Phonons with Density-Functional Perturbation Theory

These examples illustrate the applications of Density-Functional Perturbation Theory (DFPT) to the the calculation of phonon in simple semiconductors (Silicon and AlAs).

Getting ready

1. The following command must have been executed:

   ```
   module load qe/5.1
   ```

2. You may download the entire exercise file examples_phon.tgz in a directory of your choice. Uncompress and unpack the file and enter in the resulting directory:

   ```
   tar -zxvf examples_phon.tgz
   cd examples_phon
   ```

Phonons at Gamma point (q=0)

All phonon calculations start from the self-consistent charge density and Kohn-Sham orbitals, calculated for equilibrium atomic positions. For **q**=0 phonons in Si, you just need to run the phonon code afterwards.

1. Calculate the self-consistent charge density and Kohn-Sham orbitals for Si. Sample input file: si.scf.in. Edit "pseudo_dir" and "outdir" so that they point respectively to the directory containing the pseudopotential (you will need Si.pz-vbc.UPF) and to the scratch directory.

   ```
   pw.x -in  si.scf.in > si.scf.out
   ```

   You will get the usual self-consistent results for Si, and the needed data written to directory "outdir"/"prefix".save
2. Calculate phonons at wavevector **q**=(0,0,0). Sample input file: si.phG.in. The values of "outdir" and "prefix" must be the same as in the preceding scf step. The value of the mass will overwrite the value given in the scf calculation. The line after the namelist is the q-point to be computed (gamma).

   ```
   ph.x -in si.phG.in > si.phG.out
   ```

   The code should perform 3*Nat, Nat=Number of atoms, linear response calculations, one per atomic displacement. For efficiency reason, the 3*Nat atomic displacements are grouped into "irreps" (irreproducible representations), based on the symmetry of the system. At the end the dynamical matrix is computed, written to file si.dynG [i.e. the variable fildyn in "si.phG.in"], diagonalized. Note the 3 degenerate non zero frequencies and the 3 frequencies close to zero. The latter should be exactly zero due to Acoustic Sum Rule (ASR)), i.e. traslational symmetry. Notice that the degeneracy of the modes [3 + 3] corresponds to the dimensionality of the irreps: it is a property related to symmetry.

A small auxiliary program, "dynmat.x", among other things, diagonalizes the dynamical matrix writes the eigenvalues in "molden" and "XCrySDen" format. Input for dynmat.x may be very simple:

```
dynmat.x
&input fildyn='si.dynG' /
```

You may also impose the Acoustic Sum Rule, as in e.g.:

```
dynmat.x
&input fildyn='si.dynG', asr='simple' /
```

Notice the change. File "dynmat.axsf" contains normal modes in a format that can be visualized by XCrySDen: they appear as forces on atoms. Use

```
xcrysden --axsf dynmat.axsf
```

to visualize it, show forces (f) and change force options (shift-f) for a better visualization. Do they look reasonable? what did you expect?

Phonons at a generic q-point

In order to perform a phonon calculation at a generic (nonzero) **q**-point, an intermediate non-scf calculation must be run to (re)construct electronic states at **k** and **k+q**. This could be performed as a separate step, but it can more simply be done by the phonon code. In the following, the X point, **q** = (1,0,0) 2pi/a, is used. The steps are

1. Run the calculation of the self-consistent charge density and Kohn-Sham orbitals for Si, if you haven't already, or if you have overwritten or deleted the data files (recent versions of the phonon code do not overwrite any longer data files from pw.x)
2. Calculate phonons at wavevector **q**=(1,0,0) 2pi/a. Copy "si.phG.in" into "si.phX.in" and edit it: modify the title if you wish, the "fildyn" variable ("si.dynG" -> "si.dynX") and the last line, defining the q-point:

   ```
   0.0 0.0 0.0 ===> 1.0 0.0 0.0
   ```

   Sample file: si.phX.in.

   ```
   ph.x -in si.phX.in > si.phX.out
   ```

   In the first part of the output, a non-scf calculation is performed. The symmetry of the small group of **q** is assumed instead of the crystal symmetry, so the number of **k**-points increases. Note the presence of **k+q** vectors intercalated between the **k** vectors.
   The linear-response calculation is performed in the second part of the output. Notice that the number and degeneracies of irreps are different from those at **q**=0: they depend on symmetry, i.e. on the small group of **q**. The file si.dynX contains now three matrices, i.e. the three equivalent X points.

Repeat the same for L point = (0.5,0.5,0.5) 2pi/a (sample file: si.phL.in). You should find the following results, within minor numerical differences:

```
grep freq si.dyn*
```

```
si.dynG:     freq (    1) =       0.075097 [THz] =       2.504951 [cm-1]
si.dynG:     freq (    2) =       0.075097 [THz] =       2.504951 [cm-1]
si.dynG:     freq (    3) =       0.075097 [THz] =       2.504951 [cm-1]
si.dynG:     freq (    4) =      15.474522 [THz] =     516.174502 [cm-1]
si.dynG:     freq (    5) =      15.474522 [THz] =     516.174502 [cm-1]
si.dynG:     freq (    6) =      15.474522 [THz] =     516.174502 [cm-1]

si.dynL:     freq (    1) =       3.281870 [THz] =     109.471383 [cm-1]
si.dynL:     freq (    2) =       3.281870 [THz] =     109.471383 [cm-1]
si.dynL:     freq (    3) =      11.303142 [THz] =     377.032245 [cm-1]
si.dynL:     freq (    4) =      12.548126 [THz] =     418.560441 [cm-1]
si.dynL:     freq (    5) =      14.786118 [THz] =     493.211815 [cm-1]
si.dynL:     freq (    6) =      14.786118 [THz] =     493.211815 [cm-1]

si.dynX:     freq (    1) =       4.318105 [THz] =     144.036469 [cm-1]
si.dynX:     freq (    2) =       4.318105 [THz] =     144.036469 [cm-1]
si.dynX:     freq (    3) =      12.399902 [THz] =     413.616203 [cm-1]
si.dynX:     freq (    4) =      12.399902 [THz] =     413.616203 [cm-1]
si.dynX:     freq (    5) =      13.962317 [THz] =     465.732750 [cm-1]
si.dynX:     freq (    6) =      13.962317 [THz] =     465.732750 [cm-1]
```

Electric Fields and Acoustic Sum Rules

Let us consider a little bit more in detail the Gamma case. In insulating materials the system can sustain macroscopic electric fields and these can be coupled with vibrations. The physical properties that describe these properties are the dielectric tensor ("epsilon") and the effective charges ("Z*"). Note that in metals there are no macroscopic electric fields: the dielectric constant is infinite. Edit "si.phG.in" and add variable epsil=.true. to the namelist. This can be done ONLY if **q**=(0.0 0.0 0.0), otherwise the code complains. Run the phonon calculation at Gamma again:

```
ph.x -in si.phG.in > si.phG.out
```

Note that an additional linear-response calculation for electric fields in 3 independent directions is performed, dielectric constant and effective charges are calculated and stored into the "si.dynG" file. Note that effective charges are tensors: they are neither diagonal nor even symmetric in general! In high-symmetry systems like this one they reduce to a scalar, though. Due to translational symmetry, ASR requires that acoustic modes have zero frequencies AND the sum of effective charges over all atoms is zero: \sum_i Z*_i = 0. In Si, where the two atoms in the cell are equivalent, this means Z*=0. This is never exactly verified in realistic calculations because:

1. insufficiently accurate scf threshold (in pw.x and/or ph.x part). Usually pw.x takes very little time w.r.t. ph.x, so **do not** use poor (large) thresholds in pw.x in order to spare some time: you gain very little and you may get lousy results.
2. XC energy is computed on a real-space grid that assumes an explicit origin. More problematic for GGA than for LDA.
3. **k**-point sampling is not accurate enough (for Z* and epsilon): macroscopic electric fields arise from the limit **q**->0, which is known to require a dense grid of k-points.

Let us see explicitly the latter point. Change **k**-point sampling in si.scf.in with the 28 monkhorst-pack points :

```
  K_POINTS automatic
   6 6 6 1 1 1
```

Run again pw.x+ph.x, save the output for later comparison:

```
     pw.x -in si.scf.in-28k > si.scf.out-28k
     ph.x -in si.phG.in > si.phG.out-28k
```

Do the same for the 2 chadi-cohen / monkhorst-pack points :

```
  K_POINTS automatic
   2 2 2 1 1 1
```

for the 6 monkhorst-pack points :

```
  K_POINTS automatic
   3 3 3 1 1 1
```

and for the 10 chadi-cohen / monkhorst-pack points (what we have used up to now)

```
  K_POINTS automatic
   4 4 4 1 1 1
```

Notice that

1. ASR for effective charges improves by improving BZ sampling.
2. ASR for acoustic mode improves but does not go to zero => k-point sampling is not an issue for ASR for acoustic modes.
3. Optical modes are practically unaffected.

You could try to improve scf thresholds for pw.x (e.g. to tr2=1.d-10) and ph.x (e.g. to tr2_ph=1.d-16) and repeat, but it will not really get better. You can get a really good ASR by removing the XC functional, though, but this is obviously not what you want!!!

Phonons in Polar Materials

In polar materials, macroscopic electric fields are present in the long-wavelength limit and induce the so-called LO-TO splitting. This can be computed from the knowledge of epsilon, Z* and the propagation direction **q**. We consider AlAs, a polar semiconductor with the zincblende structure (i.e. like Si but one different atom per sublattices; Si is nonpolar so there is no LO-TO splitting). Create an input for scf calculation in AlAs using the following data:

```
ATOMIC_SPECIES
 Al   26.98  Al.pz-vbc.UPF
 As   74.92  As.pz-bhs.UPF
```

and alat = 10.6 (theoretical lattice parameter for LDA and those pseudopotentials). Sample file: alas.scf.in. You will need the pseudopotentials for Al, Al.pz-vbc.UPF, and for As, As.pz-bhs.UPF. Run the scf calculation as usual:

```
     pw.x -in alas.scf.in > alas.scf.out
```

Create an input for phonon at Gamma for AlAs (including electric fields: "epsil=.true.") Sample file: alas.phG.in.

```
ph.x -in alas.phG.in > alas.phG.out
```

Notice that the optical phonon frequencies are 3-fold degenerate and there is no TO-LO splitting, because the non-analytic term that produces the TO-LO splitting has to be added to the dynamical matrix. This is done by auxiliary program "dynmat.x". Create an input (alas.dynmat.in for instance) for dynmat.x, specifying the additional variables

```
&input  fildyn='alas.dynG', asr='simple'
        q(1)=1.0, q(2)=0.0, q(3)=0.0 /
```

Run the dynmat.x code:

```
dynmat.x < alas.dynmat.in
```

The code will take note of the presence of Z* and epsilon in the dynamical matrix file and will add to the dynamical matrix the nonanalytic part corresponding to the given **q**=>0. The 3-fold degenerate optical modes are now split by the TO-LO splitting. You should get two degenerate TO frequencies around 10.71 THz, one LO frequency at 11.79 THz. Note that asr='simple' (or 'three-dim', a more sophisticated version) makes the frequency of the three lowest acoustic modes at **q**=0 to vanish, but has no effect on the other modes.

In cubic system, such as AlAs, the frequencies (NOT the eigenvectors) are independent on the **q** direction. In lower symmetries both eigenvalues and eigenvectors depend on the phonon propagation direction, and modes cannot be distinguished between purely transverse (i.e. **q** perpendicular to the displacement pattern) and purely longitudinal (i.e. **q** parallel to the displacement pattern).