*, *Sci. Data* 3 (2016) 160018, <http://dx.doi.org/10.1038/sdata.2016.18>.
- [4] W. Setyawan, S. Curtarolo, *High-throughput electronic band structure calculations: Challenges and tools*, *Comput. Mater. Sci.* 49 (2010) 299–312, <http://dx.doi.org/10.1016/j.commatsci.2010.05.010>.
- [5] S. Curtarolo, W. Setyawan, G.L.W. Hart, M. Jahnátek, R.V. Chepulskii, R.H. Taylor, S. Wang, J. Xue, K. Yang, O. Levy, M.J. Mehl, H.T. Stokes, D.O. Demchenko, D. Morgan, *AFLOW: An automatic framework for high-throughput materials discovery*, *Comput. Mater. Sci.* 58 (2012) 218–226, <http://dx.doi.org/10.1016/j.commatsci.2012.02.005>.
- [6] K. Yang, C. Oses, S. Curtarolo, *Modeling off-stoichiometry materials with a high-throughput Ab-Initio Approach*, *Chem. Mater.* 28 (2016) 6484–6492, <http://dx.doi.org/10.1021/acs.chemmater.6b01449>.
- [7] D. Hicks, C. Oses, E. Gossett, G. Gomez, R.H. Taylor, C. Toher, M.J. Mehl, O. Levy, S. Curtarolo, *AFLOW-SYM: platform for the complete, automatic and self-consistent symmetry analysis of crystals*, *Acta Crystallogr. Sect. A* 74 (2018) 184–203, <http://dx.doi.org/10.1107/S2053273318003066>.
- [8] D. Hicks, C. Toher, D.C. Ford, F. Rose, C. De Santo, O. Levy, M.J. Mehl, S. Curtarolo, *AFLOW-XtalFinder: a reliable choice to identify crystalline prototypes*, *Npj Comput. Mater.* 7 (2021) 30, <http://dx.doi.org/10.1038/s41524-020-00483-4>.
- [9] R. Friedrich, D. Usanmaz, C. Oses, A. Supka, M. Fornari, M. Buongiorno Nardelli, C. Toher, S. Curtarolo, *Coordination corrected ab initio formation enthalpies*, *Npj Comput. Mater.* 5 (2019) 59, <http://dx.doi.org/10.1038/s41524-019-0192-1>.
- [10] R. Friedrich, M. Esters, C. Oses, S. Ki, M.J. Brenner, D. Hicks, M.J. Mehl, C. Toher, S. Curtarolo, *Automated coordination corrected enthalpies with AFLOW-CCE*, *Phys. Rev. Mater.* 5 (2021) 0438031, <http://dx.doi.org/10.1103/PhysRevMaterials.5.043803>.
- [11] R.H. Taylor, F. Rose, C. Toher, O. Levy, K. Yang, M. Buongiorno Nardelli, S. Curtarolo, *A RESTful API for exchanging materials data in the AFLOWLIBorg consortium*, *Comput. Mater. Sci.* 93 (2014) 178–192, <http://dx.doi.org/10.1016/j.commatsci.2014.05.014>.
- [12] F. Rose, C. Toher, E. Gossett, C. Oses, M. Buongiorno Nardelli, M. Fornari, S. Curtarolo, *AFLUX: The LUX materials search API for the AFLOW data repositories*, *Comput. Mater. Sci.* 137 (2017) 362–370, <http://dx.doi.org/10.1016/j.commatsci.2017.04.036>.
- [13] G. Bergerhoff, R. Hundt, R. Sievers, I.D. Brown, *The inorganic crystal structure data base*, *J. Chem. Inf. Comput. Sci.* 23 (1983) 66–69, <http://dx.doi.org/10.1021/ci00038a003>.
- [14] V.L. Karen, M. Hellenbrandt, *Inorganic crystal structure database: new developments*, *Acta Cryst. A* 58 (2002) c367.
- [15] A.I. Liechtenstein, V.I. Anisimov, J. Zaanen, *Density-functional theory and strong interactions: Orbital ordering in Mott-Hubbard insulators*, *Phys. Rev. B* 52 (1995) R5467–R5470, <http://dx.doi.org/10.1103/PhysRevB.52.R5467>.
- [16] S.L. Dudarev, G.A. Botton, S.Y. Savrasov, C.J. Humphreys, A.P. Sutton, *Electron-energy-loss spectra and the structural stability of nickel oxide: An LSDA+U study*, *Phys. Rev. B* 57 (1998) 1505–1509, <http://dx.doi.org/10.1103/PhysRevB.57.1505>.
- [17] C.E. Calderon, J.J. Plata, C. Toher, C. Oses, O. Levy, M. Fornari, A. Natan, M.J. Mehl, G.L.W. Hart, M. Buongiorno Nardelli, S. Curtarolo, *The AFLOW standard for high-throughput materials science calculations*, *Comput. Mater. Sci.* 108 Part A (2015) 233–238, <http://dx.doi.org/10.1016/j.commatsci.2015.07.019>.
- [18] E. Perim, D. Lee, Y. Liu, C. Toher, P. Gong, Y. Li, W.N. Simmons, O. Levy, J.J. Vlassak, J. Schroers, S. Curtarolo, *Spectral descriptors for bulk metallic glasses based on the thermodynamics of competing crystalline phases*, *Nature Commun.* 7 (2016) 12315, <http://dx.doi.org/10.1038/ncomms12315>.
- [19] Y. Lederer, C. Toher, K.S. Vecchio, S. Curtarolo, *The search for high entropy alloys: A high-throughput ab-initio approach*, *Acta Mater.* 159 (2018) 364–383, <http://dx.doi.org/10.1016/j.actamat.2018.07.042>.
- [20] J.P. Perdew, K. Burke, M. Ernzerhof, *Generalized gradient approximation made simple*, *Phys. Rev. Lett.* 77 (1996) 3865–3868, <http://dx.doi.org/10.1103/PhysRevLett.77.3865>.
- [21] C. Toher, C. Oses, J.J. Plata, D. Hicks, F. Rose, O. Levy, M. de Jong, M. Asta, M. Fornari, M. Buongiorno Nardelli, S. Curtarolo, *Combining the AFLOW GIBBS and elastic libraries to efficiently and robustly screen thermomechanical properties of solids*, *Phys. Rev. Mater.* 1 (2017) 015401, <http://dx.doi.org/10.1103/PhysRevMaterials.1.015401>.
- [22] C.W. Andersen, R. Armiento, E. Blokhin, G.J. Conduit, S. Dwaraknath, M.L. Evans, A. Fekete, A. Gopakumar, S. Gražulis, A. Merkys, F. Mohamed, C. Oses, G. Pizzi, G.-M. Rignanese, M. Scheidgen, L. Talirz, C. Toher, D. Winston, R. Aversa, K. Choudhary, P. Colinet, S. Curtarolo, D. Di Stefano, C. Draxl, S. Er, M. Esters, M. Fornari, M. Giantomassi, M. Govoni, G. Hautier, V. Hegde, M.K. Horton, P. Huck, G. Huhs, J. Hummelshøj, A. Karirya, B. Kozinsky, S. Kumbhar, M. Liu, N. Marzari, A.J. Morris, A. Mostofi, K.A. Persson, G. Petretto, T. Purcell, F. Ricci, F. Rose, M. Scheffler, D. Speckhard, M. Uhrin, A. Vaitkus, P. Villars, D. Waroquiers, C. Wolverton, M. Wu, X. Yang, *OPTIMADE: an API for exchanging materials data*, *Sci. Data* 8 (2021) 217, <http://dx.doi.org/10.1038/s41597-021-00974-z>.
- [23] C. Draxl, M. Scheffler, *NOMAD: The FAIR concept for big data-driven materials science*, *MRS Bull.* 43 (2018) 676–682, <http://dx.doi.org/10.1557/mrs.2018.208>.
- [24] A. Jain, S.P. Ong, G. Hautier, W. Chen, W.D. Richards, S. Dacek, S. Cholia, D. Gunter, D. Skinner, G. Ceder, K.A. Persson, *Commentary: The materials project: A materials genome approach to accelerating materials innovation*, *APL Mater.* 1 (2013) 011002, <http://dx.doi.org/10.1063/1.4812323>.
- [25] L. Talirz, S. Kumbhar, E. Passaro, A.V. Yakutovich, V. Granata, F. Gargiulo, M. Borelli, M. Uhrin, S.P. Huber, S. Zoupanos, C.S. Adorf, C.W. Andersen, O. Schütt, C.A. Pignedoli, D. Passerone, J. VandeVondele, T.C. Schulthess, B. Smit, G. Pizzi, N. Marzari, *Materials cloud a platform for open computational science*, *Sci. Data* 7 (2020) 299, <http://dx.doi.org/10.1038/s41597-020-00637-5>.
- [26] J.E. Saal, S. Kirklin, M. Aykol, B. Meredig, C. Wolverton, *Materials design and discovery with high-throughput density functional theory: The open quantum materials database (OQMD)*, *JOM* 65 (2013) 1501–1509, <http://dx.doi.org/10.1007/s11837-013-0755-4>.
- [27] C. Toher, J.J. Plata, O. Levy, M. de Jong, M. Asta, M. Buongiorno Nardelli, S. Curtarolo, *High-throughput computational screening of thermal conductivity, Debye temperature, and Grüneisen parameter using a quasi-harmonic Debye model*, *Phys. Rev. B* 90 (2014) 174107, <http://dx.doi.org/10.1103/PhysRevB.90.174107>.
- [28] W. Chai, C. Oses, R. Anselm, D. Hicks, C. Toher, S. Curtarolo, G. Henkelman, *ABADER: Self-consistent charge density analysis for molecules and crystals*, 2022, in preparation.
- [29] S. Curtarolo, G.L.W. Hart, M. Buongiorno Nardelli, N. Mingo, S. Sanvito, O. Levy, *The high-throughput highway to computational materials design*, *Nature Mater.* 12 (2013) 191–201, <http://dx.doi.org/10.1038/nmat3568>.
- [30] W. Setyawan, R.M. Gaumé, S. Lam, R.S. Feigelson, S. Curtarolo, *High-throughput combinatorial database of electronic band structures for inorganic scintillator materials*, *ACS Comb. Sci.* 13 (2011) 382–390, <http://dx.doi.org/10.1021/co200012w>.
- [31] W. Setyawan, R.M. Gaumé, R.S. Feigelson, S. Curtarolo, *Comparative study of nonproportionality and electronic band structures features in scintillator materials*, *IEEE Trans. Nucl. Sci.* 56 (2009) 2989–2996, <http://dx.doi.org/10.1109/TNS.2009.2027019>.
- [32] *Jmol: an open-source Java viewer for chemical structures in 3D*, <http://www.jmol.org/>.
- [33] R.M. Hanson, *Jmol — a paradigm shift in crystallographic visualization*, *J. Appl. Crystallogr.* 43 (2010) 1250–1260, <http://dx.doi.org/10.1107/S0021889810030256>.
- [34] R.H. Willens, T.H. Geballe, A.C. Gossard, J.P. Maita, A. Menth, Jr. G.W. Hull, R.R. Soden, *Superconductivity of Nb₃Sn*, *Solid State Commun.* 7 (1969) 837–841, [http://dx.doi.org/10.1016/0038-1098\(69\)90773-X](http://dx.doi.org/10.1016/0038-1098(69)90773-X).
- [35] P.P. Ewald, C. Hermann (Eds.), *Strukturbericht 1913-1928 Akademische Verlagsgesellschaft M. B. H. Leipzig, 1931*.

- [36] An archived copy of the final web page can be found via the Wayback Machine at <http://web.archive.org/web/20101222152521/cst-www.nrl.navy.mil/lattice>.
- [37] M.J. Mehl, D. Hicks, C. Toher, O. Levy, R.M. Hanson, G.L.W. Hart, S. Curtarolo, The AFLOW library of crystallographic prototypes: Part 1, *Comput. Mater. Sci.* 136 (2017) S1–S828, <http://dx.doi.org/10.1016/j.commatsci.2017.01.017>.
- [38] D. Hicks, M.J. Mehl, E. Gossett, C. Toher, O. Levy, R.M. Hanson, G.L.W. Hart, S. Curtarolo, The AFLOW library of crystallographic prototypes: Part 2, *Comput. Mater. Sci.* 161 (2019) S1–S1011, <http://dx.doi.org/10.1016/j.commatsci.2018.10.043>.
- [39] D. Hicks, M.J. Mehl, M. Esters, C. Oses, O. Levy, G.L.W. Hart, C. Toher, S. Curtarolo, The AFLOW library of crystallographic prototypes: Part 3, *Comput. Mater. Sci.* 199 (2021) 110450, <http://dx.doi.org/10.1016/j.commatsci.2021.110450>.
- [40] C.J. Smithells, *Metals Reference Book*, Butterworths Scientific, London, 1949.
- [41] C.J. Smithells, *Metals Reference Book*, second ed., Butterworths Scientific, London, 1955.
- [42] W.B. Pearson, *A Handbook of Lattice Spacings and Structures of Metals and Alloys*, No. N.R.C. (4303) in *International Series of Monographs on Metal Physics and Physical Metallurgy*, Pergamon Press, Oxford, London, Edinburgh, New York, Paris, Frankfurt, 1958, 1964 reprint with corrections edn.
- [43] W.B. Pearson, *A Handbook of Lattice Spacings and Structures of Metals and Alloys*, in: *Volume 2, International Series of Monographs on Metal Physics and Physical Metallurgy*, vol. 8, Pergamon Press, Oxford, London, Edinburgh, New York, Toronto, Sydney, Paris, Braunschweig, 1967, N.R.C. No. 8752.
- [44] G. Kresse, J. Hafner, Norm-conserving and ultrasoft pseudopotentials for first-row and transition-elements, *J. Phys.: Condens. Matter* 6 (1994) 8245–8257, <http://dx.doi.org/10.1088/0953-8984/6/40/015>.
- [45] G. Kresse, J. Furthmüller, Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set, *Comput. Mater. Sci.* 6 (1996) 15–50, [http://dx.doi.org/10.1016/0927-0256\(96\)00008-0](http://dx.doi.org/10.1016/0927-0256(96)00008-0).
- [46] G. Kresse, J. Furthmüller, Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set, *Phys. Rev. B* 54 (1996) 11169–11186, <http://dx.doi.org/10.1103/PhysRevB.54.11169>.
- [47] P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, G.L. Chiarotti, M. Cococcioni, I. Dabo, A. Dal Corso, S. de Gironcoli, S. Fabris, G. Fratesi, R. Gebauer, U. Gerstmann, C. Gougoussis, A. Kokalj, M. Lazzeri, L. Martin-Samos, N. Marzari, F. Mauri, R. Mazzarello, S. Paolini, A. Pasquarello, L. Paulatto, C. Sbraccia, S. Scandolo, G. Sclauzero, A.P. Seitsonen, A. Smogunov, P. Umari, R.M. Wentzcovitch, QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials, *J. Phys.: Condens. Matter* 21 (2009) 395502, <http://dx.doi.org/10.1088/0953-8984/21/39/395502>.
- [48] V. Blum, R. Gehrke, F. Hanke, P. Havu, V. Havu, X. Ren, K. Reuter, M. Scheffler, Ab initio molecular simulations with numeric atom-centered orbitals, *Comput. Phys. Comm.* 180 (2009) 2175–2196, <http://dx.doi.org/10.1016/j.cpc.2009.06.022>.
- [49] X. Gonze, J.M. Beuken, R. Caracas, F. Detraux, M. Fuchs, G.M. Rignanese, L. Sindic, M. Verstraete, G. Zerah, F. Jollet, M. Torrent, A. Roy, M. Mikami, P. Ghosez, J.Y. Raty, D.C. Allan, First-principles computation of material properties: the ABINIT software project, *Comput. Mater. Sci.* 25 (2002) 478–492, [http://dx.doi.org/10.1016/S0927-0256\(02\)00325-7](http://dx.doi.org/10.1016/S0927-0256(02)00325-7).
- [50] The ELK code: <http://elk.sourceforge.net/>, 2020, (Accessed 6 January 2021).
- [51] C. Oses, E. Gossett, D. Hicks, F. Rose, M.J. Mehl, E. Perim, I. Takeuchi, S. Sanvito, M. Scheffler, Y. Lederer, O. Levy, C. Toher, S. Curtarolo, AFLOW-CHULL: Cloud-oriented platform for autonomous phase stability analysis, *J. Chem. Inf. Model.* 58 (2018) 2477–2490, <http://dx.doi.org/10.1021/acs.jcim.8b00393>.
- [52] S. Curtarolo, D. Morgan, G. Ceder, Accuracy of ab initio methods in predicting the crystal structures of metals: A review of 80 binary alloys, *CALPHAD* 29 (2005) 163–211, <http://dx.doi.org/10.1016/j.calphad.2005.01.002>.
- [53] G.L.W. Hart, S. Curtarolo, T.B. Massalski, O. Levy, Comprehensive search for new phases and compounds in binary alloy systems based on platinum-group metals, using a computational first-principles approach, *Phys. Rev. X* 3 (2013) 041035, <http://dx.doi.org/10.1103/PhysRevX.3.041035>.
- [54] R.H. Taylor, S. Curtarolo, G.L.W. Hart, Guiding the experimental discovery of magnesium alloys, *Phys. Rev. B* 84 (2011) 084101, <http://dx.doi.org/10.1103/PhysRevB.84.084101>.
- [55] S. Sanvito, C. Oses, J. Xue, A. Tiwari, M. Žic, T. Archer, P. Tozman, M. Venkatesan, J.M.D. Coey, S. Curtarolo, Accelerated discovery of new magnets in the Heusler alloy family, *Sci. Adv.* 3 (2017) e1602241, <http://dx.doi.org/10.1126/sciadv.1602241>.
- [56] C. Toher, C. Oses, D. Hicks, S. Curtarolo, Unavoidable disorder and entropy in multi-component systems, *Npj Comput. Mater.* 5 (2019) 69, <http://dx.doi.org/10.1038/s41524-019-0206-z>.
- [57] O. Isayev, C. Oses, C. Toher, E. Gossett, S. Curtarolo, A. Tropsha, Universal fragment descriptors for predicting properties of inorganic crystals, *Nature Commun.* 8 (2017) 15679, <http://dx.doi.org/10.1038/ncomms15679>.
- [58] F. Legrain, J. Carrete, A. van Roekeghem, S. Curtarolo, N. Mingo, How chemical composition alone can predict vibrational free energies and entropies of solids, *Chem. Mater.* 29 (2017) 6220–6227, <http://dx.doi.org/10.1021/acs.chemmater.7b00789>.
- [59] V. Stanev, C. Oses, A.G. Kusne, E. Rodriguez, J. Paglione, S. Curtarolo, I. Takeuchi, Machine learning modeling of superconducting critical temperature, *Npj Comput. Mater.* 4 (2018) 29, <http://dx.doi.org/10.1038/s41524-018-0085-8>.
- [60] S.R. Hall, F.H. Allen, I.D. Brown, The crystallographic information file (CIF): a new standard archive file for crystallography, *Acta Crystallogr. Sect. A* 47 (1991) 655–685, <http://dx.doi.org/10.1107/S010876739101067X>.
- [61] E. Gossett, C. Toher, C. Oses, O. Isayev, F. Legrain, F. Rose, E. Zurek, J. Carrete, N. Mingo, A. Tropsha, S. Curtarolo, AFLOW-ML: A RESTful API for machine-learning predictions of materials properties, *Comput. Mater. Sci.* 152 (2018) 134–145, <http://dx.doi.org/10.1016/j.commatsci.2018.03.075>.