JCTC Journal of Chemical Theory and Computation

pubs.acs.org/JCTC Article

TribChem: A Software for the First-Principles, High-Throughput Study of Solid Interfaces and Their Tribological Properties

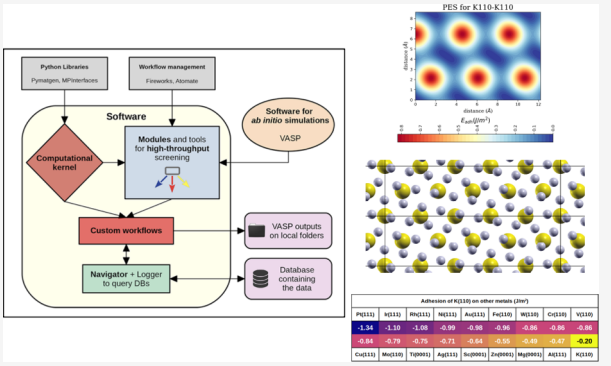
Gabriele Losi, Omar Chehaimi, and M. Clelia Righi*

Cite This: J. Chem. Theory Comput. 2023, 19, 5231−5241

Read Online

ACCESS | Metrics & More | Article Recommendations

ABSTRACT: High-throughput first-principles calculations, based on solving the quantum mechanical many-body problem for hundreds of materials in parallel, have been successfully applied to advance many materials-based technologies, from batteries to hydrogen storage. However, this approach has not yet been adopted to systematically study solid−solid interfaces and their tribological properties. To this aim, we developed TribChem, an advanced software program based on the FireWorks platform, which is here presented and released. TribChem is constructed in a modular way, allowing for the separate calculation of bulk, surface, and interface properties. At present, the calculated interfacial properties include adhesion, shear strength, and charge redistribution. Further properties can be easily added due to the general structure of the main workflow. TribChem contains a high-level interface class to store/retrieve results from its own database and connect to public databases.



1. INTRODUCTION

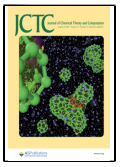
Integrating the experiments with computational tools and digital data is considered a key strategy to reduce the time and costs for materials discovery and deployment. In this context, first-principles high-throughput calculations, which allow for the density functional theory (DFT) description of many materials in parallel and in an automatized way, represent very powerful tools.[1−6] The calculated properties, usually highly accurate, are stored in databases and eventually analyzed with the aid of machine-learning algorithms, allowing for the identification of general trends and predictions. Moreover, raw data are also stored so that the calculation of further properties and rigorous validations are possible.

High-throughput calculations have been successfully applied to advance several materials-based technologies, including superconductivity,[7] catalysis,[4,8,9] and high-entropy alloying.[10] However, in this quickly developing framework, a systematic study of solid−solid interfaces and their tribological properties has not been addressed yet. Most probably, this is due to the inherent difficulties that this kind of system, composed of two different lattices matched together, poses and to the fact that the community of references (the tribology, metallurgy, and mechanics communities) traditionally relies on continuum macroscopic models.

Here, we present TribChem, a software program designed to perform the high-throughput study of solid interfaces. The software is composed of three main units for the study of bulks, surfaces, and interfaces. It is entirely written in Python and uses different packages from the Materials Project.[11] TribChem is based on FireWorks,[12] an open-source code for defining, managing, and executing workflows. Complex workflows can be defined using Python, JSON, or YAML and stored using MongoDB. For connecting to the Materials Project database, a high-level interface class named NavigatorMP has been developed. To perform the density functional theory (DFT) calculations, TribChem presently relies on the Vienna Ab initio Simulation Package (VASP).[13−16]

The first workflow developed by our group,[17] based on the Aiida platform,[18] was used for studying the interfacial properties of homogeneous interfaces.[19] TribChem contains several technical advancements, with respect to this initial workflow, related to the creation of proper error handling and the possibility to store and retrieve the data through a publicly accessible database, and has been extended to perform the study of heterointerfaces, such as those forming grain boundaries and heterojunctions. Simulating two different surfaces in contact is computationally much more demanding than considering equivalent surfaces. A common cell should be, in fact, identified to accommodate the two different lattices

Received: April 30, 2023
Published: July 4, 2023



with a reasonably small mismatch. This typically increases the size of the simulated system and its complexity. Indeed, we introduced a procedure based on the surface symmetries to calculate the potential energy surface (PES) that describes the interaction of the two surfaces in contact as a function of their relative lateral position.[17] We have successfully employed TribChem to systematically search for the optimal interface geometry and accurately determine adhesion energies of hundred metals relevant to technological applications. This allowed us to populate a database of accurate values, which can be used as input parameters for macroscopic models.[20]

The aim of the present paper is to provide a technical description of the structure, main features, and an example of the use of TribChem to accompany its release. The first three sections are devoted to the technical description of the software infrastructure, workflows, and database. In the last two sections, the instructions for the workflow execution and an example of use are presented.

2. IMPLEMENTATION AND ARCHITECTURE

The TribChem package consists of three main elements: the physics and high-throughput modules and the database (Figure 1).

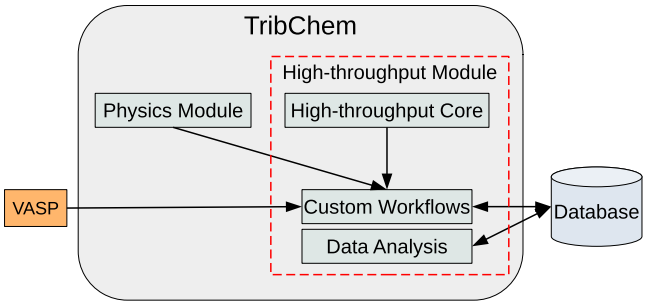


Figure 1. Schematic view of TribChem main components.

2.1. Physics Module. In the physics module, we implemented several functions to perform basic operations on solid-state physics, math, geometry, file manipulation, plotting, post-processing, and advanced operations on electronics, mechanical, and tribological properties. These functions are implemented in a highly modular way inside well-defined classes. Thanks to this design, it is possible to use these functions outside TribChem by importing them as external modules. The physics module relies on the Python Materials Genomics (Pymatgen)[21] package to manipulate bulk, slab, and interface structures and on the MPInterfaces[22] package, which implements the algorithm for geometrically matching two surfaces. This package is currently outdated and not maintained. Therefore, we are currently replacing it with similar functionalities present in Pymatgen.

Each submodule is briefly described below:

- `base`: It contains a collection of functions for manipulating atomic structures with periodic boundary conditions. It enables the replication of crystalline cells, the translation or rotation of atomic clusters, and the creation of an orthorhombic-shaped base area for any cell. It also includes mathematical functions for calculating gradients on three-dimensional surfaces and fitting data to polynomials of any degree. There is also a converter between the various physical units and CSV tables of the elemental materials classified based on their properties. Moreover, this module has advanced classes that allow working on the atomic structures of bulks, slabs, and interfaces.
- `chembas`: A class containing a set of functions useful for studying the adsorption of atoms on surfaces.
- `dft`: I/O handlers for VASP, Quantum Espresso,[23] and Lammps[24] are available, allowing the user to load input files into versatile Python dictionaries for further processing or extract information of interest from output files.
- `dynamics`: A set of tools for analyzing a dynamic simulation and calculating the positions, velocities, and forces acting on a single species or groups of atoms. It also includes many tools for plotting these properties.
- `electronics`: A set of functions and classes helpful to combine the electronic charge densities of output files and plot the electronic structure's bands and density of states.
- `ml`: An experimental submodule containing some functions for the generation of data sets suitable to train machine-learning models on them.
- `tribology`: It includes tools for computing and analyzing the tribological properties of interfaces, such as functions for matching two surfaces and calculating the potential energy surface (PES), the minimum energy path (MEP), and shear strength by combining the high-symmetry points of the surfaces, as described in previous papers of our group.[20,25]

2.2. High-Throughput Module. We implement the core classes necessary to create the building blocks of the custom workflows in the high-throughput module. Thanks to the modularity creating new custom workflows to perform new high-throughput calculations, also outside the field of tribology, is greatly simplified. Moreover, inside this module, we created some functions and custom classes to perform data analysis of the results. Finally, we added all of the functions necessary to communicate with the database, which is the central element of TribChem due to workflow management, and we stored in it all of the results.

One needs four different input files to run a VASP calculation: POTCAR, POSCAR, INCAR, and KPOINTS (for more information on the structure of these files, see the VASP manual). All of these files are generated automatically by TribChem according to the calculation one would like to execute. The creation of the POSCAR and the POTCAR files is relatively easy. The pymatgen library generates POSCAR by reading the pymatgen object from the database, while the same library also creates POTCAR by reading the pseudopotential files saved in a location specified during the installation of TribChem. Instead, for generating the INCAR and KPOINTS files, we created two classes, `VaspInputSet` and `MeshFromDensity`, that allow us to manipulate all possible setting options for file generation. Indeed, the INCAR and the KPOINTS files greatly differ from structure to structure. Hence, we developed custom classes that read from an external JSON file the parameters that should be passed as input.

3. WORKFLOWS

We briefly introduce the logic of the FireWorks library to understand how the high-throughput module works. Every workflow written in FireWorks has three fundamental

https://doi.org/10.1021/acs.jctc.3c00459
J. Chem. Theory Comput. 2023, 19, 5231−5241

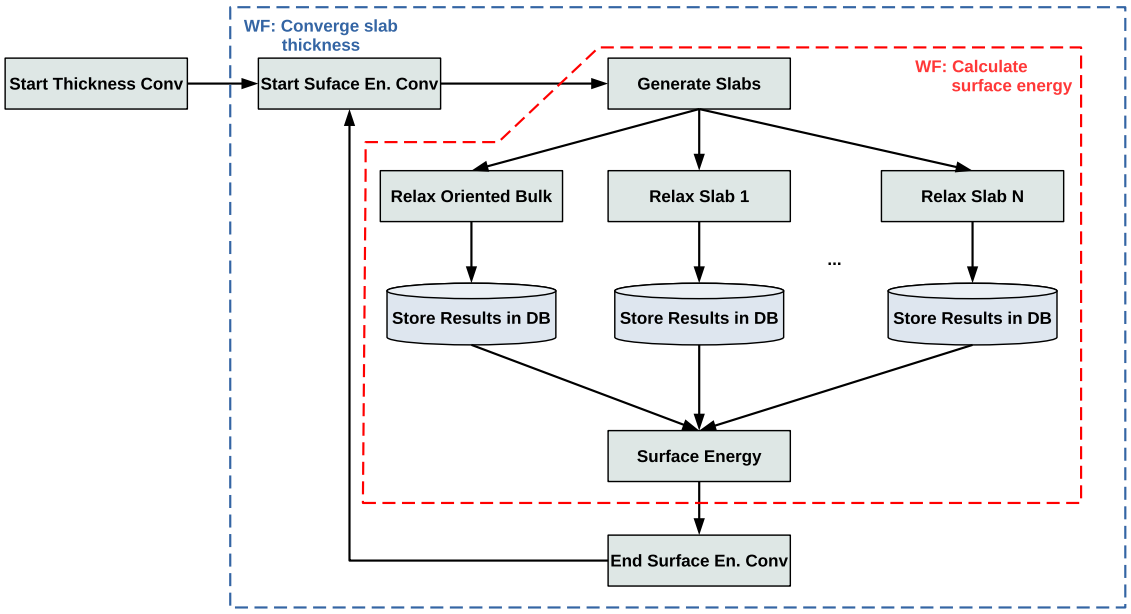


Figure 2. Flowchart of the two nested workflows used to generate crystalline slabs and converge their atomic thickness by evaluating the surface energy.

components: the FireTask, the FireWork, and the workflow. The FireTask is the most basic component, which usually implements an indivisible task (e.g., launch a VASP calculation or save data in the database). At the FireTask level, the input parameters are passed through the class variables `required_params` and `optional_params`. The FireWork has an intermediate level of complexity and can comprise one or more FireTasks, which are executed consecutively based on a given order (e.g., run a VASP calculation and then save the results in the database). Moreover, a FireWork can initialize and communicate with one or more new FireWorks (which in turn contain other FireTasks), generating a complex hierarchical structure where several operations are executed in a given order, sequentially or in parallel. Finally, the workflow at the top level of the hierarchy groups several FireWorks to execute a complex calculation (e.g., relax an entire structure or calculate the adhesion energy of an interface). The workflow is the only unit that can be launched on a workstation.

In practice, a FireTask is a class characterized by two class variables for dealing with the input parameters: `required_params`, `optional_params`, and method `run_task` to run the instructions of the FireTask. In addition, the tasks done by the FireTask are usually implemented inside the class as methods and then called inside `run_task` to be executed. Since all of the FireTasks share common operations, we developed a general FireTask class named `FireTaksTribChem`, in which we implemented all of the methods common to the different FireTasks. All of the custom FireTasks we developed inherit from the `FireTaksTribChem` class and thus have direct access to all its methods. In this way, the development of new FireTasks is greatly simplified and typically only necessitates the creation of specific methods. Of course, it is always possible to override the parent methods in the child classes, creating specific methods for the occurrence.

In short, these methods are:

- `read_runtask`: Read the optional and required parameters of a FireTask. The user passes these as input arguments when initializing the FireTask object.
- `read_params`: Update the dictionary parameters with default values read from a JSON file.
- `set_filter`: Create a filter to query the database to retrieve/store some data.
- `is_done`: Check if a specific entry within a MongoDB[26] document matches the filter. It is useful to check whether a calculation is already done and decide whether to run or stop a job.
- `query_db`: Retrieve data from a MongoDB database. The database location and tables are specified with the keywords `db_file`, database, collection, and entry.
- `update_db`: Update data in the matching field. Data are prepared to be stored in a database creating a dictionary according to the MongoDB guidelines. The database field to be updated is located by `db_file`, database, collection, and entry.
- `insert_db`: Add a new field to a database. The precise position is identified by `db_file`, database, and collection.
- `query_mp`: Query the Materials Project database, searching for a structure with a selected `mp_id` or searching for the lowest energy structure given a specific chemical formula.
- `save_files`: Create the folders to save the output files on the local machine.

3.1. Workflow Creation. Thanks to the FireWorks library structure, we can employ several strategies to create complex workflows by combining FireTasks and FireWorks. During the development of TribChem, we decided to implement the FireTasks first and then create FireWorks by employing just one FireTask to maintain the highest level of modularity. We followed this approach with some exceptions. In some situations, it is helpful to have several operations inside a FireWork. For example, every time we run a VASP simulation, all of the results must be saved in a proper location in the

https://doi.org/10.1021/acs.jctc.3c00459
J. Chem. Theory Comput. 2023, 19, 5231−5241

database. For this reason, we developed the function `run_and_save`, which returns a FireWork containing two FireTasks: one for running the simulation (`FT_RunVaspSimulation`) and another one for saving the results (`FT_MoveResults`).

This approach allowed us to exploit as much as possible the modularity given by the FireWorks library by creating atomic operations we used as building blocks to construct complex workflows.

The following example shows how a workflow is built. In this case, the workflow is composed of three serial FireWorks, each of them containing just one FireTask:

```
ft1 = CustomFiretask(...)

ft2 = CustomFiretask(...)

ft3 = CustomFiretask(...)

fw1 = Fireworks(ft1, name="FireWork 1")

fw2 = Fireworks(ft2, name="FireWork 2")

fw3 = Fireworks(ft3, name="FireWork 3")

wf = Workflow([fw1, fw2, fw3],

              links={fw1: fw2, fw2: fw3},

              name="Workflow")
```

Each FireWork communicates with the next one, and the execution order is specified inside the workflow definition as a dictionary in the variable called `links`. Hence, the order is `fw1`, `fw2`, and `fw3`.

3.1.1. Slab Generator Workflow. To better clarify the role of the modular approach we used, in the following, we describe, as an example, how the workflow generates a slab with a given orientation by the convergence of its surface energy (schematically shown in Figure 2).

3.2. Initializing FireWork. The workflow that calculates the optimal thickness is composed of two workflows: the main workflow, which starts the surface energy convergence (`Converge slab thickness`), and the internal workflow, which calculates the surface energy (`Calculate surface energy`). As shown in Figure 2, the internal workflow is called recursively until the convergence of the surface energy is reached. The `Start` Thickness Convergence! FireWork starts the whole workflow. During this step, the program first checks whether the requested slab has already been calculated. If it is not the case, the calculation is launched. In addition, an initial check is done to control which slab thicknesses have already been calculated. The simulation is launched only for the systems that have not been computed before. These checks allow us to save time and computational resources.

3.2.1. WF: Converge Slab Thickness. The `Converge` slab thickness! workflow consists of two FireWorks:

- **Start Surface Energy Convergence**: Calls the internal workflow, which calculates the surface energy.
- **End Surface Energy Convergence**: Checks if the convergence has been reached. We check the convergence by comparing the computed surface energies with the reference value defined as the energy of the maximum allowed thickness. When the relative difference in surface energy of one slab is lower than a certain threshold, the convergence is reached and the workflow finishes. If the convergence is not reached after the recursive calls, the convergence criterion is not satisfied, and the maximum thickness selected by the user is assumed to be optimized.

3.2.2. WF: Calculate the Surface Energy. This internal workflow calculates the surface energy and is composed of four different FireTasks:

- **Generate Slabs**: Generates a slab or a list of slabs starting from a bulk structure. The slab has the desired orientation, specified by the Miller indices, thickness, and vacuum to be included in the supercell for separating it from its replicas.
- **Relax Structure**: The geometry optimization is performed using VASP.
- **Store Results**: Saves the results of the first principle simulations in the database.
- **Surface Energy**: Calculates the surface energy and saves the result in the database.

3.3. Workflows Available in TribChem. The currently available workflows of the TribChem package are:

- Query online databases to save data to an internal database.
- Kinetic energy cutoff convergence.
- K-point convergence.
- Cohesion energy, calculated as

$$\epsilon_{\text{bulk}} = \frac{E_{\text{bulk}} - NE_{\text{a}}}{N} \tag{1}$$

  where E_{bulk} is the energy of the bulk system, N is the number of atoms in the simulation cell, and E_{a} is the energy of the isolated atom. Its unit is eV/atom.
- Calculation of the optimal thickness for a slab by fitting with eq 2 the slab total energies $E_{\text{slab}}(N)$, where N is the number of layers in the slab system.
- Surface energy E_γ, calculated as a fitted parameter in the following formula:

$$E_{\text{slab}} = 2{\cdot}A{\cdot}E_\gamma + N{\cdot}N_{\text{at}}{\cdot}E_{\text{bulk}} \tag{2}$$

  where N_{at} is the number of atoms per layer, E_{bulk} is the bulk energy per atom, and A is the slab in-plane area. When the convergence of the optimal slab thickness is reached, E_γ is returned as the surface energy of the system. Its unit is J/m^2.
- Matching and generation of a solid interface.
- Calculation of the PES.
- Evaluation of the adhesion energy. It is defined as

$$\gamma = \frac{E_{12} - E_1 - E_2}{A} \tag{3}$$

  where E_{12} is the energy of the interface system, E_1 and E_2 are the energies of the isolated, relaxed slabs, and A is the in-plane area of the simulation cell. Its unit is J/m^2.
- Extraction of the MEP and shear strength.
- Calculation of the perpendicular potential energy surface (PPES).
- Charge displacement analysis, defined as[27]

$$\rho_{\text{disp}} = \frac{1}{2z_0}\int_{-z_0}^{z_0} |\rho_{\text{diff}}\,\text{d}z| \tag{4}$$

  where z_0 is half of the interface distance and ρ_{diff} is the charge energy difference, and it is defined as $\frac{\rho_{\text{inter}} - \rho_{\text{top}} - \rho_{\text{bot}}}{A}$, where ρ_{inter} is the charge density of the

https://doi.org/10.1021/acs.jctc.3c00459
J. Chem. Theory Comput. 2023, 19, 5231−5241

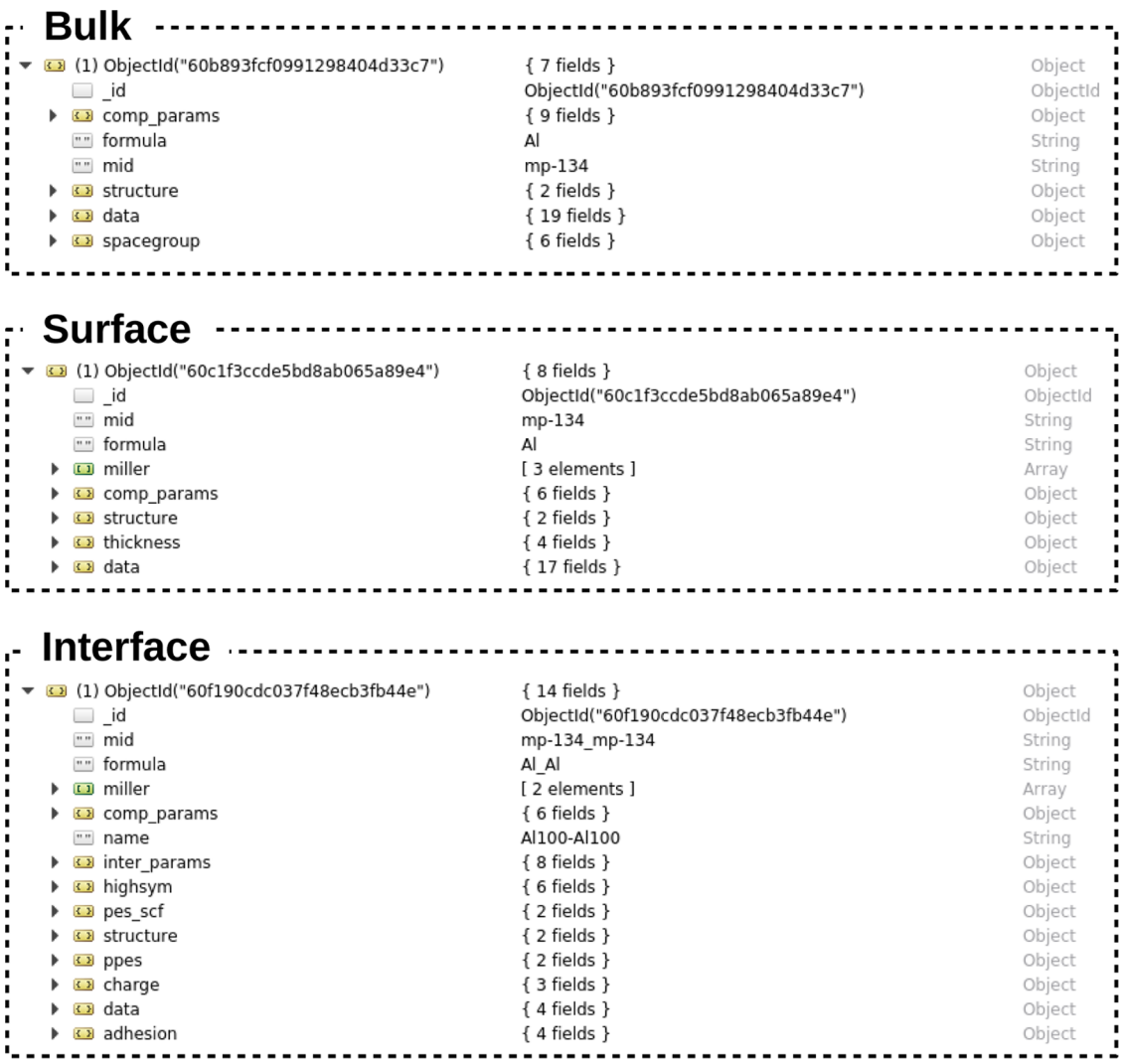


Figure 3. Typical design of the MongoDB documents in our database, showing how bulk, surface, and interface entries are saved. Aluminum is used as a case example.

interface system and ρ_{top} (ρ_{bot}) is the charge density of the isolated top (bottom) slab. Its unit is $e^-/Å^3$.

- Adsorption energies of atoms on surfaces, calculated as

$$E_{ads} = E_{surf+a} - E_{surf} - NE_a \quad (5)$$

where E_{surf+a} is the energy of the adsorbate system, E_{surf} is the energy of the isolated, relaxed surface, N is the number of adatoms in the supercell, and E_a is the energy of the adatom. The latter is calculated both by assuming the atom as isolated or interacting with its periodic replicas, i.e., by using the same supercell size adopted for simulating the adsorbate system. Its unit is eV/atom.

4. DATABASES

The database used by the FireWorks library is MongoDB,[26] a NoSQL database that uses JSON-like documents to store and manipulate data. When a new installation of TribChem is carried out, the user creates several databases. The most important and relevant ones are the `FireWorks` database, in which all of the data for the workflow execution are saved, as well as the simulation results and some relevant metadata such as the location of the VASP output files and the `TribChem` database. We designed this custom database to save, retrieve, and make it easier to share the results of the high-throughput calculations with the scientific community.

When performing a high-throughput study, it is critical to have powerful tools to efficiently perform the input/output (I/O) operations. Indeed, the amount of data generated is very high, and it is necessary to have the proper tools to manage them. To this end, we created several submodules containing Python tools for efficiently performing I/O operations. The class `Navigator` combines several low-level functions of the PyMongo[28] library. Starting from this class, we developed powerful functions to automatize the query operations inside all of the FireTasks of TribChem. Technically, we did this by implementing the functions as methods of the main FireTask `FiretaskTribchem`, which is used by all of the FireTasks of TribChem. This design greatly simplifies the structure of the software since it hides the implementations of all of the functions in just one class, making them more maintainable, less error-prone, and usable also by nontechnical users.

We also developed a high-level interface class named `NavigatorMP` for connecting to the Materials Project database.

4.1. TribChem Database. We carefully designed the structure of the `TribChem` database to ensure the best

https://doi.org/10.1021/acs.jctc.3c00459
J. Chem. Theory Comput. 2023, 19, 5231−5241

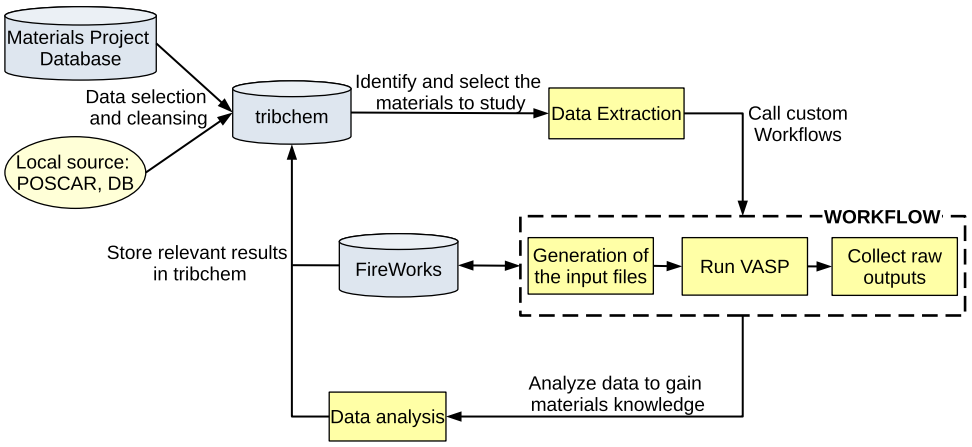


Figure 4. Flowchart showing the main logic behind the execution of a workflow within the TribChem package. Three sequential logical steps are of major importance: 1. recovery of materials data, 2. execution of a custom workflow and result collection, and 3. data analysis on the output data.

usability, maintainability, and robustness of the data stored inside it. This database includes several collections divided into different structures and functionals used for performing the DFT calculations. In our case, we have three main collections classes about bulks, slabs, and interfaces. The notation we used to name these collections is `functional.material`. For example, the collection of the elemental bulks calculated with the Perdew−Burke−Ernzerhof (PBE) functional is named `PBE.bulk_elements`. The structure of the three main collections is shown in Figure 3.

The main problem when working with data stored in a database is uniquely identifying each element. For this reason, we defined a set of identifiers for each of the three main collections we created.

In the case of the bulk, the main identifier is the material identifier (`mid`), which usually corresponds to the Materials Project ID for a given material, but it could be any alphanumeric value. In the case of aluminum, `mid` corresponds to mp-134 (Figure 3). We added two optional identifiers to the main one: `formula` and `name`. The former usually corresponds to the chemical formula of the material, and the latter is used as an additional identifier useful to speed up the search and avoid misinterpretations in some cases. For example, in case both diamond and graphene structures are saved in the bulk collection, setting the `name` to the structure name would avoid the conflict of having different elements with the same identifiers.

For identifying a slab, in addition to `mid` and `formula` identifiers, the crystalline orientation, defined by the Miller index, is used. The `miller` identifier is a Python list in the form `[h, k, l]` (e.g., `[1, 1, 1]`).

Finally, we defined the identifiers for an interface. Their name is the same for the bulk and slab collections, but their values are made up by combining the identifiers of the two slabs forming the interface. The resulting notation is: `mid=mid1_mid2`, `formula=f1_f2`, `miller=[[h1, k,1 l1], [h2, k2, l2]]`, and `name=f1m1-f2m2!`. In this case, the `miller` identifier is a list containing two lists (each for the orientation of the slab components), and the `name` is the formula combination of the two materials with the string version of the Miller indices of the relative slab. For example, a homogeneous interface of aluminum is identified by (Figure 3):

```
mid = "mp-134_mp-134"

formula = "Al_Al"

miller = [[1, 0, 0], [1, 0, 0]]

name = "Al100-Al100"
```

In addition to the consistent definition of identifiers, we also defined a coherent structure for saving the inputs, outputs, and computational parameters for running the VASP simulations. In every document, the fields `structure`, `data`, and `comp_params` can be found.

The field `structure` includes two other fields: `init`, which contains the primitive structure for the bulk downloaded from the Materials Project database, and for the slab and interface, created by our workflow, and `opt`, which contains the optimal structure obtained after the relaxation. Both these structures are saved as pymatgen objects.

The field `data` contains the results of the VASP calculations, such as the total energy, forces and stresses, band gap, and many others. All of the files produced by VASP are stored in the `calculations` folder, which is generated automatically in the same location as the TribChem installation folder that keeps the same hierarchical structure of the collections.

The field `comp_params` contains the computational parameters used by VASP. In this field, one can find the K-point density, the cutoff energy, whether the material is a metal or not, and many others. If this field does not sufficiently define all of the computational parameters, the user can introduce new ones. For example, we need information about the matching during interface formation. For this reason, we created the field `inter_params`.

Finally, other fields can be created to save the results of specific calculations. In the case of the interface, where there are many kinds of calculations and additional properties can be calculated, there are additional fields, such as `highsym`, `pes`, `adhesion`, `ppes`, and `charge`. These fields can store data concerning the high-symmetry points, the energies of the PES, the adhesion energy, the perpendicular potential energy surface (PPES), and the charge displacement calculation, respectively (Figure 3).

https://doi.org/10.1021/acs.jctc.3c00459
J. Chem. Theory Comput. 2023, 19, 5231−5241

5. EXECUTION OF A WORKFLOW

This section will explain the logic behind executing our high-throughput workflows. As shown in Figure 4, different components are involved during the execution of a workflow, with a continuous flow of data and instructions.

The first step in running a workflow consists of the selection of the desired material with its crystallographic information. The primitive structures can be stored in the `TribChem` database by downloading them from the Materials Project website, a local database, or directly from input files. Currently, only the Materials Project database can be used as an online resource. However, it is possible to add in the future the application programming interface (API) directives for the new database inside the TribChem project. Indeed, these new functions would write the new data into the unchanged `TribChem` database; hence, the same structure and functions remain the same.

Once the crystalline structures are loaded into `TribChem`, the workflows of interests can be executed. The execution of a workflow occurs in two steps: the first is to write the workflow in the database and the second is to run the calculations (for more information on how this can be done in practice, see the user manual of TribChem[29]). Based on the high-throughput study the user would like to perform, several workflows can be generated and written in the database and then run. Based on the experience we gained in using TribChem, we suggest running the same type of workflow to avoid possible race conditions. For example, the workflow that generates the slab is much faster than the workflow that calculates the PES and the adhesion energy. Although it is possible to run these two workflows at the same time for many different materials and interfaces, what would happen is that the faster workflows would be blocked by the slower ones, slowing down the whole process of getting new results.

Our workflows are designed to be also used outside a high-throughput procedure. Indeed, they can take as input an atomic structure directly from a POSCAR file by loading it as a pymatgen object. In this way, it is also possible to use TribChem in simple scripts.

Once the workflows are completed, TribChem provides several data analysis tools.

For more information on the practical usage of TribChem, see the user manual.[29]

5.1. Quality Control. We carried out the quality control of TribChem by using a set of dedicated functions present in the test folders of each module and submodule. The users can find more information on how to check if the installation process was successful and if the main functionalities are correctly working in the user manual.[29]

5.2. Availability. We host the TribChem source code at the following GitLab repository https://gitlab.com/triboteam/tribchem, where it is possible to find the complete installation procedure and the user manual.[29]

5.2.1. Operating System. The Linux operating system is required for the execution of TribChem. It is possible to use TribChem on the most relevant operating systems (Windows, macOS, and Linux) to create the workflows and analyze the data. However, we suggest using Linux.

5.2.2. Programming Language. Python 3.8.8.

5.2.3. Additional System Requirements. There are no particular requirements for using TribChem. The DFT calculations can usually be run on a high-performance computing system (HPC) to maximize computational efficiency, while the workflow creation and data analysis can also be done on an ordinary desktop system.

5.2.4. Dependencies. TribChem uses the following dependencies. Besides VASP,[13−16] which is proprietary software and must be purchased from the official channels (see https://www.vasp.at/), all of the other packages can be installed with conda or pip (see the installation guide of TribChem).

- atomate≥0.9.9:[30] A Python library containing a collection of FireWorks workflows, which make it easy to perform complex materials science computations.
- ASE≥3.21.1:[31] The Atomic Simulation Environment (ASE) is a Python library for manipulating, running, visualizing, and analyzing atomistic simulations.
- FireWorks≥1.9.7:[12] A Python library used for defining, managing, and executing workflows in computational materials science.
- Matplotlib≥3.3.4:[32] A graphical library for creating data visualization in Python.
- MongoDB≥4.0.3:[32] The Python driver for the MongoDB database.
- MPinterfaces≥2020.6.19:[22] A Python library that enables high-throughput DFT analysis of arbitrary material interfaces.
- NumPy≥1.20.2:[33] A high-level scientific computing library.
- pymatgen≥2022.0.8:[21] Python Materials Genomics is a Python library for materials analysis.
- SciPy≥1.6.2:[34] A scientific computing library containing advanced algorithms on optimization, fitting, and other classes of numerical problems.
- VASP≥6.2.0:[13−15] The Vienna Ab initio Simulation Package (VASP) is a computational package that permits us to perform DFT calculations.

6. EXAMPLE OF TRIBCHEM USE

In the following section, we show how Tribchem can be used to compute in an automatic and systematic fashion the adhesion energies of the K(110) surface interfaced with a set of 17 technologically relevant metal surfaces. All of the main workflow steps will be described starting from the bulk creation, slab thickness optimization, and surface energy calculations and ending with the heterointerface creation and computation of the adhesion energy. This example can be considered a further guide for using TribChem to study heterointerfaces and tribological properties.

6.1. Bulk Calculations. The first step in the study of any interface is bulk optimization: TribChem identifies the bulk structure equilibrium geometry and optimizes the computational parameters for the selected elements. In order to do so, the program selects the optimal kinetic energy cutoff by converging, with respect to this computational parameter, the bulk modulus and the cell volume. To perform this task and launch the calculations, the user has to execute the following command within a Python virtual environment (here applied to the case of the K bulk):

```
tribchem workflow converge_bulk

mids="[mp-58]" formulas="[K]"
```

More in detail: A Birch−Murnaghan equation of state[35] is used to fit the ab initio data for increasing cutoffs. The bulk

https://doi.org/10.1021/acs.jctc.3c00459
J. Chem. Theory Comput. 2023, 19, 5231−5241

modulus and the cell volume are extrapolated when the relative difference between two consecutive steps is below 1% for both quantities and convergence is reached. As an example, data obtained for K are reported in Table 1.

Table 1. Data of the Bulk Convergence for K: the Energy Cutoff in eV, the Bulk Modulus in GPa, the Cell Volume in Å^3, and the Lattice Parameter in Å are Reported; the Bold Values are the Ones Identified as Converged

energy cutoff	bulk modulus	cell volume	lattice parameter
200	3.218	74.477	5.301
225	3.551	73.858	5.286
250	3.491	73.523	5.278
275	3.482	73.630	5.281
300	3.488	73.668	5.282
325	3.490	73.677	5.282

In bold, the converged value of the bulk modulus and lattice parameter and the optimal kinetic energy cutoff are highlighted. The computed values are remarkably close to the experimental values[36] (3.0 GPa for the bulk modulus and 5.328 Å for the lattice parameter), confirming the reliability of DFT and of our workflow in the determination of the bulk properties.

With the same `TribChem` command, it is also possible to identify the optimal K-point sampling. In this case, TribChem fixes the energy cutoff at the optimal value while varying the K-point density until convergence is reached. We fix the energy convergence of this sampling to 1 meV/atom. For K, an optimal K-point density of 5.4 Å^{-1} is identified.

6.2. Slab and Surface Energy Calculations. The second step in the study of an interface is the generation and optimization of the two mating surfaces; as previously explained, TribChem creates a slab structure and then optimizes its thickness and computes the surface energy. To execute this workflow, the following command must be used (here, we consider the K(110) surface):

```
tribchem workflow converge_slab

mids="[mp-58]" formulas="[K]"

miller="[[1, 1, 0]]"

tmin=4 tmax=12
```

`tmin` and `tmax` identify the range of slab thicknesses to be considered (4 to 12 layers in this case). For K(110), using eq 2, 4 layers are identified as the optimal slab thickness. Correspondingly, the surface energy E_γ is 0.11 J/m^2, which matches precisely the value reported by the Materials Project.[37]

6.3. Homogeneous Interface and PES Calculation. Homogeneous interfaces are the simplest interfaces that can be studied, as no cell matching is required. To generate and compute both the PES and adhesion energy of the K(110)/K(110) interface, the following command must be used:

```
tribchem workflow

calc_interface wf="[interface, pes, adhesion]"

mids="[mp-58, mp-58]" formulas="[K, K]"

miller="[[1, 1, 0], [1, 1, 0]]"
```

The `wf="[interface, pes, adhesion]"` command indicates that the user wants to launch the `interface`, `pes`, and `adhesion` workflows altogether. First of all, the `interface` workflow is launched, and the interface structure of the required system is generated. A visual representation of the K−K interface is shown in Figure 5.

Second, the `pes` workflow begins. TribChem builds the PES landscape by mating the two slabs in different relative lateral positions and by collecting, for each displacement, the adhesion energy γ is calculated from eq 3. The relative lateral positions are identified by TribChem by pairing the high-symmetry points of the two surfaces. For the K(110) surface, four high-symmetry points (the on-top, hollow, long bridge, and short bridge sites) are found, generating six nonequivalent lateral displacements for the K(110)/K(110) interface. Collecting the adhesion energy in every lateral displacement and interpolating them through the radial basis function makes it possible to obtain a 2D representation of the PES, as shown in Figure 6.

The final interface adhesion energy is the PES minimum, a value that is identified and stored in the database with the command `adhesion`. For the K(110)/K(110) interface, we obtain an adhesion energy of 0.20 J/m^2, almost twice the computed surface energy (0.11 J/m^2). This result validates the three interface workflows as, by definition of adhesion energy, its value in the homostructure should be twice the surface energy.

6.4. Heterogeneous Interface with Other Metals. Finally, we generated the heterostructures by mating the K(110) surface with the most stable faces of 17 different metals: Ag, Al, Au, Cr, Cu, Fe, Ir, Mg, Mo, Ni, Pt, Rh, Sc, Ti, V, W, and Zn. The surfaces are matched by MPInterface,[22] which makes use of the Zur algorithm.[38] Such a procedure allows the optimization of the in-plane supercell area within the allowed mismatch for the cell sides and angles. An example of the result

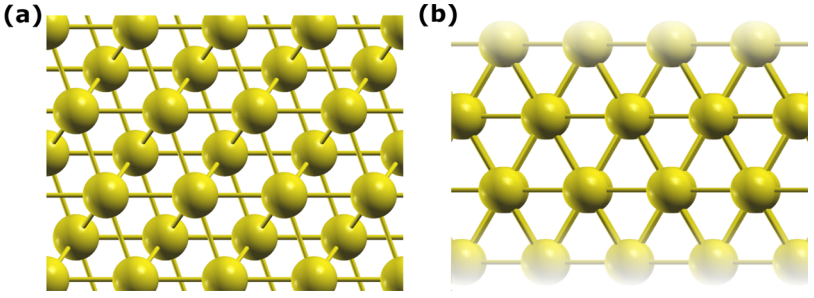


Figure 5. Top (a) and lateral (b) ball and stick representation of a K interface. The yellow color represents the K atoms.

https://doi.org/10.1021/acs.jctc.3c00459
J. Chem. Theory Comput. 2023, 19, 5231−5241

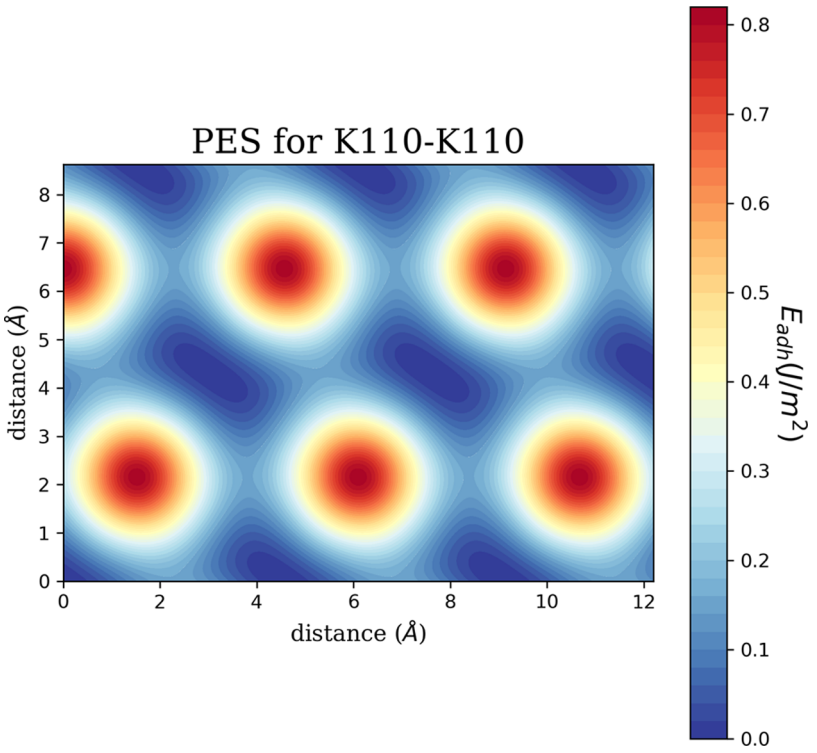


Figure 6. PES for the homogeneous interface obtained by mating two K(110) surfaces.

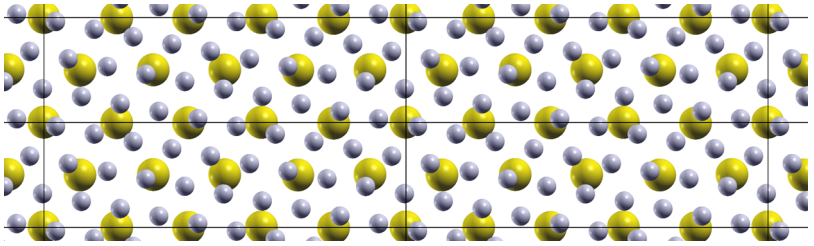


Figure 7. Top view of the interfacing atomic layers of the K/Al interface. The yellow (gray) balls represent the K (Al) atoms. The solid black lines are the boundaries of the supercells.

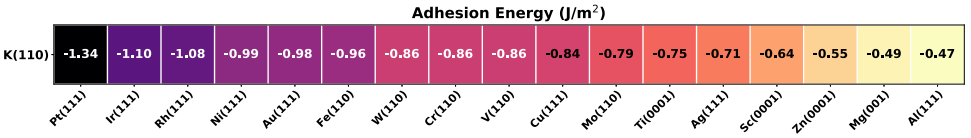


Figure 8. Adhesion energy data set for the heterogeneous interfaces obtained by combining K with 17 different metals. The data are sorted in ascending order.

of this matching procedure is shown in Figure 7 for the K(110)/Al(111) interface case.

Using the same command shown for the homogeneous interface, TribChem generates the interfaces in different relative lateral positions to identify the energy minimum and compute the adhesion energy of the system. The results of these calculations are reported in Figure 8.

K has the largest adhesion with Pt, whereas the lowest interaction is with Al. In ref 20, more than a hundred heterogeneous interfaces have been studied employing this workflow. A large amount of data, in that case, allowed for the use of a machine-learning algorithm to determine predictive models of adhesion energies in terms of the properties of single slabs.

7. SUMMARY

We present TribChem, an advanced high-throughput software to study solid interfaces through first-principles calculations. The program is based on the Atomate and FireWorks workflow managers and relies on MongoDB for database communication.

TribChem presents a modular structure, which allows for the execution of different types of calculations and can be easily extended to include new features.

At present, the first unit of TribChem is devoted to bulk calculations, which are used to set the optimal computational parameters for studying a selected element. Moreover, bulk properties, such as the cohesion energy and the bulk modulus, are calculated. In the second unit, an accurate model for the surface is automatically constructed by converging the

https://doi.org/10.1021/acs.jctc.3c00459
J. Chem. Theory Comput. 2023, 19, 5231−5241

calculated surface energy as a function of the slab thickness. In the third TribChem unit, the interface is constructed by mating two different surfaces within a common cell able to accommodate the two different lattices with a small mismatch. The potential energy surface that describes how the interaction energy of the two mated surfaces changes as a function of their relative lateral position is calculated in a smart way using the surface's high-symmetry points. The PES absolute minimum is used to define the adhesion energy between the two surfaces, while the sliding path with the highest statistical weight is used to calculate the interfacial shear strength.

The implementation of the above features has been technically described in the present manuscript, where an example for potassium bulk, surface, and 17 heterointerfaces have been presented.

TribChem represents a significant step forward in the study of solid−solid interfaces, where a systematic high-throughput analysis was still lacking. For the foreseeable future, we plan to expand TribChem to include lubricant additives, contaminants, and rough surfaces for a more realistic description of the tribological interfaces.

AUTHOR INFORMATION

Corresponding Author

M. Clelia Righi − *Physics and Astronomy Department, University of Bologna, 40137 Bologna, Italy;* orcid.org/0000-0001-5115-5801; Email: mariaclelia.righi@unimore.it

Authors

Gabriele Losi − *Physics and Astronomy Department, University of Bologna, 40137 Bologna, Italy*

Omar Chehaimi − *Physics and Astronomy Department, University of Bologna, 40137 Bologna, Italy*

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.jctc.3c00459

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

The development of TribChem is part of the "Advancing Solid Interface and Lubricants by First-Principles Material Design (SLIDE)" project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement no. 865633). The authors want to thank Dr. Paolo Restuccia for providing the data of the example shown in the manuscript and for the fruitful discussion.

REFERENCES

(1) Curtarolo, S.; Hart, G. L. W.; Nardelli, M. B.; Mingo, N.; Sanvito, S.; Levy, O. The high-throughput highway to computational materials design. *Nat. Mater.* **2013**, *12*, 191−201.

(2) Haastrup, S.; Strange, M.; Pandey, M.; Deilmann, T.; Schmidt, P. S.; Hinsche, N. F.; Gjerding, M. N.; Torelli, D.; Larsen, P. M.; Riis-Jensen, A. C.; Gath, J.; Jacobsen, K. W.; Mortensen, J. J.; Olsen, T.; Thygesen, K. S. The Computational 2D Materials Database: high-throughput modeling and discovery of atomically thin crystals. *2D Mater.* **2018**, *5*, No. 042002.

(3) Li, Z.; Yoon, J.; Zhang, R.; Rajabipour, F.; Srubar, W. V., III; Dabo, I.; Radlińska, A. Machine learning in concrete science: applications, challenges, and best practices. *npj Comput. Mater.* **2022**, *8*, No. 127.

(4) Rosen, A. S.; Fung, V.; Huck, P.; O'Donnell, C. T.; Horton, M. K.; Truhlar, D. G.; Persson, K. A.; Notestein, J. M.; Snurr, R. Q. High-throughput predictions of metal-organic framework electronic properties: theoretical challenges, graph neural networks, and data exploration. *npj Comput. Mater.* **2022**, *8*, No. 112.

(5) Hebnes, O. L.; Bathen, M. E.; Schøyen, Ø. S.; Winther-Larsen, S. G.; Vines, L.; Hjorth-Jensen, M. Predicting solid state material platforms for quantum technologies. *npj Comput. Mater.* **2022**, *8*, No. 207.

(6) Choudhary, K.; DeCost, B.; Chen, C.; Jain, A.; Tavazza, F.; Cohn, R.; Park, C. W.; Choudhary, A.; Agrawal, A.; Billinge, S. J. L.; Holm, E.; Ong, S. P.; Wolverton, C. Recent advances and applications of deep learning methods in materials science. *npj Comput. Mater.* **2022**, *8*, No. 59.

(7) Wines, D.; Choudhary, K.; Biacchi, A. J.; Garrity, K. F.; Tavazza, F. High-Throughput DFT-Based Discovery of Next Generation Two-Dimensional (2D) Superconductors. *Nano Lett.* **2023**, *23*, 969−978.

(8) Tran, K.; Palizhati, A.; Back, S.; Ulissi, Z. W. Dynamic Workflows for Routine Materials Discovery in Surface Science. *J. Chem. Inf. Model.* **2018**, *58*, 2392−2400.

(9) Rosen, A. S.; Notestein, J. M.; Snurr, R. Q. Identifying Promising Metal−organic Frameworks for Heterogeneous Catalysis via High-Throughput Periodic Density Functional Theory. *J. Comput. Chem.* **2019**, *40*, 1305−1318.

(10) Rittiruam, M.; Noppakhun, J.; Setasuban, S.; Aumnongpho, N.; Sriwattana, A.; Boonchuay, S.; Saelee, T.; Wangphon, C.; Ektarawong, A.; Chammingkwan, P.; Taniike, T.; Praserthdam, S.; Praserthdam, P. High-throughput materials screening algorithm based on first-principles density functional theory and artificial neural network for high-entropy alloys. *Sci. Rep.* **2022**, *12*, No. 16653.

(11) Jain, A.; Ong, S. P.; Hautier, G.; Chen, W.; Richards, W. D.; Dacek, S.; Cholia, S.; Gunter, D.; Skinner, D.; Ceder, G.; Persson, K. A. Commentary: The Materials Project: A materials genome approach to accelerating materials innovation. *APL Mater.* **2013**, *1*, No. 011002.

(12) Jain, A.; Ong, S. P.; Chen, W.; Medasani, B.; Qu, X.; Kocher, M.; Brafman, M.; Petretto, G.; Rignanese, G.-M.; Hautier, G.; Gunter, D.; Persson, K. A. FireWorks: a dynamic workflow system designed for high-throughput applications. *Concurr. Comput. Pract. Exp.* **2015**, *27*, 5037−5059.

(13) Kresse, G.; Hafner, J. Ab initio molecular dynamics for liquid metals. *Phys. Rev. B* **1993**, *47*, 558−561.

(14) Kresse, G.; Hafner, J. Ab initio molecular-dynamics simulation of the liquid-metal-amorphous-semiconductor transition in germanium. *Phys. Rev. B* **1994**, *49*, No. 14251.

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

(16) Kresse, G.; Furthmüller, J. Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set. *Phys. Rev. B* **1996**, *54*, No. 11169.

(17) Restuccia, P.; Levita, G.; Wolloch, M.; Losi, G.; Fatti, G.; Ferrario, M.; Righi, M. Ideal adhesive and shear strengths of solid interfaces: A high throughput ab initio approach. *Comput. Mater. Sci.* **2018**, *154*, 517−529.

(18) Pizzi, G.; Cepellotti, A.; Sabatini, R.; Marzari, N.; Kozinsky, B. AiiDA: automated interactive infrastructure and database for computational science. *Comput. Mater. Sci.* **2016**, *111*, 218−230.

(19) Wolloch, M.; Losi, G.; Ferrario, M.; Righi, M. C. High-throughput screening of the static friction and ideal cleavage strength of solid interfaces. *Sci. Rep.* **2019**, *9*, No. 17062.

(20) Restuccia, P.; Losi, G.; Chehaimi, O.; Marsili, M.; Righi, M. C. High-Throughput First-Principles Prediction of Interfacial Adhesion Energies in Metal-on-Metal Contacts. *ACS Appl. Mater. Interfaces* **2023**, *15*, 19624−19633.

(21) Ong, S. P.; Richards, W. D.; Jain, A.; Hautier, G.; Kocher, M.; Cholia, S.; Gunter, D.; Chevrier, V. L.; Persson, K. A.; Ceder, G. Python Materials Genomics (pymatgen): A robust, open-source python library for materials analysis. *Comput. Mater. Sci.* **2013**, *68*, 314−319.

(22) Mathew, K.; Singh, A. K.; Gabriel, J. J.; Choudhary, K.; Sinnott, S. B.; Davydov, A. V.; Tavazza, F.; Hennig, R. G. MPInterfaces: A Materials Project based Python tool for high-throughput computational screening of interfacial systems. *Comput. Mater. Sci.* **2016**, *122*, 183−190.
(23) Giannozzi, P.; Baroni, S.; Bonini, N.; Calandra, M.; Car, R.; Cavazzoni, C.; Ceresoli, D.; Chiarotti, G. L.; Cococcioni, M.; Dabo, I.; Corso, A. D.; de Gironcoli, S.; Fabris, S.; Fratesi, G.; Gebauer, R.; Gerstmann, U.; Gougoussis, C.; Kokalj, A.; Lazzeri, M.; Martin-Samos, L.; Marzari, N.; Mauri, F.; Mazzarello, R.; Paolini, S.; Pasquarello, A.; Paulatto, L.; Sbraccia, C.; Scandolo, S.; Sclauzero, G.; Seitsonen, A. P.; Smogunov, A.; Umari, P.; Wentzcovitch, R. M. QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials. *J. Phys.: Condens. Matter* **2009**, *21*, No. 395502.
(24) Thompson, A. P.; Aktulga, H. M.; Berger, R.; Bolintineanu, D. S.; Brown, W. M.; Crozier, P. S.; in 't Veld, P. J.; Kohlmeyer, A.; Moore, S. G.; Nguyen, T. D.; Shan, R.; Stevens, M. J.; Tranchida, J.; Trott, C.; Plimpton, S. J. LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. *Comp. Phys. Commun.* **2022**, *271*, No. 108171.
(25) Wolloch, M.; Losi, G.; Chehaimi, O.; Yalcin, F.; Ferrario, M.; Righi, M. C. High-throughput generation of potential energy surfaces for solid interfaces. *Comput. Mater. Sci.* **2022**, *207*, No. 111302.
(26) MongoDB, I. MongoDB. https://www.mongodb.com/. (accessed April 27, 2023).
(27) Wolloch, M.; Levita, G.; Restuccia, P.; Righi, M. C. Interfacial Charge Density and Its Connection to Adhesion and Frictional Forces. *Phys. Rev. Lett.* **2018**, *121*, No. 026804.
(28) PyMongo https://pymongo.readthedocs.io/en/stable/. (accessed April 27, 2023).
(29) TribChem User Guide https://gitlab.com/triboteam/tribchem/-/wikis/home.
(30) Mathew, K.; Montoya, J. H.; Faghaninia, A.; Dwarakanath, S.; Aykol, M.; Tang, H.; heng Chu, I.; Smidt, T.; Bocklund, B.; Horton, M.; Dagdelen, J.; Wood, B.; Liu, Z.-K.; Neaton, J.; Ong, S. P.; Persson, K.; Jain, A. Atomate: A high-level interface to generate, execute, and analyze computational materials science workflows. *Comput. Mater. Sci.* **2017**, *139*, 140−152.
(31) Larsen, A. H.; Mortensen, J. J.; Blomqvist, J.; Castelli, I. E.; Christensen, R.; Du lak, M.; Friis, J.; Groves, M. N.; Hammer, B.; Hargus, C.; Hermes, E. D.; Jennings, P. C.; Jensen, P. B.; Kermode, J.; Kitchin, J. R.; Kolsbjerg, E. L.; Kubal, J.; Kaasbjerg, K.; Lysgaard, S.; Maronsson, J. B.; Maxson, T.; Olsen, T.; Pastewka, L.; Peterson, A.; Rostgaard, C.; Schiøtz, J.; Schütt, O.; Strange, M.; Thygesen, K. S.; Vegge, T.; Vilhelmsen, L.; Walter, M.; Zeng, Z.; Jacobsen, K. W. The atomic simulation environment−a Python library for working with atoms. *J. Phys.: Condens. Matter* **2017**, *29*, No. 273002.
(32) Hunter, J. D. Matplotlib: A 2D graphics environment. *Comput. Sci. Eng.* **2007**, *9*, 90−95.
(33) Harris, C. R.; Millman, K. J.; van der Walt, S. J.; Gommers, R.; Virtanen, P.; Cournapeau, D.; Wieser, E.; Taylor, J.; Berg, S.; Smith, N. J.; Kern, R.; Picus, M.; Hoyer, S.; van Kerkwijk, M. H.; Brett, M.; Haldane, A.; del Río, J. F.; Wiebe, M.; Peterson, P.; Gérard-Marchant, P.; Sheppard, K.; Reddy, T.; Weckesser, W.; Abbasi, H.; Gohlke, C.; Oliphant, T. E. Array programming with NumPy. *Nature* **2020**, *585*, 357−362.
(34) Virtanen, P.; Gommers, R.; Oliphant, T. E.; Haberland, M.; Reddy, T.; Cournapeau, D.; Burovski, E.; Peterson, P.; Weckesser, W.; Bright, J.; van der Walt, S. J.; Brett, M.; Wilson, J.; Millman, K. J.; Mayorov, N.; Nelson, A. R. J.; Jones, E.; Kern, R.; Larson, E.; Carey, C. J.; Polat, I.; Feng, Y.; Moore, E. W.; VanderPlas, J.; Laxalde, D.; Perktold, J.; Cimrman, R.; Henriksen, I.; Quintero, E. A.; Harris, C. R.; Archibald, A. M.; Ribeiro, A. H.; Pedregosa, F.; van Mulbregt, P.; et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nat. Methods* **2020**, *17*, 261−272.
(35) Birch, F. Finite Elastic Strain of Cubic Crystals. *Phys. Rev.* **1947**, *71*, 809−824.
(36) Liu, L.-G. Compression and polymorphism of potassium to 400 kbar. *J. Phys. Chem. Solids* **1986**, *47*, 1067−1072.
(37) Tran, R.; Xu, Z.; Radhakrishnan, B.; Winston, D.; Sun, W.; Persson, K. A.; Ong, S. P. Surface Energies of Elemental Crystals. *Sci. Data* **2016**, *3*, No. 160080.
(38) Zur, A.; McGill, T. C. Lattice match: An application to heteroepitaxy. *J. Appl. Phys.* **1984**, *55*, 378−386.

https://doi.org/10.1021/acs.jctc.3c00459
J. Chem. Theory Comput. 2023, 19, 5231−5241