Computational Materials Science 93 (2014) 178–192

Contents lists available at ScienceDirect

Computational Materials Science

journal homepage: www.elsevier.com/locate/commatsci

A RESTful API for exchanging materials data in the AFLOWLIB.org consortium

Richard H. Taylor [a,b], Frisco Rose [b], Cormac Toher [b], Ohad Levy [b,1], Kesong Yang [c], Marco Buongiorno Nardelli [d,e], Stefano Curtarolo [f,*]

[a] National Institute of Standards and Technology, Gaithersburg, MD 20878, USA
[b] Department of Mechanical Engineering and Materials Science, Duke University, Durham, NC 27708, USA
[c] Department of Nanoengineering, University of California, San Diego, La Jolla, CA 92093, USA
[d] Department of Physics, University of North Texas, Denton, TX, USA
[e] Department of Chemistry, University of North Texas, Denton, TX, USA
[f] Materials Science, Electrical Engineering, Physics and Chemistry, Duke University, Durham, NC 27708, USA

A R T I C L E   I N F O

Article history:
Received 20 March 2014
Received in revised form 5 May 2014
Accepted 10 May 2014
Available online 24 July 2014

Keywords:
High-throughput
Combinatorial materials science
Computer simulations
Materials databases
AFLOWLIB

A B S T R A C T

The continued advancement of science depends on shared and reproducible data. In the field of computational materials science and rational materials design this entails the construction of large open databases of materials properties. To this end, an Application Program Interface (API) following REST principles is introduced for the AFLOWLIB.org materials data repositories consortium. AUIDs (Aflowlib Unique IDentifier) and AURLs (Aflowlib Uniform Resource Locator) are assigned to the database resources according to a well-defined protocol described herein, which enables the client to access, through appropriate queries, the desired data for post-processing. This introduces a new level of openness into the AFLOWLIB repository, allowing the community to construct high-level work-flows and tools exploiting its rich data set of calculated structural, thermodynamic, and electronic properties. Furthermore, federating these tools will open the door to collaborative investigations of unprecedented scope that will dramatically accelerate the advancement of computational materials design and development.

1. Introduction

Data-driven materials science has gained considerable traction over the last decade or so. This is due to the confluence of three key factors: (1) Improved computational methods and tools; (2) greater computational power; and (3) heightened awareness of the power of extensive databases in science [1]. The recent Materials Genome Initiative (MGI) [1,2] reflects the recognition that many important social and economic challenges of the 21st century could be solved or mitigated by advanced materials. Computational materials science currently presents the most promising path to the resolution of these challenges.

The first and second factors above are epitomized by *high-throughput* computation of materials properties by *ab initio* methods, which is the foundation of an effective approach to materials design and discovery [3–12]. Recently, the software used to manage the calculation work-flow and perform the analyses have trended toward more public and user-friendly frameworks. The emphasis is increasingly on portability and sharing of tools and data [13–15]. Similar to the effort presented here, the *Materials-Project* [16] has been providing open access to its database of computed materials properties through a RESTful API and a *python* library enabling ad hoc applications [17]. Other examples of online material properties databases include that being implemented by the Engineering Virtual Organization for Cyber Design (EVOCD) [18], which contains a repository of experimental data, materials constants and computational tools for use in Integrated Computational Material Engineering (ICME). The future advance of computational materials science would rely on interoperable and federatable tools and databases as much as on the quantities and types of data being produced.

A principle of high-throughput materials science is that one does not know *a priori* where the value of the data lies for any specific application. Trends and insights are deduced *a posteriori*. This requires efficient interfaces to interrogate available data on various levels. We have developed a simple WEB-based API to greatly improve the accessibility and utility of the AFLOWLIB database [14] to the scientific community. Through it, the client can access

* Corresponding author.
E-mail address: stefano@duke.edu (S. Curtarolo).
1 On leave from the Physics Department, NRCN, Israel.

http://dx.doi.org/10.1016/j.commatsci.2014.05.014
0927-0256/© 2014 The Authors. Published by Elsevier B.V.

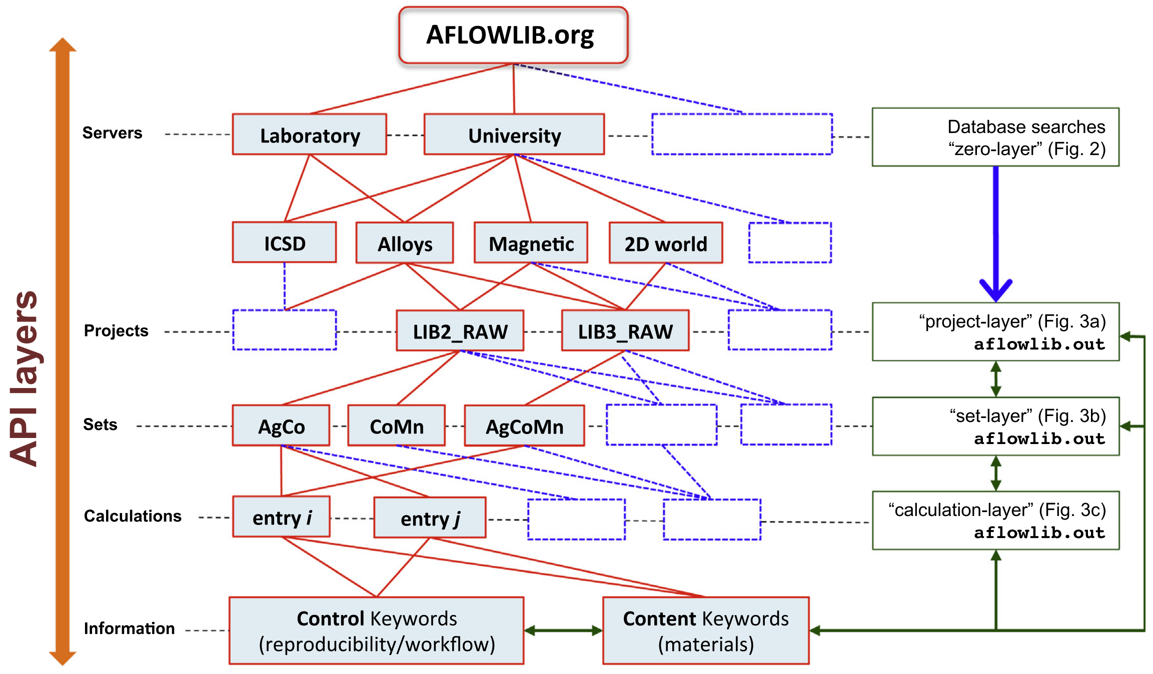

AFLOWLIB.org
API layers
Servers
Laboratory
University
Database searches "zero-layer" (Fig. 2)
ICSD
Alloys
Magnetic
2D world
Projects
LIB2_RAW
LIB3_RAW
"project-layer" (Fig. 3a) aflowlib.out
Sets
AgCo
CoMn
AgCoMn
"set-layer" (Fig. 3b) aflowlib.out
Calculations
entry i
entry j
"calculation-layer" (Fig. 3c) aflowlib.out
Information
Control Keywords (reproducibility/workflow)
Content Keywords (materials)

Fig. 1. Schematic structure of the AFLOWLIB consortium.

calculated physical properties (thermodynamic, crystallographic, or mechanical properties), as well as simulation provenance and runtime properties of the included systems. The data may be used directly (e.g., to browse a class of materials with a desired property) or integrated into higher level work-flows. The interface also allows for the sharing of updates of data used in previous published works, e.g., previously calculated alloy phase diagrams [19–31], thus the database can be expanded systematically.

The rest of this paper is organized as follows. The AFLOWLIB libraries are presented in Section 2. A new materials identifier constructed to navigate the libraries is introduced in Section 3. The data provenance and access format schemes are explained in Sections 4 and 5. The access syntax and its various options are described in Section 6. A few examples for the use of the API and two computer scripts are given in Section 7. The strategy for updates is mentioned in Section 8. A brief conclusion is included in Section 9.

2. The AFLOWLIB libraries multi-layered structure

At its core, AFLOWLIB consists of a coordinated set of libraries of first-principles data describing thermodynamic, structural, and other materials properties of alloy systems (Fig. 1). From the top, it is administered through a large SQL database [32] organized in *layers* (reminiscent of other more developed computer interfaces, i.e., IEEE-POSIX [33]). Each layer consists of searchable entries called `aflowlib.out`. The SQL interface is called the *zero-layer* (Fig. 2) because of its structural immanence.

In this *layered* organization, each `aflowlib.out` can be the child of an `aflowlib.out`-parent or the parent of an `aflowlib.out`-child. Each `aflowlib.out` is identified by a name and an address. The first is the AUID (Aflowlib Unique IDentifier – `$auid`), while the second is the AURL (Aflowlib Uniform Resource Locator – `$aurl`). The structure is summarized in Fig. 1.

The current implementation of AFLOWLIB includes three layers that can be navigated using control keywords and absolute paths.

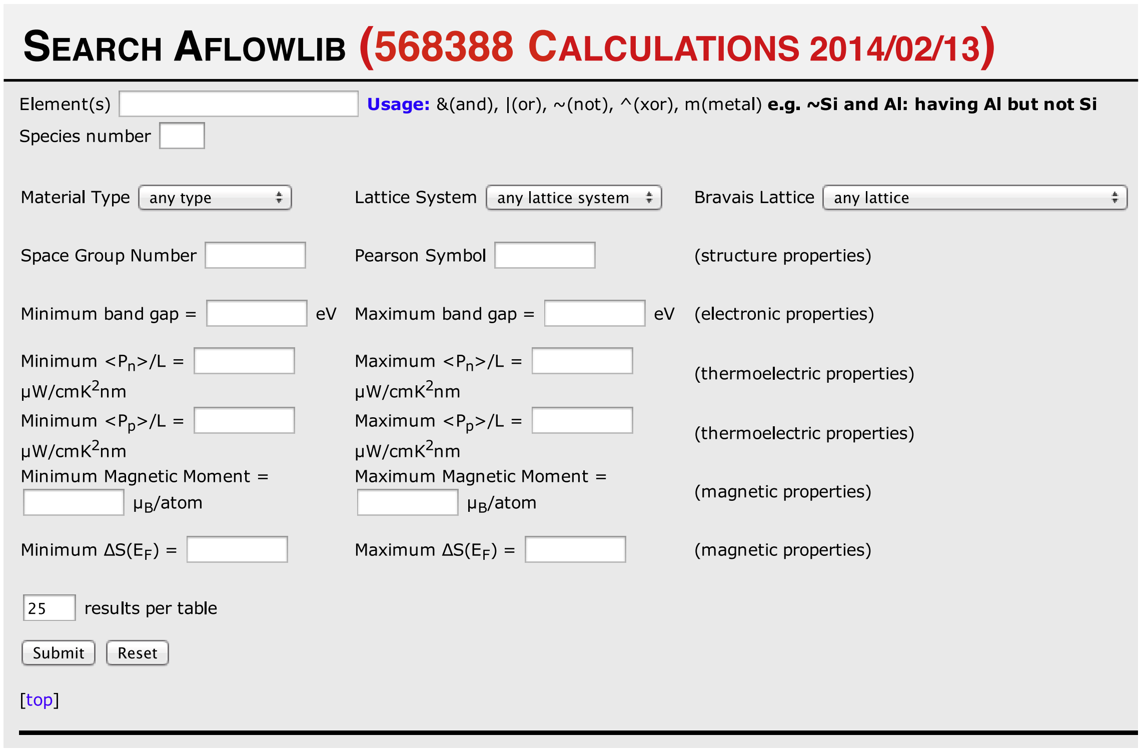
- *Project-layer.* This layer contains information about the project to which the data belongs. For example, for searches in alloys, the project's AURL could be of the type `server:AFLOWDATA/` followed by `LIB1_RAW`, `LIB1_LIB`, `LIB2_RAW`, `LIB2_LIB`, `LIB3_RAW`, or `LIB3_LIB` corresponding to the single element, binary and ternary libraries of post- or pre-processed data respectively. For HTTP access [34], the AURL would be translated from `$aurl=server:/directory/` into `$web=http://server/directory/`. Other translations are considered for future implementations. An example of `aflowlib.out` for the *project-layer* is depicted in Fig. 3(a) (with `server=aflowlib.duke.edu`).
- *Set-layer.* This layer contains information about one or more systems calculated in one or more different configurations (e.g., various structural prototypes, different unit cells required for phonons calculations using the finite difference method, etc.). An example of `aflowlib.out` for the *set-layer* is depicted in Fig. 3(b). To facilitate reproducibility, species making up the `$aurl` might include a subscript or postfix indicating the pseudopotential type. For example, in searches in this layer where the user is interested in the Ag-Ti system, the AURL could be of the type `$aurl=server:AFLOWDATA/LIB2_RAW/AgTi_sv/`. Here the "`_sv`" in `Ti_sv` indicates the "sv" type of pseudopotential in the quantum code used for the calculation, in this case VASP [35]. Other identifiers may be used to indicate pseudopotentials used in, for example, Quantum Espresso (QE) [36] or for potentials coming from other ultrasoft pseudopotential libraries [37].

- *Calculation-layer.* This layer contains information about one system calculated in one particular configuration (e.g. AgTi in configuration `$prototype=66` ($C11_b$ [38,39])). For entries in this layer, the AURL could be of the type `$aurl=server:AFLOWDATA/LIB2_RAW/AgTi_sv/66/`. This `$aurl` points to an `aflowlib.out` describing this specific structure properties. An example of `aflowlib.out` for the *calculation-layer* is depicted in Fig. 3(c). The calculated geometry (here "`66`") is one of the strings comprising the AFLOW prototypes' database. The prototypes' database acts as a look-up table to common *Strukturbericht* designations [38,39] and may also refer to geometries included in the Inorganic Crystal Structure Database (ICSD) [40–42]. In the latter case, a legitimate entry would be something like `.../AgTi_sv/ICSD_58369.AB/`, where the structure composed of Ag and Ti is calculated in the configuration defined by `prototype=ICSD_58369`, and the postfix `.AB` indicates the species ordering: e.g. `.AB` or `.BA` for Ag and Ti in `AB` or `BA` positions, respectively. For ternaries the postfix is a combination of `A,B,C`, where indices can be repeated to create binaries (e.g. `AAB`). Furthermore, the non-element `X` can be included to indicate vacancies in sublattices (e.g. from Heusler to half-Heusler systems [43]). Valid `$prototype` choices also include strings of the type "`f123`" following the enumeration scheme of Hart and Forcade [44,45]. Here the characters `f`, `b`, `h`, `s` indicate fcc, bcc, hcp, and simple cubic structures respectively. The number that follows designates the enumerated prototype. The complete list of structure designations can be accessed with the command "aflow --protos" or by consulting the online links. The options are illustrated in the AFLOW manual [13].

It is important to note that while the operations of concatenating `strings` to `$aurl` are reminiscent of a UNIX computer filesystem, the final `$aurl` is not necessarily served as a directory or a file. In fact, a *computer daemon* (process running in background) dynamically serves multiple file-system directories for the same HTTP-translated `$aurl`. The implementation was so chosen in order to better use the *calculation-layer* data belonging to different *project-layers* and to allow an `$aurl` to contain multiple servers: e.g. `$aurl=server1,server2:somewhere1/AgTi_sv/,somewhere2/AgTi_sv/`. In future updates of the AFLOWLIB.org API, the latter collection will allow users to download AgTi's data from `server1` and `server2` concurrently and transparently.

3. Materials object identifier: Aflowlib Unique IDentifier

Following the spirit of the Digital Object Identifier (DOI) [46] every entry in our database is identified by an alpha-numeric string called an AUID (Aflowlib Unique IDentifier – `$auid`). In future publications, we will use the AUID as a permanent string to the correct AURL, which might get relocated across servers. By giving a short AUID, the user can identify (1) WEB locations for the deliverables, (2) appropriate points of contact (name, emails or other means of identifying corresponding authors), and (3) copyright and publication date. We have chosen this linked approach so that future databases and expansions of the current ones (e.g., locations, servers, personnel, etc.) will not affect the retrieval of the objects. A simple WEB-FORM will be offered to the community to insert AUIDs and to retrieve the information. The AUID is constructed from a 64-bit CRC checksum (cyclic redundancy check [47]) of concatenated input and output files. For example, the bcc Ag–Ti structure `66` described above would be accessed using the AUID "`$auid=aflow:448e178e19e9e973`". The negligible probability of duplicates with this checksum length, and the opportunity of adding a "void character" to the input files (a grain of salt) guarantee the uniqueness of the AUID and its expandibility for many years to come. We welcome other groups to set up versions of AUIDs and to communicate with us the



SEARCH AFLOWLIB (568388 CALCULATIONS 2014/02/13)

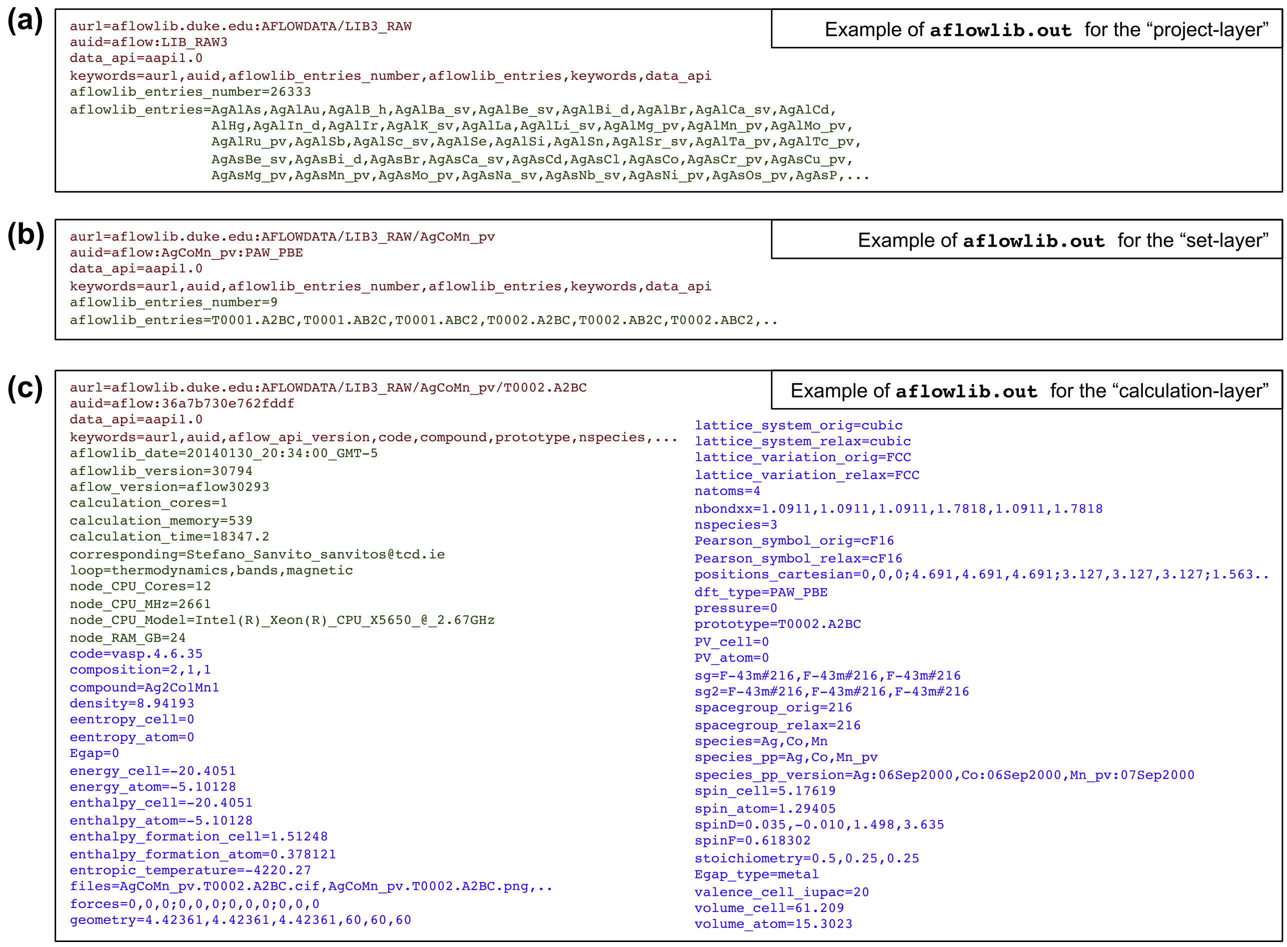
Element(s)	**Usage:** &(and), |(or), ~(not), ^(xor), m(metal) **e.g. ~Si and Al: having Al but not Si**	
Species number		
Material Type: any type	Lattice System: any lattice system	Bravais Lattice: any lattice
Space Group Number	Pearson Symbol	(structure properties)
Minimum band gap = eV	Maximum band gap = eV	(electronic properties)
Minimum <P$_n$>/L = μW/cmK^{2}nm	Maximum <P$_n$>/L = μW/cmK^{2}nm	(thermoelectric properties)
Minimum <P$_p$>/L = μW/cmK^{2}nm	Maximum <P$_p$>/L = μW/cmK^{2}nm	(thermoelectric properties)
Minimum Magnetic Moment = μ$_B$/atom	Maximum Magnetic Moment = μ$_B$/atom	(magnetic properties)
Minimum ΔS(E$_F$) =	Maximum ΔS(E$_F$) =	(magnetic properties)

25 results per table

Submit Reset

[top]

Fig. 2. Project WEB-interface search form: *zero-layer.*



(a) Example of **aflowlib.out** for the "project-layer"

```
aurl=aflowlib.duke.edu:AFLOWDATA/LIB3_RAW
auid=aflow:LIB_RAW3
data_api=aapi1.0
keywords=aurl,auid,aflowlib_entries_number,aflowlib_entries,keywords,data_api
aflowlib_entries_number=26333
aflowlib_entries=AgAlAs,AgAlAu,AgAlB_h,AgAlBa_sv,AgAlBe_sv,AgAlBi_d,AgAlBr,AgAlCa_sv,AgAlCd,
                 AlHg,AgAlIn_d,AgAlIr,AgAlK_sv,AgAlLa,AgAlLi_sv,AgAlMg_pv,AgAlMn_pv,AgAlMo_pv,
                 AgAlRu_pv,AgAlSb,AgAlSc_sv,AgAlSe,AgAlSi,AgAlSn,AgAlSr_sv,AgAlTa_pv,AgAlTc_pv,
                 AgAsBe_sv,AgAsBi_d,AgAsBr,AgAsCa_sv,AgAsCd,AgAsCl,AgAsCo,AgAsCr_pv,AgAsCu_pv,
                 AgAsMg_pv,AgAsMn_pv,AgAsMo_pv,AgAsNa_sv,AgAsNb_sv,AgAsNi_pv,AgAsOs_pv,AgAsP,...
```

(b) Example of **aflowlib.out** for the "set-layer"

```
aurl=aflowlib.duke.edu:AFLOWDATA/LIB3_RAW/AgCoMn_pv
auid=aflow:AgCoMn_pv:PAW_PBE
data_api=aapi1.0
keywords=aurl,auid,aflowlib_entries_number,aflowlib_entries,keywords,data_api
aflowlib_entries_number=9
aflowlib_entries=T0001.A2BC,T0001.AB2C,T0001.ABC2,T0002.A2BC,T0002.AB2C,T0002.ABC2,..
```

(c) Example of **aflowlib.out** for the "calculation-layer"

```
aurl=aflowlib.duke.edu:AFLOWDATA/LIB3_RAW/AgCoMn_pv/T0002.A2BC
auid=aflow:36a7b730e762fddf
data_api=aapi1.0
keywords=aurl,auid,aflow_api_version,code,compound,prototype,nspecies,...
aflowlib_date=20140130_20:34:00_GMT-5
aflowlib_version=30794
aflow_version=aflow30293
calculation_cores=1
calculation_memory=539
calculation_time=18347.2
corresponding=Stefano_Sanvito_sanvitos@tcd.ie
loop=thermodynamics,bands,magnetic
node_CPU_Cores=12
node_CPU_MHz=2661
node_CPU_Model=Intel(R)_Xeon(R)_CPU_X5650_@_2.67GHz
node_RAM_GB=24
code=vasp.4.6.35
composition=2,1,1
compound=Ag2Co1Mn1
density=8.94193
eentropy_cell=0
eentropy_atom=0
Egap=0
energy_cell=-20.4051
energy_atom=-5.10128
enthalpy_cell=-20.4051
enthalpy_atom=-5.10128
enthalpy_formation_cell=1.51248
enthalpy_formation_atom=0.378121
entropic_temperature=-4220.27
files=AgCoMn_pv.T0002.A2BC.cif,AgCoMn_pv.T0002.A2BC.png,..
forces=0,0,0;0,0,0;0,0,0;0,0,0
geometry=4.42361,4.42361,4.42361,60,60,60
```

```
lattice_system_orig=cubic
lattice_system_relax=cubic
lattice_variation_orig=FCC
lattice_variation_relax=FCC
natoms=4
nbondxx=1.0911,1.0911,1.0911,1.7818,1.0911,1.7818
nspecies=3
Pearson_symbol_orig=cF16
Pearson_symbol_relax=cF16
positions_cartesian=0,0,0;4.691,4.691,4.691;3.127,3.127,3.127;1.563..
dft_type=PAW_PBE
pressure=0
prototype=T0002.A2BC
PV_cell=0
PV_atom=0
sg=F-43m#216,F-43m#216,F-43m#216
sg2=F-43m#216,F-43m#216,F-43m#216
spacegroup_orig=216
spacegroup_relax=216
species=Ag,Co,Mn
species_pp=Ag,Co,Mn_pv
species_pp_version=Ag:06Sep2000,Co:06Sep2000,Mn_pv:07Sep2000
spin_cell=5.17619
spin_atom=1.29405
spinD=0.035,-0.010,1.498,3.635
spinF=0.618302
stoichiometry=0.5,0.25,0.25
Egap_type=metal
valence_cell_iupac=20
volume_cell=61.209
volume_atom=15.3023
```

Fig. 3. Examples of the contents defining "aflowlib.out" entries for the *project-layer* (panel a), for the *set-layer* (panel b), and for the *calculation-layer* (panel c). As per AFLOWLIB REST-API version 1.0 (this document), mandatory keywords are listed in red, optional control keywords are in green, and optional materials keywords are in blue. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

necessary identification data so that our consortium can offer WEB links to entries beyond the ones prepared by its current members.

4. Data provenance

Reproducibility is a key tenet of the scientific method, but replication occurs rarely. As experiments become more complex and costly this problem will only worsen. A particular advantage of simulated experiments is that they are uniquely suited for reproducibility. Imagine a scenario in which the original experimenter publishes his/her executable workflow with the necessary input data. Then, anyone wishing to validate or build upon those results does so with a single command on a system with access to the appropriate software. This could be done so that all the software tools, input and output data are maintained remotely, lowering cost, improving ecological sustainability (saving electricity) and increasing collaboration.

In reality, such a framework is not yet available and its implementation is not so straightforward. Despite the potential advantages of computational science in the realm of reproducibility, the crucial data provenance is often completely neglected. An overall culturally-driven disincentive to the actual reproduction of experiments is pervasive. Fortunately there are indications that this problem is being recognized and addressed [2].

AFLOWLIB data is reproducibly structured—the workflow and input parameters are defined by the AFLOW software and a single master input file `aflow.in`, leading to easily reproducible input parameters. In the case of Density Functional Theory-based simulations, this includes essential calculation parameters such as the **k**-mesh density, the energy cut-off, the exchange correlation potential, the *ab initio* software used, and the geometry of the structures. Combining the parameters contained in this file with other provenance related data (calculation time, memory, code, etc.), the AFLOW-API provides *curated and reproducible data*.

5. Data formatting, ownership, and a federatable framework

Data access can be obtained at any level through the API with the appropriate AURL strings, currently translated into WEB inquiries. WEB forms such as the one shown in Fig. 2 allow the user to search within a project for data fitting specified criteria. Alternatively, access through the API is supported by several data formats: "HTML" [34], "JSON" [48], "DUMP", "PHP" [49], "TEXT", and "NONE".

"HTML" (default): $aurl/?format=html.
"JSON" (javascript syntax): $aurl/?format=json.
"DUMP" (php constructs): $aurl/?format=dump.
"PHP" (valid php syntax): $aurl/?format=php.
"TEXT" (no syntax): $aurl/?format=text.
"NONE" (no output): $aurl/?format=none.

The format option is intended for use on the level of the entry returning the whole `aflowlib.out`. For any given property in the set of keywords contained in the entry, the mode is currently to return a simple byte sequence with no formatting. Attempting to format a single property returns the full property set in the specified format. For example, the AURL `$aurl/?density` returns the density of the specified structure. More specifically, for `$aurl=aflowlib.duke.edu:AFLOWDATA/LIB2_RAW/AgTi_sv/66/?density` it returns "6.54346". However, `.../AgTi_sv/66/?density&format=html` is equivalent to `.../AgTi_sv/66/?format=html`, returning the full set of entry properties. "HTML" is primarily for interactive use where keywords and files can be promptly explored with web browsers; "JSON" and "PHP" are valid language syntaxes to facilitate the data access by programmed codes; "DUMP" allows the user to access in his/her own method; "TEXT" returns the entry in a single line with the keywords separated by "|" so that other databases can be built on top of AFLOWLIB.org; and "NONE" may be useful as a method to test the existence of an AURL or for debug purposes.

Clear attribution of contributed data is essential for the development of distributed databases comprising inputs from a wide network of contributors. AFLOW facilitates attribution with the AUID, a unique and persistent identifier, that includes the author, laboratory, group, and affiliation as data entry fields. The shared content in the database is simple to reduce or augment according to a contributor's preference and the attribution is ensured by the unique identifier and contributor labels that are accessible with the AFLOWLIB-API.

The structure of the AFLOWLIB database is *federated*: Autonomous members of the consortium (with distinct geographical locations and affiliations) are able to transparently contribute to a composite database, preserving ownership and claim over the substance of their data. The underlying meta-data schema of the contributed data are consistent by production, to ensure the clarity and searchability of the composite database. A contributor to the consortium begins by downloading the latest version of the AFLOW binary (as of writing this paper this is version 30825) and interfacing with a quantum code. AFLOW is currently configured to run VASP automatically. Pre- and post-processing is functional for both VASP and Quantum Espresso so that agnostic standardization of inputs and outputs between the two codes can be obtained.

The layered and reentrant structure of the AFLOWLIB API allows the manipulation of data from different sources and databases, e.g. the *Materials Project* [16,17]. In order to facilitate this future extension, special keywords are introduced here to identify the source (`$aurl/?data_source`) and the translated syntax of the information (`$aurl/?data_language`). We foresee a global common interface where users can approach heterogeneous data and applications to leverage the efforts of different consortia. Note that in this scenario, due diligence is required to recognize the authorship of the original work, and not merely the serving database.

6. Table of properties and API keywords

This section includes the keywords currently present in the database: description, type, inclusion policy and the AFLOWLIB syntax for retrieval. The list is divided into mandatory, optional control and optional materials keywords. The mandatory keywords must be present in every entry at all layers of the database. Some of the optional control keywords appear at the *projects* and *systems* levels while others appear at the calculations level. The optional materials keywords usually appear just at the *calculations layer*, and not all of them are present in all of the entries. Each entry begins with the AURL and AUID keywords, denoted by the syntax words `$aurl` and `$auid`, respectively.

6.1. Mandatory keywords

- `/` or `/?format=html`, `json`, `dump`, `php`, or `text`.
  - *Description.* The whole entry. It can be read in different formats. For the *calculation-layer* it can also be read as `/?aflowlib.out`.
  - *Type.* lines of `strings`.
  - *Example.* See examples of entries at different levels in Fig. 3
  - *Request syntax.* `$aurl`
- `auid`
  - *Description.* "AFLOWLIB Unique Identifier" for the entry, AUID, which can be used as a publishable object identifier, following the spirit of the DOI foundation [46] (see Section 3).
  - *Type.* `string`.
  - *Example.* `auid=aflow:e9c6d914c4b8d9ca`
  - *Request syntax.* `$aurl/?auid`
- `aurl`
  - *Description.* "AFLOWLIB Uniform Resource Locator" returns the AURL of the entry. The web server is separated from the web directory with ":". This tautological keyword, `aurl` returning itself, is useful for debug and hyperlinking purposes.
  - *Type.* `string`.
  - *Example.* `aurl=aflowlib.duke.edu:AFLOWDATA/LIB3_RAW/Bi_dRh_pvTi_sv/T0003.ABC:LDAU2`
  - *Request syntax.* `$aurl/?aurl`
- `data_api`
  - *Description.* "AFLOWLIB" version of the entry, API. This article describes version 1.0 of the REST-API.
  - *Type.* `string`.
  - *Example.* `data_api=aapi1.0`
  - *Request syntax.* `$aurl/?data_api`
- `keywords`
  - *Description.* This includes the list of keywords available in the entry, separated by commas. All of the keywords can be requested to the database. The request `keywords` should be the first one made, so that the reader is made aware of the available keywords.

– *Type.* List of `strings` separated by ",".
– *Example.* `keywords=aurl,auid,loop,code,compound,prototype,nspecies,natoms,...`
– *Request syntax.* `$aurl/?keywords`

6.2. Optional controls keywords (alphabetic order)

- `aflowlib_entries` (`aflowlib_entries_number`)
  – *Description.* For *projects* and *set-layer* entries (see Fig. 1), `aflowlib_entries` lists the available sub-entries which are associated with the `aurl` of the subdirectories. By parsing `$aurl/?aflowlib_entries` (containing `aurl/aflowlib_entries_number` entries) the user finds the further locations to interrogate.
  – *Type.* Set of `strings` separated by "," (`number`).
  – *Example.* `aflowlib_entries=AgAl,AgAs,AgAu,AgB_h,AgBa_sv,AgBe_sv,AgBi_d,AgBr,AgCa_sv,...` (`aflowlib_entries_number=1524`)
  – *Request syntax.* `$aurl/?aflowlib_entries` (`aurl/aflowlib_entries_number`)
- `aflowlib_date` (`aflowlib_version`)
  – *Description.* Returns the date (version) of the AFLOW post-processor which generated the entry for the library. This entry is useful for debugging and regression purposes.
  – *Type.* `string`.
  – *Example.* `aflowlib_date=20140204_13:10:39_GMT-5` (`aflowlib_version=30794`)
  – *Request syntax.* `$aurl/?aflowlib_date` (`$aurl/?aflowlib_version`)
- `aflow_version`
  – *Description.* Returns the version number of AFLOW used to perform the calculation. This entry is useful for debugging and regression purposes.
  – *Type.* `string`.
  – *Example.* `aflow_version=aflow30641`
  – *Request syntax.* `$aurl/?aflow_version`
- `author`
  – *Description.* Returns the name (not necessarily an individual) and affiliation associated with authorship of the data. Multiple entries are separated by commas. Spaces are substituted with "_" to aid parsing.
  – *Type.* List of `strings` separated by ",".
  – *Example.* `author=Marco_Buongiorno_Nardelli,Ohad_Levy,Jesus_Carrete`
  – *Request syntax.* `$aurl/?author`
- `calculation_cores`, `calculation_memory`, `calculation_time`
  – *Description.* Number of processors/cores, maximum memory, total time used for the calculation.
  – *Type.* `number`, `number`, `number`.
  – *Units.* dimensionless, Megabytes, seconds.
  – *Example.* `calculation_cores=32`, `calculation_memory = 8376.13`, `calculation_time=140713`
  – *Request syntax.* `$aurl/?calculation_cores`, `$aurl/?calculation_memory`, `$aurl/?calculation_time`
- `corresponding`
  – *Description.* Returns the name (not necessarily an individual) and affiliation associated with the data origin concerning correspondence about data. Multiple entries are separated by commas. Spaces are substituted with "_" to aid parsing.
  – *Type.* List of `strings` separated by ",".
  – *Example.* `corresponding=M_Buongiorno_Nardelli_mbn@unt.edu`
  – *Request syntax.* `$aurl/?corresponding`
- `data_source`, `data_language`
  – *Description.* As mentioned in the text, the layered structure of AFLOWLIB well adapts to serve and translate data presented in other open databases. If this is the case, the source and language (API) of the data are given with these two keywords. When using non-AFLOWLIB data, due diligence is required to recognize the authorship of the original work, and not the serving database, merely.
  – *Type.* `strings`.
  – *Example.* `data_source=aflowlib data_language=translated`
  – *Request syntax.* `$aurl/?data_source` and `$aurl/?data_language`
- `loop`
  – *Description.* Informs the user of the type of post-processing that was performed.
  – *Type.* List of `strings` separated by ",".
  – *Example.* `loop=thermodynamics,bands,magnetic`
  – *Request syntax.* `$aurl/?loop`
- `node_CPU_Cores`, `node_CPU_MHz`, `node_CPU_Model`, `node_RAM_GB`
  – *Description.* Information about the node/cluster where the calculation was performed. Number of cores, speed, model, and total memory accessible to the calculation.
  – *Type.* `number`, `number`, `string`, `number`.
  – *Units.* MHz for speed, gigabytes for RAM.
  – *Example.* `node_CPU_Cores=12`, `node_CPU_MHz=2661`, `node_RAM_GB=48`, `node_CPU_Model=Intel(R)_Xeon(R)_CPU_X5650_@_2.67GHz`
  – *Request syntax.* `$aurl/?node_CPU_Cores`, `$aurl/?node_CPU_MHz`, `$aurl/?node_CPU_Model`, `$aurl/?node_RAM_GB`

- `sponsor`
  - *Description.* Returns information about funding agencies and other sponsors for the data. Multiple entries are separated by commas. Spaces are substituted with "_" to aid parsing.
  - *Type.* List of `strings` separated by ",".
  - *Example.* `sponsor=DOD_N000141310635,NIST_70NANB12H163`
  - *Request syntax.* `$aurl/?sponsor`

6.3. Optional materials keywords (alphabetic order)

- `Bravais_lattice_orig` (`Bravais_lattice_relax`)
  - *Description.* Returns the Bravais lattice [50] of the original unrelaxed (relaxed) structure before (after) the calculation.
  - *Type.* `string`.
  - *Example.* `Bravais_lattice_orig=MCLC` (`Bravais_lattice_relax=MCLC`)
  - *Request syntax.* `$aurl/?Bravais_lattice_orig` (`$aurl/?Bravais_lattice_relax`)
  - *Tolerance.* Calculations of lattices (Brillouin zones), prototypes, and symmetries (point/factor/space groups) are based on different algorithms and require different sets of tolerances. To guarantee self-consistency of the results, initial tolerances are set to very stringent values (*e.g.*, $10^{-4}\%$ for distances, $10^{-2}\%$ for angles, $10^{-4}\%$ for spectral radii of mapping matrices, etc.) and slowly increased alternatingly (by a factor of 2) until self-consistency is found amongst geometrical descriptors. The final tolerances are usually of the order of $\sim 0.5\%$ for distances and $\sim 1\%$ for angles.
- `code`
  - *Description.* Returns the software name and version used to perform the simulation.
  - *Type.* `string`.
  - *Example.* `code=vasp.4.6.35`
  - *Request syntax.* `$aurl/?code`
- `composition`
  - *Description.* Returns a comma delimited composition description of the structure entry in the calculated cell.
  - *Type.* List of `number` separated by ",".
  - *Example.* `composition=2,6,6`. (For a $A_2B_6C_6$ compound)
  - *Request syntax.* `$aurl/?composition`
- `compound`
  - *Description.* Similar to `composition`. Returns the composition description of the compound in the calculated cell.
  - *Type.* Set of {`string`·`number`}.
  - *Example.* `compound=Co2Er6Si6`
  - *Request syntax.* `$aurl/?compound`
- `density`
  - *Description.* Returns the mass density.
  - *Type.* `number`.
  - *Units.* grams/cm^3.
  - *Example.* `density=7.76665`
  - *Request syntax.* `$aurl/?density`
- `dft_type`
  - *Description.* Returns information about the pseudopotential type, the exchange correlation functional used (normal or hybrid) and use of GW.
  - *Type.* Set of `strings` separated by ",".
  - *Example.* If the calculations were performed with VASP [35], the entry could include "US", "GGA", "PAW_LDA", "PAW_GGA", "PAW_PBE", "GW", "HSE06" (February 2014).
  - *Example.* `dft_type=PAW_PBE,HSE06`
  - *Request syntax.* `$aurl/?dft_type`
- `eentropy_cell` (`eentropy_atom`)
  - *Description.* Returns the electronic entropy of the unit cell used to converge the *ab initio* calculation (smearing).
  - *Type.* `number`.
  - *Units.* Natural units of the `$code`, e.g., eV or Ry (eV/atom or Ry/atom) if the calculations were performed with VASP [35] or QE [36], respectively.
  - *Example.* `eentropy_cell=0.0011` (`eentropy_atom=0.0003`)
  - *Request syntax.* `$aurl/?eentropy_cell` (`$aurl/?eentropy_atom`)
- `Egap`
  - *Description.* Band gap calculated with the approximations and pseudopotentials described by other keywords.
  - *Type.* `number`.
  - *Units.* eV.
  - *Example.* `Egap=2.5`
  - *Request syntax.* `$aurl/?Egap`
- `Egap_fit`
  - *Description.* Simple cross-validated correction (fit) of `Egap`. See Ref. [9] for the definition.
  - *Type.* `number`.
  - *Units.* eV.

  - *Example.* `Egap_fit=3.5`
  - *Request syntax.* `$aurl/?Egap_fit`
- `Egap_type`
  - *Description.* Given a band gap, this keyword describes if the system is a metal, a semi-metal, an insulator with direct or indirect band gap.
  - *Type.* `string`.
  - *Example.* `Egap_type=insulator_direct`
  - *Request syntax.* `$aurl/?Egap_type`
- `energy_cell` (`energy_atom`)
  - *Description.* Returns the total *ab initio* energy of the unit cell E (energy per atom —the value of `energy_cell`/N).
  - *Type.* `number`.
  - *Units.* Natural units of the `$code`, e.g., eV or Ry (eV/atom or Ry/atom) if the calculations were performed with VASP [35] or QE [36], respectively.
  - *Example.* `energy_cell=-82.1656` (`energy_atom=-5.13535`)
  - *Request syntax.* `$aurl/?energy_cell` (`$aurl/?energy_atom`)
- `energy_cutoff`
  - *Description.* Set of energy cut-offs used during the various steps of the calculations.
  - *Type.* Set of `strings` separated by ",".
  - *Units.* Natural units of the `$code`, e.g., eV or Ry (eV/atom or Ry/atom) if the calculations were performed with VASP [35] or QE [36], respectively.
  - *Example.* `energy_cutoff=384.1,384.1,384.1`
  - *Request syntax.* `$aurl/?energy_cutoff`
- `enthalpy_cell` (`enthalpy_atom`)
  - *Description.* Returns the enthalpy of the system of the unit cell $H = E + PV$ (enthalpy per atom —the value of `enthalpy_cell`/N).
  - *Type.* `number`.
  - *Units.* Natural units of the `$code`, e.g., eV or Ry (eV/atom or Ry/atom) if the calculations were performed with VASP [35] or QE [36], respectively.
  - *Example.* `enthalpy_cell=-82.1656` (`enthalpy_atom=-5.13535`)
  - *Request syntax.* `$aurl/?enthalpy_cell` (`$aurl/?enthalpy_atom`)
- `enthalpy_formation_cell` (`enthalpy_formation_atom`)
  - *Description.* Returns the formation enthalpy ΔH_F per unit cell ($\Delta H_{F atomic}$ per atom). For compounds $A_{N_A}B_{N_B}\cdots$ *with* $N_A + N_B \cdots = N$ atoms per cell, this is defined as: $\Delta H_F \equiv E(A_{N_A}B_{N_B}\cdots)$-$[N_A E(A)_{atomic} + N_B E(B)_{atomic} + \cdots]$ (in the `_atom` case with $A_{x_A}B_{x_B}\cdots$ and $x_A + x_B \cdots = 1$ we have $\Delta H_{F atomic} \equiv E(A_{x_A}B_{x_B}\cdots)_{atomic} - [x_A E(A)_{atomic} + x_B E(B)_{atomic} + \cdots]$).
  - *Type.* `number`.
  - *Units.* Natural units of the `$code`, e.g., eV or Ry (eV/atom or Ry/atom) if the calculations were performed with VASP [35] or QE [36], respectively.
  - *Example.* `enthalpy_formation_cell=-33.1587` (`enthalpy_formation_atom=-0.720841`)
  - *Request syntax.* `$aurl/?enthalpy_formation_cell` (`$aurl/?enthalpy_formation_atom`)
- `entropic_temperature`
  - *Description.* Returns the entropic temperature as defined in Ref. [3,7] for the structure. The analysis of formation enthalpy is, by itself, insufficient to compare alloy stability at different concentrations and their resilience toward high-temperature disorder. The formation enthalpy represents the ordering-strength of a mixture $A_{x_A}B_{x_B}C_{x_C}\cdots$ against decomposition into its pure constituents at the appropriate concentrations $x_A, x_B, x_C, \ldots$. (ΔH_F is negative for compound forming systems). However, it does not contain information about its resilience against disorder, which is captured by the entropy of the system. To quantify this resilience we define the *entropic temperature* for each compound as:

$$T_s(A_{x_A}B_{x_B}C_{x_C}\cdots) \equiv \left\{ \frac{\Delta H_F(A_{x_A}B_{x_B}C_{x_C}\cdots)}{k_B[x_A\log(x_A) + x_B\log(x_B) + x_C\log(x_C) + \cdots]} \right\}, \quad (1)$$

    where the sign is chosen so that a positive temperature is needed for competing against compound stability. This definition assumes an ideal scenario [3] where the entropy is $S[\{x_i\}] = -k_B \sum_i x_i \log(x_i)$. T_s is a concentration-maximized formation enthalpy weighted by the inverse of its entropic contribution. Its maximum $T_s = \max_{phases}[T_s(phases)]$ represents the deviation of a system convex-hull from the purely entropic free-energy hull, $-TS(x)$, and hence the ability of its ordered phases to resist the temperature-driven deterioration into a disordered mixture exclusively promoted by configurational-entropy.
  - *Type.* `number`.
  - *Units.* Kelvin.
  - *Example.* `entropic_temperature=1072.1`
  - *Request syntax.* `$aurl/?entropic_temperature`
- `files`
  - *Description.* Provides access to the input and output files used in the simulation (provenance data).
  - *Type.* List of `strings` separated by ",".
  - *Example.* `files=Bi_dRh_pv.33.cif,Bi_dRh_pv.33.png,CONTCAR.relax,CONTCAR.relax1,`
    `DOSCAR.static.bz2,EIGENVAL.bands.bz2,KPOINTS.bands.bz2,aflow.in,edata.bands.out,`
    `edata.orig.out,edata.relax.out,...`
  - *Request syntax.* `$aurl/?files`

  - *Description.* Once the "`files`" list has been parsed, each file can be accessed with `$aurl/file` (note no "`?`" for accessing individual files).
- `forces`
  - *Description.* Final quantum mechanical forces (F_i, F_j, F_k) in the notation of the code.
  - *Type.* Triplets (`number,number,number`) separated by ";" for each atom in the unit cell.
  - *Units.* Natural units of the `$code`, e.g., eV/Å or a.u if the calculations were performed with VASP [35] or QE [36], respectively.
  - *Example.* `forces=0,-0.023928,0.000197;0,0.023928,-0.000197;...`
  - *Request syntax.* `$aurl/?forces`
- `geometry`
  - *Description.* Returns geometrical data describing the unit cell in the usual a, b, c, $\alpha,\ \beta,\ \gamma$ notation $\alpha \equiv \{\widehat{\vec{b},\vec{c}}\},\ \beta \equiv \{\widehat{\vec{c},\vec{a}}\},\ \gamma \equiv \{\widehat{\vec{a},\vec{b}}\}$.
  - *Type.* Sixtuplet (`number,number,number,number,number,number`).
  - *Units.* $a,\ b,\ c$ are the natural units of the `$code`, e.g., Å or a.u. (Bohr) if the calculations were performed with VASP [35] or QE [36], respectively. $\alpha,\ \beta,\ \gamma$ are in degrees.
  - *Example.* `geometry=18.82,18.82,18.82,32.41,32.41,32.41`
  - *Request syntax.* `$aurl/?geometry`
- `lattice_system_orig`, `lattice_variation_orig` (`lattice_system_relax`, `lattice_variation_relax`)
  - *Description.* Return the lattice system [50,51] and lattice variation (Brillouin zone) of the original-unrelaxed (relaxed) structure before (after) the calculation. See Ref. [14,52] for the lattice variation and Brillouin zones notations.
  - *Type.* `string`, `string`.
  - *Example.* `lattice_system_orig=rhombohedral`, `lattice_variation_orig=RHL1` (`lattice_system_relax=monoclinic`, `lattice_variation_relax=MCLC1`)
  - *Request syntax.* `$aurl/?lattice_system_orig`, `$aurl/?lattice_variation_orig` (`$aurl/?lattice_system_relax`, `$aurl/?lattice_variation_relax`)
- `kpoints`
  - *Description.* Set of **k**-point meshes uniquely identifying the various steps of the calculations, e.g. relaxation, static and electronic band structure (specifying the **k**-space symmetry points of the structure).
  - *Type.* Set of `numbers` and `strings` separated by "," and ";".
  - *Example.* `kpoints=10,10,10;16,16,16;G-X-W-K-G-L-U-W-L-K+U-X`
  - *Request syntax.* `$aurl/?kpoints`
- `ldau_TLUJ`
  - *Description.* This vector of numbers contains the parameters of the "DFT+U" calculations, based on a corrective functional inspired by the Hubbard model [53,54]. Standard values in the AFLOWLIB.org library come from Refs. [9,10]. There are four fields (`T`;{`L`};{`U`};{`J`}), separated by ";". The first field indicates the type (`T`) of the DFT+U corrections: type=1, the rotationally invariant version introduced by Liechtenstein et al. [55]; type=2, the simplified rotationally invariant version introduced by Dudarev et al. [56]. The second field indicates the l-quantum number ({`L`}, one number for each species separated by ",") for which the on-site interaction is added (-1 = neglected, $0 = s$, $1 = p$, $2 = d$, $3 = f$). The third field lists the effective on-site Coulomb interaction parameters ({`U`}, one number for each species separated by ","). The fourth field specifies the effective on-site exchange interaction parameters ({`J`}, one number for each species separated by ","). Although more compact, the convention is similar to the VASP notation [35].
  - *Units.* dimensionless; {dimensionless}; {eV}; {eV}.
  - *Type.* `number`;{`number`,⋯};{`number`,⋯};{`number`,⋯}.
  - *Example.* `ldau_TLUJ=2;2,0,0;5,0,0;0,0,0`
  - *Request syntax.* `$aurl/?ldau_TLUJ`
- `natoms`
  - *Description.* Returns the number of atoms in the unit cell of the structure entry. The number can be noninteger if partial occupation is considered within appropriate approximations.
  - *Type.* `number`.
  - *Example.* `natoms=12`
  - *Request syntax.* `$aurl/?natoms`
- `nbondxx`
  - *Description.* Nearest neighbors bond lengths of the relaxed structure per ordered set of species A_i, A_j $(j \geqslant i)$. For pure systems: $\{\rho[AA]\}$; for binaries: $\{\rho[AA], \rho[AB], \rho[BB]\}$; for ternaries: $\{\rho[AA], \rho[AB], \rho[AC], \rho[BB], \rho[BC], \rho[CC]\}$ and so on.
  - *Type.* Set of $N_{species}(N_{species}+1)/2$ `numbers`.
  - *Units.* Natural units of the `$code`, e.g., Å or a.u. (Bohr) if the calculations were performed with VASP [35] or QE [36], respectively.
  - *Example.* `nbondxx=1.2599,1.0911,1.0911,1.7818,1.2599,1.7818` (for a three species entry)
  - *Request syntax.* `$aurl/?nbondxx`
- `nspecies`
  - *Description.* Returns the number of species in the system (e.g., binary = 2, ternary = 3, etc.).
  - *Type.* `number`.
  - *Example.* `nspecies=3`
  - *Request syntax.* `$aurl/?nspecies`
- `Pearson_symbol_orig` (`Pearson_symbol_relax`)
  - *Description.* Returns the Pearson symbol [52,57] of the original-unrelaxed (relaxed) structure before (after) the calculation.
  - *Type.* `string`.
  - *Example.* `Pearson_symbol_orig=mS32` (`Pearson_symbol_relax=mS32`)

  - *Request syntax.* `$aurl/?Pearson_symbol_orig` (`$aurl/?Pearson_symbol_relax`)
  - *Tolerance.* See discussion about tolerances in entry `Bravais_lattice_orig`.
- `positions_cartesian`
  - *Description.* Final Cartesian positions (x_i, x_j, x_k) in the notation of the code.
  - *Type.* Triplets (`number,number,number`) separated by ";" for each atom in the unit cell.
  - *Units.* Natural units of the `$code`, e.g., in Cartesian coordinates (Å) if the calculations were performed with VASP [35].
  - *Example.* `positions_cartesian=0,0,0;18.18438,0,2.85027;...`
  - *Request syntax.* `$aurl/?positions_cartesian`
- `positions_fractional`
  - *Description.* Final fractional positions x_i, x_j, x_k with respect to the unit cell as specified in `geometry`.
  - *Type.* Triplets (`number,number,number`) separated by ";" for each atom in the unit cell.
  - *Units.* dimensionless
  - *Example.* `positions_fractional=0,0,0;0.25,0.25,0.25;...`
  - *Request syntax.* `$aurl/?positions_fractional`
- `pressure`
  - *Description.* Returns the external pressure selected for the simulation.
  - *Type.* `number`.
  - *Units.* Natural units of the `$code`, e.g., kbar or a.u. (Ry/Bohr) if the calculations were performed with VASP [35] or QE [36], respectively.
  - *Example.* `pressure=10.0`
  - *Request syntax.* `$aurl/?pressure`
- `prototype`
  - *Description.* Returns the AFLOW **unrelaxed** prototype which was used for the calculation. The list can be accessed with the command "aflow --protos" or by consulting the online links. The options are illustrated in the AFLOW manual [13]. Note that during the calculation, unstable structures can deform and lead to different relaxed configurations. It is thus **imperative** for the user to make an elaborate analysis of the final structure to pinpoint the right prototype to report. Differences in Bravais lattices, Pearson symbol, space groups, for the `_orig` and `_relax` versions are extremely useful for this task.
  - *Type.* `string`.
  - *Example.* `prototype=T0001.A2BC`
  - *Request syntax.* `$aurl/?prototype`
  - *Tolerance.* See discussion about tolerances in entry `Bravais_lattice_orig`.
- `PV_cell` (`PV_atom`)
  - *Description.* Pressure multiplied by volume of the unit cell (of the atom).
  - *Type.* `number`.
  - *Units.* Natural units of the `$code`, e.g., eV or Ry (eV/atom or Ry/atom) if the calculations were performed with VASP [35] or QE [36], respectively.
  - *Example.* `PV_cell=12.13` (`PV_atom=3.03`)
  - *Request syntax.* `$aurl/?PV_cell` (`$aurl/?PV_atom`)
- `scintillation_attenuation_length`
  - *Description.* Returns the scintillation attenuation length of the compound in cm. See Refs. [9,58].
  - *Type.* `real` number.
  - *Example.* `scintillation_attenuation_length=2.21895`
  - *Request syntax.* `$aurl/?scintillation_attenuation_length`
- `sg` (`sg2`)
  - *Description.* Evolution of the space group of the compound [50,51]. The first, second and third `string` represent space group name/number before the first, after the first, and after the last relaxation of the calculation.
  - *Tolerance.* `sg` values are calculated with 3.0% and 0.5 deg tolerances for lengths and angles, respectively. (`sg2` is with 1.5% and 0.25 deg). Symmetry is cross validated through the internal engines of AFLOW [13], PLATON [59], and FINDSYM [60].
  - *Type.* Triplet `string`, `string`, `string`.
  - *Example.* `sg=Fm-3m#225,Fm-3m#225,Fm-3m#225` (`sg2=R-3c#167,R-3c#167,R-3c#167`)
  - *Request syntax.* `$aurl/?sg` (`$aurl/?sg2`)
- `spacegroup_orig` (`spacegroup_relax`)
  - *Description.* Returns the spacegroup number [50] of the original-unrelaxed (relaxed) structure before (after) the calculation.
  - *Tolerance.* Same as `sg`.
  - *Type.* `number`.
  - *Example.* `spacegroup_orig=225` (`spacegroup_relax=225`)
  - *Request syntax.* `$aurl/?spacegroup_orig` (`$aurl/?spacegroup_relax`)
- `species`, `species_pp`, `species_pp_version`
  - *Description.* Species of the atoms, pseudopotentials species, and pseudopotential versions.
  - *Type.* List of `strings` separated by ",".
  - *Example.* `species=Y,Zn,Zr`, `species_pp=Y_sv,Zn,Zr_sv`, `species_pp_version=Y_sv:PAW_PBE:06Sep2000,Zn:PAW_PBE:06Sep2000,Zr_sv:PAW_PBE:07Sep2000`
  - *Request syntax.* `$aurl/?species`, `$aurl/?species_pp`, `$aurl/?species_pp_version`
- `spin_cell` (`spin_atom`)
  - *Description.* For spin polarized calculations, the total magnetization of the cell (magnetization per atom).
  - *Type.* `number`.

- *Units.* Natural units of the `$code`, e.g., μ_B (Bohr magneton).
- *Example.* `spin_cell=2.16419` (`spin_atom=0.541046`)
- *Request syntax.* `$aurl/?spin_cell` (`aurl/?spin_atom`)

- `spinD`
  - *Description.* For spin polarized calculations, the spin decomposition over the atoms of the cell.
  - *Type.* List of `numbers` separated by ",".
  - *Units.* Natural units of the `$code`, e.g., μ_B (Bohr magneton).
  - *Example.* `spinD=0.236, 0.236, -0.023, 1.005`
  - *Request syntax.* `$aurl/?spinD`
- `spinD_magmom_orig`
  - *Description.* For spin polarized calculations, string containing the values used to initialize the magnetic state for the *ab initio* calculation.
  - *Type.* String containing the instruction passed to the *ab initio* code with spaces substituted by "_".
  - *Units.* Natural units of the `$code`.
  - *Example.* `spinD_magmom_orig=+5_-5_+5_-5`
  - *Request syntax.* `$aurl/?spinD_magmom_orig`
- `spinF`
  - *Description.* For spin polarized calculations, the magnetization of the cell at the Fermi level.
  - *Type.* `number`.
  - *Units.* Natural units of the `$code`, e.g., μ_B (Bohr magneton).
  - *Example.* `spinF=0.410879`
  - *Request syntax.* `$aurl/?spinF`
- `stoichiometry`
  - *Description.* Similar to `composition`, returns a comma delimited stoichiometry description of the structure entry in the calculated cell.
  - *Type.* List of `number` separated by ",".
  - *Example.* `stoichiometry=0.5, 0.25, 0.25`
  - *Request syntax.* `$aurl/?stoichiometry`
- `valence_cell_std` (`valence_cell_iupac`)
  - *Description.* Returns standard valence (IUPAC valence, the maximum number of univalent atoms that may combine with the atoms [61]).
  - *Type.* `number`.
  - *Example.* `valence_cell_std=22` (`valence_cell_iupac=12`)
  - *Request syntax.* `$aurl/?valence_cell_std` (`$aurl/?valence_cell_iupac`)
- `volume_cell` (`volume_atom`)
  - *Description.* Returns the volume of the unit cell (per atom in the unit cell).
  - *Type.* `number`.
  - *Units.* Natural units of the `$code`, e.g., Å^3 or Bohr^3 (Å^3/atom or Bohr^3/atom) if the calculations were performed with VASP [35] or QE [36], respectively.
  - *Example.* `volume_cell=100.984` (`volume_atom=25.2461`)
  - *Request syntax.* `$aurl/?volume_cell` (`$aurl/?volume_atom`)

7. Examples

7.1. Generating a free-energy zero-temperature convex hull: OsTc

In this example we introduce the steps to generate a binary free-energy convex hull at zero temperature (i.e. the zero temperature phase diagram). As an example, we choose the system OsTc [7,23,62], and we illustrate the logical steps for obtaining it. The user should prepare his/her own computer code to download and analyze the data as suggested.

1. Upon interrogation of the AFLOWLIB.org database (*database searches layer*, see Figs. 1 and 2), OsTc is found to be part of the *project-layer* with AURL `$aurl=aflowlib.duke.edu:AFLOWDATA/LIB2_RAW/`. This `$aurl` is translated into the WEB address $web=http://aflowlib.duke.edu/AFLOWDATA/LIB2_RAW.

2. The user downloads and parses the query `$web/?keywords`. Being in a *project-layer*, a better and faster alternative is to download the entries' number and type with the queries `$web/?aflowlib_entries` and `$web/?aflowlib_entries_number`. The user then parses `$web/?aflowlib_entries` and the string `Os_pvTc_pv` associated with the requested OsTc free-energy zero-temperature convex hull.

3. The user downloads and parses the `$web/Os_pvTc_pv/` part of a *set-layer*. The process can be accelerated by querying `$web/Os_pvTc_pv/?aflowlib_entries` and `$web/Os_pvTc_pv/?aflowlib_entries_number` directly, to find further `$aurl`. The results are: `aflowlib_entries=1, 2, 3, 4, .. , 657.AB, 657.BA, ..,`

`$aflowlib_entries_number=260`. We enumerate and label these 260 entries with `entry`$_i$.

4. The user loops through `$entry`$_i$, $\forall i$ and collects:
`$web/Os_pvTc_pv/$entry`$_i$`/?stoichiometry` and
`$web/Os_pvTc_pv/$entry`$_i$`/?enthalpy_formation`.

5. Finally, the user collects the free-energies and plots the convex hull as depicted in Fig. 4.

6. The whole process can be performed with the AFLOW code. The command "aflow --alloy OsTc --update --server=aflowlib.org" connects to the appropriate server, downloads the information, calculates the free-energy curve and prepares a PDF document with the appropriate information and hyperlinks to the individual entries. See the AFLOW literature for more options [13]. The user still has to double check the final relaxed structure prototypes. This is performed with a combination of:

`$web/Os_pvTc_pv/$entry`$_i$`/?compound`,

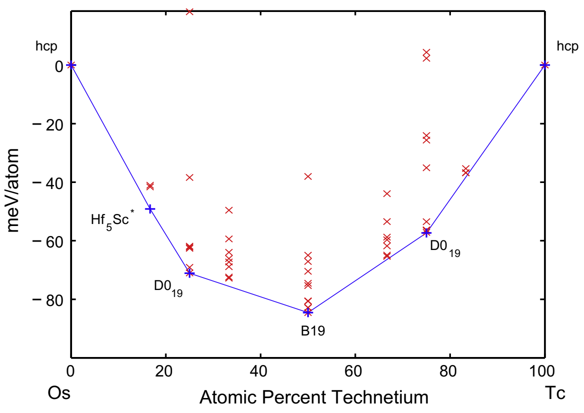


Fig. 4. The free-energy convex hull of OsTc at zero temperature, automatically calculated through the AFLOWLIB API [7,62,23].

`$web/Os_pvTc_pv/$entry`$_i$`/?geometry`,
`$web/Os_pvTc_pv/$entry`$_i$`/?positions_cartesian`,
`$web/Os_pvTc_pv/$entry`$_i$`/?prototype`, including files such as:
`$web/Os_pvTc_pv/$entry`$_i$`/edata.orig.out`, and
`$web/Os_pvTc_pv/$entry`$_i$`/edata.relax.out`, and verifying the results by consulting appropriate prototype databases (e.g., the *Naval Research Laboratory Crystal Structure database*, Ref. [39]).

7.2. Generating a zero temperature phase-diagram of CoNbSi

In this example we introduce the steps to generate a ternary zero temperature phase diagram. As an example, we choose the system CoNbSi [43], and we illustrate the logical steps for obtaining it. The user should prepare his/her own computer code to download and analyze the data as suggested.

1. Upon interrogation of the AFLOWLIB.org database (*database searches layer*, see Figs. 1 and 2), CoNbSi is found to be part of the *project-layer* with AURL `$aurl=aflowlib.duke.edu:AFLOWDATA/LIB3_RAW/`. The stability of the ternary system also depends on the stability of three binary systems: CoNb, NbSi, and CoSi. These are part of the *project-layer* with AURL `$aurl=aflowlib.duke.edu:AFLOWDATA/LIB2_RAW/`. Ternary and binary `$aurls` are translated into the WEB address as `$web=http://aflowlib.duke.edu/AFLOWDATA/LIB3_RAW`, and `$web=http://aflowlib.duke.edu/AFLOWDATA/LIB2_RAW`, respectively.

2. Similarly to the previous example, the user downloads and parses the entries of `.../LIB3_RAW/CoNb_svSi/`, `.../LIB2_RAW/CoNb_sv/`, `.../LIB2_RAW/Nb_svSi/`, `.../LIB2_RAW/CoSi/`, which are part of the *set-layer*.

3. The user loops through all the available entries and collects `/?stoichiometry` and `/?enthalpy_formation`, in `.../LIB3_RAW/CoNb_svSi/&entry`$_i$, `.../LIB2_RAW/CoNb_sv/&entry`$_i$, `.../LIB2_RAW/Nb_svSi/&entry`$_i$, and `.../LIB2_RAW/CoSi/&entry`$_i$.

4. The user calculates the convexity of the formation enthalpy landscape (we use QHULL [63]) and plots the phase diagrams (we use GNUPLOT [64]). The diagram is depicted in Fig. 5.

7.3. Obtaining band structures

In this example, we introduce the steps to obtain the band structure and density of states plots for a calculated material. As an example, we choose the compound Al_2CuMn. The user should prepare his/her own computer code to download and analyze the data as suggested.

1. Upon interrogation of the AFLOWLIB.org database (*database searches layer*, see Figs. 1 and 2), Al_2CuMn is found to be part of the *project-layer* with AURL `$aurl=aflowlib.duke.edu:AFLOWDATA/LIB3_RAW/`. This `aurl` is translated into the WEB address `$web=http://aflowlib.duke.edu/AFLOWDATA/LIB3_RAW`.

2. Within the *project-layer*, the user parses the query `web/?aflowlib_entries`, which shows that the string `AlCu_pvMn_pv` is associated with the requested AlCuMn ternary system.

3. The user parses `$web/AlCu_pvMn_pv/`. The query is part of a *set-layer*. In this case, the set contains 10 *entries*, namely the calculation for the `AlCu_pvMn_pv` system in the prototypes `ICSD_57695.ABC`, `T0001.{A2BC,AB2C,ABC2}` (Heusler configurations), `T0002.{A2BC,AB2C,ABC2}` (inverse Heusler), and `T0003.{ABC,BCA,CAB}` (half-Heusler). The postfixes `A`, `B`, `C` indicate the positions of the species in the prototype; e.g. `A2BC` indicates the material Al_2CuMn.

4. Using the `?enthalpy_formation` and `?loop` queries for these 10 entries the user finds which are stable and include a band structure calculation (indicated by a negative formation enthalpy and the string `bands` in the `?loop` query output). For this example, the user selects the *entry* `$web/AlCu_pvMn_pv/T0001.A2BC/` for the compound Al_2CuMn, which satisfies both queries.

5. At the *calculation-layer*, the user finds the full `aflowlib.out` entry. By interrogating `$web/AlCu_pvMn_pv/T0001.A2BC/?files`, the user obtains a list of all of the files available for download for this calculation, including:

- the input file for the calculation `$web/AlCu_pvMn_pv/T0001.A2BC/aflow.in`,
- the spin polarized band structure with electronic density of states `$web/AlCu_pvMn_pv/T0001.A2BC/AlCu_pvMn_pv.T0001.A2BC.png`
- the Brillouin Zone in AFLOW notation [52] `AlCu_pvMn_pv.T0001.A2BC_BZ.png`
- the total electronic density of states `AlCu_pvMn_pv.T0001.A2BC_DOS.png`
- the partial electronic density of states for inequivalent positions "Al"-`AlCu_pvMn_pv.T0001.A2BC_PEDOS_1_4_Al.png`, "Cu"-`AlCu_pvMn_pv.T0001.A2BC_PEDOS_3_4_Cu.png`, and "Mn"-`AlCu_pvMn_pv.T0001.A2BC_PEDOS_4_4_Mn.png`

A collage of these files is shown in Fig. 6.

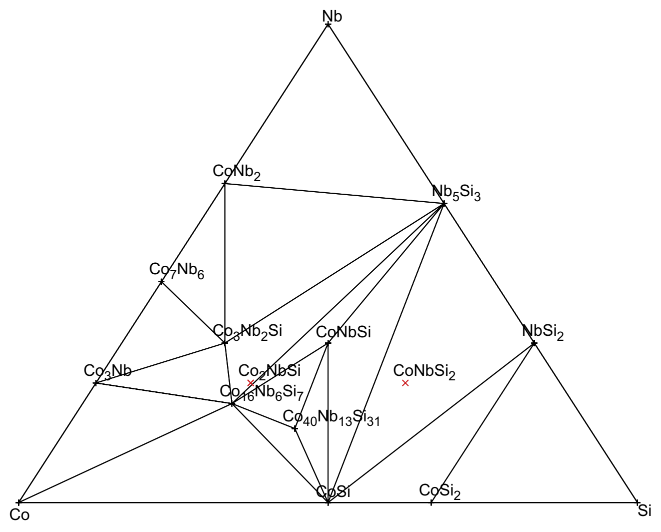


Fig. 5. Zero-temperature phase diagram of CoNbSi automatically calculated through the AFLOWLIB API. Metastable half-Heuslers are marked with red stars [43]. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

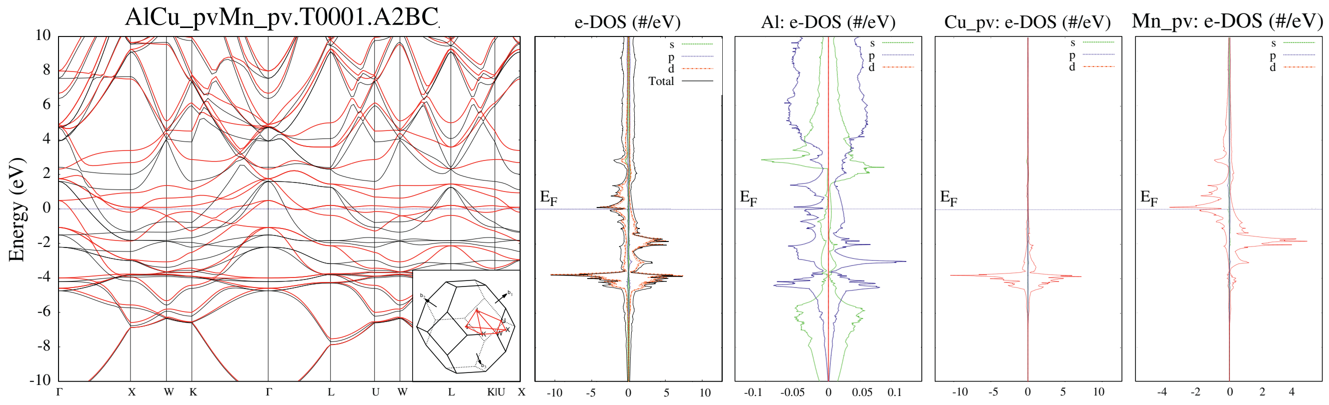


Fig. 6. Band structure, total and partial electronic densities of states for Al_2CuMn, as available through the AFLOWLIB API. The partial density of states is calculated only for inequivalent atomic positions [compound AUID = aflow:83cd2da0257e8def].

Also available in the *calculation-layer* for this entry are the input and output files from the VASP and AFLOW runs, that the user may extract from the output of the `$web/AlCu_pvMn_pv/T0001.A2BC/?files` command.

7.4. Synergy of experimental and calculation data on a rare prototype

The experimental data on binary alloys contains many gaps. It also presents a huge panoply of structural prototypes, ranging from very common ones, appearing in hundreds of compounds, to very rare ones appearing in just a few systems. HT calculations can be used to bridge those gaps and provide a more complete picture about the existence of yet unobserved compounds and their structures. They can also considerably extend the predicted range of those rare prototypes, indicating their existence in a larger set of binary systems. One such example studied the prevalence of the Pt_8Ti prototype. This structure has been experimentally observed in 11 systems, but a high-throughput search over all of the binary transition intermetallics revealed it should be stable at low temperatures in 59 systems [28]. The study verified all the experimental occurrences while offering additional predictions, including a few surprising ones in supposedly well-characterized systems (e.g., Cu–Zn). This example serves as a striking demonstration of the power of the high-throughput approach. In this section we present a new example, discussing recent reports observing the rare prototype Pd_4Pu_3 in a few transition metal binaries and computationally predicting a considerable extension of its stability or metastability in such systems.

The Pd_4Pu_3 (hR14, space group #148) was first observed in its eponymous system in 1967 [66,67]. It has since been reported in 37 additional binary systems, mostly of a lanthanide or an actinide with the elements Pt or Pd. [68]. Only 6 compounds of this prototype have been reported in transition metal binary systems: Ni_4Ti_3 [69,70], Pd_4Y_3 [71], Pd_4Zr_3 [72], Pt_4Zr_3 [73], Rh_4Zr_3 [74], and most recently Hf_3Pt_4 [75]. In these compounds, one component is a 3B or 4B element and the other is from the ninth or tenth column of the periodic table. In this example we wish to examine the possible appearance of this prototype in all transition metal binary systems of these columns (30 systems). This can be done in a few steps, as follows:

1. Consulting the complete list of structure designations of AFLOW with the command "aflow --protos" or by the online links, the user finds the label of the prototype, in this case `655.AB` or `655.BA` (depending on the order of the species).

2. Using `$aurl=aflowlib.duke.edu:AFLOWDATA/LIB2_RAW/?aflowlib_entries` the user finds the entry name for each of those 30 systems in the *set-layer*. Then, using `$aurl=aflowlib.duke.edu:AFLOWDATA/LIB2_RAW/XXX/?aflowlib_entries` for each of those names the user finds the calculations of the desired prototype in the *calculation-layer* (indicated by the string `655.BA` or `655.BA` in the query output).

3. Following the steps of Example Section 7.1 the user constructs the convex hull for each of these systems and finds the position of the desired structure in it, as either stable, metastable or unstable.

Following these steps for the Pd_4Pu_3 prototype we find that it appears as a low temperature stable compound in six systems, two reported in experiments and four newly predicted ones. The structure is also found to be metastable (with less than 30 meV/atom above the respective system convex hull) in ten systems, of which it was reported in four by experiments, and is predicted in six additional ones. Among the predicted phases, three compounds of the same stoichiometry, Pt_4Y_3, Hf_3Pd_4 and Hf_3Rh_4, are reported with an unknown structure in the experimental literature but identified with the Pd_4Pu_3 structure in the calculations. Overall, the calculation extends the prevalence of this prototype (stable or metastable) among transition metal binaries from six systems to sixteen. Fig. 7 summarizes these results.

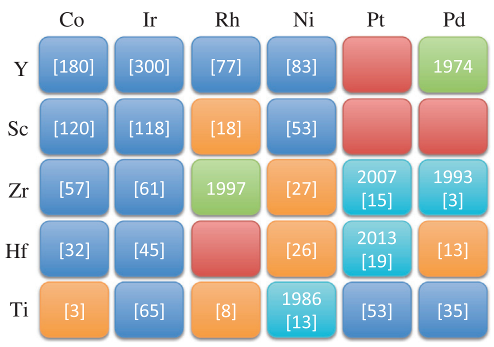


Fig. 7. Pettifor-type structure map of the Pd_4Pu_3 phase in transition metal binary systems. Both axes are labeled by increasing Mendeleev number after Pettifor [65]. Colors denote reported compounds with indicated year of discovery (green and light blue) and prediction of unreported compounds (red and orange) found to be stable (green and red) or metastable (light blue and orange) in the calculations. Systems where the structure is unstable (formation enthalpy of more than 30 meV/atom above the convex hull) are denoted in blue. The square parentheses denote the formation enthalpy of metastable and unstable structures above the convex hull of the respective system. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

7.5. Bash api.sh example

This “bash” script example `api.sh` downloads an `aflowlib.out` entry for the *project-*, *set-*, or *calculation-layers* of the binary alloys.

```
#!/bin/bash                                                       # run sh./api.sh
SERVER='http://aflowlib.duke.edu'                                 # server name
PROJECT='AFLOWDATA/LIB2_RAW/'                                     # project name
#URL=$SERVER'/'$PROJECT                                           # project-layer
URL=$SERVER'/'$PROJECT'Os_pvTc_pv/'                               # set-layer
#URL=$SERVER'/'$PROJECT'/Os_pvTc_pv/657.AB/'                      # calculation-layer
IFS=',';                                                          # options
for key in $(wget -q -O - ${URL}?keywords);                       # get all keywords
do                                                                # loop keywords
  val=$(wget -q -O - ${URL}?${key});                              # assign one keyword
  echo "${key}=${val}";                                           # print keyword
done                                                              # end loop
```

7.6. Python api.py example

This “python3” script example `api.py` downloads an `aflowlib.out` entry for the *project-*, *set-*, or *calculation-layers* of the Heusler alloys database.

```
#!/usr/bin/python3                                                # python3
import json                                                       # preamble
from urllib.request import urlopen                                # preamble
SERVER='http://aflowlib.duke.edu'                                 # server name
PROJECT='AFLOWDATA/LIB3_RAW/'                                     # project name
#URL=SERVER+'/'+PROJECT                                           # project-layer
#URL=SERVER+'/'+PROJECT+'AlCu_pvMn_pv/'                           # set-layer
URL=SERVER+'/'+PROJECT+'AlCu_pvMn_pv/T0001.A2BC/'                 # calculation-layer
entry=json.loads (urlopen (URL+'?format=json').readall ().decode ('utf-8'))   # load
for key in entry:                                                 # loop keys
  print ("{}={}".format (key, entry[key]))                        # print key
```

8. Updates beyond version 1.0

Standards, like databases, are only as good as the updates they receive when new quantities and descriptors become available. The list of `keywords` available in the current version of the standard is far from being complete for rational materials design. The user is invited to search and consult appropriate API specifications *addenda*, which will be published periodically through the consortium website AFLOWLIB.org. The entries' API version can be found by inquiring the keyword `$aurl/?data_api` as described above.

9. Conclusion

The AFLOWLIB API provides a simple and powerful tool for accessing a large set of simulated materials properties data. This will allow the community to make use of AFLOWLIB to the fullest extent possible, through search formats allowing complete accessibility of the database contents at all levels and integration of search results into externally formulated workflows. Such workflows may execute any type of investigation on the obtained data, ranging from a simple study of the properties of a specific material to extensive statistical analyses of whole structure classes for materials prediction. The full provenance of the data produced is provided, following a standard of reproducible and transparent scientific data sharing, to facilitate its straightforward reproduction and extension.

The AFLOWLIB database is growing continually by updating existent alloy libraries and adding new ones (e.g., recent attention is focused on ternary systems and electronic properties). The new API described in this paper is built on top of the AFLOW framework, developed to create the database and to interrogate it, but it can be easily extended to other materials design environments. It is constructed as a federatable tool to maximize the utility of the database to the scientific community and expedite scientific collaboration with particular emphasis on reproducibility, accessibility and attribution.

Acknowledgments

The authors thank Drs. D. Irving, G. Hart, S. Sanvito, L. Kronik, A. Kolmogorov, M. Mehl, N. Mingo, J. Carrete, A. Natan, M. Fornari, O. Isayev, A. Tropsha, K. Persson, G. Ceder, and A. Stelling for useful comments. This work is partially supported by DOD-ONR (N00014-13-1-0635, N00014-11-1-0136, N00014-09-1-0921), NIST #70NANB12H163 and by the Duke University—Center for Materials Genomics. C.T. and S.C. acknowledge partial support by DOE (DE-AC02-05CH11231), specifically the BES program under Grant #EDCBEE. The consortium AFLOWLIB.org acknowledges the Fulton Supercomputer Center and the CRAY corporation for computational assistance.

References

[1] Editorial, Nat. Mater. 12 (2013) 173.
[2] Office of Science and Technology Policy, White House, Materials Genome Initiative for Global Competitiveness, 2011. <http://www.whitehouse.gov/mgi>.

[3] S. Curtarolo, G.L.W. Hart, M. Buongiorno Nardelli, N. Mingo, S. Sanvito, O. Levy, Nat. Mater. 12 (2013) 191–201.
[4] G. Ceder, K. Persson, How Supercomputers Will Yield a Golden Age of Materials Science, Scientific American, December 2013.
[5] J. Greeley, T.F. Jaramillo, J. Bonde, I. Chorkendorff, J.K. Nørskov, Nat. Mater. 5 (2006) 909–913.
[6] K. Yang, W. Setyawan, S. Wang, M. Buongiorno Nardelli, S. Curtarolo, Nat. Mater. 11 (2012) 614–619.
[7] G.L.W. Hart, S. Curtarolo, T.B. Massalski, O. Levy, Phys. Rev. X 3 (2013) 041035.
[8] R. Armiento, B. Kozinsky, M. Fornari, G. Ceder, Phys. Rev. B 84 (2011) 014103.
[9] W. Setyawan, R.M. Gaume, S. Lam, R.S. Feigelson, S. Curtarolo, ACS Combust. Sci. 13 (2011) 382–390.
[10] S. Wang, Z. Wang, W. Setyawan, N. Mingo, S. Curtarolo, Phys. Rev. X 1 (2011) 021012.
[11] L. Yu, A. Zunger, Phys. Rev. Lett. 108 (2012) 068701.
[12] L.A. Agapito, A. Ferretti, A. Calzolari, S. Curtarolo, M. Buongiorno Nardelli, Phys. Rev. B 88 (2013) 165127.
[13] S. Curtarolo, W. Setyawan, G.L.W. Hart, M. Jahnatek, R.V. Chepulskii, R.H. Taylor, S. Wang, J. Xue, K. Yang, O. Levy, M. Mehl, H.T. Stokes, D.O. Demchenko, D. Morgan, Comput. Mater. Sci. 58 (2012) 218–226.
[14] 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, Comput. Mater. Sci. 58 (2012) 227–235.
[15] A. Jain, G. Hautier, C.J. Moore, S.P. Ong, C.C. Fischer, T. Mueller, K.A. Persson, G. Ceder, Comput. Mater. Sci. 50 (2011) 2295–2310.
[16] 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, APL Mater. 1 (2013) 011002.
[17] S.P. Ong, W.D. Richards, A. Jain, G. Hautier, M. Kocher, S. Cholia, D. Gunter, V.L. Chevrier, K.A. Persson, G. Ceder, Comput. Mater. Sci. 68 (2013) 314–319.
[18] T. Haupt, M. Horstemeyer, N. Sukhija, G. Henley, Engineering Virtual Organization for Cyber Design (EVOCD), 2013. <https://icme.hpc.msstate.edu/>.
[19] S. Curtarolo, D. Morgan, G. Ceder, Calphad 29 (2005) 163–211.
[20] S. Curtarolo, A.N. Kolmogorov, F.H. Cocks, Calphad 29 (2005) 155.
[21] O. Levy, G.L.W. Hart, S. Curtarolo, J. Am. Chem. Soc. 132 (2010) 4830–4833.
[22] O. Levy, G.L.W. Hart, S. Curtarolo, Acta Mater. 58 (2010) 2887–2897.
[23] O. Levy, G.L.W. Hart, S. Curtarolo, Phys. Rev. B 81 (2010) 174106.
[24] R.V. Chepulskii, W.H. Butler, A. van de Walle, S. Curtarolo, Scr. Mater. 62 (2010) 179–182.
[25] O. Levy, R.V. Chepulskii, G.L.W. Hart, S. Curtarolo, J. Am. Chem. Soc. 132 (2010) 833–837.
[26] R.H. Taylor, S. Curtarolo, G.L.W. Hart, Phys. Rev. B 84 (2011) 084101.
[27] R.H. Taylor, S. Curtarolo, G.L.W. Hart, Phys. Rev. B 81 (2010) 024112.
[28] R.H. Taylor, S. Curtarolo, G.L.W. Hart, J. Am. Chem. Soc. 132 (2010) 6851–6854.
[29] R.V. Chepulskii, S. Curtarolo, Appl. Phys. Lett. 97 (2010) 221908.
[30] O. Levy, M. Jahnatek, R.V. Chepulskii, G.L.W. Hart, S. Curtarolo, J. Am. Chem. Soc. 133 (2011) 158–163.
[31] M. Jahnatek, O. Levy, G.L.W. Hart, L.J. Nelson, R.V. Chepulskii, J. Xue, S. Curtarolo, Phys. Rev. B 84 (2011) 214110.
[32] A. Beaulieu, Learning SQL, O'Reilly, Sebastapol, CA, USA, 2009.
[33] The Open Group, 2013. <http://opengroup.org/austin/papers/posix_faq.html>.
[34] World Wide Web Consortium, HTML 4.01 Specification, 1999. <http://www.w3.org/TR/REC-html40/>.
[35] G. Kresse, J. Furthmüller, Phys. Rev. B 54 (1996) 11169–11186.
[36] 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, J. Phys.: Condens. Matter 21 (2009) 395502.
[37] K.F. Garrity, J.W. Bennett, K.M. Rabe, D. Vanderbilt, Comput. Mater. Sci. 81 (2014) 446–452.
[38] T.B. Massalski, H. Okamoto, P.R. Subramanian, L. Kacprzak (Eds.), Binary Alloy Phase Diagrams, American Society for Metals, Materials Park, OH, 1990.
[39] M. Mehl, Naval Research Laboratory Crystal Structure database, 2011. <http://cst-www.nrl.navy.mil/lattice/>.
[40] A.D. Mighell, V.L. Karen, Acta Crystallogr. Sect. A 49 (1993) c409.
[41] V.L. Karen, M. Hellenbrandt, Acta Cryst. A58 (2002) c367.
[42] I.D. Brown, S.C. Abrahams, M. Berndt, J. Faber, V.L. Karen, W.D.S. Motherwell, P. Villars, J.D. Westbrook, B. McMahon, Acta Cryst. A61 (2005) 575–580.
[43] J. Carrete, W. Li, N. Mingo, S. Wang, S. Curtarolo, Phys. Rev. X 4 (2014) 011019.
[44] G.L.W. Hart, R.W. Forcade, Phys. Rev. B 77 (2008) 224115.
[45] G.L.W. Hart, R.W. Forcade, Phys. Rev. B 80 (2009) 014120.
[46] International Digital Object Identifier Foundation, Digital Object Identifier, 2011. <http://www.doi.org>.
[47] W.W. Peterson, D.T. Brown, Proc. IRE 49 (1961) 228–235.
[48] D. Crockford, Introducing JSON, 2009. <http://www.json.org>.
[49] The PHP Group, 2001. <http://www.php.net>.
[50] T. Hahn (Ed.), International Tables of Crystallography, Volume A: Space-group Symmetry, Kluwer Academic Publishers, International Union of Crystallography, Chester, England, 2002.
[51] M.I. Aroyo, J.M. Perez-Mato, C. Capillas, E. Kroumova, S. Ivantchev, G. Madariaga, A. Kirov, H. Wondratschek, Z. Kristallogr. 221 (2006) 15–27.
[52] W. Setyawan, S. Curtarolo, Comput. Mater. Sci. 49 (2010) 299–312.
[53] V.I. Anisimov, F. Aryasetiawan, A.I. Lichtenstein, J. Phys.: Condens. Matter 9 (1997) 767.
[54] B. Himmetoglu, A. Floris, S. de Gironcoli, M. Cococcioni, Int. J. Quantum Chem. 114 (2014) 14–49.
[55] A.I. Liechtenstein, V.I. Anisimov, J. Zaanen, Phys. Rev. B 52 (1995) R5467.
[56] S.L. Dudarev, G.A. Botton, S.Y. Savrasov, C.J. Humphreys, A.P. Sutton, Phys. Rev. B 57 (1998) 1505.
[57] P. Villars, L. Calvert, Pearson's Handbook of Crystallographic Data for Intermetallic Phases, second ed., ASM International, Materials Park, OH, 1991.
[58] W. Setyawan, R.M. Gaume, R.S. Feigelson, S. Curtarolo, IEEE Trans. Nucl. Sci. 56 (2009) 2989–2996.
[59] A.L. Spek, Acta Cryst. D65 (2009) 148–155.
[60] H.T. Stokes, D.M. Hatch, J. Appl. Cryst. 38 (2005) 237–238.
[61] A.D. McNaught, A. Wilkinson, IUPAC, Compendium of Chemical Terminology, 2nd ed. (the "Gold Book"), second Revised ed., WileyBlackwell, 1997.
[62] O. Levy, J. Xue, S. Wang, G.L.W. Hart, S. Curtarolo, Phys. Rev. B 85 (2012) 012201.
[63] C.B. Barber, D.P. Dobkin, H. Huhdanpaa, ACM Trans. Math. Soft. 22 (1996) 469–483.
[64] P.K. Janert, Gnuplot in Action, Manning Publications Co., Shelter Island, NY, 2009.
[65] D.G. Pettifor, J. Phys. C: Solid State Phys. 19 (1986) 285–313.
[66] V.I. Kutaitsev, N.T. Chebotarev, M.A. Andrianov, V.N. Konev, I.G. Lebedev, V.I. Bagrova, A.V. Beznosikova, A.A. Kruglov, P.N. Petrov, E.S. Smotritskaya, Sov. Atom. Energy 23 (1967) 1279–1287.
[67] D.T. Cromer, A.C. Larson, R.B. Roof, Acta Crystallogr. B 29 (1973) 564–567.
[68] P. Villars, K. Cenzual, Pearson's Crystal Data – Crystal Structure Database for Inorganic Compounds, ASM International, Materials Park, OH, 2013.
[69] T. Saburi, S. Nenno, T. Fukuda, J. Less-Common Met. 125 (1986) 157–166.
[70] W. Tirry, D. Schryvers, K. Jorissen, D. Lamoen, Acta Crystallogr. B 62 (2006) 966–971.
[71] A. Palenzona, A. Iandelli, J. Less-Common Met. 34 (1974) 121–124.
[72] L.A. Bendersky, J.K. Stalick, R.M. Waterstrat, J. Alloys Compound. 201 (1993) 121–126.
[73] J.K. Stalick, R.M. Waterstrat, J. Alloys Compound. 430 (2007) 123–131.
[74] L.A. Bendersky, R.M. Waterstrat, J. Alloys Compound. 252 (1997) L5–L7.
[75] J.K. Stalick, R.M. Waterstrat, J. Phase Equilib. Diffus. 35 (2014) 15–23.