0. Current QE status

* "base" or "core" QE
  - Installation utilities
  - Modules and libraries
  - Documentation and testing
  - Tools and sample code (COUPLE)
  - PWscf
  - CP
  - PostProc

* Additional QE packages, residing on the QE svn,
  downloaded by make in the released version.
  Not relevant for this meeting:
  - NEB
  - atomic
  - PWCOND
  - XSpectra
  Relevant for this meeting:
  - PHonon
  - GWL (under directory GWW/) [i]
  - WEST [e]
  - GIPAW [e]
  - EPW   [E]

e = stored in an external svn, can be downloaded into the QE svn
E = as above, no longer downloadable into the QE svn
i = in the released packages, "make" does not install it, must be done manually

1. PHonon

The Phonon packages includes various codes

```
    - PH     general-purpose code and utilities
             depends upon base QE
    - Gamma  Gamma-only, q=0 only code
             depends upon base QE
    - FD     finite-difference code
             depends upon base QE
    - D3     code computing some anharmonic coefficients
             depends upon base QE and PH
```

The focus will be on PH, but I'll first briefly mention Gamma, FD, D3

* PHonon/Gamma:

Simplified linear-response code, implements DFPT by minimizing the energy functional (global minimization) expanded at second order in the perturbation. Conjugate-gradient algorithm, Gamma-only electronic structure, "Gamma tricks". Perturbation at q=0 only, norm-conserving only, nonmagnetic insulators only, computes force constants, effective charges, nonresonant Raman coefficients (with finite differences)

Lean and efficient, but limited in scope, little used. Code is specialized and mostly orthogonal to the rest of PHonon. Simple: good for experiments.

* PHonon/FD:

Produces interatomic force constants with finite differences and supercells.
Uses a combination of fortran tools and scripts. Advantages and disavantages
of frozen-phonon approach. An alternative to PHonon/PH when the latter is not
implemented or does not work.

Too little documentation. Implementation simple but clumsy to use. Relies on
reading output files (maintainability problems). Should either be transformed
into a fortran-only code, or into a python wrapper, or both.

* PHonon/D3:

Produces anharmonic force constants using linear response and 2n+1 theorem.
Computes only C(0,q,-q) coefficients, with norm-conserving PP only.

Rather complex, calls some PH code (broken more than once by changes in PH,
also due to poor testing: there is a single, not so easy to check, example)

2. PHonon/PH

The main PH code ph.x computes
* phonon frequencies and eigenvectors at any q, for
  - a complete q-vector grid needed for interatomic force constants
  - a single wave-vector
  - all, a group, or just one irreducible representations (irreps)
* effective charges and dielectric tensors
* More exotic calculations:
  - dynamical polarizability
  - electron-phonon interactions (old algorithm)
  - electro-optic and nonresonant Raman coefficients with second-order response

From PHonon/PH/phonon.f90:

```
  ! ... This is the main driver of the phonon code.
  ! ... It reads all the quantities calculated by pwscf, it
  ! ... checks if some recover file is present and determines
  ! ... which calculation needs to be done. Finally, it calls do_phonon
  ! ... that does the loop over the q points.
  ! ... Presently implemented:
  ! ... 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] Hubbard U
  ! [8] Hybrid functionals
  ! [9] non-local/semiempirical vdW functionals
  ! [10] External Electric field
  ! [11] nonperiodic boundary conditions.
```

Reasons for non-implementation:
- almost always, mathematical and algorithmical complexity

Perspectives for future implementation
- DFT-D2: simple stuff, might come soon
- Nonlocal vdW-DF functionals: currently in a branch
- Hubbard U: an implementation exists, to be aligned with svn
- Raman tensor and electro-optical coefficients: extension to GGA
  requires 2nd-order derivative of Vxc, worth it?
- Hybrid functionals: already present in TDDFPT
- Most of the remaining cases: presumably perspectives are nonexistent

3. Parallelization

ph.x is parallelized on
- "images", using MPI, or "grid-like": a complete dispersion calculation can be
  split into independent calculations, distributing wave-vectors and irreps
- k-points (aka "pools") with the same restricstions as in pw.x
- plane waves (R- and G-grids) with the same logic as in pw.x
- "task groups" (bands in all-band FFT operations) partially (and in a rather
  obscure way)

ph.x is NOT parallelized on
- linear algebra (does not apply in the linear-responsa calculation, but it
  might be useful in the initial non-scf calculation)
- OpenMP threads, not explicitly ar least (but it can use OpenMP-parallel
  libraries and might use OpenMP-aware parts of QE)

Parallelization efficiency: as good as it gets
- Image parallelization has load balancing issues
- K-point parallelization is currently limited to k-points of the scf case
  (at least when computing all wave-vectors)
- plane-wave parallelization has the known behavior and limitations

4. Stability and algorithmic efficiency

ph.x is relatively efficient (that is: wrt the scf calculation) for
norm-conserving pseudopotentials. Factors limiting efficiency are
- self-consistency algorithm less efficient than the one of pw.x; moreover
  the good value of the scf threshold (tr2_ph) is hard to quantify and it
  is not easily related to any error estimate like in pw.x
- long-wavelength q tend to converge slowly, sometimes not at all (there are
  even cases of divergence)
- symmetry lowering at finite q leads to an increased k-point number
  (the current algorithm requires Irreducible Brillouin Zone wrt symmetry of q)
- memory requirements are quite larger than for the scf calculation
- a significant amount of scratch I/O is done

ph.x is very inefficient IMHO for ultrasoft pseudopotentials and PAW.
Factors negatively impacting efficiency are, in addition to those above:
- for obscure reasons, the needed scf threshold (tr2_ph) is MUCH smaller
  than the one that usually sufficient for norm-conserving pseudopotentials
- in spite of the much smaller cutoff than for NCPP, memory requirements are
  still significant, due to non-scalable arrays allocated on all processors
- the calculation of US- and PAW-specific terms takes a very significant
  amount of time, making the advantage of a lower cutoff not obvious
- at least with some USPPs, convergence wrt the cutoff on the charge density
  is much worse than for the energy in a scf calculation, sometimes to the
  point of making the calculation unfeasable or not convenient anyway

ph.x occasionally suffers from numerical problems:
- badly wrong frequencies if
  - pseudopotentials have semicore states, and
  - the system is metallic, and
  - a wave-vector q is not commensurate with the k-point grid (which means
    that there is no equivalent supercell describing such perturbation)
  AFAIK this problem is still unsolved, but it seems to be intrinsic and
  to disappear only for exceedingly denseak-point grids.
- the infamous and likely unsolvable Acoustic-Sum-Rule violation problem,
  leading to nonzero frequencies at q=0

In practice, it is not easy to run PH for systems of more than a few tens of
atoms, especially if the system is not a "simple" one (metallic, with hard
elements, noncolinear/spinorbit ...). In the latter cases, even systems of
a few tens of atoms are highly nontrivial to run, even on powerful machines.

5. Code maintainability

The phonon code is relatively large and complex, with 45000+ lines of code
(for reference: Gamma 4030 lines, FD 2150 lines). It uses a lot of code
from QE (PWscf, modules, libraries) and a large amount of QE variables in
addition to its own set. In turn, it is used by other linear-response codes,
notably: D3, TDDFPT, GWL, GIPAW, so it cannot be changed without some care.

Reasons for complexity:
- the (possibly obsolete) "one k-point at the time" style of calculation
  inherited from PWscf requires the initialization of a bunch of variables
  (or reading a bunch of files) at every loop on k-points

- calculations at finite q introduce the need for q-dependent quantities such as indices, projectors and wavefunctions for k+q, symmetry operations, etc
- symmetrization uses a rather complex algorithm (symmetrize inside an irrep with all k-points in the small group of q) that strikes a compromise between needed computation and needed RAM
- USPP-PAW calculations introduce a large number of additional terms everywhere
- magnetization with LSDA/noncolinear/spinorbit introduces several versions of the same sections of code
- restarting is made complex by the large number of variables to be saved and re-read
- the code has several different execution modes: from one irrep and one wave-vector at the time, to all irreps and wave-vectors; with or without electron-phonon coefficients (which can be done in one or in two steps)...

6. Phonon usage and perspective

Hits on mailing list archives https://www.mail-archive.com/pw_forum@pwscf.org/ (26656 messages):

phonon	3565
frequencies	819
electron-phonon	620
raman	416
DFPT	126
D3	121
infrared	43

While not a sign of overwhelming interest, not bad compared to:

pwscf	5989
relax	2686
CP	659
MD	582
NEB	390

The baseline:
- there is some interest on the phonon code. Is it sufficient to warrant a serious time investment in its extension and optimization?
- there is ample space for better optimization and parallelization, but there are also some serious numerical problems as well
- there is ample space for refactoring, but the complexity of the phonon code is here to stay