PHonon User's Guide (v. 6.1)

Contents

1 Introduction

This guide covers the usage of the PHonon package, a part of the QUANTUM ESPRESSO distribution. Further documentation, beyond what is provided in this guide, can be found in the directory PHonon/Doc/, containing a copy of this guide.

This guide assumes that you know the contents of the general User's Guide for QUANTUM ESPRESSO and of the User's Guide for PWscf. It also assumes that you have already installed QUANTUM ESPRESSO (PHonon is not a stand-alone package: it requires PWscf to be compiled and used). If not, please locate the general User's Guide in directory Doc/ two levels above the one containing this guide, and the User's Guide for PWscf in PW/Doc/; or consult the web site: http://www.quantum-espresso.org. It is also assumed that you know the physics behind QUANTUM ESPRESSO, the methods it implements, and in particular the physics and the methods of PHonon.

PHonon has the following directory structure, contained in a subdirectory PHonon/ of the main QUANTUM ESPRESSO tree:

Doc/	: contains the user_guide and input data description
examples/	: some running examples
PH/	: source files for phonon calculations and analysis
Gamma/	: source files for Gamma-only phonon calculation

Important Notice: since v.5.4, many modules and routines that were common to all linear-response QUANTUM ESPRESSO codes are moved into the new LR_Modules subdirectory of the main tree. Since v.6.0, the D3 code for anharmonic force constant calculations has been superseded by the D3Q coda, available on http://www.qe-forge.org/gf/project/d3q/.

The codes available in the PHonon package can perform the following types of calculations:

- phonon frequencies and eigenvectors at a generic wave vector, using Density-Functional Perturbation Theory;
- effective charges and dielectric tensors;
- electron-phonon interaction coefficients for metals;
- interatomic force constants in real space;
- Infrared and Raman (nonresonant) cross section.

Phonons can be plotted using the PlotPhon package. Calculations of the vibrational free energy in the Quasi-Harmonic approximations can be performed using the QHA package. *Note:* since v.5.4, these two packages are separately distributed and no longer bundled with PHonon. Their latest version can be found in the tarballs of PHonon v.5.3.

2 People

The PHonon package was originally developed by Stefano Baroni, Stefano de Gironcoli, Andrea Dal Corso (SISSA), Paolo Giannozzi (Univ. Udine), and many others. We quote in particular:

- Michele Lazzeri (Univ.Paris VI) for the 2n+1 code and Raman cross section calculation with 2nd-order response;
- Andrea Dal Corso for the implementation of Ultrasoft, PAW, noncolinear, spin-orbit extensions to PHonon;
- Mitsuaki Kawamura (U.Tokyo) for implementation of the optimized tetrahedron method in phonon and electron-phonon calculations.

The PlotPhon and QHA packages were contribute by the late Prof. Eyvaz Isaev.

Other contributors include: Lorenzo Paulatto (Univ. Paris VI) for PAW, 2n+1 code; William Parker (Argonne) for phonon terms in dielectric tensor; Tobias Wassmann (Univ. Paris VI) for third-order derivatives of GGA potential.

We shall greatly appreciate if scientific work done using this code will contain an explicit acknowledgment and the following reference:

P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, G. L. Chiarotti, M. Cococcioni, I. Dabo, A. Dal Corso, S. Fabris, G. Fratesi, S. de Gironcoli, R. Gebauer, U. Gerstmann, C. Gougoussis, A. Kokalj, M. Lazzeri, L. Martin-Samos, N. Marzari, F. Mauri, R. Mazzarello, S. Paolini, A. Pasquarello, L. Paulatto, C. Sbraccia, S. Scandolo, G. Sclauzero, A. P. Seitsonen, A. Smogunov, P. Umari, R. M. Wentzcovitch, J.Phys.:Condens.Matter **21**, 395502 (2009), http://arxiv.org/abs/0906.2569

3 Installation

`PHonon` is a package tightly bound to QUANTUM ESPRESSO. For instruction on how to download and compile QUANTUM ESPRESSO, please refer to the general Users' Guide, available in file `Doc/user_guide.pdf` under the main QUANTUM ESPRESSO directory, or in web site `http://www.quantum-espresso.org`.

Once QUANTUM ESPRESSO is correctly configured, `PHonon` can be automatically downloaded, unpacked and compiled by just typing `make ph`, from the main QUANTUM ESPRESSO directory.

3.1 Compilation

Typing `make ph` from the root QUANTUM ESPRESSO directory, or `make` from the `PHonon` directory, produces the following codes:

- `PH/ph.x`: Calculates phonon frequencies and displacement patterns, dielectric tensors, effective charges (uses data produced by `pw.x`).
- `PH/dynmat.x`: applies various kinds of Acoustic Sum Rule (ASR), calculates LO-TO splitting at $\mathbf{q} = 0$ in insulators, IR and Raman cross sections (if the coefficients have been properly calculated), from the dynamical matrix produced by `ph.x`
- `PH/q2r.x`: calculates Interatomic Force Constants (IFC) in real space from dynamical matrices produced by `ph.x` on a regular $\mathbf{q}$-grid
- `PH/matdyn.x`: produces phonon frequencies at a generic wave vector using the IFC file calculated by `q2r.x`; may also calculate phonon DOS, the electron-phonon coefficient λ, the function $\alpha^2F(\omega)$
- `PH/lambda.x`: also calculates λ and $\alpha^2F(\omega)$, plus T_c for superconductivity using the McMillan formula
- `PH/fqha.x`: a simple code to calculate vibrational entropy with the quasi-harmonic approximation
- `Gamma/phcg.x`: a version of `ph.x` that calculates phonons at $\mathbf{q} = 0$ using conjugate-gradient minimization of the density functional expanded to second-order. Only the Γ ($\mathbf{k} = 0$) point is used for Brillouin zone integration. It is faster and takes less memory than `ph.x`, but does not support USPP and PAW.

Links to the main QUANTUM ESPRESSO `bin/` directory are automatically generated.

4 Using PHonon

Phonon calculation is presently a two-step process. First, you have to find the ground-state atomic and electronic configuration; Second, you can calculate phonons using Density-Functional Perturbation Theory. Further processing to calculate Interatomic Force Constants, to add macroscopic electric field and impose Acoustic Sum Rules at $\mathbf{q} = 0$ may be needed. In the following, we will indicate by $\mathbf{q}$ the phonon wavevectors, while $\mathbf{k}$ will indicate Bloch vectors used for summing over the Brillouin Zone.

The main code `ph.x` can be used whenever `PWscf` can be used, with the exceptions of DFT+U, hybrid functionals, external electric fields, constraints on magnetization, nonperiodic boundary conditions. USPP and PAW are not implemented for higher-order response calculations. See the header of file `PHonon/PH/phonon.f90` for a complete and updated list of what `PHonon` can and cannot do.

Since version 4.0 it is possible to safely stop execution of `ph.x` code using the same mechanism of the `pw.x` code, i.e. by creating a file `prefix.EXIT` in the working directory. Execution can be resumed by setting `recover=.true.` in the subsequent input data. Moreover the execution can be (cleanly) stopped after a given time is elapsed, using variable `max_seconds`. See `example/Recover_example/`.

4.1 Single-q calculation

The phonon code `ph.x` calculates normal modes at a given $\mathbf{q}$-vector, starting from data files produced by `pw.x` with a simple SCF calculation. NOTE: the alternative procedure in which a band-structure calculation with `calculation='phonon'` was performed as an intermediate step is no longer implemented since version 4.1. It is also no longer needed to specify `lnscf=.true.` for $\mathbf{q} \neq 0$.

The output data files appear in the directory specified by the variable `outdir`, with names specified by the variable `prefix`. After the output file(s) has been produced (do not remove any of the files, unless you know which are used and which are not), you can run `ph.x`.

The first input line of `ph.x` is a job identifier. At the second line the namelist `&INPUTPH` starts. The meaning of the variables in the namelist (most of them having a default value) is described in file `Doc/INPUT_PH.*`. Variables `outdir` and `prefix` must be the same as in the input data of `pw.x`. Presently you can specify `amass(i)` (a real variable) the atomic mass of atomic type i or you can use the default one deduced from the periodic table on the basis of the element name. If `amass(i)` is not given as input of `ph.x`, the one given as input in `pw.x` is used. When this is `0` the default one is used.

After the namelist you must specify the $\mathbf{q}$-vector of the phonon mode, in Cartesian coordinates and in units of $2\pi/a$.

Notice that the dynamical matrix calculated by `ph.x` at $\mathbf{q} = 0$ does not contain the non-analytic term occurring in polar materials, i.e. there is no LO-TO splitting in insulators. Moreover no Acoustic Sum Rule (ASR) is applied. In order to have the complete dynamical matrix at $\mathbf{q} = 0$ including the non-analytic terms, you need to calculate effective charges by specifying option `epsil=.true.` to `ph.x`. This is however not possible (because not physical!) for metals (i.e. any system subject to a broadening).

At $\mathbf{q} = 0$, use program `dynmat.x` to calculate the correct LO-TO splitting, IR cross sections, and to impose various forms of ASR. If `ph.x` was instructed to calculate Raman coefficients, `dynmat.x` will also calculate Raman cross sections for a typical experimental setup. Input

documentation in the header of `PHonon/PH/dynmat.f90`.

See Example 01 for a simple phonon calculations in Si, Example 06 for fully-relativistic calculations (LDA) on Pt, Example 07 for fully-relativistic GGA calculations.

4.2 Calculation of interatomic force constants in real space

First, dynamical matrices are calculated and saved for a suitable uniform grid of **q**-vectors (only those in the Irreducible Brillouin Zone of the crystal are needed). Although this can be done one **q**-vector at the time, a simpler procedure is to specify variable `ldisp=.true.` and to set variables `nq1`, `nq2`, `nq3` to some suitable Monkhorst-Pack grid, that will be automatically generated, centered at $\mathbf{q} = 0$.

Second, code `q2r.x` reads the dynamical matrices produced in the preceding step and Fourier-transform them, writing a file of Interatomic Force Constants in real space, up to a distance that depends on the size of the grid of **q**-vectors. Input documentation in the header of `PHonon/PH/q2r.f90`.

Program `matdyn.x` may be used to produce phonon modes and frequencies at any **q** using the Interatomic Force Constants file as input. Input documentation in the header of `PHonon/PH/matdyn.f90`.

See Example 02 for a complete calculation of phonon dispersions in AlAs.

4.3 Calculation of electron-phonon interaction coefficients

Since v.5.0, there are two ways of calculating electron-phonon coefficients, distinguished according to the value of variable `electron_phonon`. The following holds for the case `electron_phonon= 'interpolated'` (see also Example 03).

The calculation of electron-phonon coefficients in metals is made difficult by the slow convergence of the sum at the Fermi energy. It is convenient to use a coarse **k**-point grid to calculate phonons on a suitable wavevector grid; a dense **k**-point grid to calculate the sum at the Fermi energy. The calculation proceeds in this way:

1. a scf calculation for the dense **k**-point grid (or a scf calculation followed by a non-scf one on the dense **k**-point grid); specify option `la2f=.true.` to `pw.x` in order to save a file with the eigenvalues on the dense **k**-point grid. The latter MUST contain all **k** and $\mathbf{k} + \mathbf{q}$ grid points used in the subsequent electron-phonon calculation. All grids MUST be unshifted, i.e. include $\mathbf{k} = 0$.

2. a normal scf + phonon dispersion calculation on the coarse **k**-point grid, specifying option `electron_phonon='interpolated'`, and the file name where the self-consistent first-order variation of the potential is to be stored: variable `fildvscf`). The electron-phonon coefficients are calculated using several values of Gaussian broadening (see `PHonon/PH/elphon.f90`) because this quickly shows whether results are converged or not with respect to the **k**-point grid and Gaussian broadening.

3. Finally, you can use `matdyn.x` and `lambda.x` (input documentation in the header of `PHonon/PH/lambda.f90`) to get the $\alpha^2F(\omega)$ function, the electron-phonon coefficient λ, and an estimate of the critical temperature T_c.

See the appendix for the relevant formulae. **Important notice**: the $q \to 0$ limit of the contribution to the electron-phonon coefficient diverges for optical modes! please be very careful, consult the relevant literature. .

4.4 DFPT with the tetrahedron method

Inn order to use the tetrahedron method for phonon calculations, you should run `pw.x` and `ph.x` as follows:

1. Run `pw.x` with `occupation = "tetraaehdra_opt"` and `K_POINT automatic`.
2. Run `ph.x`.

There is an example in `PHonon/example/tetra_example/`.

4.5 Calculation of electron-phonon interaction coefficients with the tetrahedron method

When you perform a calculation of electron-phonon interaction coefficients with the tetrahedron method, you have to use an offset q-point grid in order to avoid a singularity at $q = \Gamma$; you can perform this calculation as follows:

1. Run `pw.x` with `occupation = "tetraehdra_opt"` and `K_POINT automatic`.
2. Run `ph.x` with `lshift_q = .true.` and `electron_phonon = ""` (or unset it) to generate the dynamical matrix and the deformation potential (in `_ph*/{prefix}_q*/`) of each q.
3. Run `ph.x` with `electron_phonon = "lambda_tetra"`. You should use a denser k grid by setting `nk1`, `nk2`, and `nk3`. Then `lambda*.dat` are generated; they contain $\lambda_{q\nu}$.
4. Run `alpha2f.x` with an input file as follows:

```
&input
      ne = (a),
  ltetra = (b),
  fildyn = (b),
  mustar = (d),
  prefix = (e),
/
```

   (a) The number of frequencies for $\alpha^2F(\omega)$
   (b) $= 1$ for the linear tetrahedron method, $= 2$ for the optimized tetrahedron method.
   (c) It must be the same as that in `ph.x` input.
   (d) Coulomb pseudo potential μ^*
   (e) It must be the same as that in `ph.x` input.

   Then $\alpha^2F(\omega)$, λ, and $\omega_{\ln}$ are calculated.

There is an example in `PHonon/example/tetra_example/`.

5 Parallelism

We refer to the corresponding section of the PWscf guide for an explanation of how parallelism works.

ph.x may take advantage of MPI parallelization on images, plane waves (PW) and on **k**-points ("pools"). Currently all other MPI and explicit OpenMP parallelizations have very limited to nonexistent implementation. phcg.x implements only PW parallelization. All other codes may be launched in parallel, but will execute on a single processor.

In "image" parallelization, processors can be divided into different "images", corresponding to one (or more than one) "irrep" or **q** vectors. Images are loosely coupled: processors communicate between different images only once in a while, so image parallelization is suitable for cheap communication hardware (e.g. Gigabit Ethernet). Image parallelization is activated by specifying the option `-nimage N` to ph.x. Inside an image, PW and **k**-point parallelization can be performed: for instance,

```
mpirun -np 64 ph.x -ni 8 -nk 2 ...
```

will run 8 images on 8 processors each, subdivided into 2 pools of 4 processors for **k**-point parallelization. In order to run the ph.x code with these flags the pw.x run has to be run with:

```
mpirun -np 8 pw.x -nk 2 ...
```

without any -nimage flag. After the phonon calculation with images the dynmical matrices of **q**-vectors calculated in different images are not present in the working directory. To obtain them you need to run ph.x again with:

```
mpirun -np 8 ph.x -nk 2 ...
```

and the recover=.true. flag. This scheme is quite automatic and does not require any additional work by the user, but it wastes some CPU time because all images stops when the image that requires the largest amount of time finishes the calculation. Load balancing between images is still at an experimental stage. You can look into the routine image_q_irr inside PHonon/PH/check_initial_status to see the present algorithm for work distribution and modify it if you think that you can improve the load balancing.

A different paradigm is the usage of the GRID concept, instead of MPI, to achieve parallelization over irreps and **q** vectors. Complete phonon dispersion calculation can be quite long and expensive, but it can be split into a number of semi-independent calculations, using options start_q, last_q, start_irr, last_irr. An example on how to distribute the calculations and collect the results can be found in examples/GRID_example. Reference:
Calculation of Phonon Dispersions on the GRID using Quantum ESPRESSO, R. di Meo, A. Dal Corso, P. Giannozzi, and S. Cozzini, in *Chemistry and Material Science Applications on Grid Infrastructures*, editors: S. Cozzini, A. Laganà, ICTP Lecture Notes Series, Vol. 24, pp.165-183 (2009).

6 Troubleshooting

ph.x stops with *error reading file* The data file produced by pw.x is bad or incomplete or produced by an incompatible version of the code. In parallel execution: if you did not set wf_collect=.true., the number of processors and pools for the phonon run should be the same as for the self-consistent run; all files must be visible to all processors.

ph.x mumbles something like *cannot recover* or *error reading recover file* You have a bad restart file from a preceding failed execution. Remove all files `recover*` in `outdir`.

ph.x says *occupation numbers probably wrong* and continues You have a metallic or spin-polarized system but occupations are not set to `'smearing'`.

ph.x does not yield acoustic modes with zero frequency at $\mathbf{q} = 0$ This may not be an error: the Acoustic Sum Rule (ASR) is never exactly verified, because the system is never exactly translationally invariant as it should be. The calculated frequency of the acoustic mode is typically less than 10 cm^{-1}, but in some cases it may be much higher, up to 100 cm^{-1}. The ultimate test is to diagonalize the dynamical matrix with program `dynmat.x`, imposing the ASR. If you obtain an acoustic mode with a much smaller ω (let us say $< 1\mathrm{cm}^{-1}$) with all other modes virtually unchanged, you can trust your results.

"The problem is [...] in the fact that the XC energy is computed in real space on a discrete grid and hence the total energy is invariant (...) only for translation in the FFT grid. Increasing the charge density cutoff increases the grid density thus making the integral more exact thus reducing the problem, unfortunately rather slowly...This problem is usually more severe for GGA than with LDA because the GGA functionals have functional forms that vary more strongly with the position; particularly so for isolated molecules or system with significant portions of "vacuum" because in the exponential tail of the charge density a) the finite cutoff (hence there is an effect due to cutoff) induces oscillations in rho and b) the reduced gradient is diverging."(info by Stefano de Gironcoli, June 2008)

ph.x yields really lousy phonons, with bad or "negative" frequencies or wrong symmetries or gross ASR violations Possible reasons:

- if this happens only for acoustic modes at $\mathbf{q} = 0$ that should have $\omega = 0$: Acoustic Sum Rule violation, see the item before this one.
- wrong data file read.
- wrong atomic masses given in input will yield wrong frequencies (but the content of file fildyn should be valid, since the force constants, not the dynamical matrix, are written to file).
- convergence threshold for either SCF (`conv_thr`) or phonon calculation (`tr2_ph`) too large: try to reduce them.
- maybe your system does have negative or strange phonon frequencies, with the approximations you used. A negative frequency signals a mechanical instability of the chosen structure. Check that the structure is reasonable, and check the following parameters:
  - The cutoff for wavefunctions, `ecutwfc`
  - For USPP and PAW: the cutoff for the charge density, `ecutrho`
  - The **k**-point grid, especially for metallic systems.
- For metallic systems: it has been observed that the convergence with respect to the k-point grid and smearing is very slow in presence of semicore states, and for phonon wave-vectors that are not commensurate i with the k-point grid (that is, $\mathbf{q} \neq \mathbf{k}_i - \mathbf{k}_j$)

Note that "negative" frequencies are actually imaginary: the negative sign flags eigenvalues of the dynamical matrix for which $\omega^2 < 0$.

***Wrong degeneracy* error in star_q** Verify the **q**-vector for which you are calculating phonons. In order to check whether a symmetry operation belongs to the small group of **q**, the code compares **q** and the rotated **q**, with an acceptance tolerance of 10^{-5} (set in routine `PW/eqvect.f90`). You may run into trouble if your **q**-vector differs from a high-symmetry point by an amount in that order of magnitude.

A Appendix: Electron-phonon coefficients

The electron-phonon coefficients g are defined as

$$g_{\mathbf{q}\nu}(\mathbf{k},i,j) = \left(\frac{\hbar}{2M\omega_{\mathbf{q}\nu}}\right)^{1/2} \langle\psi_{i,\mathbf{k}}|\frac{dV_{SCF}}{d\hat{u}_{\mathbf{q}\nu}}\cdot\hat{\epsilon}_{\mathbf{q}\nu}|\psi_{j,\mathbf{k}+\mathbf{q}}\rangle. \tag{1}$$

The phonon linewidth $\gamma_{\mathbf{q}\nu}$ is defined by

$$\gamma_{\mathbf{q}\nu} = 2\pi\omega_{\mathbf{q}\nu}\sum_{ij}\int\frac{d^3k}{\Omega_{BZ}}|g_{\mathbf{q}\nu}(\mathbf{k},i,j)|^2\delta(e_{\mathbf{q},i}-e_F)\delta(e_{\mathbf{k}+\mathbf{q},j}-e_F), \tag{2}$$

while the electron-phonon coupling constant $\lambda_{\mathbf{q}\nu}$ for mode ν at wavevector **q** is defined as

$$\lambda_{\mathbf{q}\nu} = \frac{\gamma_{\mathbf{q}\nu}}{\pi\hbar N(e_F)\omega^2_{\mathbf{q}\nu}} \tag{3}$$

where $N(e_F)$ is the DOS at the Fermi level. The spectral function is defined as

$$\alpha^2F(\omega) = \frac{1}{2\pi N(e_F)}\sum_{\mathbf{q}\nu}\delta(\omega-\omega_{\mathbf{q}\nu})\frac{\gamma_{\mathbf{q}\nu}}{\hbar\omega_{\mathbf{q}\nu}}. \tag{4}$$

The electron-phonon mass enhancement parameter λ can also be defined as the first reciprocal momentum of the spectral function:

$$\lambda = \sum_{\mathbf{q}\nu}\lambda_{\mathbf{q}\nu} = 2\int\frac{\alpha^2F(\omega)}{\omega}d\omega. \tag{5}$$

Note that a factor $M^{-1/2}$ is hidden in the definition of normal modes as used in the code.
McMillan:

$$T_c = \frac{\Theta_D}{1.45}\exp\left[\frac{-1.04(1+\lambda)}{\lambda(1-0.62\mu^*)-\mu^*}\right] \tag{6}$$

or (better?)

$$T_c = \frac{\omega_{log}}{1.2}\exp\left[\frac{-1.04(1+\lambda)}{\lambda(1-0.62\mu^*)-\mu^*}\right] \tag{7}$$

where

$$\omega_{log} = \exp\left[\frac{2}{\lambda}\int\frac{d\omega}{\omega}\alpha^2F(\omega)\log\omega\right] \tag{8}$$