CellMatch: combining two unit cells into a common supercell with minimal strain

Predrag Lazić [a,1,2],

[a] *Department of Physics, University at Buffalo, New York 14260-1500, USA*

Abstract

Recent emergence of 2D materials (the so called van der Waals materials), of which graphene is the most famous one, opens new routes in creation of novel materials by mere layer-by-layer combinations. Moreover, a growth of such materials is typically done on a substrate. In both cases structures appear that are periodical in the plane but the periodicity is very different from a simple 1×1 commensurate unit cells combinations which appears for materials with very similar values of lattice constants. Much more common is the case in which a new periodic cell is of a moiré type - such as 10×10 over 9×9 in case of graphene on Ir(111). Once the shape of the common supercell for 2 different 2D materials, or a material and the surface is found - it is easy to do a computational treatment with appropriate method for electronic structure - such as density functional theory, tight binding or some other. The purpose of the *CellMatch* code is to generate such common super cell given the two unit cells of selected materials. The *CellMatch* code searches within given combinatorial space and sorts results by the strain imposed on one of the components, while the other component experiences zero strain.

PACS: 71.15.-m; 71.15.Mb; 71.45.Gm

Key words: Electronic structure; Density functional theory; Van der Waals materials; commensurate structures; moiré patterns; epitaxial growth,

PROGRAM SUMMARY/NEW VERSION PROGRAM SUMMARY

Manuscript Title:
Authors:
Program Title: CellMatch
Journal Reference:
Catalogue identifier:
Licensing provisions: none

[1] Corresponding author
[2] On leave of absence from the Rudjer Bošković Institute, Zagreb, Croatia.

Programming language: python
Computer: any architecture with a python interpreter
Operating system: Linux, AIX.
RAM: even for large systems almost negligible usage of memory.
Number of processors used: 1
Supplementary material:
Keywords:
PACS:
Classification:
External routines/libraries:
- *none*

Subprograms used:
*Catalogue identifier of previous version:**
*Journal reference of previous version:**
*Does the new version supersede the previous version?:**

Nature of problem:
Contracting a common supercell that fits the atoms of two unit cells with minimal strain. This is used as input for any total energy or electronic structure code.
Solution method:
Straightforward systematic search in the phase space of combinations of unit cell vectors.
*Reasons for the new version:**

*Summary of revisions:**

Restrictions:

Unusual features:
Output, atomic structure of the supercell, can be used in any total energy program.
Additional comments:

Running time:
Usually very short (seconds) if the search parameters are kept at reasonable values.
References:

[1] Reference 1

[2] Reference 2

[3] Reference 3

* Items marked with an asterisk are only required for new versions of programs previously published in the CPC Program Library.

1 Introduction

The revolution that began with discovery of graphene [1] is not losing its momentum. On the contrary, since then quite a few two dimensional materials have shown potential for applications in electronics [2], optoelectronics [3] etc. Such materials represent a class of so called van der Waals materials [4] whose main characteristic is that they are made of layers of atoms which are held in bulk structure only by weak van der Waals forces (hence the name) which enables relatively easy extraction of a single layer of such material. Layers of such materials are characterized by the absence of dangling bonds since within the layer atoms are bonded together by strong covalent forces. Due to this property they tend to stick dominantly by vdW forces to other materials. Besides the simple Scotch tape method (mechanical exfoliation) for extracting a single layer of material they are also produced in monolayer thickness by chemical vapor deposition CVD [5]. Moreover by molecular beam epitaxy a heterostructures of vdW materials can be grown efficiently [6] with atomically sharp and clean interfaces. When grown by chemical vapor deposition different materials are used as substrates - for example graphene is very often grown on the Ir(111) surface [7], MoS_2 can be grown on a sapphire [8]. In order to study growth of the materials on a substrate by computational method a common supercell of at least 2 different materials (substrate and deposited material) is required. Moreover, such class of materials opens new directions in novel materials design by simply stacking layers of one vdW material onto the other obtaining a so called vdW heterostructures. Additionally such structured can typically be intercalated, or decorated by adsorption giving a rather large phase space for design of novel materials.
In order to study theoretically the smallest possible vdW heteroststructure composed of two vdW materials a common supercell is required.
Of course, in the experimental growth common supercell will be obtained following the energy minimization which will typically be a competition between bonding between the layers of vdW materials and strain since typically such materials will have different lattice constants. Moreover, besides the CVD procedure where some relative orientations of layers appear as preferred in mechanical stacking procedure almost any relative orientation between the layers can be achieved [9]. Due to the intrinsic nature of the vdW materials that they stick to surfaces by weak vdW forces (since within the layers atoms are strongly covalently bound) they tend to grow incommensurately with the substrate while keeping the small strain within the layer.
The purpose of the *CellMatch* code is to create that common supercell for the two given unit cells of different materials. The common supercell should ide-

ally have small strain and small number of atoms, however usually a tradeoff is required between the two. The code performs a brute force search for a common supercell offering different possibilities in which one material is strained to fit onto the other. The possible structures are sorted by the value of strain. Sometimes it is relatively easy to produce a common supercell because of the geometrical similarity of the two given unit cells - for example graphene on Ir(111) which gives a well known moiré 10×10 graphene unit cells over 9×9 Ir(111) unit cells, structure (shown in figure 6). However, usually it is rather difficult to perform this cell matching task manually. In this paper, after theoretical introduction of the method, somewhat mixed with description of running the code, we demonstrate the cell matching procedure on several selected examples.
Since the code is rather simple all available options are explained in detail in the appendix making this paper also a complete manual for the *CellMatch* code.
By recursion the code can be used to create structures with arbitrary number of layers.

1.1 Technical remarks

The files describing input unit cells and the supercell generated by the *CellMatch* code are tailored for the VASP code [10] but it can be used easily with any other code regardless of the concrete implementation (density functional theory (DFT) [11], tight binding [12], empirical, etc.) - since the *CellMatch* only deals with the geometry of the supercell and position of the atoms in it. It is worth mentioning here that current state of the art DFT codes are now mature enough to include van der Waals forces in several ways. In our examples we prefer the selfconsistent implementation of the vdW-DF functional [13–17] as the best cost/benefit option of including vdW interaction from the first principles - i.e. without a single empirical parameter. Alternative route being RPA [18] or semi-empirical methods - such as DFT-D2 [19] might be used increasing the numerical effort or decreasing the accuracy, respectively. With the CPU power of the average cluster and vdW-DF functional approach one can expect to treat supercells with several hundreds atoms. In lower levels of approximation such as tight binding this number can be much higher.
Also, it is worth mentioning here that by chance, recently, dealing with supercells for purpose of studying impurities and alloys the so called bandstructure unfolding procedures were developed. Such procedures enable projection of the bandstructure of the supercell into the Brillouin zones of the original small unit cells giving effective band structure which can be easily compared with the pristine materials bandstructure. The theory [20] and implementation [21] for band unfolding are readily available. In one of the examples we demonstrate band unfolding for the vdW heterostructure consisting of MoS_2 and

graphene.

2 Theoretical background and the description of the code

We explain the general principle of generating a common supercell out of two unit cells by example.
Starting from the two given unit cells shown in figure 1 we search for the common supercell in which atomic structures, defined by original unit cells, can be fitted with smallest possible strain.

We use the example of the graphene and vicinal Ir surface with somewhat larger indices - namely 332. This two cells are hard to fit manually in the common supercell. The unit cells are defined by the files named *POSCAR_GRAPHENE* and *POSCAR_slab_332* (given in the appendix B) and search for common supercell is done by using the python code:

```
> python match_cells.py POSCAR_GRAPHENE POSCAR_slab_332
```

There are several options to the command, such as cell rotation, which are explained in the appendix A. The first unit cell (POSCAR_GRAPHENE) will be strained in the supercell while the second will not.
For the POSCAR files we instruct the user to see VASP manual [10], however the *CellMatch* code does not support all the VASP features (for example negative scaling number will not be recognized as the cell volume).
The unit cells used with the code must be of the slab type having one unit cell vector perpendicular to the plane spanned by the other two. In figure 1 we show both side and top views of the unit cells, while in the rest of the paper mostly the top views are used.
Starting from the in plane unit cell vectors of the two unit cells - namely $\mathbf{a_1}$,$\mathbf{a_2}$ and $\mathbf{b_1}$, $\mathbf{b_2}$ the code first searches for the *common vectors* $\mathbf{v_1}$ and $\mathbf{w_2}$ in the plane within the given tolerance. The common vectors are generated as combinations of unit cell vectors:

$$\begin{aligned} \mathbf{v_1} &= \mathrm{n}_1\mathbf{a_1} + \mathrm{n}_2\mathbf{a_2} \\ \mathbf{w_2} &= \mathrm{m}_1\mathbf{b_1} + \mathrm{m}_2\mathbf{b_2}, \end{aligned}$$

where indices n_1, n_2, m_1, m_2 are covering all possible combinations of integer numbers between $-nindex$ to $nindex$, avoiding the null-vectors. The default value of $nindex$ is 10.
In figure 2 several common vectors generated by such procedure are shown

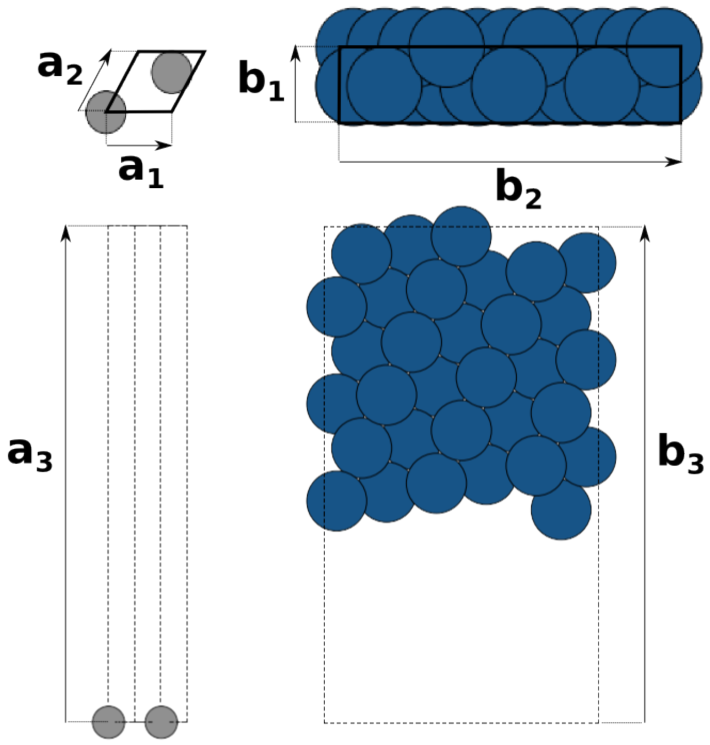

Fig. 1. Unit cell of graphene - left, top and side view. Unit cell of a 332 Ir surface slab - right, top and side view.

– all vectors begin at the common origin and only the tips of the common vectors are shown represented by colored circles.

If the two common vectors generated by different unit cells $\mathbf{v_1}$ and $\mathbf{w_2}$ are close enough – i.e. within the tolerance which is defined as distance between ending points of the vectors divided by vector length (default being 1 percent) - they enter a set of vectors for possible common supercell construction.
Having the set of common vectors all possible combinations of the two sets of common vectors ($\mathbf{v_1}$, $\mathbf{w_1}$), ($\mathbf{v_2}$, $\mathbf{w_2}$) spanning the common supercell are generated. Notice that for every two pairs of common vectors two supercells are generated each one created from its respective unit cell.

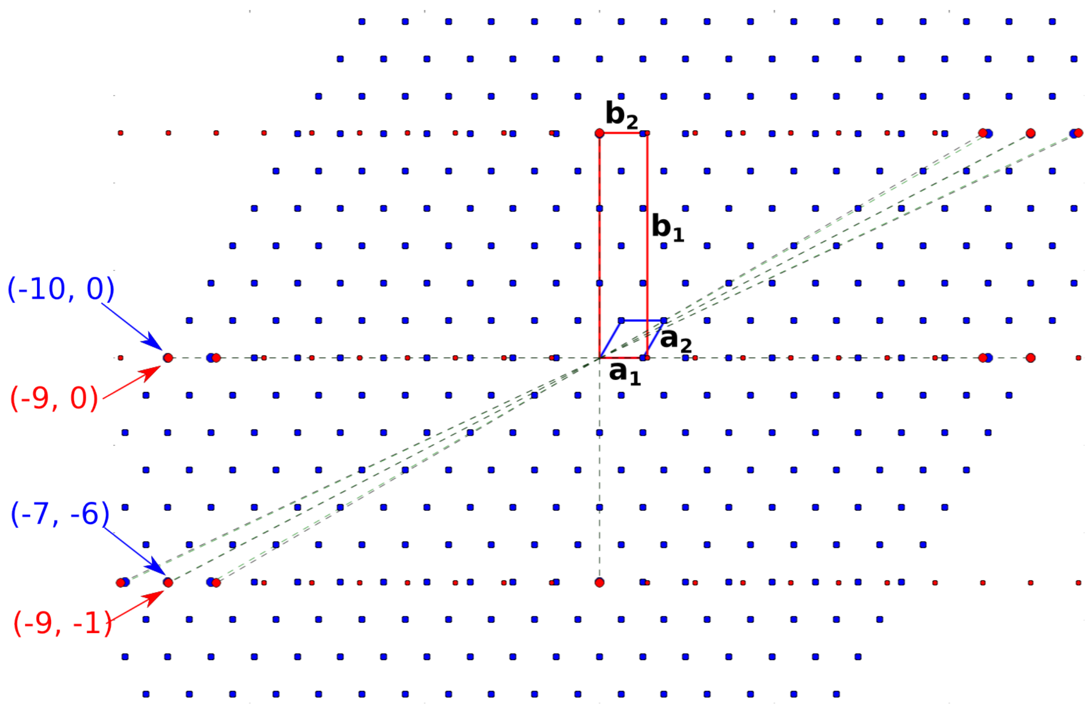

Fig. 2. Example of searching for common supercell vectors. The blue dots represent potential supercell vectors generated from the first unit cell vectors a_1 and a_2, while the red dots represent potential supercell vectors generated from the second unit cell vectors b_1 and b_2. In some cases the supercell vectors obtained from different unit cells are close enough (defined by tolerance) so that they can be considered in search for common supercell with small strain. Such cases are connected to the origin by dashed lines – representing potential supercell vectors. For the case with the smallest strain the indices n_1, n_2, m_1 and m_2 are given according to equation 1.

Possible common vectors are shown in figure 2 as red and blue circles. Common vectors that satisfy tolerance criteria are shown as somewhat larger circles and dashed lines connect them to the origin representing the possible vectors of the common supercell.

All pairs of supercells (one supercell is generated from unit cell 1 and the other from unit cell 2) are then considered for the common supercell which is generated so that one supercell is deformed to fit perfectly the other. The strain due to this modification is calculated. Calculation of the strain is following the standard unit cell strain calculation given in reference [22].

At this point the code has written files *results.dat, POSCAR_slab_332_copy* and *POSCAR_GRAPHENE_rotated_0.0.*

All obtained results are written, sorted by strain in ascending order, into the file *results.dat* - printed here:

results.dat

```
POSCAR_GRAPHENE_rotated_0.0 POSCAR_slab_332_copy
----------------------------------------------------- RESULTS -------------------------------------------
------------------------------------------------------------- -------------------------------------------
|  index  |       strain       |    atoms  |  surf_ratio  |        indices1       |        indices2        |
---------------------------------------------------------------------------------------------------------
|      1  |       0.00081232  |      408  |     60    9  |  -10    0   -9    0  |   -3    6    0    1   |
|      2  |       0.00435935  |      364  |     54    8  |   -3    6    0    1  |   -9    0   -8    0   |
|      3  |       0.06861402  |       44  |      6    1  |   -7   -6   -9   -1  |   -6   -6   -8   -1   |
---------------------------------------------------------------------------------------------------------
```

The coefficients given for the first supercell suggested in results.dat are shown in the figure 2 next to their representative vectors.

The second python code *generate_cell.py*, by default, uses the *results.dat* file to generate the user selected common supercell.

The choice under number 1 from the results has the smallest strain of the scanned options and should be taken as the common supercell. Selection of the result number 1 from the *results.dat* is done by:

```
> python generate_cell.py 1
```

Several options are available for this command - see appendix A. In particular the options for shifting the two cells relative to each other are important since this is only one possible relative position of one layer with respect to the other. To establish the energetically most favorable relative positions its phase space has to be thoroughly explored. The same is true for relative rotations between the layers – rotation being the option for the first python code that searches for the common supercell. The second code generates the actual supercell based on the results of the first python code. The second code repeats periodically the atoms from the original (input) unit cells in order to appropriately fill the supercell. This is rather straightforward and we do not describe the procedure here.
The final output file with the generated supercell will be named *POSCAR_COMMON_CELL* (given in the appendix A) and it can be used as input in electronic structure/total energy codes. For VASP no modifications should be needed and for other codes the user has to rearrange data to fit the given code.
The obtained common supercell is shown in figure 3.

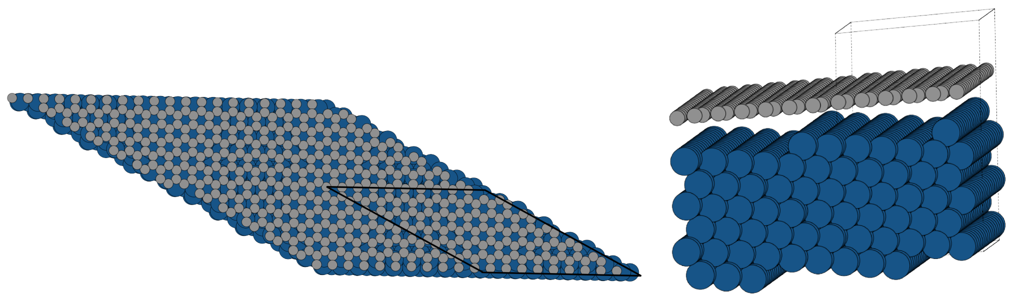

Fig. 3. Common supercell of graphene and Ir(332) slab, top (left) and side (right) view.

Notice that in this example it might be tempting to select the choice number 3 from the results which contains only 44 atoms in the supercell. Generating such supercell will give the structure shown in figure 4 in which graphene is obviously severely distorted.

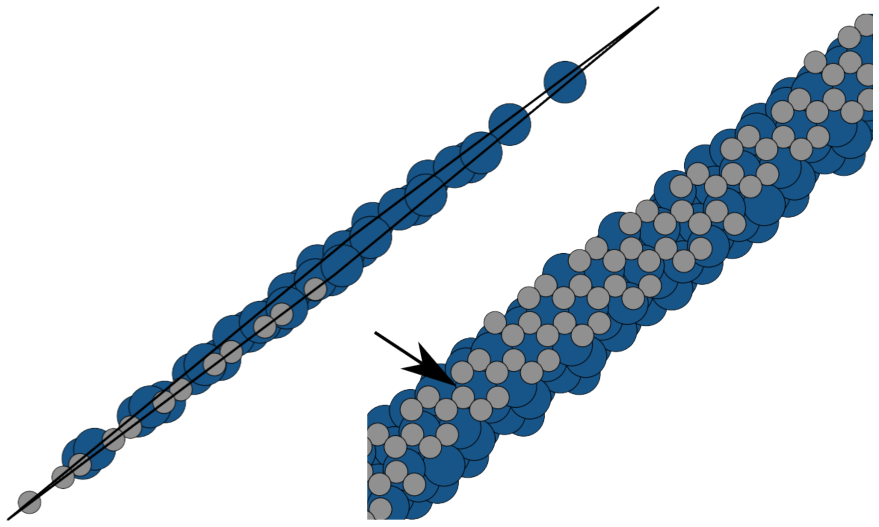

Fig. 4. Common supercell of graphene and Ir(332) slab with larger strain than the one shown in figure 3 - notice large distortion in graphene structure marked by arrow.

This is the consequence of the definition of the tolerance when searching for the common vectors and should be checked. Elongated cells satisfy tolerance criteria more easily since the supercell vectors are long. Due to the definition used for the common points it is not guaranteed that the angle between atoms in the supercell will be preserved - which typically costs a lots of energy that is not visible in the calculated "*geometrical strain*" value. The user should pay attention to this before selecting the structure with relatively small amount of atoms and seemingly small strain.

The working of the code described above is shown as flowchart in Figure 5:

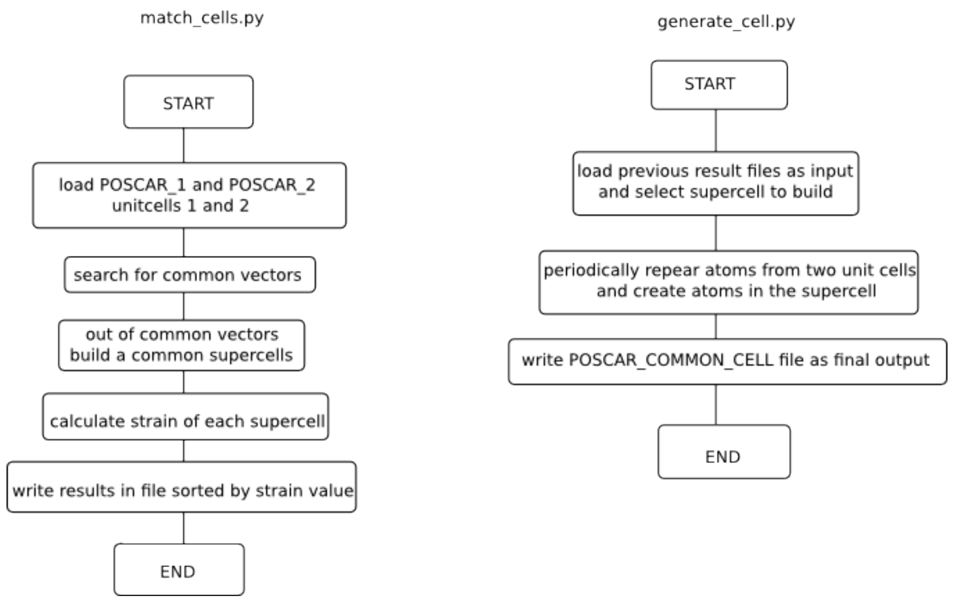
match_cells.py
START
load POSCAR_1 and POSCAR_2 unitcells 1 and 2
search for common vectors
out of common vectors build a common supercells
calculate strain of each supercell
write results in file sorted by strain value
END
generate_cell.py
START
load previous result files as input and select supercell to build
periodically repear atoms from two unit cells and create atoms in the supercell
write POSCAR_COMMON_CELL file as final output
END

Fig. 5. Flowcharts of the two python codes - match_cells.py - left and generate_cell.py - right.

3 Installation and running the code

The code consists of two python scripts (*match_cells.py* and *generate_cell.py*) and requires no installation, compilation or special python packages out of the standard Linux distributions. It is distributed as a gzipped tar file containing the two python scripts and 2 directories with examples.
Running of the code is somewhat described in the theory part and we repeat it here briefly. Having the POSCAR files of the selected input unit cells - POSCAR_1 and POSCAR_2 the user first runs the first script:

```
> python match_cells.py POSCAR_1 POSCAR_2
```

obtaining the output file *results.dat* in which typically several options are offered for generating a common supercell - given under the index numbers.

Selecting the supercell with index 1 is done by a command:

```
> python generate_cell.py 1
```

which produces a common supercell file named *POSCAR_COMMON_CELL*. This ends the program run. Several options to the both python codes are available and are described in detail in the appendix A.

4 Examples

As examples we have selected three systems. In the first two examples we shall only demonstrate generating a supercell while for the third one we perform a DFT calculations yielding common structure relaxed geometry, binding energy, strain energy and electronic structure.
The examples are chosen following some important challenges from vdW materials, namely: their growth by CVD, their contact with metals for electronic and catalytic applications and the most general creation of the two layer vdW heterostructure.
The first example of graphene on Ir(111) surface is a typical system in CVD growth of graphene, relatively popular for having graphene with almost unperturbed electronic structure compared to the free standing graphene due to its very weak interaction with the iridium substrate. Also this system has a very famous (10×10 over (9×9) moiré structure nicely visible in the STM experiments [23].
The second example follows not so widespread research direction of vicinal surfaces with somewhat larger indices - namely the 331 surface of the iridium is matched again to the graphene. The idea that metal can modify electronic structure of the 2D materials attached to it [24] and be used for tailoring catalytic properties can be pushed even further using the vicinal surfaces which on its steps show exotic and interesting electronic properties [25]. The vicinal surfaces with indices larger than 1 are not so popular experimentally nor computationally. Regarding computation we believe that part of the problem lies in the fact that such surfaces itself were somewhat technically challenging to generate. Until recently there was no available code for preparing the unit cell of arbitrary vicinal surface. The recent work of Sun and Ceder [26] published such code that we used in generating both 332 (used in theory part of the paper) and 331 surface of iridium used in examples.

For the final example we select a real vdW heterostructure, a type of a system which is probably the main target for the *CellMatch* code. We chose graphene and MoS_2 heterostructure. Very recent development [**?**] shows a tremendous potential in creating such surfaces experimentally and obtaining different physical properties - moreover, making very concrete devices based on them.

4.1 Graphene on Ir(111)

The system of graphene on Ir(111) is very well studied both experimentally and computationally [23] and under the STM a very nice moiré pattern is observed. Feeding the unit cells of the Ir(111) surface and graphene to the *CellMatch* code we indeed obtain that structure as the best suggestion - shown in figure 6. Notice that default *nindex* parameter is just barely large enough for the code to find this structure. If the moiré structure that is searched for has one of the indexes larger than 10 one has to increase *nindex* value so that the *CellMatch* discovers such structure. The lattice mismatch between graphene and Ir(111) is around 10 percent. If one wants *CellMatch* code to generate a simple 1×1 unit cell in which either graphene or Ir(111) will be strained by ± 10 percent - it can be achieved by increasing the linear tolerance value to around 0.1 (`--tolerance` option). At the same time *nindex* value should be reduced to 1, otherwise with default value of 10 and tolerance of 0.1 the number of combinations of common vectors would be enormous (2070 common vectors) and the calculations of common supercells can take a very long time. Also if the tolerance of 0.1 is given together with the *nindex* value of 3 the code actually finds many equivalent possibilities - but always writes only the one with the smallest possible number of atoms - to see all the `--unique 0` option should be used - see appendix A.

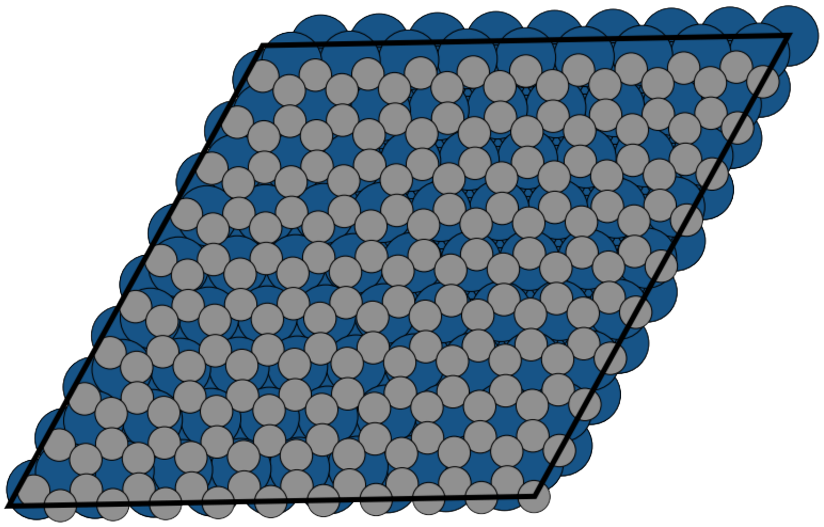

Fig. 6. Common supercell for graphene and Ir(111). The well known moiré 10×10 over 9×9 structure is obtained.

4.2 Graphene on Ir(331)

Graphene on Ir(331) is chosen to demonstrate the interesting field of vicinal surfaces. We have used the recently published code by Sun and Ceder [26] to generate the 331 surface slab. Matching it with a graphene unit cell we obtain a structure shown in figure 7.

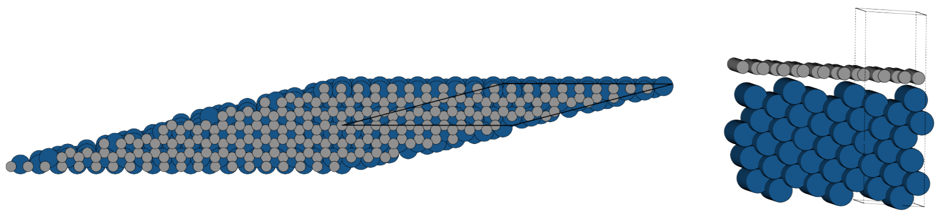

Fig. 7. Common supercell for graphene and Ir(331) - top (left) and side (right) view, obtained with default value of *nindex* = 10.

Notice that in this cell because of the reasons explained at the end of theoretical background section the angles between graphene atoms deviate form the ideal ones in graphene. Because of this we give it another run with `--nindex 15` option which yields a larger supercell with much smaller strain shown in figure 8.

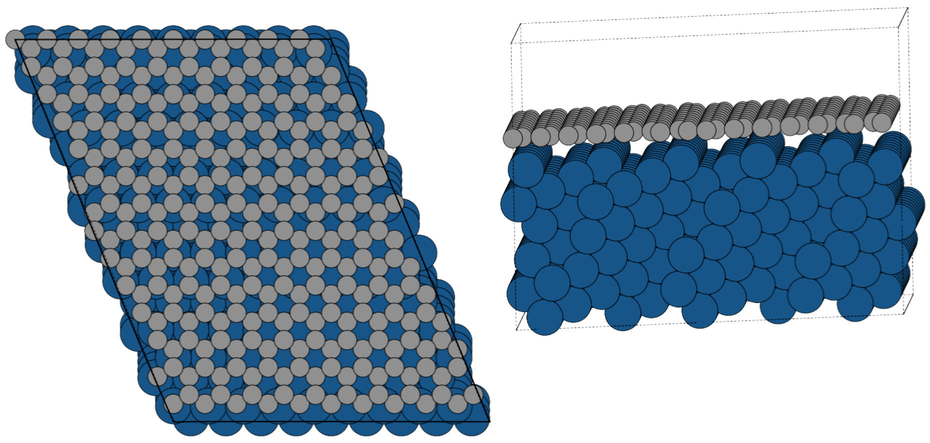

Fig. 8. Common supercell for graphene and Ir(331) - top (left) and side (right) view, obtained with custom value of *nindex* = 15. This structure has smaller strain than the one shown in figure 7.

The acceptable amount of strain we leave to the user - however the physical reasoning behind the strain value is that it has to be smaller than the characteristic binding energy between the layers so that the best configuration can be determined. In the above example the smaller supercell cell yields a strain of 60 meV per carbon atom which is exactly the value of the average binding energy per carbon atom in the system.
Therefore the larger cell has to be used where the strain in graphene is almost zero. For subtle effects of cell alignment and comparable strain and binding energies see the example of MoS_2 on sapphire in reference [27].

4.3 Graphene on MoS_2

Finally, we select the example of the real vdW heterostructure of MoS_2 and graphene. MoS_2 has the bandgap and is probably second most popular vdW material after graphene. Some very promising demonstrations of its usage have been done in the past five years - the demonstration of a single layer transistor probably being the most famous one [2]. After selecting a common supercell shown in figure **??** we perform a DFT calculation with selfconsistently implemented vdW-DF functional in VASP code to obtain the relaxed structure. In the relaxed structure both graphene and MoS_2 remain flat with the distance between them of 3.45 Å(distance between graphene and closer sulfur atoms plane).

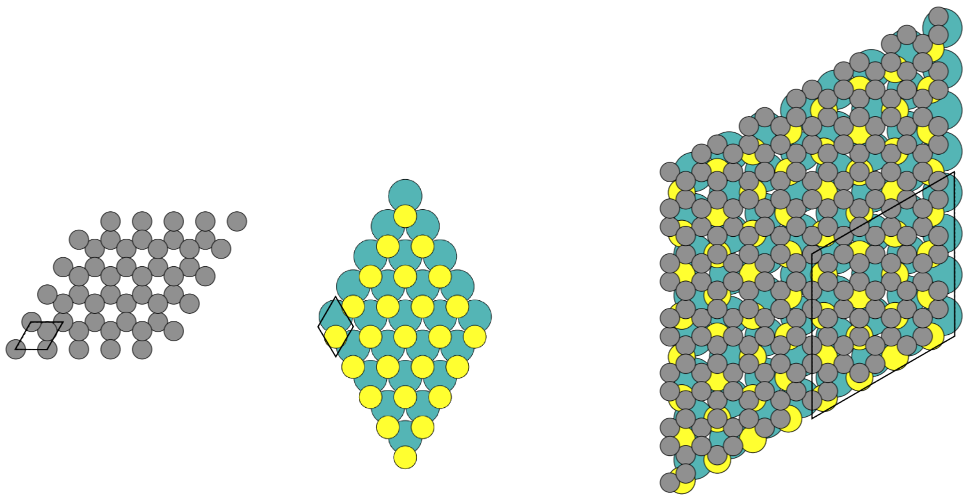

Fig. 9. Graphene unit cell – left. MoS_2 unit cell – middle. Common supercell for graphene and MoS_2 – right.

Such distance immediately signals the dominant vdW character of binding (i.e. weak binding) which is confirmed by calculated binding energy of 63 meV per C atom. However, drawing the charge transfer shown in figure 10 shows that some small amount of charge transfer occurs. Calculating the purely nonlocal binding energy density, visualized in figure 11, we establish that the pure nonlocal binding energy is actually 90 meV per carbon atom leading to conclusion that the charge rearrangement is actually of repulsive character. This is not unexpected, it is rather expected, as a signature of a dominant vdW bonding. The charge transfer is not completely absent in the dominantly physisorbed systems (held together by vdW interaction) it appears and serves as the counterbalance to the attractive vdW forces which would otherwise bring together two layers to complete overlap. The repulsive character of the charge transfer determines the equilibrium distance between the layers. This is shown in detail for the graphene on Ir(111) in reference [23] and it is known as a push back or pillow effect [28]. Calculating the strain value on the graphene by comparison with the unstrained geometry of the graphene supercell we obtain a small value of 10 meV per C atom. Based on the calculated strain and binding energies we can not claim that the obtained structure is the most stable one but we can definitely claim that it is stable. Of course there are other two parameters that we did not cover - relative shift between graphene and MoS_2 and the relative orientation between the layers.

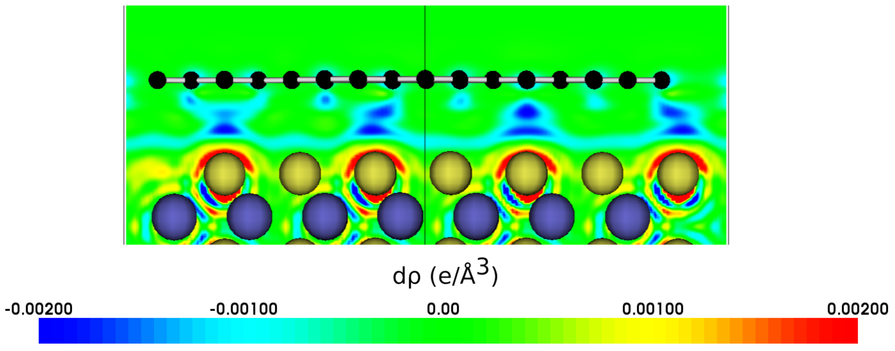

Fig. 10. Charge density difference for graphene on MoS_2.

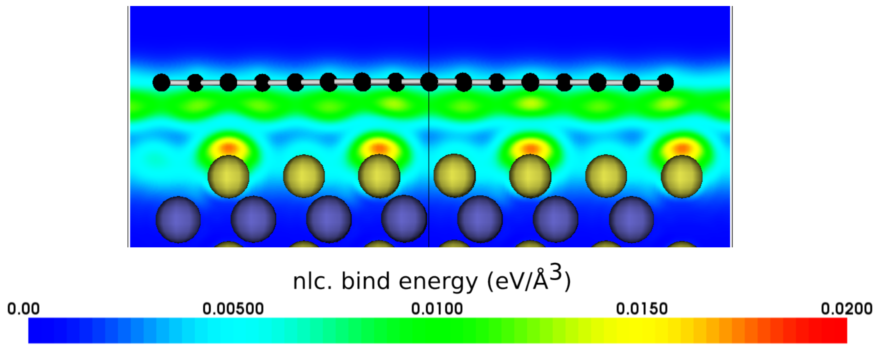

Fig. 11. Nonlocal binding energy density for graphene on MoS_2. For definition see [].

It might seem that relative shift or orientation plays a minor role in dominantly vdW bound layers since vdW interaction is so dispersed - but the truth is actually completely different as is shown in the reference [27].
Having the atomic structure relaxed one can explore its electronic structure by

looking at the density of states for example, but we decide to show the effective bandstructure along high symmetry lines of the original (small) unit cells. Since the interaction between the two layers is rather weak and is expected to be weak in almost any vdW heterostructure, we expect the bandstructure of the MoS_2/Graphene to contain recognizable features from the bandstructure of it components. However it is difficult to compare the bandstructure directly from the supercell with the one from the small unit cell of MoS_2 or graphene. For this purpose we use the unfolding procedure for which the theory is given in [20] and implementation from reference [21] is employed here through the BandUp code [29]. After the unfolding procedure we obtain the bandstructure in the Brillouin zone of the MoS_2 unit cell where it can be directly compared with the original MoS_2 bandstructure - both shown in figure 12.

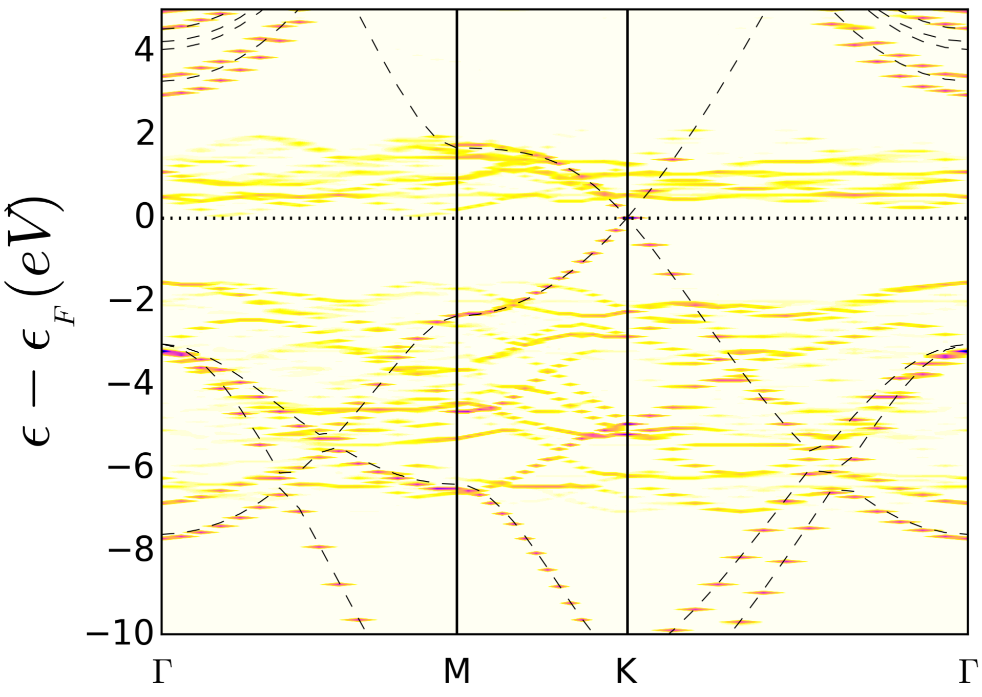

Fig. 12. Unfolded bandstructures from the graphene/MoS_2 supercell into the small graphene unit cell. Dashed lines represent pure graphene bandstructure.

From figure 13 it can be seen that in spite of the small charge transfer the MoS_2 lines have been shifted a lot in the energy – more precisely they are shifted by the gap value of the MoS_2. Originally fermi level in the MoS_2 lies at the top of its valence band maximum (VBM) and the smallest charge accumulation in the MoS_2 will shift it to the conduction band minimum (CBM) – which is exactly what is shown in figure 13. By shifting the original MoS_2 bands by -1.55 eV – dashed lines, we can conclude that such doping of MoS_2 follows closely the rigid band approximation, which can not be determined a–priori.
In figure 12 we show the unfolded bandstructure but this time with respect to the unit cell of graphene, so that graphene features are recognized easily.

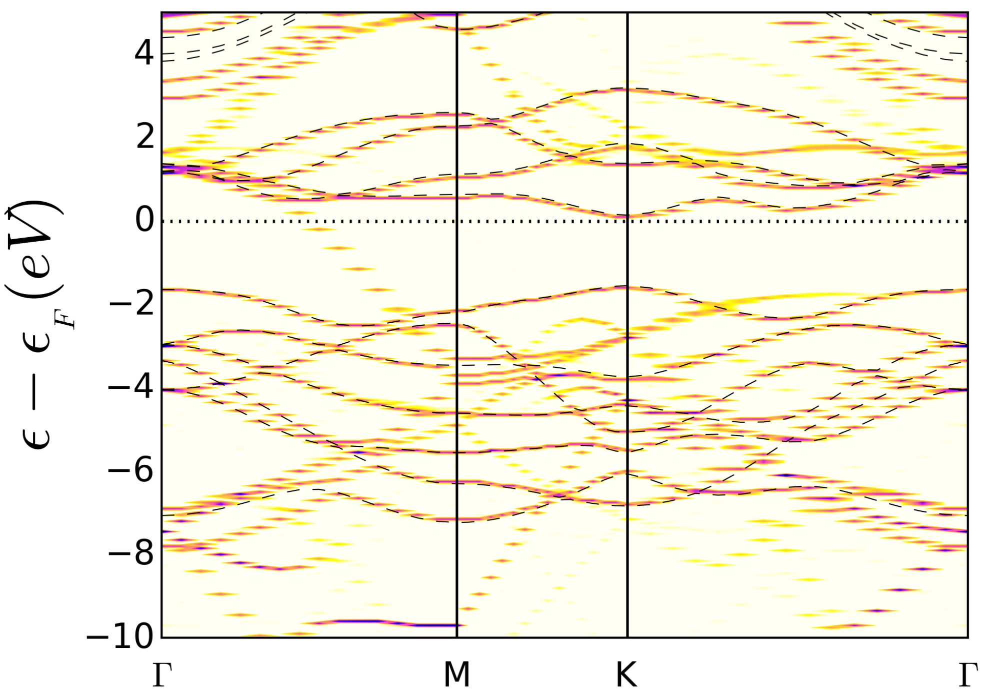

Fig. 13. Unfolded bandstructures from the graphene/MoS_2 supercell into the small MoS_2 unit cell. Dashed lines represent pure MoS_2 bandstructure – shifted by −1.55 eV.

In all given examples once the common supercell is prepared it can be used to further study modifications of the heterostructure (novel material) properties through intercalation, adsorption, gating, strain, etc.

5 Conclusions

We have demonstrated a novel code which can be used as a handy tool in creating various layered type heterostructures, such as found in epitaxial growth and in particular in the emerging field of vdW heterostructures.
The *CellMatch* is a universal tool which generates atomic structures regardless of the computational method that will be applied on them afterwards. Since it only systematically searches through a phase space of possible combinations of the two unit cells combinations it has no convergence issues.
In principle the code generates one common supercell out of two unit cells, but used recursively one can keep adding more layers to the structure. The developed *CellMatch* code is very general and it represents a valuable tool not only for the *ab initio* computational community but also for anyone studying vdW heterostructures or epitaxial growth by other means.

6 Acknowledgements

P. L. was supported by U.S. ONR Grant N000141310754 and the UB Center for Computational Research.

7 Appendix A. code options

For match_cells.py

The list of available options is:

`--nindex` nind
Integer number *nind* for search of common unit cell vectors, the values in code go from -nind to +nind, the default value is 10.

`--tolerance` tol
Tolerance – float value *tol* – for detecting that two supercell vectors constructed from different unit cells are considered identical. Defined as length of the vector difference divided by vector length. Default value 0.01.

`--rotate` rot
Rotation angle *rot* (in degrees) for the first unit cell. Default value is 0. The first unit cell (POSCAR_1) will be copied into a file POSCAR_1_rotated_0.0.

`--maxatoms` mat
Integer number *mat* for maximum number of atoms in the supercell, if the supercell with larger number of atoms is found it will not be printed - default value -1 - meaning that maximum number of atoms is infinite.

`--maxstrain` mstr
Float value *mstr* for maximum strain in the supercell, supercells with larger strain value will not be printed - default value -1 - meaning that maximum strain is infinite.

`--unique` 0/1
Unique switch 0 or 1, default value is 1. In case of 0 value all possible realizations of the equivalent supercells will be printed out.

`--output` outfile
Name of the output file *outfile*. Default value is *results.dat*, this file is used by generate_cell.py to create common supercell.

`--show_progress` 0/1
Show progress switch 0 or 1. Default value is 1 and the progress is printed on the screen. For example in high-throughput applications running on cluster etc. it might be convenient to suppress the on screen output.

For generate_cell.py

`--input_file` infile
Input file name $infile$ from which the parameters are read to generate the common supercell. The default value is $results.dat$.

`--tolerance` tol
Tolerance integer value tol used in determining repeating periodic cells in order to generate atoms for the common supercell. This parameter with default value of 1 should not be changed. If it happens that some atoms are missing in the final supercell - please report, and try to increase the value.

`--tolerance_float` tolf
Tolerance float value $tolf$ used to determine which atoms obtained by repeating the original (small) unit cells lie within a new common supercell. This number as well as tolerance should not be changed. In case that some atoms are missing try to increase the value, while in case that there are too many atoms in the supercell decrease the value. Default value is $1e-4$ - and please report such cases.

`--shift11 --shift12 --shift13 --shift21 --shift22 --shift23` shift
Shifts along the unit cell vectors for the two original unit cells. For example $--shift110.5$ option will shift the atoms of the first unit cell by 0.5 a_1 unit cell vector.

`--shift1x --shift1y --shift1z --shift2x --shift2y --shift2z` shift
Shifts along the Cartesian axes for the two original unit cells. For example $--shift1x3.5$ option will shift the atoms of the first unit cell by 3.5 Å along the x–axis.

`--output` outfile
The name of the output file. Default value is $POSCAR_COMMON_CELL$.

`--zfix` zf
The value zf in Å of the z–coordinate below which the atoms of the common super cell will be fixed for VASP selective dynamics. The option "F F F" will be written. In the case of negative value zf the atoms above the $-zf$ coordinate will be fixed.

`--draw` 0/1
The draw switch 0 or 1. In case of value of 1 the code will draw a supercell

generated from unit cell 1 and unit cell 2 - the mismatch will be visible to estimate the strain. Original unit cells are drawn as well.

8 Appendix B. sample files

POSCAR_slab_332

```
Ir.10 (3,3,2) static
1.0
2.740553 0.000000 0.000000
0.000000 12.854334 0.000000
0.000000 0.000000 23.220926
Ir
32
direct
0.000000 0.316932 0.964700 Ir
0.000000 0.680568 0.608854 Ir
0.000000 0.407841 0.680023 Ir
0.000000 0.589659 0.502100 Ir
0.000000 0.862386 0.430931 Ir
0.000000 0.953295 0.537685 Ir
0.000000 0.226023 0.466515 Ir
0.000000 0.044204 0.644438 Ir
0.000000 0.316932 0.573269 Ir
0.000000 0.589659 0.893531 Ir
0.000000 0.953295 0.929115 Ir
0.000000 0.226022 0.857946 Ir
0.000000 0.498750 0.786777 Ir
0.000000 0.771477 0.715608 Ir
0.000000 0.862386 0.822362 Ir
0.000000 0.135113 0.751192 Ir
0.500000 0.680568 0.804569 Ir
0.500000 0.589659 0.697815 Ir
0.500000 0.771477 0.519892 Ir
0.500000 0.226022 0.662231 Ir
0.500000 0.498750 0.591062 Ir
0.500000 0.135113 0.555477 Ir
0.500000 0.407840 0.484308 Ir
0.500000 0.862386 0.626646 Ir
0.500000 0.044204 0.448723 Ir
0.500000 0.953295 0.733400 Ir
0.500000 0.771477 0.911323 Ir
```

```
0.500000 0.316931 0.768985 Ir
0.500000 0.044204 0.840154 Ir
0.500000 0.407840 0.875739 Ir
0.500000 0.135113 0.946908 Ir
0.500000 0.498750 0.982492 Ir
```

POSCAR_GRAPHENE

```
C
1.0000000000000000
2.4680008987391253 0.0000000000000000 0.0000000000000000
1.2340004493695627 2.1373514748709082 0.0000000000000000
0.0000000000000000 0.0000000000000000 23.220926
C
2
Cartesian
0.0000000000000000 0.0000000000000000 0.0000000000000000
2.4680008987391164 1.4249009832472670 0.0000000000000000
```

POSCAR_COMMON_CELL

```
C Ir.10 (3,3,2) static
1.00000000000000
-24.6649769999999968 0.0000000000000000 0.0000000000000000
-24.6649769999999968 -12.8543339999999997 0.0000000000000000
0.0000000000000000 0.0000000000000000 23.2209259999999986
C Ir
120 288
Direct
1.0000000000000000 1.0000000000000000 0.2583876284692522
0.8999999999999999 1.0000000000000000 0.2583876284692522
0.7999999999999998 1.0000000000000000 0.2583876284692522
0.9166666666666667 0.8333333333333333 0.2583876284692522
0.7000000000000001 1.0000000000000000 0.2583876284692522
0.8166666666666669 0.8333333333333333 0.2583876284692522
0.9333333333333333 0.6666666666666666 0.2583876284692522
0.6000000000000000 1.0000000000000000 0.2583876284692522
0.7166666666666668 0.8333333333333333 0.2583876284692522
0.8333333333333333 0.6666666666666666 0.2583876284692522
0.9500000000000000 0.5000000000000000 0.2583876284692522
.
.
```

```
.
.. (for full listing see example files)
```

References

[1] K. S. Novoselov, A. K . Geim, S. V. Morosov, D. Jiang, Y. Zhang, S. V. Dubonos, I. V. Grigorieva, A. A. Firsov, Science **306**, 666 (2004).

[2] B. Radisavljevic, A. Radenovic, J. Brivio, V. Giacometti, A. Kis, Nat. Nanotechnol. **6**, 147 (2011).

[3] A. Ramasubramaniam, Phys. Rev. B **86**, 115409 (2012).

[4] A. K. Geim and I. V. Grigorieva, *Van der Waals heterostructures*, Nature 499, 419 (2013).

[5] S. Choon-Ming, C. Siang-Piao, M. A. Rahman, Carbon **70**, 1 (2014).

[6] F. Withers, O. Del Pozo-Zamudio, A. Mishchenko, A. P. Rooney, A. Gholinia, K. Watanabe, T. Taniguchi, S. J. Haigh, A. K. Geim, A. I. Tartakovskii, K. S. Novoselov, arXiv:1412.7621

[7] M. Petrović, I. Šrut Rakić, S. Runte, C. Busse, J. T. Sadowski, P. Lazić, I. Pletikosić, Z-H. Pan, M. Milun, P. Pervan, N. Atodiresei, R. Brako, D. Šokčević, T. Valla, T. Michely, M. Kralj, Nat. Commun. **4**, 2772 (2013).

[8] S. Entani, L. Y. Antipina, P. V. Avramov, M. Ohtomo, Y. Matsumoto, N. Hirao, I. Shimoyama, H. Naramoto, Y. Baba, P. B. Sorokin, S. Sakai, Nano Res., 10.1007/s12274-014-0640-7 (2014).

[9] L. Meng, R. Wu, L. Zhang, L. Li, S. Du, Y. Wang, H-J. Gao, J. Phys.: Condens. Matter **24**, 314214 (2012).

[10] G. Kresse and J. Furthmüller, Phys. Rev. B **54** 11169, (1996).

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

[12] N. W. Ashcroft,N. D. Mermin, Solid State Physics (Thomson Learning, Toronto, 1976).

[13] M. Dion, H. Rydberg, E. Schröder, D. C. Langreth, and B. I. Lundqvist, Phys. Rev. Lett. **92** 246401, (2004).

[14] J. Klimeš, D. R. Bowler, and A. Michelides, J. Phys.: Condens. Matter, **22**, 022201 (2010).

[15] J. Klimeš, D. R. Bowler, and A. Michelides, Phys. Rev. B **83**, 195131 (2011).

[16] G. Román-Pérez, and J. M. Soler, Phys. Rev. Lett. **103**, 096102, (2009).

[17] K. Berland, V. R. Cooper,K. Lee, E. Schröder, T. Thonhauser, P. Hyldgaard, and B. I. Lundqvist, Rep. Prog. Phys. **78**, 066501 (2015).

[18] J. Harl, G. Kresse, Phys. Rev. Lett. **103**, 056401 (2009).

[19] S. Grimme, J. Comput. Chem. **27**, 1787 (2006).

[20] V. Popescu, A. Zunger, Phys. Rev. B, **85**, 085201 (2012).

[21] P. V. C. Medeiros, S. Stafström, J. Björk, Phys. Rev. B **89**, 041407(R) (2014).

[22] http://www.cryst.ehu.es/cryst/strain.html

[23] C. Busse, P. Lazić, R. Djemour, J. Coraux, T. Gerber, N. Atodiresei, V. Caciuc, R. Brako, S. Blügel, J. Zegenhagen, T. Michely, Phys. Rev. Lett. **107**, 036101 (2011).

[24] W. Chen, E. J. G. Santos, W. Zhu, E. Kaxiras, Z. Zhang, Nano Lett. **13**, 509 (2013).

[25] O. P. Polyakov, O. V. Stepanyuk, A. M. Saletsky, V. S. Stepanyuk, J. Phys.: Condens. Matter **26**, 445005 (2014).

[26] W. Sun, G. Ceder, Surf. Sci. **617**, 53 (2013).

[27] D. Dumenco, D. Ovchinnikov, K. Marinov, P. Lazić, M. Gibertini, N. Marzari, O. Lopez-Sanchez, D. Krasnozhon, M.-W. Chen, P. Gillet, A. Fontcuberta i Moral, A. Radenovic, and A. Kis, ACS Nano **9**, 4611 (2015).

[28] Vázquez, Y. J. Dappe, J. Ortega, and F. Flores, J. Chem. Phys. **126**, 144703 (2007).

[29] https://www.ifm.liu.se/theomod/compphys/band-unfolding/