Simulating 2D materials using Quantum Espresso

Installation

To run Quantum Espresso (QE) natively, a C and FORTRAN compiler is required, alongside various other dependencies. Therefore, it is recommended that QE is run via the virtual machine Quantum Mobile (QM), which contains Quantum Espresso and all its dependencies, in addition to a range of other computational physics tools, including Aiida which can be used to devise complex workflows leveraging the utility of parallel processing.

To run QM, the virtualisation software VirtualBox is required. Below is a step-by-step guide for installing this application, alongside common errors you may encounter.

1) Install all dependencies of VirtualBox, most importantly the latest distribution of Python and the Python Windows interface pywin32.
    a. Download Python3 using the link and ensure Python is added to PATH (see tutorial) so Python commands can be used directly from the command line.
    b. Use the command `$pip install pywin32` to install the pywin32 package.
2) Download the latest version of VirtualBox and run through the installation menu. At this point, the installation wizard will flag up if you have not downloaded any necessary dependencies.

Once VirtualBox has been set up, a virtual machine (VM) image of QM must be downloaded and imported into VirtualBox as below.

1) Download the latest QM image here (NB the page links through to the necessary file housed in a Google Drive, which flags up that it cannot scan the file for malware due to its size).
2) Open VirtualBox and import the downloaded image using File -> Import Appliance. After this, a VM labelled Quantum Mobile (version) should appear in the main VirtualBox menu.
3) Launch the QM virtual machine by using the Run command in VirtualBox. At this point, the VM will either launch and run without error, or the machine may crash and return an error inside VirtualBox, two possible explanations for which are given below.
    a. NS_ERROR_INVALID_ARG (0X80070057) – Indicates insufficient hard disk memory is present to create a VM image. Solve this by clearing sufficient space (on the order of 5-10 GB)
    b. vt-x not enabled in BIOS (encountered on Windows) – Indicates the system has not given permission for the execution of the Intel virtualisation software necessary to run VirtualBox on a Windows machine.
        i. Solve this by opening the BIOS following system-specific instructions (NB Only modify the BIOS if confident and the machine has been

backed up entirely because any errors may have severe consequences).
    1. E.g. for Windows 10 - Settings -> Update & Security -> Recovery -> Restart now (under Advanced Settings)
    2. Upon restart - Troubleshoot -> Advanced Options -> UEFI Firmware Settings -> Restart

ii. Once the BIOS is opened, enable vt-x virtualisation by following method 3 outlined here.
    1. After entering BIOS upon reboot – Security -> System Security -> Enable Virtualization Technology or VTx/VTd
    2. Reboot computer and attempt to run QM from VirtualBox

Once QM is installed and imported into VirtualBox and the VM can be successfully launched, we need to check all packages are configured and functional. Execute the following commands to check that all core services with QM are working as intended.

`$ systemctl --type=service` – checks core Aiida services.
`$ systemctl status slurm-resources` – checks slurm scheduler.
`$conda env list` – shows a list of environments already setup.
`$conda activate aiida` – activates the aiida environment.
`$verdi status`
`$ verdi computer test local_direct` – checks computer is setup to run jobs on slurm scheduler.
`$aiida-quantumespresso calculation launch pw -d -i -X qe.pw@local_slurm -F` – submits test job.
`$verdi process list -a` – displays outcome of test job.

QM contains a range of environments (python environments allow you to isolate projects, for example, to prevent dependency conflicts) housing the different tools contained within QM. The aiida environment allows you to use the aiida package for construction of workflows. All subsequent guides will instead make use of the qespresso environment for simple use of QE in electronic structure calculations.

To access this environment, use the command `$conda activate qespresso`

The final step of configuring QM for use is installation of the package PWTK for scripting batch files (useful for running repeated calculations over a range of values for a specific parameter to check for convergence). To do so, execute the following commands from the root directory.

`$mkdir pwtk` – makes a directory named pwtk.
`$cd pwtk` – navigates into the pwtk directory.
`$wget "http://pwtk.ijs.si/download/pwtk-2.0.tar.gz"` – retrieves the latest version of PWTK from the URL given, modify as appropriate if a newer version of PWTK has been released.
`$tar -zxvf pwtk-2.0.tar.gz` – extracts the pwtk directory from the downloaded file.
`$cd` – navigates to the root directory.

`$nano .bashrc` – opens PATH file using the text editor nano. Other text editors can be used e.g. vi.
`$export PATH="/home/max/pwtk/pwtk-2.0:$PATH"` – add this to bottom of PATH file (if using nano as the text editor, save changes using ctrl+o and exit using ctrl+x).
`source .bashrc` - reboots the terminal to allow changes made to PATH file to take effect
`$env | grep PATH` – shows files in PATH, should include pwtk directory.
`$pwtk` – tests out pwtk command. Should return a list of options explaining how to use the command, rather than a command not found error.

Optionally, if you like to work in Jupyter notebooks, execute the command `$pip install jupyter` notebook inside of the qespresso environment. This is also recommended for the package matplotlib (for plotting data).

Using QE (All basic information is taken from this tutorial which is incredibly useful)

Types of basic calculations which can be performed using QE

- SCF – Self-consistent field calculation used to determine energy of an input structure by minimising energy with respect to the electron density functional, until a threshold difference in energy between successive iterations is met.
- NSCF – Non-self-consistent field calculation, used to determine energy in a non-self-consistent fashion by reading in the optimised electron density from a prior SCF calculation. Less computationally expensive than an SCF calculation so permits calculations to be run using a denser k-point grid for use as input in density of states or band structure calculations.
- Density of states – Used to determine the number of available energy states as a function of energy, yielding an output .dat file which can be plotted using matplotlib.pyplot.
- Band structure – Used to determine the band structure for an input structure, which can be plotted from the command line or using matplotlib.pyplot.
- Relax/vc-relax – Used to optimise an input structure by minimising the energy and forces exerted on constituent atoms, altering either only atomic positions (relax) or atomic positions and lattice constants (vc-relax) NB the latter calculation type is extremely computationally expensive.

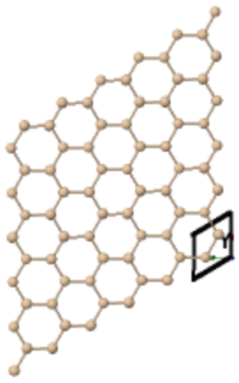
Once inside the qespresso environment, all QE calculations can be executed from the command line using the commands listed below:

- pw.x – used to perform self-consistent field (SCF) and non-self-consistent field (NSCF) energy calculations, as well as relax and vc-relax structure optimisation calculations.
- pwtk – used to script batch calculations, whereby a calculation is repeated varying a single parameter over a defined set of input values.
- dos.x – used to perform density of states calculations.
- bands.x – used to perform band structure calculations.
- plotband.x – used to plot output data from a band structure calculation from the command line.

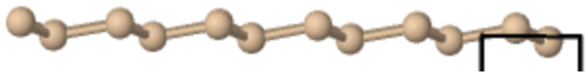
Key parameters for all calculations

Key input parameters for all calculations will be highlighted below using a sample input file for silicene, the 2D allotrope of silicon and silicon analogue of graphene, which has a hexagonal honeycomb structure akin to graphene but is buckled, as shown below.

Top-down view

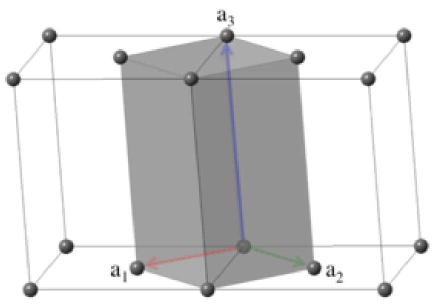


Side-on view



Silicene has an orthorhombic base-centred lattice structure shown below.
Lattice vectors a_1, a_2 and a_3 are given in terms of the 3 lattice constants a, b and c by the expressions below:

$a_1 = (a/2, -b/2, 0)$ $a_2 = (a/2, b/2, 0)$ $a_3 = (0, 0, c)$



A sample input file for an SCF calculation for silicene is given below. Text in black is the raw input file, whilst red text is explanatory. The basic inputs for the pw.x command, which executes the SCF calculation below, are linked here.

silicene_scf.in

```
&CONTROL #Input file is split into a set of sections, of which some are mandatory e.g.
         &CONTROL, &SYSTEM and &ELECTRONS. Others are optional or only required
         for specific calculation types.
    calculation = 'scf' #Indicates calculation type
    prefix = 'silicene' #Tags output file, must be consistent across all calculations for a
                        specific structure to allow the output of one calculation to be used
                        in a subsequent one.
    outdir = '.' #Sets directory to place output file in.
```

```
        pseudo_dir = './pseudos/' #Gives file path for finding the directory containing the
                                  pseudopotentials used for this calculation.
/

&SYSTEM
         ibrav = -9 #The Bravais lattice index of the input structure, -9 denotes the
                    orthorhombic one-face base-centred A-type of silicene.
        celldm(1) = -7.2962 #lattice constant a in Bohr radii
        celldm(2) = -1.7321 #b/a
        celldm(3) = 5.1800 #c/a
        nat = 2 #The number of atoms in the unit cell.
        ntyp = 1 #The number of distinct atom types in the unit cell.
        ecutwfc = 50 #KE cut-off for wavefunctions in units of Rydbergs (1 Ry ~ 13.6eV).
        ecutrho = 200 #KE cut-off for charge density, by default 4 x ecutwfc which is suitable
                      for PAW pseudopotentials, but a larger value may be required for
                      ultrasoft pseudopotentials.
        occupations = 'smearing' #option used to improve convergence for metals (silicene
                                 falls into this category as a semimetal with zero band gap).
                                 Not used for semiconductors or insulators.
        smearing = 'gaussian' #sets method of smearing, gaussian is the default option.
        degauss = 0.01 #sets value of gaussian smearing.
/

&ELECTRONS
        conv_thr =   1d-8 #convergence threshold for self-consistency
/

ATOMIC_SPECIES
Si 28.0855 Si.pbe-n-kjpaw_psl.1.0.0.UPF #list of atom types in the unit cell, atomic masses,
                                        then the name of the corresponding
                                        pseudopotential file. Each atom type is placed on a
                                        successive line.

ATOMIC_POSITIONS crystal #coordinates of each atom of the basis relative to each lattice
                         point. Can be specified in Cartesian coordinates, lattice
                         coordinates (as a fraction of the lattice constant a) or in crystal
                         coordinates (as a linear superposition of lattice vectors).
        Si      0.6666666667     0.3333333333     0.3952910667
        Si      0.3333333333     0.6666666667     0.4172089333

K_POINTS automatic #array of k points in Brillouin zone to sample, automatic results in
generation of an n1 x n2 x n3 (first 3 numbers) Monkhorst-Pack grid with an off-set from the
origin given by the latter 3 numbers.
        10 10 1 0 0 0
```

To execute an SCF calculation using this input file, use the command below.

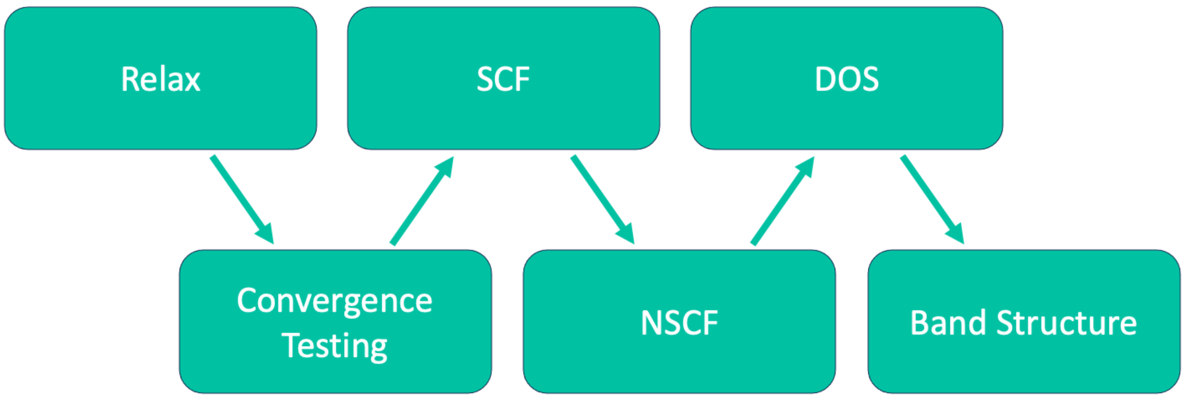
```
$pw.x < silicene_scf.in > silicene_scf.out
```

This performs the SCF calculation and creates an output file in the directory with file path given by the parameter ‘outdir’. This output file is often extremely long and difficult to read. The most useful information is obtained by searching for ‘total energy’ which gives the energy of the input structure in Rydbergs (~13.6 eV). The highest occupied energy level can be obtained by searching for ‘highest occupied’ whilst the lowest unoccupied energy level will only be calculated if the number of bands to calculate is explicitly stated by setting the value of ‘nbnd’ in the &SYSTEM section. By default, unless the input structure is identified as a metal, only valence bands are calculated (i.e. half the total number of electrons specified at the top of the output file). To additionally calculate unoccupied conduction bands, explicitly state the value of ‘nbnd’ as half the total number of electrons + however many conduction bands you wish to obtain. In addition to the output file, SCF calculations also produce a .save directory and .xml file which must be kept in the same directory as the SCF files, along with all input files for subsequent calculations which use the SCF output as an input.

Pseudopotentials are used in addition to the functionals used as the input for other popular computational DFT packages such as Gaussian (the basis set is not specified as pw.x calculations explicitly use a plane-wave basis set). Pseudopotentials approximate the full-electron potential, by modelling core electrons as static (with them pre-computed) creating a potential which screens the attraction between the nucleus and valence electrons (see the linked presentation). Different pseudopotentials have different methods for dealing with electron-ion interaction. There are 3 common methods for calculating electron-ion interactions: PAW (projector augmented wave), ultrasoft or norm-conserving described here. A pseudopotential is by default associated with an XC (exchange correlation) functional, detailed in the raw file. However, this functional can be explicitly specified using the ‘input_dft’ parameter in the &SYSTEM section of a pw.x input file. The XC functional can be broken into 4 components: exchange type, correlation type and the gradient corrections to both. A list of XC functionals QE is compatible with is linked here.

Pseudopotentials can be downloaded from various online libraries including a library maintained by QE itself. All pseudopotentials are stored as plain text files which outline the pseudopotential type (method for calculating electron-ion interactions) and XC functional in the first 10 lines. To run the SCF calculation outlined prior, an Si pseudopotential must be downloaded and placed in a directory, the relative file path to which is given by the parameter ‘pseudo-dir’ and the file name for which is given as the 3rd value of ‘ATOMIC_SPECIES’ line.

Typical workflow



Relax – Relax calculations iteratively vary cell parameters of an input structure and calculate the energy of the resultant structure (via an SCF calculation) to minimise the forces on all atoms in an input structure (thus achieving a more physically reasonable version of the input). There are 2 types of relax calculation: relax (only atomic positions permitted to vary) and vc-relax (atomic positions and lattice constants permitted to vary). Naturally, with more degrees of freedom to explore, vc-relax calculations are more computationally expensive.

Relax calculation input files will likely differ from the SCF input file outlined above in the following ways:

```
&CONTROL
        calculation = 'vc-relax' #different calculation type, could also be 'relax'
        etot_conv_thr = 1e-5 #convergence threshold for energy
        forc_conv_thr = 1e-4 #convergence threshold for forces

&CELL
        cell_dofree = 'ibrav' #selects which cell parameters can be altered, 'ibrav' indicates
                               all axes and angles can be altered whilst remaining consistent
                               with the initial choice of Bravais lattice structure.
```

Relax calculations are executed from the command line in the same way as SCF calculations. All subsequent calculations should make use of the newly calculated optimised lattice constants and atomic positions, retrieved from the output file.

SCF – Outlined above.

Convergence Testing – Convergence testing involves repeating SCF calculations for a range of input parameters to determine the minimally computationally expensive set of values which will obtain a converged value of energy. Parameters to be varied include ecutwfc, ecutrho, kpoints and separation between successive layers for a 2D material. The package PWTK can be used to execute batch calculations varying the input value for a single parameter. For example, an input file for calculating energy for a range of values of kpoints is given below.

silicene_scf_kpoints.pwtk

```
load_fromPWI silicene_scf.in #loads the SCF input from file.

set fid [open etot_vs_kpoints.dat w] #open a file for writing resulting total energies.

foreach k { 6 8 10 12 14 } { #loops over different "kpoints" values.

   set name si_scf_kpoints-$k #sets the names of the input and output files.

   K_POINTS automatic "$k $k 1 0 0 0" #sets the kpoints values for each input file (k x k x 1
                                           grid used for our 2D material)

   runPW $name.in #executes SCF calculation

   set Etot [::pwtk::pwo::totene $name.out]
   puts $fid "$ecut $Etot" #extracts the "total energy" from the SCF output file and writes
                              it to the .dat output file
}

close $fid
```

This program is executed by running the command `$pwtk silicene_scf_kpoints.pwtk`
The output is written to a .dat file which can be plotted using Matplotlib.pyplot.

NSCF – Non-self-consistent field calculations as less computationally expensive so permit a denser k-point grid to be probed, typically in preparation for a density of states calculation. Therefore, an NSCF input file will differ from an SCF input file in the following ways only:

```
&CONTROL
        calculation = ‘nscf’ #calculation type changed to NSCF.

&SYSTEM
        occupations = ‘tetrahedron’ #suitable setting for a subsequent DOS calculation.

K_POINTS automatic
        12 12 1 0 0 0 #denser k-point grid used than for SCF calculation.
```

NSCF calculations are executed using the same command as for SCF calculations. NSCF input files must be placed in the same directory as the corresponding SCF files, because they load the optimised electron density from the previous SCF calculation.

DOS (density of states) – Calculates the number of energy states available as a function of energy. As for SCF calculations, ‘nbnd’ must be explicitly stated for conduction bands to be calculated for non-metals. Increasing the value of ecutrho for the pre-requisite SCF and NSCF calculations will likely give more accurate results. DOS calculation input files are of the form below:

silicene_dos.in

```
&DOS
        prefix = 'silicene',
        outdir = '.'
        fildos = 'silicene_dos.dat' #determines the file name for the output file.
        !emin = -15
        !emax = 4 #alongside emin, sets range of energy values (eV) to calculate g(E) over.
                   Defaults to band extrema so only necessary if you want to study part of
                  the band structure specifically.
/
```

This calculation is executed using the command `$dos.x < silicene_dos.in > silicene_dos.out` The useful output of this calculation is a .dat file containing the range of energy values calculated over (eV), density of states and the cumulative integral of g(E) vs E, as well as the Fermi energy. This can be plotted using Matplotlib.pyplot to generate useful DOS plots.

Band structure – Band structure calculations have 3 stages: calculation, post-processing, and plotting. Bands calculations have an input file differing from that for an SCF calculation in the following ways:

silicene_bands.in

```
&CONTROL
        calculation = ‘bands’ #calculation type set to a band structure calculation.
        !occupations #smearing not used.

K_points {crystal_b} #crystal_b indicates k points are high symmetry points used in band
                       structure calculations, given in crystal coordinates.
4
# Gamma-M-K-Gamma -> #high symmetry k points in Brillouin zone. Formatted as fractions
                      of a1, a2 and a3 and then weighting, indicating how many k points
                      to sample between this point and the next.
        0.0000000000   0.0000000000   0.0000000000   40
        0.5000000000   0.0000000000   0.0000000000   20
        0.3333333333   0.3333333333   0.0000000000   34
        0.0000000000   0.0000000000   0.0000000000   0
```

This calculation is executed using the same command as an SCF calculation. The bands calculation is an NSCF calculation, so must be carried out in the same directory as a prior SCF calculation. To post-process the results, a program of the form below is used.

silicene_bands_pp.in

```
&BANDS
 prefix = 'silicene'
 outdir = '.'
 filband = 'silicene_bands.dat' #sets the filename the processed band structure data is
                                  entered into.
/
```

This program is executed using the command `$bands.x < silicene_bands_pp.in > silicene_bands_pp.out` The resultant .dat file can be plotted using Matplotlib.pyplot, or from the command line using the command `$plotband.x`

This program asks you to provide the file name for the post-processed .dat file, as well as the minimum and maximum energies for plotting, Fermi energy (which can be retrieved from a prior NSCF/DOS calculation), deltaE (the tick marks for the y axis energy scale) and referenceE (the energy zero-point, usually either 0 eV or the Fermi energy).

Other Calculation Types

pDOS – pDOS or projected density of states calculations enable the total density of states to be decomposed into contributions from the atomic wavefunctions of each atom of the unit cell. pDOS calculations can only be performed after SCF and NSCF calculations as with DOS calculations. pDOS input files are of the form shown below.

silicene_projwfc.in

```
&PROJWFC
 prefix= 'Silicene_pDOS', #must match prefix for the prior SCF and NSCF calculations.
 outdir= '.',
 filpdos= 'Silicene_pDOS.dat'
/
```

This program is executed using the command `$projwfc.x < silicene_projwfc.in > silicene_projwfc.out` The useful outputs of this calculation are .dat.pdos files for the atomic wavefunctions of each atom of the unit cell (2 Si atoms for silicene), collected into subshells. Each of these files contains the energies density of states is calculated at, along with the density of states contributions from each atomic wavefunction of the subshell (labelled pDOS) as well as the total contribution from all wavefunctions of the subshell (labelled lDOS). The order in which wavefunctions of the subshell are indexed is: p_z, p_x, p_y for the p subshell and d_{z2}, d_{zx}, d_{zy}, d_{x2-y2}, d_{xy} for the d subshell. Matplotlib.pyplot scripts can be used to plot this output data to obtain projected density of states plots.

Common problems

Using/configuring the VM

How can the VM be moved to a new directory?
To move a VM to a new directory, you cannot simply move the virtual disk image from which the machine is run. Doing so will trigger the error **'**Trying to open a VM config which has the same UUID as an existing virtual machine'**.** To avoid this, use the command Machine -> Move within VirtualBox.

How can the RAM of the VM be expanded?
Use the command Settings -> System and then use the slider to increase the RAM available to the VM.

NB To make any settings changes to a VM, it must be powered down. This means the machine must not be currently booted and cannot be in a 'saved state'.

How can the disk memory of the VM be expanded?
NB Before attempting any of the steps detailed below, backup your VM by taking a snapshot using the command Machine -> Take Snapshot within the booted VM (menu at the top of the window when the VM is booted).

First, use the command Tools -> Media, click on the virtual disk image you are running the VM from and use the sliding scale to increase the virtual disk size. Next, to enlarge the virtual disk partition, boot the VM and install the application GParted using the commands below.

```
$sudo apt update
$sudo apt install gparted
```

Then, open GParted from the list of applications, granting it access to the root directory using the password moritz. Right click on the row corresponding to the VM (labelled with File System 'lvm 2 pv' and Mount Point 'ubuntu-vg'. Click Resize/Move and use the sliding scale to resize the partition and grant all unallocated memory to the VM. Click Resize and then the green tick at the top of the application page to enact these changes.

Reboot the VM. To allow ubuntu to access the new disk memory you have made available, use the commands below.

```
$sudo su-  #grants root access to the user.
$lvs  #returns current disk memory.
$lvdisplay  #indicates path to logical volume (what needs to be expanded to increase disk
               memory) under LV Path.
```

`$lvextend -r -L+10G /dev/ubuntu-vg/ubuntu-lv` #increases disk memory by numerical value (adjust to how much you have increased disk space by).

To check disk space has been resized, open Disk Usage Analyzer from the applications page.

What is the best way to transfer files between the VM and the machine it is hosted on?
The easiest way to achieve this is to create a shared folder, the contents of which are mirrored between the host machine and VM. To do so, first create a folder on the host machine. Then, boot up the VM and select the menu option Devices -> Shared Folders -> Shared Folders Settings (from the menu at the top of the VM screen).

Select the Machine Folders entry at the centre of the pop-up box, and then click on the folder icon containing a green + sign. This will pop up an Add Share window. Click the downward arrow on the right-hand side of the Folder Path box and select the folder on the host machine you created earlier to mirror. This should populate the folder path and folder name boxes. Leave the Mount point box unfilled, and check the boxes labelled Auto-mount and Make Permanent, before pressing ok.

Open the terminal in the VM and run the command `$ls -l /media` This should show the name of the now mirrored folder (your folder name prefixed by sf_). To create an alias to this directory in the home directory of the VM, run the command `$ln -s /media/sf_vmShareData /home/max/vmShareData` replacing vmShareData with the name of your mirrored folder.

Using Quantum Espresso

How do you determine input crystal structure?
The easiest starting point is to search for a .CIF or .xyz file of the desired structure. A good reference database for 2D materials is the C2DB (Computational 2D Materials Database). Such files can be visualised using the free software VESTA. An input SCF file can be generated using this tool. However, you will likely want to tailor the output to your preferred settings, as outlined above. In addition, 'ibrav' will always be set to 0, denoting a free structure for which the lattice vectors must be given explicitly under CELL_PARAMETERS. For simplicity, it is preferable to replace this with a non-zero Bravais lattice index and specified lattice constants if known.

Why are energy values drastically different for the same structure using different pseudopotentials?
Energy values can differ wildly because all energies given are not referenced to the vacuum level, so the zero-point is pseudopotential specific. Therefore, only differences in energy (e.g. band gap = CBM (conduction band minimum) – VBM (valence band maximum)) have physical meaning and are not pseudopotential specific. Energy differences calculated using different pseudopotentials will vary depending on the pseudopotential type and XC functional but are comparable.

How do you determine the high symmetry k points for a band structure calculation?

The k-points path through the Brillouin zone can be determined using the SeeK-path tool. Note that this tool will produce an explicit list of all k-points to sample (i.e. a list of points each with weighting 1). In addition, the tool can produce repeating lists of k-points, not giving the most reduced form of the band structure and increasing computation time. Therefore, comparing the output of this tool with other references to try and create a minimal set of high symmetry points to sample between is the preferable approach.

How can you construct a supercell of multiple layers for a 2D material?

For 2D materials with lattice vectors a_1 and a_2 with only non-zero x and y components and lattice vector a_3 only having a non-zero z component, monolayers of the material largely lie in the xy plane (with some atoms above or below it) and the z axis is perpendicular to monolayer (when approximating it as flat). The easiest way to create a supercell is to modify celldm(3) (altering the z component of all atoms' positions) whilst leaving celldm(1) (and celldm(2) if specified) unchanged. We do this to expand the size of the unit cell to accommodate additional layers of the 2D material. This process is outlined below for platinum diselenide. Platinum diselenide is a 2D material with a hexagonal unit cell, with lattice vectors:

a_1 = a(1, 0, 0) a_2 = a(-0.5, sqrt(3)/2, 0) a_3 = a(0, 0, c/a)

Our initial SCF input file is of the form:

PtSe2_SCF.in

```
&CONTROL
  calculation = 'scf'
  prefix = 'PtSe2_Bands'
  outdir = '.'
  pseudo_dir = '../pseudos/'
  verbosity = 'high'
/

&SYSTEM
  ibrav = 4 #denotes hexagonal unit cell.
  celldm(1) = 7.082 #defines lattice point locations in the xy plane, value taken from
                     literature.
  celldm(3) = 4.700 #defines lattice point locations in the z plane, for monolayer, value set
                    using convergence testing to be sufficiently large to prevent spurious
                    layer self-interactions.
  nat = 3
  ntyp = 2
  ecutwfc = 50 #optimised using convergence testing.
/

&ELECTRONS
  mixing_mode = 'plain'
  mixing_beta = 0.7
  conv_thr = 1.0e-8
```

```
/

ATOMIC_SPECIES
Pt    195.084 Pt.pbe-n-kjpaw_psl.1.0.0.UPF
Se     78.96 Se.pbe-dn-kjpaw_psl.1.0.0.UPF

ATOMIC_POSITIONS crystal
Pt        0.0000000000     0.0000000000     0.4999982100
Se        0.6667059256     0.3333424276     0.5743557871
Se        0.3333424276     0.6666848552     0.4256406324

K_POINTS automatic
12 12 1 0 0 0
```

To modify this input to include 2 layers of PtSe2, we double the value of celldm(3) to double the height of the unit cell, we double the value of nat (because we have doubled the number of layers per unit cell) and we shift the z components of the atomic positions to account for our expansion of the unit cell.

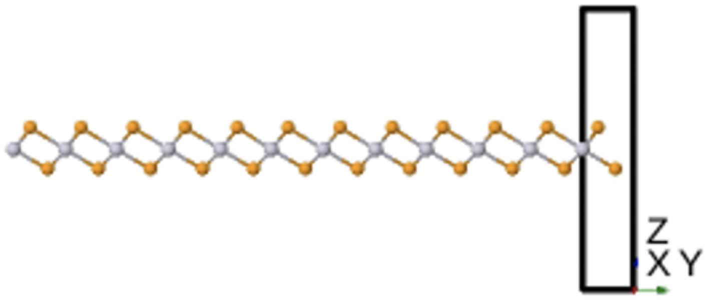
PtSe2_SCF_2Layers.in

```
…

&SYSTEM
 ibrav = 4 #denotes hexagonal unit cell.
 celldm(1) = 7.082 #unmodified because xy positions of atoms left unchanged.
 celldm(3) = 9.400 #doubled to double the height of the unit cell.
 nat = 6 #doubled to account for twice the number of atoms in the unit cell.
 ntyp = 2
 ecutwfc = 50
/

…

ATOMIC_POSITIONS crystal
Pt        0.0000000000     0.0000000000     0.2499991050
Se        0.6667059256     0.3333424276     0.2871778936
Se        0.3333424276     0.6666848552     0.2128203162 #z components halved to map
                                                            1st set of 3 atomic positions to
                                                            those of monolayer in original
                                                            input file.
Pt        0.0000000000     0.0000000000     0.4499991050
Se        0.6667059256     0.3333424276     0.4871778936
Se        0.3333424276     0.6666848552     0.4128203162 #z components all linearly
                                                           shifted by 0.2, to create
                                                           interlayer spacing whilst
```

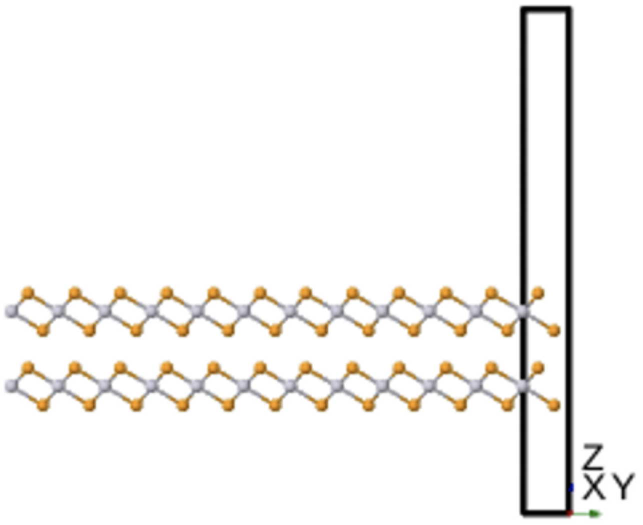
maintaining the same relative positions of atoms within each layer.

…

Whilst leaving the x and y components of the ATOMIC_POSITIONS unchanged, we halve the z components of the first 2 atoms to map them to the monolayer in the original input layer. We then linearly shift the z components of the 2nd set of atoms by the same amount, to ensure their relative positions are the same as for the first set. This value of the linear shift can be adjusted to reflect the actual inter-layer spacing (either by using a reference literature value or by repeating SCF calculations with a range of values and inspecting the change in total energy). You can visualise your modified SCF file using the SeeK-path tool, yielding images such as those shown below (which are captured in the yz plane).



Monolayer PtSe2



Bilayer PtSe2