Computational Materials Science 139 (2017) 140–152

Contents lists available at ScienceDirect

Computational Materials Science

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

Atomate: A high-level interface to generate, execute, and analyze computational materials science workflows

Kiran Mathew [a,b,*], Joseph H. Montoya [a], Alireza Faghaninia [a], Shyam Dwarakanath [a], Muratahan Aykol [a], Hanmei Tang [c], Iek-heng Chu [c], Tess Smidt [e,f,g], Brandon Bocklund [d], Matthew Horton [a], John Dagdelen [a], Brandon Wood [b], Zi-Kui Liu [d], Jeffrey Neaton [e,f,g], Shyue Ping Ong [c], Kristin Persson [a,b], Anubhav Jain [a,*]

[a] Energy Storage and Distributed Resources Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA
[b] Department of Materials Science, University of California Berkeley, Berkeley, CA 94720, USA
[c] Department of Nanoengineering, University of California San Diego, La Jolla, CA 92093, USA
[d] Department of Materials Science and Engineering, The Pennsylvania State University, University Park, PA 16801, USA
[e] Department of Physics, University of California, Berkeley, CA 94720, USA
[f] Molecular Foundry, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA
[g] Kavli Energy NanoSciences Institute at Berkeley, Berkeley, CA 94720, USA

ARTICLE INFO

Article history:
Received 12 June 2017
Received in revised form 22 July 2017
Accepted 24 July 2017

ABSTRACT

We introduce atomate, an open-source Python framework for computational materials science simulation, analysis, and design with an emphasis on automation and extensibility. Built on top of open source Python packages already in use by the materials community such as pymatgen, FireWorks, and custodian, atomate provides well-tested workflow templates to compute various materials properties such as electronic bandstructure, elastic properties, and piezoelectric, dielectric, and ferroelectric properties. Atomate also enables the computational characterization of materials by providing workflows that calculate X-ray absorption (XAS), Electron energy loss (EELS) and Raman spectra. One of the major features of atomate is that it provides both fully functional workflows as well as reusable components that enable one to compose complex materials science workflows that use a diverse set of computational tools. Additionally, atomate creates output databases that organize the results from individual calculations and contains a *builder* framework that creates summary reports for each computed material based on multiple simulations.

1. Introduction

Computational materials science methods are continually growing in predictive power due to advances in theory, computing, and software development. Today, there exists several examples of new functional materials such as batteries [1,2], thermoelectrics [3,4], and catalysts [5,6] that have been designed primarily through such methods [7] and the use of computations has in some cases proven to save significant R&D costs and time [8]. As computational methods become applicable to a greater span of problems, the audience that could potentially benefit from their use grows. However, computational softwares such as density functional theory calculation codes typically require careful and manual setup of many parameters. The interface for performing calculations is typically highly tuned for performing a few very detailed studies. However, emerging applications efforts towards high-throughput screening for functional materials and building libraries of materials properties may involve thousands or even millions of calculations, for which it would be impossible to manually generate input files or fix various error messages that occur during such calculations. In addition, learning to correctly conduct multiple different types of analyses is difficult: calculation procedures are typically not well documented or even standardized, and certain types of calculations involve multiple, labor-intensive steps prone to errors. These difficulties can lead to inefficient and in some cases incorrect usage of these tools, hampering user productivity and data integrity. Thus, there have previously been multiple efforts to build abstraction layers intended to facilitate the use of computational methods. Such efforts include commercial offerings such as Medea [9], Materials Studio [10], and GoVasp (now part of Medea) and academic codes such as AiiDA [11], MAST [12], qmpy/OQMD [13], ASE [14], AFLOW [15], the Harvard Clean Energy Project [16], iochem-bd [17], Quixote [18], and our own previous efforts (MPWorks [19] and an earlier Java/SQL-based framework

* Corresponding authors at: Energy Storage and Distributed Resources Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA (K. Mathew).
E-mail addresses: kmathew@lbl.gov (K. Mathew), ajain@lbl.gov (A. Jain).

http://dx.doi.org/10.1016/j.commatsci.2017.07.030
0927-0256/

[20]). The common goals of these frameworks are multi-fold. First, they enhance productivity by freeing researchers to focus their attention on scientific aspects of the problem rather than the minutia of calculation execution. Second, they create a set of easily replicable and testable community standards for simulations. Finally, they enable new applications such as high-throughput computing by automating many tasks that are typically performed manually.

In this paper, we introduce **atomate**, which facilitates automatic and semi-automatic calculations of materials properties. The goal of atomate is to collect knowledge about calculation procedure for various types of materials analyses into easily-usable workflows and workflow components that can be modified and recomposed as needed. Some of the workflows currently available in atomate include the calculation of band structures, bulk modulus and elastic tensors, Raman spectra, dielectric constants, ferroelectricity, and multiple types of spectra calculation (XAS, EELS). Atomate currently interfaces with the VASP software for density functional theory calculations [21], the FEFF [22] code for spectroscopic properties, and has preliminary support for molecular dynamics simulations with LAMMPS [23]. Support for additional codes such as QChem [24] is planned for the future.

Atomate is a redesign of our previous Java/SQL based high-throughput infrastructure [20] as well as our second-generation Python/Mongo *MPWorks* effort [19], which powered the Materials Project [25] database of over 1 million individual calculations. Atomate aims to improve the extensibility, usability, and composability of workflows over our previous efforts. One major distinguishing feature of atomate versus many similar efforts is that it is built on top of multiple powerful open-source tools including pymatgen (software to generate/manipulate structures, create input files and post process output files) [26], custodian (software to recover from calculation errors) [27], and FireWorks (a workflow library) [28]. It also makes use of external libraries such as phonopy [29] for specialized calculations. This design allows almost all the source code of atomate itself to be dedicated to high-level specifications of calculation procedure. In addition, atomate contains tools both for executing calculations as well as managing the results within a well-structured database so that one can not only perform calculations but efficiently analyze their outputs.

Atomate is well-tested, with currently over 50 unit and integration tests that are run as part of continuous integration suites (CircleCI and TravisCI). The package is Python 2 and 3 compatible and includes substantial documentation. The source code, distributed under a modified BSD license, can be obtained at https://www.github.com/hackingmaterials/atomate. We note that although atomate itself is open-source, some of the software packages that it interfaces with (e.g., VASP [30,21]) are not and require a commercial license to use. Atomate can be easily installed using pip or anaconda package managers. The version of atomate at the time of writing is v0.5.6; updates to the code are clearly documented in the changelog, https://hackingmaterials.github.io/atomate/changelog.html.

2. Overview and code design

In this section, we briefly discuss some of the programming design decisions for atomate. Some of these design features, such as integration with multiple queuing systems, are common to most computational frameworks presented earlier. Other features, such as a built-in database for querying calculation results, are present in some form in other libraries (e.g., AiiDA) but currently absent in others (e.g., MAST). Finally, some design choices, such as the use of a noSQL document store (MongoDB) and a builders framework to post-process results, are to our knowledge unique to our design.

2.1. Modular code design and use of existing packages

The goal of atomate code is to document, codify, and automate procedures and algorithms for performing various types of chemistry and materials science calculations. However, many aspects of running such calculations are relatively mundane, such as correctly writing queue scripts for various types of queue managers such as PBS or SLURM, correcting calculation errors by modifying input files, and parsing various types of calculation file formats to extract data. One major design decision of atomate is to focus only on specifying materials workflows at a high level, i.e., at the level of detail typical of the Methods section of a research manuscript, and to use existing libraries for materials manipulation, file I/O, error correction, and workflow management and execution.

In particular, we use:

- The pymatgen library [26] for structural manipulations, input file generation, and output file parsing.
- The FireWorks [28] library, which is a general workflow manager, as the language to define workflows and the mechanism to execute and manage workflows.
- The custodian [27] library for detecting errors during the calculation and automatically fixing them.

The above listed library dependencies are pictorially depicted in Fig. 1. Details of these three libraries can be found in their respective publications [26,28] as well as documentation [27]. All three of these libraries are open-source under MIT or modified BSD licenses.

Atomate benefits in two major ways from this type of modular design. First, it greatly reduces the length of code needed for the atomate library itself and keeps workflow specifications at a relatively high level. Such high-level specifications tend to be more general (e.g., not specific to a particular hardware) and easier to read, compose, reuse, and debug. Second, these three libraries continue to grow and develop new features independently of atomate's development. For example, pymatgen currently includes contributions from 71 developers and FireWorks from 28 developers. A non-materials scientist that contributes improvements to the visual workflow dashboard of FireWorks or improves its performance has in essence also contributed this functionality to atomate's user base. Similarly, new structural manipulation or analysis routines coded in pymatgen [26] by its user base become immediately available to atomate users. Thus, modularity serves as

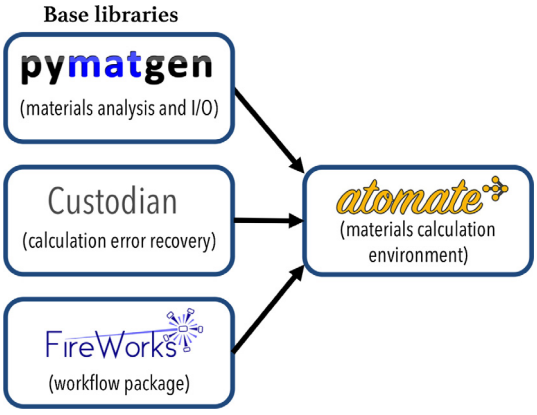


Fig. 1. Atomate dependencies.

force multiplier for the developer base for atomate and improves the rate at which users see improvements and new features.

2.2. Workflow model and reusable workflow components

A calculation workflow is composed of one or more calculations to perform a particular type of analysis of a material. For example, a band structure workflow may consist of a structure optimization, a charge optimization, and a non self-consistent run on a fine mesh k-point grid. Examples of calculation workflows implemented in atomate are provided in the next section; for example, Fig. 3 depicts the major workflow components for a structure relaxation and band structure calculation. Here, we summarize the principles behind the workflow model used in atomate, which are based on the classes and structure of its underlying FireWorks workflow library. More details about this model are presented in a previous paper on the architecture of the FireWorks software [28].

A workflow can be represented as a directed acyclical graph in which the nodes of the graph involve computational operations. In FireWorks, the definition of a workflow is extended to include control operations such as branching, looping, and *detours* (inserting a sub-workflow into workflow) based on conditionals that are evaluated during runtime. This is referred to in FireWorks as *dynamic workflows*, i.e., workflows that can programmatically modify themselves based on the results of completed steps of the workflow. Although each workflow in atomate is intended to be one *complete* type of analysis of a material, workflows can be chained or appended to one another within FireWorks.

An individual unit of work within a workflow is referred to as a *Firework* (note: a capitalization difference helps distinguish this object from the name of the FireWorks package). Thus, a workflow consists of one or more Fireworks with various dependencies; upstream Fireworks can pass data to downstream Fireworks through a FWAction object. Examples of Fireworks in atomate include performing a structure optimization calculation based on an input structure or performing a charge optimization calculation based on a structure. Splitting a workflow into Fireworks has both organizational and operational consequences. From an organizational standpoint, Fireworks serve as a more atomic unit in which to reuse workflow components. For example, the structure optimization Firework is reused in multiple different workflows, thus reducing the effort and required code needed to implement various workflows. From an operational standpoint, each Firework can have a shared space of parameters called the fw_spec that all operations within that Firework can use. When executing on a queue, all operations within a single Firework are necessarily performed as part of a single queue submission and will not be split across compute resources. However, from a workflow standpoint, the different Fireworks in that workflow may be run across different machines or across multiple queue submissions, depending on the chosen execution mode. Finally, some features such as setting priorities, overriding queue parameters, or setting custom variable overrides (fw_env) are specific to each Firework.

Fireworks themselves are composed of smaller functions called Firetasks. Within a Firework, Firetasks must run in sequence. Firetasks can share parameters that are defined within their enclosing Firework. For example, the most basic version of the structure optimization Firework is made up of 4 Firetasks: one to write input files given the structure and additional optional parameters, one to run the VASP executable either directly or through custodian error-correcting framework, one to parse the outputs and write a summary report to the file system or database, and a final one to pass results and the calculation location to the next Firework. Like Workflows and Fireworks, Firetasks are reusable components and a single implementation of a Firetask may be used in multiple types of Fireworks and therefore multiple Workflows.

Atomate also introduces a concept called *powerups* to help customize certain desired workflow behaviors. Powerups modify workflows in a similar way that decorators are used to modify functions in the Python language: namely, they take a workflow as input and return a modified workflow as output. For example, existing powerups can add prioritization rules to workflows, control whether a simulation software is run directly or through the custodian library, add metadata tags to the database for user tracking of workflows, or write files that help in organizing the contents of various calculation directories on a file system. Thus, the base workflow implementations are kept as minimal and as general as possible, aiming to contain only the necessary steps needed to properly execute a series of calculations. The user is then free to choose which modifications they desire by adding in the appropriate powerups without these custom modifications being written into the base workflow definition. A configuration file can be used to avoid having to re-specify what powerups and workflow settings the user chooses to use by default.

Because there are at least three levels at which self-sufficient code can be reused (Firetasks, Fireworks, and Workflows), and workflows can be modified by feeding them through one more *powerup* functions, creating new workflows is in some cases no more than complicated than assembling together previously-coded Firetasks and Fireworks into new configurations or with different parameters.

2.3. Workflow tracking, execution, and provenance

Workflow execution is managed by FireWorks. This includes interaction with various queue systems (currently, PBS, SLURM, Sun Grid Engine, and IBM Loadleveler are supported), job packing (multiple calculations executed one after one another within a queue submission as well as weak parallelization across multiple nodes), user prioritization of runs (at both the workflow and individual Firework level), and control of reruns. We note that execution details can differ across machines: as a simple example, the location of an executable or of the desired scratch directory might be different at different supercomputing centers. FireWorks includes a functionality called fw_env that allows user-defined variables to be tailored for a particular compute resource. In atomate, we extend this concept further, allowing variables to be either explicitly set or overridden by the compute resource through a concept called env_chk.

FireWorks includes a database that tracks the status of each run and can also track the progress of user-defined output files (e.g., to report back the last few lines of current OSZICAR file for all currently running VASP calculations). Users can quickly generate reports summarizing statistics of completion, failure, and job start as well as use a built-in web frontend to explore workflow status (Fig. 2). FireWorks can even inspect the job specification parameters of completed versus failed jobs and report what parameters appear to be correlated with higher than usual failure rates. For example, within atomate, this feature can help identify if runs containing a particular element are more likely to fail. The FireWorks database also serves as a source of provenance that, when taken together with the version of the all the various codebases used in the analysis, provides a comprehensive record of how the simulation was run.

2.4. Calculation results database

Atomate includes routines for parsing the output files generated by calculations and entering the results into a structured database. Atomate uses the noSQL database MongoDB, which organizes the information for a single calculation within a single document. We note that in our experience, MongoDB is easier for non

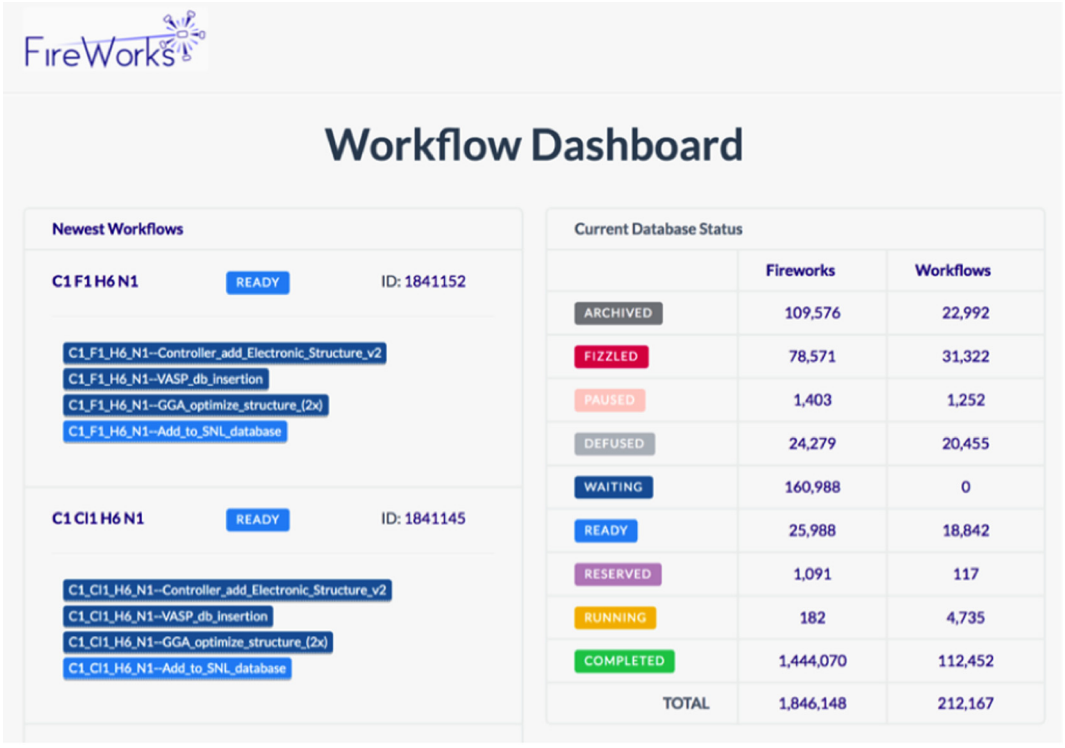


Fig. 2. Fireworks dashboard for the Materials Project production workflow manager. A live version is hosted by the Materials Project at http://fireworks.dash.materialsproject.org.

specialized users to use, understand, and develop on versus traditional SQL-style databases. This is because developing SQL databases requires designing a specialized schema that must be well-planned in advance, and querying SQL databases requires intimate knowledge of the schema format across various tables or requires the use of wrapper libraries to assist in querying. In contrast, noSQL databases are easier to design and more open to modification because they do not adhere to a rigid schema, and they are easier to query because queries are conducted on explicit documents. We have thus found it easier and more productive for a distributed team of non-specialist scientist-programmers to work with and develop noSQL-style databases.

In atomate, a collection of MongoDB documents called *tasks* contains the output data for all calculations. This document includes raw data (e.g., initial and final structures, input parameters, energy, magnetic moments), processed data (e.g., band structures, maximum residual force experienced by any atom at end of run), and metadata (time of completion, total calculation runtime, automatically-generated structure metadata).

We note that a single material might have several calculations or tasks associated with it (e.g., for various types of analyses). Similarly, some properties of a material may require combining data from multiple calculations on the same material or across different materials (for example, evaluating thermodynamic stability requires analyzing energies for all compounds in the same chemical system as a material). To combine information across multiple tasks, atomate includes a set of *builders* whose role is to aggregate and reduce data. For example, the *materials builder* combines and summarizes information from all tasks for a particular material (same composition and crystal structure). If a new type of calculation is later done on the material, the builder will add to that material's summary document. Similarly, the *E_hull builder* will run an analysis of the thermodynamic stability of a material based on the known energies of all materials in the same chemical systems. The builders play similar roles to *agents* described in the Computational Materials Repository [31]. The builders in atomate are implemented as *incremental* wherever possible - i.e., they minimize the amount of processing needed for execution by only processing new information since the last builder run.

A rudimentary web interface to the calculation results is available through the pymatgen-db [32] codebase. The web interface allows performing Mongo-like queries and exploring results in a text format. Future work may develop more graphical presentations of the results, e.g., crystal structure representations as is implemented in ase-db [33].

2.5. Usage modes

Common ways to interact with a software library include graphical, file-based, and programmatic interfaces. Atomate currently includes both file-based and programmatic interfaces, but does not include a rich graphical interface (which is typically only a feature of commercial software). It is possible that atomate will include a graphical interface in the future, although our current emphasis is on expanding the breadth and usability of the other two modes of interaction with atomate.

The file-based interface to atomate is through a shell script named *atwf* that is installed with atomate. With *atwf* one can use a combination of a structure file (e.g., in Crystallographic Information File or CIF format, as a Materials Project id interacting through the Materials API [34], or in one of several other supported formats such as POSCAR) and a workflow template file (in Yet Another Markup Language, or YAML format) to generate a workflow and submit it for calculation. Further details of the workflow template file are present in the atomate documentation [35]. However, we mention here that such files are generally only 10–20 lines of short text and thus are easy to read and modify. Because many of the workflows are simply different arrangements of Fireworks building blocks, using a file to define how these building blocks

interconnect to form various workflows makes workflow implementation simple and transparent.

A second usage mode available to users is programmatic access, which is the most powerful access mode. We have attempted to make this mode as easily accessible as possible to new programmers. For example, the VASP subpackage of atomate includes multiple *preset* workflow templates in which the user only needs to call a function with a structure as input in order to generate a full workflow. For small modifications to that workflow, one can use powerups which are again a simple function call. Further modifications can be obtained by changing the parameters at the Workflow, Firework, or Firetask level. Very fine-grained control is obtained by implementing or modifying workflow components through interactions with the pymatgen base library [26], which is also open-source. Thus, simple operations are made simple through the programmatic interface, and more advanced modifications are always possible. It is important to note that full customization of input files, and thus access to all the features of underlying software packages such as VASP, is always possible through the atomate interface.

2.6. Testing and validation

A major consideration when using automated or semi-automated workflow implementations is whether accuracy is retained compared to manual tailoring of input files for each specific calculation. In general, calculations involve two distinct types of accuracy. The first type is validation of the numerical approach - i.e., whether the calculation was performed and executed correctly (i.e., someone else would reproduce the same result using the same physical model). For example, the choice of plane wave cutoff energy or density of k-point grid in density functional theory calculations control numerical accuracy, but are not determining the underlying physics of the model. The second type of validation is the accuracy of the physical model itself. For example, the appropriate choice of functional that relates charge density to energy is a major physical consideration for density functional calculations.

The numerical parameters used in atomate are designed to achieve a balance of accuracy and low calculation time. Typically, this is done by performing convergence tests for a benchmark set of common systems and then defining cutoffs and settings based on those tests. Where possible, atomate uses parameters (e.g., pseudopotentials) that have already been tested by the Materials Project [36] and for which results are available for tens of thousands of compounds. A second method atomate employs is to include numerical validation checks as part of the workflow. For example, the workflow for elastic tensor includes explicit checks to ensure that the symmetry of the final tensor is commensurate with that of the lattice, and the workflow for ferroelectricity includes checks for zero band gap mid-workflow and checks on maximum atom displacement and polarization curve fit at the end of the workflow.

Our experience indicates that issues in the numerical approach are rarely the major issue because the error bar arising from these settings are usually small compared to deficiencies and errors in the physical model. Thus, one must take care to validate the accuracy of the physical model itself when using automated workflows. The default behavior of atomate is to again balance accuracy with calculation time. Many of the workflows in atomate have been previously and extensively benchmarked against experimental results [20,37–39]. However, the user must still decide what is the best course of action for their study. For example, atomate by default uses the GGA-PBE [40] functional for computing band structures, which is known to severely underestimate band gaps. Switching the functional to a more accurate Heyd-Ernzerhof-Scuseria (HSE) [41] hybrid functional is straightforward via a *powerup*, but requires the user to actively call the powerup function. Rather than always attempt to guess what the user should be doing for a particular system, atomate's goal is to make it simple for the user to choose between different frameworks.

3. Workflows

The majority of workflows currently included in atomate utilize the Vienna Ab-initio Software Package (VASP) [21,42–44], which enables plane wave density functional theory (DFT) calculations of periodic systems. This package is widely used in the computational research community to conduct first-principles calculations of materials properties such as electronic bandstructure, charge density, elastic tensors, and piezoelectric tensor. In addition to the properties that can be directly obtained, properties like energy, volume, forces and its derivatives can be processed further to compute other important properties such as phase diagrams, phonon bandstructure, thermal expansion coefficients, bulk modulus, Grüneisen parameters, thermal conductivity and Raman spectral intensities. In the following subsections we go through in detail selected VASP-based workflows available in atomate that can be used to compute direct and indirect properties of materials. In our final workflow subsection, we also outline a workflow used to simulate X-ray spectroscopies of materials using an effective scattering amplitude software (FEFF).

3.1. Crystal and electronic structure

A common workflow implemented in atomate is designed to determine optimized crystal structure and bandstructure from density functional theory calculations. This type of workflow may also be used to determine the structural stability of compounds using the potential energy as collected in VASP output and closely mimics the production workflow used in the Materials Project that has been used to collect structure and stability data on over 68,000 materials.

The standard workflow proceeds via an initial structure optimization. Following this, the optimized structure is passed to a Firework representing a static VASP calculation that determines the potential energy and density at a higher k-point mesh. From this, two non-self-consistent bandstructure calculations are performed in separate Fireworks, one in *line*-mode, in which a bandstructure is calculated along a high-symmetry k-path, and another in *uniform*-mode, in which the bandstructure is calculated with a uniform k-point grid. For each of the *static* Fireworks, structure and charge density determined from previous Fireworks is passed to the current Firework. In addition, in each step, a task document is created that stores VASP output information in a JSON-style document that may either be inserted into a Mongo database or output to disk as a JSON file. Fig. 3 depicts the workflow diagram along with a sample plot of the bandstructure and the density of states obtained from the workflow output.

The standard bandstructure and structure optimization workflow may be also be modified or enhanced based on user preferences. For example, atomate contains variations on the standard workflow that determine semi-classical transport coefficients via the BoltzTraP code [45], that add an additional step to calculate the bandstructure or bandgap with the more accurate HSE functional [41,46], or that include spin-orbit coupling in a final calculation. A simpler workflow that only performs the initial structure optimization but bypasses the fine mesh and special k-point electronic structure may also be chosen in the standard preset library.

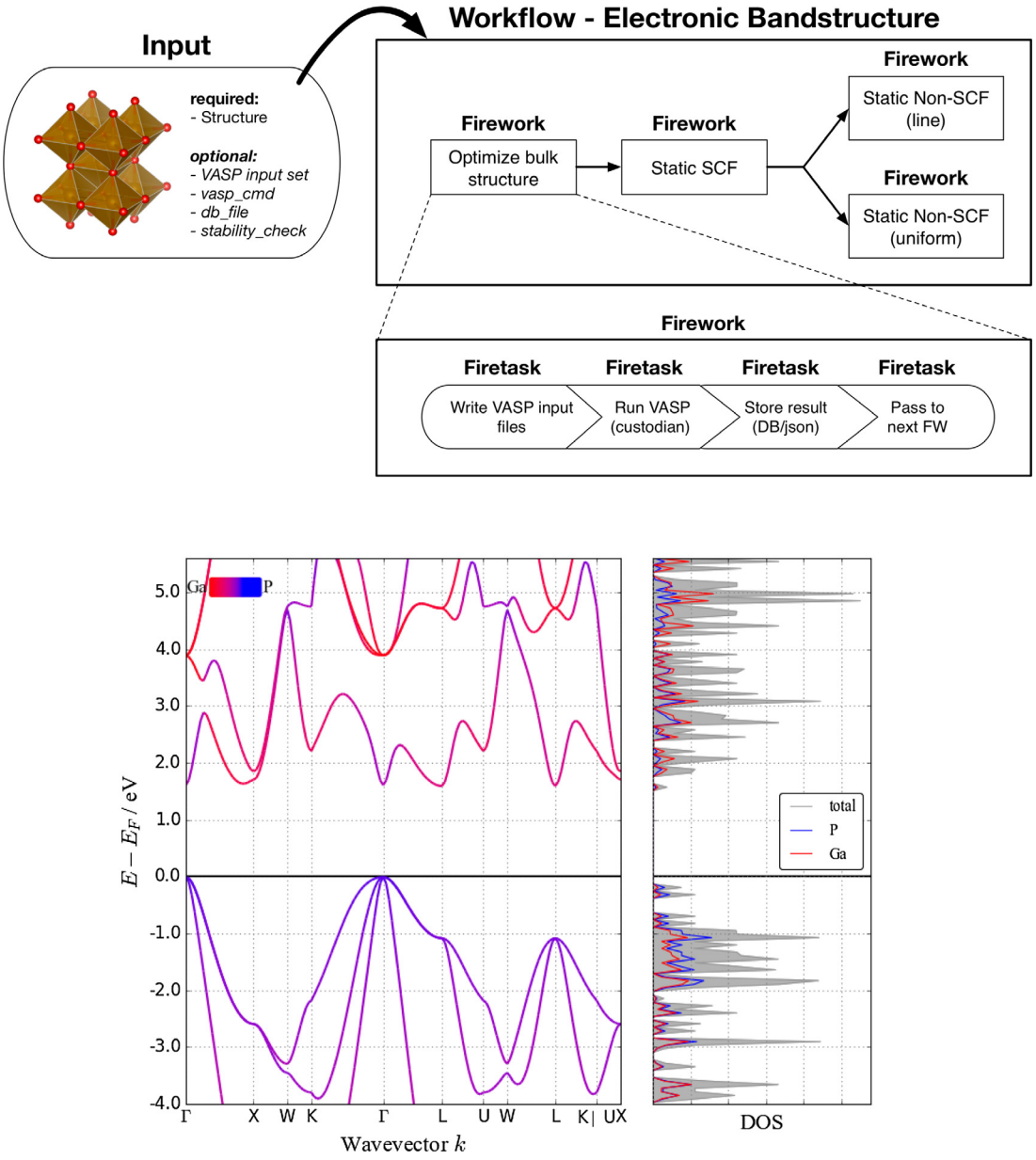


Fig. 3. Top: Workflow diagram for standard structure relaxation and bandstructure calculation. Structure is the only required input, but parameters may be customized using pymatgen VASP input set objects. In addition, users may supply custom VASP commands (e.g. for a custom compiled binary) and db credentials. Bottom: A plot (generated using pymatgen and matplotlib [47]) of a sample electronic bandstructure and density of states obtained from the atomate workflow.

3.2. Elastic tensor

To determine the elastic tensor of a given structure, we implement a workflow (shown in Fig. 4) that perturbs a structure by independent deformations corresponding to each of the 6 unique strain modes (i.e., 3 normal modes and 3 shear modes). This workflow is constructed via the use of several *Transmuter* Fireworks that transform structures according to a supplied pymatgen Transformation object, which in this case represents the application of a deformation gradient to the structure. The remaining Fireworks calculate the stress of the deformed structures and pass the stress and strain data to a final Firework, which analyzes the results. This final analysis Firework performs a linear least-squares fitting for each element of the elastic tensor and inserts the resultant elastic tensor and various derived properties (e.g., Voigt-Reuss-Hill averaged shear and bulk moduli) into a separate MongoDB collection.

The default operation of the elastic workflow in atomate is equivalent to that which has been used to generate data on the Materials Project website [25]. This preset uses a relatively high number of deformations, 4 for each mode, and adds parameters for the plane-wave cutoff and k-point mesh density appropriate for the high-throughput methodology outlined by de Jong et al. [37] However, users may also choose to customize the number and magnitude of the normal and shear deformations, and a preset workflow that uses the minimal number of perturbations for a general material is also available.

3.3. Bulk modulus, equation of state, and thermal properties

Atomate includes a standard procedure modeled after the Automatic Gibbs Library (AGL) [48] workflow for fitting energy-volume data to equations of state and the subsequent computation of

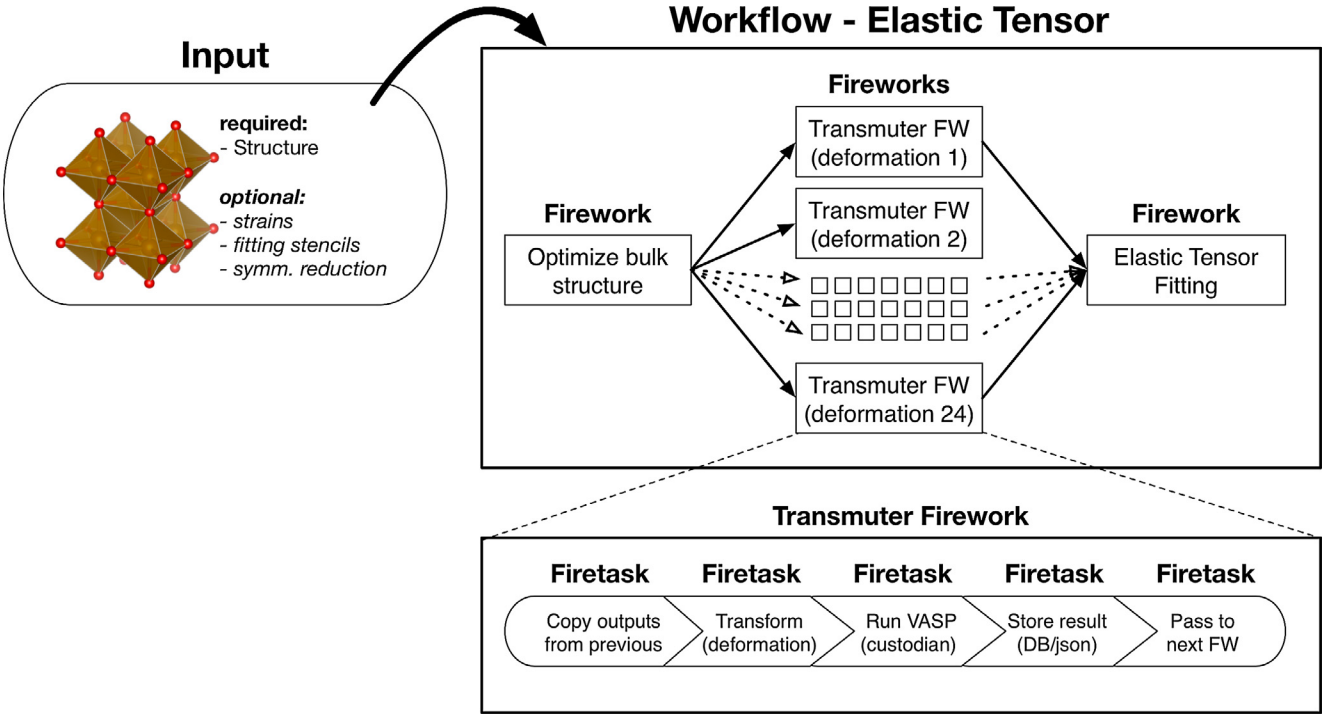


Fig. 4. Workflow diagram for the elastic tensor workflow. Structure is required as an input to the preset workflow, which creates Fireworks for each of 24 deformations corresponding to 4 perturbations to the 6 linearly independent strains in Voigt notation. These strain states and perturbation stencils may be further customized, and a *minimal* workflow which uses the fewest possible deformations by symmetry is also optional.

various thermal properties. This enables a user to efficiently determine the bulk modulus, estimate temperature-dependent free energies and various thermal properties such as thermal conductivity, Grüneisen parameter, etc. The structure of the standard workflow for equation of state closely mirrors that of the elastic workflow, in which deformations corresponding the isotropic volume expansion and compression are constructed and applied in Transmuter Fireworks. However, the specific deformations and the analysis procedure are different. The default operation of the equation of state workflow uses a set of 21 deformations ranging from ±10% of the structure's equilibrium lattice constant. For the equation of state and bulk modulus calculations the post-processing step uses the equation of state models available in pymatgen. For the other thermal properties the default post-processing step of the workflow is to feed the generated energy-volume data to the quasi-harmonic Debye approximation [49,50] as implemented in pymatgen to compute the thermal properties of interest. Also supported is the capability to run the workflow in the phonopy mode so that phonopy [29] can be used in the final post-processing step to compute the thermal properties using their phonon-based quasi-harmonic approximation [51]. We note that various thermal parameters derived from detailed equations of state (e.g. Grüneisen parameters, etc.) might require higher-order elastic constants, which are significantly more expensive and require more numerous calculations than either of the standard elastic or bulk modulus workflow.

3.4. Piezo and dielectric tensors

The piezoelectric tensor workflow uses VASP's internal density functional perturbation theory routines to determine the piezoelectric constants. The preset settings have been tested with respect to cutoff and k-point convergence for a variety of materials that are consistent with a previous high-throughput study, which contains more details on this calculation workflow and its parameter choices [38]. A visual depiction of the workflow is shown in Fig. 5.

3.5. Ferroelectricity

Ferroelectrics are materials with a spontaneous polarization switchable by an applied electric field. To determine the spontaneous polarization of a polar material, we implement a workflow that calculates the change in polarization from a nonpolar high-symmetry reference structure to the given polar structure. The polarization value extracted from this workflow is directly comparable to the change in polarization measured experimentally due to ferroelectric switching.

This workflow takes a polar ferroelectric candidate structure and a nonpolar reference structure transformed to the polar low-symmetry setting as input. These structures must have their atoms ordered so intermediate structures between the nonpolar and polar endpoints can be created using linear interpolations. Determining an appropriate nonpolar reference structure for a given polar structure and transforming the nonpolar structure to the polar setting is non-trivial and is currently not included in the workflow.

The workflow relaxes the nonpolar and polar structures, performs a static calculation of the band gap, and if both endpoints are insulating, calculates the dipole moment of the endpoints. The workflow also calculates the polarization of several linearly-interpolated structures between the endpoints, which must also be insulating. Note that the polarization for a metallic system is ill-defined because metals screen electric fields and VASP calculates the Berry phase of all occupied bands to determine the polarization of the system. As such, the workflow halts upon any static calculation that yields a metallic bandstructure (in either the initial or interpolated polar or nonpolar structures).

The outputs from the calculation of the polarization of all structures along the distortion are post-processed using the *Polarization*

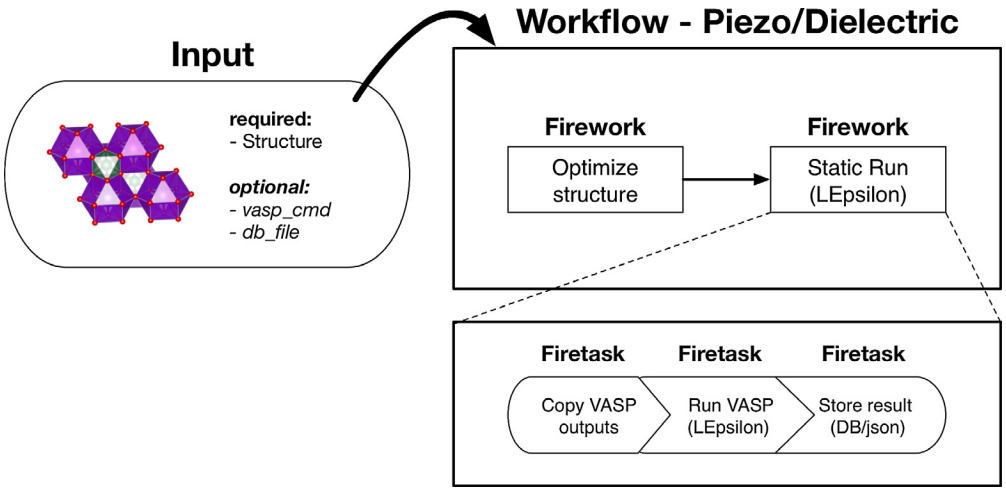


Fig. 5. Workflow diagram for Piezoelectric/Dielectric workflow: calculations begin with a structure optimization followed by a static run with the LEPSILON flag set to true. This static Firework uses VASP's internal routines for calculating the piezo and dielectric tensors and stores the result in the working directory of the Fireworks or in the task document inserted into a database.

and *EnergyTrend* classes in pymatgen to extract the spontaneous polarization and trends in total energy. Polarization is a lattice vector, meaning it is only defined modulo a quantum of polarization defined by the electron charge, lattice vectors and unit cell volume [52]. Because of this, we must adjust the polarization values from our calculations which may be on different *branches* (including different integer multiples of the quantum of polarization) to the same branch. Once the polarization values are on the same branch, we can simply subtract the polar and nonpolar polarizations to extract the spontaneous polarization. The *PolarizationToDbTask* performs this processing with the Polarization class and stores values to the database such as the raw electronic and ionic polarization, adjusted total polarization, change in polarization along the lattice vectors, and magnitude of polarization change. The *Polarization* and *EnergyTrend* classes are also used to store information relevant to the smoothness of the polarization and energy trends using splines. Fig. 6 depicts the overall ferroelectric workflow diagram.

Further details on the ferroelectric workflow, including obtaining valid nonpolar - polar structure pairs in an automated fashion, post-processing polarization calculations to extract the spontaneous polarization, and scientific results from this workflow will be the subject of future publications.

3.6. Kinetic barriers from NEB calculations

Climbing-image nudged elastic band (CI-NEB) approach has been widely employed to study the kinetics of materials such as the migration barriers of the mobile ions [53]. Compared to the traditional NEB approach [54], CI-NEB can accurately determine the transition state along the migration path (and hence the associated migration barrier) with less computational effort. A standard CI-NEB calculation consists of five major steps (see the NEB workflow diagram 7):

1. initial relaxation of the parent structure that does not contain any impurities/defects.
2. construction of the initial and final structures of the migration path (also known as end-point structures) from the parent structure.

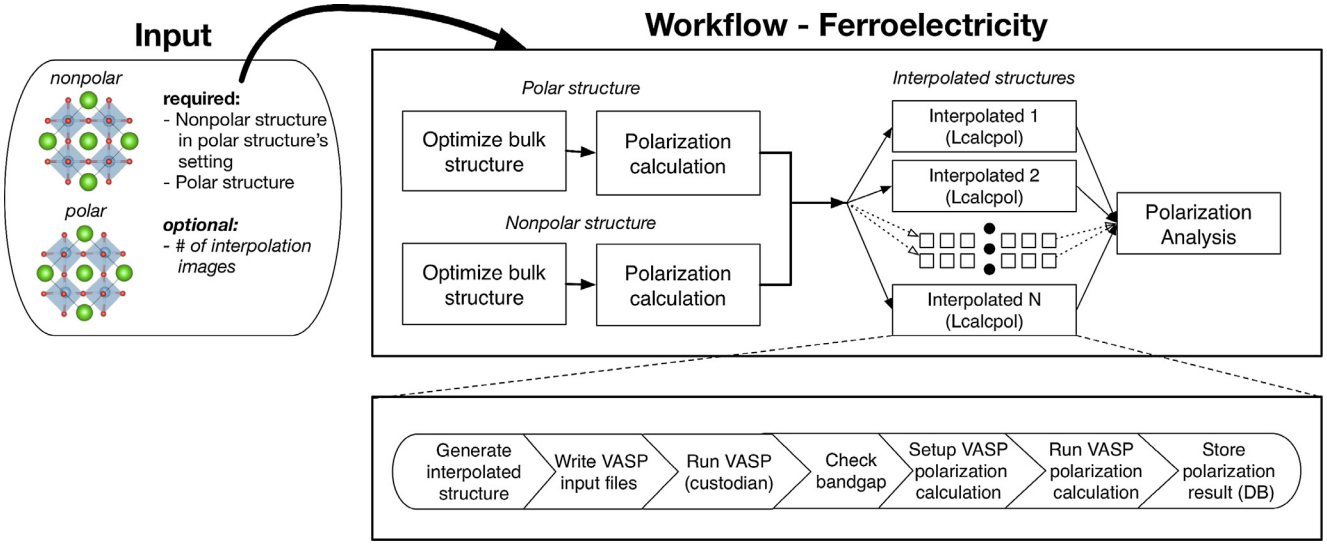


Fig. 6. Workflow diagram for a ferroelectric workflow: Inputs to this workflow must include a polar structure and a non-polar structure in the polar structure's setting. From these, polarization is calculated for both structures. Using similar calculations of the polarization of structure interpolated between the polar and non-polar setting, the spontaneous polarization of the material may be determined. This workflow halts if any given LcalcpolFW finds a non-insulating bandgap.

3. relaxation of the two end-point structures.
4. construction of the initial guess of the intermediate structures (also known as the image structures) along the migration path.
5. CI-NEB calculation that yields the minimum-energy migration path between the two end-point structures, in which the transition state is also identified.

In atomate, we implement a CI-NEB workflow that can be launched through three different methods (see Fig. 7). As a first method, the user can provide a parent structure along with a pair of atomic indices that define the migration path under the single vacancy diffusion mechanism. Alternatively, the user can provide two end-point structures under different diffusion mechanisms. Note that the construction of the image structures is required in both scenarios. This can be achieved by using either the traditional linear interpolation of the atomic coordinates between the end-point structures or the image dependent pair potential (IDPP) approach [55]. The former is implemented in pymatgen itself whereas the latter is implemented in pymatgen-diffusion, an add-on to the pymatgen package [26,56]. The IDPP approach has been shown to substantially improve the convergence speed of CI-NEB calculations and is thus set as the default approach for initial path construction. In the third scenario, the user provides a complete migration path comprising both end-point and image structures. To further accelerate the convergence speed of CI-NEB, the default workflow performs two rounds of CI-NEB calculations, wherein looser input parameters are used in the first round and tighter input parameters are used in the second round of calculations. All the preset settings in the CI-NEB workflow have been tested and tend to improve the overall efficiency in CI-NEB calculations, although the user can tune these parameters as needed for their study.

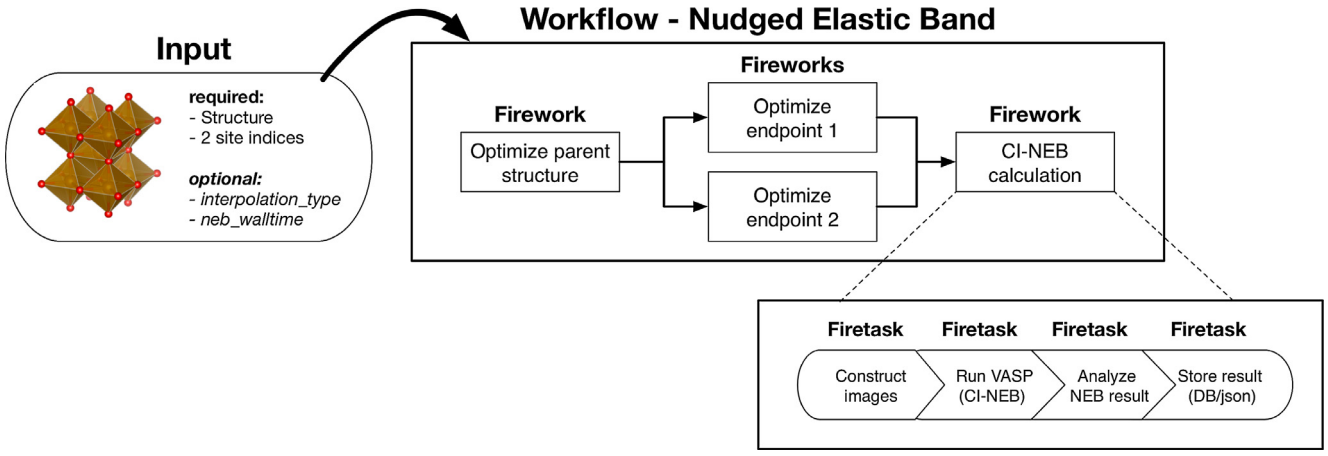


Fig. 7. Workflow diagram for Nudged Elastic Band (NEB) workflow: NEB workflows proceed via optimization of an initial parent structure and two end points. From these, an intermediate reaction path is estimated and a CI-NEB calculation conducted. The CI-NEB workflow also features automatic restart functionality since NEB calculations often exceed allowed walltimes on supercomputing resources. CI-NEBs may also be calculating using two images, rather than a single parent structure with specified sites.

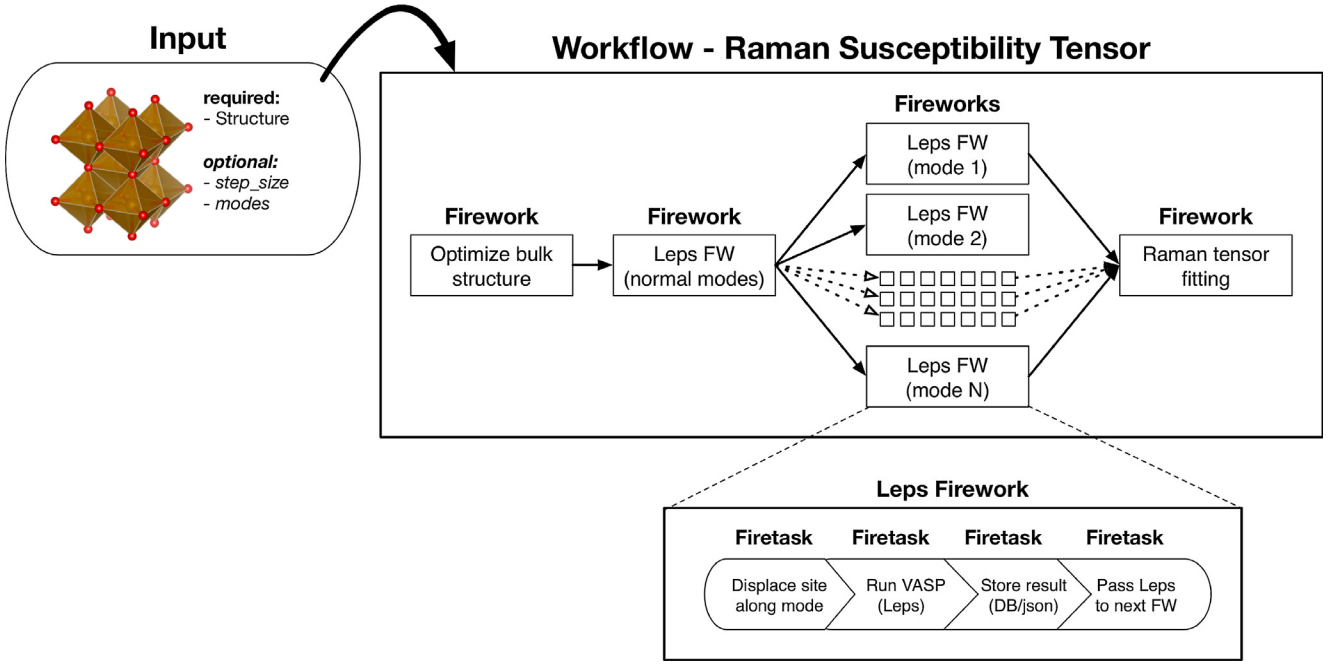
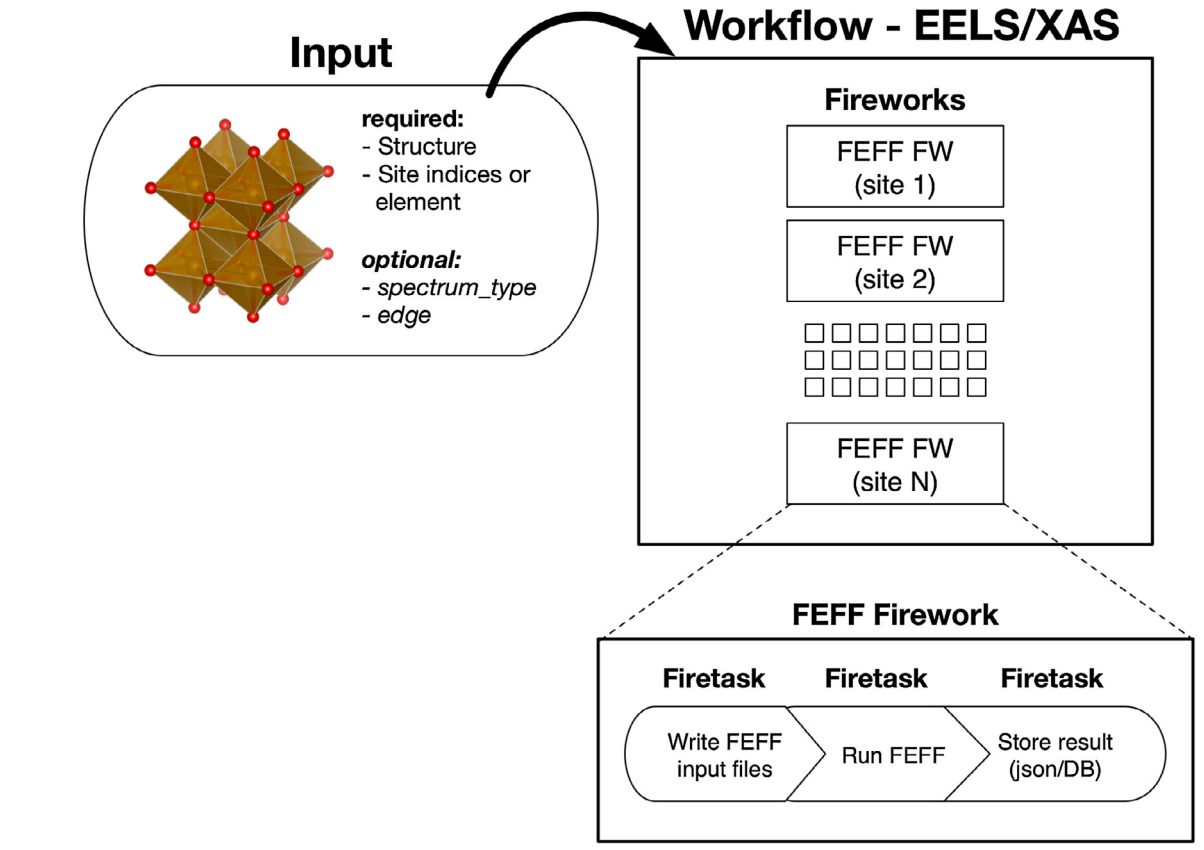


Fig. 8. Workflow diagram for Raman tensor: Raman tensors are calculated using a single LepsFW to calculate the normal modes for a given material after structural relaxation. The material is perturbed with respect each normal mode to calculate the finite difference derivatives corresponding to the Raman tensor.



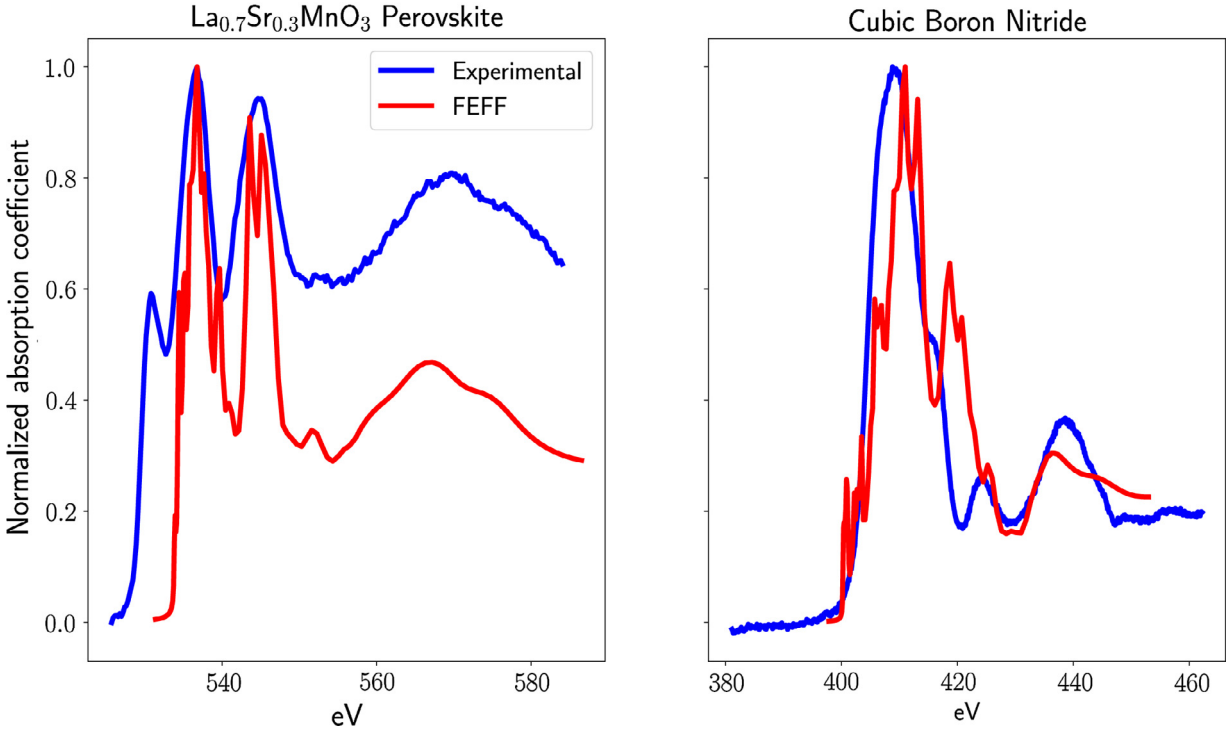


Fig. 9. Top: Workflow diagram for FEFF: site-specific X-ray spectra are generated from individual FEFF FWs in this workflow, which may simulate either EELS or XAS spectra. Bottom: Comparison of experimental K edge eels spectrum with the simulation obtained by the atomate workflow.

3.7. Ab-initio molecular dynamics

Molecular dynamics (MD) can simulate complex dynamic experiments (such as melt and quench routes for liquid and non-crystalline materials) and can provide structural, thermodynamic and kinetic properties [57–59,42]. MD workflows often require case-by-case structuring and specific post-processing and therefore in atomate, we only provide a generic ab initio MD (AIMD) Firework (MDFW) that can either be run by itself or chained together by the user into a multi-step procedure. The essential input parameters for the MDFW are the initial structure, start and end temperatures, and number of MD steps, accompanied by time step with a default of 2 femtoseconds. In the current default input settings, we use gamma-point only, the default kinetic energy cutoff for plane-waves (the maximum among the constituent elements), and *normal* precision for other VASP settings, but the user can customize these settings as needed. By utilizing the high-throughput capabilities of FireWorks, one can for example run multiple different conditions in parallel.

3.8. Raman spectroscopy

Solid state Raman spectroscopy has emerged as a powerful tool for characterization of materials [60], particularly for verifying structural characteristics of complex or two-dimensional polymorphs [61,62]. The Raman workflow (Fig. 8) begins with a structural optimization following by a static run with the LEPSILON setting enabled in VASP. From this, the normal modes are collected and the structure is perturbed along each mode for a positive and negative value of a user-supplied step size (default 0.005 Å). The Raman tensor for each mode is then calculated by determining the finite difference derivative of the dielectric tensor with respect to the mode perturbation and scaling the resulting tensor according to the structural volume. The results from this procedure are ultimately stored in a database or local JSON file.

3.9. XAS and EELS from FEFF

Various X-ray spectroscopies provide detailed information about local and bulk electronic structure, making them essential characterization tools in modern materials science. As such, ab initio software frameworks to simulate the results of such characterization techniques based on theoretical structures have also grown in popularity as they can help in interpreting these experimental results [63]. One such software is FEFF [22]. FEFF implements the real space Green's function (RSGF) approach for the calculation of X-ray absorption spectra (XAS). It provides a parameter free ab initio framework to compute the near edge (XANES) and extended (EXAFS) spectra. It also enables the calculation of other spectra such as X-ray Raman scattering (XRS) and electron energy loss (EELS). Atomate includes a framework for running and analyzing results from FEFF for the various spectroscopic simulations it supports. In its current form, preset workflows for the collection of XAS and EELS are included in the package.

For both sets of workflows, required inputs include the absorbing atom, which can either be chosen as a site index corresponding to the input pymatgen structure or an elemental symbol for the chemical identity of sites to include. Fireworks are generated for FEFF analysis of each absorbing site. For the XAS workflow, either of the XANES or EXAFS modes of FEFF may be chosen. Similarly, for the EELS workflow, either ELNES or EXELFS spectra may be generated. Named keywords are also included in the workflow constructors for commonly specified parameters, including but not limited to adsorption edge (K, L1, etc.), cluster radius, and incident beam energy (for EELS).

FEFF workflows behave similarly to VASP workflows in that they include a Firetask to write a set of files, run the FEFF binary, and store the generated spectrum into a MongoDb database or a local JSON file. Functionality for running EXAFS with customized scattering paths are also included. In addition to the database insertion of the computed spectrum, it is also possible to insert the FEFF generated density of states to the MongoDB GridFS with a simple modification to the default workflow, making it easy to search and retrieve density of states across multiple calculations. Fig. 9 depicts the FEFF spectra workflow diagram along with a sample plot of the EELS spectrum obtained from the workflow output.

4. Conclusion

In this work, we outline an infrastructure for performing atomistic simulations of materials. The purpose of this infrastructure is twofold. Firstly, atomate is intended to make simulations, particularly in high-throughput mode, simpler to execute and analyze. This is achieved by providing built-in workflows that are simple to set up yet highly customizable and by providing direct access to important calculation outputs in an easy-to-use database format. The ability to construct and execute an entire workflow from structural and parameter inputs significantly reduces individual researchers' time spent on manual file management, data collection, and code debugging. Secondly, atomate is intended to serve as a repository for standardized methods of materials simulation. A sensible structure, unit-testing framework, and community of maintainers ensure that results from a given workflow may be easily examined, reproduced, and extended for new applications, which reinforces the consistency and reliability of computational materials science results generated using atomate.

We also note here that atomate's potential can be even further enhanced with a larger user base. To this end, we intend to grow the community of individual researchers and research groups using atomate by encouraging contributions, collaboration, and modification where appropriate. It is our hope that atomate may serve as a repository of methods for researchers seeking to provide the research community with a transparent view of their methodology and simplified procedures for result validation.

Acknowledgments

The authors thank J. Rehr, A. Dozier, Chen Zheng and Chi Chen for their contributions towards the FEFF workflow and C. Toher for the discussions on AGL. This work was intellectually led by the U.S. Department of Energy, Office of Basic Energy Sciences, Early Career Research Program (ECRP), which directly supported A.J and A.F's contributions. K.M. and K.P. are supported by the Data Infrastructure Building Blocks (DIBBS) Local Spectroscopy Data Infrastructure (LSDI) project funded by National Science Foundation (NSF), under Award Number 1640899 and the Joint Center for Energy Storage Research (JCESR) project. J.H.M., J.B.N., T.E.S, and K.P. acknowledge support from the Materials Project Center through Grant No. EDCBEE through the U.S. Department of Energy, Office of Basic Energy Sciences, Materials Sciences and Engineering Division, under Contract No. DE-AC02 05CH11231. Work at the Molecular Foundry (J.B.N. and T.E.S.) was supported by the Office of Science, Office of Basic Energy Sciences, of the U.S. Department of Energy under Contract No. DE-AC0205CH11231. M.A. was supported as part of the Computational Materials Sciences Program funded by the U.S. Department of Energy, Office of Science, Basic Energy Sciences, Materials Sciences and Engineering Division. B.B. and Z.L. are supported by the NSF National Research Trainee Fellowship under grant DGE-1449785. S.D. acknowledges support from the Center for the Next Generation of Materials by Design,

an Energy Frontier Research Center funded by the U.S. Department of Energy, Office of Science, Basic Energy Sciences under Contract No. DE-AC36-08GO28308 to NREL. H. Tang, I.-H. Chu and S.P. Ong acknowledge support from the NSF, SI2-SSI Program under Award No. 1550423 for the development of the NEB workflow. Computational resources were provided by National Energy Research Supercomputing Center (NERSC), a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. Computational resources for NEB workflows were provided by Triton Shared Computing Cluster (TSCC) at the University of California, San Diego, NERSC, and the Extreme Science and Engineering Discovery Environment (XSEDE) supported by NSF under Grant No. ACI-1053575.

References

[1] G. Ceder, Y.-M. Chiang, D. Sadoway, M. Aydinol, Y.-I. Jang, B. Huang, Identification of cathode materials for lithium batteries guided by first-principles calculations, Nature 392 (6677) (1998) 694–696.
[2] K. Kang, Y.S. Meng, J. Bréger, C.P. Grey, G. Ceder, Electrodes with high power and high capacity for rechargeable lithium batteries, Science 311 (5763) (2006) 977–980.
[3] G.K. Madsen, Automated search for new thermoelectric materials: the case of LiZnSb, J. Am. Chem. Soc. 128 (37) (2006) 12140–12146.
[4] H. Zhu, G. Hautier, U. Aydemir, Z.M. Gibbs, G. Li, S. Bajaj, J.-H. Pöhls, D. Broberg, W. Chen, A. Jain, et al., Computational and experimental investigation of TmAgTe 2 and XYZ 2 compounds, a new group of thermoelectric materials identified by first-principles high-throughput screening, J. Mater. Chem. C 3 (40) (2015) 10554–10565.
[5] T.F. Jaramillo, K.P. Jørgensen, J. Bonde, J.H. Nielsen, S. Horch, I. Chorkendorff, Identification of active edge sites for electrochemical H_2 evolution from MoS_2 nanocatalysts, Science 317 (5834) (2007) 100–102.
[6] F. Studt, I. Sharafutdinov, F. Abild-Pedersen, C.F. Elkjær, J.S. Hummelshøj, S. Dahl, I. Chorkendorff, J.K. Nørskov, Discovery of a ni-ga catalyst for carbon dioxide reduction to methanol, Nat. Chem. 6 (4) (2014) 320–324.
[7] A. Jain, Y. Shin, K.A. Persson, Computational predictions of energy materials using density functional theory, Nat. Rev. Mater. 1 (2016) 15004.
[8] J.A. Christodoulou, Integrated computational materials engineering and materials genome initiative: accelerating materials innovation, Adv. Mater. Process. 171 (3) (2013) 28–31.
[9] MedeA. <http://www.materialsdesign.com/medea>.
[10] Materials Studio. <http://accelrys.com/products/collaborative-science/biovia-materials-studio/>.
[11] G. Pizzi, A. Cepellotti, R. Sabatini, N. Marzari, B. Kozinsky, AiiDA: automated interactive infrastructure and database for computational science, Comput. Mater. Sci. 111 (2016) 218–230.
[12] T. Mayeshiba, H. Wu, T. Angsten, A. Kaczmarowski, Z. Song, G. Jenness, W. Xie, D. Morgan, The materials simulation toolkit (MAST) for atomistic modeling of defects and diffusion, Comput. Mater. Sci. 126 (2017) 90–102.
[13] J.E. Saal, S. Kirklin, M. Aykol, B. Meredig, C. Wolverton, Materials design and discovery with high-throughput density functional theory: the open quantum materials database (OQMD), JOM 65 (11) (2013) 1501–1509.
[14] S.R. Bahn, K.W. Jacobsen, An object-oriented scripting interface to a legacy electronic structure code, Comput. Sci. Eng. 4 (3) (2002) 56–66.
[15] S. Curtarolo, W. Setyawan, G.L. Hart, M. Jahnatek, R.V. Chepulskii, R.H. Taylor, S. Wang, J. Xue, K. Yang, O. Levy, et al., AFLOW: an automatic framework for high-throughput materials discovery, Comput. Mater. Sci. 58 (2012) 218–226.
[16] J. Hachmann, R. Olivares-Amaya, S. Atahan-Evrenk, C. Amador-Bedolla, R.S. Sánchez-Carrera, A. Gold-Parker, L. Vogt, A.M. Brockway, A. Aspuru-Guzik, The Harvard clean energy project: large-scale computational screening and design of organic photovoltaics on the world community grid, J. Phys. Chem. Lett. 2 (17) (2011) 2241–2251.
[17] M. Alvarez-Moreno, C. De Graaf, N. Lopez, F. Maseras, J.M. Poblet, C. Bo, Managing the computational chemistry big data problem: the ioChem-BD platform, J. Chem. Inform. Model. 55 (1) (2014) 95–103.
[18] The Quixote Web Framework. <http://quixote.ca/>.
[19] MPWorks. <https://github.com/materialsproject/MPWorks>.
[20] A. Jain, G. Hautier, C.J. Moore, S.P. Ong, C.C. Fischer, T. Mueller, K.A. Persson, G. Ceder, A high-throughput infrastructure for density functional theory calculations, Comput. Mater. Sci. 50 (8) (2011) 2295–2310.
[21] G. Kresse, J. Furthmüller, Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set, Phys. Rev. B 54 (16) (1996) 11169.
[22] J.J. Rehr, J.J. Kas, F.D. Vila, M.P. Prange, K. Jorissen, Parameter-free calculations of x-ray spectra with FEFF9, PCCP 12 (21) (2010) 5503–5513.
[23] S. Plimpton, Fast parallel algorithms for short-range molecular dynamics, J. Comput. Phys. 117 (1) (1995) 1–19.
[24] Y. Shao, Z. Gan, E. Epifanovsky, A.T.B. Gilbert, M. Wormit, J. Kussmann, A.W. Lange, A. Behn, J. Deng, X. Feng, D. Ghosh, M. Goldey, P.R. Horn, L.D. Jacobson, I. Kaliman, R.Z. Khaliullin, T. Kús, A. Landau, J. Liu, E.I. Proynov, Y.M. Rhee, R.M. Richard, M.A. Rohrdanz, R.P. Steele, E.J. Sundstrom, H.L. Woodcock III, P.M. Zimmerman, D. Zuev, B. Albrecht, E. Alguire, B. Austin, G.J.O. Beran, Y.A. Bernard, E. Berquist, K. Brandhorst, K.B. Bravaya, S.T. Brown, D. Casanova, C.-M. Chang, Y. Chen, S.H. Chien, K.D. Closser, D.L. Crittenden, M. Diedenhofen, R.A. DiStasio Jr., H. Dop, A.D. Dutoi, R.G. Edgar, S. Fatehi, L. Fusti-Molnar, A. Ghysels, A. Golubeva-Zadorozhnaya, J. Gomes, M.W.D. Hanson-Heine, P.H.P. Harbach, A. W. Hauser, E.G. Hohenstein, Z.C. Holden, T.-C. Jagau, H. Ji, B. Kaduk, K. Khistyaev, J. Kim, J. Kim, R.A. King, P. Klunzinger, D. Kosenkov, T. Kowalczyk, C. M. Krauter, K.U. Lao, A. Laurent, K.V. Lawler, S.V. Levchenko, C.Y. Lin, F. Liu, E. Livshits, R.C. Lochan, A. Luenser, P. Manohar, S.F. Manzer, S.-P. Mao, N. Mardirossian, A.V. Marenich, S.A. Maurer, N.J. Mayhall, C.M. Oana, R. Olivares-Amaya, D.P. O'Neill, J.A. Parkhill, T.M. Perrine, R. Peverati, P.A. Pieniazek, A. Prociuk, D.R. Rehn, E. Rosta, N.J. Russ, N. Sergueev, S.M. Sharada, S. Sharmaa, D. W. Small, A. Sodt, T. Stein, D. Stück, Y.-C. Su, A.J.W. Thom, T. Tsuchimochi, L. Vogt, O. Vydrov, T. Wang, M.A. Watson, J. Wenzel, A. White, C.F. Williams, V. Vanovschi, S. Yeganeh, S.R. Yost, Z.-Q. You, I.Y. Zhang, X. Zhang, Y. Zhou, B.R. Brooks, G.K.L. Chan, D.M. Chipman, C.J. Cramer, W.A. Goddard III, M.S. Gordon, W.J. Hehre, A. Klamt, H.F. Schaefer III, M.W. Schmidt, C.D. Sherrill, D.G. Truhlar, A. Warshel, X. Xua, A. Aspuru-Guzik, R. Baer, A.T. Bell, N.A. Besley, J.-D. Chai, A. Dreuw, B.D. Dunietz, T.R. Furlani, S.R. Gwaltney, C.-P. Hsu, Y. Jung, J. Kong, D.S. Lambrecht, W. Liang, C. Ochsenfeld, V.A. Rassolov, L.V. Slipchenko, J.E. Subotnik, T. Van Voorhis, J.M. Herbert, A.I. Krylov, P.M.W. Gill, M. Head-Gordon, Advances in molecular quantum chemistry contained in the q-chem 4 program package, Mol. Phys. 113 (2015) 184–215.
[25] A. Jain, S. Ong, G. Hautier, W. Chen, W. Richards, S. Dacek, S. Cholia, D. Gunter, D. Skinner, G. Ceder, K. Persson, The materials project: a materials genome approach to accelerating materials innovation, APL Mater 1 (2013) 011.
[26] S.P. Ong, W.D. Richards, A. Jain, G. Hautier, M. Kocher, S. Cholia, D. Gunter, V.L. Chevrier, K.A. Persson, G. Ceder, Python materials genomics (pymatgen): a robust, open-source python library for materials analysis, Comput. Mater. Sci. 68 (2013) 314–319.
[27] Custodian. <https://github.com/materialsproject/custodian>.
[28] A. Jain, S.P. Ong, W. Chen, B. Medasani, X. Qu, M. Kocher, M. Brafman, G. Petretto, G.-M. Rignanese, G. Hautier, et al., Fireworks: a dynamic workflow system designed for high-throughput applications, Concurr. Comput.: Pract. Exper. 27 (17) (2015) 5037–5059.
[29] A. Togo, I. Tanaka, First principles phonon calculations in materials science, Scripta Mater. 108 (2015) 1–5.
[30] VASP. <https://www.vasp.at/>.
[31] D.D. Landis, J.S. Hummelshøj, S. Nestorov, J. Greeley, M. Dułak, T. Bligaard, J.K. Nørskov, K.W. Jacobsen, The computational materials repository, Comput. Sci. Eng. 14 (6) (2012) 51–57.
[32] pymatgen-db. <https://github.com/materialsproject/pymatgen-db>.
[33] A. Larsen, J. Mortensen, J. Blomqvist, I. Castelli, R. Christensen, M. Dulak, J. Friis, M. Groves, B. Hammer, C. Hargus, et al., The atomic simulation environmenta python library for working with atoms, J. Phys.: Condens. Matter. (2017).
[34] S.P. Ong, S. Cholia, A. Jain, M. Brafman, D. Gunter, G. Ceder, K.A. Persson, The materials application programming interface (API): a simple, flexible and efficient API for materials data based on representational state transfer (REST) principles, Comput. Mater. Sci. 97 (2015) 209–215.
[35] Atomate documentation. <https://hackingmaterials.github.io/atomate/>.
[36] Materials Project Calculations Guide. <https://materialsproject.org/docs/calculations>.
[37] M. De Jong, W. Chen, T. Angsten, A. Jain, R. Notestine, A. Gamst, M. Sluiter, C.K. Ande, S. Van Der Zwaag, J.J. Plata, et al., Charting the complete elastic properties of inorganic crystalline compounds, Sci. Data 2 (2015).
[38] M. De Jong, W. Chen, H. Geerlings, M. Asta, K.A. Persson, A database to enable discovery and design of piezoelectric materials, Sci. Data 2 (2015).
[39] J.H. Montoya, K.A. Persson, A high-throughput framework for determining adsorption energies on solid surfaces, NPJ Comput. Mater. 3 (2017) 1.
[40] J.P. Perdew, K. Burke, M. Ernzerhof, Generalized gradient approximation made simple, Phys. Rev. Lett. 77 (18) (1996) 3865.
[41] J. Heyd, G.E. Scuseria, M. Ernzerhof, Hybrid functionals based on a screened coulomb potential, J. Chem. Phys. 118 (18) (2003) 8207–8215.
[42] G. Kresse, J. Hafner, Norm-conserving and ultrasoft pseudopotentials for first-row and transition elements, J. Phys.: Condens. Matter 6 (40) (1994) 8245.
[43] G. Kresse, D. Joubert, From ultrasoft pseudopotentials to the projector augmented-wave method, Phys. Rev. B 59 (3) (1999) 1758.
[44] G. Kresse, J. Furthmüller, Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set, Comput. Mater. Sci. 6 (1) (1996) 15–50.
[45] G.K. Madsen, D.J. Singh, BoltzTraP. A code for calculating band-structure dependent quantities, Comput. Phys. Commun. 175 (1) (2006) 67–71.
[46] J. Heyd, G.E. Scuseria, M. Ernzerhof, Hybrid functionals based on a screened coulomb potential, J. Chem. Phys. 118 (18) (2003) 8207–8215.
[47] J.D. Hunter, Matplotlib: a 2d graphics environment, Comput. Sci. Eng. 9 (3) (2007) 90–95.
[48] C. Toher, J.J. Plata, O. Levy, M. de Jong, M. Asta, M.B. Nardelli, S. Curtarolo, High-throughput computational screening of thermal conductivity, Debye temperature, and Grüneisen parameter using a quasiharmonic Debye model, Phys. Rev. B 90 (17) (2014) 174107.
[49] S.-L. Shang, Y. Wang, D. Kim, Z.-K. Liu, First-principles thermodynamics from phonon and debye model: application to ni and Ni_3Al, Comput. Mater. Sci. 47 (4) (2010) 1040–1048.
[50] M. Blanco, E. Francisco, V. Luaa, Gibbs: isothermal-isobaric thermodynamics of solids from energy curves using a quasi-harmonic debye model, Comput. Phys. Commun. 158 (1) (2004) 57–72.

[51] A. Togo, L. Chaput, I. Tanaka, G. Hug, First-principles phonon calculations of thermal expansion in Ti_3 SiC_2, Ti_3 AlC_2, and Ti_3 GeC_2, Phys. Rev. B 81 (17) (2010) 174301.
[52] N.A. Spaldin, A beginner's guide to the modern theory of polarization, J. Solid State Chem. 195 (2012) 2–10.
[53] G. Henkelman, H. Jónsson, Improved tangent estimate in the nudged elastic band method for finding minimum energy paths and saddle points, J. Chem. Phys. 113 (22) (2000) 9978–9985.
[54] G. Mills, W. Jacobsen, et al., Classical and quantum dynamics in condensed phase simulations.
[55] S. Smidstrup, A. Pedersen, K. Stokbro, H. Jónsson, Improved initial guess for minimum energy path calculations, J. Chem. Phys. 140 (21) (2014) 214106.
[56] Pymatgen-diffusion. <https://github.com/materialsvirtuallab/pymatgen-diffusion>.
[57] B.B. Karki, First-principles molecular dynamics simulations of silicate melts: structural and dynamical properties, Rev. Mineral. Geochem. 71 (1) (2010) 355–389.
[58] G.C. Sosso, G. Miceli, S. Caravati, F. Giberti, J. Behler, M. Bernasconi, Fast crystallization of the phase change compound GeTe by large-scale molecular dynamics simulations, J. Phys. Chem. Lett. 4 (24) (2013) 4241–4246.
[59] A. Pasturel, E.S. Tasci, M.H. Sluiter, N. Jakse, Structural and dynamic evolution in liquid au-si eutectic alloy by ab initio molecular dynamics, Phys. Rev. B 81 (14) (2010) 140202.
[60] P. Hasnip, K. Refson, M. Probert, J. Yates, S. Clark, C. Pickard, Density functional theory in the solid state, Philos. Trans.: Math. Phys. Eng. Sci. 372 (2011), http://dx.doi.org/10.1098/rsta.2013.0270.
[61] Y. Feng, W. Zhou, Y. Wang, J. Zhou, E. Liu, Y. Fu, Z. Ni, X. Wu, H. Yuan, F. Miao, et al., Raman vibrational spectra of bulk to monolayer Re S_2 with lower symmetry, Phys. Rev. B 92 (5) (2015) 054110.
[62] M. Akhtar, M. Menon, M. Sunkara, G. Sumanasekera, A. Durygin, J.B. Jasinski, High-pressure synthesis of rhombohedral α-$AgGaO_2$ via direct solid state reaction, J. Alloy. Compd. 641 (2015) 87–92.
[63] N. Yabuuchi, M. Nakayama, M. Takeuchi, S. Komaba, Y. Hashimoto, T. Mukai, H. Shiiba, K. Sato, Y. Kobayashi, A. Nakao, M. Yonemura, K. Yamanaka, K. Mitsuhara, T. Ohta, for lithium-ion batteries, Nat. Publ. Group 7 (2016) 1–10.