# The PHonon package

**Andrea Dal Corso**

SISSA and IOM DEMOCRITOS
Trieste (Italy)

## Outline

## Quantities calculated by ph.x (28/03/13)

```
! ... dynamical matrix (q/=0)   NC [4], US [4], PAW [4]
! ... dynamical matrix (q=0)    NC [5], US [5], PAW [4]
! ... dielectric constant       NC [5], US [5], PAW [3]
! ... born effective charges    NC [5], US [5], PAW [3]
! ... polarizability (iu)       NC [2], US [2]
! ... electron-phonon           NC [3], US [3]
! ... electro-optic             NC [1]
! ... raman tensor              NC [1]
!
! NC = norm conserving pseudopotentials
! US = ultrasoft pseudopotentials
! PAW = projector augmented-wave
! [1] LDA,
! [2] [1] + GGA,
! [3] [2] + LSDA/sGGA,
! [4] [3] + Spin-orbit/nonmagnetic,
! [5] [4] + Spin-orbit/magnetic (experimental when available)
!
! Not implemented in ph.x:
! [6] [5] + constraints on the magnetization
! [7] [6] + Hubbard U
! [8] [7] + Hybrid functionals
! [9] ? + External Electric field
! [10] ? + nonperiodic boundary conditions.
```

## Input variables that control the flow

```
fpol,              if .TRUE. computes the frequency dependent polarizability
epsil,             if .TRUE. computes the dielectric constant
zeu,               if .TRUE. computes eff. charges as induced forces
lraman,            if .TRUE. computes the raman tensor
elop,              if .TRUE. computes the el-optical coefficient
trans,             if .TRUE. computes the dynamical matrix
zue,               if .TRUE. computes eff. charges as induced polarization
elph               if .TRUE. computes the electron phonon coupling

nfs                ! number of frequencies
fiu(nfs)           ! frequencies in Ry

nq1, nq2, nq3      ! the mesh of q points
xq                 ! the coordinates of a q point

start_iq           ! initial q to calculate
last_iq            ! last q to calculate
start_irr          ! initial representation to calculate
last_irr           ! last representation to calculate

nat_todo           ! the number of atoms to move
atomo(nat_todo)    ! which atoms to move
modenum            ! the response to a single mode
```

## Flow of the code - I

```
1) Read input and set the flags of the quantities to compute
   1.1) Read all the quantities written by pw.x
   1.2) Read the pseudopotential data

2) Decide what must be calculated.
   2.1) If not already on disk, compute the grid of q points and
        all the modes for all q points and save on disk (SD)
   2.2) If image parallelization is requested divide the work among images

3) In a recover run check what is already available on the .xml files and
   sets the appropriate done flags to .TRUE.

4) Start a main loop over the q points:

   4.1) Compute all quantities that do not depend on the response of the system
   NB: the following points are executed only when q is Gamma.
     4.2) Start a loop on the frequencies
          4.2.1) Compute the polarizability as a function of iu SD
     4.3) Compute the response to an electric field
     4.4) Compute epsilon and SD
     4.5) Compute zeu and SD
     4.7) Compute the electro-optic coefficient and SD
     4.6) Compute the second order response to E
     4.8) Compute Raman tensor and SD
   END NB
```

## Flow of the code - II

```
5) Start a loop over the irreducible representation
     5.1) Compute the response to an irreducible representation
     5.1.1) Accumulate the contribution to electron-phonon SD
     5.1.2) Accumulate the contribution to the dynamical matrix
     5.1.3) Accumulate the contribution to zue
     5.1.4) SD this contribution to the dynamical matrix and to zue
continue the loop 5) until all representations of the current q point
have been computed

6) diagonalize the dynamical matrix and SD (only if all representations of
   this q have been computed)

7) Sum over k and bands the electron-phonon couplings to calculate gamma_mat
   SD (only if all representations of this q have been computed)

8) continue the loop at point 4 until all q points have been computed
```

## Grid Control

```
comp_iq(nqs)=.TRUE.                 ! .FALSE. when this q is not computed in
                                    ! this run (controlled by start_iq, last_iq,
                                    ! or by the image controller)

comp_irr_iq(0:3*nat,nqs)=.TRUE.     ! .FALSE. for the representations that are
                                    !  not calculated in this run.
                                    ! (controlled by start_iq, last_iq,
                                    !  start_irr, last_irr,
                                    !  or by the image controller)

comp_iu(nfs)=.TRUE.                 ! .FALSE. for the frequencies not calculated
                                    ! in this run.


done_iq(nqs)=.FALSE.           ! .TRUE. when the dyn. mat. and, if required, the
                               ! electron-phonon coefficients at the q point
                               ! have been calculated

done_bands(nqs)=.FALSE.        ! .TRUE. when the bands for that q are already
                               !  on disk

done_irr_iq(0:3*nat,nqs)=.FALSE. ! The representations that have been already
                                 ! calculated for each q are set .TRUE..
                                 ! The representation 0 is the part of the
                                 ! dynamical matrix computed by drho and
                                 ! dynmat0.

done_iu(nfs)=.FALSE.           ! .TRUE. when the polarization(iu) is available.
```

## Recover

```
!  rec_code   where_rec     status description
!
!    -1000                 Nothing has been read. There is no recover file.
!    -50      init_rep.. All displacement have been written on file.
!    -40      phq_setup  Only the displacements u have been read from file
!    -30      phq_init   u and dyn(0) read from file
!    -25      solve_e_fp all previous. Stopped in solve_e_fpol. There
!                        should be a recover file.
!    -20      solve_e    all previous. Stopped within solve_e. There
!                        should be a recover file.
!    -10      solve_e2   epsilon and zstareu are available if requested.
!                        Within solve_e2. There should be a recover file.
!     2       phescf     all previous, raman tensor and elop tensor are
!                        available if required.
!     10      solve_linter all previous. Stopped within solve linter.
!                        Recover file  should be present.
!     20      phqscf     all previous dyn_rec(irr) and zstarue0_rec(irr) are
!                        available.
!     30      dynmatrix  all previous, dyn and zstarue are available.
```

## Files

The `outdir` directory contains the following files:

```
outdir/prefix.wfc outdir/prefix.save/ outdir/_ph#im
```

When `lqdir=.false.` `outdir/_ph#im` contains:

```
outdir/_ph#im/prefix.wfc  outdir/_ph#im/prefix.save outdir/_ph#im/prefix.phsave
outdir/_ph#im/prefix.bar  outdir/_ph#im/prefix.dwf  outdir/_ph#im/prefix.prd
outdir/_ph#im/prefix.mixd outdir/_ph#im/prefix.recover
```

while when `lqdir=.true.` `outdir/_ph#im` contains:

```
outdir/_ph#im/prefix.q_#iq/...  outdir/_ph#im/prefix.phsave
```

`outdir/_ph#im/prefix.phsave` contains:

```
status_run.xml        control_run.xml       tensors.xml     patterns.#iq.xml
dynmat.#iq.#irr.xml   elph.#iq.#irr.xml
```

## phonon.f90

```
CALL phq_readin()
CALL check_initial_status(auxdyn)
DO iq = 1, nqs
   CALL prepare_q(auxdyn, do_band, do_iq, setup_pw, iq)
   IF (.NOT.do_iq) CYCLE
   IF (setup_pw) CALL run_nscf(do_band, iq)
   CALL initialize_ph()
   IF (epsil) CALL phescf()
   IF ( trans ) THEN
      CALL phqscf()
      CALL dynmatrix_new(iq)
   END IF
   IF ( elph ) THEN
      ....
   END IF
   CALL clean_pw_ph(iq)
END DO
```

## check_initial_status.f90

```
IF (.NOT.recover) THEN
   IF (ldisp) THEN
      CALL q_points()
   ELSE
      nqs = 1
      ALLOCATE(x_q(3,1))
      x_q(:,1)=xq(:)
   END IF
   CALL ph_writefile('init',0,0,ierr)
   CALL init_representations()
ENDIF
CALL allocate_grid_variables()
CALL initialize_grid_variables()
IF (nimage > 1) CALL image_q_irr()
IF (recover) THEN
   CALL check_directory_phsave()
   ...
   CALL check_available_bands()
ENDIF
```

## run_nscf.f90

```
IF (done_bands(iq)) THEN
   CALL clean_pw( .TRUE. )
   CALL close_files(.true.)
   CALL read_file()
   RETURN
ENDIF
CALL clean_pw( .FALSE. )
CALL close_files(.true.)
CALL setup_nscf ( newgrid, xq )
CALL init_run()
!
IF (do_band) THEN
   CALL electrons()
   CALL punch( 'all' )
ENDIF
```

## initialize_ph.f90

```
  CALL allocate_phq()
  CALL phq_setup()
  CALL phq_recover()
  CALL phq_summary()
  CALL openfilq()
  CALL phq_init()

SUBROUTINE phq_init()
  IF ( nlcc_any ) CALL set_drhoc( xq, drc )
  DO nt = 1, ntyp
     CALL setlocq( vlocq ...
  END DO
  DO ik = 1, nksq
     ...
     CALL calbec (npw, vkb, evc, becp1(ik) )
     ...
        CALL calbec (npw, vkb, aux1, alphap(ipol,ik) )
  ENDDO
  CALL dvanqq()
  CALL drho()
  IF ( trans ) CALL dynmat0_new()
```

## dynmat0.f90

```
! first electronic contribution arising from the term  <psi|d2v|psi>
call dynmat_us()
!   Here the ionic contribution
call d2ionq (nat, ntyp, ityp, zv, tau, alat, omega, xq, at, bg, g, &
     gg, ngm, gcutm, nmodes, u, dyn)
!   Add non-linear core-correction (NLCC) contribution (if any)
call dynmatcc()
```

## phqscf.f90

```
DO irr = 1, nirr
   IF ( (comp_irr (irr)) .AND. (.NOT.done_irr (irr)) ) THEN
      npe=npert(irr)
      ALLOCATE (drhoscfs( dfftp%nnr , nspin_mag, npe))
      !
      CALL solve_linter (irr, imode0, npe, drhoscfs)
      !
      IF (convt) THEN
         CALL drhodv (imode0, npe, drhoscfs)
         IF (zue) CALL add_zstar_ue (imode0, npe )
         IF (zue.AND. okvan) CALL add_zstar_ue_us(imode0, npe )
      ELSE
         WRITE( stdout, '(/,5x,"No convergence has been achieved "
      ENDIF
   ENDIF
ENDDO
```

## solve linter.f90

```
DO iter = 1, niter_ph
   drhoscf(:,:,:) = (0.d0, 0.d0)
   DO ik = 1, nksq
      ...
      CALL init_us_2 (npwq, igkq, xk (1, ikq), vkb)
      call davcio (evc, lrwfc, iuwfc, ikk, - 1)
      call davcio (evq, lrwfc, iuwfc, ikq, - 1)
      DO ipert = 1, npe
         IF (iter>1) THEN
            CALL davcio (dvpsi, lrbar, iubar, nrec, - 1)
            CALL apply_dpot(dffts%nnr,aux1, dvscfins(1,1,ipert), current_spin)
         ELSE
            CALL dvqpsi_us (ik, u (1, mode),.false. )
            CALL davcio (dvpsi, lrbar, iubar, nrec, 1)
         ENDIF
         CALL orthogonalize(dvpsi, evq, ikk, ikq, dpsi, npwq)
         call cgsolve_all (ch_psi_all, cg_psi, et(1,ikk), dvpsi, dpsi, ...
         CALL incdrhoscf (drhoscf(1,current_spin,ipert), weight, ik, ...
      ENDDO
   ENDDO
   CALL addusddens (drhoscfh, dbecsum, imode0, npe, 0)
   CALL psymdvscf (npe, irr, drhoscfh)
   CALL dv_of_drho (imode0+ipert, dvscfout(1,1,ipert), .true.)
   CALL mix_potential (2*npe*dfftp%nnr*nspin_mag, dvscfout, dvscfin, &
                    ...
ENDDO
```

## drhodvloc.f90

```
do nu_j = 1, 3 * nat
   call compute_dvloc (nu_j, dvloc)
   do ipert = 1, npe
      nu_i = nu_i0 + ipert
      do is = 1, nspin_lsda
         dynwrk (nu_i, nu_j) = dynwrk (nu_i, nu_j) + &
              zdotc (dffts%nnr, drhoscf (1, is, ipert), 1, dvloc, 1) * &
                omega / (dffts%nr1 * dffts%nr2 * dffts%nr3)
      enddo
   enddo
enddo
```

## More info

Additional information can be found in the directory `PHonon/Doc` of the QE distribution in the file `developer_man.pdf`. Its bibliography contains a list of papers in which the formulas implemented in the `ph.x` code are reported.