# Excercise sheet 5

**General remark:** If you get an error that a libmkl or similar cannot be found, run the command

module load intel64

in putty (or the terminal) while being logged in to the HPC. This should load the MKL linear algebra libraries, which pp.x and similar programs from Quantum Espresso depend on.

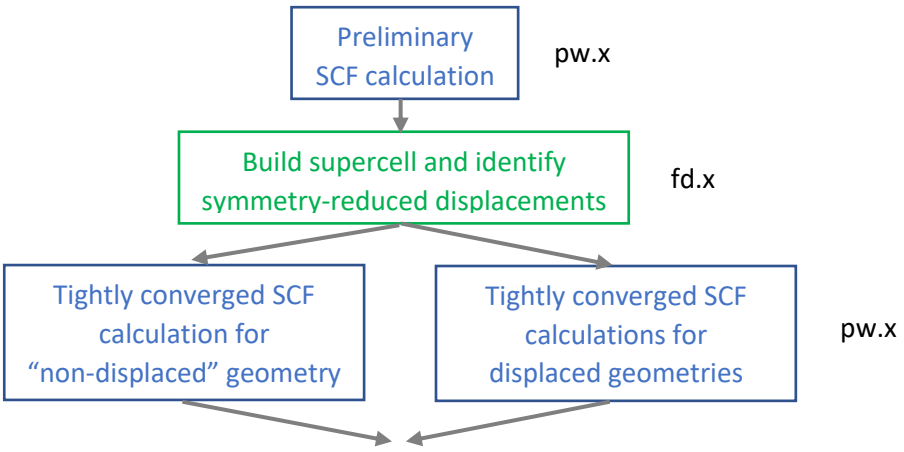
## Exercise 1: Phonon spectrum of silicon

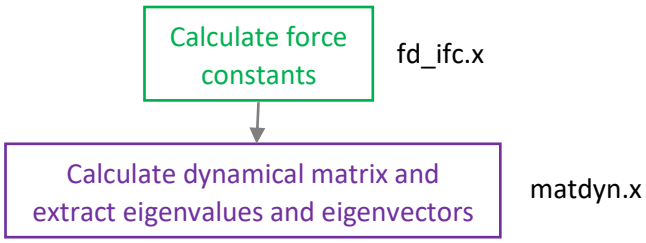
(a) Depending on the method used, the dynamical matrix is either calculated from DFT through evaluation of interatomic forces after small atomic displacements, or through evaluation of the second derivative of the total energy with atomic displacements. In both cases, the accuracy and consistency of the results will depend on the quality of the unperturbed groundstate. Hence, we first need to prepare an atomic geometry that is as tightly converged as possible (and convenient) to ensure a good prediction of phonon frequencies. Silicon crystalizes in the diamond structure (i.e. an FCC lattice with two atoms per primitive unit cell) with a cubic lattice constant of 5.4 Å.

Use this information to converge atomic positions and lattice vectors such that the interatomic forces and the cell pressure are smaller than 0.0002 Ry/a0 and 0.01 kbar, respectively.

Use the provided normconserving pseudopotential with a cutoff energy of ecutrho=70 Ry and a shufted 6x6x6 k-point grid. If you feel experimental, you can also download a pseudo-potential from the Quantum Espresso website (https://www.quantum-espresso.org/pseudopotentials) and converge the cutoff energy (or energies) yourself.

(b) Now, we will calculate the Brillouin zone center phonons ("Γ-point" phonons) using the finite displacement method. In principle, this could be done relatively straightforwardly by oneself and some basic programming skills, but we will use the already implemented FD part of Quantum Espresso for the calculations. As laid out in lecture 11, the calculations follow the following scheme






The calculations of phonon modes from FD is a bit complex, as it requires several calculations with different codes. The corresponding input files are already prepared in the folder Exercise1/b in the .zip file. The input file Si.scf.in corresponds to the first step and is used to perform a quick SCF calculation to obtain atomic geometry and symmetry operations of the crystal in a form that can be read by FD. fd.x is used for the second step. First have a look at the input file. By virtue of the prepared comments, it is somewhat self-explanatory: The block `&inputfd` contains the keywords prefix and outdir, which have to be equal to the values in the Si.scf.in file. These are used by fd.x to read the geometry and symmetry operations of silicon. The additional keywords control the parameters for the finite displacement calculation. The supercell used for the phonon calculations consist of `nrx1` x `nrx2` x `nrx3` primitive cells (or rather: the unit cells defined in Si.scf.in). As we are only interested in the Γ -point phonons here, `nrx1`, `nrx2`, and `nrx3` are set to 1. The atoms in the unit cell will be displaced by `de=0.01` Å in each of the *lattice vectors*. This value has to be chosen small enough to make sure that the harmonic approximation is valid, but large enough that the resulting interatomic forces are large enough to not be affected significantly by numerical noise. 0.01 Å usually is a good value, if one makes sure that the numerical noise is small. The displacements will be done both in negative and positive direction (due to `innx=2`), to use the more accurate central differences stencil for numerical derivatives.

The `&verbatim` block defines a template for input files for SCF calculations of supercells with displaced atomic positions. Most of this should not be changed, but important control parameters are the K_POINT sampling, which depends on the supercell size (the larger the supercell, the less k-points do we need) and the `conv_thr` and `mixing_beta`. `conv_thr` should be set to a value that is as small as possible. If the value is too small, especially for defective systems or systems with insufficiently optimized atomic positions, the SCF procedure might not be able converge below that value. `conv_thr` between $10^{-14}$ and $10^{-15}$ works in most cases, but again: the smaller, the more trustworthy the results are. `mixing_beta` can be used to help convergence. You can also adjust `prefix`, `outdir` and `pseudo_dir` to your liking. The systems2 block and the information about atomic species will be automatically filled with the values from the Si.scf.in input file.

Running the command $PATH_TO_FD.X/`fd.x <fd.in >fd.out`

(as before, replace $PATH_TO_FD.X with the folder, where fd.x is located in your case) will cause fd.x to read atomic positions and symmetry operations and use the information to identify, which atomic displacements are equivalent by symmetry. It will then use the template to generate and write a number of input files in the folder `fd_files` (you can change the folder with the keyword `fd_outfile_dir`), which each contain the supercell with one slightly displaced atom. The input files names have the format 'displaced.I.M.N.in', where N and M are the atom and the lattice vector, along which the atom is displaced, respectively,

and I indicates whether the atom is displaced in positive or negative direction. Silicon is a cubic crystal and the two atoms, as well as the three lattice vectors, are equivalent. Hence, there are only three SCF calculations to make: the two atomic displacements and a calculation for the completely unperturbed geometry.

This is, of course, done by running the command `pw.x <displaced.I.M.N.in > displaced.I.M.N.out`

for each of the input files.

We now need to extract the forces for each of the calculations and store them in a way that FD can work with them. Open each output file and copy the forces into new files forces.I.M.N. Each of these files should then contain two lines of three numbers.

Using these files, we can now calculate the force constants with the program fd_ifc.x. the input file, fd_ifc.in, should be self-explanatory at this point.

Running `fd_ifc.x <fd_ifc.in >fd_ifc.out`

will read these files and write the force constants into a file Si.fc. This alone is not sufficient to be used further with Quantum Espresso, it still lacks the header, which is found in fd_files/header.txt .

Write a new file Si_ifc.FC, which contains first the header from fd_files/header.txt and the content of Si.fc.

Now will have noticed by now, that working with FD requires either a variety of manual manipulation of files and running of programs, or writing a script that does all of these steps one after the other. I prepared such a script for use on the HPC (j.submit.sh_fd), but one should it on any linux.

We now have the force constant matrix, but still not the phonon frequencies that we are actually interested in. For this, we need to take the force constant matrix and Fourier transform them to obtain the dynamical matrix at q=0 (i.e the Γ point). This is done with the program matdyn.x. have a look at the input file matdyn.in.

Again, the content is rather simple: `flfrc` defines the file containing the force constant matrix, in our case Si_ifc.FC. `asr` defines the method that should be used (if any) to enforce the acoustic sum rule. For 3D crystals, we have two possibilities: ‘simple’ and ‘crystal’. Both enforce the acoustic sum rule for the three translational degrees of freedom, in the first case by correcting the diagonal of the forces constants matrix, in the second case by optimized correction of all force constant through a projection method. Both work fine, even though ‘crystal’ often gives slightly better results (as it should). Additional options are ‘one-dim’ and ‘zero-dim’, which also impose the asr for existing rotational degrees of freedom. `asr='no'` deactivates enforcement of the asr. `flfrq` and `flvec` define the file names for the output of the calculated phonon frequencies and normalized phonon displacements, which we are interested in. The other keywords are crucial for the functionality of matdyn: they define what q-point(s) we are interested in and in what coordinates system we give them to matdyn.x. We will see other definitions later in the exercise, but here, we are only interested in one point,

the Brillouin zone center. After the &input block, we thus state that we only want q-point and in the next line define the coordinates of this q-point. `q_in_cryst_coord=.true.` tells matdyn.x that the q-point(s) given below is in crystal coordinates, i.e. in units of the reciprocal lattice vectors, otherwise they are given in carthesian coordinates in units of 2π/alat [1].

A final note: it is very important that you use the correct atomic weights for the elements in the system. Most conveniently, you already define the correct atomic weights in the Si.scf.in file, which will then carry over to all subsequent calculations. Alternatively, you can also define the correct atomic weights here, using the keyword `amass(1)=28.086`. A second species in the systems would then be defined with `amass(2)` etc. Using wrong atomic masses (for example using a n atomic weight of 14 for silicon) can make the predicted phonon frequencies nonsensical.

Run `matdyn.x <matdyn.in >matdyn.out`

You should now have obtained the zone center phonon frequencies and the phonon eigendisplacements. Note that matdyn uses a Fourier transformation, i.e. we could obtain frequencies at arbitrary q-points through Fourier interpolation. However, these frequencies will probably be bad, as the force constants we obtained are only exact for $\vec{q}$=0, due to the use of the primitive cell.

Probably the easiest way you can plot the eigendisplacements is by using the phonon website (http://henriquemiranda.github.io/phononwebsite/phonon.html).

For this, we have to convert the .modes file to something the website can read. Part of the phonon website is a python module, which can do this, but it unfortunately has to be installed locally. Download the phononwebsite.zip file from StudOn and copy the contents to a convenient place or onto the HPC. On your home computer, you will need to have python installed (on Linux it is installed by default, on Windows you have to install it). On the HPC, you can load a python module by typing

module load python/2.7-anaconda

Then enter the folder phononwebsite-gh-pages that you decompressed from the zip file. Type (on Linux, on Windows there is probably a similar command)

python setup.py install –user

This should install the phonon website module.

After this is done, go back to the folder where you did your calculations. Now type

python read_qe_phonon.py Si silicon -s Si.scf.in -w

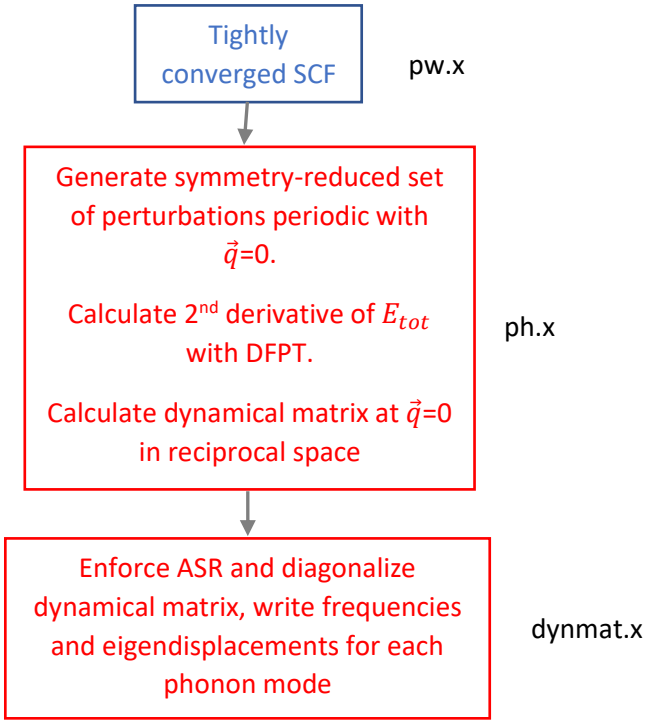
This will read the contents of Si.scf.in and Si.modes and write and output file silicon.json. To make this work, you might need to replace “(crystal)” in Si.scf.in by “{crystal}”. You can visualize the vibrations by going to the phonon website and loading your new silicon.json (if

[1] Obviously, the unit does not make a difference here, but will do so later.

on the HPC, you obviously have to first download it to your own computer) using the "Browse" button. You can now change the number of unit cells displayed, the speed of animation etc. You can change the visualized mode by clicking onto a phonon branch in the "Phonon dispersion" window on the right side. Unfortunately, the phonon website is meant to the display of phonon dispersions, which we do not have here. To change the mode, you will thus have to click at the right spot on the line, you should have two clickable points at 0 $cm^{-1}$ and at ~510 $cm^{-1}$. Another problem is that we can only display one of the degenerate modes. Still, the phonon website is a useful tool for the visualization of atomic vibrations, especially for q=/=0 phonons.

(c) Now, we will recompute the zone-center phonon modes of Si using the DFPT method. This method is significantly simpler to use (at least manually) than the FD method before, which is, however, mainly because of the way both methods were implemented in Quantum Espresso.

The calculations will be performed in three steps, outlined in the following scheme:




Use the input file Si.scf.in in Exercise1/c (or write your own) and make sure you use the geometry obtained from (a). Do a SCF calculation with very tight convergence (like in step (b) ) parameters.

Before performing the ph.x run to calculate the phonons, have a look at the provided input file ph.in. Here again, we have prefix and outdir, which define where ph.x can find the data, especially the electron density and the KS potential, from the preceding SCF run. As in (b), amass defines the atomic weights of the species in the system. This keyword is optional, if the correct atomic mass was already defined in the SCF input file. It can also be defined in dynmat.x, but as you will see, ph.x itself will already report the phonon frequencies (even though without

ASR), and wrong atomic weights here will make it difficult to assess the quality of the calculations without postprocessing the results.

`tr2_ph` defines the convergence criterion of the selfconsistent calculation of the variation of the electron density needed to calculate the energy derivatives. As for the convergence of the total energy, this value should be chosen to be as small as possible; values of $10^{-14}$ or $10^{-15}$ seem to work well in many cases, but the lower the better.

In contrast to FD, ph.x does not calculate the force constant matrix, but directly the dynamical matrix in reciprocal space for a specific q-point [2]. One way to define these points is giving their explicit coordinates after the &inputph block, in a similar to matdyn.in in (b). Here, however, the coordinates are always in units of 2π/alat, which can be a bit inconvenient. After the calculation is finished, the dynamical matrix is written to a file Si.dyn defined by the keyword `fildyn`.

Run the calculation with the command `$PATH_TO_PH.X/ph.x <ph.in >ph.out`

and examine the output file.

(On the HPC, run the calculations in parallel using an appropriate job submission script. As always, a suitable script is provided, which will run the pw.x and ph.x calculations one after another.)

ph.x takes the electron density and the potential from the preceding SCF calculation and starts the phonon calculation. note that it analyses the symmetry operations of the system and assigns a point group ($O_h$) and uses group theory to do a factor group analysis and derive the irreducible symmetry representations that the phonon modes will have. For each of these representations (i.e. 2), it then does a selfconsistent calculation. after this is finished, it diagonalizes the obtained dynamical matrix and reports the phonon frequencies. Notice how the three acoustic modes indeed do not have zero frequency, but a negative frequency. This effect comes from numerical noise due to the fact that we only know the electron density and the wavefunctions at a set of discrete points, and thus slightly violating the translational invariance of space.

It also assigns the previously derived symmetry representations to each of the frequencies and states whether the modes are infrared (I) or Raman active (R) by symmetry. It does, however, not make a statement about the *intensities* that each of these modes would contribute to Raman or infrared spectra, which we have to calculate explicitly.

The symmetry analysis of ph.x is quite useful for the assignment of calculated phonon modes to experimental data and is more tedious to achieve within a finite displacement scheme.

We will now impose the acoustic sum rule and extract the phonon eigenvectors. This is done with the tool dynmat.x. The input file, dynmat.in, is simple: all we have to do here is tell dynmat.x where it can find the dynamical matrix and what scheme for ASR correction we want

[2] To be precise, ph.x can calculate the dynamical matrixes at several q-points in one run. This requires setting the keyword `qplot=.true.` in the `&inputph` block, otherwise ph.x will only try to read the coordinates of one q-point from the first line after the "/". If `qplot=.true.`, one has to define a block (in QE language it is actually a "card") `qPointsSpecs` after the &inputph block. This block then defines the q-points to be calculated. Details about the definition can be found at https://www.quantum-espresso.org/Doc/INPUT_PH.html#idm276.

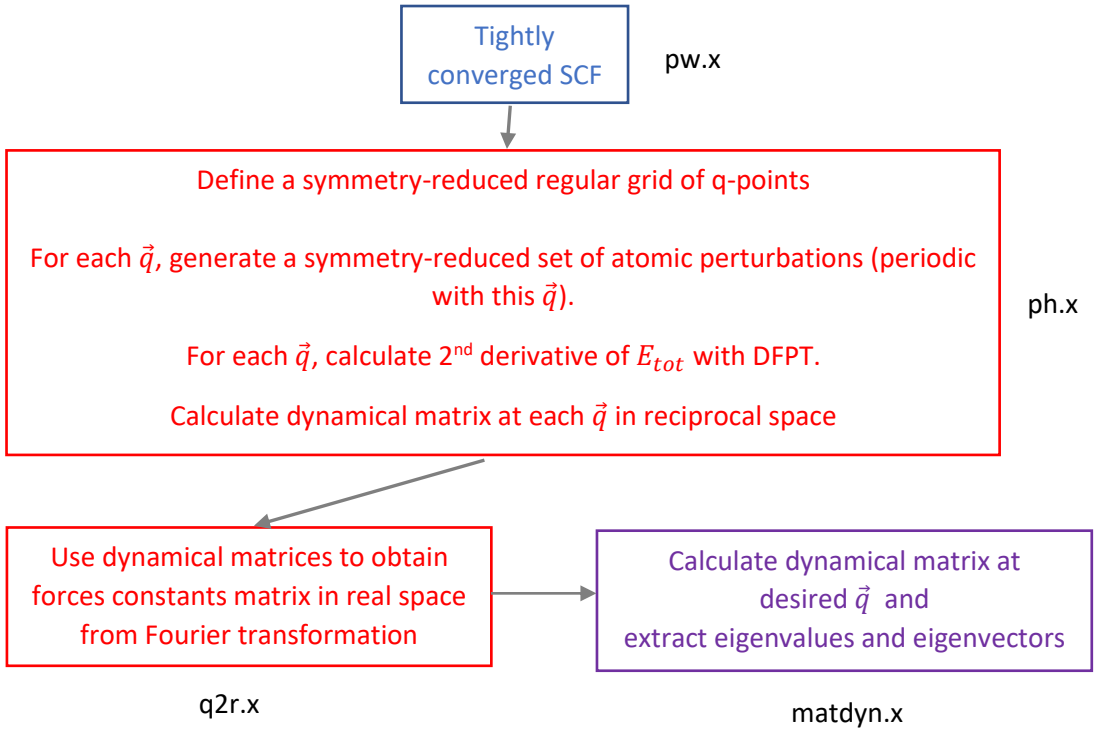
to use. The remaining three keywords define three files into which the frequencies and normalized phonon displacements will be written in three different formats.

The .axsf format can be read by XCrysDen (i.e. under Linux) and shows the displacements in the form of arrows ("forces" within XCrysDen). The .molden format can be read with a number of programs, several are available under Windows, for instance Chemcraft (a trial licence should be valid for 180 days). Finally, the contents of the .modes format can be visualized using the method outlined in (b).

Run dynmat.x with the provided input file. Note that the ASR correction is successful and the three acoustic modes now have zero frequency.

(d) We will now go one step further and calculate the phonon dispersion of silicon. This can in principle be done withing the FD approach, but is much more conveniently done in DFPT.

The procedure is as follows:




Doing this only requires a few minor changes in ph.in: `ldisp=.true.` tells Quantum Espresso to do a calculation that can be used to derive the phonon dispersion. More specifically, it tells Quantum Espresso to calculate the phonons on a set of q-points on the regular Monkhorst-Pack grid defined by the three keywords `nq1`, `nq2` and `nq3`. Here, we will use a 5x5x5 sampling [3]. Consequently, we now do not have to explicitly define the q-points after the `&inputph` block anymore (or rather: everything given there will be ignored).

[3] I should note that the phonon dispersion is only fully converged for a 6x6x6 sampling or larger, which is also a reason for the deviations of the phonon dispersion shown in lecture 11 compared to experiment. These samplings take even more time to calculate.

Run the preceding SCF calculation and ph.x. This calculation will take a while, much longer than the calculations in (b) and (c). Make sure to run the calculations *in parallel* on the HPC or to schedule a few hours on a home computer. You can speed up the calculation by using a 4x4x4 q-point grid only and use less converged cutoff energies and k-point samplings in the SCF step.

ph.x will first generate a symmetry-reduced set of q-vectors for which the phonons will be calculated (in this case 10), which are reported at the beginning of the output file and in the file Si.dyn0.

After this definition, ph.x uses DFPT to calculate the dielectric permitivity tensor and the Born effective charges of the system, which is used later to include LO-TO splitting into the dispersion [4]. ph.x always does this for calculations of the dispersion, but it is not activated by default for DFPT calculations for single q-vectors.

For each of the symmetry-reduced q-points, ph.x will then run a phonon calculation and write the dynamical matrix into the file Si.dynX, where X is the index of the q-point. Notice that for (almost) each q-point, ph.x runs a non-selfconsistent calculation on the k-point grid used in the SCF calculation (this k-point grid can be changed, for instance to a denser one, see Exercise 2) before working on the phonons. This is done because a perturbation periodic with a certain q-vector changes the symmetry of the system and ph.x needs the band energies on an irreducible set of k-points that is consistent with this reduced symmetry. However, in case of the Γ point, no symmetry reduction occurs.

As ph.x heavily exploits crystal symmetry, you will notice that the calculations for the q-vectors that strongly reduce the crystal symmetry take significantly longer than calculations for the Γ point or the first q-points in the list, which usually lie on some symmetry line. This exploitation of crystal symmetry cannot be done to the same extent in the FD method.

After the calculation finished, you should now have 11 .dyn files. In order to obtain the actual phonon dispersion, we now need to transform the calculated dynamical matrices to obtain the force constants matrix in real space. This is useful, because these force constants decay quickly with the interatomic distance, so that we can use Fourier interpolation.

Running `$PATH_TO_Q2R.X/q2r.x <q2r.in` will write the force constants into the file Si.FC.

We will now use matdyn.x again. Now, however, we are not interested in a single q-point, but in the phonon dispersion along a path in reciprocal space. A standard path for FCC lattices is the path Γ*->X*-> Γ->L. Here, the star indicates that Γ and X points lie in a different unit cell, the advantage of this is that the X*-> Γ path also include the K point. The coordinates of the high symmetry points in the Brillouin zone of the FCC lattice can be found easily on the internet.

We could define the q-points on the q-point path explicitly using the method in (b). However, matdyn.x also offers the possibility of only defining the start and end points of the path segments and automatically filling the segments with a specified number of points. This can be invoked by setting `q_in_band_form = .true.` . Similar to calculations of the bandstrucutre in pw.x, we now give the number of 'special' points between the segments (4) and then the coordinates of these q-points and the number of q-points we want to lie on the following path

[4] Silicon is non-polar, so these calculations do not make much sense in this case, but there is no way (apart from hacking the code) to tell ph.x to not calculate the dielectric tensor.

segment. The coordinates in the input file matdyn.in corresponds to the crystal coordinates of Γ*,X*,Γ,L.

Run matdyn.x . This should write a file Si.freq and a file Si.freq.gp. the latter file contains the phonon dispersion along the specified path that can be plotted with gnuplot or similar programs. Plot the phonon dispersion and compare it with the experimental data shown in lecture 11. Note that the Si calculations shown in lecture 11 used a different pseudopotential than the calculations here, which also changed the predicted phonon frequencies.

Using the Si.modes file, you can visualize the phonon modes on the phonon website. Do this and have a look at the atomic vibrations for different q-vectors.

(e) We will now calculate the phonon or vibrational density-of-states (VDOS). We do not have to do much here, but can resort to simply obtaining the phonon frequencies on a dense regular grid from Fourier interpolation of the force constant matrix obtained in (d).

All we have to do is modify matdyn.in accordingly. Have a look at the provided matdyn.in. `flfrc` is changed to point to the Si.FC in the folder of the calculation of part (d). We can trigger a VDOS calculation by setting `dos=.true.` in the `&input` block. The dense regular q-point grid to interpolate phonon frequencies on is defined using the (not aptly named) keywords nk1, nk2, nk3. No additional definition of q-points is necessary. Fldos and `deltaE` define the file where matdyn will write the calculated VDOS to and the energy resolution (in $cm^{-1}$), respectively. The VDOS will be calculated efficiently using the tetrahedron method.

Run matdyn. The output file Si.dos does not only contain the VDOS, but also projected VDOS' that give the contribution of each atom to the total VDOS. Plot the results and compare the peak position with the phonon dispersion from (d). You should notice that regions of flat dispersion give rise to prominent peaks in the VDOS.

(f) Calculated phonon spectra can be used to derive thermodynamic quantities of silicon. We will do this here using the example of the temperature-dependent thermal expansion of silicon. For this, we will use the quasi-harmonic approximation, which allows the inclusion of volume-dependent anharmonic effects under the assumption that at each specific volume, the harmonic approximation is valid. A nice summary of thermodynamic quantities derivable from harmonic lattice dynamics can be found at http://tutorials.crystalsolutions.eu/tutorial.html?td=Tutorial_QHA&tf=QHA#sampling .

A central quantity in the quasi-harmonic approximation is the QHA version of the Helmholtz free energy,

$$F^{QHA}(T,V) = U^0(V) + F_{vib}^{QHA}(T,V)$$

where $U^0$ is the internal energy of the system at T=0 K and $F_{vib}^{QHA}(T,V)$ contains the vibrational contributions to the free energy. $U^0$ can be replaced with the total energy from DFT, $E_{tot}^{DFT}$.

The vibrational part

$$F_{vib}^{QHA}(T,V) = E_0^{ZP}(V) + k_B T \sum_{q,j} \left[ ln\left(1 - e^{-\frac{\hbar\omega_{qj}}{k_B T}}\right)\right]$$

is determined through the phonon spectrum of the system is thus quite straight-forward to access for us. $E_0^{ZP}(V)$ is the zero-point energy at a given volume $V$ and $\omega_{qj}$ is the phonon frequency of branch $j$ at q-point $q$.

We can do such a calculation of the free energy directly from what the results from (d) and (e). Quantum Espresso already offers a tool, QHA, that uses the tetrahedron method to derive the vibrational part of the free energy from the phonon spectrum [5].

Unfortunately, QHA is not very user friendly and requires many small steps. Unfortunately, it also does not seem to be available in the precompiled QE packages for Windows or in Quantum Mobile. This part can thus only be done on the HPC. To make life easier, a script QHA.sh is provided that runs the calculation steps on after another, without need for manual running.

At the beginning of QHA.sh , the species and their atomic masses in the system, are defined, together with the energy resolution (in $cm^{-1}$) of the density-of-states that is calculated during the QHA run. The only part of QHA.sh that maybe needs to be edited, is the definition of `QE_DIR`. Change it to point to the folder where all your quantum espresso programs are. The QHA run needs a number of files that are provided in the .zip file, too. The file ttrinp defines a set of tetrahedra vertices specifically for a fcc lattice that are used to obtain the VDOS. matdyn.init is a template file that will be sued to interpolate a dense regular grid. These files do not have to be edited. The file Temperature defines a range of temperatures, for which the vibrational part of the free energy will be calculated. This is set to do the calculations in steps of 10 K between 100 K and 400 K. You can change make the sampling denser, which will generate more data, but also give more information. You also need to have a file with the force constants, Si.FC, in the folder where you run QHA.sh.

Now run QHA.sh for the ground state geometry obtained in (a) by submitting the job script j.submit.sh_QHA (the calculations will run for a few minutes). This produces a number of files. We are aiming at the file Si.QHA.out. As you see in the legend, we are interested in the values in the third column. Adding these values to the DFT total energy will give us the temperature-dependent free energy $F^{QHA}(T,V)$ for this specific volume.

Minimization of $F^{QHA}(T,V)$ for a given temperature will give us the equilibrium volume $V^{eq}$ (and thus the lattice constants) at this temperature. This can be achieved by manually changing the volume of the crystal within a certain range, calculating the total energy and the phonon spectrum of the crystal, and thus $F^{QHA}$, at each volume, and fitting the results to find the minimum.

For this purpose, it is useful to define the geometry of the crystal by its Bravais lattice (ibrav=2) and the lattice constants (A, in Angstrom). Such an input file is found in Exercise1/f.

Use the method from (d) to calculate the force constant matrices silicon with lattice constants 5.2, 5.3, 5.5 and 5.6 Angstrom and then use QHA to derive the vibrational part of the free energy. Ideally, you create a new folder for each of these lattice constants and run the calculations in parallel, to save time.

[5] A version for use on the HPC can be found on StudOn. Put it on the HPC and uncompress with tar -xvzf QHA.tar.gz

In order to extract the equilibrium volume for each temperature, we need to perform a fit of the Birch-Murnaghan equation (refer to lecture about geometry optimization) to the calculated free energies in dependence on the volume. This is quite tedious to do by hand. Therefore, a simple python script is provided that calculates the free energy for all volumes and temperatures and does the fitting. For each temperature, it writes a file Ffree_XXX and a file fit_XXX, which contain the calculated free energy and the corresponding Birch-Murnaghan fit, respectively.

You can run the script on the HPC by the commands

module load anaconda/2.7-python

python fit.py 5.2/Si.QHA.out 5.3/Si.QHA.out 5.4/Si.QHA.out 5.5/Si.QHA.out 5.6/Si.QHA.out

This assumes that you performed the calculations for the different volumes in separate folders. You also need a file DFT_energies that contains a column with all lattice constants and a second column with the DFT total energy for each volume.

The script also calculates the thermal expansion coefficient

$$\alpha_V = \frac{1}{V(T)}\left(\frac{dV(T)}{dT}\right)_{P=0}$$

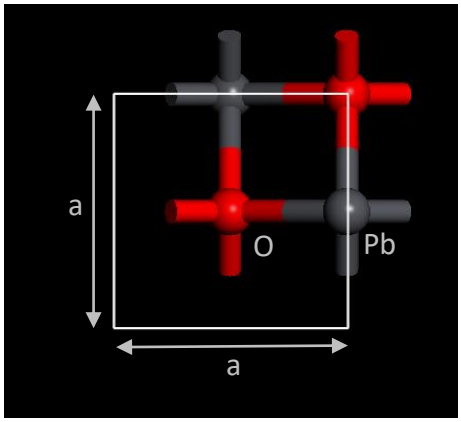
using a central difference formula from the free energy data.

Plot and inspect the results. Interestingly, silicon is one of the few materials that feature a negative thermal expansion coefficient, in this case at low temperatures up to about 150 K (see also Solid State Communications 10, 159 (1972)).

(g) Finally, calculate the bandstructure of silicon for the equilibrium volume at 300 K, as obtained from (f) and compare the electronic band gap with the band gap you obtain for the lattice constants from (a).

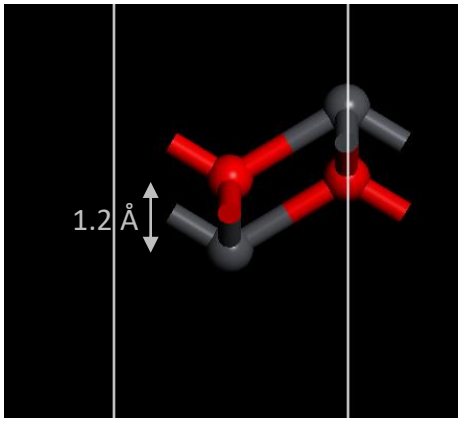
## Exercise 2: Phonon dispersion of monolayer PbO

In this exercise, we will have a look at the Γ point phonons of one-dimensional lead monooxide (PbO). It has a quadratic unit cell with lattice constant a=3.9 Ang and assume a buckled structure with lead atoms pointing out of a layer of oxygen atoms. The structure is depicted in the figure.




(a) First, use the pictures to derive an initial guess for the atomic positions of PbO. Use lattice constant of 20 angstrom in z direction (a reference file is provided for comparison).

Then use your initial guess to perform a geometry relaxation that yields atomic positions and lattice vectors that are suitable for phonon calculations (i.e. forces below 0.002 Ry/a0 and pressure below 0.01 kbar). Use a 8x8x1 k-point sampling, the normconserving pseudopotentials provided in the folder Exercise2 and a cutoff energy of 90 Ry (or do your own convergence tests).




(b) The phonon frequencies are somewhat sensitive to the energy cutoff used, and one usually needs to use quite large cutoff energies in order to converge all phonon frequencies to less than 0.1 $cm^{-1}$. One always has to keep this fact in mind, as the cutoffs for a good geometry relaxation are usually lower than those for a very converged phonon calculation.

To see this dependence, use the provided input file (ph.in) to calculate the Γ point phonons for cutoff energies of 90, 100, 110 and 120 Ry. In each case, run a SCF calculation with a certain cutoff energy and then a phonon calculation on top. Again, you can run these calculations simultaneously on the HPC, if you create a separate folder for each cutoff energy, copy the input files and the job submissions script there (again: make sure that the pseudo_dir points to the correct folder) and start the calculations.

Examine how the two highest frequencies change with the cutoff energy. It is enough to use the values given in the output file of the ph.x run instead of running dynmat.x for each calculation.

In a similar way, the k-point grid for the electronic states used in the DFPT calculations can have some effect as well. As mentioned before, the k-point grid for the DFPT run can be chosen differently than that used in the SCF calculation. This is done by adding the keywords `nk1`, `nk2` and `nk3` to the `&inputph` block in ph.in [6]. Do another calculation with a cutoff of 90 Ry, but use a 10x10x1 k-point grid for the DFPT calculations.

(c) Let's now use quantum Espresso to derive the Raman and infrared activities together with the phonon frequencies. This is actually very simple: the infrared activities are calculated automatically if the dielectric tensor is calculated, for the calculation of the Raman tensor we only need to set the keyword lraman=.true. in the `&inputph` block. Start the calculation. This will take a while, so you should run it on the HPC. Examine the output file. After calculating the Born charges and the dielectric tensor, ph.x will calculate the necessary quantities for the Raman tensor, before continuing with the dynamical matrix.

After the calculation is finished, use the input file dynmat.in to postprocess the calculation. Notice that in the table listing the frequencies, dynmat.x also reports the infrared and Raman activities for each phonon mode. As mentioned in the lecture, the reported Raman activity is

[6] It can be advisable to choose the k-point grid sampling to be an integer multiple of the q-point grid used for the calculation of the phonon dispersion, because the code internally calculates k+q vectors representing electrons that are scattered by a phonon. These k+q vectors have to be included in the k-point grid, which is the case if k- and q-point grids are commensurate, otherwise quantum espresso fills up the k-grid with the missing k+q points, which can cost computational time.

an average over all direction of incoming and scattered light, thus it should not be compared 1:1 with spectra from Raman experiments on solid crystalline materials. Notice that the symmetry analysis of ph.x predicts the degenerate low-frequency mode at ~80 $cm^{-1}$ to be Raman active, while the calculated Raman activities of these modes turn out to be very low. The symmetry analysis thus gives hints at the Raman activity of phonon modes but only shows one part of the picture.

You could now create a theoretical estimate at the Raman and infrared spectra of PbO by using a superposition of gaussian or Lorentzian peaks centered at the calculated phonon frequencies, with peak heights determined by the Raman or infrared activities.

Another property is missing, however: PbO is a polar material and should exhibit LO-TO splitting at the Γ point. This is effect is not included by dynmat.x by default (for phonon dispersions it is), because the size of the LO-TO splitting depends on the direction from which one approaches q=0. This direction can be specified by the additional keywords q(1), q(2), q(3), which are given in carthesian coordinates, in units of 2π/alat. The input file dynmat.in_loto tells dynamt.x to calculate the non-analytical correction for q->0 in x-direction. Notice how this causes the double degenerate $E_u$ modes at ~280 $cm^{-1}$ to split, by pushing the LO mode to a frequency of ~400 $cm^{-1}$. Obviously, this has a significant effect on the infrared spectrum, with another peak appearing.

## Extra-Exercise: Aluminum-doped $SiO_2$

Here, we continue the extra exercise from exercise sheet 4.

Remember to run the calculations in parallel on the HPC.

(d) In (c), we were concerned with the total energy of a defect without total charge. This corresponds to a defect charge state of 0. In reality, however, defects or impurities will act as charge traps, and release or bind electrons. This is, in the end, the idea behind employing donor or acceptor atoms to enhance the electrical conductivity of semiconductor materials. The formation energy for a defect with charge q can be quite non-trivial to calculate, for a variety of reasons. A detailed discussion can be found in Phys. Rev. B 78, 235104 (2018). Many practical problems can be traced back to the periodic boundary conditions (PBC). Due to PBC, unit cells that are not charge neutral lead to an infinitely large total energy. For this reason, all codes using PBCs add a homogeneous background charge [7] to compensate for the total charge in the unit cell. This leads to a small residual interaction energy between the electrons in the unit cell and the compensating background charge. Also, charges localized on defects will act, due to the PBC, like an interacting lattice of charges in a compensating background charge, which adds another residual interaction energy. Residual, because the defect or impurity concentrations that can be studied with DFT are several orders of magnitudes larger than what can be achieved in real materials.

[7] This is usually not done by explicitly adding a background charge but implicitly by zeroing the G=0 terms in the pseudopotential and the Hartree and exchange-correlation potentials.

We will thus in the following neglect these terms and use a simplified equation for the formation energy of a system with a charged defect (with charge *q*) or impurity:

$$\Delta E = E_{tot}^{SiO2:Al,q} - E_{tot}^{SiO2} + \mu_{Si} - \mu_{Al} + q(\Delta E_F + E_{VBM})$$

We here again have the problem of comparability of the total energies between two different systems: For this reason, the valence band maximum (VBM) of pristine $SiO_2$ in the dilute limit, $E_{VBM}$, will be taken as the reference energy. The picture here is rather simple: removing an electron from an infinitely large $SiO_2$ crystal and keeping wavefunctions and density distribution the same will shift the total energy of the system by $E_{VBM}$.

In principle, we can derive $E_{VBM}$ from shifting the vacuum level to 0 eV again, but this is a bit troublesome, because it requires build a $SiO_2$ slab and additional geometry optimization.

A faster way is calculating the total energy of a unit cell of $SiO_2$ with a very small charge of q=0.001 and obtaining $E_{VBM}$ by

$$E_{VBM} = \frac{E_{tot}^{SiO2,q=0} - E_{tot}^{SiO2,q=0.001}}{0.001}$$

Do this using the input file for the unit cell from (b) and adding `tot_charge=0.001` to the `&system` block (Si.scf_VBM.in is a reference input file for this task). This should give you a value of 0.1334 Ry.

What is now missing is the total energy $E_{tot}^{SiO2:Al,q}$ of different charge states. In addition to the case q=0 from before, we will look at two different charge states, q=+1, and q=-1 (all in units of the elementary charge).

Create two folders charge_1 and charge_-1. Copy the input file and the job submission script from the total energy calculation of the charge-neutral case from (c) into each of these folders.

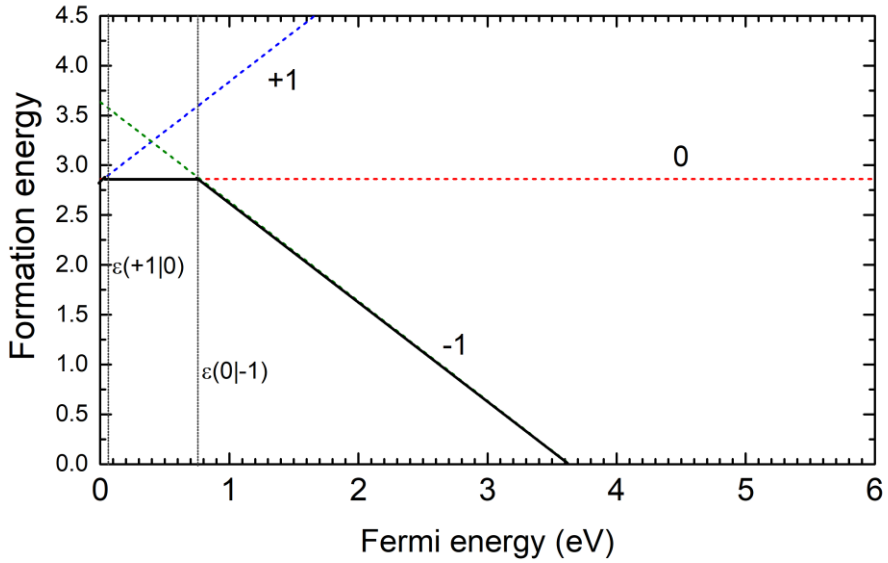
Now modify each of the input files by adding the keyword tot with an appropriate value to the &system block. Also, make sure that pseudo_dir points to the folder with you Al, Si and O pseudopotentials. The additional charge probably changes the atomic positions, too. For this reason, use the geometry from (c) as a pre-relaxed starting point and perform a geometry optimization of only the atomic positions, but not the lattice constants (takes too long). Reference files for this task are SiO2.scf.in_charge1 and SiO2.scf.in_charge-1 in Exercise3/d.

Submit the calculations to the HPC and extract the final total energies.

(e) The expression for the formation energy given earlier also depends on the Fermi energy (relative to the valence band maximum). This is not surprising, because the Fermi energy corresponds to the number of electrons in the system and it should become less likely to have a positively charged defect. Depending on the Fermi energy of the system, one of the charge states of the impurity will be more stable than the others, meaning that the purity is prone to accept or donate electrons by trapping negative or positive charges.

Plot the formation energy in dependence of the Fermi energy ($E_F$=0 is the valence band maximum, the band gap of $SiO_2$ from PBEsol is 6 eV) for each of the charge states.

This should give you a plot that looks like the following:




This suggests that the aluminum impurity (or rather: the impurity and the adjacent atoms) prefer trapping an electron for most values of the Fermi energy. Note that a Fermi energy around the middle of the band gap suggests that the system overall is charge neutral, meaning that for the trapped electron near the aluminum impurity, there is a free hole somewhere else in the material. Only for stronger doping (Fermi energies below 1 eV above the VBM), the neutral charge state becomes more stable, because there are less electrons available for trapping by an aluminum center. For very low Fermi energies, there is another "transition state" and the positive charge states becomes slightly more stable.

Similar investigations can be done for essentially any defect or impurity and are a strong tool for understanding from a theoretical point of view, what defects are likely to be present in a real system and what elements work well as donor or acceptor species (and why).

To obtain a complete picture, we would need to include more positive or negative charge states to see whether there are additional transition states present in the band gap. However, higher-order charge states require additional considerations, because the larger charge introduces additional residual interaction effects between defect-localized charges in different unit cells, which can have a magnitude of several eV. As this is a somewhat tedious work, we won't do it here, but a description and assessment can be found in Phys. Rev. B 86, 045112 (2012).

Another error comes from intrinsic shortcomings of local XC functionals, often the differences in formation energies between different charge states are somewhat underestimated by these functionals. For this reason, hybrid functionals are nowadays seen as the state-of-the-art method.