Computational Materials Science
DIMS: A tool for setting up defects and impurities CASTEP calculations
--Manuscript Draft--

Manuscript Number:	COMMAT-D-21-02455
Article Type:	Full Length Article
Section/Category:	Electronic Structure
Keywords:	software; bash; defects; tool; crystallography; castep; hpc
Corresponding Author:	Stavros-Richard G. Christopoulos, (Ph.D. Coventry University Coventry, West Midlands UNITED KINGDOM
First Author:	Stavros-Richard G. Christopoulos, (Ph.D.
Order of Authors:	Stavros-Richard G. Christopoulos, (Ph.D.
	Konstantina Papadopoulou
	Alexandros Konios
	David Parfitt
Abstract:	In the present study, we introduce a new tool for calculating the properties of different crystallographic structures, either pure or with defects. The proposed software's three different modes regarding the inputs it accepts, i.e., automatic, semi-automatic and manual, are explained. Vacancies, anti-sites, interstitials and dopants can be processed, in any number of combinations. In addition, research studies where the tool has been already applied are provided. Finally, we describe the advantages of the proposed tool regarding mass calculations, time management and human error, and we showcase, through the means of performance analysis, its weak and strong points using two case studies.
Suggested Reviewers:	Panayiotis Varotsos National and Kapodistrian University of Athens: Ethniko kai Kapodistriako Panepistemio Athenon pvaro@otenet.gr
	Yerassimos Panagiotatos University of West Attica Aigaleo Grove Campus: Panepistemio Dytikes Attikes Panepistemioupole Alsous Aigaleo gpana@uniwa.gr
	Ioannis Goulatis yannislgoul@gmail.com

Title: DIMS: A tool for setting up defects and impurities CASTEP calculations

Authors: Stavros-Richard G. Christopoulos , Konstantina A. Papadopoulou, Alexandros Konios, and David Parfitt

Coventry, September 2021

Dear Editor,

Here we present DIMS, a new tool for calculating the properties of different crystallographic structures, either pure or with defects, working alongside CASTEP as supporting software. Vacancies, anti-sites, interstitials and dopants can be processed by the tool, in any number of combinations.

We anticipate that the present study will have a high impact in the community, as it will minimize the potential for human error, as well as control mass calculations and offer better time management. Research studies where the tool has already been applied are provided so as to showcase its usefulness.

Yours faithfully,

The authors

begindocument/before

DIMS: A tool for setting up defects and impurities CASTEP calculations

Stavros-Richard G. Christopoulos,[1,*] Konstantina A. Papadopoulou,[1] Alexandros Konios,[2] and David Parfitt[1]

[1]*Faculty of Engineering, Environment and Computing, Coventry University, Priory Street, Coventry, CV1 5FB, United Kingdom*
[2]*School of Digital, Technologies and Arts, Staffordshire University, United Kingdom*

In the present study, we introduce a new tool for calculating the properties of different crystallographic structures, either pure or with defects. The proposed software's three different modes regarding the inputs it accepts, i.e., automatic, semi-automatic and manual, are explained. Vacancies, anti-sites, interstitials and dopants can be processed, in any number of combinations. In addition, research studies where the tool has been already applied are provided. Finally, we describe the advantages of the proposed tool regarding mass calculations, time management and human error, and we showcase, through the means of performance analysis, its weak and strong points using two case studies.

Keywords: software, bash, defects, tool, crystallography, castep, hpc

I. INTRODUCTION

When it comes to the study of large crystallographic structures using the CASTEP package[1–5], it is of grave importance to minimize the run-time as well as the potential for human error, especially in the cases where the number of jobs a user has to run surpasses the thousands.

In the present paper, we introduce a tool called DIMS (Defects and Impurities Setup) created in order to automate the process of adding defects in a structure. The tool deals with three kinds of point defects[6,7]: vacancies, anti-sites and interstitials. Furthermore, two kinds of dopants[8,9] can be accepted: substitutional, including multidoping, and interstitial.

The proposed software is able to create all the necessary folders and files that CASTEP needs in order to execute a number of submitted jobs, however numerous, instead of the user having to do so by hand. In the case of too many calculations, the latter is impossible, not only regarding human error, but also time-wise.

In Section II, we perform a Performance Analysis of the tool, testing it against supercells as large as 937 atoms and 100,000 inserted interstitials. In addition, in Section III, we discuss the advantages and disadvantages of the software, as well as future improvements. Examples of publications where the software has already been used are also showcased. Finally, in Section IV, we briefly describe how the tool needs to be set up from the user in order to run, alongside how the different modules of it are executed.

II. RESULTS

In order to evaluate the performance of the proposed tool, we tested it against two separate supercell sizes of Ti_3AlC_2, examining various numbers of interstitial dopants. The aim was to examine how the number of submitted jobs affects the run-time of the programme regarding the creation of all necessary folders and files.

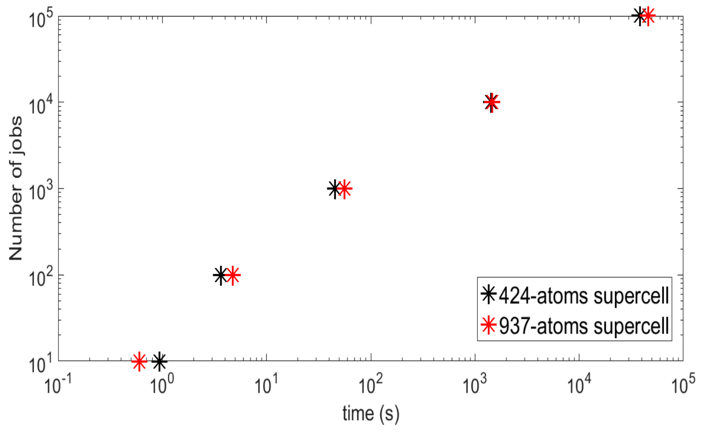

FIG. 1: How the run-time of the proposed tool increases with the number of submitted jobs to run when using an HDD and Linux operating system.

The first supercell examined was the 3x3x3 containing 424 atoms while the second was the 4x4x4 containing 937 atoms.

In the first case, we run the software in a system utilizing Linux Ubuntu 20.04, a Hitachi HTS545050A7E380 500GB Hard Disk Drive (HDD) with 300MB/s writing speed, an i3-3217U processor, and 3832MB RAM. In Figure 1, we can see the dependence of the run-time on the number of jobs to execute, for the two sizes of supercells.

We conducted five different doping cases for each of the supercells so that we could examine how the run-time is affected when extra number of atoms are added to the initial supercell. As it was expected, increasing the number of dopants and hence the number of submitted jobs, the programme's execution time also increases, as it is evident in Fig. 1.

The 937-atoms supercell is 121% larger than the 424-atoms one. For the first three cases of 10, 100 and 1,000 interstitials, the execution time is less than one minute for both supercell sizes. However, for 10,000 intersti-

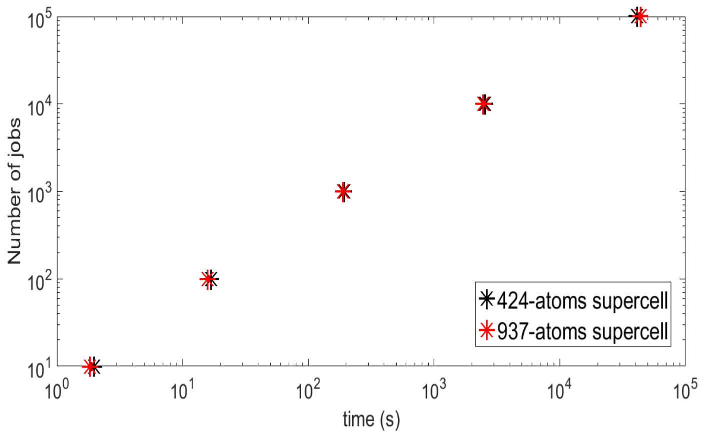

FIG. 2: How the run-time of the proposed tool increases with the number of submitted jobs to run, when we use an SSD and Windows 10 operating system.

tials, the software has a run-time of 23.9min for the 424-atoms supercell and 24.3min for the 937-atoms supercell. In addition, for the last case of 100,000 interstitials, the software has a run-time of 10.6h for the 424-atoms supercell and 12.8h for the 937-atoms supercell, an increase of 20.75%.

As we can see, as the workload has increased from the first to the last case by 999, 900%, the run time has also increased drastically from less than a second to hours. However, if one is to compare the run-times of the software with the ones it would take a user to create and submit all the jobs manually while ensuring there are no mistakes, it is logical that the automated software we propose in this article will be of great use.

Using Solid State Drives (SSD) instead of HDD, one can have the advantage of reading and writing data faster while using less energy. For comparison, we ran the aforementioned doping cases anew, utilizing a KIOXIA, XG6 series 1024GB SSD, reaching up to sequential read speeds of 3180MB/s, sequential write speeds of 2960MB/s and delivering up to 355,000 random read and 365,000 random write IOPS. The operating system was Windows 10 Pro, Version 20H2, with Linux Bash Shell installed, an i5-10210U processor, and 8GB RAM.

The dependence of the execution time on the number of submitted jobs for this case is shown in Fig. 2.

Again, for the first two cases of 10 and 100 interstitials, the execution time is less than one minute for both supercell sizes. However, for 1,000 interstitials, the SSD is slower than the HDD, with both supercell sizes running for approximately 3min. This is also the case for the 10,000 interstitials where the 3x3x3 supercell has a run-time of 42.48min, an 77.67% increase from the corresponding HDD case, and the 4x4x4 supercell has a run-time of 41.13min, 69.58% slower than the corresponding HDD case.

Finally, for the case of 100,000 interstitials, for the 3x3x3 supercell the software's execution time is 11.39h, an 7.25% increase from the corresponding HDD case. For the 4x4x4 supercell however, the execution time is 12.19h, which is actually 4.78% faster than the corresponding HDD case. The latter percentage practically translates to 36.6min in real time, which is negligent when compared to the time it would take a user to create 100,000 folders and the corresponding files manually.

It is evident that the use of Linux has more advantages regarding the proposed tool's speed, regardless of the use of an SSD. In addition, the run-time does not increase linearly with the number of jobs, a fact that showcases the proposed tools usefulness when it comes to setting up an amount of jobs that surpasses the thousands.

In general, it can be concluded that despite the fact that the execution time increases with the number of the inserted interstitial atoms and, thus, with the number of submitted jobs to run, the tool performs equally well in all cases, showing that it can cope with extremely large number of calculations for significantly large crystallographic structures.

III. DISCUSSION

In the present paper, we present a tool developed to calculate the properties of different crystallographic structures. The main advantage of the proposed software is that it utilizes only bash, therefore no compilers need to be built-in in order to execute it in Linux operating systems, or High-Performance Cluster (HPC) with Linux operating systems. Due to that fact, it can be expanded for use with other platforms, for example Windows if the Linux Bash Shell is installed.

In addition, the tool allows for the execution of millions of calculations automatically, when a manual approach could take a lot of time and the risk of human error would be high. That being said, the software is designed in such a way so that if someone was to skip the automatic process, the programme could still run either semi-automatically or completely manually. Furthermore, it can be easily incorporated into any other script for automating even more complex calculations for similar group of compositions or even extend the current version in such a way that it will include migration, multi-doping and other cases.

Finally, the software is student-friendly, as the necessary parameters can be inserted without previous knowledge of HPC or Linux.

The software has already been proved useful in a number of studies. For example, in Ref. 10, the authors examined carbon interstitial defects and self-interstitials in a 250-atoms supercell of neutron-irradiated silicon containing carbon. Moreover, in Ref. 11, the authors studied carbon and oxygen-related defects formed upon irradiation of silicon. A supercell of 250 silicon atomic sites was used. In addition, in Ref. 12, substitutional phosphorous-vacancy pairs were studied in a random silicon germanium alloy. The special quasirandom structures[13] (SQS) method was used to reduce the size of the supercell, and hence the time of the calculations. 896 phosphorous substitutional-vacancy defects were considered, and more than 1350 DFT calculations were performed.

Furthermore, in Ref. 14, n-type and p-type dopants were examined also in a random silicon germanium alloy, using a 64-atomic site supercell. Finally, in Ref. 15, a 250-atomic site supercell was used to study the structure and energetics of carbon interstitials and substitutionals in silicon. A 10x10x10 grid of interstitials was used for each step of adding interstitials, resulting in 3000 submitted jobs.

Despite the software's many advantages, it is true that some points could still be optimized. For instance, as it is shown in Section II, the run-time increases drastically with the number of the jobs submitted. Moreover, while parsing the .cell files, we refer to the file line for an atom's position and not to the atom itself. That means that if for example we want the position of the 100^{th} atom, we should look into the 116^{th} line of the .cell file. That fact increases the potential for human error, especially for a beginner user, however the method is in place because HPC allows the user to view a file through its lines. Additionally, instead of running all the jobs simultaneously using `./runAll.sh` (see Section IV), we could find a way of utilizing blocks of jobs, e.g. one block per n number of jobs, where n is chosen by the user, so that we could execute the calculations in parts.

Another issue might arise from the use of the resubmit.sh script which resubmits for continuation all the jobs that have not finished after the initial allocated time for the run. We incorporated this script because we judged it useful when it comes to a high number of jobs, else the user would have to resubmit the unfinished jobs manually. That, in cases of thousands of jobs, is impractical. However, a user needs to be careful to adjust the resubmit.sh script accordingly if they use an HPC.

Finally, as it currently stands, the software does not allow for multiple interstitials. The solution to this would be something to look into in the future.

IV. METHODS

The proposed tool enables a CASTEP user to create all the necessary folders and files in order to set up a big number of calculations regarding the properties of different crystallographic structures. In addition, it gives the user control by automating the process, minimizing the potential for human error when it comes to thousands of calculations, while drastically reducing the time needed to set up the calculations.

The software's two basic computational modes, i.e., automatic and semi-automatic, operate under a main algorithm which determines the mode according to the inputs. The flowchart of this main algorithm is pictured in Fig. 3, while we will explain its main characteristics in the subsequent paragraphs.

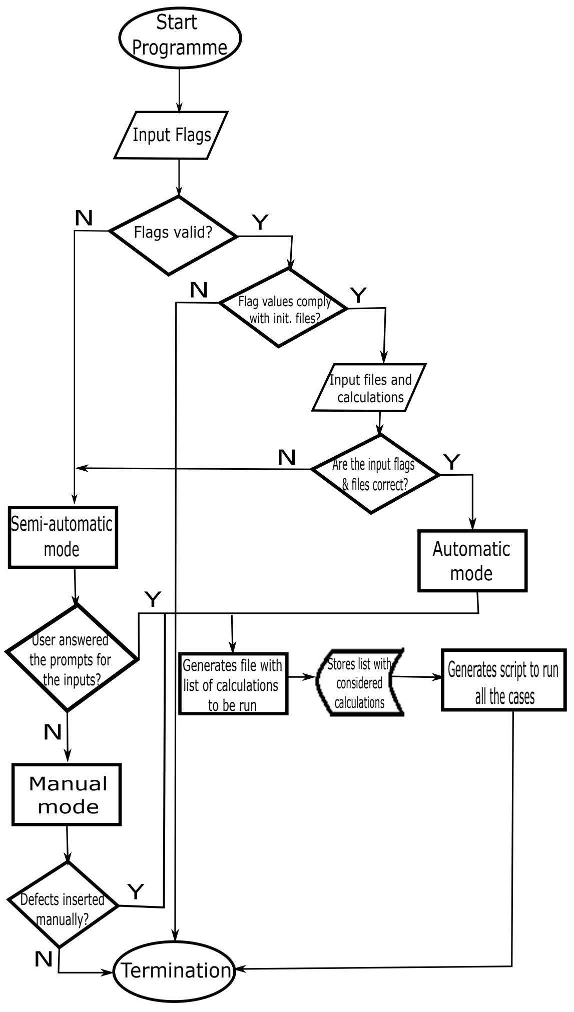
Start Programme
Input Flags
Flags valid?
N
Y
Flag values comply with init. files?
N
Y
Input files and calculations
Are the input flags & files correct?
N
Y
Semi-automatic mode
Automatic mode
User answered the prompts for the inputs?
Y
N
Generates file with list of calculations to be run
Stores list with considered calculations
Generates script to run all the cases
Manual mode
Defects inserted manually?
Y
N
Termination

FIG. 3: The flowchart of the main algorithm, which determines whether the automatic, the semi-automatic or the manual mode will be selected, according to the programme's inputs.

Initially, the user needs to define the flags that will set up the process. A list of the available flags is given in Table I. The flags -f, -i, -s, -d must be followed by appropriate files that contain the positions of the vacancies and anti-sites, the interstitials, the substitutionals and the interstitial dopants. The format of these files is given in the Supplementary Material. The flags -v and -a denote whether or not we want to do calculations for vacancies and anti-sites respectively. The label -l notes the job's name, which should be the same as the name of the .param file. The label -c defines the initial .cell file and should be followed by it, while the label -p denotes whether we want to perform calculations for the initial .cell file. In addition, apart from the individual flags, we can use the combinations -vaf, -vf, -af, where the -f flag that is followed by the file containing the positions of vacancies and anti-sites should always come last.

In order to run the programme, the user needs to create a folder containing the .resubmit, .cell, .param, and modified .slurm files. An example of a command submitting the script to run is the following:

```
DefectCalcV5.3.1.multidoping.sh -l jobname -p -c jobname.initial.cell

-vaf Vacancy_positions.dat -i Interstitial_positions.dat

-s Substitutional_Doping.dat -d Interstitial_Doping.dat
```

The .slurm file is the job submission file, used to submit the programme and ask for resources from the HPC or the system used. It is the file that calls CASTEP to submit the jobs and run the calculations. In it, the run-time, the partition of the HPC we want to use, and the number of nodes is also noted. The user must be careful to modify this file according to the HPC they use, if any, or according to the system they have CASTEP installed in. The parameter of interest here is the "SETJOBNAME". Using this parameter, the user can transfer the input parameter of the -l label to the corresponding working directories and case names.

After initialization, the programme checks the validity of the flags and whether the format of the input files is compatible with its commands. At this stage, the process can be terminated only in the cases where all the flags are invalid or all the input files do not exist. That means that the calculations can continue to run even if a partially wrong command is entered, ignoring the invalid flags. In that case, the programme chooses the semi-automatic mode and prompts the user to input the right command.

When the programme has completed the flags and files validity check and has chosen between the two operating modes, a series of files and scripts is generated, necessary to store the calculations and run them.

First of all, in order to ensure that when the programme is executed for a second time it does not leave any remnants of its previous run, all the previously generated files are deleted. Then all the necessary subfolders for the submission jobs are created and the .param, .cell, .slurm files are copied into each of them. The parameter "SETJOBNAME" in the .slurm file is also changed accordingly.

Furthermore, a file titled runAll.sh is generated, which contains all the jobs to be run. The user is then prompted to run them using `./runAll.sh`.

In addition, the CheckContinuefile.sh script is created. This script allows the user to check which jobs have reached the allocated time-limit and need to be resubmitted. Basically, it collects the case names stopped due to time-limit by checking whether the slurm-#.out files contain the word "LIMIT".

The results are gathered in the AllRES.dat file, and the final energies in particular in the finalEnergyFile.sh script. The latter can be run by the user using `./finalEnergyFile.sh` to check all the final energies for all cases running in the AllRES.dat file. Moreover, the checkWarnings.sh script provides any warnings the calculations may have produced.

Finally, before its termination, the programme gathers all the .castep output files into a directory called `castep.Job_name.Files`.

Both the automatic and semi-automatic modes have counters put in in order to count the times each individual calculation (vacancies, anti-sites, interstitials, substitutional, interstitial doping) is to be run. However, the semi-automatic mode has s few extra counters in order to store the data values inserted manually for the vacancy, interstitial, interstitial doping and substitutional doping calculations. In addition, the highest number of the different elements that can be entered manually by the user is set to $1,000$.

We will now describe briefly the main points regarding the way the different calculations are set up, using the automatic, semi-automatic and manual modes in the Subsections IV A and IV B respectively.

A. Automatic mode

1. *Initial cell calculations*

If the flag -p is equal to "true", the programme copies the initial .cell file into the path `./Job_name/cells`, where the `Job_name` is the parameter following the -l label. The file is then renamed, following the syntax `Job_name.initial.cell`. The latter is then moved to the directory named *cells*.

At this stage, the programme assigns to the first element of an array called `Unique_nameDef` the `Job_name.initial.cell` filename. `Unique_nameDef` will also be used to store the defect calculations.

The programme then proceeds to make the folders with the necessary files where the calculations of the final energy of the initial crystal will be executed. However, if the -p flag is not activated, it skips this calculation and carries on with the remaining ones.

The -p flag should always be accompanied by the -c one, followed by the initial .cell file.

2. *Vacancy calculations*

If the -f flag is activated, it should be followed by the file `Vacancy_positions.dat` which contains the positions of the vacancies and the anti-sites. The file is then read line by line. The `Vacancy_positions.dat` file has three columns: the first is for the number of the lines where the vacancies will be created in the .cell file, the second stores the names of the elements existing in those lines, and the third stores the labels for the different crystallographic positions that we want to examine for the respective elements.

As a next step, the activation of the -v flag is examined. If it is activated, the initial .cell file is copied into the folder named after the `Job_name` parameter like before. Then, in order to create the vacancies, the programme starts parsing through the lines of the .cell indicated by the first column of `Vacancy_positions.dat` and commenting them out.

Finally, the .cell file is moved to the *cells* folder for backup, and all the unique elements that were found in the `Vacancy_positions.dat` file are stored in the `Unique_nameDef` array. So if, for example, we created vacancies where three different elements should be, then the `Unique_nameDef` array will expand by three lines. But if we created vacancies in three positions where the same element existed, the `Unique_nameDef` array will be expanded by one line, since the programme will run one job for the vacancies of the same type.

3. *Anti-site calculations*

First, the activation of the -a flag is examined. If it is activated, the programme initially checks whether the two elements to be displaced are the same or not. If they are different, using the `Vacancy_positions.dat` file all the possible combinations for the anti-sites are derived.

Next, the programme copies the `Job_name.initial.cell` file into the *cells* directory renaming it into `Job_name.ElementONElement.cell`. Finally, the anti-sites are created by processing the first column of the `Vacancy_positions.dat`, which indicates the line in the initial .cell file where the displacement takes place. The existing element in that line is then replaced as required.

4. *Interstitial calculations*

The flag that needs to hold the value "true" for the interstitials calculations is the -i, followed by the file where the interstitial positions are stored.

The programme then copies the `Job_name.initial.cell` file into the *cells* directory renaming it according to the line the interstitial will be and the `Job_name`. Finally, under the `BLOCK_POSITIONS_FRAC` section of the latter .cell file, lines including the coordinates of the interstitial elements are created and the final .cell file is moved to the *cells* directory.

For the interstitial calculations, interstitials are created for all the elements noted in the file containing the vacancy positions. As such, the -f flag needs to also be used alongside the file containing the vacancy positions, even if we do not want to perform calculations for the vacancies. More information on that is included on the Supplementary Material.

5. *Interstitial Doping calculations*

For this section, the flag -d needs to be activated, followed by the file containing the positions and the name of the interstitials. Same as before, the programme copies the `Job_name.initial.cell` file into the *cells* directory, renaming it accordingly. After that, the programme functions the same way as in paragraph IV A 4.

6. *Substitutional Doping calculations*

For the substitutional doping calculations, the activation of the -s flag is first examined, and whether it is followed by the file containing the positions of the substitutional elements. If the aforementioned condition holds true, the programme copies the `Job_name.initial.cell` file into the *cells* directory, renaming it according to the element to be substituted and its position in the .cell file.

In order to create the substitutional, the programme then finds the line in the .cell file where the substitution will occur and substitutes the element existing in it with the one indicated by the substitutional doping file. The final .cell file after this procedure is finished, is then moved to the *cells* directory.

Finally, the software creates all the necessary files and subfolders inside the main folder named after the job-name as it was input using the -l label.

B. Semi-Automatic and Manual modes

For the semi-automatic mode to be activated, no flags should have been entered. In that case, the programme prompts the user to enter the job-name to be submitted. Next, the existence of an initial .cell file is examined. If such a file is not found, the user is also prompted to provide it manually.

The various calculations are set up as in Subsection IV A. The only difference is that in the semi-automatic mode, the programme will prompt the user to answer with "yes" or "no" whether they want a specific calculation to be run. If the answer is "yes", the respective flag will be activated and the user will be prompted anew to enter the file corresponding to that particular calculation. If no file is entered, a completely manual mode emerges where the user must input all the defect positions by hand.

Data Availability

No data were used relevant to the purpose of this paper. The software is designed so as to accept any .cell file as input for setting up CASTEP calculations.

Author Contributions

Stavros-Richard G. Christopoulos was responsible for the investigation of the methodology, conceptualization, code development, formal analysis, review and editing; Konstantina A. Papadopoulou performed the performance analysis evaluation, writing, formal analysis, review and editing; Alexandros Konios did the writing; David Parfitt did the review and editing.

Acknowledgments

Konstantina Papadopoulou acknowledges support from the International Consortium of Nanotechnologies (ICON) funded by Lloyd's Register Foundation, a charitable foundation which helps to protect life and property by supporting engineering-related education, public engagement and the application of research.

TABLE I: List of the available flags.

Flag	Meaning
-p	calculation for the initial file(pure crystal)
-l	label for HPC, same as the .param file
-c	always followed by the initial .cell file
-v	vacancies calculations
-a	anti-site calculations
-f	file with vacancies and anti-sites positions
-i	interstitials calculations
-s	substitutionals calculations
-d	interstitial doping calculations
-r	resubmit unfinished jobs

* Electronic address: ac0966@coventry.ac.uk

[1] S. J. Clark, M. D. Segall, C. J. Pickard, P. J. Hasnip, M. I. Probert, K. Refson, and M. C. Payne, Zeitschrift für Kristallographie-Crystalline Materials **220**, 567 (2005).

[2] P. Hohenberg and W. Kohn, Phys. Rev. **136**, B864 (1964).

[3] W. Kohn and L. J. Sham, Phys. Rev. **140**, A1133 (1965).

[4] M. C. Payne, M. P. Teter, D. C. Allan, T. Arias, and J. D. Joannopoulos, Rev. Mod. Phys. **64**, 1045 (1992).

[5] B. G. Pfrommer, M. Cote, S. G. Louie, and M. L. Cohen, J. Comput. Phys. **131**, 233 (1997).

[6] A. Lidiard, Science Progress (1933-) pp. 103–129 (1968).

[7] C. Kittel and P. McEuen, *Introduction to solid state physics*, vol. 8 (Wiley New York, 1976).

[8] B. Averill and P. Eldredge, Washington, DC: Saylor Academy p. 1472 (2015).

[9] A. Lavakumar, Morgan & Claypool Publishers (2017).

[10] C. Londos, S.-R. Christopoulos, A. Chroneos, T. Angeletos, M. Potsidi, and G. Antonaras, Journal of Materials Science: Materials in Electronics **31**, 930 (2020).

[11] M. S. Potsidi, N. Kuganathan, S.-R. G. Christopoulos, A. Chroneos, T. Angeletos, N. V. Sarlis, and C. A. Londos, Crystals **10**, 1005 (2020).

[12] S.-R. G. Christopoulos, N. Kuganathan, and A. Chroneos, Scientific Reports **9**, 1 (2019).

[13] A. Zunger, S.-H. Wei, L. Ferreira, and J. E. Bernard, Physical Review Letters **65**, 353 (1990).

[14] S.-R. G. Christopoulos, N. Kuganathan, and A. Chroneos, Scientific Reports **10**, 1 (2020).

[15] S.-R. G. Christopoulos, E. N. Sgourou, R. V. Vovk, A. Chroneos, and C. A. Londos, Materials **11**, 612 (2018).

Declaration of Interest Statement

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.