AFLOW-SYM: Platform for the complete, automatic and self-consistent symmetry analysis of crystals

David Hicks,[1, 2] Corey Oses,[1, 2] Eric Gossett,[1, 2] Geena Gomez,[1, 2] Richard H. Taylor,[1, 2, 3]
Cormac Toher,[1, 2] Michael J. Mehl,[4] Ohad Levy,[1, 2, 5] and Stefano Curtarolo[1, 2, 6, *]

[1]*Department of Mechanical Engineering and Materials Science, Duke University, Durham, North Carolina 27708, USA*
[2]*Center for Materials Genomics, Duke University, Durham, North Carolina 27708, USA*
[3]*Department of Materials Science and Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, USA*
[4]*United States Naval Academy, Annapolis, Maryland 21402, USA*
[5]*Department of Physics, NRCN, P.O. Box 9001, Beer-Sheva 84190, Israel*
[6]*Fritz-Haber-Institut der Max-Planck-Gesellschaft, 14195 Berlin-Dahlem, Germany*

(Dated: May 28, 2018)

Determination of the symmetry profile of structures is a persistent challenge in materials science. Results often vary amongst standard packages, hindering autonomous materials development by requiring continuous user attention and educated guesses. Here, we present a robust procedure for evaluating the complete suite of symmetry properties, featuring various representations for the point-, factor-, space groups, site symmetries, and Wyckoff positions. The protocol determines a system-specific mapping tolerance that yields symmetry operations entirely commensurate with fundamental crystallographic principles. The self consistent tolerance characterizes the effective spatial resolution of the reported atomic positions. The approach is compared with the most used programs and is successfully validated against the space group information provided for over 54,000 entries in the Inorganic Crystal Structure Database. Subsequently, a complete symmetry analysis is applied to all 1.7+ million entries of the AFLOW data repository. The AFLOW-SYM package has been implemented in, and made available for, public use through the automated, *ab-initio* framework AFLOW.

1. INTRODUCTION

Symmetry fundamentally characterizes all crystals, establishing a tractable connection between observed phenomena and the underlying physical/chemical interactions. Beyond crystal periodicity, symmetry within the unit cell guides materials classification [1], optimizes materials properties calculations, and instructs structure enumeration methods [2, 3]. Careful exploitation of crystal symmetry has made possible the characterization of electronic [4], mechanical [5, 6], and thermal properties [7–9] in high-throughput fashion [10] — giving rise to large materials properties databases such as AFLOW [1, 4, 11–19], NoMaD [20], Materials Project [21], and OQMD [22]. As these databases incorporate more properties and grow increasingly integrated, access to rapid and consistent symmetry characterizations becomes of paramount importance.

Central to each symmetry analysis is the identification of spatial and angular tolerances, quantifying the threshold at which two points or angles are considered equivalent. These tolerances must account for numerical instabilities, and, more importantly, for atypical data stemming from finite temperature measurements or deviations in experimentally measured values [23]. Existing symmetry platforms — such as FINDSYM [24, 25], Platon [26], and Spglib [27] — all cater to different symmetry objectives, and thus address tolerance issues in unique ways. FINDSYM — designed for ease-of-use — acknowledges that its algorithms cannot handle noisy data, and it applies no treatments for ill-conditioned data [24]. The Platon geometry package, containing the subroutine ADDSYM, allows a small percentage of candidate atomic mappings to fail and attempts to capture missing higher symmetry descriptions [26]. Lower symmetry descriptions in atomic coordinates can originate from **i.** extraction issues with X-ray diffraction data (*e.g.*, incorrectly identified crystal system, altered Laue class within a crystal system, and neglected inversion) and **ii.** *ab-initio* relaxations (*e.g.*, lost internal translations) [28–30]. The Spglib package applies independent tolerance scans within subroutines — *e.g.*, in its methods for finding the primitive cell (`get_primitive`) and Wyckoff positions (`ssm_get_exact_positions`) — if certain crystallographic conventions are violated, potentially yielding globally inconsistent symmetry descriptions [27]. These packages present suggested default tolerance values that are largely arbitrary, and likely justified *a posteriori* on a limited test set. In the general case, or in the event where these global defaults fail, the packages fall back on user-defined tolerances. Unfortunately, it is difficult to compare results across packages outside of these default values because tolerances are defined differently. FINDSYM and Spglib both offer a tunable atomic mapping tolerance, along with a lattice tolerance (FINDSYM) and an angular tolerance (Spglib); whereas Platon has four separate input tolerances, each specific to a particular operation type. Ultimately, these inconsistencies are symptomatic of an underlying inability to appropriately address tolerances in symmetry analyses.

* stefano@duke.edu

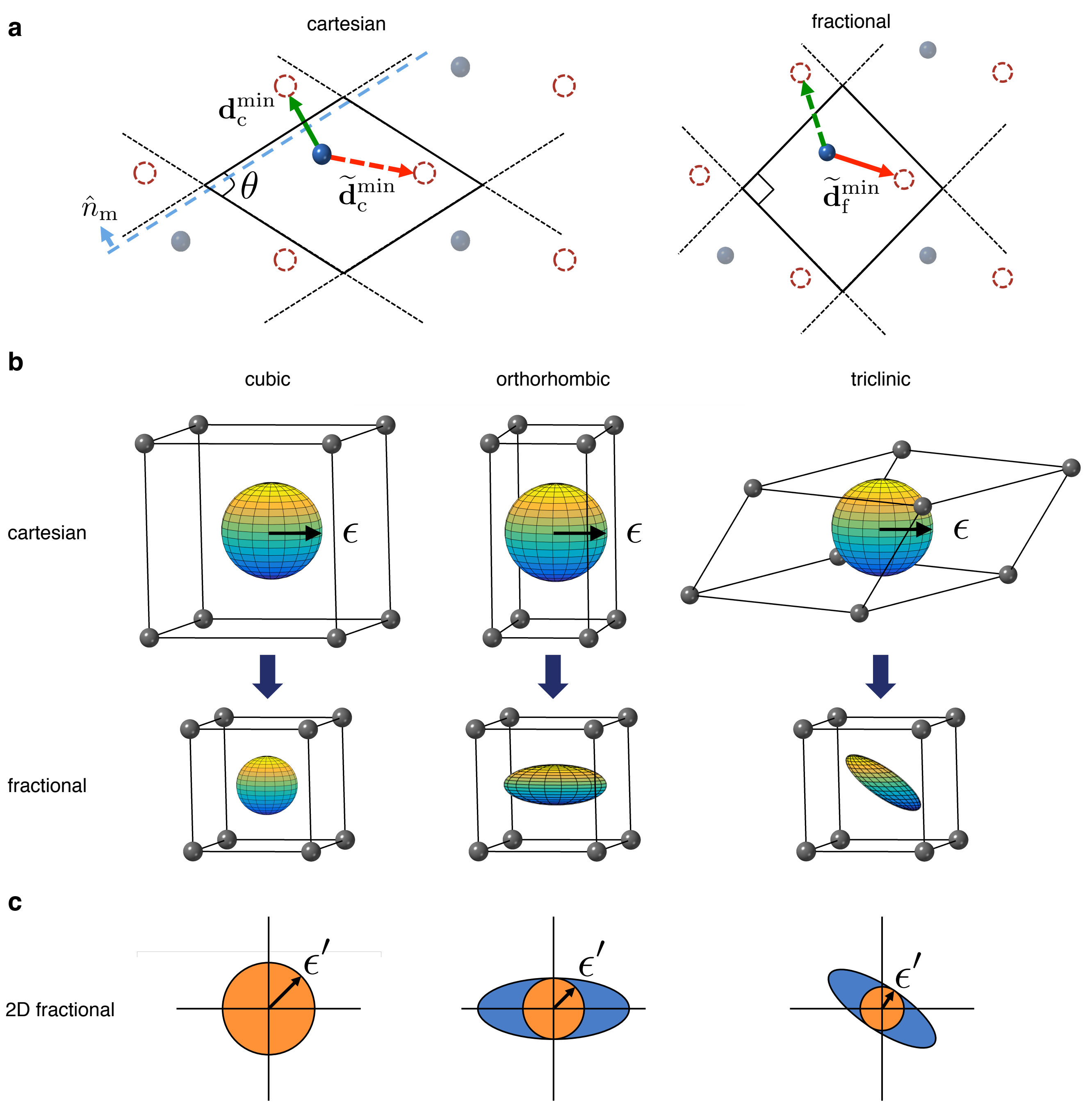
a
cartesian
$\mathbf{d}_\mathrm{c}^\mathrm{min}$
$\widetilde{\mathbf{d}}_\mathrm{c}^\mathrm{min}$
θ
$\hat{n}_\mathrm{m}$
fractional
$\widetilde{\mathbf{d}}_\mathrm{f}^\mathrm{min}$
b
cubic
orthorhombic
triclinic
cartesian
ϵ
fractional
c
2D fractional
ϵ'

FIG. 1. **Visualizations of space warping with a basis transformation.** (**a**) To validate a candidate mirror operation (described by $\hat{n}_\mathrm{m}$) on a crystal (blue atoms), the operation is applied to yield a transformed crystal (hollow orange atoms superimposed on the original crystal). The true minimum distance between the blue and orange atoms is resolved in cartesian space, indicated by the green $\mathbf{d}_\mathrm{c}^\mathrm{min}$ vector. However, the bring-in-cell method determines another periodic image to be closer, highlighted by the dashed red vector. The mismatch is obscured in fractional space, where the red vector appears smaller than the green, indicated by $\widetilde{\mathbf{d}}_\mathrm{c}^\mathrm{min}$. (**b**) An atom is placed in the middle of the lattice with a surrounding sphere of radius ϵ. Mapping occurs when the position of an atom transformed by a symmetry operator is within the sphere. The size and shape of the sphere is warped with a basis transformation (cartesian to fractional): uniform compression occurs in cubic cells, oblate compression in orthorhombic cells, non-uniform (sheared) compression in triclinic lattices. (**c**) 2D illustration of how the tolerance (ϵ) warps in fractional space for cubic, orthorhombic and triclinic lattices. The orange circle with radius ϵ' in fractional coordinates indicates the bounds of the safe mapping region, independent of direction.

Managing input/output formats for these packages also presents a challenge. FINDSYM and Platon both read `CIF` and `SPF` files, which are particularly useful for structures deriving from larger crystal structure databases, such as the Inorganic Crystal Structure Database (ICSD) [31, 32] and Cambridge Structural Database (CSD) [33]. Platon also supports a few other useful input formats, while Spglib created its own input format. Package-specific formats are useful for the developers, but create an unnecessary hurdle for the user to implement structure-file converters. This is particularly problematic when package developers change the formats of these inputs with new version updates, forcing the user to continuously adapt workflows/frameworks. Additionally, all output formats are package-specific, with a medley of symmetry descriptions and representations provided among the three. The assortment of outputs presents yet another hurdle for users trying to build custom solutions for framework integration. Furthermore, it forces users to become locked-in to these packages.

These issues require extensive maintenance on the side of the user, with little guarantee of the validity of the resulting symmetry descriptions. In the case of large materials properties databases, providing such individual attention to each compound's symmetry description becomes entirely impractical. Herein, we present a robust symmetry package implemented in the automated, *ab-initio* computational framework AFLOW, known as AFLOW-SYM. The module delivers a complete symmetry analysis of the crystal, including the symmetry operations for the lattice point group, reciprocal space lattice point group, factor group, crystal point group, dual of the crystal point group, symmetrically equivalent atoms, site symmetry, and space group (see Appendix A for an overview of symmetry groups). Moreover, it provides general crystallographic descriptions including the space group number and label(s), Pearson symbol, Bravais lattice type and variations, Wyckoff positions, and standard representations of the crystal. The routine employs an adaptive, structure-specific tolerance scheme capable of handling even the most skewed unit cells. By default, two independent symmetry procedures are applied, enabling corroboration of the characterization. The scheme has been tested on 54,000 ICSD compounds in the aflow.org repository, showing substantial improvement in characterizing space groups and lattice types compared to other packages. Along with a standardized text output, AFLOW-SYM presents the results in JavaScript Object Notation (JSON) for easy integration into different workflows. The software is completely written in C++ and it can be compiled in UNIX, Linux, and MacOSX environments using the gcc/g++ suite of compilers. The package is open-source and is available under the GNU-GPL license. An AFLOW-SYM Python module is also available to facilitate integration with other workflow packages, *e.g.*, AFLOWπ [19, 34] and NoMaD [20]. Thus, AFLOW-SYM serves as a robust one-stop symmetry shop for the materials science community.

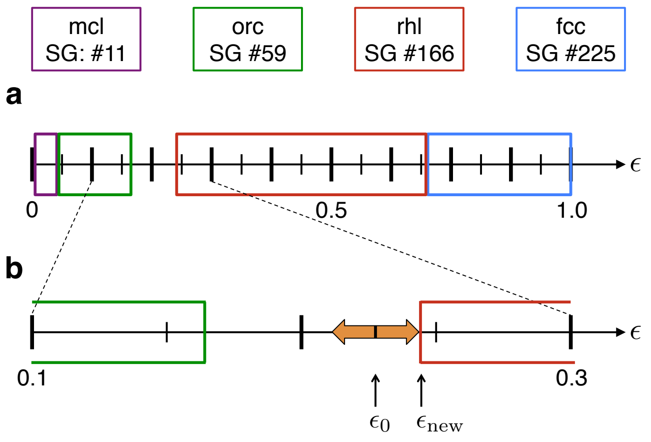

FIG. 2. **Variation of space group with mapping tolerance for AgBr (ICSD #56551) as determined by AFLOW-SYM.** (**a**) Space groups and tolerance ranges identified are as follows (ascending order): $1.0 \times 10^{-6} - 4.0855 \times 10^{-2}$ Å is monoclinic (SG #11), $4.08556 \times 10^{-2} - 1.64186 \times 10^{-1}$ Å is orthorhombic (SG #59), $2.46281 \times 10^{-1} - 6.69605 \times 10^{-1}$ Å is rhombohedral (SG #166), and $6.69606 \times 10^{-1} - 1.0$ Å is face-centered cubic (SG #225). (**b**) A gap is highlighted between $1.64186\times10^{-1}-2.46281\times10^{-1}$ Å where no consistent space group is identified. The orange arrows illustrate how the algorithm scans possible tolerances to find the closest consistent space group.

2. METHODS

A. Periodic boundary conditions in skewed cells

Analyzing the symmetry of materials involves determining the full set of their isometries. Algorithmically, candidate symmetry operators are applied to a set of atoms and validated if **i.** distances between atoms and their transformed counterparts are within a mapping tolerance ϵ, and **ii.** the mappings are isomorphic (one-to-one). For convenience, ϵ is defined in units of a Euclidean space — Angstroms in this case. An explicit mapping function is defined, indicating whether atom mappings are successful:

$$\mathrm{map}_{\mathrm{atom}}\left(\mathbf{d}_{\mathrm{c}}\right) = \begin{cases} \text{true} & \text{if } \|\mathbf{d}_{\mathrm{c}}\| < \epsilon \\ \text{false} & \text{otherwise} \end{cases}, \tag{1}$$

where $\mathbf{d}_{\mathrm{c}}$ is the cartesian distance vector between an atom and a transformed atom. Symmetries of the crystal are discovered when successful isomorphic mappings — given by Equation (1) — exist between all of the original and transformed atoms. Under periodic boundary conditions, the minimum distance for the mapping function is identified by considering equivalent atoms of nearby cells (so-called method of images [35]):

$$\mathbf{d}_{\mathrm{c}}^{\mathrm{min}} = \min_{n_a,n_b,n_c} \left(\mathbf{d}_{\mathrm{c}} + n_a\mathbf{a} + n_b\mathbf{b} + n_c\mathbf{c}\right), \tag{2}$$

where $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ are the lattice vectors; n_a, n_b, n_c are the indices of neighboring cells; and $\mathbf{d}_{\mathrm{c}}^{\mathrm{min}}$ is the globally optimal cartesian distance vector. In the simplest case of a purely orthorhombic cell, the approach requires exploration of the 26 surrounding unit cells ($-1 \leq n_a, n_b, n_c \leq 1$). However, additional neighboring cells should be considered with increased skewness of the lattice vectors (see Section 2 G), making it prohibitively expensive. Instead, many algorithms minimize the distance vector through a greedy, bring-in-cell approach [35]. Working with fractional coordinates, each component i of the distance vector $\mathbf{d}_{\mathrm{f}}$ is minimized using the nearest-integer function (nint [35]):

$$\begin{aligned} \widetilde{\mathbf{d}}_{\mathrm{f},i}^{\mathrm{min}} &= \mathbf{d}_{\mathrm{f},i} - \mathrm{nint}\left(\mathbf{d}_{\mathrm{f},i}\right) \quad \forall i, \\ \widetilde{\mathbf{d}}_{\mathrm{c}}^{\mathrm{min}} &= \mathbf{L}\widetilde{\mathbf{d}}_{\mathrm{f}}^{\mathrm{min}}, \end{aligned} \tag{3}$$

where $\mathbf{L}$ is the column-space matrix representation of the lattice, and $\widetilde{\mathbf{d}}_{\mathrm{c}}^{\mathrm{min}}$ is $\widetilde{\mathbf{d}}_{\mathrm{f}}^{\mathrm{min}}$ converted to cartesian coordinates for the mapping determination in Equation (1).

While a convenient shortcut, the bring-in-cell minimum distance is not generally equivalent to the globally optimized distance: $\widetilde{\mathbf{d}}_{\mathrm{c}}^{\mathrm{min}} \neq \mathbf{d}_{\mathrm{c}}^{\mathrm{min}}$ (see Figure 1(a)). A component-by-component minimization of the distance vector assumes independent basis vectors (no skewness) and neglects potentially closer images only considered by exploring all neighboring cells. Occurrence of a distance mismatch depends on the lattice type and compromises the integrity of the mapping determination. The issue becomes particularly elusive in fractional coordinates, where the size and shape of the cell is warped to yield a unit cube as shown in Figure 1(b). The greater the anisotropy of the lattice, the larger the warping. Figure 1(c) illustrates how the tolerance changes between cartesian and fractional space. In the general case, a spherical tolerance in cartesian coordinates warps into an ellipsoid in fractional coordinates. Hence, the criteria for successful mappings in fractional space are direction dependent unless the distance is sufficiently small, *i.e.*, within the circumscribed sphere of radius ϵ' (highlighted in orange). Distances within ϵ' in fractional space always map within ϵ in cartesian space, but a robust check (global optimization) is needed for larger distances to account for the extremes of the ellipsoid. Since most distances outside of ϵ' do not yield mappings, such a robust check is generally wasteful. Instead, more useful insight can be garnered from Figure 1(c): tolerances sufficiently bounded by the skewness can still yield a proper mapping determination using the inexpensive bring-in-cell algorithm.

The goal is to define an upper tolerance threshold to safely ensure that the bring-in-cell minimum ($\widetilde{\mathbf{d}}_{\mathrm{c}}^{\mathrm{min}}$) and global minimum ($\mathbf{d}_{\mathrm{c}}^{\mathrm{min}}$) yield the same mapping results, in spite of a distance mismatch:

$$\begin{aligned} \mathrm{map}_{\mathrm{atom}}\left(\mathbf{d}_{\mathrm{c}}^{\mathrm{min}}\right) \equiv \mathrm{map}_{\mathrm{atom}}\left(\widetilde{\mathbf{d}}_{\mathrm{c}}^{\mathrm{min}}\right) \Big| \, \epsilon < \left\|\mathbf{d}_{\mathrm{c}}^{\mathrm{min}}\right\|, \\ \forall\, \mathbf{d}_{\mathrm{c}} \, \Big| \, \mathbf{d}_{\mathrm{c}}^{\mathrm{min}} \neq \widetilde{\mathbf{d}}_{\mathrm{c}}^{\mathrm{min}} . \end{aligned} \tag{4}$$

A mismatch is encountered when the image identified by the bring-in-cell method is not the optimal neighbor, therefore: $\mathbf{d}_{\mathrm{c}}^{\mathrm{min}} \leq \widetilde{\mathbf{d}}_{\mathrm{c}}^{\mathrm{min}}$. A suitable threshold needs to overcome the difference between the two methods for all mismatch possibilities, *i.e.*, ϵ would need to be below $\mathbf{d}_{\mathrm{c}}^{\mathrm{min}}$ or above $\widetilde{\mathbf{d}}_{\mathrm{c}}^{\mathrm{min}}$ to yield a consistent mapping determination. A threshold greater than $\widetilde{\mathbf{d}}_{\mathrm{c}}^{\mathrm{min}}$ is ruled out to ensure that ϵ is always smaller than the minimum interatomic distance $\left(d_{\mathrm{c}}^{\mathrm{nn(min)}}\right)$, making it possible to distinguish nearest-neighbors. To find a tolerance in the remaining region $\left(\epsilon < ||\mathbf{d}_{\mathrm{c}}^{\mathrm{min}}||\right)$, the largest mismatch possible should be addressed directly, which yields the smallest $\mathbf{d}_{\mathrm{c}}^{\mathrm{min}}$ and thus the most restrictive bound on the tolerance. Given the angles between the lattice vectors (α, β, γ), a maximum skewness is defined as

$$\xi_{\mathrm{max}} = \max\left(\cos\alpha, \cos\beta, \cos\gamma\right), \tag{5}$$

where the cosine of the angle derives from the normalized, off-diagonal terms of the metric tensor. ξ_{max} ranges from $[0, 1)$, where $\xi_{\mathrm{max}} = 0$ characterizes a perfectly orthorhombic cell. A suitable maximum mapping tolerance is heuristically defined as

$$\epsilon_{\mathrm{max}} = \left(1 - \xi_{\mathrm{max}}\right) d_{\mathrm{c}}^{\mathrm{nn(min)}}, \tag{6}$$

which appropriately reduces $d_{\mathrm{c}}^{\mathrm{nn(min)}}$ — an absolute upper bound for the tolerance to maintain resolution between atoms — with increasing skewness. The form of the coefficient $(1 - \cos\theta)$ decays quickly with basis vector overlap (on the order of θ^2), ensuring a safe enveloping bound. Tolerances well below ϵ_{max} should yield the correct mapping determination with the bring-in-cell approach (in spite of a distance mismatch); otherwise, the global minimization algorithm should be employed:

$$\mathbf{d}_{\mathrm{c}}^{\mathrm{map}} = \begin{cases} \mathbf{L}\widetilde{\mathbf{d}}_{\mathrm{f}}^{\mathrm{min}} & \text{if } \epsilon \lll \epsilon_{\mathrm{max}} \\ \min\limits_{n_a, n_b, n_c} \left(\mathbf{d}_{\mathrm{c}} + n_a\mathbf{L}_a + n_b\mathbf{L}_b + n_c\mathbf{L}_c\right) & \text{otherwise} \end{cases} . \tag{7}$$

To demonstrate the robustness of ϵ_{max}, extreme hypothetical cases are presented in Appendix C.

B. Adaptive tolerance scheme

While ϵ_{max} offers a practical upper tolerance bound for the choice of the distance minimization algorithm, it offers no insight for choosing a specific tolerance. Of course, there are fundamental constraints, such as the minimum interatomic distance and the precision of the input structure parameters: $\epsilon_{\mathrm{precision}} < \epsilon < d_{\mathrm{c}}^{\mathrm{nn(min)}}$, but these can span over several orders of magnitude, throughout which a variety of results are possible. Figure 2 illustrates the different space groups that may be assigned to AgBr (ICSD #56551) [1] with various tolerance choices

[1] http://www.aflow.org/material.php?id=56551

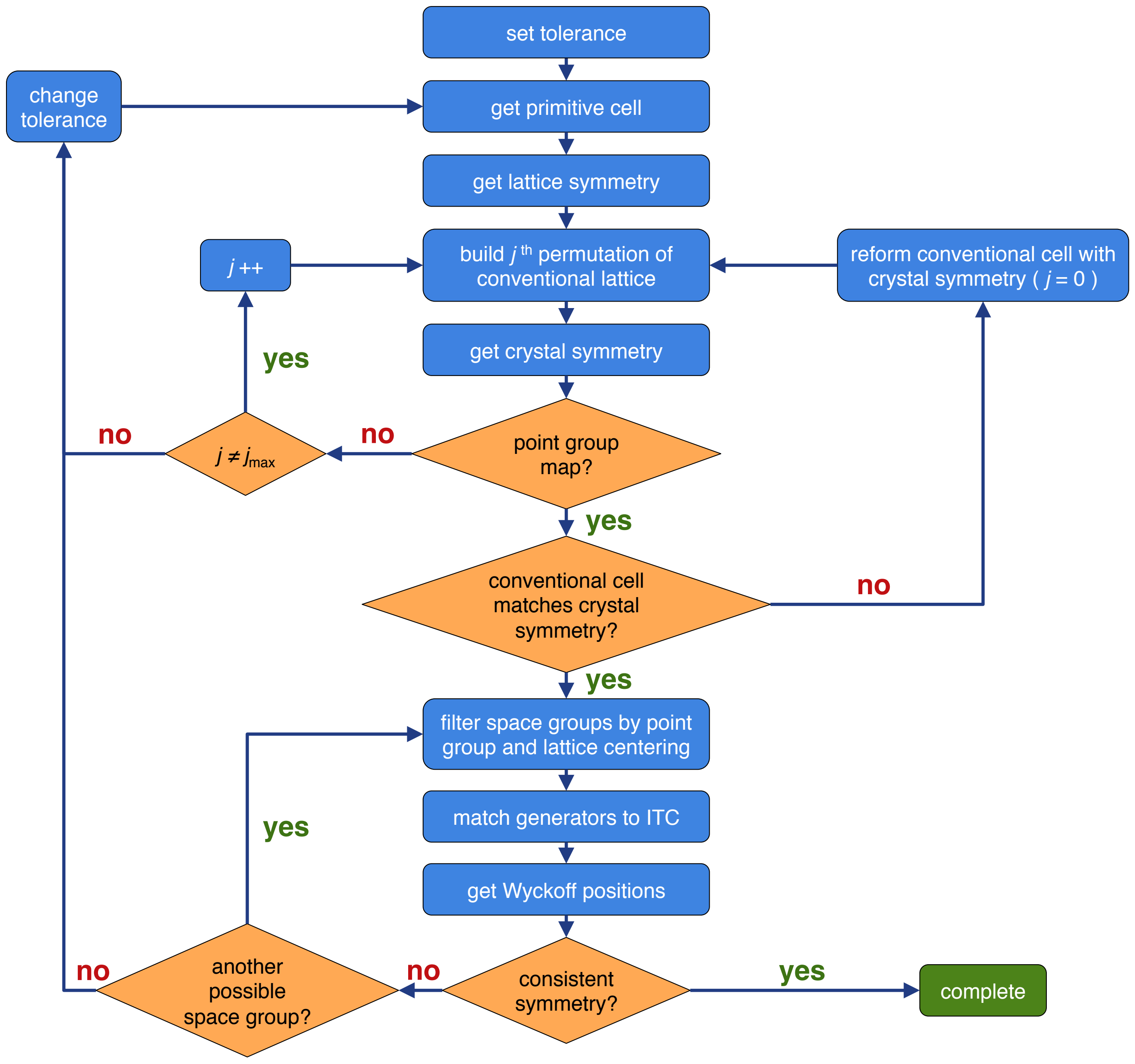
set tolerance
change tolerance
get primitive cell
get lattice symmetry
j++
build j^{th} permutation of conventional lattice
reform conventional cell with crystal symmetry ($j = 0$)
get crystal symmetry
yes
no
$j \neq j_{max}$
no
point group map?
yes
conventional cell matches crystal symmetry?
no
yes
filter space groups by point group and lattice centering
yes
match generators to ITC
get Wyckoff positions
no
another possible space group?
no
consistent symmetry?
yes
complete

FIG. 3. **Workflow for the algorithm converting a structure to the standard representation as defined by the International Tables for Crystallography.** Functions are represented by blue rectangles, and validation schemes by orange diamonds.

(the ICSD reports space group #11). Interestingly, adjacent space group regions show non-isomorphic subgroup relations: between space groups #59 and #11 and between #225 and #166. Of particular concern is the gap highlighted in Figure 2(b) between space group regions #166 and #59. Not surprisingly, these space groups share no subgroup relations. These gaps represent problematic regions where noise in the structural information interferes with determination of satisfied symmetry operations, yielding profiles inconsistent with any possible space group. Rather than an *a posteriori* selection of the symmetry elements to include in the analysis, we employ an adaptive tolerance approach. A radial tolerance scan is performed surrounding the initial input tolerance ϵ_0 to overcome the "*confusion*" region, as shown in Figure 2(b). With each adjustment of the tolerance, the algorithm updates and validates all symmetry properties and operations, yielding a globally consistent profile and an effective spatial resolution for the structure.

To fully characterize a structure's symmetry, AFLOW-SYM employs two major symmetry procedures. The first calculates the symmetry of the crystal in the

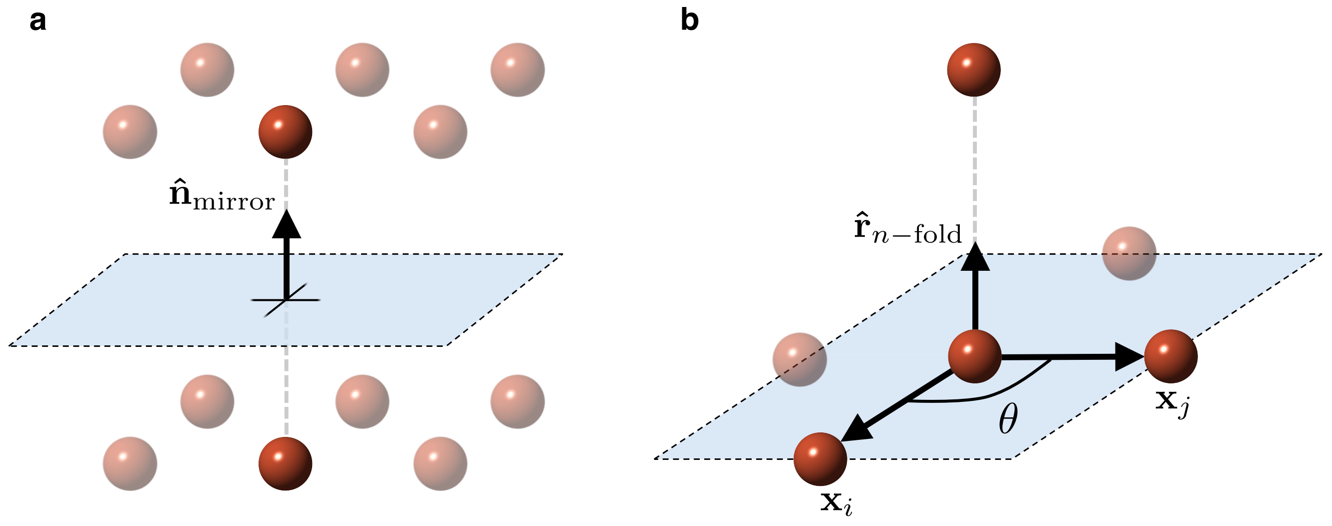
a
$\hat{\mathbf{n}}_{\text{mirror}}$
b
$\hat{\mathbf{r}}_{n-\text{fold}}$
θ
$\mathbf{x}_j$
$\mathbf{x}_i$

FIG. 4. **Minimum symmetry identifiers of the lattice system:** (**a**) mirror operations, (**b**) n-fold rotations. The resulting lattice vectors are denoted by gray dotted lines.

International Tables for Crystallography [36–39] (ITC) conventional cell, yielding the space group and Wyckoff positions. The second resolves the symmetry profile of the structure in the original (input) representation, including: the lattice point group, reciprocal lattice point group, factor group, crystal point group, dual of the crystal point group, space group, inequivalent and equivalent atoms, and the site symmetry. While both routines can be employed independently, the two are combined in AFLOW-SYM by default, affording additional validation schemes to ensure a stricter consistency.

Ultimately, the combination of the tolerance scan and integrated workflow (with robust validation schemes) ensures the automatic determination of a consistent symmetry profile. While the option remains for a user defined tolerance (with and without the scan), AFLOW-SYM heuristically defines two default tolerances values: `tight` ($\epsilon_{\text{tight}} = d_{\text{c}}^{\text{nn(min)}}/100$) and `loose` ($\epsilon_{\text{loose}} = d_{\text{c}}^{\text{nn(min)}}/10$). Generally, an expected symmetry profile (perhaps from experiments) can be found in either of the two tolerances. If no tolerance is defined, AFLOW-SYM defaults to the tight tolerance. The tolerance chosen for the analysis is compared against $\epsilon_{\max}$ to identify the required minimization technique to yield consistent mappings (see Equation (7)). Overall, the AFLOW-SYM tolerance scheme has been validated against 54,000 ICSD entries, and subsequently applied to all 1.7 million entries stored in the aflow.org repository. The symmetry results can be retrieved from the AFLOW repository via the REST-API [40] or the AFLUX Search-API [41].

C. Tolerance types and conversions

Outside of mapping distances, there are a number of relevant quantities for which an equivalence criterion is required, *e.g.*, lattice vectors, axes, angles, and symmetry operations. Instead of defining separate tolerances for each, AFLOW-SYM leverages the single spatial tolerance, converting quantities to cartesian distances whenever possible. For vector quantities such as lattice vectors and axes, the difference is taken, converted to the cartesian form (if necessary), and the Euclidean norm of the resulting vector is compared to the spatial tolerance. For angles, each angle θ_i is converted to a straight-line distance d_i:

$$d_i = \bar{x}_i \sin(\theta_i)\,, \tag{8}$$

where $\bar{x}_i$ is the average length of the angle-defining vectors in cartesian space. The two straight-line distances are subtracted and compared with the input spatial tolerance. To compare rotation matrices for a particular lattice, each matrix is transformed into its fractional form, resulting in two integer matrices that can be matched exactly.

D. International Tables for Crystallography standard representation

One strategy for uncovering a structure's symmetry profile is to convert it to a standard form, such as the one defined by the ITC. In this representation, the symmetry operations, space group, and Wyckoff positions are well tabulated, mitigating the computational expense involved in combinatorial operation searches. To efficiently explore the possibilities, the algorithm exploits the lattice symmetry to resolve the crystal symmetry, from which the conventional cell is defined. The full workflow is illustrated in Figure 3.

First, the algorithm finds a primitive representation of the crystal (of which there are many) by exploring possible internal translations forming a smaller lattice [36]. To optimize the search, only the vectors between the least frequently occurring atomic species are considered. The translation vector should preserve cell periodicity, and the resulting reduced representation should conserve the stoichiometry.

Next, the symmetry of the lattice is determined by calculating the mirror and n-fold rotation operations. The primitive cell is expanded from -1 to 1 in each direction [23] and combinations of lattice points are considered for defining the following: **i.** mirror operations characterized by a plane (normal $\hat{\mathbf{n}}_{\mathrm{mirror}}$) between two lattice points about which half the lattice points can be reflected onto the other, and **ii.** n-fold rotations ($n \in \{2,3,4,6\}$) described by an axis ($\hat{\mathbf{r}}_{n-\mathrm{fold}}$) and angle ($\theta$) such that a rotation about $\hat{\mathbf{r}}_{n-\mathrm{fold}}$ by θ yields an isomorphic mapping of lattice points. The two types of operations are illustrated in Figure 4.

The cardinality of each operation type defines the lattice system, as detailed in Table 1. If the lattice and crystal systems are the same, the characteristic vectors of the lattice operators ($\hat{\mathbf{n}}_{\mathrm{mirror}}$ and $\hat{\mathbf{r}}_{n-\mathrm{fold}}$) and corresponding lattice points define the lattice vectors of the conventional cell (also outlined in Table 1). For all cases, these lattice vectors are used to construct an initial conventional cell. The aim is to find a conventional cell whose corresponding symmetry operators (tabulated in the ITC in Table 11.2.2.1 [36]) are validated for the crystal, which can have symmetry equal to or less than the lattice. If a mismatch in cardinality is encountered, permutations of the lattice vectors are attempted. Should a mismatch remain after all permutations have been exhausted, the conventional cell is reformed to reflect the crystal symmetry. The reformed cell is chosen based on the observed cardinality of the symmetry operations (refer again to Table 1).

The resulting crystal point group set and internal translations (lattice centerings) are then used to filter candidate space groups. To pin down a space group exactly, the symmetry elements of the crystal are matched to the ITC generators — the operations generating symmetrically equivalent atoms for the general Wyckoff position [36]. However, a shift in the origin may differentiate the two sets of operators — a degree of freedom that should be addressed carefully. The appropriate origin shift should transform the symmetry elements to the ITC generators, thus forming a set of linear equations. Consider two symmetrically equivalent atom positions ($\mathbf{x}$ and $\mathbf{x}'$) in the crystal,

$$\mathbf{x}' = \mathbf{U}\mathbf{x} + \mathbf{t}, \tag{9}$$

where $\mathbf{U}$ and $\mathbf{t}$ are the fixed-point and translation operations, respectively, between the two atoms. An origin shift $\mathcal{O}$ relates these positions to those listed in the ITC:

$$\mathbf{x}_{\mathrm{ITC}} = \mathbf{x} + \mathcal{O}, \tag{10}$$

$$\mathbf{x}'_{\mathrm{ITC}} = \mathbf{x}' + \mathcal{O}. \tag{11}$$

Applying $\mathbf{U}$ to Equation (10) and subtracting it from Equation (11) yields

$$\mathbf{x}'_{\mathrm{ITC}} - \mathbf{U}\mathbf{x}_{\mathrm{ITC}} = \mathbf{x}' + \mathcal{O} - \mathbf{U}\mathbf{x} - \mathbf{U}\mathcal{O}. \tag{12}$$

The ITC translation $\mathbf{t}_{\mathrm{ITC}}$ and the crystal translation $\mathbf{t}$ are related via

$$\mathbf{t}_{\mathrm{ITC}} = \mathbf{t} + \mathcal{O} - \mathbf{U}\mathcal{O}. \tag{13}$$

Combining Equations (12)-(13) and incorporating Equations (10)-(11) produces the following system of equations:

$$(\mathbf{I} - \mathbf{U})\,\mathcal{O} = (\mathbf{t}_{\mathrm{ITC}} - \mathbf{t})\,, \tag{14}$$

where $\mathbf{I}$ is the identity. Equation (14) must be solved for each generator, often resulting in an overdetermined system. Periodic boundary conditions should also be considered when solving the system of equations, as solutions may reside in neighboring cells. If a commensurate origin shift is not found, the next candidate space group is tested.

With the shift into the ITC reference frame, the Wyckoff positions are identified by grouping atoms in the conventional cell into symmetrically equivalent sets. These sets are compared with the ITC standard to identify the corresponding Wyckoff coordinates, site symmetry designation, and letter. The procedure to find the origin shift is similarly applied to determine any Wyckoff parameters (x, y, z). For some space groups, the Wyckoff positions only differ by an internal translation (identical site symmetries), introducing ambiguity in their identification. In these cases, AFLOW-SYM favors the Wyckoff scheme producing the smallest enumerated Wyckoff lettering.

After finding the Wyckoff positions, the algorithm is complete. AFLOW-SYM returns the space group, conventional cell, and Wyckoff positions in the ITC standard representation.

E. Input orientation symmetry algorithm

The standard conventional cell representation described in the previous section affords easy access to the full symmetry profile of the structure. Nevertheless, other representations, such as the AFLOW standard primitive representation [4], are often preferred for reducing the computational cost of subsequent calculations/-analyses, such as density functional theory calculations [4]. While conversions are always possible, such as with a Minkowski lattice reduction or as was done to find the standard conventional cell, in practice it introduces errors in the structural parameters, becoming particularly

lattice/crystal system	# mirrors	# n-fold rotations	conventional cell
cubic	9	3 (four-fold)	$\mathbf{a}$,$\mathbf{b}$,$\mathbf{c}$: parallel to three equivalent four-fold axes
hexagonal	7	1 (three-/six-fold)	$\mathbf{c}$: parallel to three-/six-fold axis $\mathbf{a}$,$\mathbf{b}$: parallel to mirror axes ($\lvert\mathbf{a}\rvert \equiv \lvert\mathbf{b}\rvert$ and $\beta = 120°$)
tetragonal	5	1 (four-fold)	$\mathbf{c}$: parallel to four-fold axis $\mathbf{a}$,$\mathbf{b}$: parallel to mirror axes ($\lvert\mathbf{a}\rvert \equiv \lvert\mathbf{b}\rvert$ and $\beta = 90°$)
rhombohedral	3	1 (three-/six-fold)	$\mathbf{c}$: parallel to three-/six-fold axis $\mathbf{a}$,$\mathbf{b}$: parallel to mirror axes ($\lvert\mathbf{a}\rvert \equiv \lvert\mathbf{b}\rvert$ and $\beta = 120°$)
orthorhombic	3	-	$\mathbf{a}$,$\mathbf{b}$,$\mathbf{c}$: parallel to three mirror axes
monoclinic	1	-	$\mathbf{b}$: parallel to mirror axis (unique axis) $\mathbf{a}$,$\mathbf{c}$: parallel to two (choice of three) smallest translations perpendicular to $\mathbf{b}$ [36]
triclinic	0	-	$\mathbf{a}$,$\mathbf{b}$,$\mathbf{c}$: same as original lattice

TABLE 1. **Conventional cell construction rules based on symmetry operations.**

problematic in "*confusion*" tolerance regions (Figure 2) and tolerance-sensitive algorithms, *e.g.*, calculations of force constants [9, 42]. To mitigate the need for error-accumulating conversions, a general-representation symmetry algorithm is also incorporated in AFLOW-SYM. The integration of the two symmetry algorithms affords additional validation schemes that combat "*confusion*" tolerance regions and ensures an overall stricter consistency. The full workflow of this algorithm is outlined in Figure 5. For descriptions of the different symmetry groups, refer to Appendix A.

First, the point group of the lattice is calculated by finding all identical lattice cells of an expanded grid (see Section 2 G). The unique set of matrices that transform the rotated cells to the original cell define the lattice point group, as depicted in Figure 5(a). The search first considers all lattice points within a radius no smaller than that of a sphere encapsulating the entire unit cell. These points define the candidate lattice vectors (origin to lattice point), and those not of length a, b, or c (lattice vector lengths of original cell) are eliminated. Next, all combinations of these candidate lattice vectors are considered, eliminating sets by matching the full set of lattice parameters (lattice vector lengths and angles of the original cell). The transformation matrix is calculated as

$$\mathbf{U}_{\mathrm{c}} = \mathbf{L}\left(\mathbf{L}'\right)^{-1} \tag{15}$$

where $\mathbf{U}_{\mathrm{c}}$ is the cartesian form of the transformation (rotation) matrix, $\mathbf{L}$ is the original, column-space matrix representation of the lattice, and $\mathbf{L}'$ is the rotated lattice. The fractional form of the transformation matrix ($\mathbf{U}_{\mathrm{f}}$) is similarly derived replacing $\mathbf{L}$ and $\mathbf{L}'$ with their fractional counterparts (the fractional form of $\mathbf{L}$ is trivially the identity matrix).

The calculation of the lattice point group allows rapid determination of its reciprocal space counterpart, describing the point group symmetry of the Brillouin zone. The transformation of symmetry operators is straightforward, following standard basis change rules in dual spaces. A contragredient transformation converts the real-space form of the operator to its reciprocal counterpart, which is trivial for the cartesian form of the operator (orthogonal matrix):

$$\begin{aligned}\mathbf{V}_{\mathrm{c}} &= \left(\mathbf{U}_{\mathrm{c}}^{-1}\right)^{\mathrm{T}} = \mathbf{U}_{\mathrm{c}},\\ \mathbf{V}_{\mathrm{f}} &= \left(\mathbf{U}_{\mathrm{f}}^{-1}\right)^{\mathrm{T}},\end{aligned} \tag{16}$$

where $\mathbf{U}_{\mathrm{c}}/\mathbf{U}_{\mathrm{f}}$ and $\mathbf{V}_{\mathrm{c}}/\mathbf{V}_{\mathrm{f}}$ are the cartesian/fractional forms of the symmetry operator in real and reciprocal spaces, respectively [43].

Next, the coset representatives of the factor group are determined, characterizing the symmetry of the unit cell. These operations are characterized by a fixed-point rotation (lattice point group) and an internal translation that yield an isomorphic mapping among the atoms. The smallest set of candidate translation vectors can be found among atoms of the least frequently occurring species. This symmetry description is represented by Figure 5(b).

The point group of the crystal is then extracted from the coset representatives of the factor groups. By exploiting the homomorphism — or isomorphism for primitive cells — between the factor group and the crystal point group, the internal translations of the coset representatives are removed and the unique elements yield the crystal point group. This is portrayed in Figure 5(c). The

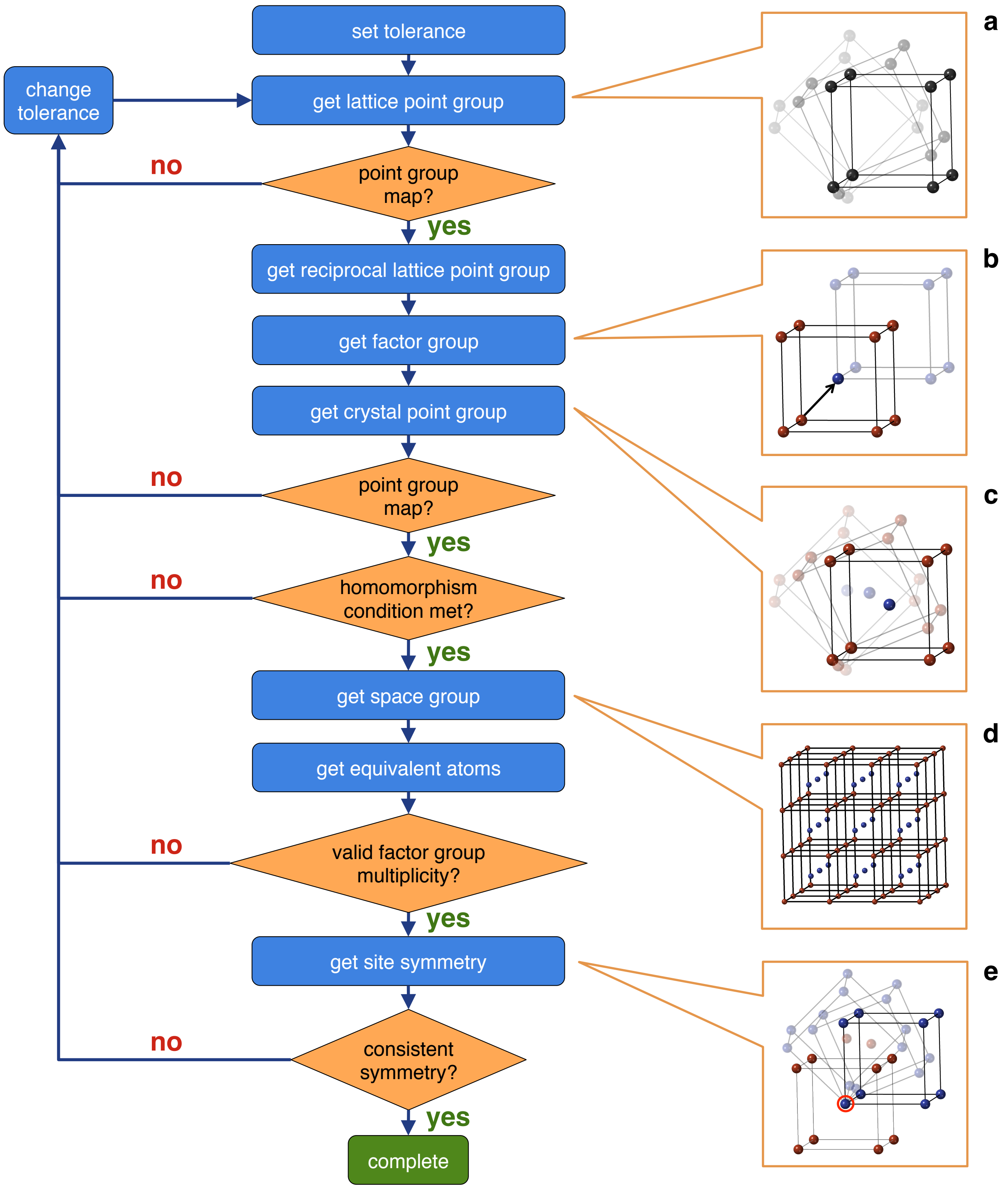

FIG. 5. **Workflow for the algorithm calculating the symmetry operations of the system in its original representation.** Functions are represented by blue rectangles, and validation schemes by orange diamonds.

dual space counterpart of the crystal point group is derived by performing the contragredient transformation, as shown in Equation (16).

The space group operations are similarly derived from the coset representatives of the factor group. The space group describes the symmetry of the infinitely periodic crystal, resulting from the propagation of the unit cell symmetry throughout the lattice. A finite set of space group operators are generated by applying the lattice translations to each of the coset representative operations out to a specified radius. The operation is depicted in Figure 5(d).

The coset representatives of the factor group also resolve the symmetrically equivalent atoms (Wyckoff positions). Atoms that are symmetrically equivalent map onto one another through a coset representative operation. This organization is convenient for calculating the site symmetry of the crystal. The site symmetry, or site point group, are exposed by centering the reference frame onto each atomic site and applying the operations of the crystal point group, as illustrated in Figure 5(e). To expedite this process, the site symmetries are explicitly calculated for all inequivalent atoms. They are then propagated to equivalent atoms with the appropriate change of basis (dictated by the coset representative mapping the inequivalent atom to the equivalent atom).

F. Consistency of symmetry

There are a finite number of operation sets that a crystal can exhibit [44]. A set of symmetry operations outside of those allowed by crystallographic group theory are attributed to noisy data, thus warranting the adaptive tolerance scan. Numerous symmetry rules are validated throughout the AFLOW-SYM routines. The list of consistency checks are indicated below.

1. Point group (lattice/crystal) contains (at the very least) the identity element.
2. Point group (lattice/crystal) matches 1 of 32 point groups.
3. Coset representative of the factor group is an integer multiple of the crystallographic point group (homomorphic/isomorphic condition).
4. Space group symbol decomposes into crystallographic point group symbol by removing translational components (with the exception of derivative structures).
5. Number of symmetrically equivalent atoms is divisible by the ratio of the number of operations in the factor and crystal point groups.
6. Space group and Wyckoff positions match ITC convention [36].

G. Exploring the atomic environment

A description of the local atomic environments in a crystal is required for determination of atom coordination and atom/lattice mappings. Depending on the cell representation, an expansion is generally warranted for sufficient exploration of the nearest neighbors. Here, an algorithm is outlined for determining the number of neighboring cells to explore in order to capture the local environment within a given exploration radius (r_{sphere}). In AFLOW-SYM, the default exploration radius is the largest distance between any two lattice points in a single unit cell. First, the normal of each pair of lattice vectors is calculated and scaled to be of length r_{sphere}, *e.g.*, $\mathbf{n}_1 = r_{\text{sphere}} \cdot \mathbf{b} \times \mathbf{c} / \|\mathbf{b} \times \mathbf{c}\|$, where $\mathbf{b}$ and $\mathbf{c}$ are lattice vectors. Next, the scaled normals are converted to the basis of the lattice, *e.g.*, $\mathbf{n}_1' = \mathbf{L}^{-1}\mathbf{n}_1$, where $\mathbf{L}$ is the column-space matrix representation of the lattice. The magnitude (rounded up to the nearest integer) of the i^{th} component of the $\mathbf{n}_i'$ vector reveals the pertinent grid dimensions (d_1, d_2, d_3). A uniform sphere of radius r_{sphere} centered at the origin fits within a 3-D grid spanning $[-d_i, d_i]$.

3. RESULTS

Highlighted here are benchmarks to compare the various standard symmetry packages: AFLOW-SYM, Spglib, FINDSYM, and Platon. The results are calculated with the most recent versions available for download:

- AFLOW version 3.1.169,
- Spglib version 1.10.2.4,
- FINDSYM version 5.1.0,
- Platon version 30118.

The default tolerances are employed as reported by the authors:

- AFLOW-SYM: $\epsilon_{\text{tight}} = d_{\text{c}}^{\text{nn(min)}}/100$,
- Spglib: `symprec` $= 1 \times 10^{-5}$ Å, `angle_tolerance` derives from `symprec` — default listed on web page [45],
- FINDSYM: $\epsilon_{\text{lattice}} = 1 \times 10^{-5}$ Å, $\epsilon_{\text{atomic position}} = 1 \times 10^{-3}$ Å— default from web interface [46],
- Platon: $\epsilon_{\text{metric}} = 1.00°$, $\epsilon_{\text{rotation}} = 0.25$ Å, $\epsilon_{\text{inversion}} = 0.25$ Å, $\epsilon_{\text{translation}} = 0.25$ Å [23].

Alternative tolerances values are also used for Spglib, FINDSYM, and Platon. In general, the alternative tolerances are 100 times the default tolerances, except in the case of Platon, where the default tolerances are divided by 100:

- Spglib: `symprec` $= 1 \times 10^{-3}$ Å
- FINDSYM: $\epsilon_{\text{lattice}} = 1 \times 10^{-3}$ Å, $\epsilon_{\text{atomic position}} = 1 \times 10^{-1}$ Å
- Platon: $\epsilon_{\text{metric}} = 0.01°$, $\epsilon_{\text{rotation}} = 2.5 \times 10^{-3}$ Å, $\epsilon_{\text{inversion}} = 2.5 \times 10^{-3}$ Å, $\epsilon_{\text{translation}} = 2.5 \times 10^{-3}$ Å.

The results from the alternative tolerances are denoted with $^+$.

A. Accuracy of space group analyses

The `CIF` files stored in the ICSD contain information such as the structural parameters and atomic species/-positions, as well as the space group (often reported from experiments), publication date, and citation. The experimentally reported space group information provides a unique validation opportunity for the various symmetry

package	# space group mismatches	# lattice mismatches	# crystal system mismatches	# space group not found
AFLOW-SYM	834	420	377	0
Spglib	10,022 (3,389)	9,644 (2,917)	9,523 (2,832)	0 (0)
FINDSYM	3,540 (1,067)	3,066 (531)	2,982 (483)	127 (156)
Platon	3,000 (1,217)	1,092 (588)	1,083 (486)	195 (1,351)‡

TABLE 2. **Mismatch counts between reported and calculated space groups for entries in the ICSD.** The test set is comprised of 54,015 ICSD entries stored in the aflow.org repository, as of 6 October 2017. The columns indicate the number of entries whose space group, lattice type, and crystal family do not match those reported by the ICSD. The results using the user-defined/non-default tolerance values for Spglib, FINDSYM, and Platon are shown in parentheses. For more details, refer to the Supplementary Information. The superscript ‡ indicates 2 entries for which the space group calculation exceeded 48 hours.

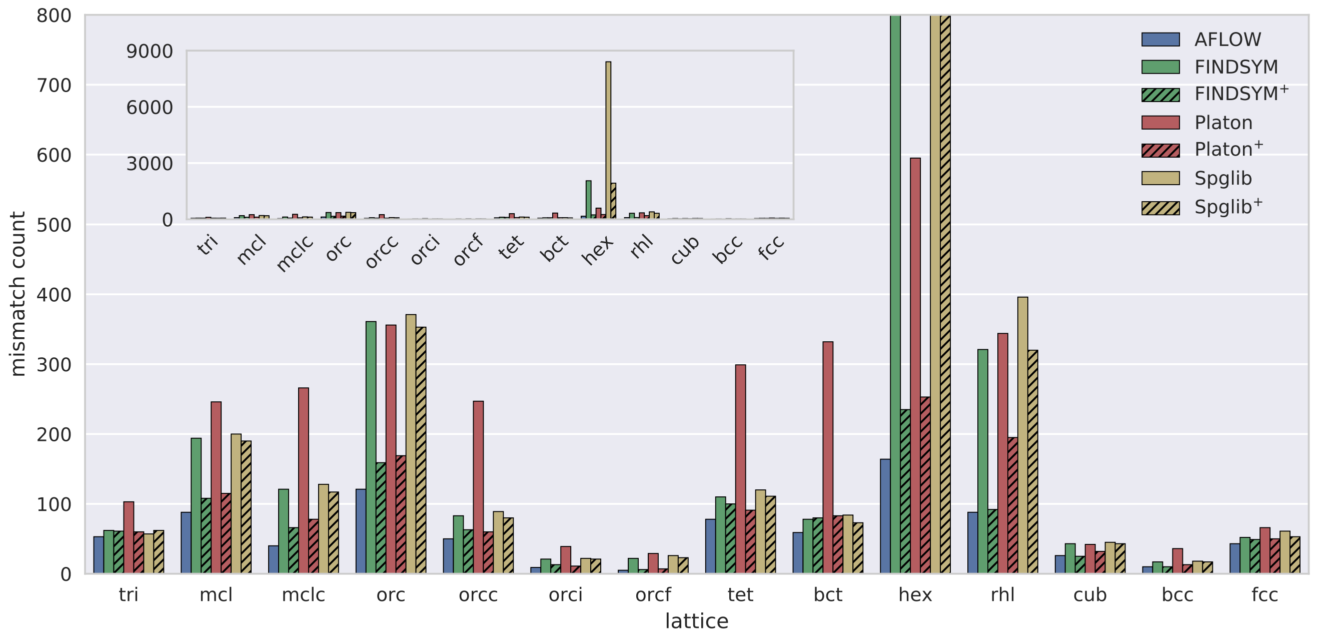

FIG. 6. **Breakdown of space group mismatches with the ICSD organized by lattice type.** The lattice types are derived from the space group number reported in the ICSD. The superscript + indicates the results using the user-defined/non-default tolerance values. The lattice abbreviations are as follows: triclinic (tri), monoclinic (mcl), base-centered monoclinic (mclc), orthorhombic (orc), base-centered orthorhombic (orcc), body-centered orthorhombic (orci), face-centered orthorhombic (orcf), tetragonal (tet), body-centered tetragonal (bct), hexagonal (hex), rhombohedral (rhl), cubic (cub), body-centered cubic (bcc), and face-centered cubic (fcc).

packages. The mismatch counts between the reported and calculated space groups are shown in Table 2. The counts are additionally broken down by lattice and crystal system (Figure 6) to highlight the severity of the mismatch. The full comparison of results is provided in the Supplementary Information, organized in tables by the reported crystal system, with mismatches highlighted in red.

AFLOW-SYM shows the best agreement with the ICSD with a deviation of about 1.5% (reduced to 1.3% if the mismatch is rectified at the loose tolerance). The mismatch is almost halved when comparing only the lattice and crystal systems, suggesting the algorithm found similar/nearby space groups (*e.g.*, see Figure 2). Using their respective default tolerances, Platon performs second best with a 5.6% deviation, followed by FINDSYM and Spglib with deviations of about 6.6% and 18.5%, respectively. With the alternative tolerances, the overall number of mismatches decrease for each package: Platon reduces to 2.3%, FINDSYM reduces to 2.0%, and Spglib reduces to 6.3%. Table 2 also shows that there are a number of systems for which FINDSYM and Platon are unable to identify any space group.

Figure 6 illustrates the space group mismatch from each package organized by lattice type. Overall, AFLOW-SYM is the most consistent with the ICSD for all lattice types for both the default and alternative tolerances, except for cubic systems where FINDSYM has one less mismatch than AFLOW-SYM using the alternative tolerance. The default tolerance values certainly play a role in the large deviation count, *e.g.*, a tighter tolerance can yield a lower symmetry than expected. This is evident with hexagonal and rhombohedral lattices, where Spglib resolves isomorphic subgroups neglecting the 3-/6-fold rotations (see Supplementary Information). However, increases in tolerance do not necessarily yield more consistent space group determinations. Figure 6 shows that the default tolerance is more accurate than the alternative tolerance for the triclinic (tri) and body-centered tetragonal (bct) systems calculated by Spglib and FINDSYM, respectively. To guarantee consistent symmetry results, users of Spglib, FINDSYM, and Platon should tune the tolerance for each system. The structure-specific tolerance choice and adaptive tolerance scheme incorporated into AFLOW-SYM allows for the automatic calculation of results that are generally consistent with experiments.

Overall, the results indicate the strength of the AFLOW-SYM approach. Other packages can reach similar performance of AFLOW-SYM, but they require continuous *ad-hoc* user adjustments of tolerances, possibly producing results incommensurate with other characteristics of the systems, such as its Pearson symbol. Only the self-consistent approach of AFLOW-SYM is ripe for the automation required by autonomous materials design.

B. Symmetry characterizations and representations

Of primary concern among the various standard packages is the identification and characterization of crystal symmetry, *i.e.*, a symmetry description considering the lattice and basis of atoms. In addition, AFLOW-SYM characterizes crystals with a sequence of symmetry-breaking features, including the lattice, superlattice (lattice with a uniform basis), crystal, and crystal-spin. With the progression of symmetry-breaking, each characterization offers a new dimension of physical insight, and is of particular importance for understanding complex phenomena [47]. The suite of characterizations[2] offered by each package is presented in Table 3. With integration into the automated framework AFLOW, new tools and symmetry descriptions will continue to be incorporated. The forums at `aflow.org/forum` are the venues for presenting updates and discussing new functionalities. Anticipated future work includes going beyond translationally invariant structures and characterizing disordered/off-stoichiometric structures [12, 48].

Furthermore, AFLOW-SYM presents the symmetry operations in a wealth of representations. Both AFLOW-SYM and Spglib explicitly offer representations for the symmetry operations.[3] Table 4 compares the operation representations provided by the two packages. Both provide the unit cell symmetry operators (coset representatives of the factor group). AFLOW-SYM offers the symmetry operations in the rotation matrix (cartesian and fractional), axis-angle, generator, and quaternion representations [49, 50]; while Spglib only provides the rotation matrix representation in its fractional form. AFLOW-SYM also presents the corresponding mappings for each symmetry operation, almost entirely eliminating the need to reapply the operators for symmetry-reduced analyses such as calculating the force constants [9, 42]. Along with the factor group coset representatives, AFLOW-SYM provides the lattice point group, reciprocal lattice point group, crystal point group, dual of the crystal point group, site point group, and space group symmetry operators. Catering to electronic structure calculations, AFLOW-SYM also returns additional symmetry information not explicitly provided by other routines, such as the Pearson symbol, Bravais lattice type, and Bravais lattice variation, necessary for constructing the most efficient Brillouin zone [4]. The full set of descriptions and representations offered by AFLOW-SYM is detailed in Appendix B.

4. USING AFLOW-SYM

A. Input/output formats

AFLOW-SYM reads crystal structure information from a geometry file containing the lattice vectors and atomic coordinates (coordinate model), which is treated as the *bona fide* representation of the structure. Information can be lost during the transcription of the X-ray diffraction/reflection data to the coordinate model, resulting in a lower symmetry profile. While a means to verify the two representations offers higher fidelity symmetry descriptions, the diffraction data is not nearly as accessible as the coordinate model representation. Furthermore, the geometry file is the *de facto* input format for *ab-initio* packages, and thus AFLOW-SYM resolves the material's symmetry based on this representation.

With AFLOW-SYM well-integrated into the high-throughput *ab-initio* software package AFLOW, it can

[2] Some packages provide more information than listed in Table 3. For example, Platon presents additional useful structural/chemical information such as bonding, coordination, planes, and torsions. However, the comparison presented in Table 3 is limited to symmetry information pertaining to space groups.

[3] FINDSYM and Platon do provide the general Wyckoff position, though they do not explicitly present the symmetry operators.

symmetry	AFLOW-SYM	Spglib	FINDSYM	Platon
lattice	✓			
superlattice	✓			✓ (`EQUAL`)
reciprocal lattice	✓			
crystal	✓	✓	✓	✓
crystal-spin	✓	✓	✓	

TABLE 3. **List of the symmetry descriptions provided by each of the four packages.** The superlattice analysis refers to the structure symmetry if each atomic site is decorated equally (same atom type), while crystal-spin indicates the structure symmetry including the magnetic moment of each atom. Adding the keyword EQUAL to the Platon command performs a superlattice analysis.

operator information	AFLOW-SYM	Spglib
operator type	✓	
Hermann-Mauguin	✓	
Schönflies	✓	
transformation matrix (cartesian)	✓	
transformation matrix (fractional)	✓	✓
generator matrix	✓	
so(3) coefficients $(\mathbf{L}_x, \mathbf{L}_y, \mathbf{L}_z)$	✓	
angle	✓	
axis	✓	
quaternion (vector)	✓	
quaternion (2×2 matrix)	✓	
quaternion (4×4 matrix)	✓	
su(2) coefficients (Pauli)	✓	
inversion boolean	✓	
internal translation (cartesian)	✓	
internal translation (fractional)	✓	✓
atom index map	✓	
atom type map	✓	
lattice translation (cartesian)	✓	
lattice translation (fractional)	✓	

TABLE 4. **List of operation representations provided by AFLOW-SYM compared to Spglib.** The internal translations are only applicable for the coset representative of the factor group and space group symmetry operators. Likewise, the lattice translations are only applicable for the space group symmetry operators.

process many standard input file types, including that of the ICSD/CSD [31–33] (`CIF`), VASP [51–54] (`POSCAR`), QUANTUM ESPRESSO [55], ABINIT [56], and FHI-AIMS [57].

Furthermore, all symmetry functions support the JSON object output format. This allows AFLOW-SYM to be employed from other programming languages such as Java, Go, Ruby, Julia and Python; facilitating smooth integration into numerous applications and workflows [19, 20]. These functionalities can be accessed by either the command line or a Python environment. A summary of the output for each command is provided in Appendix D 2.

B. Command line options

There are three main functions that provide all symmetry information for a given input structure. These functions allow an optional tolerance value (`tol`) to be specified via a number or the strings "tight" or "loose" corresponding to ϵ_{tight} and ϵ_{loose}, respectively. To perform the symmetry analysis of a crystal, the functions are called with the following commands:

- `aflow --aflowSYM[=<tol>] [--print=txt|json] < file`
  - Calculates and returns the symmetry operations for the lattice point group, reciprocal lattice point group, coset representatives of the factor group, crystal point group, dual of the crystal point group, site symmetry, and space group. It also returns the unique and equivalent sets of atoms.
- `aflow --edata[=<tol>] [--print=txt|json] < file`
  - Calculates and returns the extended crystallographic symmetry data (crystal, lattice, reciprocal lattice, and superlattice symmetry), while incorporating the full set of checks (see Section 2 F, 1-6) for robust symmetry determination.
- `aflow --sgdata[=<tol>] [--print=txt|json] < file`
  - Calculates and returns the space group symmetry of the crystal, while only validating the symmetry descriptions match with the ITC conventions (see Section 2 F, 6).

Square brackets `[...]` indicate optional arguments. The `--print` flag specifies the output format. The `--aflowSYM` function stores the isometries of the different symmetry groups to their own files `aflow.<group>.out` or `aflow.<group>.json`. The `<group>` labels are as follows: `pgroup` (lattice point group), `pgroupk` (reciprocal lattice point group), `fgroup` (coset representatives of the factor group), `pgroup_xtal` (crystal point group), `pgroupk_xtal`, (dual of the crystal point group), `agroup` (site symmetry), and `sgroup` (space group).

Crystal-spin symmetry functionality is also available in AFLOW-SYM. The magnetic moment of each site (collinear or non-collinear) can be specified for each of the commands listed above by adding the magnetic moment flag: `[--magmom=m1,m2,...|INCAR|OUTCAR]`. The magnetic moment information can be specified in three formats: **i.** explicitly via m_1, m_2, ... m_n in the same order as the input file (or $m_{1,x}$, $m_{1,y}$, $m_{1,z}$, $m_{2,x}$ ... $m_{n,z}$ for non-collinear), **ii.** read from the VASP INCAR or **iii.** the VASP OUTCAR. Magnetic moment readers for other *ab-initio* codes will be added in later versions.

C. Python environment

Given the recent prevalence of Python programming, we offer a module that employs AFLOW-SYM within a Python environment (see Appendix D 1). It connects to a local AFLOW installation and imports the AFLOW-SYM results into a `Symmetry` class. A `Symmetry` object is initialized with:

```
from aflow_sym import Symmetry
from pprint import pprint

with open('test.poscar', 'r') as input_file:
  sym = Symmetry(aflow_executable='./aflow')
  output = sym.get_edata(input_file)
  pprint(output)
```

By default, the `Symmetry` object searches for an AFLOW executable in the `PATH`. However, the location of an AFLOW executable can be specified as follows:

`Symmetry(aflow_executable='your_executable')`. The symmetry object has three built-in methods, which correspond to the command line calls mentioned previously:

- `get_symmetry(input_file, tol, magmoms)`
- `get_edata(input_file, tol, magmoms)`
- `get_sgdata(input_file, tol, magmoms)`

Each method requires a Python file handler (`input_file`), while the tolerance (`tol`) and magnetic moments of each site (`magmoms`) are optional arguments.

D. AFLOW-SYM support

Functionality requests and bug reports should be posted on the AFLOW Forum `aflow.org/forum` under the board "Symmetry analysis".

5. CONCLUSION

In this article, we present AFLOW-SYM, a symmetry platform catered to — but not limited to — high-throughput frameworks. We address problems stemming from numerical tolerance in symmetry analyses by using a mapping procedure uniquely designed to handle skewed cells and an advanced adaptive tolerance scheme. AFLOW-SYM also includes consistency checks of calculated isometries with respect to symmetry principles. These solutions are validated against the experimental structures data reported by the ICSD. Comparison with other symmetry analysis suites, Spglib, FINDSYM, and

Platon, shows that AFLOW-SYM is the most consistent with the ICSD.

For general use of AFLOW-SYM, the routines include both a standard text output and a JSON output for easy integration into other computational workflows. Lastly, a comprehensive list of the symmetry descriptions are presented (see Appendix D 2), illustrating the vast symmetry information available to users of AFLOW-SYM.

6. ACKNOWLEDGMENTS

DH acknowledges support from the Department of Defense through the National Defense Science and Engineering Graduate (NDSEG) Fellowship Program. CO acknowledges support from the National Science Foundation Graduate Research Fellowship under Grant No. DGF1106401. SC acknowledges support by DOD-ONR (N00014-17-1-2090) and the Alexander von Humboldt-Foundation.

Appendix A: Crystallographic symmetry

1. Mathematical group

A group is an abstract mathematical structure comprised of a set of elements (g), and an operation that combines two elements to form a third [58]. There are four axioms that a group satisfies.

1. closure: The combination of two elements with the operator yields an element that exists in the set; it does not create a new element outside the set.

2. associativity: The order of combining elements with the operator is inconsequential given the sequence of operands is unaltered.

3. identity: There exists a neutral element (I) that when combined with another element leaves that element unchanged ($gI = g$).

4. inverse: For each element g in the set, there exists a corresponding inverse element g^{-1}, such that $gg^{-1} = I$.

An abelian group includes the additional axiom of commutativity. These rules are the foundation of group theory and underline the construction of the different symmetry groups.

2. Point group

A point group is a set of symmetry transformations about a fixed point $\{\mathbf{U}_1, \mathbf{U}_2, \ldots, \mathbf{U}_n\}$ that leave a system invariant[4]. The elements of the group are classified as **i.** n-fold rotations, where n describes the rotation order (*i.e.*, the number of symmetric points it generates), **ii.** inversions, and **iii.** roto-inversions — compound operations comprised of a rotation and inversion. Three dimensional crystals are confined to one of 32 point groups due to the crystallographic restriction theorem, which limits the rotation order in a periodic system to two-, three-, four-, and six-fold [36]. The 32 crystallographic point groups are categorized into one of seven crystal systems: cubic, hexagonal, trigonal, tetragonal, orthorhombic, monoclinic, and triclinic. The classifications are based on the lattice parameters (a, b, c, α, β, γ) of the crystal.

- **cubic:** $a = b = c$, $\alpha = \beta = \gamma = 90°$
- **hexagonal/trigonal:** $a = b \neq c$, $\alpha = \beta = 90°, \gamma = 120°$
- **tetragonal:** $a = b \neq c$, $\alpha = \beta = \gamma = 90°$
- **orthorhombic:** $a \neq b \neq c$, $\alpha = \beta = \gamma = 90°$
- **monoclinic:** $a \neq b \neq c$, $\alpha = \gamma = 90°, \beta \neq 90°$
- **triclinic:** $a \neq b \neq c$, $\alpha \neq \beta \neq \gamma \neq 90°$

In crystallography, two types of point groups are of particular importance — the lattice and crystal (vector) point group. Each operates in a different space: the lattice point group characterizes the symmetry of the lattice points (an affine space), while the crystal point group additionally considers the atomic basis and acts on the underlying vector space of the crystal face normals. Fundamentally, the vector space captures the symmetry of the macroscopic crystal [36]. The crystal point group operations are defined as the linear mappings of the vector space, *i.e.*, the unique set of fixed-point transformations of the factor group[5] [36, 59]. Owing to symmetry breaking from the atomic basis, the cardinality of the crystal point group is at most as large as that of the lattice. Furthermore, the dual (reciprocal) counterparts of the lattice and crystal point group play an important role in electronic structure theory: resolving the symmetries of the Brillouin and irreducible Brillouin zones, respectively. In AFLOW-SYM, the output for the lattice, reciprocal lattice, crystal, and dual of the crystal point group operations are labeled `pgroup`, `pgroupk`, `pgroup_xtal`, and `pgroupk_xtal`, respectively.

[4] Here, the rotation matrix $\mathbf{U}$ is used to represent the different symmetry groups; however, all rotation elements can also be described in axis-angle, matrix generator, and quaternion form.

[5] Without the relevant internal translations (complete coset representatives), the crystal point group operations do not generally apply in the affine point space (lattice points and atoms), as is the case for non-symmorphic space groups. Conversely, the set of operations that do apply in the point space define the site symmetries.

3. Space group

In periodic systems, translational symmetry gives rise to another mathematical group — the space group. Its elements are comprised of those found in the point group, along with glide (mirror and translation) and screw (rotation and translation) operations. The translational degree of freedom extends the number of unique sets of symmetry operations to 230. The translations of a crystal are divided into lattice translations ($\mathbf{T}$) and internal translations ($\mathbf{t}$).

$$\{\mathbf{U}_1, \mathbf{U}_2, \ldots, \mathbf{U}_n | \mathbf{T} + \mathbf{t}\} \tag{A1}$$

Subsequently, a space group describes the full symmetry of a periodic system. The space group operations are labeled `sgroup` in AFLOW-SYM.

4. Factor group

From the space group, the elements of the factor group are defined as the cosets of the subgroup of lattice translations ($\mathbf{T}$):

$$\{\mathbf{I}|\mathbf{0}\}\,\{\mathbf{I}|\mathbf{T}\}\,,\ \{\mathbf{U}_i|\mathbf{t}_i\}\,\{\mathbf{I}|\mathbf{T}\}\,,\ \{\mathbf{U}_j|\mathbf{t}_j\}\,\{\mathbf{I},\mathbf{T}\}\,,\ \ldots\,, \tag{A2}$$

where $\mathbf{U}_i$ are the point group operations, $\mathbf{t}_i$ are the associated internal translation, and $\mathbf{I}$ is the identity. The unit cell symmetry is exposed via the coset representatives:

$$\{\mathbf{I}|\mathbf{0}\}\,,\,\{\mathbf{U}_i|\mathbf{t}_i\}\,,\,\{\mathbf{U}_j|\mathbf{t}_j\}\,,\,\ldots\,. \tag{A3}$$

The coset representatives themselves do not necessarily form a mathematical group, since they violate the closure condition. Repeated application of an internal translation will eventually traverse beyond the unit cell. The unit cell symmetry elements (coset representatives) are labeled `fgroup` in AFLOW-SYM.

In general, there exists a homomorphism between the factor group and the crystal point group, *i.e.*, the factor group cardinality is an integer multiple of the crystal point group cardinality. The multiplicative factor (m) is dictated by the number of internal translations in the system. A crystal in a primitive representation exhibits an isomorphic correspondence ($m = 1$), while non-primitive representations possess the general homomorphic relationship ($m > 1$).

5. Site point group

The site point group — or site symmetry — describes the point group symmetry centered on a single site in the crystal, revealing the local symmetry environment. The analysis is performed on each atomic site in the crystal, with symmetrically equivalent atoms (Wyckoff positions) exhibiting the same point group symmetries. The origin of the fixed-point operations differentiates the site symmetry from the lattice/crystal point group, which are centered on the unit cell origin. In the finite difference method for calculating phonons, the unique distortions for a given atomic site are resolved with its site symmetry [9, 42]. In AFLOW-SYM, the site symmetry elements are designated by `agroup` ("atomic site group").

6. Crystal-spin symmetry

Introducing the spin degree of freedom can break crystal symmetry. AFLOW-SYM includes functionality for a crystal-spin (lattice, atoms, and spin) description, including the relevant point group, factor group, space group, and site symmetry operations. For magnetic systems, these are the symmetry descriptions employed by *ab-initio* packages, such as VASP [51–54]. Note that the crystal-spin symmetry differs from the magnetic symmetry, which accounts for time-reversal symmetry (spin-flips). The magnetic symmetry will be incorporated into AFLOW-SYM in a later version.

Appendix B: Mathematical representation of symmetry

Symmetry elements are characterized into three types of transformation: translation, fixed-point, and fixed-point-free (a combination of the two, *i.e.*, screw and glide operations) [36]. Translation are generally indicated by 3×1 vectors

$$\mathbf{t} = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix}. \tag{B1}$$

Fixed-point symmetries O(3) describing rotations, inversions, and roto-inversions are represented by rotation matrices. The rotation symmetries SO(3), *i.e.*, a subgroup of the orthogonal group O(3), can be represented in three additional forms: axis-angle, matrix generator, and quaternions. AFLOW-SYM provides the symmetry operations for rotations in each of these four forms, which are discrete subgroups of the continuous SO(3) group.

1. Rotation matrix

A rotation matrix describes a transformation between two reference frames. In three-dimensions, the symmetry operators are 3×3 square matrices with the following form

$$\mathbf{U} = \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \\ u_{31} & u_{32} & u_{33} \end{pmatrix}. \tag{B2}$$

All transformations are unitary (norm preserving), and therefore have $\det(\mathbf{U}) = \pm 1$. The matrix representation affords fast computation through use of optimized linear algebra computational packages.

2. Axis-angle

Rotation operations are also characterized by their axis and angle of rotation. The axis, $\hat{\mathbf{r}} = (r_1, r_2, r_3)$, indicates the direction of the rotation operator, pointing perpendicular to the fixed point motion. The angle, θ, specifies the magnitude of the rotational motion (following the right-hand rule). The angle and axis components are related to the matrix elements of $\mathbf{U}$ by

$$\begin{aligned}
\theta &= \cos^{-1}\left(\frac{\mathrm{Tr}(\mathbf{U})-1}{2}\right), \\
r_d &= \sqrt{(u_{32}-u_{23})^2+(u_{13}-u_{31})^2+(u_{21}-u_{12})^2}, \\
r_1 &= \frac{u_{32}-u_{23}}{r_d},\ r_2 = \frac{u_{13}-u_{31}}{r_d},\ r_3 = \frac{u_{21}-u_{12}}{r_d},
\end{aligned} \tag{B3}$$

where $\mathrm{Tr}(\mathbf{U})$ is the trace of $\mathbf{U}$. The axis-angle representation is directly applied to a point $\mathbf{p}$ via Rodrigues' rotation formula

$$\mathbf{p}_{\mathrm{rot}} = \mathbf{p}\cos\theta + (\hat{\mathbf{r}}\times\mathbf{p})\sin\theta + \hat{\mathbf{r}}\,(\hat{\mathbf{r}}\cdot\mathbf{p})\,(1-\cos\theta)\,, \tag{B4}$$

where $\mathbf{p}_{\mathrm{rot}}$ is the rotated point. This description highlights the operation order n via $n = 360°/\theta$ and identifies the conventional cell lattice vectors, since they are parallel to certain symmetry axes.

3. Matrix generator

The Lie group SO(3) grants the use of the corresponding Lie algebra so(3), which are comprised of the infinitesimal matrix generators $\mathbf{G}$. The generator is a skew-symmetric matrix that describes the rotation about a symmetry axis, with the following form:

$$\mathbf{G} = \begin{pmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_2 & r_1 & 0 \end{pmatrix}, \tag{B5}$$

where r_1, r_2, r_3 are the components of the symmetry unit axis $\hat{\mathbf{r}}$. The identity and inverse elements have no axis; therefore, the generator is not defined and is returned as a zero matrix. While the rotation matrix transforms one reference frame to another, the generator operates about a single axis. The matrix exponential of the generator with the angle maps the operations into the rotation matrix form ($\mathbf{U} = \exp(\theta\mathbf{G})$). For convenience, AFLOW-SYM returns the generator multiplied with the angle $\mathbf{A} = \theta\mathbf{G}$. AFLOW-SYM also provides the expansion coefficients of the generator matrix onto the following so(3) basis:

$$\mathbf{G} = x\mathbf{L}_x + y\mathbf{L}_y + z\mathbf{L}_z \tag{B6}$$

where

$$\begin{aligned}
\mathbf{L}_x &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix},\ \mathbf{L}_y = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} \\
\mathbf{L}_z &= \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.
\end{aligned} \tag{B7}$$

The expansion coefficients x, y, and z of this basis set are the unit axis components r_1, r_2, and r_3, respectively.

4. Quaternion

A quaternion is a mathematical representation of 3D space with both real and imaginary components. Though developed in 1843, the quaternion has only recently gained relevance through the field of computer graphics and modeling. As opposed to using a nine-element 3×3 matrix to represent a rotation in space, quaternions have a concise format consisting of four components. The reduced element count increases computational efficiency, and thus is particularly suitable for high-throughput frameworks.

Given an axis and angle, the corresponding quaternion representation, $\mathbf{q} = (q_0, q_1, q_2, q_3)$, is

$$\begin{aligned}
q_0 &= \cos(\theta/2), \\
q_1 &= r_1\sin(\theta/2), \\
q_2 &= r_2\sin(\theta/2), \\
q_3 &= r_3\sin(\theta/2),
\end{aligned} \tag{B8}$$

which are equivalent to the Euler parameters. Alternate forms of the quaternion are 2×2 and 4×4 matrices. The complex 2×2 unitary matrix of a quaternion is

$$\mathbf{C} = \begin{pmatrix} q_0+q_3 i & q_2+q_1 i \\ -q_2+q_1 i & q_0-q_3 i \end{pmatrix}, \tag{B9}$$

which is an element of the SU(2) Lie group. The $\mathbf{C}$ matrix can be expanded onto a basis formed by the Pauli matrices:

$$\boldsymbol{\sigma}_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},\ \boldsymbol{\sigma}_2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix},\ \boldsymbol{\sigma}_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \tag{B10}$$

where multiplying by i ($=\sqrt{-1}$) yields the following decomposition:

$$\mathbf{C} = q_0\mathbf{I} + q_1 i\boldsymbol{\sigma}_1 + q_2 i\boldsymbol{\sigma}_2 + q_3 i\boldsymbol{\sigma}_3. \tag{B11}$$

The corresponding Lie algebra, su(2), is [60]

$$\mathbf{g} = \frac{i}{2}\begin{pmatrix} r_3 & r_1-r_2 i \\ r_1+r_2 i & -r_3 \end{pmatrix}. \tag{B12}$$

AFLOW-SYM lists the su(2) generator coefficients expanded on the Pauli matrices

$$\mathbf{g} = x\boldsymbol{\sigma}_1 + y\boldsymbol{\sigma}_2 + z\boldsymbol{\sigma}_3, \tag{B13}$$

where the expansion coefficients x, y, and z are $(i/2)r_1$, $(i/2)r_2$, and $(i/2)r_3$, respectively. Similar to the SO(3) rotations, the matrix exponential of the su(2) generator $\mathbf{g}$ with the angle maps the operations into the complex 2×2 SU(2) matrix ($\mathbf{C} = \exp(\theta\mathbf{g})$). The 4×4 matrix representation of the quaternion is

$$\mathbf{Q} = \begin{pmatrix} q_0 & q_1 & q_2 & q_3 \\ -q_1 & q_0 & -q_3 & q_2 \\ -q_2 & q_3 & q_0 & -q_1 \\ -q_3 & -q_2 & q_1 & q_0 \end{pmatrix}; \tag{B14}$$

which includes all four components of the quaternion vector in a matrix, allowing transformations to be performed through matrix multiplication rather than quaternion algebra. This method is useful for performing operations with other transformations in matrix or vector form, whereas the quaternion vector notation has its own algebra similar to the operations between complex numbers with an additional scalar component (q_0).

5. Basis transformations of operators

The representations of the symmetry operations are basis dependent and are customarily given with respect to cartesian or fractional coordinates systems. It is straight-forward to transform symmetry operations between these vector spaces via a basis change. In matrix notation, the fixed-point operation in cartesian ($\mathbf{U}_\mathrm{c}$) and fractional ($\mathbf{U}_\mathrm{f}$) coordinates are related via the following similarity transformations:

$$\begin{aligned} \mathbf{U}_\mathrm{f} &= \mathbf{L}^{-1}\mathbf{U}_\mathrm{c}\mathbf{L}, \\ \mathbf{U}_\mathrm{c} &= \mathbf{L}\mathbf{U}_\mathrm{f}\mathbf{L}^{-1}. \end{aligned} \tag{B15}$$

Here, $\mathbf{L}$ is the column-space form of the lattice vectors:

$$\mathbf{L} = \begin{pmatrix} \mathbf{a} & \mathbf{b} & \mathbf{c} \end{pmatrix} = \begin{pmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{pmatrix}, \tag{B16}$$

where a_i, b_i, and c_i are the corresponding components of the lattice vectors. A translation vector $\mathbf{t}_{\mathrm{c(f)}}$ is transformed between cartesian and fractional coordinates by $\mathbf{t}_\mathrm{f} = \mathbf{L}^{-1}\mathbf{t}_\mathrm{c}$ and $\mathbf{t}_\mathrm{c} = \mathbf{L}\mathbf{t}_\mathrm{f}$.

6. Example representations

An example of a three-fold rotation in cartesian coordinates is shown below in its rotation matrix, axis-angle, matrix generator, and quaternion vector and matrix representations.

$$\begin{aligned} \mathbf{U}_{\text{3-fold}} &= \begin{pmatrix} 0 & \text{-1} & 0 \\ 0 & 0 & \text{-1} \\ 1 & 0 & 0 \end{pmatrix} \\ \hat{\mathbf{r}} &= (0.57735, -0.57735, 0.57735) \\ \theta &= 120^\circ \\ \mathbf{A} &= \begin{pmatrix} 0.0 & -1.2092 & -1.2092 \\ 1.2092 & 0.0 & -1.2092 \\ 1.2092 & 1.2092 & 0.0 \end{pmatrix} \\ \mathbf{q} &= (0.5, 0.5, -0.5, 0.5) \\ \mathbf{C} &= \begin{pmatrix} 0.5 + 0.5i & -0.5 + 0.5i \\ 0.5 + 0.5i & 0.5 - 0.5i \end{pmatrix} \\ \mathbf{Q} &= \begin{pmatrix} 0.5 & 0.5 & -0.5 & 0.5 \\ -0.5 & 0.5 & -0.5 & -0.5 \\ 0.5 & 0.5 & 0.5 & -0.5 \\ -0.5 & 0.5 & 0.5 & 0.5 \end{pmatrix} \end{aligned}$$

Appendix C: Extreme cases of minimal distance discrepancy between cartesian and fractional spaces

The bring-in-cell procedure applied to a crystal with lattice parameters $a = b = c = 5$ Å, $\alpha = \gamma = 90^\circ$ and $\beta = 60^\circ$ identifies the minimum distance between the fractional coordinates $(0, 0, 1/2)$ and $(1/2, 0, 0)$ to be $||\widetilde{\mathbf{d}}_\mathrm{c}^\mathrm{min}|| = 4.3301$ Å, compared to the true minimum of $||\mathbf{d}_\mathrm{c}^\mathrm{min}|| = 2.5$ Å. A more extreme mismatch occurs if $\beta = 5^\circ$, yielding a minimum of $||\widetilde{\mathbf{d}}_\mathrm{c}^\mathrm{min}|| = 4.9952$ Å with the bring-in-cell method, differing significantly from the true minimum of $||\mathbf{d}_\mathrm{c}^\mathrm{min}|| = 0.2181$ Å. Applying the heuristic threshold to the aforementioned skewed examples give bounds of $\epsilon_\mathrm{max} = 1.2130$ Å (with $d_\mathrm{c}^\mathrm{nn(min)} = 2.4259$ Å) and $\epsilon_\mathrm{max} = 0.0017$ Å (with $d_\mathrm{c}^\mathrm{nn(min)} = 0.4362$ Å) for $\beta = 60^\circ$ and $\beta = 5^\circ$, respectively. Both thresholds are sufficiently below the true minimum distances — even in the worst cases — validating our choice of the heuristic threshold.

Appendix D: AFLOW-SYM details

1. Python module

The module to run the AFLOW-SYM commands referenced in Section 4 C is provided below.

```python
import json
import subprocess
import os

class Symmetry:

    def __init__(self, aflow_executable='aflow'):
        self.aflow_executable = aflow_executable
```

```
def aflow_command(self, cmd):
    try:
        return subprocess.check_output(
            self.aflow_executable + cmd,
            shell=True
        )
    except subprocess.CalledProcessError:
        print "Error aflow executable not found
          ↪  at: " + self.aflow_executable

def get_symmetry(self, input_file, tol=None,
    ↪ magmoms=None):
    fpath = os.path.realpath(input_file.name)
    command = ' --aflowSYM'
    output = ''

    if tol:
        command += '=' + str(tol)
    if magmoms:
        command += ' --magmom=' + magmoms

    output = self.aflow_command(
        command + ' --print=json --screen_only'
          ↪  + ' < ' + fpath
    )
    res_json = json.loads(output)
    return res_json

def get_edata(self, input_file, tol=None,
    ↪ magmoms=None):
    fpath = os.path.realpath(input_file.name)
    command = ' --edata'
    output = ''

    if tol:
        command += '=' + str(tol)
    if magmoms:
        command += ' --magmom=' + magmoms

    output = self.aflow_command(
        command + ' --print=json' + ' < ' +
          ↪ fpath
    )
    res_json = json.loads(output)
    return res_json

def get_sgdata(self, input_file, tol=None,
    ↪ magmoms=None):
    fpath = os.path.realpath(input_file.name)
    command = ' --sgdata'
    output = ''

    if tol:
        command += '=' + str(tol)
    if magmoms:
        command += ' --magmom=' + magmoms

    output = self.aflow_command(
        command + ' --print=json' + ' < ' +
          ↪ fpath
    )
    res_json = json.loads(output)
    return res_json
```

2. Output list

This section details the output fields for the symmetry group operations, extended crystallographic data (`edata`), and space group data (`sgdata`) routines. The lists describe the keywords as they appear in the JSON format. Similar keywords are used for the standard text output.

Symmetry operations output.

- `pgroup`
  - *Description:* lattice point group symmetry operations.
  - *Type:* `array of symmetry operator objects`
- `pgroupk`
  - *Description:* reciprocal lattice point group symmetry operations.
  - *Type:* `array of symmetry operator objects`
- `fgroup`
  - *Description:* coset representative of factor group symmetry operations.
  - *Type:* `array of symmetry operator objects`
- `pgroup_xtal`
  - *Description:* crystal point group symmetry operations.
  - *Type:* `array of symmetry operator objects`
- `pgroupk_xtal`
  - *Description:* dual of the crystal point group symmetry operations.
  - *Type:* `array of symmetry operator objects`
- `sgroup`
  - *Description:* space group symmetry operations out to a given radius.
  - *Type:* `array of symmetry operator objects`
- `iatoms`
  - *Description:* groupings of symmetrically equivalent/unique atoms.
  - *Type:* `iatom object`
- `agroup`
  - *Description:* site (atom) symmetry operations (point group).
  - *Type:* `array of symmetry operator objects`

Each symmetry group contains an array of symmetry objects, including the operation representations listed in Table 4. The `symmetry operator` object contains the following:

- `Hermann_Mauguin`
  - *Description:* Hermann-Mauguin symbol of the symmetry operation.
  - *Type:* `string`

- `Schoenflies`
  - *Description:* Schönflies symbol of the symmetry operation.
  - *Type:* `string`
- `Uc`
  - *Description:* transformation matrix with respect to cartesian coordinates.
  - *Type:* 3×3 `array`
- `Uf`
  - *Description:* transformation matrix with respect to fractional coordinates.
  - *Type:* 3×3 `array`
- `angle`
  - *Description:* angle corresponding to symmetry operation.
  - *Type:* `float`
- `axis`
  - *Description:* axis of symmetry operation.
  - *Type:* 3×1 `array`
- `generator`
  - *Description:* matrix generator of symmetry operation.
  - *Type:* 3×3 `array`
- `generator_coefficients`
  - *Description:* matrix generator expansion coefficients onto $\mathbf{L}_x$, $\mathbf{L}_y$, and $\mathbf{L}_z$ basis.
  - *Type:* 3×1 `array`
- `group`
  - *Description:* specifies the group type (“`pgroup`”, “`pgroupk`“, “`fgroup`”, “`pgroup_xtal`”, “`pgroupk_xtal`, “`sgroup`”, and “`agroup`”).
  - *Type:* `string`
- `inversion`
  - *Description:* indicates if inversion exists.
  - *Type:* `bool`
- `quaternion_matrix`
  - *Description:* quaternion matrix.
  - *Type:* 4×4 `array`
- `SU2_matrix`
  - *Description:* complex quaternion matrix; element of SU(2).
  - *Type:* 2×2 `array`
- `su2_coefficients`
  - *Description:* su(2) generator coefficients onto Pauli matrices ($\boldsymbol{\sigma}_1$, $\boldsymbol{\sigma}_2$, and $\boldsymbol{\sigma}_3$).
  - *Type:* 3×1 `array`
- `quaternion_vector`
  - *Description:* quaternion vector.
  - *Type:* 4×1 `array`
- `type`
  - *Description:* point group operation type (unity, rotation, inversion, or roto-inversion).
  - *Type:* `string`
- `ctau`
  - *Description:* internal translation component in cartesian coordinates (“`fgroup`” and “`sgroup`” only).
  - *Type:* 3×1 `array`
- `ftau`
  - *Description:* internal translation component in fractional coordinates (“`fgroup`” and “`sgroup`” only).
  - *Type:* 3×1 `array`
- `ctrasl`
  - *Description:* lattice translation component in cartesian coordinates (“`sgroup`” only).
  - *Type:* 3×1 `array`
- `ftrasl`
  - *Description:* lattice translation component in fractional coordinates (“`sgroup`” only).
  - *Type:* 3×1 `array`

The `iatom` object contains:

- `inequivalent_atoms`
  - *Description:* symmetrically distinct atom indices.
  - *Type:* `array`
- `equivalent_atoms`
  - *Description:* groupings of symmetrically equivalent atom indices.
  - *Type:* `2D array`

edata output.

- `lattice_parameters`
  - *Description:* lattice parameters in units of Angstroms and degrees $(a, b, c, \alpha, \beta, \gamma)$.
  - *Type:* 6×1 `array`
  - *Similar to:*
    - ◇ FINDSYM: `Lattice parameters, a, b, c, alpha, beta, gamma:`
    - ◇ Platon: first six fields in the line containing `CELL`
- `lattice_parameters_Bohr_deg`
  - *Description:* lattice parameters in units of Bohr and degrees $(a, b, c, \alpha, \beta, \gamma)$.
  - *Type:* 6×1 `array`
- `volume`
  - *Description:* real space cell volume.
  - *Type:* `float`
  - *Similar to:*
    - ◇ Platon: last field in the line containing `CELL`.
- `c_over_a`
  - *Description:* ratio of c and a lattice parameters.
  - *Type:* `float`
- `Bravais_lattice_type`
  - *Description:* Bravais lattice of the crystal (“FCC”, “BCC”, “CUB”, “HEX”, “RHL”, etc.).
  - *Type:* `string`
- `Bravais_lattice_variation_type`
  - *Description:* lattice variation type of the crystal in the AFLOW standard [4].
  - *Type:* `string`
- `Bravais_lattice_system`
  - *Description:* Bravais lattice of the crystal.
  - *Type:* `string`
  - *Similar to:*
    - ◇ Platon: `CrystalSystem` column in `Cell Lattice` table.
- `Pearson_symbol`

- *Description:* Pearson symbol of the crystal.
- *Type:* `string`

- `crystal_family`
  - *Description:* crystal family.
  - *Type:* `string`
- `crystal_system`
  - *Description:* crystal system.
  - *Type:* `string`
- `point_group_Hermann_Mauguin`
  - *Description:* Hermann-Mauguin symbol corresponding to the point group of the crystal.
  - *Type:* `string`
  - *Similar to:*
    - ◇ Spglib: `SpglibDataset.pointgroup_symbol`.
- `point_group_Schoenflies`
  - *Description:* Schönflies symbol for the point group of the crystal.
  - *Type:* `string`
- `point_group_orbifold`
  - *Description:* orbifold of the point group.
  - *Type:* `string`
- `point_group_type`
  - *Description:* point group type of the crystal.
  - *Type:* `string`
- `point_group_order`
  - *Description:* number of point group operations describing the crystal.
  - *Type:* `int`
- `point_group_structure`
  - *Description:* point group structure of the crystal.
  - *Type:* `string`
- `Laue`
  - *Description:* Laue symbol of the crystal.
  - *Type:* `string`
  - *Similar to:*
    - ◇ Platon: field after the line containing `Laue`.
- `crystal_class`
  - *Description:* crystal class.
  - *Type:* `string`
- `space_group_number`
  - *Description:* space group number.
  - *Type:* `int`
  - *Similar to:*
    - ◇ Spglib: `SpglibDataset.spacegroup_number`.
    - ◇ FINDSYM: field after line containing `_symmetry_Int_Tables_number`.
    - ◇ Platon: field after line containing `No` (number).
- `space_group_Hermann_Mauguin`
  - *Description:* Hermann-Mauguin space group label.
  - *Type:* `string`
  - *Similar to:*
    - ◇ Spglib: `SpglibDataset.International_symbol`.
    - ◇ FINDSYM: field after line containing `_symmetry_space_group_name_H-M`.
    - ◇ Platon: field after line containing `Space Group  H-M`.
- `space_group_Hall`
  - *Description:* Hall space group label.
  - *Type:* `string`
  - *Similar to:*
    - ◇ Spglib: `SpglibDataset.hall_symbol`.
    - ◇ FINDSYM: field after line containing `_space_group.reference_setting`.
    - ◇ Platon: field after line containing `Space group - Hall`.
- `space_group_Schoenflies`
  - *Description:* Schönflies space group label.
  - *Type:* `string`
  - *Similar to:*
    - ◇ Spglib: `Spg_get_schoenflies`.
    - ◇ FINDSYM: second field after line containing `Space Group`.
    - ◇ Platon: field after line containing `Schoenflies`.
- `setting_ITC`
  - *Description:* ITC setting of conventional cell (AFLOW-SYM defaults to the first setting that appears in the ITC and the hexagonal setting for rhombohedral systems).
  - *Type:* `int`
  - *Similar to:*
    - ◇ Spglib: `SpglibDataset.choice`.
- `origin_ITC`
  - *Description:* corresponding origin shift of the crystal to align with the ITC representation.
  - *Type:* 3×1 `array`
  - *Similar to:*
    - ◇ Spglib: `SpglibDataset.choice`.
    - ◇ FINDSYM: field after line containing `Origin at`.
    - ◇ Platon: field after line containing `Origin Shifted to`.
- `general_position_ITC`
  - *Description:* general Wyckoff position (x, y, z) as indicated by the ITC.
  - *Type:* `2D array`
  - *Similar to:*
    - ◇ FINDSYM: field after line containing `_space_group_symop_operation_xyz`.
    - ◇ Platon: in the `Symmetry Operation(s)` table.
- `Wyckoff_positions`
  - *Description:* indicates the Wyckoff, letter, multiplicity, site symmetry, position (3×1 array), and atom name.
  - *Type:* `array of objects`
  - *Similar to:*
    - ◇ Spglib: `get_symmetry_dataset.wyckoffs` (letters only).
    - ◇ FINDSYM: in the loop with `_atom` prefix.
- `Bravais_lattice_lattice_type`
  - *Description:* Bravais lattice of the lattice.
  - *Type:* `string`
- `Bravais_lattice_lattice_variation_type`
  - *Description:* lattice variation type of the lattice in the AFLOW standard [4].
  - *Type:* `string`
- `Bravais_lattice_lattice_system`
  - *Description:* Bravais lattice system of the lattice.

  – *Type:* `string`
- `Bravais_superlattice_lattice_type`
  – *Description:* Bravais lattice of the superlattice.
  – *Type:* `string`
- `Bravais_superlattice_lattice_variation_type`
  – *Description:* lattice variation type of the superlattice in the AFLOW standard [4].
  – *Type:* `string`
- `Bravais_superlattice_lattice_system`
  – *Description:* Bravais lattice system of the superlattice.
  – *Type:* `string`
- `Pearson_symbol_superlattice`
  – *Description:* Pearson symbol of the superlattice.
  – *Type:* `string`
- `reciprocal_lattice_vectors`
  – *Description:* reciprocal lattice vectors.
  – *Type:* 3×3 `array`
- `reciprocal_lattice_parameters`
  – *Description:* reciprocal lattice parameters $(a, b, c, \alpha, \beta, \gamma)$.
  – *Type:* 6×1 `array`
- `reciprocal_volume`
  – *Description:* reciprocal cell volume.
  – *Type:* `float`
- `reciprocal_lattice_type`
  – *Description:* Bravais lattice of the reciprocal lattice ("FCC", "BCC", "CUB", "HEX", "RHL", etc.).
  – *Type:* `string`
- `reciprocal_lattice_variation_type`
  – *Description:* lattice variation type of the reciprocal lattice in the AFLOW standard [4].
  – *Type:* `string`
- `reciprocal_lattice_system`
  – *Description:* lattice system of the reciprocal lattice.
  – *Type:* `string`
- `standard_primitive_structure`
  – *Description:* AFLOW standard primitive crystal structure representation.
  – *Type:* `structure object`
- `standard_conventional_structure`
  – *Description:* AFLOW standard conventional crystal structure representation.
  – *Type:* `structure object`
- `wyccar`
  – *Description:* ITC conventional crystal structure representation.
  – *Type:* `structure object`
  – *Similar to:*
    ◇ Spglib: `Spg_standardize_cell(to_primitive=0)`.
    ◇ FINDSYM: after `Space Group` line.

The `structure object` lists the following information regarding the crystal structure:
- `title`
  – *Description:* geometry file title.
  – *Type:* `string`
- `scale`
  – *Description:* scaling factor of lattice vectors.
  – *Type:* `float`
- `lattice`
  – *Description:* row-space representation of lattice vectors $(\mathbf{a}, \mathbf{b}, \mathbf{c})$.
  – *Type:* 3×3 `array floats`
- `species`
  – *Description:* list of atomic species in crystal.
  – *Type:* `array of strings`
- `number_each_type`
  – *Description:* number of atoms for each distinct atomic species.
  – *Type:* `array of ints`
- `coordinates_type`
  – *Description:* indicates the coordinate representation ("cartesian" or "direct").
  – *Type:* `string`
- `atoms`
  – *Description:* atom information.
  – *Type:* `array of atom objects`

where the `atom` object contains,
- `name`
  – *Description:* atomic species name.
  – *Type:* `string`
- `occupancy`
  – *Description:* site occupancy.
  – *Type:* `float`
- `position`
  – *Description:* cartesian or fractional coordinate.
  – *Type:* 3×1 `array`

sgdata output. The output from this function is a subset of `edata` containing the space group and Wyckoff position information.

[1] M. J. Mehl, D. Hicks, C. Toher, O. Levy, R. M. Hanson, G. L. W. Hart, and S. Curtarolo, *The AFLOW Library of Crystallographic Prototypes: Part 1*, Comput. Mater. Sci. **136**, S1–S828 (2017).
[2] M. J. Buerger, *Derivative Crystal Structures*, J. Chem. Phys. **15**, 1–16 (1947).
[3] G. L. W. Hart and R. W. Forcade, *Algorithm for generating derivative structures*, Phys. Rev. B **77**, 224115 (2008).
[4] W. Setyawan and S. Curtarolo, *High-throughput electronic band structure calculations: Challenges and tools*, Comput. Mater. Sci. **49**, 299–312 (2010).
[5] C. Toher, J. J. Plata, O. Levy, M. de Jong, M. D. Asta, M. Buongiorno Nardelli, and S. Curtarolo, *High-throughput computational screening of thermal conductivity, Debye temperature, and Grüneisen parameter using*

a quasiharmonic Debye model, Phys. Rev. B **90**, 174107 (2014).

[6] C. Toher, C. Oses, J. J. Plata, D. Hicks, F. Rose, O. Levy, M. de Jong, M. D. Asta, M. Fornari, M. Buongiorno Nardelli, and S. Curtarolo, *Combining the AFLOW GIBBS and Elastic Libraries to efficiently and robustly screen thermomechanical properties of solids*, Phys. Rev. Mater. **1**, 015401 (2017).

[7] P. Nath, J. J. Plata, D. Usanmaz, R. Al Rahal Al Orabi, M. Fornari, M. Buongiorno Nardelli, C. Toher, and S. Curtarolo, *High-throughput prediction of finite-temperature properties using the quasi-harmonic approximation*, Comput. Mater. Sci. **125**, 82–91 (2016).

[8] P. Nath, J. J. Plata, D. Usanmaz, C. Toher, M. Fornari, M. Buongiorno Nardelli, and S. Curtarolo, *High throughput combinatorial method for fast and robust prediction of lattice thermal conductivity*, Scr. Mater. **129**, 88–93 (2017).

[9] J. J. Plata, P. Nath, D. Usanmaz, J. Carrete, C. Toher, M. de Jong, M. D. Asta, M. Fornari, M. Buongiorno Nardelli, and S. Curtarolo, *An efficient and accurate framework for calculating lattice thermal conductivity of solids: AFLOW-AAPL Automatic Anharmonic Phonon Library*, NPJ Comput. Mater. **3**, 45 (2017).

[10] S. Curtarolo, G. L. W. Hart, M. Buongiorno Nardelli, N. Mingo, S. Sanvito, and O. Levy, *The high-throughput highway to computational materials design*, Nat. Mater. **12**, 191–201 (2013).

[11] 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, and D. Morgan, *AFLOW: An automatic framework for high-throughput materials discovery*, Comput. Mater. Sci. **58**, 218–226 (2012).

[12] K. Yang, C. Oses, and S. Curtarolo, *Modeling Off-Stoichiometry Materials with a High-Throughput Ab-Initio Approach*, Chem. Mater. **28**, 6484–6492 (2016).

[13] J. Carrete, N. Mingo, S. Wang, and S. Curtarolo, *Nanograined Half-Heusler Semiconductors as Advanced Thermoelectrics: An Ab Initio High-Throughput Statistical Study*, Adv. Func. Mater. **24**, 7427–7432 (2014).

[14] O. Levy, M. Jahnátek, R. V. Chepulskii, G. L. W. Hart, and S. Curtarolo, *Ordered Structures in Rhenium Binary Alloys from First-Principles Calculations*, J. Am. Chem. Soc. **133**, 158–163 (2011).

[15] O. Levy, G. L. W. Hart, and S. Curtarolo, *Structure maps for hcp metals from first-principles calculations*, Phys. Rev. B **81**, 174106 (2010).

[16] O. Levy, R. V. Chepulskii, G. L. W. Hart, and S. Curtarolo, *The New face of Rhodium Alloys: Revealing Ordered Structures from First Principles*, J. Am. Chem. Soc. **132**, 833–837 (2010).

[17] O. Levy, G. L. W. Hart, and S. Curtarolo, *Uncovering Compounds by Synergy of Cluster Expansion and High-Throughput Methods*, J. Am. Chem. Soc. **132**, 4830–4833 (2010).

[18] G. L. W. Hart, S. Curtarolo, T. B. Massalski, and 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**, 041035 (2013).

[19] A. R. Supka, T. E. Lyons, L. S. I. Liyanage, P. D'Amico, R. Al Rahal Al Orabi, S. Mahatara, P. Gopal, C. Toher, D. Ceresoli, A. Calzolari, S. Curtarolo, M. Buongiorno Nardelli, and M. Fornari, *AFLOWπ: A minimalist approach to high-throughput ab initio calculations including the generation of tight-binding hamiltonians*, Comput. Mater. Sci. **136**, 76–84 (2017).

[20] M. Scheffler, C. Draxl, and Computer Center of the Max-Planck Society, Garching, *The NoMaD Repository*, http://nomad-repository.eu (2014).

[21] A. Jain, G. Hautier, C. J. Moore, S. P. Ong, C. C. Fischer, T. Mueller, K. A. Persson, and G. Ceder, *A high-throughput infrastructure for density functional theory calculations*, Comput. Mater. Sci. **50**, 2295–2310 (2011).

[22] J. E. Saal, S. Kirklin, M. Aykol, B. Meredig, and C. Wolverton, *Materials Design and Discovery with High-Throughput Density Functional Theory: The Open Quantum Materials Database (OQMD)*, JOM **65**, 1501–1509 (2013).

[23] Y. Le Page, *Computer derivation of the symmetry elements implied in a structure description*, J. Appl. Crystallogr. **20**, 264–269 (1987).

[24] H. T. Stokes and D. M. Hatch, *FINDSYM: Program for identifying the space group symmetry of a crystal*, J. Appl. Crystallogr. **38**, 237–238 (2005).

[25] H. T. Stokes, *Using symmetry in frozen phonon calculations*, Ferroelectrics **164**, 183–188 (1995).

[26] A. L. Spek, *Single-crystal structure validation with the program PLATON*, J. Appl. Crystallogr. **36**, 7–13 (2003).

[27] *Spglib*, https://atztogo.github.io/spglib/ (2017).

[28] W. H. Baur and E. Tillmanns, *How to avoid unnecessarily low symmetry in crystal structure determinations*, Acta Crystallogr. Sect. B **42**, 95–111 (1986).

[29] F. H. Herbstein and R. E. Marsh, *Changes in space and Laue groups of some published crystal structures*, Acta Crystallogr. Sect. B **38**, 1051–1055 (1982).

[30] R. E. Marsh and F. H. Herbstein, *Some additional changes in space groups of published crystal structures*, Acta Crystallogr. Sect. B **39**, 280–287 (1983).

[31] G. Bergerhoff, R. Hundt, R. Sievers, and I. D. Brown, *The inorganic crystal structure data base*, J. Chem. Inf. Comput. Sci. **23**, 66–69 (1983).

[32] A. Belsky, M. Hellenbrandt, V. L. Karen, and P. Luksch, *New developments in the Inorganic Crystal Structure Database (ICSD): accessibility in support of materials research and design*, Acta Crystallogr. Sect. B **58**, 364–369 (2002).

[33] C. R. Groom, I. J. Bruno, M. P. Lightfoot, and S. C. Ward, *The Cambridge Structural Database*, Acta Crystallogr. Sect. B **72**, 171–179 (2016).

[34] L. A. Agapito, S. Curtarolo, and M. Buongiorno Nardelli, *Reformulation of* $\mathrm{DFT}+U$ *as a Pseudohybrid Hubbard Density Functional for Accelerated Materials Discovery*, Phys. Rev. X **5**, 011006 (2015).

[35] M. Hloucha and U. K. Deiters, *Fast Coding of the Minimum Image Convention*, Mol. Simul. **20**, 239–244 (1998).

[36] T. Hahn, ed., *International Tables of Crystallography. Volume A: Space-group symmetry* (Kluwer Academic publishers, International Union of Crystallography, Chester, England, 2002).

[37] H. Wondratschek and U. Müller, eds., *International Tables of Crystallography. Volume A1: Symmetry relations between space groups* (Kluwer Academic publishers, International Union of Crystallography, Chester, England, 2004).

[38] M. I. Aroyo, J. M. Perez-Mato, C. Capillas, E. Kroumova, S. Ivantchev, G. Madariaga, A. Kirov, and H. Wondratschek, *Bilbao Crystallographic Server: I. Databases and crystallographic computing programs*, Zeitschrift fuer Kristallographie **221**, 15–27 (2006).

[39] M. I. Aroyo, A. Kirov, C. Capillas, J. M. Perez-Mato, and H. Wondratschek, *Bilbao Crystallographic Server II: Representations of crystallographic point groups and space groups*, Acta Crystallogr. Sect. A **62**, 115–128 (2006).

[40] R. H. Taylor, F. Rose, C. Toher, O. Levy, K. Yang, M. Buongiorno Nardelli, and S. Curtarolo, *A RESTful API for exchanging materials data in the AFLOWLIB.org consortium*, Comput. Mater. Sci. **93**, 178–192 (2014).

[41] F. Rose, C. Toher, E. Gossett, C. Oses, M. Buongiorno Nardelli, M. Fornari, and S. Curtarolo, *AFLUX: The LUX materials search API for the AFLOW data repositories*, Comput. Mater. Sci. **137**, 362–370 (2017).

[42] M. Jahnátek, O. Levy, G. L. W. Hart, L. J. Nelson, R. V. Chepulskii, J. Xue, and S. Curtarolo, *Ordered phases in ruthenium binary alloys from high-throughput first-principles calculations*, Phys. Rev. B **84**, 214110 (2011).

[43] D. E. Sands, *Vectors and Tensors in Crystallography* (Addison-Wesley, 1982).

[44] C. Giacovazzo, H. L. Monaco, G. Artioli, D. Viterbo, M. Milanesio, G. Ferraris, G. Gilli, P. Gilli, G. Zanotti, and M. Catti, *Fundamentals of Crystallography*, IUCr Texts on Crystallography (Oxford University Press, 2011), 3rd edn.

[45] *Phonopy - Crystal symmetry*, https://atztogo.github.io/phonopy/symmetry.html (2017).

[46] *FINDSYM*, http://stokes.byu.edu/iso/findsym.php (2017).

[47] K. Matano, M. Kriener, K. Segawa, Y. Ando, and G. Zheng, *Spin-rotation symmetry breaking in the superconducting state of $Cu_xBi_2Se_3$*, Nat. Phys. **12**, 852–854 (2016).

[48] E. Perim, D. Lee, Y. Liu, C. Toher, P. Gong, Y. Li, W. N. Simmons, O. Levy, J. J. Vlassak, J. Schroers, and S. Curtarolo, *Spectral descriptors for bulk metallic glasses based on the thermodynamics of competing crystalline phases*, Nat. Commun. **7**, 12315 (2016).

[49] C. F. F. Karney, *Quaternions in molecular modeling*, J. Mol. Graph. Model. **25**, 595–604 (2007).

[50] H. P. Fritzer, *Molecular symmetry with quaternions*, Spectrochim. Acta A: Mol. Biomol. Spectrosc. **57**, 1919–1930 (2001).

[51] G. Kresse and J. Hafner, *Ab initio molecular dynamics for liquid metals*, Phys. Rev. B **47**, 558–561 (1993).

[52] G. Kresse and J. Hafner, *Ab initio molecular-dynamics simulation of the liquid-metal-amorphous-semiconductor transition in germanium*, Phys. Rev. B **49**, 14251–14269 (1994).

[53] G. Kresse and J. Furthmüller, *Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set*, Comput. Mater. Sci. **6**, 15–50 (1996).

[54] G. Kresse and J. Furthmüller, *Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set*, Phys. Rev. B **54**, 11169–11186 (1996).

[55] 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, and R. M. Wentzcovitch, *QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials*, J. Phys.: Condens. Matter **21**, 395502 (2009).

[56] 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, and D. C. Allan, *First-principles computation of material properties: the ABINIT software project*, Comput. Mater. Sci. **25**, 478–492 (2002).

[57] V. Blum, R. Gehrke, F. Hanke, P. Havu, V. Havu, X. Ren, K. Reuter, and M. Scheffler, *Ab initio molecular simulations with numeric atom-centered orbitals*, Comp. Phys. Comm. **180**, 2175–2196 (2009).

[58] M. Tinkham, *Group Theory and Quantum Mechanics* (McGraw-Hill, Inc., New York, New York, 1964).

[59] M. Nespolo and B. Souvignier, *Point groups in crystallography*, Zeitschrift für Kristallographie International journal for structural, physical, and chemical aspects of crystalline materials **224**, 127–136 (2009).

[60] R. Gilmore, *Lie Groups, Physics, and Geometry* (Cambridge University Press, 2008).