Computer Physics Communications 226 (2018) 165–179

Contents lists available at ScienceDirect

Computer Physics Communications

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

PyCDT: A Python toolkit for modeling point defects in semiconductors and insulators☆

Danny Broberg [a,b,*,1], Bharat Medasani [c,d,**,1], Nils E.R. Zimmermann [c,**,1], Guodong Yu [e], Andrew Canning [c], Maciej Haranczyk [c], Mark Asta [a,b], Geoffroy Hautier [e,*]

[a] Department of Materials Science and Engineering, University of California, Berkeley, CA 94720, USA
[b] Materials Sciences Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA
[c] Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA
[d] Physical and Computational Sciences Directorate, Pacific Northwest National Laboratory, Richland, WA 99354, USA
[e] Institute of Condensed Matter and Nanosciences (IMCN), Université catholique de Louvain, 1348 Louvain-la-Neuve, Belgium

ARTICLE INFO

Article history:
Received 22 December 2016
Received in revised form 17 October 2017
Accepted 2 January 2018
Available online 13 February 2018

Keywords:
Point defects
Charged defects
Semiconductors
Insulators
Density functional theory
Python

ABSTRACT

Point defects have a strong impact on the performance of semiconductor and insulator materials used in technological applications, spanning microelectronics to energy conversion and storage. The nature of the dominant defect types, how they vary with processing conditions, and their impact on materials properties are central aspects that determine the performance of a material in a certain application. This information is, however, difficult to access directly from experimental measurements. Consequently, computational methods, based on electronic density functional theory (DFT), have found widespread use in the calculation of point-defect properties. Here we have developed the Python Charged Defect Toolkit (PyCDT) to expedite the setup and post-processing of defect calculations with widely used DFT software. PyCDT has a user-friendly command-line interface and provides a direct interface with the Materials Project database. This allows for setting up many charged defect calculations for any material of interest, as well as post-processing and applying state-of-the-art electrostatic correction terms. Our paper serves as a documentation for PyCDT, and demonstrates its use in an application to the well-studied GaAs compound semiconductor. We anticipate that the PyCDT code will be useful as a framework for undertaking readily reproducible calculations of charged point-defect properties, and that it will provide a foundation for automated, high-throughput calculations.

Program summary
Program title: PyCDT
Program Files doi: http://dx.doi.org/10.17632/7vzk5gxzh3.1
Licensing Provisions: MIT License.
Programming language: Python
External routines/libraries: NumPy [1], matplotlib [2], and Pymatgen [3],
Nature of problem: Computing the formation energies and stable point defects with finite size supercell error corrections for charged defects in semiconductors and insulators.
Solution method: Automated setup, and parsing of defect calculations, combined with local use of finite size supercell corrections. All combined into a code with a standard user-friendly command line interface that leverages a core set of tools with a wide range of applicability.
Additional comments: This article describes version 1.0.0. Program obtainable from https://bitbucket.org/mbkumar/pycdt

☆ This paper and its associated computer program are available via the Computer Physics Communication homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding address: Department of Materials Science and Engineering, 210 Hearst Mining Building, University of California, Berkeley, CA 94720-1760, USA.
** Corresponding authors.
E-mail addresses: dbroberg@berkeley.edu (D. Broberg), bharat.medasani@pnnl.gov (B. Medasani), nerz@lbl.gov (N.E.R. Zimmermann), geoffroy.hautier@uclouvain.be (G. Hautier).

[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.cpc.2018.01.004

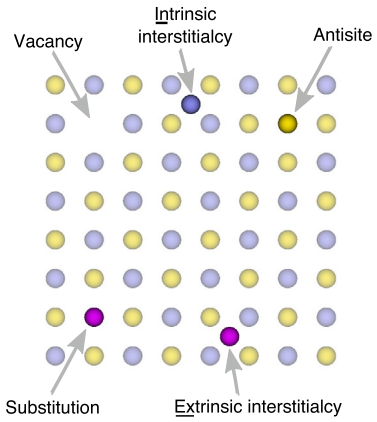

Fig. 1. Intrinsic point defects (top: vacancy, intrinsic interstitialcy, antisite) and extrinsic point defects (bottom: substitution, extrinsic interstitialcy).

1. Introduction

Point defects in semiconductors and insulators govern a range of mechanical, transport, electronic, and optoelectronic properties [1–5]. Due to the fact that the properties of these defects are difficult to characterize fully from experiment [5,6], computational tools have been widely applied. Many applications such as lanthanide-doped scintillator materials [7,8], transparent conducting oxide materials [9–11], photovoltaic materials [12,13], and new thermoelectric materials [14,15] have benefited from leveraging theory for calculating point defect properties in next generation technologies.

For this reason, calculations using electronic density functional theory (DFT) have arisen as a reliable route to explore the dopability of materials at the atomic scale [6,16–18]. However, two sources of error in the associated point defect calculations limit the application of charged defect DFT efforts in a high-throughput framework. First, semi-local exchange–correlation approximations (e.g., generalized gradient approximation (GGA)) can severely underestimate the band gap so that usage of post DFT methods becomes pivotal (e.g., GW [19–21] and GGA+U methods [22,23], and hybrid functionals [24]). Second, applying periodic boundary conditions with finite sized defect supercells to model point defects makes a defect interact with its own images [6,25], thus, causing departure from the key assumption made in the dilute limit formation energy formalism [6,25]. In the case of charged point defects, the finite sized supercell assumption also introduces the need for correcting the electrostatic potential [6,17]. Typically, the strongest defect–defect interaction is the Coulomb interaction between charged point defects. Based on well-known scaling laws, these interactions were first treated with computationally costly supercell scaling methods, which require multiple calculations for each defect [25]. A faster route to computing defect formation energies became available with the development of *a posteriori* correctional techniques. While the *a posteriori* corrections allow for fewer calculations to be performed, their usage requires experience in addressing issues arising from delocalization of the defect wavefunction [17]. Furthermore, the calculations are often resource demanding and tedious because of the large number of pre- and post-processing steps involved.

To address these problems we have developed the Python Charged Defects Toolkit (PyCDT), which enables expanded applications in the context of materials discovery and design. Our python-based tools automate the setup and analysis of DFT calculations of isolated intrinsic and extrinsic point defects (vacancies, antisites, substitutions, and interstitials) in semiconductors and insulators. While other efforts have recently been made available with similar objectives [26–28], PyCDT is unique in its direct queries to the Materials Project [29] database (expediting chemical potential and stability analysis for Perdew–Burke–Ernzerhof (PBE) GGA calculations) [30].

A central objective of defects modeling in non-metallic systems is determining the relative stability of different defect charge states. PyCDT therefore implements the defect formation energy formalism reviewed in Sections 2.1–2.4. To minimize the errors in defect formation energies arising from the periodic boundary conditions, PyCDT supports the commonly used correction scheme due to Freysoldt et al. [31] and its extension to anisotropic systems by Kumagai and Oba [32] (Section 2.4). Our tools also include charge-state assignment procedures developed on the basis of extensive literature data (Section 2.5) and an effective interstitial-finding algorithm [33] (Section 2.6). Furthermore, PyCDT provides a user-friendly command-line interface that provides ready access to all tools. We demonstrate the setup and analysis of the defect calculations from the command line in Section 3 using gallium arsenide (GaAs) as an example system and by employing the widely-used Vienna Ab initio Simulation Package (VASP) [34,35] as a backend DFT software. In Section 4, we validate the finite-size charge correction schemes implemented and verify the results obtained for GaAs. We emphasize that our approaches and implementations are entirely general, thus, seamlessly facilitating extensions to other DFT packages.

2. Background and methods

In general, point defects can be divided into two categories: intrinsic and extrinsic [5]. Intrinsic (or native [5,6]) point defects (Fig. 1: top) involve only chemical species that are part of the perfect bulk material (e.g., Si in silicon). For elemental materials, there are two basic intrinsic defect types: vacancies (e.g., vac_{Si}, denoting a vacancy on Si site) and self-interstitials (e.g., Si_i). For compounds (e.g., GaAs), there is an additional defect type: antisites (e.g., As_{Ga}). Because intrinsic point defects are equilibrium defects due to configurational entropy, they can be well described and their occurrence understood in the framework of equilibrium thermodynamics (formation energies, E^f, used to predict equilibrium concentrations, c).

Extrinsic defects (Fig. 1: bottom), which are also referred to as impurities [5,6], introduce a foreign chemical species into the perfect bulk material [1]. These include substitutional defects (e.g., Mn_{Ga}) and extrinsic interstitials (e.g., Mn_i). We distinguish between extrinsic and

intrinsic defects because impurities are often inserted on purpose (intentional doping) under well-defined conditions to achieve desired material properties; in particular, electrical and optoelectronic properties [5]. The conditions under which extrinsic defects are inserted (e.g., via implantation or quenching) often differ extremely from the thermodynamic equilibrium assumption made in the assessment of intrinsic point defects. Despite the limited conceptual applicability, the thermodynamic framework still represents the most commonly pursued route to assessing "dopability" of materials [6–15].

In contrast to metals, point defects in semiconductors and insulators can carry a charge [5] localized around the defect site. Physically, these charged point defects introduce states within the band gap which can trap charge carriers (electrons and holes). Defect states that are close to the band edges are able to ionize to create free carriers, while states that are deep in the gap lead to strong carrier trapping. This may be wanted (e.g., in photovoltaics [13]) or not (e.g., solid-state electrolyte batteries [36]). Because many technological applications use intentional doping to improve performance, knowledge of the capacity to dope a material ("dopability") is desirable and motivates the exploration of defect properties with theoretical methods.

2.1. Formalism for equilibrium point defects

The thermodynamics of point defects has been the subject of many excellent reviews (see, for example, Refs. [6,16,17] and references therein), which have presented and discussed the underlying physics and properties in great detail. In the following sections, we focus on describing the procedures implemented in PyCDT for computing quantities of interest for point defects (i.e., formation energies and transition levels) with DFT calculations. The applications of the defect formalism to be described are limited by the intrinsic shortcomings of DFT (e.g., the well-known underestimation of the band gap [17]).

There is a hierarchy of DFT-based methods that can be used within the defect formalism implemented by PyCDT. The simplest approximation in DFT is the use of a semilocal functional (i.e., local-density approximation (LDA), GGA), which is computationally most efficient, but has well-known limitations due to band gap inaccuracies. Higher levels of theory include hybrid-functionals and meta-GGA, both of which can be used to achieve higher accuracy, however, at an increased computational cost [37,38]. Despite the limitations of semilocal DFT, defect calculations have proven useful for revealing the dominating defects under different growth conditions encountered in many experiments, such as the III–V semiconductors [39]. While recent developments in hybrid functionals and meta-GGA have shown promise in addressing the inherent limitations in accuracy associated with semi-local functionals [37,38], recent evidence shows that new improvements to the approximations for exchange and correlation in one system do not always yield universal improvements for other systems with similar chemistries [40]. With this fact considered, semi-local approximations at least have the benefit of having predictable errors which can be corrected with appropriate techniques [17].

While PyCDT's unique interface with the Materials Project (MP) database, which is composed of GGA and GGA+U level data, suggests a restriction to semi-local approaches, PyCDT has many functionalities which help place defect formation energetics closer to those obtained from higher levels of theory. One feature that is particularly useful is the ability for PyCDT to help expedite the setup and parsing stages of defect calculations performed on higher levels of DFT theory (*e.g.* for improved chemical potentials, setting up a user's personal phase diagram calculation based on a composition of interest). These features are described further in the following sections.

2.2. Defect formation energies

The primary quantity of interest is the formation energy, $E^{\mathrm{f}}[X]$, which is the energy cost to form or create an isolated defect, X, in a bulk or host material. The formation energy of an isolated defect (i.e., in the dilute limit) depends on the defect charge state, q: $E^{\mathrm{f}}[X^q]$. It can be calculated from DFT supercells using:

$$E^{\mathrm{f}}[X^q] = E_{\mathrm{tot}}[X^q] - E_{\mathrm{tot}}[\mathrm{bulk}] - \sum_i n_i \mu_i + qE_{\mathrm{F}} + E_{\mathrm{corr}}. \tag{1}$$

We illustrate this equation graphically in Fig. 2 and note that each term will be described in detail in subsequent sub-sections. $E_{\mathrm{tot}}[X^q]$ and $E_{\mathrm{tot}}[\mathrm{bulk}]$ are the total DFT-derived energies of the defective and pristine bulk supercells, respectively. The third term, $-\sum_i n_i \mu_i$, is a summation over the atomic chemical potentials, or the energy cost of an atom, μ_i, being added ($n_i = +1$) or removed ($n_i = -1$) from the bulk undefective supercell. The atomic chemical potential can reflect the growth conditions of the material, allowing this formalism to be used to guide defect engineering approaches (cf., Section 2.3). The fourth term, qE_{F}, represents the energetic cost of adding or removing electrons, where E_{F} is the Fermi energy, which serves as the chemical potential of the electron reservoir. The Fermi energy is usually referenced to the valence band maximum from a band structure calculation, such that the formation energy can be plotted as a function of the Fermi energy across the band gap. Finally, E_{corr} is a correction term due to the presence of periodic images that becomes necessary for charged defects in DFT supercell calculations. This correction has drawn significant attention from the defects modeling community, resulting in a number of alternative computational approaches that are discussed in more detail in Section 2.4.

2.3. Chemical potentials

The atomic chemical potential is associated with the thermodynamic energy cost for exchanging atoms between the defect and a thermodynamic reservoir. The individual chemical potentials are set by the composition of the material (e.g., the mole fraction of As in GaAs) which itself is determined by the defect formation energies. Hence, two approaches can be used to derive the individual chemical potentials. One involves the use of a statistical thermodynamic formalism (*canonical* ensemble approach), to compute the concentration dependent free energy of the compound and to derive the relationship between the chemical potentials and composition [41]. More frequently, bounds on the chemical potentials are set (*grand-canonical* ensemble approach), in a manner first defined by Zhang and Northrup [42], from zero-temperature energies alone. Only the grand-canonical approach is currently included in PyCDT, whereas future versions will also support canonical approaches to chemical potential calculations. Below we demonstrate the grand-canonical approach for the simple example of GaAs.

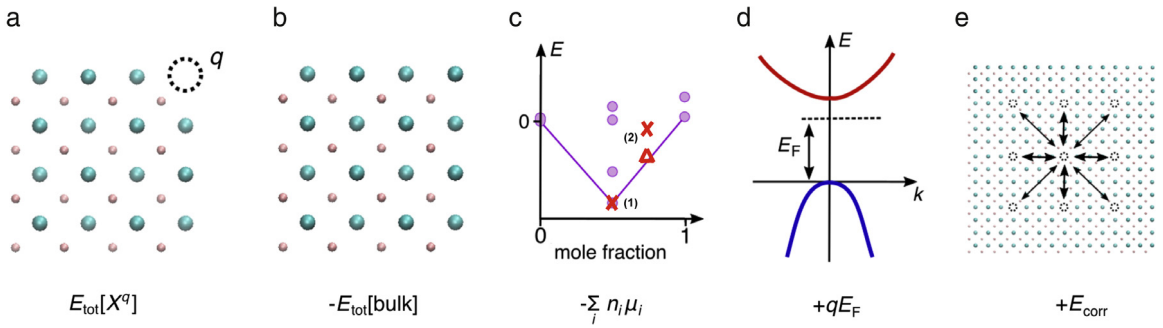

Fig. 2. Different contributions to the formation energy: (a) energy of defective supercell in charge state q, (b) energy of pristine bulk supercell, (c) atomic chemical potential computed from the ground state hull, (d) electron/hole chemical potential generated from electron reservoir, and (e) correction terms to account for defect–defect interactions arising from periodic boundary conditions as well as for the homogeneous background charge which requires potential re-alignment. The different labeled X's given in 2c show stable and metastable compounds. The latter case yields a complication which will be discussed in Section 2.3.

The bulk energy (or free energy at finite temperature) per formula unit, μ^0_{GaAs}, fixes a relation for the chemical potentials of gallium, μ_{Ga}, and arsenic, μ_{As}, respectively: $\mu^0_{GaAs} = \mu_{Ga} + \mu_{As}$. For Ga-rich compositions, the values of μ_{Ga} are constrained by stability of the compound relative to the precipitation of excess Ga to form a bulk Ga phase. At zero temperature, this constraint can be expressed as $\mu_{Ga} < \mu^0_{Ga}$, where μ^0_{Ga} is the energy per atom of bulk Ga. Thus, one extremum can be selected as the "Ga-rich" limit, where $\mu_{Ga} = \mu^0_{Ga}$. The atomic chemical potential of As is then fixed: $\mu_{As} = \mu^0_{GaAs} - \mu^0_{Ga}$. The same approach holds for As-rich compositions (above: interchange Ga and As labels with each other). While the Ga-As system has a phase diagram with just one unique compound, the more general case has multiple stable compounds, which requires the limits of stability to be expressed in terms of the formation of compounds with neighboring compositions within the phase diagram. In general, the chemical potentials in an n-component system will be defined in PyCDT by defining the limits of stability for the different possible n-phase states of equilibria.

As an example, consider the Sn-Se system which has a 0 K ground-state hull that contains the phases Sn, SnSe, $SnSe_2$ and Se. When calculating defects in the SnSe phase, the "Se-rich" limit would instead be defined by equilibrium with the $SnSe_2$ phase Eq. (2), combined with the stability condition for bulk SnSe Eq. (3). This forms a system of equations for the chemical potentials of μ_{Sn} and μ_{Se}.

$$\mu^0_{SnSe_2} > \mu_{Sn} + 2\mu_{Se} \tag{2}$$

$$\mu^0_{SnSe} = \mu_{Sn} + \mu_{Se}. \tag{3}$$

This formalism for calculating equilibrium bounds on the chemical potentials requires knowledge of the ground-state hull, which governs the zero-temperature limit of the phase diagram of the system. The advantage of this formalism is that defect formation energies can be obtained entirely from first principles calculations, rendering experimental input unnecessary. The drawback is that one must compute the full phase diagram of the system using the same functional choice used for the defect calculations. In applications of the PyCDT code based on the PBE-GGA exchange–correlation potential, PyCDT uses the ground-state hulls that are made available through the MP database [29]. PyCDT has integrated functionality that queries the MP database for every computed DFT entry in the phase diagram so that no new calculations are required to compute the bounds on the chemical potentials. Note that correct usage of the MP data for defect calculations requires consistency between personal defect calculations and the calculations from the MP. This is easily checked through the compatibility tools available in pymatgen [43]. If the bulk phase is thermodynamically stable and is not already computed in the MP database, PyCDT manually inserts the computed phase into the phase diagram and then provides all of the associated bounds on the chemical potentials. If a user prefers to compute chemical potentials on a different level of theory than is provided by the MP, the core code of PyCDT can be used to setup and calculate atomic chemical potentials through first pulling the composition's phase diagram, and using the structural information to setup a personal phase diagram calculation for the user. In a similar manner, this information can be setup for any DFT code desired by the user through the use of Pymatgen's code agnostic classes.

For highly correlated systems such as transition metal oxides, MP settings default to GGA+U. When computing the phase diagram that contains a mixture of GGA and GGA+U computed phases, MP employs the mixing scheme of Jain et al. [44], which adds an empirical correction to the energies of GGA+U compounds. The mixing scheme was shown to give formation energies that are consistent with experimental data with a mean absolute relative error of under 2%. The resulting correction term to the chemical potentials was found to be important in several defect studies [45,46].

One complication that should be mentioned is the case where the compound under consideration does not reside on the convex hull. That is, the compound is higher in energy than another compound with the same composition or with respect to phase separation to compounds with other compositions. In this instance, the calculation is predicting the compound to not be present in the equilibrium phase diagram in the limit of zero temperature. For small energy-above-hull values, this situation could indicate that the compound is stabilized by entropic contributions at finite temperature [47], or could be an artifact of the previously mentioned inaccuracies of DFT [48], or the experimentally synthesized phase exists in a state of metastable equilibrium [49]. Regardless of the reason, a positive energy-above-hull value presents a practical problem for defining the chemical potential.

In such cases, PyCDT issues a warning and the chemical potentials used are with respect to the phases in equilibrium at the given composition in the phase diagram. In Fig. 2c, this situation is graphically represented by the data points labeled "(2)", where the red X above the hull is the compound of interest and PyCDT uses the red triangle to define a set of compounds in equilibrium for defining the atomic chemical potentials. In these instances, the computed defect physics should be interpreted with caution. The user may wish to define the chemical-potential limits based on more detailed knowledge of the growth conditions or compute the chemical potentials from a full finite-temperature free energy model of the compound of interest.

2.4. Periodic supercell corrections

Periodic boundary conditions (PBCs) are the standard way to deal with the regular arrangement of crystalline solids in DFT calculations. Once a defect is introduced, PBCs can give rise to sizable interactions of the defect with its periodic images, contrasting the assumption made above for the dilute-defect limit. Because this limit is consistent with the thermodynamic formalism outlined in Section 2.2, the interactions between neighboring defect images should be minimized to yield accurate formation energies. For charged defects in semiconductors and insulators, Coulombic interaction with neighboring images exists, which decays as 1/L, where L is the supercell periodic length. The charge interactions are the dominant effect that need to be taken into account when correcting the formation energy of defects in non-metals. Elastically-mediated interactions which are due to the strain fields induced when the positions of atoms near the defects also exist, but decay more rapidly in real space, and are often minimal [6]. In cases where these interactions are important, methods have been developed to account for them (see [6] and references therein), which will not be addressed in the following discussion.

To account for the charge correction one approach has been to create successively larger defect supercells. Scaling laws for the electrostatic interactions with respect to system size are then used in order to extrapolate E^{f} to the dilute limit [50,51]. An alternative approach is based on an *a posteriori* analysis of the electrostatic potential for a single supercell calculation [31,52]. An important requirement for the alternative approach is that the charge be sufficiently localized within the vicinity of the defect. If so, a moderately sized defect supercell typically suffices, hence, offering a computationally more efficient route to calculating reliable defect formation energies. The latter methodology, referred to as "correction methods" in the following, is employed in PyCDT.

Correction methods address two issues:

1. the electrostatic energy from the interaction between the charged defect and its images, and
2. a potential alignment term that corrects for a fictitious jellium background required to maintain overall charge neutrality in the system.

Many different methods have been proposed to correct for these two terms, as summarized in several comprehensive reviews (see, e.g., [6,16,17,25] and references therein).

The theoretical starting point for the correction methods considers a periodic array of point charges (cf., Fig. 2e) with an associated Madelung energy, E_{M}:

$$E_{\mathrm{M}} = \frac{qV_{\mathrm{M}}}{2} = \frac{q^2\alpha}{2\epsilon L} \tag{4}$$

where V_{M} is the Madelung potential, ϵ is the dielectric constant, and α is the Madelung constant which solely depends on the geometry of the periodic array. Makov and Payne [53] introduced one of the earliest charge correction methodologies by deriving the next leading order term to the interaction potential. This results in a term that scales as L^{-3}, and, therefore, most supercell scaling approaches fit uncorrected formation energies to the form of $aL^{-1} + bL^{-3}$. Komsa et al. [54] used this supercell scaling method for evaluating the performance of different correction methods that are based on single supercell calculations. They concluded that the correction by Freysoldt et al. [31] produces the most reliable charge corrections for defects with charges that are well localized within the supercell. From all considered defects, the authors calculated a mean absolute error of 0.09 eV in the formation energy between the estimate from the 64-atom supercell with charge corrections and the estimate from the supercell-scaling method (i.e., using extrapolation toward the dilute-defect limit, but without applying any charge correction).

PyCDT includes a Python implementation of the correction scheme derived by Freysoldt et al. [31] and implemented in the open-source DFT software S/PHI/nX [55]. The approach is based on a separation of the long-range and short-range interactions between charged defects, using information directly outputted from a DFT calculation. Originally, an isotropic dielectric constant was assumed. Recently, Kumagai and Oba [32] extended the approach to anisotropic systems, by considering the full dielectric tensor. The analytic expression of the Madelung potential under isotropic conditions facilitates the use of a Gaussian distribution for the defect charge, whereas the analytic expression of Madelung potential for anisotropic systems is limited to point charges.

The authors of the two correction methods suggest different approaches to calculating the potential alignment correction. The isotropic correction by Freysoldt et al. uses a planar average of the electrostatic short range potential while the anisotropic correction by Kumagai and Oba takes averages of this same potential at each atomic site outside a given radius from the defect. Both approaches are available in the PyCDT code, with the isotropic correction by Freysoldt et al. being the default. The planar averaging method can become problematic when large relaxation occurs, as the atomic sites contribute heavily to the change in electrostatic potential. The atomic site averaging method can become problematic if a small cell size results in a small number of atoms being sampled, causing statistical sampling errors. While, in principle, these alignment corrections should be equivalent, tests that we conducted revealed non-negligible discrepancies. However, the potential-alignment term often tends to be small ($\sim$0.1 eV), and, therefore, to not change overall trends in defect formation energies.

The correction methods used for point defects in semiconductors and insulators have been an intensely debated topic in the past decade [6]. One issue upon which there is common agreement is that large defect–defect interactions change the energetics of the system so that the computed defect formation energies are no longer relevant for physical quantities like defect concentrations or thermodynamic transition levels. These unwanted defect–defect interactions frequently lead to delocalization of the defect charges, the instance of which has to be ascertained manually. Several methods that address delocalization can be found in the literature [6,17,50].

If we assume that the defect charge can indeed be localized within the level of DFT used, then best practice demands to balance computational expediency (supercell size) with sufficient localization of the charge around a defect as indicated by the outputs of the charge correction method chosen. In the original derivation of the isotropic correction by Freysoldt et al., the middle "plateau" region of the electrostatic potential yields information about the separation of long range and short range effects. A flat plateau indicates that the Coulomb potential has been removed from the total potential generated by DFT, and short range effects have not delocalized throughout the entire supercell. As a result, Freysoldt et al. [31] suggested that the flatness of the resultant "plateau" yields a qualitative metric for the success of the calculation. When running the isotropic correction by Freysoldt et al. in PyCDT, the planar averaged electrostatic potential is analyzed for variations larger than 0.2 eV—a number that stems from experience, and can be altered in the code if the user prefers to. If this criterion is not met, the code raises a warning. In such a case, the user should consider the possibility of delocalization.

For users who desire additional corrections related to improving corrections with semilocal functional approaches, the development branch of PyCDT also includes the ability to include band edge level alignment with respect to the average electrostatic potential, shallow level corrections based on the values for band edge alignment, and Moss–Burstein band filling corrections [17]. Parts of these corrections require some subjective judgment calls to be made by the user, so they are not included in the automation procedure by default.

2.5. Charge ranges

Charge ranges, $[q_{\min}, q_{\max}]$, have to be estimated beforehand for a given defect X, and for this purpose ionic models typically form the basis of such predictions [56]. Known oxidation states of the element(s) involved in X can then be used to define the charge states to be considered. However, common ionic models do not always predict the most stable defect. Tahini et al. have, in this context, shown that combining gallium or aluminum with group-V elements can yield negatively charged anion vacancies, whereas an ionic model predicts a $+3$ charge state [39]. In PyCDT, we implemented different procedures to determine the range of defect charges for semiconductors and insulators. Users can choose between either of these two options and a custom range of defect charges for each defect, as described in Section 3.1.

To address the issue of uncommon charge states found in semiconductors, we developed a data-driven approach that combines elemental oxidation states with results from literature for determining the optimal charge assignment process. We compiled a list of stable charge states (Table A.2) from previous studies for various defects in zinc blende and diamond-like semiconductor structures [24,39,56–62]. Procedures adhering more or less strictly to ionic models resulted in too few charge states when compared with the literature. The most effective approach that we found employs a bond-valence estimation scheme [43,63] to obtain formal charges of elements in the bulk structure, as well as minima and maxima of common oxidation states of bulk and defect elements. The formal charges and common oxidation-state ranges are subsequently used in a defect type-dependent assignment procedure:

1. *Vacancies*: Use the formal charge of the species originally located on the vacant site, *oxi*, to define the charge range: $[-oxi, +oxi]$. For GaAs, this procedure results in defect charges ranging from -3 to 3 for both V_{Ga} and V_{As}.
2. *Anti-sites*: Use the minimum and the maximum from combining all oxidation states of *all* elements in the bulk structure, $oxis_{\text{bulk}}$, to define the relevant charge range: $[\min(oxis_{\text{bulk}}), \max(oxis_{\text{bulk}})]$. Data mining determined that the upper range limit can, in fact, be decreased by 2: $[\min(oxis_{\text{bulk}}), \max(oxis_{\text{bulk}}) - 2]$. With this procedure, the antisites in GaAs are assigned charge values from -3 to +3.
3. *Substitutions*: Determine the oxidation states of the foreign (or, extrinsic) species, $oxis_{\text{ex}}$. Then subtract the formal charge of the site species to be replaced from this list. Use the minimum and maximum of this set to produce $[\min(oxis_{\text{sub}}), \max(oxis_{\text{sub}})]$. Data mining determined that, if the new range has more than 3 charge states and has an upper bound larger than 2, one can cap the range by 3 to prevent excessively high charge states. For example, when GaAs is doped with Si, Si_{Ga} generates charges in the range of [-7,1], and Si_{As} generates charges in the range of [-1,4].
4. *Interstitials*: Use the minimum and maximum of all oxidation states of the interstitial species: $[\min(oxis_{\text{int}}), \max(oxis_{\text{int}})]$. If 0 is not included, data mining suggests that we extend the range to 0 accordingly. For As interstitials in GaAs, the resulting defect charges are in the range [-3, 5].

The algorithm successfully includes all charge states from our benchmark list in Table A.2 by yielding, on an average, 6.4 states per defect. The average number of excess charge states produced per defect, in comparison to the literature, is 1.1 and 1.9 at the lower (more negative) and at the upper (more positive) charge bound, respectively. This is desirable because including more charge states on either side ensures no extra states become stabilized when varying the Fermi level within the band gap. Hence, the effective relative excess in charge states is 20% and, thus, acceptable.

For insulators, the number of defect charge states is typically less than for the above discussed semiconductors. For example, the charge states in MgO range from -2 to 0 and 0 to $+2$ for cation and anion vacancies, respectively [64]. Any other charge state is not considered because of the high ionization energy required to form Mg^{3+} and the high electron affinity of O^{2-} to form O^{3-}. Hence, the oxidation states of cations and anions are limited to $[0, y]$ and $[-x, 0]$ for a binary A_xB_y insulator, where A is a cation and B is an anion.

2.6. Interstitials

PyCDT uses an effective and easily extendable approach for interstitial site finding (Interstitial Finding Tool: InFiT) that has been recently introduced by Zimmermann et al. [33]. The procedure systematically searches for tentative interstitial sites by employing coordination pattern-recognition capabilities [65,66] implemented in pymatgen [43]. In Algorithm 1, we provide a simplified pseudo-code representation of the approach. The detected interstitial sites exhibit coordination patterns that resemble basic structural motifs (e.g., tetrahedral and octahedral environments). Such interstitial sites are particularly important because several [67–76] β^- emission channeling measurements [77,78] have identified them as the most prevalent types of isolated defects after substitutions for impurities implanted into similar materials as we consider here (zinc blende/wurtzite-like and diamond-like structures). Bond-center interstitials in a so-called split-vacancy configuration are also observed frequently. However, these are defect complexes – not isolated defects – and, thus, beyond the scope of the present PyCDT implementation. The interstitial search approach should also be suitable for intercalation and ion diffusion applications because related design rules typically rely on detection of tetrahedral, octahedral, bcc-, and fcc-like environments [79,80].

2.7. Default DFT calculation details

PyCDT includes mechanisms to input user-defined settings for all DFT calculations. If no user settings are specified, then the following initial settings are specified for the calculation. The ions in the defective supercells are geometrically relaxed at constant volume until the force on each ion is less than 0.01 eV/Å. At each geometric step, the energy of the supercell is converged to 10^{-6} eV. By default, spin polarization is turned on, and crystal symmetry is ignored to account for any symmetry breaking relaxations such as Jahn–Teller

Algorithm 1 Interstitial Site Searching

GetNeighbors(*struct*, *inter_trial_site*, *dmin*, *delta*) ▷ Get a list of neighbors of site *inter_trial_site* in *struct* that are within a sphere of (1 + *delta*) *dmin*.
GetCoordinationDescriptor(*point*, *neighs*, *coord_type*) ▷ Calculate the value of the descriptor for the target coordination environment *coord_type* of an atom at *point* with neighbors *neighs*.

procedure GetCoordinationPatternInterstitials(*struct*, *inter_elem*, *all_coord_types*= [tet, oct], *thresh* = [0.3, 0.5], *dl* = 0.2, *dstart* = 0.1, *ddelta* = 0.1, *dend* = 0.7)
    Let *inter_sites* be a new empty list
    Let *coord_descr* be a new empty list
    for *point* on a regularly mashed grid in *struct* with resolution *dl* **do**
        Let *dmin* be the distance from *point* to the closest crystal atom
        if *dmin* > 1 Å **then**
            Let *inter_trial_site* be a new *Site* object of type *inter_elem* located at *point*
            Let *struct_plus_inter* be a new *Structure* object ← *struct* appended by *inter_trial_site*
            for *delta* starting from *dstart* in *ddelta* steps to *dend* **do**
                Let *neighs* be a new list ← *GetNeighbors*(*struct_plus_inter*, *inter_trial_site*, *dmin*, *delta*)
                for *coord_type* having index *icoord* in *all_coord_types* **do**
                    Let *this_coord_descr* be a float number ← *GetCoordinationDescriptor*(*point*, *neighs*, *coord_type*)
                    if *this_coord_descr* > *thresh*[*icoord*] **then**
                        Add entry to *inter_sites* list ← *inter_trial_site*
                        Add entry to *coord_descr* list ← *this_coord_descr*
                        break
    Let *labels* be a list of *coord_type*-specific cluster labels for each entry in *inter_sites* which are found with a distance threshold of 1.01 *dl*.
    Let *include* be a new empty list.
    for *unique_label* in *labels* **do**
        Find site with index *imax* in *inter_sites* that has highest *coord_descr* among site with this *unique_label*.
        Add entry to *include* ← *imax*
    final_inter_sites ← Prune the *inter_sites* in *include* further to include symmetrically distinct sites only
    return *final_inter_sites*

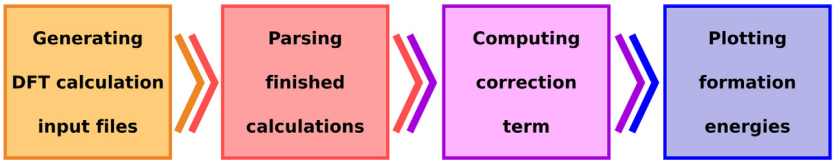

Fig. 3. Steps in the computation of charged-defect formation energies with PyCDT.

distortions. Electronic states are populated using a Gaussian smearing method [81] with a width of 0.05 eV. While no geometric relaxation is performed for the non-defective bulk supercell, a calculation that is necessary for the charge correction, the electronic degrees of freedom are optimized with the same settings applied to the defect-supercell calculations. A 2 ×2 ×2 Monkhorst–Pack *k*-point mesh is used for the defect calculations. For any other parameters, we adopt the standard MP settings [43]. Finally, we emphasize that PyCDT also includes mechanisms to input user-defined settings for DFT calculations which can be used for extending calculations beyond the exchange and correlation approximation of GGA. The practical implementation of user-defined settings is described at length in the Supplementary Information (SI).

3. PyCDT usage and examples

PyCDT was developed in a way that reflects different analysis stages (Fig. 3): setup of DFT calculations, parsing of finished jobs, computation of a correction term, and plotting of formation energies. This allows reuse and integration of parts of the code in other packages. Our package has a dependency on pymatgen [43], matplotlib [82] and numpy [83], and it was developed and tested for Linux and Mac OS X. However, we also expect it to work on Windows (with cygwin).

It is possible to designate three levels of involvement which an end user may desire from the PyCDT code. Ranked in increasing order of involvement required these are: (Level 1) standard command-line user interface calls, (Level 2) isolated python scripts mimicking command-line calls but for more personalized user involvement, and (Level 3) user customized workflows for performing personalized high-throughput defect calculations. In the subsections that follow, we illustrate these three levels of user involvement in the example of zinc-blende GaAs with VASP [34,35] as the backend DFT code. In order to be involved at (Level 2) and (Level 3) it is necessary to first understand the manner in which the PyCDT command line tool makes use of the PyCDT core functionality. Accordingly, in Sections 3.1–3.4 we describe how the command line approach to PyCDT (Level 1) can be used for each step of the general charged-defect calculation workflow outlined in Fig. 3.

Table 1
Default defects set up for GaAs with PyCDT.

Defect type	Folder name
Vac_{Ga}	GaAs/vac_1_Ga
Vac_{As}	GaAs/vac_2_As
Ga_{As}	GaAs/as_1_Ga_on_As
As_{Ga}	GaAs/as_2_As_on_Ga

For users interested in further customized applications of PyCDT (Levels 2 and 3), the Supplementary Information (SI) includes a description of the code's structure and the manner in which the command line code makes use of the core functionality of PyCDT. This helps to reiterate that the core classes within PyCDT are entirely general and can be used for any desired applications that involves setting up, computing charge corrections for, and/or parsing defect calculations. As a brief example of user customization beyond the command line tool, the SI also demonstrates customizable approaches to (a) initializing and parsing a personalized phase diagram for chemical potentials and (b) screening over non-intuitive charge states of defects in a computationally efficient manner. As a demonstration of the brute power of these tools, Section 3.5 displays a test set of high-throughput computation of defects (Level 3) by performing intrinsic defect calculations on 15 additional zinc blende systems beyond GaAs and 5 oxide systems.

3.1. Setup of DFT defect calculations

The starting point for setting up charged defect calculations is the crystal structure. The user can provide the bulk structure in one of two ways:

(1) by the name of a structure file of conventional format (e.g., cif, cssr), or code specific formats such as POSCAR that are recognized by pymatgen, or
(2) via a Materials Project identifier (MPID).

Crystal structures from MP [29] are obtained through the Materials Application Programming Interface (MAPI) [43]. In the MP database, each structure is assigned a unique identifier. These MPIDs have the format *mp-XXX* in which *mp-* is prefixed to a positive integer *XXX*. In the following, we use GaAs (*mp-2534*), which has the zinc-blende structure, as an example for performing all different stages of charged defect-property calculations with PyCDT.

We first generate the defect supercells and the bulk supercell using pymatgen's defect structure generator and the defect structure classes in the core of PyCDT. The two steps for generating the input files are combined into a single command:

```
▷ pycdt generate_input  (  --structure_file  ⟨structure file⟩  |  --mpid ⟨mpid⟩  )
                        [  --mapi_key       ⟨mapi_key⟩                        ]
                        [  --nmax           ⟨max_no_atoms_in_supercell⟩       ]
```

With *–nmax*, the user defines the maximal number of atoms in the defect supercell. If the parameter is not given, a default value of 128 is used, which was shown to result in well converged defect formation energies after finite size corrections were included in systems with dielectric constants greater than 5.0 [24,54]. The mapi_key is required if querying the MP database, and is found on the Dashboard after logging into the MP website.

The input file-generation command creates a folder, representing the reduced chemical formula of the crystal structure (e.g., *GaAs*). It contains several subfolders, whose names are indicative of the calculations to be performed:

- *bulk*: calculation of pristine crystal structure,
- *dielectric*: calculation of macroscopic static dielectric tensor (ion clamped high frequency, ϵ_∞, plus the ionic contribution, ϵ_{ion}) from DFT perturbation theory, (used by the charged defect correction)
- *deftype_n_info*: calculation of the *n*th symmetrically distinct defect of type *deftype* (vacancies: *vac*; antisites: *as*; substitutions: *sub*; interstitials: *inter*) with properties *info*.

By default, the above command generates vacancy and antisite defects only. Hence, there are four defect folders for GaAs, which have two sublattices, corresponding to one antisite and one vacancy defect on each sublattice. Table 1 summarizes the default defect types and resulting folder names for GaAs.

Substitutional and interstitial defects have to be invoked explicitly with the keyword *–sub host_species substitution_species*. Multiple substitutional defects can be generated by repeating the *–sub* keyword with the desired host species and the corresponding substituting species. The substitution folders are labeled in the same manner as the antisites, only changing *as* to *sub*.

The setup of coordination-pattern resembling interstitials is invoked by the *–include_interstitials* command-line option. PyCDT produces intrinsic interstitials as per default only. Extrinsic interstitials can be achieved by providing a list of elements as positional arguments (e.g., *–include_interstitials Mn*). To obtain both intrinsic and extrinsic interstitials the intrinsic elements have to be explicitly mentioned (e.g., *–include_interstitials Ga As Mn*). As for the other defect types, PyCDT enumerates the interstitial calculation folders according to symmetrically distinct sites found. The *info* part of the interstitial folder names indicates (1) the type of the atom located on an interstitial site having a certain (2) coordination pattern and (3) chemical environment. For example, *inter_1_As_oct_Ga6* shows that we are dealing with an As interstitial that is octahedrally coordinated by six Ga atoms.

For each defect type in semiconductors, multiple charge states are considered according to the algorithm outlined in Section 2.5. For insulators a conservative charge assignment is used as described in Section 2.5. By default, the input structure is considered a

semiconductor. To specify the input structure is of insulator type, the option *–type insulator* can be used. The user can also modify the charge assignments for each defect by specifying either of the two flags, *–oxi_state* or *–oxi_range*. Alternatively, the option *–type manual* allows for the user to specify every charge state that is desired. The DFT input files associated with each of these charge states, q, are deposited into subfolders named *charge_q*. For example, seven charge states are generated for the gallium vacancy in GaAs.

Apart from the structure file, PyCDT automatically generates all other input files according to the settings used for the MP. The input settings can also be easily modified by supplying the parameters in a yaml or json file and using the keyword *–input_settings_file* ⟨*settings_file.yaml*⟩. For instance, the file *user_settings.yaml* that we provide in the *examples* folder changes the default functional from PBE to PBEsol and increases the energy cutoff to 620 eV. The structure of this file is described in further detail in the supplementary information. When such changes are made, the user has to keep in mind that the atomic chemical potentials obtained from the MP database in the final parsing step have to be replaced by user computed ones with the corresponding changes included during chemical potential calculations — a process which can be sped up substantially with PyCDT's phase diagram set up and parsing feature. Any DFT settings that are specific to either the bulk, or the dielectric, or the defect calculations can be thus realized, too, as demonstrated in the example file. The input settings whether chosen by default or by the user are expected to be tested for appropriate convergence criteria. In addition to the input files for DFT calculations, PyCDT saves a *transformation.json* in each calculation folder, except for *dielectric*, to facilitate post-processing.

3.2. Parsing finished calculations

The DFT calculations can be run either manually, with bash scripts, or with high throughput frameworks [84]. Once all the calculations have successfully completed, the generated output files are parsed to obtain all the data needed to compute defect formation energies, E^f. This part of PyCDT is executed by reading the *transformation.json* file that was output from the previous file generation step. To initiate parsing from the command line interface, the user issues:

```
▷pycdt parse_output   [  --directory   ⟨directory⟩   ]
                      [  --mpid        ⟨mpid⟩        ]
                      [  --mapi_key    ⟨mapi_key⟩    ]
```

Here, directory is the root folder of the calculations. If executed within the folder of the calculations the option can be omitted. Once the parsing is completed, PyCDT stores all the data required for next steps in a file called "defect_data.json". If any of the calculations were not successfully converged according to the code output files, PyCDT raises a warning, but continues parsing the rest of the calculations. The output file "defect_data.json" contains the parsed energies of the defect and bulk supercells as well as other information required in the next steps to calculate finite size charge corrections and defect formation energies. Some of the additional data such as the dielectric constant is obtained by parsing the output from the dielectric calculation. The band gap and atomic chemical potentials generated in this step are obtained from computed entries in the MP database. Note that these band gaps are only accurate at the level of GGA, which often underestimates the gap by about 50%. The output file "defect_data.json" is highly readable and users can edit the file to supplant parameters either parsed from the DFT calculations or obtained from the MP database. If the user prefers to have formation energies and transition levels closer to a higher level of theory (which can provide better band gaps than the GGA approximation), the band edge alignment procedures suggested in Section 2.4 can be used. Furthermore note that some structures in the MP database do not have fully computed band structures, which results in poorly converged band-gap characteristics.

3.3. Computation of correction term

Correcting the errors due to long-range Coulomb interactions in finite-size supercells results in improved defect formation energies. A feature of PyCDT is the possibility to compute such corrections with minimal work from the end user. The following command-line call computes individual correction values for all charged defects found in the present directory:

```
▷pycdt compute_corrections   [  --correction_method   ⟨correction method⟩   ]
                             [  --epsilon             ⟨epsilon tensor⟩      ]
                             [  --input_file_name     ⟨defect data file⟩    ]
                             [  --plot_results        ]
```

Here, correction method keywords can be either *freysoldt* for the correction due to Freysoldt et al. [31,52] or *kumagai* for the approach extended to anisotropic systems by Kumagai and Oba [32]. As shown in Section 4.1, these codes have been rigorously tested against the results from the codes of the original authors. The command line interface requires the defect data file generated in the previous step, *defect_data.json*, for computing the corrections. This flag can be omitted if the file name is unchanged. The calculated corrections for each defect charge state are stored in a file called *corrections.json*. By rerunning the command with different correction keywords, one can quickly obtain the corrections computed with different frameworks. Shown in Fig. 4 are the resulting potential alignment plots for each correction type on the Ga_{As}^{-2} defect in GaAs.

The sampling regions for obtaining the potential alignment correction defaults to 1 Å in the middle region of the planar average plots recommended by Freysoldt et al. [52], and to the region outside of the Wigner–Seitz radius for atomic site averaging method, following the approach described in Ref. [32]. The width of these default sampling regions can be changed by modifying the instantiation of the relevant PyCDT correction classes.

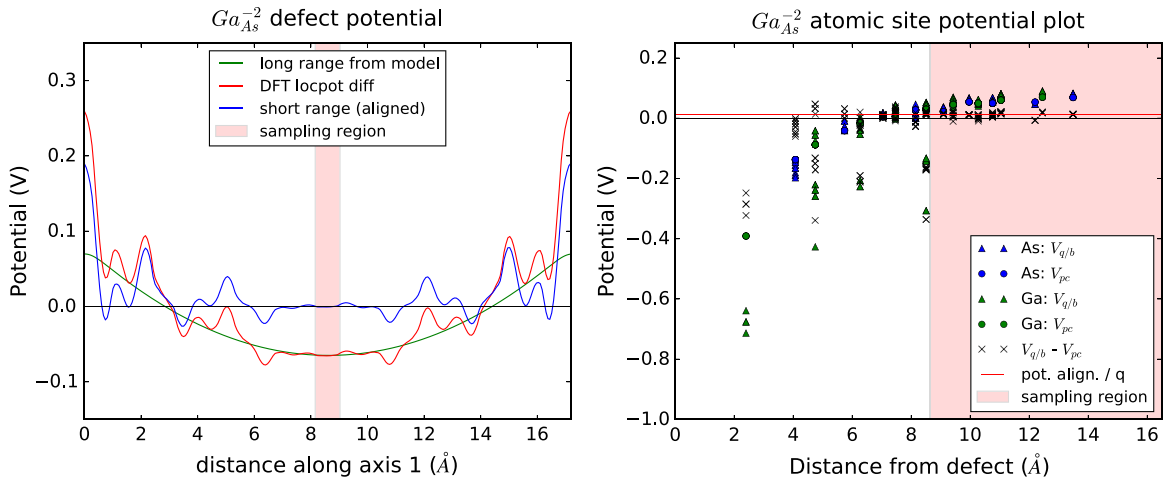

Fig. 4. Two different methods for computing the potential alignment correction on GaAs calculation. At left is isotropic correction, developed by Freysoldt et al., using the planar average method [31,52], and at right is anisotropic correction by Kumagai and Oba, using the atomic site averaging method [32].

3.4. Formation energy plots and transition levels

Once the defect energetics and the correction values are obtained and specified in the *defect_data.json* and *corrections.json* files, the transition levels and formation energies of each defect across the band gap can be determined by:

```
▹pycdt compute_formation_energies  [  --input_file_name  ⟨defectsdata json file⟩  ]
                                   [  --corrections_file_name  ⟨corrections json file⟩  ]
                                   [  --bandgap  ⟨band gap⟩  ]
                                   [  --plot_results  ]
```

By default, PyCDT uses the band gap stored in the MP database, which is computed with GGA-PBE and, hence, under-predicted when compared to the experimental band gap. As mentioned several times in this work, this approximation can be improved upon with the optional corrections included, such as the band edge realignment feature of PyCDT. A simpler approximation to improving the defect formation energies is to plot the defect formation energies across the experimental gap the user can specify the experimental band gap. This can be done from the command line with the keyword *–bandgap ⟨band gap⟩*. This has the effect of extending the gap by shifting the conduction band minimum, but keeping the position of the valence band maximum and the defect levels fixed, and it is often called the "extended gap" scheme. We note that the extended gap option is strictly for plotting purposes and, thus, does not alter the defect formation energies nor the transition levels. If the names of the files obtained in the previous two steps are not changed, the corresponding options can be omitted. If the *corrections.json* file is not found and an alternative corrections file is not specified, PyCDT assumes that electrostatic corrections are not desired and computes the defect formation energies without any corrections. As detailed in Section 2.3, the defect formation energies are influenced by the chemical environment, and their range is determined by the phase stability of the various compounds formed by the constituent host and defect elements. Hence, two plots are generated corresponding to the chemical availability of the constituent host elements in the compound. The files "Ga_rich_formation_energy.eps" and "As_rich_formation_energy.eps" are shown in Fig. 5. These results are verified to be consistent with literature results in Section 4.2.

3.5. Example usage for high-throughput point defect calculations

To demonstrate the capabilities of PyCDT in a high-throughput environment, we have computed the defect formation energetics of 15 additional zinc blende systems and 5 oxide systems. The results for these calculations are shown in Fig. 6.

The calculations were performed using the core functionality of PyCDT which has been outlined in Sections 3.1–3.4 and described at length in the SI. For running the DFT calculations, the Fireworks software [84] was used to set up atomic workflow tasks from a structure which had random local symmetry breaking performed (a task which is trivially performed with the Pymatgen software [43]). To expedite the handling of many common DFT failures encountered (e.g. problems with electronic self-consistency etc.), the Custodian software [43] was implemented to correct for standard job failures. The core DefectsAnalyzer class, described further in the SI, was then implemented in a manner similar to the charge screening procedure defined in the SI, to ensure additional charge states were not stabilized within the gap. Parsing and accumulating all relevant quantities for formation energetic analysis into a database was simple to implement with the use of the DefectsAnalyzer class.

Out of a total 376 calculations that were run for this analysis, 56 jobs failed to finish on the first run — either due to walltime errors or additional electronic convergence issues. Fireworks database management coupled with PyCDT core analysis of the additional charge states allowed for quick identification of jobs with errors that required follow up calculations. This approach allowed for fast re-submissions which took on the order of minutes, rather than multiple hours/full days of analysis of failed defect jobs. This demonstrates the massive

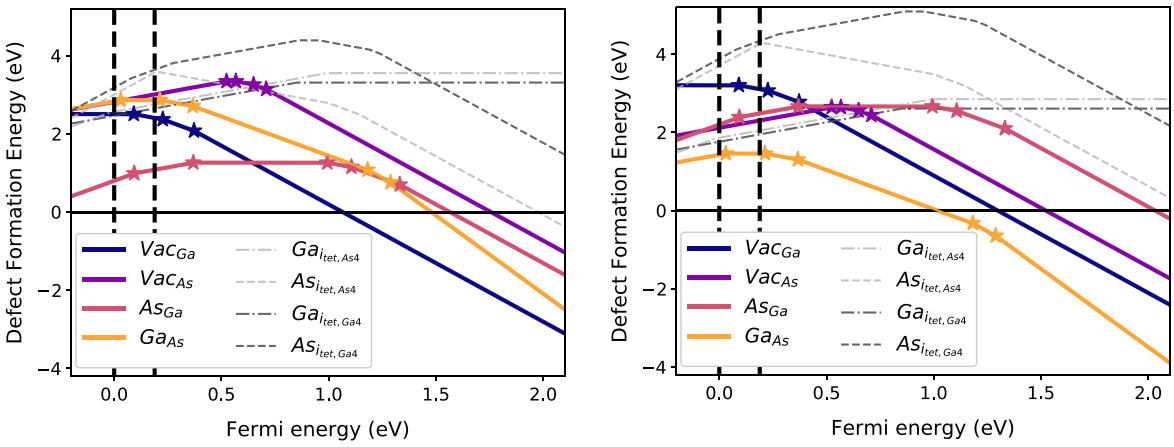

Fig. 5. Defect formation-energy plots from PyCDT for GaAs. The left panel is obtained in the As-rich growth regime, whereas the right panel is obtained under Ga-rich conditions. Interstitial defects are colored according to their site within the lattice, with dashed lines given to As interstitials and dash–dot lines given to Ga interstitials. The thick vertical dashed black line indicates the GGA-PBE band gap of GaAs [29].

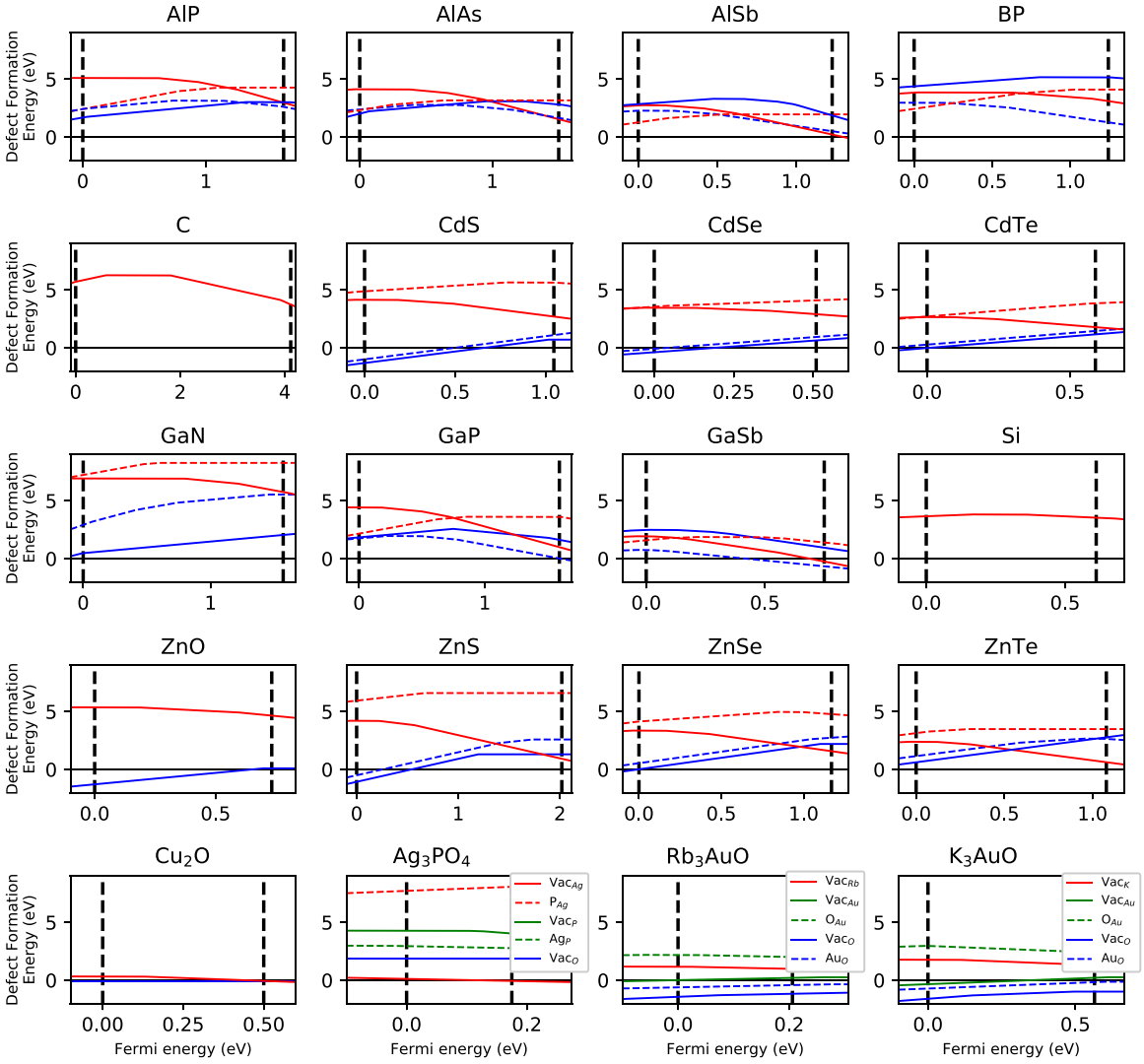

Fig. 6. Defect formation energy plots for 15 zinc blende systems and 5 oxide systems. The spacegroups and materials project identification numbers for each of these structures are included in a Table in the SI. Solid red lines indicate cation vacancies, while dashed red lines are cation antisites (Anion_on_Cation site defects). Solid blue lines are anion vacancies, while dashed blues lines are anion antisites (Cation_on_Anion site defects). For the four ternary oxide systems, green lines are also used to indicate defects on the secondary cation site. Legends are included as insets for ternary systems. The cation-rich growth condition dictates all of the atomic chemical potentials as described in Section 2.3. Due to zero gap predictions at the GGA level of DFT, the experimental gap was used for GaSb.

scalability of calculations which PyCDT can provide for a researcher who desires to intelligently implement defect calculations in a high-throughput environment.

4. Validation and verification

4.1. Validation

To validate PyCDT's implementation of each correction method, we ran the correction methods on the defects generated and computed for the 15 zinc blende structures (binary and elemental), comparing the charge corrections generated by PyCDT with the open-source code developed by Freysoldt et al., SXDEFECTALIGN, as well as with the command line code developed by Kumagai and Oba. Over a total of 224 (non-zero charge) defects calculated, the root mean square difference between PyCDT and the original author's codes were 16.5 meV and 19.4 meV for the correction by Freysoldt et al. and the correction by Kumagai and Oba, respectively. The differences in the corrections are almost entirely attributed to the resolution in the calculation of the potential correction.

4.2. Verification

To verify the results predicted from the example GaAs test set, we compared the defect formation energies in GaAs obtained from PyCDT with the data reported in literature. We note that all of the transition levels for the Vac_{Ga} and As_{Ga} defects are within the range of transition levels that we found for semi-local functional approximations in the literature. An exception is the As_{Ga}(-1/-2) transition, which appears very far into the GGA-PBE conduction band. This outlier is off by 0.273 eV relative to the reported results by Chroneos et al. [58]. For all transition levels that we predicted, we find a root mean square deviation of 0.218 eV from the window of values found in the literature [39,58,85–90]. This is a modest variation that reflects the difficulty in predicting defect levels consistently—even within the same level of theory.

In a similar manner, we use Refs. [39] and [58] to verify 63 additional transition levels in 5 systems within the high-throughput test set of Section 3.5 (AlP, AlAs, AlSb, GaP, GaSb). We find a root mean square absolute error of 0.29 eV from the transition levels reported in these two references, with a maximum error of 0.89 eV for the Aluminum vacancy (-2/-3) transition level in AlP. Again, this deviation is consistent with the variation of predicted transition levels from DFT and suggests a need for standardizing approaches to computing defect formation energetics.

5. Conclusion

We have introduced PyCDT, a Python toolkit which facilitates the setup and post-processing of point defect calculations of semiconductor and insulator materials with widely available DFT suites. This open source code allows for coupling automated defect calculations to the massive amount of data generated by the Materials Project database. Apart from the underlying theory, approaches, and algorithms, this paper presents a detailed guide for how to use PyCDT at every step of the computation of charged defect properties employing the well-studied example of GaAs. While the example results were obtained from VASP [34,35] calculations, we carefully developed PyCDT in an abstracted form that adopts the advantageous code agnosticism of pymatgen [43]. This makes the provided tools attractive to any user interested in running defect calculations, regardless of DFT code preference. However, we emphasize that, despite its convenience and our effort to construct sensible defaults, the computation of defect properties with PyCDT still requires user expertise for choosing appropriate settings in certain circumstances and for interpreting the results meaningfully in general (i.e., we discourage purely "black box" usage).

The PyCDT version presented here is 1.0.0. Future updates will, amongst others, include adaptations related to improved charge delocalization analysis, further defect corrections for issues like artificial band dispersion, as well as the possibility of generating defect complexes. On the application side, PyCDT could also be extended to compute configuration coordinate diagrams, so as to evaluate optical and luminescence transitions associated with the point defects in materials targeting optical applications.

We hope that our openly available tools will help to standardize computational research in the realm of charged defects. In particular, we hope that reproducibility issues commonly encountered in DFT calculations [91] can be more effectively identified and tackled.

Acknowledgments

This work was primarily funded by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, Materials Sciences and Engineering Division under Contract No. DE-AC02-05-CH11231: Materials Project program KC23MP. B. Medasani was supported by the U.S. DOE, Office of BES, Division of Materials Sciences and Engineering, FWP 56909. Pacific Northwest National Laboratory is a multiprogram national laboratory operated for DOE by Battelle. This work used resources of the National Energy Research Scientific Computing Center, supported by the BES of the U.S. DOE under Contract No. DE-AC02-05CH11231. The authors would like to thank Yu Kumagai for insightful discussions about charge corrections.

Appendix A. Charge states from literature

See Table A.2.

Table A.2
Charge states from literature; brackets refer to hybrid functional rather than (semi-)local functional results.

Structure	Defect	Charge states from	to	Ref.
C	V_C	+2	−2	[57]
	N_C	+1	−1	
AlP	V_{Al}	0	−3	[39]
	V_P	+1	−2	
	Al_P	+1	−2	[58]
	P_{Al}	+2	−2	
AlAs	V_{Al}	0	−3	[39]
	V_{As}	+1	−2	
	Al_{As}	+1	−2	[58]
	As_{Al}	+1	−1	
AlSb	V_{Al}	0	−3	[39]
	V_{Sb}	+1	−3	
	Al_{Sb}	0	−2	[58]
	Sb_{Al}	+1	−1	
Si	V_{Si}	+2	−2	[59]
ZnS	V_S	+2	0	[60]
	V_{Zn}	0 (+1)	−2	[60] ([56])
	S_{Zn}	0	−2	[60]
	Zn_S	+2	0	
	Zn_i	+2	0	
	S_i	0	−2	
ZnSe	V_{Se}	+2	−2	[61]
	V_{Zn}	+2	−2	
	Cl_{Se}	+2	−1	
	F_{Se}	+1	−2	
	F_{Zn}	−2	−2	
ZnTe	V_{Zn}	0 (+1)	−2	[56]
GaN	V_{Ga}	0 (+1)	−3	[62] ([56])
	V_N	+1	+1	[62]
	Ga_N	+1	−2	
	N_{Ga}	+2	−1	
	N_i	+3	−1	
	Ga_i	+3	+1	
	C_N	0 (+1)	−1	[56]
GaP	V_{Ga}	0	−3	[39]
	V_P	+1	−3	
	Ga_P	0	−2	[58]
	P_{Ga}	+2	−2	
GaAs	V_{Ga}	−1	−3	[39]
	V_{As}	+1	−3	
	Ga_{As}	0	−3	[58]
	As_{Ga}	+1	−2	
GaSb	V_{Ga}	0	−3	[39]
	V_{Sb}	0	−3	
	Ga_{Sb}	0	−2	[58]
	Sb_{Ga}	+1	−1	
CdS	V_S	+2	0	[24]
	V_{Cd}	0	−2	
	Cd_S	+2	+2	
	S_{Cd}	+4	−2	
	Cd_i	+2	+2	
	S_i	+4	−2	
	Mn_{Cd}	+1	0	
	Fe_{Cd}	+2	0	
	Co_{Cd}	+1	0	
	Ni_{Cd}	+1	0	
	Mn_i	+3	+2	
	Fe_i	+3	+2	
	Co_i	+3	+2	
	Ni_i	+2	+1	
InP	V_{In}	0	−3	[39]
	V_P	+1	−1	
	In_P	+1	−2	[58]
	P_{In}	+2	−1	
InAs	V_{In}	0	−2	[39]
	V_{As}	+1	0	
	In_{As}	0	−1	[58]
	As_{In}	+1	0	
InSb	V_{In}	−1	−2	[39]
	V_{Sb}	+1	0	
	In_{Sb}	0	−1	[58]
	Sb_{In}	+1	0	

Appendix B. List of acronyms

See list of acronyms below.

Acronym	Full form
DFT	Electronic density functional theory
GGA	Generalized gradient approximation
LDA	Local-density approximation
MAPI	Materials Application Programming Interface
MP	The Materials Project
MPID	Materials Project identifier
PBC	Periodic boundary conditions
PBE	Perdew–Burke–Ernzerhof
PyCDT	Python Charged Defect Toolkit
Pymatgen	Python Materials Genomics
VASP	Vienna Ab initio Simulation Package

Appendix C. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cpc.2018.01.004.

References

[1] W.D. Callister Jr., Materials Science and Engineering: An Introduction, seventh ed., John Wiley & Sons, Inc., New York, NY, USA, 2007.
[2] H. Queisser, J. Spaeth, H. Overhof, Point Defects in Semiconductors and Insulators: Determination of Atomic and Electronic Structure from Paramagnetic Hyperfine Interactions, in: Springer Series in Materials Science, Springer Berlin Heidelberg, 2013.
[3] M. McCluskey, E. Haller, Dopants and Defects in Semiconductors, CRC Press, 2012.
[4] P. Rodnyi, Physical Processes in Inorganic Scintillators, in: Laser & Optical Science & Technology, Taylor & Francis, 1997.
[5] E.G. Seebauer, M.C. Kratzer, Mater. Sci. Eng., R 55 (3–6) (2006) 57–149. http://dx.doi.org/10.1016/j.mser.2006.01.002.
[6] C. Freysoldt, B. Grabowski, T. Hickel, J. Neugebauer, G. Kresse, A. Janotti, C.G. Van de Walle, Rev. Modern Phys. 86 (2014) 253–305. http://dx.doi.org/10.1103/RevModPhys.86.253.
[7] P. Dorenbos, Physica A 202 (2) (2005) 195–200. http://dx.doi.org/10.1002/pssa.200460106.
[8] A. Chaudhry, R. Boutchko, S. Chourou, G. Zhang, N. Grønbech-Jensen, A. Canning, Phys. Rev. B 89 (2014) 155105. http://dx.doi.org/10.1103/PhysRevB.89.155105.
[9] S. Lany, A. Zunger, Phys. Rev. Lett. 98 (2007) 045501. http://dx.doi.org/10.1103/PhysRevLett.98.045501.
[10] D.O. Scanlon, G.W. Watson, J. Phys. Chem. Lett. 1 (21) (2010) 3195–3199. http://dx.doi.org/10.1021/jz1011725.
[11] J.B. Varley, V. Lordi, A. Miglio, G. Hautier, Phys. Rev. B 90 (2014) 045205. http://dx.doi.org/10.1103/PhysRevB.90.045205.
[12] A. Zakutayev, C.M. Caskey, A.N. Fioretti, D.S. Ginley, J. Vidal, V. Stevanovic, E. Tea, S. Lany, J. Phys. Chem. Lett. 5 (7) (2014) 1117–1125. http://dx.doi.org/10.1021/jz5001787. PMID: 26274458.
[13] A. Walsh, D.O. Scanlon, S. Chen, X.G. Gong, S.-H. Wei, Angew. Chem. 127 (6) (2015) 1811–1814. http://dx.doi.org/10.1002/ange.201409740.
[14] H. Zhu, G. Hautier, U. Aydemir, Z.M. Gibbs, G. Li, S. Bajaj, J.-H. Pohls, D. Broberg, W. Chen, A. Jain, M.A. White, M. Asta, G.J. Snyder, K. Persson, G. Ceder, J. Mater. Chem. C 3 (2015) 10554–10565. http://dx.doi.org/10.1039/C5TC01440A.
[15] G.S. Pomrehn, A. Zevalkink, W.G. Zeier, A. vandeWalle, G.J. Snyder, Angew. Chem., Int. Ed. 53 (13) (2014) 3422–3426. http://dx.doi.org/10.1002/anie.201311125.
[16] C.G. Van de Walle, J. Neugebauer, J. Appl. Phys. 95 (8) (2004) 3851–3879. http://dx.doi.org/10.1063/1.1682673.
[17] S. Lany, A. Zunger, Modelling Simulation Mater. Sci. Eng. 17 (8) (2009) 084002. http://stacks.iop.org/0965-0393/17/i=8/a=084002.
[18] H. Peng, D.O. Scanlon, V. Stevanovic, J. Vidal, G.W. Watson, S. Lany, Phys. Rev. B 88 (11) (2013) 115201.
[19] M. Giantomassi, M. Stankovski, R. Shaltaf, M. Grüning, F. Bruneval, P. Rinke, G.-M. Rignanese, Physica B 248 (2) (2011) 275–289.
[20] J.E. Northrup, M.S. Hybertsen, S.G. Louie, Phys. Rev. Lett. 59 (1987) 819–822. http://dx.doi.org/10.1103/PhysRevLett.59.819.
[21] J.-L. Li, G.-M. Rignanese, E.K. Chang, X. Blase, S.G. Louie, Phys. Rev. B 66 (2002) 035102. http://dx.doi.org/10.1103/PhysRevB.66.035102.
[22] A.I. Liechtenstein, V.I. Anisimov, J. Zaanen, Phys. Rev. B 52 (1995) R5467–R5470. http://dx.doi.org/10.1103/PhysRevB.52.R5467.
[23] E. Finazzi, C. Di Valentin, G. Pacchioni, A. Selloni, J. Chem. Phys. 129 (15) (2008) 154113.
[24] J.-C. Wu, J. Zheng, P. Wu, R. Xu, J. Phys. Chem. C 115 (13) (2011) 5675–5682. http://dx.doi.org/10.1021/jp109567c.
[25] C.W.M. Castleton, A. Hglund, S. Mirbt, Modelling Simulation Mater. Sci. Eng. 17 (8) (2009) 084003. http://stacks.iop.org/0965-0393/17/i=8/a=084003.
[26] A. Goyal, P. Gorai, H. Peng, S. Lany, V. Stevanović, Comput. Mater. Sci. 130 (2017) 1–9.
[27] E. Péan, J. Vidal, S. Jobic, C. Latouche, Chem. Phys. Lett. (2017).
[28] K. Yim, J. Lee, D. Lee, M. Lee, E. Cho, H. Lee, H. Nahm, S. Han, Sci. Rep. 7 (2017) 40907.
[29] A. Jain, S.P. Ong, G. Hautier, W. Chen, W.D. Richards, S. Dacek, S. Cholia, D. Gunter, D. Skinner, G. Ceder, et al., APL Mater. 1 (1) (2013) 011002.
[30] J.P. Perdew, K. Burke, M. Ernzerhof, Phys. Rev. Lett. 77 (18) (1996) 3865.
[31] C. Freysoldt, J. Neugebauer, C.G. Van de Walle, Phys. Rev. Lett. 102 (2009) 016402. http://dx.doi.org/10.1103/PhysRevLett.102.016402.
[32] Y. Kumagai, F. Oba, Phys. Rev. B 89 (2014) 195205. http://dx.doi.org/10.1103/PhysRevB.89.195205.
[33] N.E.R. Zimmermann, M.K. Horton, A. Jain, M. Haranczyk, Front. Mater. 4 (2017) 34. http://dx.doi.org/10.3389/fmats.2017.00034.
[34] G. Kresse, J. Hafner, Phys. Rev. B 47 (1) (1993) 558.
[35] G. Kresse, J. Hafner, Phys. Rev. B 49 (20) (1994) 14251.
[36] W.D. Richards, L.J. Miara, Y. Wang, J.C. Kim, G. Ceder, Chem. Mater. 28 (1) (2015) 266–273.
[37] J. Sun, R.C. Remsing, Y. Zhang, Z. Sun, A. Ruzsinszky, H. Peng, Z. Yang, A. Paul, U. Waghmare, X. Wu, et al., Nature Chemistry 8 (9) (2016) 831–836.
[38] B.G. Janesko, T.M. Henderson, G.E. Scuseria, Phys. Chem. Chem. Phys. 11 (2009) 443–454. http://dx.doi.org/10.1039/B812838C.
[39] H.A. Tahini, A. Chroneos, S.T. Murphy, U. Schwingenschlögl, R.W. Grimes, J. Appl. Phys. 114 (6) (2013). http://dx.doi.org/10.1063/1.4818484.
[40] M.G. Medvedev, I.S. Bushmarinov, J. Sun, J.P. Perdew, K.A. Lyssenko, Science 355 (6320) (2017) 49–52. http://dx.doi.org/10.1126/science.aah5975. arXiv:http://science.sciencemag.org/content/355/6320/49.full.pdf. http://science.sciencemag.org/content/355/6320/49.
[41] J.W. Doak, K.J. Michel, C. Wolverton, J. Mater. Chem. C 3 (40) (2015) 10630–10649.
[42] S.B. Zhang, J.E. Northrup, Phys. Rev. Lett. 67 (1991) 2339–2342. http://dx.doi.org/10.1103/PhysRevLett.67.2339.
[43] S.P. Ong, W.D. Richards, A. Jain, G. Hautier, M. Kocher, S. Cholia, D. Gunter, V.L. Chevrier, K.A. Persson, G. Ceder, Comput. Mater. Sci. 68 (2013) 314–319.
[44] A. Jain, G. Hautier, S.P. Ong, C.J. Moore, C.C. Fischer, K.A. Persson, G. Ceder, Phys. Rev. B 84 (2011) 045115. http://dx.doi.org/10.1103/PhysRevB.84.045115.
[45] B. Medasani, M.L. Sushko, K.M. Rosso, D.K. Schreiber, S.M. Bruemmer, J. Phys. Chem. C 121 (3) (2017) 1817–1831. http://dx.doi.org/10.1021/acs.jpcc.7b00071.
[46] H.A. Tahini, X. Tan, U. Schwingenschlgl, S.C. Smith, ACS Catal. 6 (8) (2016) 5565–5570. http://dx.doi.org/10.1021/acscatal.6b00937.
[47] P. Ágoston, K. Albe, R.M. Nieminen, M.J. Puska, Phys. Rev. Lett. 103 (24) (2009) 245501.
[48] G. Hautier, S.P. Ong, A. Jain, C.J. Moore, G. Ceder, Phys. Rev. B 85 (2012) 155208. http://dx.doi.org/10.1103/PhysRevB.85.155208.

[49] W. Sun, S.T. Dacek, S.P. Ong, G. Hautier, A. Jain, W.D. Richards, A.C. Gamst, K.A. Persson, G. Ceder, Sci. Adv. 2 (11) (2016). http://dx.doi.org/10.1126/sciadv.1600225. arXiv:http://advances.sciencemag.org/content/2/11/e1600225.full.pdf.
[50] S. Lany, A. Zunger, Phys. Rev. B 78 (23) (2008) 235104.
[51] S.E. Taylor, F. Bruneval, Phys. Rev. B 84 (7) (2011) 075155.
[52] C. Freysoldt, J. Neugebauer, C.G. Van de Walle, Physica B 248 (5) (2011) 1067–1076. http://dx.doi.org/10.1002/pssb.201046289.
[53] G. Makov, M.C. Payne, Phys. Rev. B 51 (1995) 4014–4022. http://dx.doi.org/10.1103/PhysRevB.51.4014.
[54] H.-P. Komsa, T.T. Rantala, A. Pasquarello, Phys. Rev. B 86 (2012) 045112. http://dx.doi.org/10.1103/PhysRevB.86.045112.
[55] S. Boeck, C. Freysoldt, A. Dick, L. Ismer, J. Neugebauer, Comput. Phys. Comm. 182 (3) (2011) 543–554. http://dx.doi.org/10.1016/j.cpc.2010.09.016. http://www.sciencedirect.com/science/article/pii/S0010465510003619.
[56] G. Petretto, F. Bruneval, Phys. Rev. B 92 (22) (2015) 224111. http://dx.doi.org/10.1103/PhysRevB.92.224111.
[57] P. Deák, B. Aradi, M. Kaviani, T. Frauenheim, A. Gali, Phys. Rev. B 89 (7) (2014) 075203. http://dx.doi.org/10.1103/PhysRevB.89.075203.
[58] A. Chroneos, H.A. Tahini, U. Schwingenschlögl, R.W. Grimes, J. Appl. Phys. 116 (2) (2014). http://dx.doi.org/10.1063/1.4887135.
[59] F. Corsetti, A.A. Mostofi, Phys. Rev. B 84 (3) (2011) 035209. http://dx.doi.org/10.1103/PhysRevB.84.035209.
[60] P. Li, S. Deng, L. Zhang, G. Liu, J. Yu, Chem. Phys. Lett. 531 (2012) 75–79. http://dx.doi.org/10.1016/j.cplett.2012.02.008.
[61] L.S. dos Santos, W.G. Schmidt, E. Rauls, Phys. Rev. B 84 (11) (2011) 115201. http://dx.doi.org/10.1103/PhysRevB.84.115201.
[62] J. Neugebauer, C.G. Van de Walle, Phys. Rev. B 50 (11) (1994) 8067–8070. http://dx.doi.org/10.1103/PhysRevB.50.8067.
[63] M. O'Keeffe, N.E. Brese, J. Am. Chem. Soc. 113 (9) (1991) 3226–3229. http://dx.doi.org/10.1021/ja00009a002.
[64] A. Gibson, R. Haydock, J.P. LaFemina, Phys. Rev. B 50 (1994) 2582–2592. http://dx.doi.org/10.1103/PhysRevB.50.2582.
[65] B. Peters, J. Chem. Phys. 131 (24) (2009) 244103.
[66] N.E.R. Zimmermann, B. Vorselaars, D. Quigley, B. Peters, J. Am. Chem. Soc. 137 (41) (2015) 13352–13361.
[67] S. Decoster, B. De Vries, U. Wahl, J.G. Correia, A. Vantomme, Appl. Phys. Lett. 93 (14) (2008) 141907. http://dx.doi.org/10.1063/1.2996280.
[68] S. Decoster, S. Cottenier, B. De Vries, H. Emmerich, U. Wahl, J.G. Correia, A. Vantomme, Phys. Rev. Lett. 102 (6) (2009) 065502. http://dx.doi.org/10.1103/PhysRevLett.102.065502.
[69] S. Decoster, B. De Vries, U. Wahl, J.G. Correia, A. Vantomme, J. Appl. Phys. 105 (8) (2009) 083522. http://dx.doi.org/10.1063/1.3110104.
[70] S. Decoster, S. Cottenier, U. Wahl, J.G. Correia, A. Vantomme, Phys. Rev. B 81 (15) (2010) 155204. http://dx.doi.org/10.1103/PhysRevB.81.155204.
[71] S. Decoster, S. Cottenier, U. Wahl, J.G. Correia, L.M.C. Pereira, C. Lacasta, M.R. Da Silva, A. Vantomme, Appl. Phys. Lett. 97 (15) (2010) 151914. http://dx.doi.org/10.1063/1.3501123.
[72] L.M.C. Pereira, U. Wahl, S. Decoster, J.G. Correia, M.R. da Silva, A. Vantomme, J.P. Araújo, Appl. Phys. Lett. 98 (20) (2011) 201905. http://dx.doi.org/10.1063/1.3592568.
[73] L.M.C. Pereira, U. Wahl, S. Decoster, J.G. Correia, L.M. Amorim, M.R. da Silva, J.P. Araujo, A. Vantomme, Phys. Rev. B 86 (12) (2012) 125206. http://dx.doi.org/10.1103/PhysRevB.86.125206.
[74] S. Decoster, U. Wahl, S. Cottenier, J.G. Correia, T. Mendonça, L.M. Amorim, L.M.C. Pereira, A. Vantomme, J. Appl. Phys. 111 (5) (2012) 053528. http://dx.doi.org/10.1063/1.3692761.
[75] L.M. Amorim, U. Wahl, L.M.C. Pereira, S. Decoster, D.J. Silva, M.R. da Silva, A. Gottberg, J.G. Correia, K. Temst, A. Vantomme, Appl. Phys. Lett. 103 (26) (2013) 262102. http://dx.doi.org/10.1063/1.4858389.
[76] D.J. Silva, U. Wahl, J.G. Correia, L.M.C. Pereira, L.M. Amorim, M.R. da Silva, E. Bosne, J.P. Araújo, J. Appl. Phys. 115 (2) (2014) 023504. http://dx.doi.org/10.1063/1.4861142.
[77] H. Hofsäss, G. Lindner, Phys. Rep. (Rev. Sec. Phys. Lett.) 201 (3) (1991) 121–183. http://dx.doi.org/10.1016/0370-1573(91)90121-2.
[78] M.R. Silva, U. Wahl, J.G. Correia, L.M. Amorim, L.M.C. Pereira, Rev. Sci. Instrum. 84 (7) (2013) 073506. http://dx.doi.org/10.1063/1.4813266.
[79] Z. Rong, R. Malik, P. Canepa, G.S. Gautam, M. Liu, A. Jain, K. Persson, G. Ceder, Chem. Mater. 27 (17) (2015) 6016–6021. http://dx.doi.org/10.1021/acs.chemmater.5b02342.
[80] Y. Wang, W.D. Richards, S.P. Ong, L.J. Miara, J.C. Kim, Y. Mo, G. Ceder, Nature Mater. 14 (10) (2015) 1026–1031. http://dx.doi.org/10.1038/NMAT4369.
[81] A. De Vita, The Energetics of Defects and Impurities in Metals and Ionic Materials from First Principles, Keele University, 1992.
[82] J.D. Hunter, Comput. Sci. Eng. 9 (3) (2007) 90–95.
[83] Numpy Developers, http://numpy.org/.
[84] A. Jain, S.P. Ong, W. Chen, B. Medasani, X. Qu, M. Kocher, M. Brafman, G. Petretto, G.-M. Rignanese, G. Hautier, D. Gunter, K.A. Persson, Concurr. Comput.: Pract. Exper. 27 (17) (2015) 5037–5059. http://dx.doi.org/10.1002/cpe.3505. CPE-14-0307.R2.
[85] P.A. Schultz, O.A. von Lilienfeld, Modelling Simulation Mater. Sci. Eng. 17 (8) (2009) 084007. http://stacks.iop.org/0965-0393/17/i=8/a=084007.
[86] F. El-Mellouhi, N. Mousseau, Phys. Rev. B 71 (2005) 125207. http://dx.doi.org/10.1103/PhysRevB.71.125207.
[87] J.T. Schick, C.G. Morgan, P. Papoulias, Phys. Rev. B 66 (2002) 195302. http://dx.doi.org/10.1103/PhysRevB.66.195302.
[88] H.-P. Komsa, A. Pasquarello, Microelectron. Eng. 88 (7) (2011) 1436–1439. http://dx.doi.org/10.1016/j.mee.2011.03.081. Proceedings of the 17th Biennial International Insulating Films on Semiconductor Conference17th Biennial International Insulating Films on Semiconductor Conference. http://www.sciencedirect.com/science/article/pii/S0167931711003406.
[89] J.E. Northrup, S.B. Zhang, Phys. Rev. B 50 (1994) 4962–4964. http://dx.doi.org/10.1103/PhysRevB.50.4962.
[90] H.-P. Komsa, A. Pasquarello, Phys. Rev. B 84 (2011) 075207. http://dx.doi.org/10.1103/PhysRevB.84.075207.
[91] K. Lejaeghere, G. Bihlmayer, T. Bjoerkman, P. Blaha, S. Bluegel, V. Blum, D. Caliste, I.E. Castelli, S.J. Clark, A. Dal Corso, S. de Gironcoli, T. Deutsch, J.K. Dewhurst, I. Di Marco, C. Draxl, M. Dulak, O. Eriksson, J.A. Flores-Livas, K.F. Garrity, L. Genovese, P. Giannozzi, M. Giantomassi, S. Goedecker, X. Gonze, O. Granaes, E.K.U. Gross, A. Gulans, F. Gygi, D.R. Hamann, P.J. Hasnip, N.A.W. Holzwarth, D. Iusan, D.B. Jochym, F. Jollet, D. Jones, G. Kresse, K. Koepernik, E. Kuecuekbenli, Y.O. Kvashnin, I.L.M. Locht, S. Lubeck, M. Marsman, N. Marzari, U. Nitzsche, L. Nordstrom, T. Ozaki, L. Paulatto, C.J. Pickard, W. Poelmans, M.I.J. Probert, K. Refson, M. Richter, G.-M. Rignanese, S. Saha, M. Scheffler, M. Schlipf, K. Schwarz, S. Sharma, F. Tavazza, P. Thunstroem, A. Tkatchenko, M. Torrent, D. Vanderbilt, M.J. van Setten, V. Van Speybroeck, J.M. Wills, J.R. Yates, G.-X. Zhang, S. Cottenier, Science 351 (6280) (2016) 1415–U81.