# work function

This document explains how to calculate the work function of a surface, using Quantum Espresso.

The example case is the (001) surface of fcc aluminum, modeled by a slab with 3 layers of atoms and 3 layers vacuum. This is way too small to model a real surface (e.g. there is not a single atom in this slab that will feel itself to be in a bulk situation), but it gives a relatively fast example to calculate. All atoms in the slab are at their ideal bulk positions, no position optimization is done (which should be done when you want to calculate an accurate work function).

Rather than producing a Quantum Espresso input file for the surface cell starting from the unit cell of bulk aluminum (which could be done, no problem), you'll get right away a cif file for this surface slab:

```
data_global
_cell_length_a    2.854964
_cell_length_b    2.854964
_cell_length_c   12.112595
_cell_angle_alpha   90.000000
_cell_angle_beta    90.000000
_cell_angle_gamma   90.000000
_symmetry_space_group_name_H-M          'P 1'
_symmetry_space_group_number  1

loop_
_symmetry_equiv_pos_as_xyz
  'x,y,z'

loop_
_atom_site_label
_atom_site_type_symbol
_atom_site_fract_x
_atom_site_fract_y
_atom_site_fract_z
Al001  Al  0.00000000  0.00000000  0.33333333
Al002  Al  0.50000000  0.50000000  0.50000000
Al003  Al  0.00000000  0.00000000  0.66666667
```

Visualize this cell, and convince yourself that this is indeed a (001) surface for a fcc lattice with a lattice parameter of 4.037529 Å (if you would build this supercell yourself, there is a fair chance you might have found a cell with twice as many atoms for the same surface, i.e. a supercell – the one given here is the smallest possible cell for this slab).

Convert this into input for Quantum Espresso, with this pseudopotential, ecutwfc=60 Ry, ecutrho=300 Ry and a 25x25x5 k-mesh.

Run this calculation (it will take 5-10 minutes).

Why a x5 in the k-mesh and not a x25?

The bulk fcc Al crystal was tested before to be numerically converged with a 25x25x25 k-mesh. Imagine you are still modelling bulk fcc Al, but with a unit cell that that is a stack of 6 fcc cubes (no vacuum yet). The first Brillouin zone of that larger cell with be 6 times shorter in the direction that corresponds to the stacking of unit cell boxes. That means that 25 mesh points in that direction would be 6 times nearer together. This would make your calculation more precise, but a mesh with the orginal spacing was already tested to be sufficient. Hence, you spend too much calculation time. Therefore, by taking only 1/6 of all k-points in that direction, you'll sample the Brillouin zone equally well as for the original cell with 25 k-points. 25/6 is not an integer number, so take the first higher integer, which is 5. A 25x25x5 k-mesh gives us the same numerical quality for this large unit cell than a 25x25x25 mesh did for the original cell. This does not change when 3 layers of atoms are deleted – the size of the first Brillouin zone is determined by the dimensions of the cell, not by the atoms inside.

Then prepare this input file for pp.x , and give it any name you want (we'll call it ppinput.in in this document):

```
&INPUTPP
  prefix='reduced',
  outdir='.',
  plot_num=11,
  filplot='reduced.pot',
/

&PLOT
  iflag=3,
  output_format=3,
/
```

The 'prefix' should be the same as the one you used for the pw.x calculation. plot_num=11 means that the file 'filplot' will contain the sum of the potential due to the nuclei and the Hartree potential (cfr. The Kohn-Sham equations). In principle, one would want here to have the exchange-correlation potential added as well (that would be plot_num=1). However, this is a small contribution that needs a long distance to converge to its vacuum limit. Therefore, one makes often a larger error by including it than by omitting it.)

With iflag=3 we ask to produce output in three dimensions, in a particular format specified by output_format=3. As visualization is not our prime concern here, the &PLOT block is not very important.

Now run pp.x (the name of the output file can be freely chosen, we'll call it ppoutput.out here):

pp.x -input ppinput.in > ppoutput.out

The file specified by filplot (reduced.plot in this document) now contains the potential for every point of the unit cell (including for every point of the vacuum above the slab, which is part of the unit cell too). What we need for a work function calculation, is the value of the potential in the vacuum at a large distance above the surface. The largest distance to the surface, is in the center of the vacuum region – equally far away from the surface 'below' and 'above'.

If we are really far away from the surface, it will not matter where exactly in the plane halfway the vacuum we consider the potential: it will have the same value everywhere in that plane. But our slabs will often not be that large that this is true. Therefore, we better average the potential over that plane. That is done by using the Quantum Espresso program average.x . This is how an input file for average.x looks like (let's call it average.in here):

```
1
reduced.pot
1
600
3
3.8149
```

The first and the third line should always be '1' for this application (the first line is the number of files that will be averaged, and the third line is the weight of each file – we are working here with one file only). The second line is the name for the file that contains the quantity you want to average, which in this case is the reduced.pot file that contains the potential you just calculated.

The average.x program will ask you in the 5$^{th}$ line over which plane you want to average the potential (see hereafter). It will then make a large number of slices through the unit cell, always parallel with the chosen plane. The potential will be averaged within each of these slides.

The number of slices is specified by the 4th line (in our example, the unit cell has a length corresponding to 6 bulk layers of aluminum – if you take 100 slices between two subsequent layers, then you can follow the average with a good spatial resolution. Whence the 6x100=600 in the input file. It is always wise to test this: a plot of the averages along the unit cell should not change if the slices are taken closer to each other.

The 5th line contains the variable 'idir', which specifies the orientation of the plane – all slices will be taken parallel to that plane. The way of defining this is a bit cryptic:

idir = 1 : the chosen plane is parallel to the plane spanned by the lattice vectors **b** and **c**
idir = 2 : the chosen plane is parallel to the plane spanned by the lattice vectors **a** and **c**
idir = 3 : the chosen plane is parallel to the plane spanned by the lattice vectors **a** and **b**

As we need for the work function the average in a plane parallel to the surface, we must make sure we use a supercell where two lattice vectors span a plane that is parallel to the surface. Otherwis this averaging procedure will not give us a useful number. In our example, the plane spanned by the **a** and **b** lattice vectors is parallel to the surface, and therefore idir=3 is the right choice.

The 6th line contains another parameter to eliminate spurious effects. The sequence of averaged potential values will often show small oscillations, due to interference effects between the two surfaces that border the vacuum. These are artefacts : for a true surface there is only vacuum above, hence we don't want these oscillations. Therefore, we will average the potential averages over several slices. A typical size for the number of slices, is the distance between two layers (in atomic units). In our example, this is 2.0187 Å or 3.8149 au.

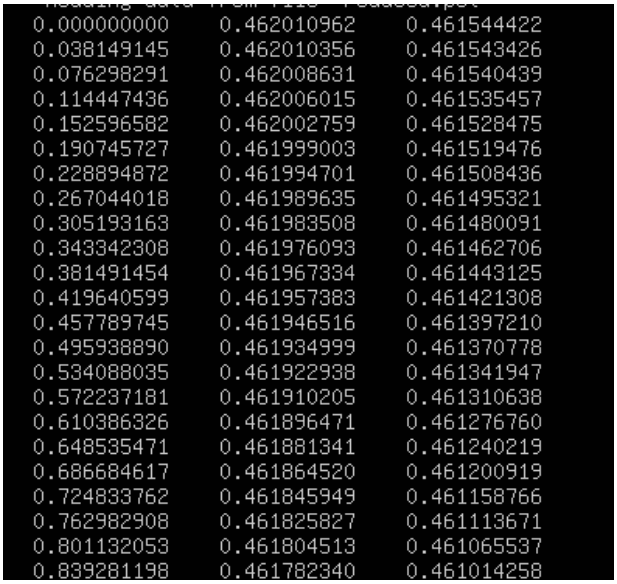
Now you can run average.x (specify an output file with the name of your choice, here it is called average.out) :

average.x -input average.in > average.out

This will produce apart from the output file average.out also the file **avg.dat**, with three columns:

- The distance in au along the line through the unit cell
- The potential averaged over the slice through that point
- The averages over all slices within the given value (3.8149 au), where the current point is at the center (i.e. we average over all slices that are 3.8149/2 au before and after the current point). This is a sliding window type of averaging.

These two snapshots show the beginning and the center of the unit cell :

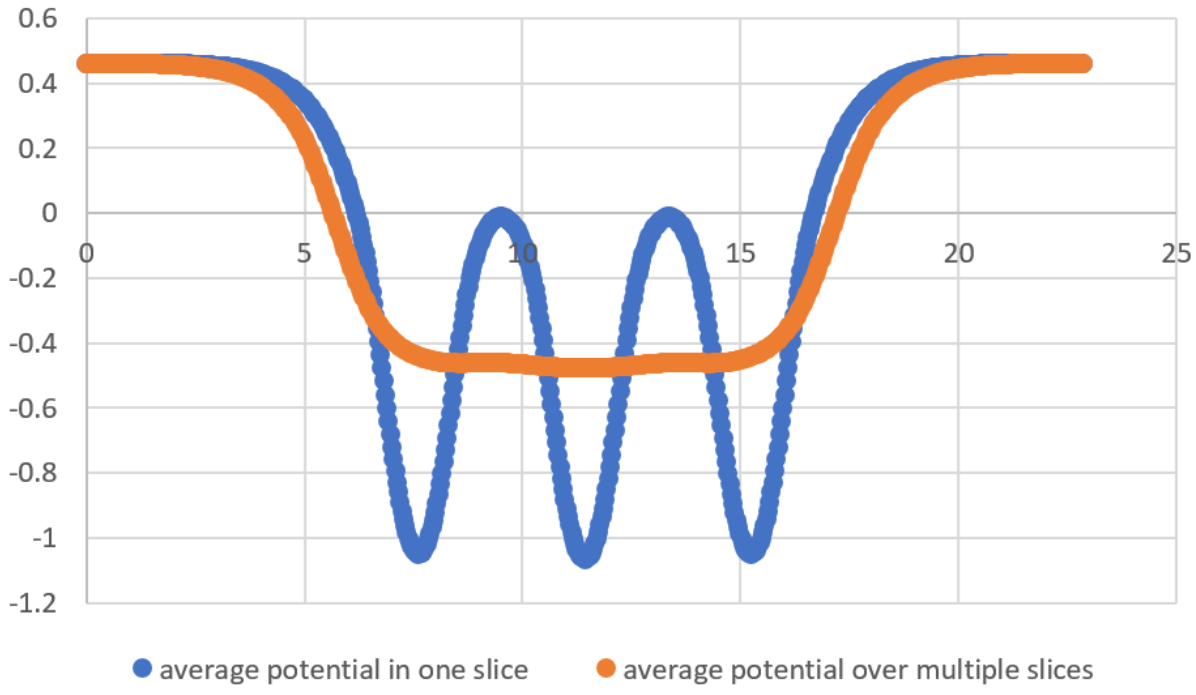


| | | |
|---|---|---|
| 0.000000000 | 0.462010962 | 0.461544422 |
| 0.038149145 | 0.462010356 | 0.461543426 |
| 0.076298291 | 0.462008631 | 0.461540439 |
| 0.114447436 | 0.462006015 | 0.461535457 |
| 0.152596582 | 0.462002759 | 0.461528475 |
| 0.190745727 | 0.461999003 | 0.461519476 |
| 0.228894872 | 0.461994701 | 0.461508436 |
| 0.267044018 | 0.461989635 | 0.461495321 |
| 0.305193163 | 0.461983508 | 0.461480091 |
| 0.343342308 | 0.461976093 | 0.461462706 |
| 0.381491454 | 0.461967334 | 0.461443125 |
| 0.419640599 | 0.461957383 | 0.461421308 |
| 0.457789745 | 0.461946516 | 0.461397210 |
| 0.495938890 | 0.461934999 | 0.461370778 |
| 0.534088035 | 0.461922938 | 0.461341947 |
| 0.572237181 | 0.461910205 | 0.461310638 |
| 0.610386326 | 0.461896471 | 0.461276760 |
| 0.648535471 | 0.461881341 | 0.461240219 |
| 0.686684617 | 0.461864520 | 0.461200919 |
| 0.724833762 | 0.461845949 | 0.461158766 |
| 0.762982908 | 0.461825827 | 0.461113671 |
| 0.801132053 | 0.461804513 | 0.461065537 |
| 0.839281198 | 0.461782340 | 0.461014258 |

Center of the unit cell:

| | | |
|---|---|---|
| 10.834357287 | -0.747102628 | -0.477165592 |
| 10.872506433 | -0.783698395 | -0.477402974 |
| 10.910655578 | -0.818694206 | -0.477621895 |
| 10.948804723 | -0.851860182 | -0.477823135 |
| 10.986953869 | -0.883019599 | -0.478006890 |
| 11.025103014 | -0.912041821 | -0.478173068 |
| 11.063252159 | -0.938827488 | -0.478321651 |
| 11.101401305 | -0.963292165 | -0.478452983 |
| 11.139550450 | -0.985354738 | -0.478567878 |
| 11.177699596 | -1.004934369 | -0.478667523 |
| 11.215848741 | -1.021955974 | -0.478753209 |
| 11.253997886 | -1.036360557 | -0.478825993 |
| 11.292147032 | -1.048114708 | -0.478886435 |
| 11.330296177 | -1.057214075 | -0.478934502 |
| 11.368445322 | -1.063678264 | -0.478969668 |
| 11.406594468 | -1.067538362 | -0.478991193 |
| 11.444743613 | -1.068821512 | -0.478998449 |
| 11.482892759 | -1.067538362 | -0.478991193 |
| 11.521041904 | -1.063678264 | -0.478969668 |
| 11.559191049 | -1.057214075 | -0.478934502 |
| 11.597340195 | -1.048114708 | -0.478886435 |
| 11.635489340 | -1.036360557 | -0.478825993 |
| 11.673638486 | -1.021955974 | -0.478753209 |
| 11.711787631 | -1.004934369 | -0.478667523 |
| 11.749936776 | -0.985354738 | -0.478567878 |
| 11.788085922 | -0.963292165 | -0.478452983 |

And this is a plot of the two averages along the unit cell:




Now we come to our final answer for the potential in the vacuum: as the center of the vacuum in our example was at r=0 (or equivalently at the maximal value of r), we can read the potential in the vacuum as the first (last) value of the third column of the output of average.x : 0.4615 Ry.

Calculating the work function is now straightforward:

Read the Fermi energy from the output file of the slab calculation:

```
   End of self-consistent calculation

   Number of k-points >= 100: set verbosity='high' to print the bands.

   the Fermi energy is     1.8450 ev

   total energy              =    -118.43713066 Ry
```

work function = potential in the vacuum minus Fermi energy of the slab
= 0.4615 * 13.6058 – 1.845 eV
= 4.4 eV

Given our quite crude approximations, this is not far away from the experimental value for the work function of aluminum, which is 4.06 to 4.26 eV (https://en.wikipedia.org/wiki/Work_function)