USER GUIDE

for

CoFFEE

Corrections For Formation Energy and Eigenvalues for charged defect simulations

Contents

1 Outline

CoFFEE is a complete electrostatic corrections package, applicable to charged defects in materials ranging bulk solids, interfaces, surfaces/slabs, two-dimensional (2D) materials, nanowires and nanoribbons. We implement a generalized Poisson solver based on the FNV correction scheme with a gaussian model charge distribution. The code is written entirely in Python. We use Message Passing Interface (MPI) to parallelize the code and Cython to accelerate slow steps. Our implementation can be used alongside any DFT package to obtain an *a posteriori* correction for the formation energy and the defect level position in the gap, for the charged defect being simulated. The code is released under the BSD licence.

For more details on the theory and the equations being solved, please refer to our paper: https://arxiv.org/abs/1705.01491

The formation energy of a defect in charge state q is given by:

$$\mathrm{E}_q^f[\vec{\mathrm{R}}_q](\epsilon_F) = \{\mathrm{E}_q^{\mathrm{tot}}[\vec{\mathrm{R}}_q] + \mathrm{E}_q^{\mathrm{corr}}\} - \mathrm{E}_{\mathrm{pristine}} + q\{\epsilon_{\mathrm{vbm}}^{\mathrm{pristine}} + \epsilon_F - \Delta V_{0/p}\} - n_x\mu_x \tag{1.1}$$

The formation energy is a function of the Fermi level in the system, ϵ_F, with respect to the pristine valence band maximum (VBM), $\epsilon_{\mathrm{vbm}}^{\mathrm{pristine}}$. The first term on the right hand side is the total energy of a system containing a defect in charge state q. $\mathrm{E}_{\mathrm{pristine}}$ represents the total energy of a pristine super cell of the same size. n_x refers to the number of atoms of type x added (positive) or removed (negative) from the pristine system, and μ_x is the atom's chemical potential. $\mathrm{E}_q^{\mathrm{corr}}$ is the finite-size electrostatic correction term. In the FNV scheme, this term is given by:

$$\mathrm{E}_q^{\mathrm{corr}} = \mathrm{E}_q^{\mathrm{lat}} - q\Delta V_{q-0/m} \tag{1.2}$$

where $\mathrm{E}_q^{\mathrm{lat}}$ is obtained from a model calculation.

$$\mathrm{E}_q^{\mathrm{lat}} = \mathrm{E}_q^{\mathrm{iso,m}} - \mathrm{E}_q^{\mathrm{per,m}} \tag{1.3}$$

$\Delta V_{q-0/m}$ is a potential alignment term found by comparing the model potential to the DFT difference potential:

$$\Delta V_{q-0/m} = (V_q^{\mathrm{DFT}} - V_0^{\mathrm{DFT}})|_{\mathrm{far}} - V_q^{\mathrm{per,m}}|_{\mathrm{far}} \tag{1.4}$$

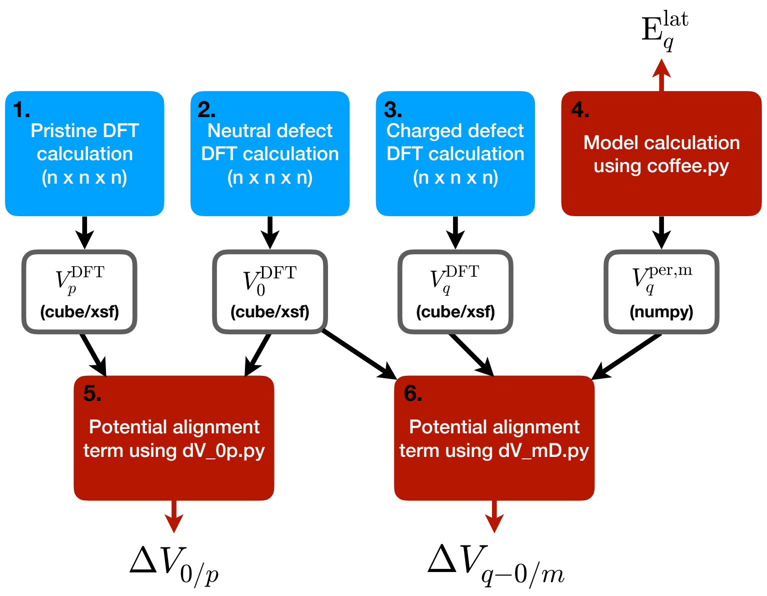
$\Delta V_{0/p}$ is another potential alignment term, found by comparing the electrostatic potentials from a pristine calculation and far from the defect in a neutral defect calculation:

$$\Delta V_{0/p} = V_0|_{\mathrm{far}} - V_p \tag{1.5}$$

The general steps involved in computing corrections for the formation energy of a charged defect are the following (Fig. 1.1):

1. Compute the total energy of the pristine super cell of the same size. Save the DFT potential in cube/xsf format.
2. Compute the total energy of the super cell (say n×n×n) containing the neutral defect. Save the DFT potential in cube/xsf format.
3. Compute the total energy of the super cell containing the charged defect. Save the DFT potential in cube/xsf format.
4. Compute $\mathrm{E}_q^{\mathrm{lat}}$ term: Compute the model energy for various super cell sizes and extrapolate to obtain $\mathrm{E}_q^{\mathrm{iso,m}}$. $\mathrm{E}_q^{\mathrm{lat}}$ is then given by: $\mathrm{E}_q^{\mathrm{iso,m}} - \mathrm{E}_q^{\mathrm{per,m}}$ (n×n×n).
5. Compute the potential alignment term $\Delta V_{0/p}$, Eqn. 1.5. The utility script *dV_0p.py* can be used to compute this.
6. Compute the potential alignment term $\Delta V_{q-0/m}$, Eqn. 1.4. The utility *dV_mD.py* can be used to compute this.

The first three steps are performed by the user with the DFT electronic structure code of his/her choice. The xsf/cube file formats are commonly used to visualize data. Most electronic structure codes provide utilities to convert the DFT potential after a self consistent calculation into these formats. These formats act as an interface between the DFT calculation and the CoFFEE code. The CoFFEE code aids in computing steps 4, 5 and 6. The model



$E_q^{\rm lat}$
1. Pristine DFT calculation (n x n x n)
2. Neutral defect DFT calculation (n x n x n)
3. Charged defect DFT calculation (n x n x n)
4. Model calculation using coffee.py
$V_p^{\rm DFT}$ (cube/xsf)
$V_0^{\rm DFT}$ (cube/xsf)
$V_q^{\rm DFT}$ (cube/xsf)
$V_q^{\rm per,m}$ (numpy)
5. Potential alignment term using dV_0p.py
6. Potential alignment term using dV_mD.py
$\Delta V_{0/p}$
$\Delta V_{q-0/m}$

Figure 1.1: (Color online) The workflow involved in computing the formation energy of a charged defect. The blue boxed are performed using a DFT electronic structure code. CoFFEE aids in computing the corrections, the red boxes. (n×n×n) refers to the super cell size.

calculations are performed by solving the Poisson equation, as detailed in the CoFFEE paper. The next section describes the installation procedure. Sections 3, 4 and 5 describe the workflow involved in computing the model energy, potential alignment terms and running the examples.

2 Installation and prerequisites

The code is obtainable for download from:
`http://www.physics.iisc.ernet.in/~mjain/pages/software.html`

The code requires python with the following libraries installed:

1. SciPy, NumPy, Matplotlib
   In Linux and Mac, run the following command:

   ```
   $pip install numpy scipy matplotlib ipython jupyter pandas sympy nose
   ```

2. mpi4py
   In Linux and Mac, run the following command:

   ```
   $pip install mpi4py
   ```

   or for Linux OS:

   ```
   $apt-get install python-mpi4py
   ```

   For Anaconda users on either OS:

   ```
   $conda install -c anaconda mpi4py
   ```

To install the code in Linux or Mac operating system, first download the tar zip file and extract using (xxx refers to the version):

```
$tar -zxvf coffee_xxx.tar.gz
```

Enter the CoFFEE_xxx folder created and run the following command:

```
  $cd CoFFEE_xxx
  $python setup.py build_ext -b PoissonSolver/
```

This checks for the dependencies and compiles the C routines in the folders:
PoissonSolver/MV_2D_cy/,
PoissonSolver/MV_1D_cy/ and
PoissonSolver/PS_3D_cy/

If you have Cython intalled, this command will first Cythonize the .pyx files present in these directories, and then compile the .c file generated.

On compiling the C code, this creates the following .so files in the PoissonSolver/ folder:
matvec1D.so
matvec2D.so
ps3D.so
(In case these file names have changed for some reason, you would have to rename them to the above.)

If you wish to recompile, first run clean.py to remove the .so files.

3 Model calculations

Once installed, the coffee.py file in the CoFFEE/ directory acts as your main executable. The type of system and other input parameters are supplied via an input file.
This section describes the input files and available options for 3D (bulk), 2D (slab/2D material) and 1D (nanowires/nanoribbons) systems. The input file is divided into three sections designated with
&CELL_PARAMETERS
&DIELECTRIC_PARAMETERS
&GAUSSIAN_PARAMETERS
Each section is ended with a "/".

The CELL_PARAMETERS and GAUSSIAN_PARAMETERS are common to all the systems.

The CELL_PARAMETERS section contains the normalized lattice vectors, the cell dimensions and the plane wave cut off for the model calculation (generally much smaller than the DFT wavefunction cut off)
Example:

```
&CELL_PARAMETERS
#Normalized lattice vectors: a1, a2 and a3
Lattice_Vectors(normalized):
```

```
1.000000000   0.000000000   0.000000000  # a1
0.000000000   1.000000000   0.000000000  # a2
0.000000000   0.000000000   1.000000000  # a3

# Cell dimensions. Provide "angstrom" if you wish to provide
# these values in angstrom units.
# These are multiplied to a1, a2 and a3 respectively.
Cell_dimensions bohr
26.594331775231996 26.594331775231996 26.594331775231996

# G-vectors will be used upto this kinetic energy cut off.
# Provide "Rydberg" if you wish to specify the cut off in Rydberg
# atomic units.
Ecut=20.0 Hartree
/
```

The GAUSSIAN_PARAMETERS section contains parameters for the model Gaussian charge. It contains the Total_charge (-1 if an electron has been added to the system), Sigma (Gaussian width), Centre of the Gaussian in crystal units. We provide a script *g_fit.py* in the folder GaussianFit/ to fit the Sigma to the wavefunction charge denity. See Examples/Fit_gaussian/ for an example.

```
&GAUSSIAN_PARAMETERS:
# The charge state of the defect being simulated.
Total_charge = -2
# The width of the model Gaussian charge being used.
# (default: bohr units)
Sigma = 2.614
# These set the center of the Gaussian in crystal units.
Centre_a1 = 0.5
Centre_a2 = 0.5
Centre_a3 = 0.5
/
```

The DIELECTRIC_PARAMETERS section contains parameters for the dielectric profile. The general form is set by the string following &DIELECTRIC_PARAMETERS, which could be Bulk/Slab/Gaussian/Ribbon/Wire. Bulk is for bulk solids. Slab is for a slab of material or for more than one atom thick 2D materials (like MoS2/phosphorene). Gaussian is for one atom thick 2D materials (like BN/graphene) Ribbon is for nanoribbon system. Periodicity assumed in the z-direction. (see Examples/1D/) Wire is for a nanowire like system (periodicity assumed in the z-direction). Please refer to the following subsections for the &DIELECTRIC_PARAMETERS for the various profiles.

3.1 3D, BULK SYSTEMS

```
# Set "Bulk" for 3D, bulk systems.
&DIELECTRIC_PARAMETERS Bulk
# Sets the value of the dielectric constant along a1, a2, a3.
Epsilon1_a1 = 5.76
Epsilon1_a2 = 5.76
Epsilon1_a3 = 5.76
/
```

3.2 2D SYSTEMS

For slab systems and 2D materials with more than one atom thickness:

```
&DIELECTRIC_PARAMETERS Slab
# Epsilon1 is the value of dielectric constant inside the
# material, and Epsilon2 is the value outside (1.0 if vacuum).
# These values are provided for the profile
# along every lattice vector.
Epsilon1_a1 = 15.0
Epsilon2_a1 = 1.0
Epsilon1_a2 = 15.0
Epsilon2_a2 = 1.0
Epsilon1_a3 = 2.0
Epsilon2_a3 = 1.0
# The width of the slab-like  dielectric profile.
Width = 11.40454 bohr
# The center of the profile.
Centre = 5.70226 bohr
# The smoothness parameter for the edges of the slab-like profile.
Smoothness = 0.37807 bohr
# To write the epsilon profile to file.
# The array is written as a numpy save array.
Plot_eps = True
/
```

For one-atom thick 2D materials like BN, the profile used is like a Gaussian inside the material. The Gaussian does not go to zero in the vacuum however, it is shifted so that it goes to 1.0 (dielectric constant in vacuum).

```
&DIELECTRIC_PARAMETERS Gaussian
# Gauss_amp parameters specify the value of dielectric constant
# inside the material. The value outside the material is assumed to be 1.0.
# These values are provided for the profile
# along every lattice vector.
```

```
Gauss_amp_a1 = 11
Gauss_amp_a2 = 11
Gauss_amp_a3 = 2
# The center of the profile.
Centre = 5.70226 bohr
# Width of the Gaussian (default: bohr units)
Sigma = 0.85868842
# To write the dielectric profile to file.
# The array is written as a numpy save array.
Plot_eps = True
/
```

3.3 1D SYSTEMS

For 1D systems, we assume periodicity in the z direction. We also assume the angle between the lattice vectors is 90 degrees. (This is not a bad assumption since the a1, a2 directions are aperiodic, hence it is always possible to construct orthogonal lattice vectors, perpendicular to the periodic direction.)

The Ribbon profile for BN is a combination of a Gaussian profile along x and slab-like profile along the y direction. Sample input for the dielectric parameters:

```
# Defines a Ribbon profile.
&DIELECTRIC_PARAMETERS Ribbon
# Sets a Gaussian profile along x
Gauss_along_x = True
# Assumes slab-like profile along y.
# For a slab-like profile along x and y,
# set both to be False.
Gauss_along_y = False
# Center of the Gaussian
C_x = 14.1728712347 bohr
# Dielectric constant along a1, a2 and a3
# for a Gaussian profile along either direction.
# Can be left unspecified if this is not the
# case.
Gauss_amp_a1 = 2.94
Gauss_amp_a2 = 12.0
Gauss_amp_a3 = 12.0
# Width of the Gaussian profile.
Sigma = 0.85868842
# Center of the slab, along the y direction.
C_y = 17.96054782
# Width of the slab. Provide W_x
# for slab along x direction.
W_y = 26.0150 bohr
# Smoothness parameter for edges of the slab.
Smoothness = 0.37 bohr
# The 2D epsilon array for each direction are written
# to files. The files are numpy save files,
# which can be loaded into a different script to
# visualize
Plot_eps = True
```

To construct a wire with a particular cross-section, like hexagonal, the vertices of the cross-section can be provided in a file. See Examples/1D/Wire_samples/ Hexagonal_cross-section/ for an example. The dielectric parameters are provided in the following manner for this case:

```
&DIELECTRIC_PARAMETERS Wire
# Epsilon1 is the value of dielectric constant inside the
# material, and Epsilon2 is the value outside (1.0 if vacuum).
# These values are provided for the profile
# along every lattice vector.
Epsilon1_a1 = 7.34
Epsilon2_a1 = 1.0
Epsilon1_a2 = 7.34
Epsilon2_a2 = 1.0
Epsilon1_a3 = 8.73
Epsilon2_a3 = 1.0
# The vertices of the cross-sectional polygon
# are provided using a file. See
# Examples/1D/Wire\_samples/Hexagonal\_cross-section/
# for an example
Vertices_file =  vertices
/
```

A wire with a circular cross-section with a smoothening of the edges can also be constructed. See Examples/1D/Wire_samples/ Circular_cross-section/ for an example. The dielectric parameters for this case:

```
&DIELECTRIC_PARAMETERS Wire
# Epsilon1 is the value of dielectric constant inside the
# material, and Epsilon2 is the value outside (1.0 if vacuum).
# These values are provided for the profile
# along every lattice vector.
Epsilon1_a1 = 7.34
Epsilon2_a1 = 1.0
Epsilon1_a2 = 7.34
```

```
Epsilon2_a2 = 1.0
Epsilon1_a3 = 8.73
Epsilon2_a3 = 1.0
# Defines a circular cross-section
Circle = True
# Radius of the disk/circle
Radius = 8.0
# Center in crystal units
c_a1 = 0.5
c_a2 = 0.5
# Smoothening parameter for the edges
# of the circle.
Smoothness = 2.0
/
```

4 POTENTIAL ALIGNMENT

The scripts in the PotentialAlignment folder can be used to obtain the two potential alignment terms, Eqn. 1.4 and Eqn. 1.5.
dV_0p.py is used to compute $V_0 - V_p$ and plots the planar averaged potential along the desired direction.
dV_mD.py plots the planar averaged DFT difference potential and the model potential along the desired direction.
The input parameters for dV_0p.py and dV_mD.py are read from an input file.
Format of the input files:

```
&dV_0p
file_type = cube # No quotes. Takes cube/xsf
file_neutral =  # No quotes. Path to the neutral DFT potential file
file_pristine = # No quotes. Path to the pristine DFT potential file
plt_dir = a1 # No quotes. Takes a1/a2/a3. If a1
             # is specified, the data is averaged along a2
             # and a3 directions and the planar averaged data is plotted
             # along a1 in a file pa_dv0p_a1.plot
factor = Ryd # factor to be multiplied to the cube/xsf data. If the data is in
             # rydberg and the plot is needed in eV, specify
             # factor = Ryd. If the data is in Hartree units,
             # specify factor = Hartree
/
```

Output: pa_dv0p_a1.plot (for plt_dir = a1)

```
&dV_mD
file_type = cube # No quotes. Format of the DFT potential files: cube/xsf
```

```
file_model =  # No quotes. Path to the model potential file (.pysave)
file_charged = # No quotes. Path to the charged DFT potential file
file_neutral = # No quotes. Path to the neutral DFT potential file
plt_dir = a1 # No quotes. Takes a1/a2/a3. If a1
             # is specified, the data is averaged along a2
             # and a3 directions and the planar averaged data is plotted
             # along a1 in files DFTdiff_a1.plot, model_a1.plot
factor = Ryd # factor to be multiplied to the cube/xsf data. If the data is in
             # rydberg and the plot is needed in eV, specify
             # factor = Ryd. If the data is in Hartree units,
             # specify factor = Hartree
/
```

Output: DFTdiff_a1.plot, model_a1.plot (for plt_dir = a1)

5 EXAMPLES

We provide examples for bulk (Diamond), 2D (monolayer MoS_2) and 1D (BN nanoribbon) systems.

5.1 DIAMOND

The Diamond example is located in Examples/3D/Diamond folder. This example studies a C vacancy in Diamond in the -2 charge state. This example will compute the complete corrections for this system (steps 4, 5 and 6 in the Outline) for a 4×4×4 super cell simulation of this defect.

The E_q^{lat} term is computed in the Model_Scaling/ directory.
Go to the Model_Scaling/ directory.

```
  $cd Examples/3D/Diamond/Model_Scaling
```

The lattice correction term, E_q^{lat}, is given by Eqn. 1.3. To obtain the isolated model energy, the model periodic energy is computed for three supercell sizes, say 4x4x4, 5x5x5 and 6x6x6 for this case. These energies are fitted with a polynomial, of the form Eqn. 13 in CoFFEE paper,

$$p(\Omega) = f_1 + f_2/\Omega^{1/3} + f_3/\Omega \tag{5.1}$$

and extrapolated to infinity to obtain $\mathrm{E}_q^{\mathrm{iso,m}}$. Ω here is the volume of the super cell. See also Fig 4 (a) in the paper for the fit.

Note: Extrapolation is dependent on the Gaussian width used in the calculation! See Table 5.1

To run this example, perform the following steps (replace path_to_coffee_folder with your path to the CoFFEE_xxx/ folder):

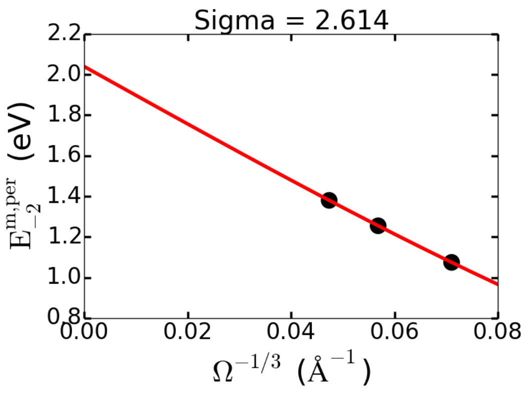


Figure 5.1: Extrapolation of model energies in Diamond for Gaussian width of 2.614 bohr

1.
```
$cd 4x4x4/
$path_to_coffee_folder/coffee.py in > out
$grep ! out
```
2.
```
$cd ../5x5x5/
$path_to_coffee_folder/coffee.py in > out
$grep ! out
```
3.
```
$cd ../6x6x6/
$path_to_coffee_folder/coffee.py in > out
$grep ! out
$cd ../
```

The model total energy is printed in the out files. The following command will show this energy (while you are in the right folder):

```
$grep ! out
```

For this example, the model energies should be:
4×4×4: 1.0744 eV
5×5×5: 1.2549 eV
6×6×6: 1.3800 eV

Use plot_fit.py in the same Model_Scaling/ directory to compute the polynomial fit and extrapolation to obtain $E_q^{iso,m}$ for these values. See Fig. 5.1

```
$python plot_fit.py
```

$E_q^{iso,m}$ is computed to be 2.04 eV.

The lattice corrections, for these super cells then are: ($E_q^{iso,m} - E_q^{per,m}$ (n×n×n))
4×4×4: 0.97

	$\sigma = 2.61$ bohr	$\sigma = 1.61$ bohr
$E_q^{\mathrm{per,m}}$ (4×4×4) (eV)	1.07	2.31
$E_q^{\mathrm{per,m}}$ (5×5×5) (eV)	1.25	2.50
$E_q^{\mathrm{per,m}}$ (6×6×6) (eV)	1.38	2.64
$E_q^{\mathrm{iso,m}}$ (eV)	2.04	3.30
Correction (4×4×4) (eV)	0.97	0.98
Correction (5×5×5) (eV)	0.79	0.79
Correction (6×6×6) (eV)	0.66	0.66

Table 5.1: Model calculations for the -2 vacancy in Diamond computed for two widths of the model Gaussian, $\sigma = 2.61$ bohr and $\sigma = 1.61$ bohr. The first three rows are the model energies for the 4×4×4, 5×5×5 and 6×6×6 super cell sizes. These are extrapolated to infinity to find $E_q^{\mathrm{iso,m}}$. The corrections (($E_q^{\mathrm{iso,m}} - E_q^{\mathrm{per,m}}$ (n×n×n)) are independent of the width of the model Gaussian.

5×5×5: 0.785
6×6×6: 0.66

The model potential and charge density can be plotted from the Plot/ directory inside the super cell directories.

Plot the model potential for the 4x4x4/ model calculation:

```
$cd 4x4x4/Plot/
$ln -s ../V_r.npy .
$path_to_coffee_folder/PotentialAlignment/Utilities/plavg.py in_V
$cd ../../../
```

This will create a file plavg_a1.plot which we will use for computing the potential alignment term.

Next, we compute the $\Delta V_{q-0/m}$ term. (Eqn. 1.4) This is done from the PA_q0/ folder in the Diamond/ folder.

```
$cd PA_q0
```

We provide the planar averaged V_q^{DFT} and V_0^{DFT} potentials computed using Quantum Espresso for the 4x4x4 super cell in the files: plavg_q_a1.plot and plavg_0_a1.plot. These can be generated from your DFT calculation by writing the DFT potentials into a cube/xsf format. The cube/xsf file can then be planar averaged using the utility plavg.py, located in the path:
path_to_CoFFEE_folder/PotentialAlignment/Utilities/. The details on how to run this script is provided in the same folder.

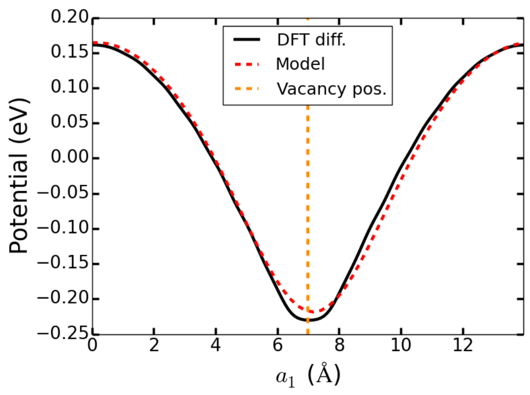


Figure 5.2: Plot of the planar averaged DFT difference potential: $V_q - V_0$, compared to the model potential.

We compare the DFT difference potential with the model potential (far from the defect) computed in the folder: ../Model_Scaling/4x4x4/Plot/ using the script plot_DVq0.py. Run this script to obtain the plot:

```
$python plot_DVq0.py
```

You will find that the difference between the potentials, far from the defect is negligibly small. See Fig. 5.2 There is hence no contribution from this term to the correction. It is however necessary to always check if this is true. This term being small indicates that the charged defect has been modelled well.

Next, we compute the $\Delta V_{0/p}$ term (Eqn. 1.5). Go back to the Diamond folder and move to the PA_0p/ folder.

```
$cd ../PA_0p/
```

We provide the planar averaged V_p and V_0 potentials computed using Quantum Espresso for the 4×4×4 super cell in the files: plavg_p_a1.plot and plavg_0_a1.plot These can be generated from your DFT calculation by writing the DFT potentials into a cube/xsf format as discussed above.

Use the script plot_DV0p.py to plot this difference and find the value far from the defect site:

```
$python plot_DV0p.py
```

This term is about 0.1 eV. See Fig. 5.3

The correction for the 4×4×4 super cell is then given by:

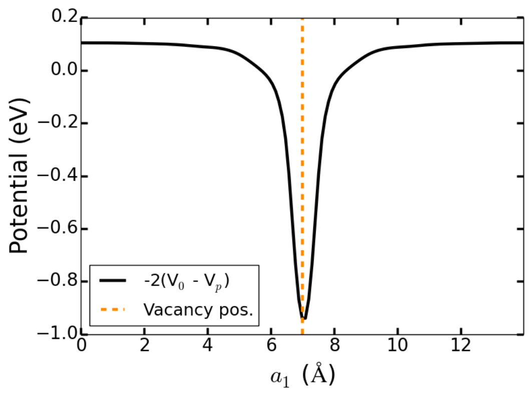


Figure 5.3: Plot of the planar averaged difference potential: $V_0 - V_p$.

$E_q^{lat} + (-\Delta V_{q-0/m}) + (-q\Delta V_{0/p})$ = 0.97 + 0.0 + 0.1 eV = 1.07 eV

Uncorrected formation energy: 13.189 eV
Corrected formation energy: 14.259 eV

5.2 Monolayer MoS_2

This example studies a S vacancy in monolayer MoS_2 in the -1 charge state. It is located in Examples/2D/MoS2 folder. This example will compute the complete corrections for this system, steps 4, 5 and 6 in the Outline, for a 6×6×6 super cell simulation of this defect.

The E_q^{lat} term , given by Eqn. 1.3, is computed in the Model_Scaling/ directory.
Go to the Model_Scaling/ directory.

```
$cd Examples/2D/MoS2/Model_Scaling/
```

To obtain the isolated model energy for this system, the model periodic energy is computed for several supercell sizes. These values are then fit with a fifth order polynomial as shown in Fig. 6 (c) in the CoFFEE paper. This polynomial has the form:

$$p(\alpha) = f_0 + f_1/\alpha + f_2/\alpha^2 + f_3/\alpha^3 + f_4/\alpha^4 + f_5/\alpha^5 \tag{5.2}$$

where α denotes the supercell dimension: $\alpha \times \alpha \times \alpha$. We uniformly scale the vacuum with the in-plane super cell. The third α refers to the scaling of the cell dimension in the out-of-plane direction. For $\alpha = 6$, the vacuum thickness is ~ 16Å and thickness of MoS_2 is 3.2Å. The polynomial extrapolation, $1/\alpha$ -> 0, gives the $E_q^{iso,m}$.
Note that computing this term for large supercell sizes can be time consuming. The users can make use of the MPI parallelization of this code.

We provide the input files for the various supercell sizes in folders alpha.*. To understand how this example works, the model energy can be computed for a few small supercell sizes.

We provide the results for the larger supercell sizes and a script to plot the polynomial.

To run this example, perform the following steps:

1.
   ```
   $cd alpha.4/
   $path_to_coffee_folder/coffee.py in > out
   $grep ! out
   ```

2.
   ```
   $cd ../alpha.5/
   $path_to_coffee_folder/coffee.py in > out
   $grep ! out
   ```

3.
   ```
   $cd ../alpha.7/
   $path_to_coffee_folder/coffee.py in > out
   $grep ! out
   ```

4.
   ```
   $cd ../alpha.8/
   $path_to_coffee_folder/coffee.py in > out
   $grep ! out
   $cd ../
   ```

(Feel free to run the other supercell sizes as well)

The model total energy is printed in the out files. The following command will show this energy (while in the right folder):

```
$grep ! out
```

The dielectric profile, model potential and model charge density are written out in numpy save files. Instructions on how to plot the model charge density and potential are provided in the Plot folder within the alpha.* folders
The script plot_eps.py in alpha.6/ folder can be used to visualize the epsilon profile:

```
$cd alpha.6/
$python plot_eps.py
$cd ../
```

See Fig. 5.4.

Go to alpha.6 and plot the model potential:

```
$cd alpha.6/Plot
$ln -s ../V_r.npy .
$path_to_coffee_folder/PotentialAlignment/Utilities/plavg.py in_V
$cd ../../
```

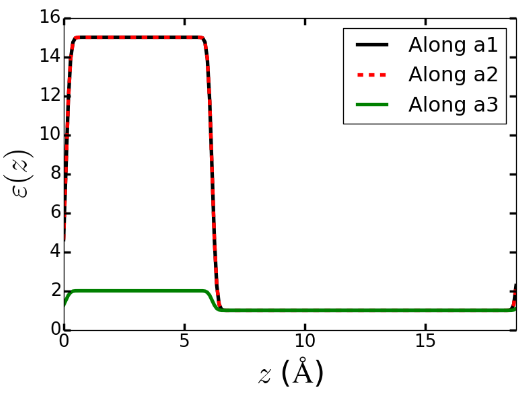


Figure 5.4: Epsilon profile along the in-plane a1, a2 directions and the out-of-plane a3 direction.

This produces a file plavg_a3.plot. We will use this later to compute the potential alignment term.

For this example, the model energies should be:
4x4x4: 0.421 eV
5x5x5: 0.463 eV
6x6x6: 0.488 eV
8x8x8: 0.516 eV

10x10x10: 0.529 eV
20x20x20: 0.558 eV
40x40x40: 0.588 eV
80x80x80: 0.618 eV

Use plot_fit.py in the Model_Scaling/ folder to compute the fifth order polynomial fit and extrapolation to obtain $E_q^{\mathrm{iso,m}}$ for these values.

```
$python plot_fit.py
```

$E_q^{\mathrm{iso,m}}$ is computed to be 0.66 eV. See Fig. 5.5.

The corrections, are then given by: ($E_q^{\mathrm{iso,m}} - E_q^{\mathrm{per,m}}$ (n×n×n))
4x4x4: 0.239 eV
5x5x5: 0.197 eV
6x6x6: 0.172 eV
8x8x8: 0.144 eV

Next, we compute the $\Delta V_{q-0/m}$ term (Eqn. 1.4). Go back to the MoS2/ folder and move to the PA_q0/ folder.

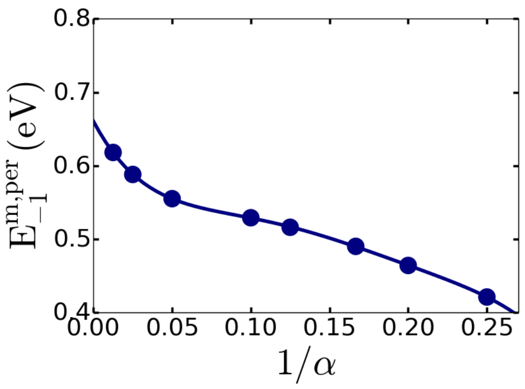


Figure 5.5: Scaling of the model energy for a -1 charged sulfur vacancy in MoS_2.

```
$cd ../PA_q0/
```

We provide the planar averaged V_q^{DFT} and V_0^{DFT} potentials computed using Quantum Espresso for the 6x6x6 super cell in the files: plavg_q_a3.plot and plavg_0_a3.plot These can be generated from your DFT calculation by writing the DFT potentials into a cube/xsf format. The cube/xsf file can then be planar averaged using the utility plavg.py, located in the path: path_to_CoFFEE_folder/PotentialAlignment/Utilities/ The details on how to run this script is provided in the same folder.

We compare the DFT difference potential with the model potential (far from the defect) computed in the folder: ../Model_Scaling/alpha.6/Plot/ using the script plot_DVq0.py.

```
$python plot_DVq0.py
```

See Fig. 5.6. You will find that the difference between the potentials, far from the defect is negligibly small. There is hence no contribution from this term to the correction. It is however necessary to always check if this is true. This term being small indicates that the charged defect has been modelled well.

Next, we compute the $\Delta V_{0/p}$ term (Eqn. 1.5). Go back to the MoS2/ folder and move to the PA_0p/ folder.

```
$cd ../PA_0p/
```

We provide the planar averaged V_p and V_0 potentials computed using Quantum Espresso for the 6x6x6 super cell in the files: plavg_p_a3.plot and plavg_0_a3.plot These can be generated from your DFT calculation by writing the DFT potentials into a cube/xsf format as discussed above.

Use the script plot_DV0p.py to plot this difference and find the value far from the defect site.

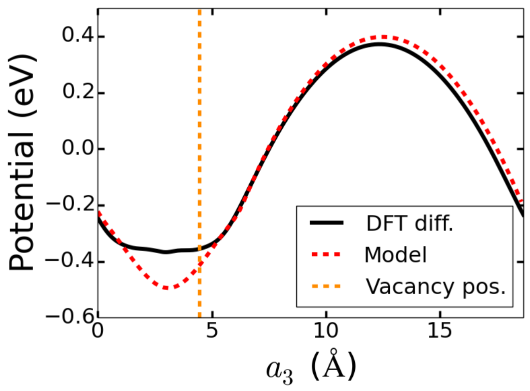


Figure 5.6: Plot to compare the DFT difference potential, $V_q^{DFT} - V_0^{DFT}$, with the model potential.

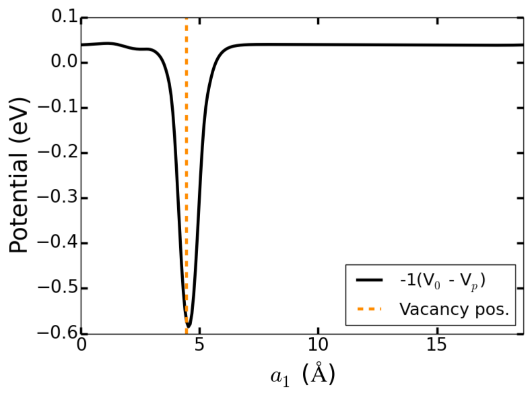


Figure 5.7: Plot of the difference potential, $-1(V_0 - V_p)$, for $\Delta V_{0/p}$ term

```
$python plot_DV0p.py
```

This term is small as well, about 0.04 eV. See Fig. 5.11.

The correction for the 6×6×6 super cell is then given by:

$E^{\mathrm{lat}}_q + (-\Delta V_{q-0/m}) + (-q\Delta V_{0/p}) = 0.172 + 0.0 + 0.04$ eV $= 0.212$ eV

Uncorrected formation energy: 4.438 eV
Corrected formation energy: 4.650 eV

5.3 BN NANORIBBON

This example studies a B vacancy in a BN nanoribbon in the -1 charge state. It is located in Examples/1D/NanoRibbon_BN/ folder. This example will compute the complete corrections for this system, steps 4, 5 and 6 in the Outline, for a 6×14×6 super cell simulation of this defect.

The E^{lat}_q term , given by Eqn. 1.3, is computed in the Model_Scaling/ directory.
Go to the Model_Scaling/ directory.

```
$cd Examples/1D/NanoRibbon_BN/Model_Scaling/
```

To obtain the isolated model energy for this system, the model periodic energy is computed for several supercell sizes. These values are then fit with a third order polynomial as shown in Fig. 8 (e) in the CoFFEE paper. This polynomial has the form:

$$p(\alpha) = f_0 + f_1\alpha^{-1} + f_2\alpha^{-2} + f_3\alpha^{-3} \tag{5.3}$$

f_0 then corresponds to the isolated model energy, $E^{\mathrm{iso,m}}_{-1}$. α denotes the supercell dimension: α (1×2.31×1).

For α = 6, the simulation cell dimensions are 15Å, 35Å and 15Å, in the x, y and z directions repectively. The z direction is the periodic direction. A vacuum of 15Å and 22Å has been introduced in the out-of-plane x direction and lateral y direction. The thickness of the ribbon is fixed in these cell sizes, with the number of atoms in the periodic direction and the vacuum padding scaling with α. The polynomial extrapolation, $1/\alpha$ -> 0, gives the $E^{\mathrm{iso,m}}_q$.

Note that computing this term for large supercell sizes can be time consuming. The users can make use of the MPI parallelization of this code.

We provide the input files for the various supercell sizes in folders alpha.*. To understand how this example works, the model energy can be computed for a few small supercell sizes. We provide the results for the larger supercell sizes and a script to plot the polynomial.

To run this example, perform the following steps:

1.
   ```
   $cd alpha.6/
   $path_to_coffee_folder/coffee.py in > out
   $grep ! out
   ```

2.
   ```
   $cd ../alpha.8/
   $path_to_coffee_folder/coffee.py in > out
   $grep ! out
   ```

3.
   ```
   $cd ../alpha.10/
   $path_to_coffee_folder/coffee.py in > out
   $grep ! out
   $cd ../
   ```

(Feel free to run the other supercell sizes as well)

The model total energy is printed in the out files. The following command will show this energy (while in the right folder):

```
$grep ! out
```

The dielectric profile, model potential and model charge density are written out in numpy save files. Instructions on how to plot the model charge density and potential are provided in the Plot folder within the alpha.* folders

To plot the dielectric profile, run the plot_eps.py script from alpha.6/ folder.

```
$cd alpha.6/
$python plot_eps.py
#cd ../
```

See Fig. 5.8.

Go to alpha.6 and plot the model potential:

```
$cd alpha.6/Plot
$ln -s ../V_r.npy .
$path_to_coffee_folder/PotentialAlignment/Utilities/plavg.py in_V
$cd ../../
```

This produces a file plavg_a3.plot. We will use this to compute the potential alignment term.
For this example, the model energies should be:
6x14x6: 1.457 eV
8x18.66x8: 1.589 eV
10x23.33x10: 1.651 eV

15x34.99x15: 0.529 eV
20x46.66x20: 0.588 eV

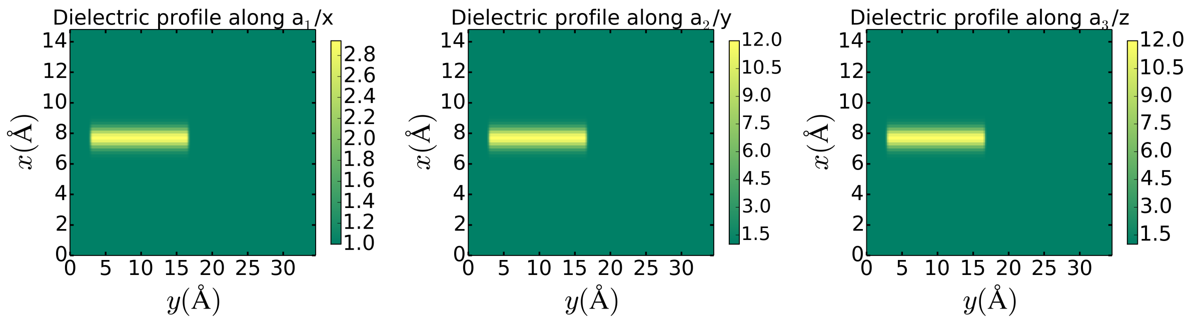


Figure 5.8: Dielectric profile constructed for the BN nanoribbon for $\alpha = 6$. The periodic direction of the ribbon is along the z/a$_3$ direction. The x direction is perpendicular to the plane of the ribbon, the dielectric constant in the material along that direction is thus lower than the in-plane directions (y and z).

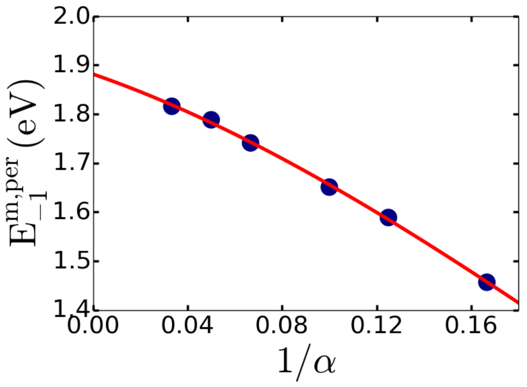


Figure 5.9: Scaling of the model energy for a -1 charged B vacancy in a BN nanoribbon.

30x69.99x30: 0.618 eV

Use plot_fit.py in the Model_Scaling/ folder to compute the third order polynomial fit and extrapolation to obtain $E_q^{iso,m}$ for these values.

```
$python plot_fit.py
```

$E_q^{iso,m}$ is computed to be 1.88 eV. See Fig. 5.9.

The corrections, are then given by: ($E_q^{iso,m} - E_q^{per,m}$ (n×n×n))
6x14x6: 0.423 eV
8x18.66x8: 0.291 eV
10x23.33x10: 0.229 eV

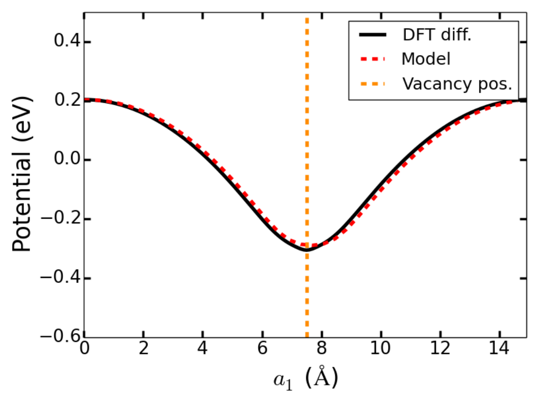


Figure 5.10: Plot to compare the DFT difference potential, $V_q^{DFT} - V_0^{DFT}$, with the model potential.

Next, we compute the $\Delta V_{q-0/m}$ term (Eqn. 1.4). Go back to the NanoRibbon_BN/ folder and move to the PA_q0/ folder.

```
$cd ../PA_q0/
```

We provide the planar averaged V_q^{DFT} and V_0^{DFT} potentials computed using Quantum Espresso for the 6x14x6 super cell in the files: plavg_q_a3.plot and plavg_0_a3.plot These can be generated from your DFT calculation by writing the DFT potentials into a cube/xsf format. The cube/xsf file can then be planar averaged using the utility plavg.py, located in the path: path_to_CoFFEE_folder/PotentialAlignment/Utilities/ The details on how to run this script is provided in the same folder.

We compare the DFT difference potential with the model potential (far from the defect) computed in the folder: ../Model_Scaling/alpha.6/Plot/ using the script plot_DVq0.py.

```
$python plot_DVq0.py
```

See Fig. 5.10. Note that in the DFT calculations, the nanoribbon was placed at the bottom of the simulation cell and in the model, it was placed in the middle. We have hence shifted the DFT difference potential to compare the two. The difference between the potentials, far from the defect is negligibly small. There is hence no contribution from this term to the correction. It is however necessary to always check if this is true. This term being small indicates that the charged defect has been modelled well.

Next, we compute the $\Delta V_{0/p}$ term (Eqn. 1.5). Go back to the NanoRibbon_BN/ folder and move to the PA_0p/ folder.

```
$cd ../PA_0p/
```

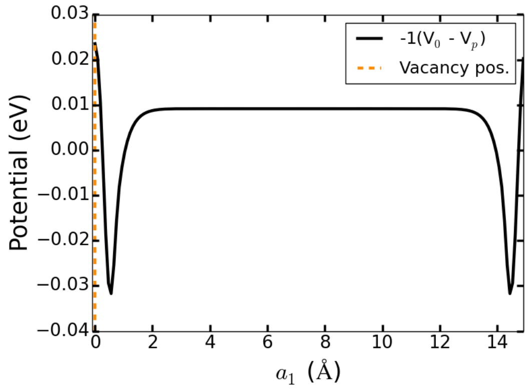


Figure 5.11: Plot of the difference potential, $-1(V_0 - V_p)$, for $\Delta V_{0/p}$ term

We provide the planar averaged V_p and V_0 potentials computed using Quantum Espresso for the 6x14x6 super cell in the files: plavg_p_a3.plot and plavg_0_a3.plot These can be generated from your DFT calculation by writing the DFT potentials into a cube/xsf format as discussed above.

Use the script plot_DV0p.py to plot this difference and find the value far from the defect site.

```
$python plot_DV0p.py
```

This term is small as well, about 0.01 eV. See Fig. 5.11.

The correction for the 6×14×6 super cell is then given by:

$E_q^{\text{lat}} + (-\Delta V_{q-0/m}) + (-q\Delta V_{0/p}) = 0.423 + 0.0 + 0.01 \text{ eV} = 0.433 \text{ eV}$

Uncorrected formation energy: 8.94 eV
Corrected formation energy: 9.373 eV

6 Additional references

Here are some additions references relevant to computing electrostatic corrections in 2D and 1D systems:

2D systems:

1. `http://link.aps.org/doi/10.1103/PhysRevLett.110.095505`
2. `http://link.aps.org/doi/10.1103/PhysRevB.89.205417`
3. `http://link.aps.org/doi/10.1103/PhysRevX.4.031044`

1D systems:

1. `http://link.aps.org/doi/10.1103/PhysRevB.90.085435`