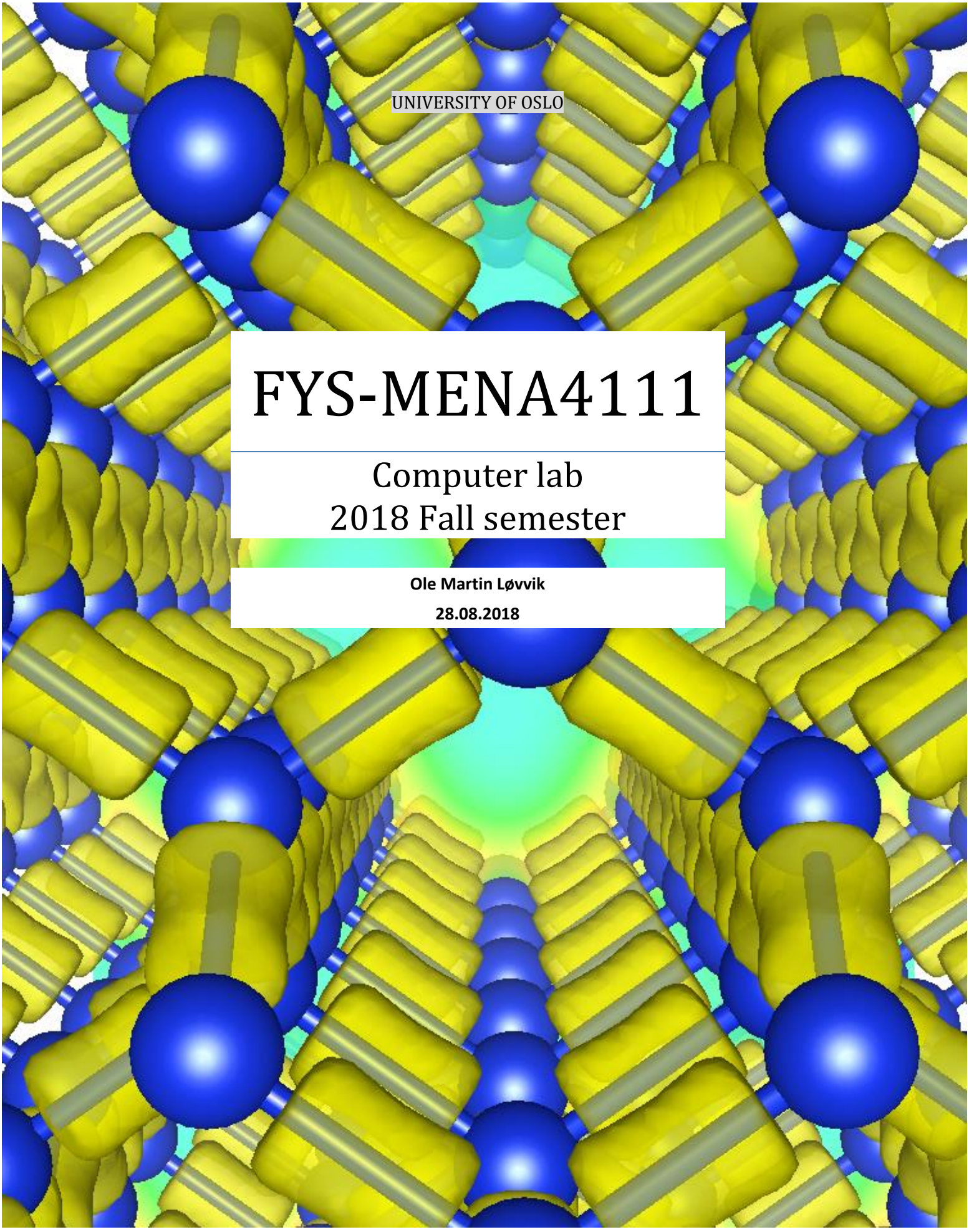


UNIVERSITY OF OSLO

FYS-MENA4111

Computer lab
2018 Fall semester

Ole Martin Løvvik

28.08.2018

Lab 1. Introduction to VASP

Density functional theory (DFT) is a very powerful tool which, when used properly, can give detailed and reliable insight into fundamental properties of materials and nanostructures. It has for this reason become tremendously popular and is now one of the most common tools in materials science. However, the theory also has its limitations, and using it without care can be misleading or even wrong in some cases. It is therefore crucial to be aware of such limitations; not only when using DFT in projects on your own, but also when assessing results and claims presented by other researchers. One of the central goals of this lab course is to provide insight in some general possibilities and limitations of DFT by hands-on examples.

When DFT is used to predict properties of a materials system in a project, a few steps should always be performed initially. This will be treated in the first labs of the course:

1. Prepare initial input files. (Lab 1.)
2. Run the DFT program once and check that the starting point is reasonable. (Lab 1.)
3. Check convergence of numerical input parameters. (Lab 2.)
4. Decide which input parameters should be used for various calculations and applications. (Lab 2.)
5. Relax the crystal structure(s) by minimizing interatomic forces. (Lab 3.)

After completing of these steps, the work can spread in several different directions depending on the project goals; this will be the topic of the remaining labs.

We will in this course demonstrate how these tasks typically are performed with a state-of-the-art code designed to calculate the electronic band structure of materials and nanostructures using DFT. The code is called VASP (Vienna *ab initio* Simulation Package) and is perhaps the most well-known and appraised band structure code on the market.

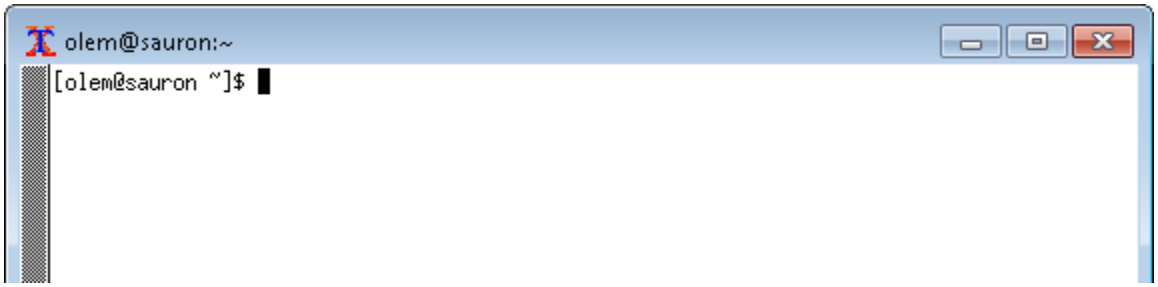
In order to run the code, you will have to know a few commands of the linux/unix language. We will use the machine abel.uio.no to run the program; this is UiO's number cruncher with enough capacity to run complicated programs like VASP in a very efficient manner. You can find more information about abel on http://www.uio.no/english/services/it/research/hpc/abel/. In order to interpret and visualize the results, you will gain from a number of supporting programs; they will be presented as you need them. All important commands are summarized on a separate reference card.

1 Start a terminal window and log on

Unix is based on commands written to a terminal window. If you are using a unix, Linux, or OS X operative system, this is integrated in the system, and all you have to do is open a terminal window from the menu. If you are using a Windows computer, you have to install an extra program in order to get

access to such a window, and you usually need to have internet access to an external computer on which you can run your commands. You can e.g. use X-Win32 (UiO has site license on this program), the combination of Xming and Putty, which are both freeware, or the free MobaXterm program.

The datalab computers all have Linux installed and you simply have to start a terminal window via the menu. When this has started, you will get something resembling this:



In the following, commands to the terminal window will be written as follows. The line following the command shows the output; in this case it is the date and time (do not type the > prompt!).

```
> date
Tue Sep  4 12:22:27 CEST 2018
```

We are going to do most of our work on the supercomputer abel, so we first need to log on to this machine. Simply type your UiO password at the prompt.

```
> ssh -Y abel.uio.no
olem@abel.uio.no's password:

Welcome to Abel, a supercomputing service by UiO/USIT/UAV/ITF

################################################################################

Abel login nodes run an automatic nice daemon so that your actions do not
impact other users.  Any process consuming more than 30 minutes of CPU time
will automatically be killed.

################################################################################

< some more text>
login-0-1:~$
```

The very first time you log in, you are asked to provide passphrases etc. Simply type <enter> two-three times until you see a prompt similar to that above.

The screen may look slightly different, but you should now hopefully be logged on to abel!

2 Basic commands in unix

It is necessary to know a few useful unix commands. If you are familiar with unix/linux, you can skip to the next section.

To list the files and directories at your current working directory, use ls (**l**ist file**s**):

```
> ls
fil1 fil2 fil3
```

You should organize all your work in directories (folders). Create new directories with the mkdir command (**m**a**k**e **dir**ectory):

```
> mkdir test
> ls
fil1 fil2 fil3 test
```

Go to another directory with cd (**c**hange **d**irectory):

```
> cd test
> ls
```

To find our where you are located in the file tree, use pwd (**p**ath of **w**orking **d**irectory):

```
> pwd
/usit/abel/u1/olem
```

(Your user name will be shown here.) An empty file is created with touch:

```
> touch foo
> ls
foo
```

Changing names or moving files and directories are done with mv (**m**o**v**e):

```
> mv foo bar
> ls
bar
```

Use cp (**co**p**y**) to create a copy of a file:

```
> cp bar bar2
> ls
bar bar2
```

Files are removed with rm (**re**m**o**ve). Use * as wildcard to match any set of characters:

```
> rm bar*
bar: ? (y/n) y
bar2: ? (y/n) y
> ls
>
```

Depending on the default, you will be asked whether it is OK to remove the files or not. If prompted, you must type ‘y’ to remove the file. Be aware: there is no Recycle bin in Unix! If a file has been removed, it is often very difficult to get it back. (At UiO we have a backup system, so it is possible to get back yesterday’s version of a file if this is important.)

In unix, the directory above the current one has a specific name: .. (two periods). This can be used to move around in the file system:

```
> pwd
/usit/abel/u1/olem/test
> cd ..
> pwd
/usit/abel/u1/olem
```

Directories are removed with the command rmdir (**re**m**o**ve **dir**ectory):

```
> ls
```

```
  fil1 fil2 fil3 test
  > rmdir test
  > ls
  fil1 fil2 fil3
```

Files and directories can be copied from abel to e.g. your home folder at UiO (M:) with the command scp (**s**ecure **c**o**p**y):

```
  > scp fil* login.uio.no:
fil1 100%    0     0.0KB/s   00:00
fil2 100%    0     0.0KB/s   00:00
fil3 100%    0     0.0KB/s   00:00
```

(You may have to provide your UiO password.)

3 Your first session on abel

To set things up properly, you need to copy a couple of files to your home directory. This should only be done the first time you log on, later you can go directly to the real work. First make sure you are located at the highest level of your home directory by writing cd without any argument:

```
  > cd
  > pwd
  /usit/abel/u1/olem
```

Then copy two files which will be read every time you log on to abel:

```
  > cp ~olem/fm4111/.bashrc .
  > cp ~olem/fm4111/.bash_profile .
```

The period designates the present (working) directory. The ~ character designates a home directory. ~olem translates to /usit/abel/u1/olem, while simply ~ translates to your own home directory. Mind that even if you write ls, you will not see this file, since files starting with a period are hidden:

```
  > ls
  fil1 fil2 fil3
```

But if you use the option -a, ls will show you all files and directories:

```
  > ls -a
  .   .bash_history  .bashrc  fil2  .forward  .Xauthority
  ..  .bash_login    fil1     fil3  .ssh
```

To ensure that the content of .bashrc is read in, you could log out and then in again. Slightly more elegant is the following solution:

```
  > source .bashrc
```

This loads the content of the file as if you logged on. You should now be ready to start your first DFT calculation on the supercomputer.

4 Prepare the first VASP calculation: bulk Si

VASP needs four input files in order to work properly: POSCAR, INCAR, KPOINTS, and POTCAR. In addition, a job script is needed to submit the job to the queue system; we usually call this "jobfile". We will now go briefly through these files to make you able to run your first job. They are available on the directory ~olem/fm4111/datalab/si_bulk . We have chosen bulk Si as the first system; this is a well-studied system with most properties well established.

First create a new directory (mkdir) where you can keep the files of this calculation. Then copy the 5 necessary files (cp) from ~olem/fm4111/datalab/si_bulk to this new directory. You can now look at the files using the *more* command (you can also choose to use the *less* command):

```
> more POSCAR
Si_bulk_icsd_60385
    1.0
     5.43101    0.0        0.0
     0.0        5.43101    0.0
     0.0        0.0        5.43101
…
```

Much more information about the input files can be found in the online guide:
http://cms.mpi.univie.ac.at/vasp/vasp/vasp.html

POSCAR. This is the most important file, defining the crystal structure (atomic positions and unit cell parameters) of your system. A typical POSCAR file looks like this:

```
Si_bulk_icsd_60385
    1.0
     5.43101    0.0        0.0
     0.0        5.43101    0.0
     0.0        0.0        5.43101
   Si
   8
Direct
  0.00  0.00  0.00
  0.00  0.50  0.50
  0.50  0.00  0.50
  0.50  0.50  0.00
  0.25  0.25  0.25
  0.25  0.75  0.75
  0.75  0.25  0.75
  0.75  0.75  0.25
```

The lines are in a fixed order, with the following content:

1. A comment line. Specify what it is, and where you got it from.
2. This line contains a scaling factor with which the next three lines will be multiplied. We usually leave it at 1.0.

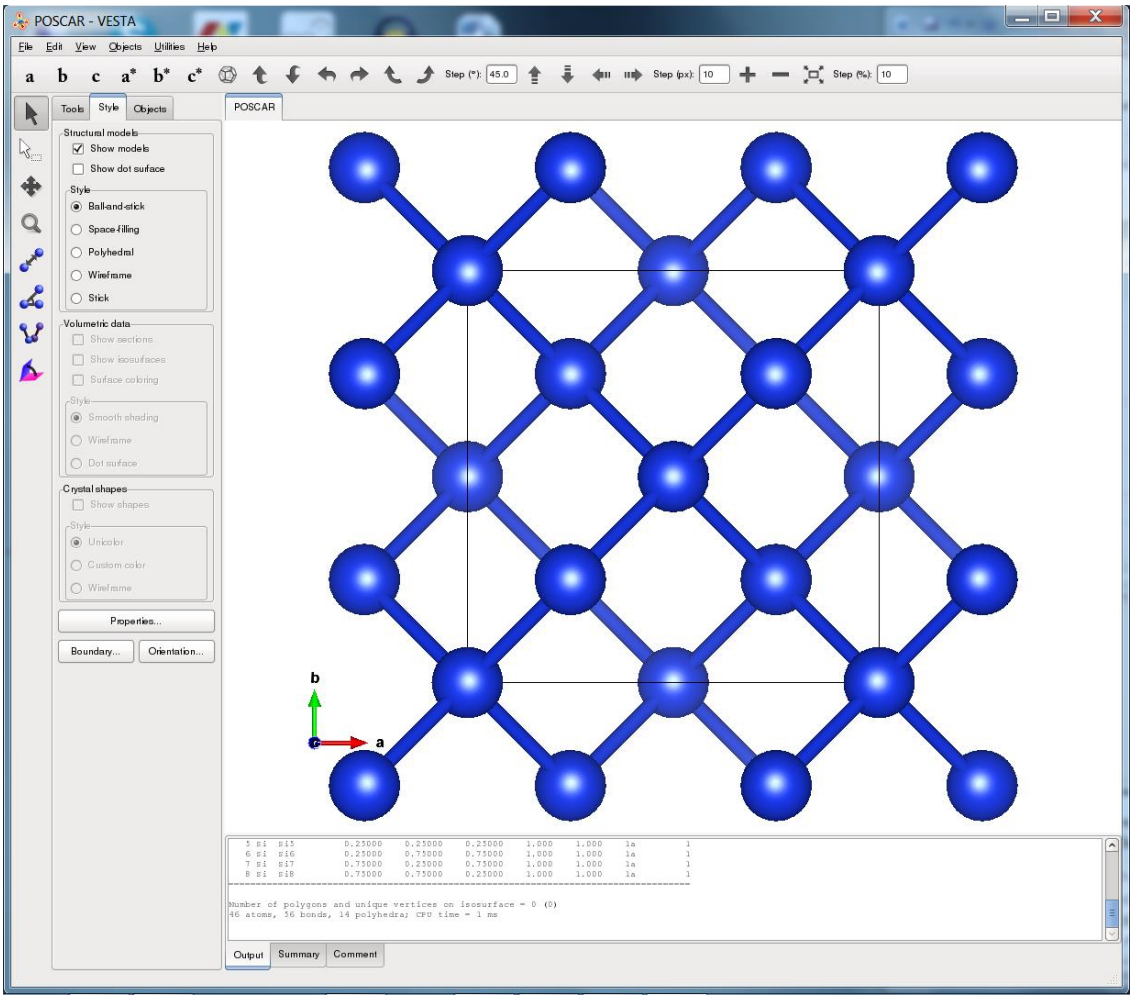
3-5. These three lines define the unit cell vectors a, b, and c in Å. Line 3 contains the x, y, and z components of vector a, line 4 those of vector b, and line 5 those of vector c. In the example above, the unit cell is cubic with a=5.43101 Å.

6. The elements making up the crystal structure. In this case, there is only one element: Si.
7. The number of atoms for each of the elements on line 6. The cubic unit cell of bulk Si contains 8 atoms.
8. Specifies how the atomic coordinates will be represented. There are only two possibilities:
   - *Direct* gives the coordinates in fractions of the unit cell vectors, in units of 1.
   - *Cartesian* specifies the coordinates as Cartesian, in units of Å.

9-End. The remaining lines give the atomic coordinates. The number of lines should match the number of atoms specified in line 7.

It can often be nice to look at the crystal structure, to make sure it is correct. We have at abel access to the VESTA program, which is convenient for this purpose (see also http://jp-minerals.org/vesta/en/). Simply type vesta, and you have the possibility to open the POSCAR file with File->Open... (you may have to find it in the file tree):

```
> vesta
```



We will come back to what we can do with VESTA at a later stage; now you should try to rotate the structure and get a feeling of how it looks like in space.

INCAR. This file tells VASP what to do, and how to do it. It can be very short or quite long, but it has to exist. The following example is quite typical; here some important numerical parameters are defined, and some output options are specified. We will not go into detail at this stage; simply copy the file and use the content without any changes. Only two examples will be mentioned here. EDIFF gives the criterion for self-consistence – here it is defined to be 1E-6 eV, which means that the difference in calculated total energy or band energies should be less than 10^{-6} eV from one iteration to the next. ENCUT specifies the

energy cut-off of the plane wave expansion. The plane waves constitute the basis set for the variational method, and are expanded with increasing frequency. Usually the cut-off is specified as the energy corresponding to such frequencies. In this case, the energy cut-off is 250 eV, which is relatively low – we will come back to this later on.

```
INCAR for high-accuracy total energy calculation.

! Electronic relaxation
ALGO   = Fast      ! Algorithm for electronic relaxation
NELMIN = 4         ! Minimum # of electronic steps
EDIFF  = 1E-6      ! Accuracy for electronic groundstate
ENCUT  = 250       ! Cut-off energy for plane wave expansion
PREC   = Normal    ! Low/Normal/Accurate
LREAL  = Auto      ! Projection in reciprocal space?
ISMEAR = -5        ! Smearing of partial occupancies.
SIGMA  = 0.2       ! Smearing width
ISPIN  = 1         ! Spin polarization?

! Ionic relaxation
NSW    = 0         ! Final high-accuracy calculation without relaxation

! Output options
LWAVE  = .FALSE.   ! Write WAVECAR?
NEDOS  = 2101      ! Number of grid points for DOS
EMIN   = -9        ! Minimum energy for evaluation of DOS
EMAX   =  12       ! Maximum energy for evaluation of DOS
LELF   = .TRUE.    ! Electron localization function
RWIGS  = 1.11      !
LORBIT = 11        ! Write LDOS to DOSCAR + PROCAR

! Memory handling
NPAR    = 1
```

KPOINTS. Many of the numerical integrals will be performed in reciprocal space ("**k** space"), after a Fourier transform of the wave functions. This file specifies how the integration points should be distributed. It has a few different formats, but we will for most of the time use the following, which is an automatic distribution of points according to the numbers given. We will come back to this file also more closely at a later stage; here we only note that the most important line is number 4, where the number of k points in each direction is given. In this case a 5×5×5 grid is used.

```
Automatic
0
Gamma
 5 5 5
 0  0  0
```

POTCAR. How is the exchange-correlation functional represented? And how is the core treated? This is specified in POTCAR, which defines the functional numerically. We will in this course primarily use a generalized gradient approximation (GGA) made by Perdew, Burke, and Ernzerhof (PBE). The core is here represented by the projector augmented wave (PAW) method, which is a very efficient and accurate way to take care of the core region. You can check which potential is used with the pot command:

```
> pot POTCAR
POTCAR PAW_PBE  Si
```

We will never manipulate the POTCAR files, only copy them from a library. The "potentials" contained in these files are universal, and are not fitted to any experimental quantities. This is the reason why DFT often is called *ab initio*; from first principles.

jobfile. This is the job script, which is only used to submit the job to the batch queue. On abel the SLURM system is used for job administration; a brief introduction to this and to how it is used on abel can be found here: https://wiki.uio.no/usit/suf/vd/hpc/index.php/Abel_User_Guide . This page also serves as a reference if you need additional information. We are not going to change anything on the jobfile file right now, but will nevertheless take a look at the first lines of the file:

```
#!/bin/bash
# Specify jobname:
#SBATCH --job-name=si_bulk
# Specify the number of nodes and the number of CPU's (tasks) per node:
#SBATCH --nodes=1 --ntasks-per-node=4
# Specify account.
#SBATCH --account=ln0003k
# The maximum time allowed for the job, in hh:mm:ss
#SBATCH --time=00:10:00
```

The # character is usually starting comments in shell scripts like this. However, the SLURM system looks for the keyword SBATCH after #, which can then be used to place the job properly in the queue system. Most of the keywords are self-explanatory, e.g. the *--job-name*. It is wise to change the job name for new jobs, since it makes it easier to see the difference between different jobs when you are running more than one at a time. The *--time* tag is used to specify the maximum amount of time the job should use. When you are running more advanced calculations later on, this needs to be changed to higher than 10 min. For lager jobs it may also be convenient to increase the number of CPU's ("tasks") dedicated to the job – it is now 4 CPU's by default. Each node has 16 available CPU's on abel.

5 Submit a job to the queue

When all the files have been prepared, you can submit the job to the queue system. This is done with the sbatch command:

```
> sbatch jobfile
sbatch: Submitted batch job 4422623
```

Since we are going to write this command several times, an alias has been created – it is enough simply to write sub in order to submit a job (if you already submitted the job, you should not do it again now):

```
> sub
sbatch: Submitted batch job 4422623
```

You can se if the job has been successfully submitted with the squeue command. An alias has been created for this as well; this is called st (the status of the queue):

```
> st
  JOBID     NAME       TIME  NODES   CPUS TIMELIMIT NODELIST(REASON)
4422669  si_bulk       0:00      1      4 10:00 c-4-29
```

The entries should be pretty self-explanatory in this case also.

If you waited too long before writing st, you will not see the last line; this means that the job has already finished. When this glorious moment appears, you will see that a few more files have been created:

```
> ls
CHG       DOSCAR    IBZKPT   KPOINTS  PCDAT   PROCAR              WAVECAR
CHGCAR    EIGENVAL  INCAR    OSZICAR  POSCAR  slurm-4422669.out   XDATCAR
CONTCAR   ELFCAR    jobfile  OUTCAR   POTCAR  vasprun.xml
```

You have successfully performed your first DFT calculations, and the time has come to look at the output files.

6 Analyze the output

You should always check at least three output files after a calculation finished to ensure that everything went well: OSZICAR, slurm-JOBID.out, and OUTCAR. In addition, there are several files with more detailed information that we will look more closely at in the following. There is a lot of information contained in the output files of even this simple job (more than 8 Mb). We will only look at a small part of this, but it should give you the first taste of some of the powers of DFT.

OSZICAR. First check the electronic convergence of the self-consistency loop. This is contained in the file OSZICAR:

```
>  more OSZICAR
       N       E                     dE             d eps       ncg     rms          rms(c)
DAV:   1     0.434038181093E+02    0.43404E+02   -0.79690E+03   420   0.692E+02
DAV:   2    -0.401213073843E+02   -0.83525E+02   -0.79250E+02   560   0.130E+02
DAV:   3    -0.439506863363E+02   -0.38294E+01   -0.37857E+01   560   0.271E+01
DAV:   4    -0.440124622734E+02   -0.61776E-01   -0.61650E-01   532   0.420E+00
DAV:   5    -0.440140804018E+02   -0.16181E-02   -0.16175E-02   521   0.532E-01    0.600E+00
RMM:   6    -0.435184806159E+02    0.49560E+00   -0.27596E-01   423   0.220E+00    0.358E+00
RMM:   7    -0.432919738023E+02    0.22651E+00   -0.60517E-01   426   0.335E+00    0.299E-01
RMM:   8    -0.432972782735E+02   -0.53045E-02   -0.18068E-02   460   0.617E-01    0.444E-02
RMM:   9    -0.432976709819E+02   -0.39271E-03   -0.12937E-03   487   0.173E-01    0.218E-02
RMM:  10    -0.432977443777E+02   -0.73396E-04   -0.23923E-04   466   0.833E-02    0.660E-03
RMM:  11    -0.432977360140E+02    0.83637E-05   -0.16795E-05   485   0.200E-02    0.169E-03
RMM:  12    -0.432977357011E+02    0.31286E-06   -0.96947E-07   320   0.515E-03
   1 F= -.43297736E+02 E0= -.43297736E+02  d E =0.000000E+00
```

When a calculation is finished, this is the first thing you do: check that the job has converged properly. We see that 12 iterations were needed to reach self-consistence – that is, both dE (difference in calculated total energy E between two iterations) and d eps (difference in band energies) should be less than 1.0E-6 eV. This was specified in the INCAR file, as noted above. After self-consistence has been achieved, a line is printed with the resulting total electronic energy E0.

slurm-JOBID.out. Messages about errors, warnings (and a few successes) etc. are printed to the standard output and redirected to the slurm-JOBID.out file. The JOBID number is the same you see with squeue. If

you try to run a job several times there might be several slurm files in a directory; the one with the highest JOBID number is the most recent one.

OUTCAR and vasprun.xml. A lot more information is contained in the OUTCAR file. Much of the same is found in vasprun.xml in a more machine-readable format, and we will focus on OUTCAR here. You should open the OUTCAR file in a text editor (or just scroll through using *more* or *less*) to get familiar with the structure of the file when a job has finished without errors. You can search for text strings within *emacs* using the ctrl-s command. Within *more* or *less* you can use "/".

A powerful way to search for keywords in a file is the *grep* command. It has many opportunities, but can often be used straightforward like this:

```
> grep TITEL OUTCAR
TITEL  = PAW_PBE Si 05Jan2001
```

Some properties are always of interest, and searching for them have been combined in a convenient script called *vaspout*:

```
> vaspout OUTCAR
MxForce Atom #  Pressure        Drift   TOTEN           Filename
0.000   1       18.020000       0.000   -43.297736      OUTCAR
```

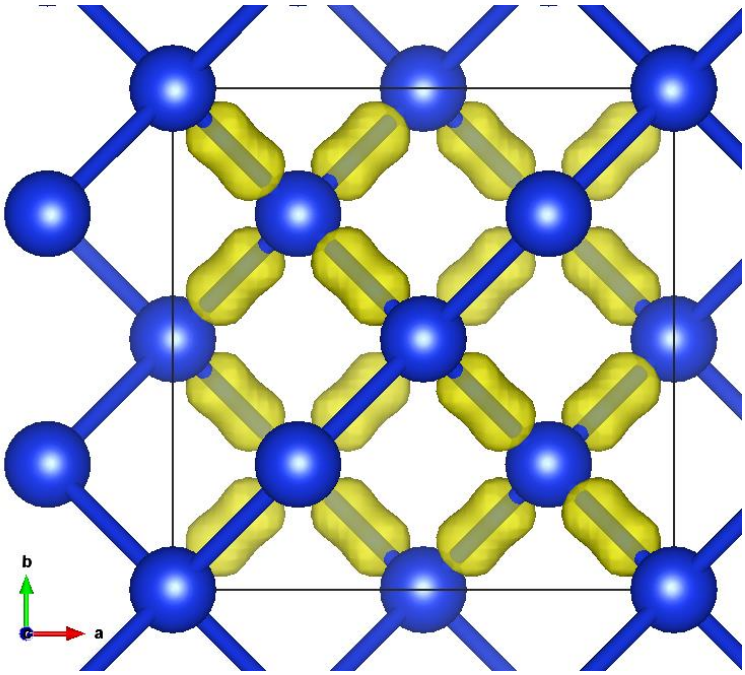
In addition to the total energy (TOTEN) in eV, this prints out the maximal interatomic force (MxForce) in eV/Å, the atom number (from the list in POSCAR) experiencing this force, the calculated electrostatic pressure in kbar, and the maximal drift in forces due to numeric inaccuracy at the unit cell boundaries (Drift). The latter gives an estimate of the uncertainty of the calculated forces, and should always be less than the force convergence criterion when performing ionic relaxation (see later sections).

The total energy TOTEN is among the most important results from such a calculation. It quantifies the energy difference between infinitely separated atoms and the crystal structure specified in POSCAR. In this case, the 8 atoms in POSCAR have gained 43.297736 eV by forming the crystal structure of Si, that is approximately 5.4 eV/atom.

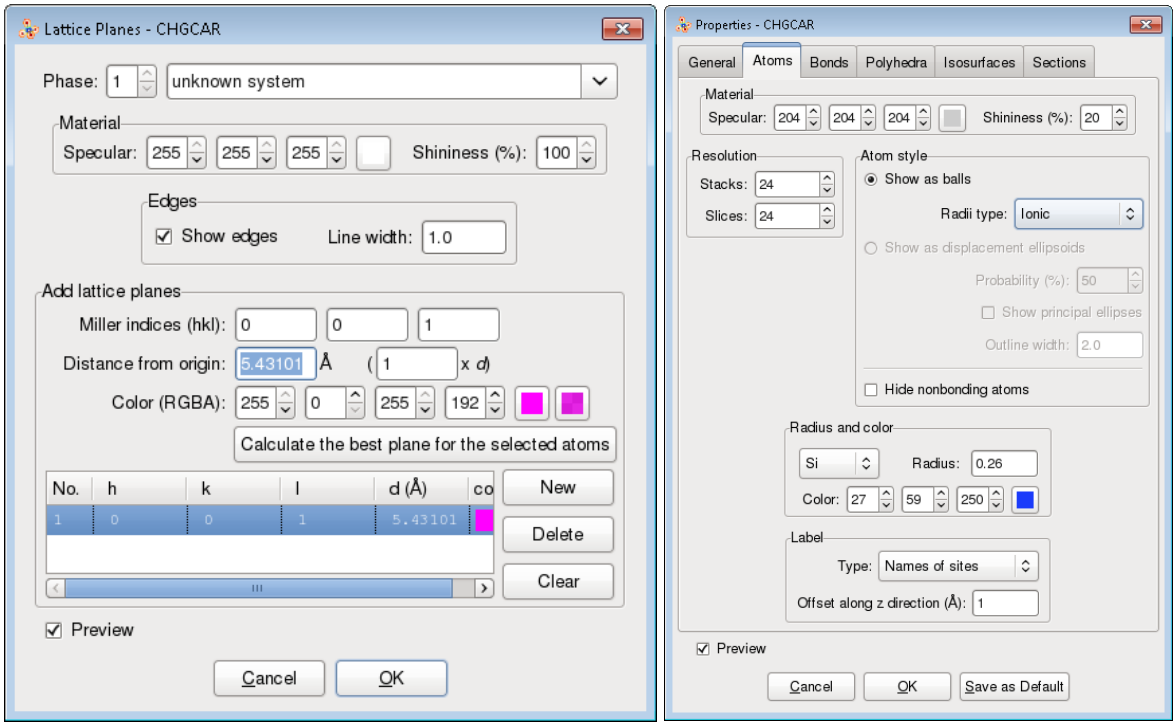
No temperature effects or other contributions involving the movement of nuclei have been included in the calculation of TOTEN. The correct name of this number should thus be the total *electronic* energy. You will often see in the literature that the word *electronic* is taken for granted.

Note that the accuracy of the calculated numbers is often much less than the impression one can get from the number of digits presented in the output. As an example, the total energy usually has a numerical uncertainty of at least 1 meV, depending on the choice of parameters (cut-off energy etc.) This means that you should never specify more than 3 digits after the decimal point for this number: E_{Tot} = -43.298. You can find several sad examples of EUD (excessive use of digits) in the literature.

CHGCAR. The charge density is of course calculated by a (charge) density functional theory code like VASP, and it can often be quite instructive to look closer on this. For this we can also use the VESTA program. Start VESTA again, and open the file CHGCAR (File->Open…):

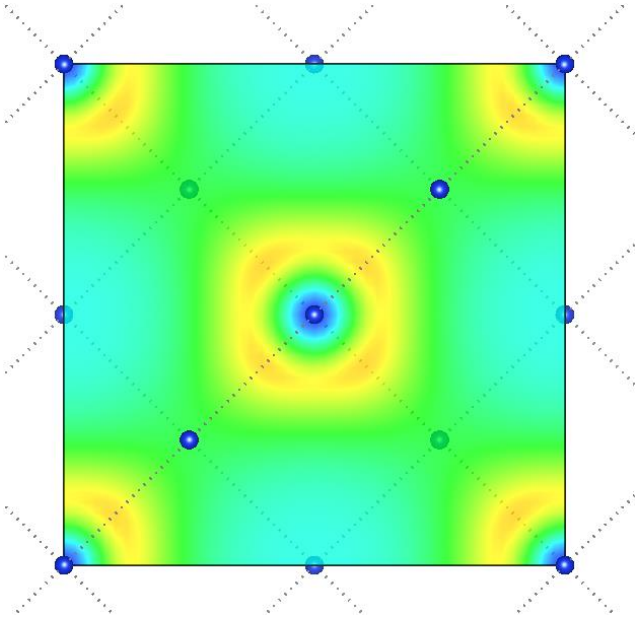


It is slightly difficult to see clearly how the electrons are distributed in this view. One way to visualize this is to add a lattice plane on which the electron density is plotted as a contour plot. Go to Edit -> Lattice Planes... and add a 0 0 1 lattice plane:



Also change the atomic radius to make the balls smaller: Properties -> Atoms -> Atom Style -> Show as balls, Radii type: Ionic. Also turn off "Show Isosurfaces" for now. And finally, only draw the bonds as

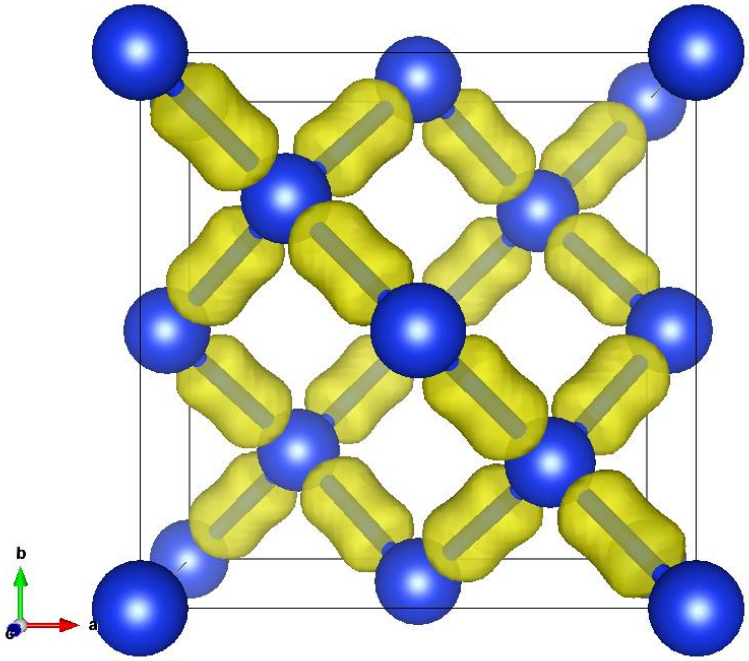
dotted lines: Properties -> Bonds -> Bond style -> Dotted line. This gives a view of the electronic structure as follows:



We can clearly see that bonds are pointing out in four directions from the middle atom, where the electron density is depleted.

Another way to look at the electronic structure is to use isosurfaces. Turn on isosurfaces again, and delete the lattice plane. Then change the level you want to show by going to Properties -> Isosurfaces-> Isosurface level. Play around with different values to see how the bonds are distributed. The default value around 0.07 is quite good.

Sometimes it can be good to have perspective turned on (View->Perspective). Also, you may want to display only the atoms inside the unit cell (Edit -> Bonds -> Boundary mode -> Do not search…) When all this has been specified, the structure may look like this:



It is now very instructive to rotate the structure again; one can clearly see the bonds between the Si atoms, and much less electron density is located at the atomic sites. This is typical for a covalent bond.

When you need to export a figure e.g. for a report this can be done with File->Export Raster Image… This can be copied to your computer with scp (either within linux like in Section 2 or e.g. using the WinSCP program from a windows computer).

7 Log out

Remember to log out from abel when you are finished. This can be done with the exit command:

```
> exit
```

8 Report

Submit a short report (1-2 pages) to Canvas with the following contents:

- A brief introduction
- Summary of important information from the output files:
  - Which warnings did you find in the slurm output file?
  - From OUTCAR: how many primitive cells build up your supercell? How many irreducible k-points were used in the calculations? What was the shortest nearest neighbor distance of the structure? What was the number of bands (NBANDS) and total number of plane-waves (NPLWV)? What was the total CPU time used? What was the maximum memory used?
- A few (2-3) pictures from Vesta similar (but not identical) to those above.
- A brief conclusion