THERMO_PW User's Guide (v.1.5.0)

Andrea Dal Corso (SISSA - Trieste)

Contents

1 Introduction

This guide covers the installation and usage of the thermo_pw package. It assumes that you have some familiarity with the QUANTUM ESPRESSO package. If not please consult the web site: http://www.quantum-espresso.org.

thermo_pw computes material properties. At low level it calls QUANTUM ESPRESSO routines and, at high level, it has pre-processing tools to reduce the information provided by the user and post-processing tools to convert the output of QUANTUM ESPRESSO into plots of material properties directly comparable with experiment.

thermo_pw has the following directory structure, contained in a subdirectory thermo_pw/ that should be put in the root directory of the QUANTUM ESPRESSO tree:

Doc/	: contains this user's guide and other documentation
examples/	: some examples
examples_qe/	: QUANTUM ESPRESSO examples run using thermo_pw
inputs/	: a collection of useful inputs
pseudo_test/	: a collection of inputs to test a pseudopotential library
space_groups/	: a collection of structures for many space groups
lib/	: source files for modules used by thermo_pw
src/	: source files for thermo_pw
tools/	: source files for auxiliary tools
tools_input/	: examples of inputs for the auxiliary tools

The thermo_pw package can calculate the following quantities:

- Plot of the Brillouin zone (the structure can be seen by reading the input of thermo_pw by the XCrySDen program).
- Plot of the X-rays powder diffraction pattern of the input crystal.
- Total energy at fixed geometry.
- Total energy as a function of the kinetic energy cut-off.
- Total energy as a function of **k**-points and smearing.
- Electronic band structure at fixed geometry.
- Electronic density of states at fixed geometry.
- Electronic heat capacity as a function of temperature (for metals only).
- Complex dielectric constant as a function of the complex frequency ω at fixed geometry. Complex index of refraction for all systems except monoclinic and triclinic. Reflectivity at normal incidence and adsorption coefficient for cubic solids.
- Inverse dielectric constant at a given wavevector **q** as a function of the complex frequency ω at fixed geometry.
- Phonon frequencies at fixed geometry.

- Phonon dispersions at fixed geometry and computation of the harmonic thermodynamic properties: vibrational energy, vibrational free energy, vibrational entropy, and constant volume heat capacity as a function of temperature. Atomic Debye-Waller factors as a function of temperature.
- Frozen ions and relaxed ions elastic constants at fixed geometry.
- Relaxed ions temperature dependent elastic constants at fixed unperturbed geometry.
- Fit of the total energy as a function of the lattice parameters with a quadratic or quartic polynomial and determination of equilibrium lattice parameters. Murnaghan fit.
- Electronic band structure at the minimum of the total energy.
- Electronic density of states at the minimum of the total energy.
- Complex dielectric constant as a function of the complex frequency ω at the minimum of the total energy. Complex index of refraction for all systems except monoclinic and triclinic. Reflectivity at normal incidence and adsorption coefficient for cubic solids.
- Inverse dielectric constant at a given wavevector $\mathbf{q}$ as a function of the complex frequency ω at the minimum of the total energy.
- Phonon frequencies at the minimum of the total energy.
- Phonon dispersions and harmonic thermodynamic quantities at the minimum of the total energy.
- Frozen ions and relaxed ions elastic constants at the minimum of the total energy.
- Anharmonic properties within the quasi-harmonic approximation: lattice parameters, thermal expansion tensor, volume, volume thermal expansion, and constant strain heat capacity as a function of temperature; phonon frequencies and mode Grüneisen parameters interpolated at a given geometry or at the equilibrium geometry at a given temperature (limited to cubic, tetragonal, orthorhombic, and hexagonal systems). Bulk modulus and pressure derivative of the bulk modulus, isobaric heat capacity, isoentropic bulk modulus, and average Grüneisen parameter as a function of temperature (limited to cubic systems). Minimum Helmholtz (or Gibbs at finite pressures) free energy as a function of temperature.
- Isothermal and isoentropic elastic constants and elastic compliances as a function of temperature within the "quasi-static" approximation.
- Isothermal and isoentropic elastic constants and elastic compliances as a function of temperature within the "quasi-harmonic" approximation.

- Surface band structure identification and plot of the projected bulk band structure.

`thermo_pw` can run on both serial and parallel machines using all the parallellization options of QUANTUM ESPRESSO. Moreover, `thermo_pw` can run using several images. When possible, the image parallelization is used in an asynchronous way. One image takes the role of master and distributes the work to all the images that carry it out independently. Presently the total energies of several geometries for the determination of the equilibrium geometry are calculated in parallel when there are several images. Stresses or total energies at different strained geometries needed for the calculation of the elastic constants are calculated in parallel. The phonon calculations are carried out in parallel, each image doing one irreducible representation of one **q** point. For frequency dependent calculation, each frequency, or group of frequencies, can be calculated in parallel by different images. The phonon dispersions of several geometries needed for the quasi-harmonic calculation of the thermodynamic properties or of the elastic constants can be calculated in parallel (one geometry at a time or all geometries together).

2 People

The `thermo_pw` code is designed, written, and maintained by Andrea Dal Corso (SISSA - Trieste). It is an open source code distributed, as is, within the GPL license.

I would like to thank all the people that contributed with comments, requests of improvements, and bug reports. Some people also found bug corrections or wrote new routines and shared their improvements with me. In particular I would like to mention M. Palumbo, O. Motornyi, A. Urru, and C. Malica.

thermo_pw	QUANTUM ESPRESSO	release date
1.5.0	6.7	19/07/2021
1.4.1	6.7	29/12/2020
1.4.0	6.6	22/12/2020
1.3.2	6.6	17/8/2020
1.3.1	6.6	13/8/2020
1.3.0	6.5	12/8/2020
1.2.1	6.5	23/1/2020
1.2.0	6.4.1	28/12/2019
1.1.1	6.4.1	16/04/2019
1.1.0	6.4	16/04/2019
1.0.9	6.3	6/3/2019
1.0.0	6.3	17/7/2018
0.9.9	6.2.1	5/7/2018
0.9.0	6.2.1	20/12/2017
0.8.0	6.2	24/10/2017
0.8.0-beta	6.2-beta	31/8/2017
0.7.9	6.1	06/7/2017
0.7.0	6.1	18/3/2017
0.6.0	6.0.0	5/10/2016
0.5.0	5.4.0	26/4/2016
0.4.0	5.3.0	23/1/2016
0.3.0	5.2.1, 5.2.0	23/6/2015
0.2.0	5.1.2	13/3/2015
0.1.0	5.1.1	28/11/2014

Table 1: Compatibility between the versions of `thermo_pw` and of `QE`.

3 Installing, Compiling, and Running

3.1 Installing

The `thermo_pw` package is tightly bound to QUANTUM ESPRESSO. It cannot be compiled without it. To download and compile QUANTUM ESPRESSO, please refer to the general User's Guide, available in the file `Doc/user_guide.pdf` in the root QUANTUM ESPRESSO directory, or on the web site

`http://www.quantum-espresso.org`.

The main distribution page of `Thermo_pw` is

`https://dalcorso.github.io/thermo_pw/`

where you can download one of the `.tar.gz` files. Please match carefully `thermo_pw` and QUANTUM ESPRESSO versions as illustrated in Table 1. For the versions of QUANTUM ESPRESSO not listed here, there is no `thermo_pw` package. The source files of `thermo_pw` can be obtained by unpacking the `.tar.gz` file in the root QUANTUM ESPRESSO directory, for instance with the

command `tar -xzvf thermo_pw.1.4.1.tar.gz`.
You can also download the `git` version of `thermo_pw` as described at the web page: `https://dal corso.github.io/thermo_pw/thermo_pw_help.html`. The `git` version of `thermo_pw` contains the most recent features and bug fixes but it might work only with the `git` version of QUANTUM ESPRESSO and its use for production is not recommended.
Please read the web page:

`http://dalcorso.github.io/thermo_pw/thermo_pw_help.html`

for updated information about the compatibility between the `git` version of `thermo_pw` and QUANTUM ESPRESSO. This web page contains also information on critical bugs found in the `thermo_pw` package and should be consulted before using `thermo_pw`.

3.2 Compiling

In order to compile `thermo_pw`, the main `Makefile` and the files `install/makedeps.sh` and `install/plugins_makefile` of QUANTUM ESPRESSO must be changed. This is done by giving the command `make join_qe` inside the `thermo_pw` directory. This command exchanges the `thermo_pw` files with those of the QUANTUM ESPRESSO package.

Typing `make thermo_pw` inside the main QUANTUM ESPRESSO directory, or `make` inside the `thermo_pw` directory, produces the executable `thermo_pw/src/thermo_pw.x` that appears in the QUANTUM ESPRESSO `bin/` directory. A few other tool codes are produced as well and linked in the QUANTUM ESPRESSO `bin/` directory.

`thermo_pw` has been written on a PC with the Linux operating system using a `gfortran` compiler and `openMPI` parallelization. It has been run in parallel on a Linux cluster with several hundreds processors. It has not been tested with other combinations of computer/operating system, but it is supposed to run on the same systems where QUANTUM ESPRESSO runs. If you have a machine in which you can compile and run QUANTUM ESPRESSO but not `thermo_pw`, please report the problem.

3.3 Searching help and reporting bugs

For problems installing `thermo_pw` or for help with some of its features, please subscribe to the `thermo_pw-forum` mailing list (`https://lists.quantum-espresso.org/mailman/listinfo/thermo_pw-forum`). Requests of new features are also welcome. If you think you have found a bug in `thermo_pw`, you can report it to the mailing list or write me: `dalcorso.at.sissa.it`.

3.4 Uninstalling

In order to remove `thermo_pw`, give the command `make leave_qe` in the `thermo_pw` directory. Then just remove the directory. Note that the com-

mand `make leave_qe` is needed to restore the original QUANTUM ESPRESSO files.

3.5 Running `thermo_pw`

In order to run `thermo_pw`, you need an input for `pw.x`, a file called `thermo_control`, and an input for `ph.x` (which must be called `ph_control`) if required by the task. These files must be in your working directory. The input of `pw.x` can have any name and is given as input to the `thermo_pw` code. It is better not to specify an `outdir` directory in the `ph.x` input. Specifying an `outdir` directory is not forbidden, but for some tasks `thermo_pw.x` might add a geometry number to `outdir` and the `outdir` written in the `ph.x` input must be consistent.

A typical command for running `thermo_pw` is:

```
mpirun -n np thermo_pw.x -ni ni ... < input_pw > output_thermo_pw
```

where `np` is the number of processors and `ni` is the number of images. The dots indicate the other QUANTUM ESPRESSO parallelization options that you can find in its manual.

Note that it is very easy to waste resources using too many images. Unused images wait for the working images to complete their tasks wasting cpu-time in an endless loop. Some options do not use the image feature, so you have to know how the calculation is divided and the number of images must not be larger than the number of tasks (below I give this number for each option). If you have doubts on this point use one image (`ni=1`).

The outputs of the `thermo_pw` code are one or more `postscript` or `pdf` files with plots of the material properties. `thermo_pw` produces also files with the data of the plot and scripts for the `gnuplot` program. Usually, the user does not need to modify these files, but they allow the improvement of the figures when needed. The plot of the Brillouin zone (BZ) is made with the help of the `asymptote` code. `Thermo_pw` produces a script for the `asymptote` code and can also run it to produce the `pdf` file of the BZ. Starting from version 1.5.0 `thermo_pw` writes also a script to plot the BZ using the `freeCAD` software.

4 Input variables

The pw.x and ph.x input files are described in the QUANTUM ESPRESSO documentation. In this section we discuss only the creation of the file thermo_control. This file contains a namelist:

```
&INPUT_THERMO
  what=' ',
  ...
 /
```

The what variable controls the sequence of calculations made by thermo_pw. For each possible value of what, we discuss briefly the input variables that you can use to control the output plots. Usually default values of the input variables are sufficient to carry out the basic thermo_pw tasks and you are not supposed to set any variable except what, but in some cases these input variables give you more control on the calculation and can be used to tune its accuracy.

thermo_pw actually writes on file the data to plot and writes a script to plot these data. The output postscript or pdf files are produced by invoking the gnuplot program. Usually any modern Linux distribution provides a package to install this code, or has it already installed. If not, you can download it from http://www.gnuplot.info/.
The following input variables control the use of the gnuplot code:

```
lgnuplot     : if .TRUE. gnuplot is called from within the
               program and the postscript or pdf files are
               immediately available. Otherwise give the
               command gnuplot gnuplot_files/*.
               Default: logical .TRUE.
gnuplot_command  : the command used to call gnuplot.
               Default: character(len=*) 'gnuplot'
flgnuplot    : initial part of the name of the files where
               gnuplot scripts are written.
               Default: character(len=*) 'gnuplot.tmp'
flext        : extension of the output files. Presently .ps
               and .pdf are supported for postscript or pdf
               output. The latter is available only if gnuplot
               supports the pdfcairo terminal.
               Default: character(len=*) '.ps'
```

If your system has not gnuplot you can disable the production of the postscript or pdf files and use other graphical tools to do plots from the output data.

4.1 Temperature and pressure

Several quantities in thermo_pw can be calculated as a function of temperature. Moreover the equilibrium geometry can be searched at fixed pressure

minimizing the enthalpy instead of the energy. For the options where these features are active, the values of temperature and pressure are controlled by the following variables:

```
tmin        : minimum temperature.
              Default: real 1 K
tmax        : maximum temperature.
              Default: real 800 K
deltat      : interval between two temperatures. Be careful
              with this value because it is used also to
              compute temperature derivatives numerically.
              Too small or too large values could give
              inaccurate anharmonic properties.
              Default: real 3 K
ntemp       : number of temperatures
              Default: integer determined from previous data
pressure    : The external pressure. The crystal parameters are
              calculated minimizing the enthalpy at this
              pressure. Given in kbar units.
              Default: real 0.0 kbar
```

Note that when you fix the external pressure, the geometries chosen to fit the enthalpy must be about the minimum geometry at that pressure.

4.2 Coordinates and structure

The `thermo_pw` code requires the Bravais lattice of the solid. Moreover for computing some quantities it assumes that the direct lattice vectors are those provided by the routine `latgen.f90` of the QUANTUM ESPRESSO distribution. For this reason it is not recommended to use `ibrav=0` in the `pw.x` input. The preferred method is to give the value of `ibrav` and use the primitive vectors provided by QUANTUM ESPRESSO. It is also possible to specify the `space_group` number and the coordinates of the nonequivalent atoms. When the `pw.x` input contains the `ibrav=0` option, `thermo_pw` writes on output the values of `ibrav`, `celldm`, and of the atomic coordinates that should be used in the input of `pw.x` to simulate the same solid and stops. There are however two input variables of `thermo_pw` that can modify this behavior:

```
continue_zero_ibrav : when ibrav=0 in the input of pw.x and
              this variable is set to .TRUE. thermo_pw runs
              with ibrav=0 (not recommended except when you
              deal with a supercell). When this variable is
              .FALSE. and ibrav=0 the behavior depends on
              find_ibrav.
              Default : logical .FALSE.

find_ibrav  : This variable is active only when continue_zero_
              ibrav=.FALSE.. When this variable is set to .TRUE.
```

```
                and the input of pw.x has ibrav=0, thermo_pw finds
                the values of ibrav, celldm, and of the atomic
                positions that produce the same crystal and
                continue the calculation. The geometry used by
                thermo_pw might be rotated with respect to the
                input and have different primitive vectors. When
                this variable is .FALSE. the code stops after
                writing in output ibrav, celldm, and the atomic
                positions. These variables can be copied in the
                pw.x input. Note that the automatic identification
                of the lattice does not work for supercells.
                Default : logical .FALSE.
```

4.3 WHAT='SCF'

With this option the code computes only the total energy. This is a single calculation as if running `pw.x` with the given input. No other input variable is necessary. An example for this option can be found in `example01`.
Number of tasks for this option: `1`.

4.4 WHAT='SCF_KE'

With this option the code makes several self-consistent calculations, in parallel on several images, varying the kinetic energy cut-off for the wavefunctions and for the charge density. In the input of `pw.x` one specifies the minimum values for these two cut-offs. These values are then increased in fixed intervals controlled by the following variables. The energy is then plotted as a function of the wavefunctions kinetic energy cut-off, a different curve for each value of the charge density cut-off.
The variables that control this option are:

```
nke         : number of kinetic energies tested for the
              wavefunctions cut-off.
              Default: integer 5
deltake     : delta of wavefunctions kinetic energy cut-off
              in Ry. (can be either positive or negative)
              Default: real 10 Ry
nkeden      : number of kinetic energies tested for the
              charge density cut-off.
              Default: integer 1
deltakeden  : delta of charge density kinetic energy
              cut-off in Ry. (can be either positive or
              negative)
              Default: real 100 Ry.
flkeconv    : name of the file where the data with the
              total energy as a function of the kinetic
              energy is written.
```

```
              Default: character(len=*) 'output_keconv.dat'
flpskeconv :  name of the postscript file with the plot
              of the total energy as a function of the
              kinetic energy cut-off.
              Default: character(len=*) 'output_keconv'
```

An example for this option can be found in `example10`.
Number of tasks for this option: `nke * nkedens`.

4.5 WHAT='SCF_NK'

With this option the code makes several self-consistent calculations, in parallel on several images, varying the size of the **k**-point grid, and optionally for metals the smearing parameter `degauss`. In the input of `pw.x` the minimum value of these parameters is given and these values are increased in fixed intervals controlled by the following variables. On output the energy is plotted as a function of the mesh size, one curve for each smearing parameter.
The variables that control this option are:

```
nnk         : the number of different values of nk to test
              Default: integer 5
deltank(3)  : the interval between nk values. All three values
              of nk1, nk2, and nk3 are updated simultaneously.
              Default: integer 2  2  2
nsigma      : the number of smearing intervals.
              Default: integer 1
deltasigma  : the distance between different smearing values.
              (can be either positive or negative)
              Default: 0.005 Ry
flnkconv    : file where the data with the k point convergence
              is written
              Default: character(len=*) 'output_nkconv.dat'
flpsnkconv  : name of the postscript file with the k points
              convergence plot.
              Default: character(len=*) 'output_nkconv'
```

An example for this option can be found in `example11`.
Number of tasks for this option: `nnk * nsigma`.

4.6 WHAT='SCF_BANDS'

With this option the code makes a self-consistent calculation followed by a band structure calculation. There is no image parallelization and no advantage to use several images. The output of the band structure calculation is further processed in order to produce a plot of the band structure. The zero of the energy is the highest valence band of the first **k** point in insulators and the Fermi energy in metals.
The energy bands plot can be modified by the following variables:

```
emin_input : minimum energy for the band dispersion plot (in eV).
             Default: real minimum of the bands
emax_input : maximum energy for the band dispersion plot (in eV).
             Default: real maximum of the bands
nbnd_bands : the number of bands in the band calculation
             Default: integer 2*nbnd, where nbnd is the number
             of bands given in pw.x input or calculated by pw.x.
only_bands_plot: if the files with the bands and the represen-
             tations are already on files, this option allows to
             change the parameters of the plot (such as the
             maximum or minimum energy) and do another plot
             without additional calculation. If the files are
             missing and this variable is .TRUE. an error
             occurs. Note that using this option you cannot
             change the path.
             Default: logical .FALSE.
lsym       : if .TRUE. does the symmetry analysis of the bands
             Default: .TRUE.
enhance_plot: if .TRUE. writes on the band plot the point
             group labels, and colors with different background
             colors lines at the zone border.
             Default: .FALSE.
long_path  : if .TRUE. plots the bands in all the Brillouin
             zone path. Otherwise makes a faster calculation
             on a short path. The short path is indicated also
             for two-dimensional layers perpendicular to the
             z direction.
             Default: .TRUE.
old_path   : if .TRUE. use an alternative path, usually more
             similar to the one used in experimental papers
             (available only for a few lattices).
             Default: .FALSE.
path_fact  : A factor that multiply the number of points along
             each line of the default path. Note that this is
             a real number so you can also decrease the default
             number of points along each line.
             Default: real 1.0
filband    : file where the bands are written in the QE format
             Default: character(len=*) 'output_band.dat'
flpband    : file(s) where the bands are written in gnuplot
             format.
             Default: character(len=*) 'output_pband.dat'
flpsband   : postscript file with the electronic band structure
             Default: character(len=*) 'output_band'
```

Number of tasks for this option: 1.

By default, the bands are plotted along a fixed path in the Brillouin zone,

but the user can modify this behavior giving the path at the end of the INPUT_THERMO namelist with the same format used for the pw.x input. The automatic path generation is not available for base-centered monoclinic and for triclinic Bravais lattices. For these lattices the path must be given explicitly. The following variables control the path:

```
q_in_band_form   : only the first and last point of each k
                   path are given. The weight of each k point
                   is an integer, the number of points in the
                   line that starts at this k point.
                   Default: logical .TRUE.
q_in_cryst_coord : the k - points are given in crystal
                   coordinates. For centered lattices the
                   crystal coordinates refer to the primitive
                   cell (not the conventional one). Same
                   convention as in QE.
                   Default: logical .FALSE.
point_label_type : the label definition (see the BZ manual)
                   Default: SC
q2d              : the q points define a rectangle in
                   reciprocal space. See the QE guide for
                   more details.
                   Default: logical .FALSE.
is_a_path        : if .TRUE. the q points are in a path in
                   reciprocal space. This is usually the case
                   except when q2d=.TRUE. or when the input
                   points are in an arbitrary order. Set this
                   to .FALSE. only if you want to skip the plot
                   of the bands.
                   Default: logical .TRUE.
```

Note that the path cannot be given in the input of pw.x that must contain the information to generate the mesh of **k** points for the self-consistent calculation. An example for this option can be found in example02. If you give explicitly the path, be careful with options that require geometry changes (see below). Only automatic paths, or path given through letter labels are easily recalculated. The other paths could turn out to be correct only for one geometry.

It is also possible to separate the self-consistent and the band calculation, by running first thermo_pw.x using what='scf' and then running, on the same directory, thermo_pw.x using what='scf_bands'. The same input can be used in the two calculations, only the thermo_control file need to be changed. The number or processors and pools can be changed in the same cases in which this is possible in pw.x. You cannot however run twice thermo_pw.x on the same directory using what='scf_bands' and two different paths.

4.7 WHAT='SCF_2D_BANDS'

With this option the code calculates the bands after a self-consistent calculation as with the option `what='scf_bands'`, but it assumes that the cell contains a slab with surfaces perpendicular to the z direction. Therefore the two-dimensional Bravais lattice of the surface is identified and the default path is chosen on the two-dimensional Brillouin zone. There are three options: Plot of the projected band structure (PBS); plot of the bands of the slab; plot of the bands of the slab above the projected band structure (the Fermi energies are aligned). In the first case the code computes several paths of **k**-points parallel to the surface (at different k_z) and does not plot the individual bands but selects the energy regions in which there are bulk states. The second case is similar to a standard band plot. The default path contains only **k**-points parallel to the surface (with $k_z = 0$). The third case assumes that the projected band structure has been already calculated and the information to plot it can be found on the file `flpbs`. For the rest it is similar to case two. For each direction, bands belonging to different irreducible representations of the point co-group of **k** can be plotted in the same panel or on different panels.
This option is controlled by the following variables:

```
lprojpbs: When .TRUE. the projected band structure (PBS) is
     calculated if nkz > 1 otherwise it is read from file.
     Usually this variable is .TRUE.. Set it to .FALSE. if
     you do not want to see the PBS, or if you want to see
     the bands of a bulk projected on the surface Brillouin
     zone without the PBS.
     Default: logical .TRUE. (forced to .FALSE. if what is
     not 'scf_2d_bands')
nkz: The number of k_z values used for the PBS plot.
     If lprojpbs is .FALSE. a plot of the bulk bands projected
     on the surface Brillouin zone is produced.
     Default: integer 4 (forced to 1 if what is not
     'scf_2d_bands').
gap_thr: minimum size (in eV) of the gaps in the PBS
     Default: real 0.1 eV
sym_divide: When .TRUE. the bands belonging to different
     irreducible representations are plotted in different
     panels. This option can be controlled by variables
     specified in the path (see below)
     Default : logical .FALSE.
identify_sur: When .TRUE. the surface bands are searched
     and identified on the surface band structure.
     Default : logical .FALSE.
dump_states: If .TRUE. and identify_sur is .TRUE. dump
     on the file 'dump/state_k_#' the planar averages of
     the density (and in the noncollinear case also of
     the magnetization density) of each state. One file for
     each k point is produced and # is the number of
```

```
     the k points. (Use with a small number of k points
     or it might create quite large files).
     Default: logical .FALSE.
sur_layers: The number of surface layers on which we add
     the charge density of each state to check if it is
     a surface state.
     Default : integer 2
sur_thr: the threshold (in percentage) of the charge
     density that must be on the surface layers to
     identify a state as a surface state.
     Default: calculated from the actual charge density
     values of the states.
sp_min : minimum distance between layers. Two atoms
     form different layers only if their distance along
     z is larger than this number. Should be smaller
     than the interplanar distance (in a.u.) written by
     the tool gener_3d_slab.
     Default : real 2.0 a.u.
subtract_vacuum: if .TRUE. the charge density of each
     state on vacuum is subtracted (to remove the vacuum
     states that are confused with surface states)
     Default : .TRUE.
force_bands: when .TRUE. the bands are plotted in any case.
     Used to plot the bulk bands on top of the PBS,
     mainly for debugging.
     Default: logical .FALSE.
only_bands_plot: if the files with the bands, the
     representations, the pbs and the projections are already
     on files, this option allows to change the parameters of
     the plot (such as the maximum energy or sur_thr) and
     do another plot without any additional calculation. If the
     files are missing and this variable is .TRUE. an error
     occurs.
     Default: logical .FALSE.
flpbs: the name of the file that contains the information on
     the projected band structure.
     Default: character(len=*) 'output_pbs'
flprojlayer: the name of the file that contains the information
     of the projection of the charge density of each state on
     each layer. Calculated only when identify_sur is .TRUE..
     Default: character(len=*) 'output_projlayer'
```

The bands and the gnuplot scripts are saved on the same files that would be used with the option `what='scf_bands'`.
Number of tasks for this option: `1`. Image parallelization is not useful with this option.

By default the symmetry separation is not carried out. The code plots the

bands of the slab on the same panel with a different color for each representation as in the bulk band structure plot (color refer to the representations of the slab point co-group of **k**). In order to plot in different panels the different representations the user can specify `sym_divide= .TRUE.`. By default this option is disabled and its use is rather tricky. In order to use it you must indicate explicitly the path on the two dimensional Brillouin zone using the option `q_in_band_form=.TRUE.`. Close to the starting point of a given line you indicate the number of representations for that line (0 means all representations) and which ones. For instance for a (111) surface of an fcc metal in the direction $\bar{\Gamma} - \bar{M}$ you may want to plot separately the states even or odd with respect to the mirror plane perpendicular to the surface that contains the $\bar{\Gamma} - \bar{M}$ line. In order to do so you can specify the path as follows:

```
5
gG    30   0
K     30   0
M     30   1   1
gG    30   1   2
M      1   0
```

The representations to plot are indicated by their numbers (in this case 1 or 2). The number of the representation and the point co-group of each **k**-point can be found in the output of `thermo_pw`. These representation numbers refer to the point co-group of each **k**-point in the slab when you plot the slab band structures and to the point co-group of each **k**-point in the bulk when you plot a PBS. Some particular values of k_z, such as $k_z = 0$ might have a point co-group in the bulk different from the point co-group of a point with a generic k_z but in this case the representations are transformed into those of the smaller group using the group-subgroup relationships and the symmetry descent of the irreducible representations (only when `sym_divide=.TRUE.`). The representations of the smaller point co-group have to be used in the PBS input.

In general, the point co-group of a **k**-point $\mathrm{k} = (\mathrm{k}_\parallel, k_z)$ with component $\mathrm{k}_\parallel$ parallel to the surface and a generic k_z in the bulk is different from the point co-group of a **k**-point $\mathrm{k} = (\mathrm{k}_\parallel, 0)$ in the slab. Moreover, experimentally one cannot consider symmetries of the slab that exchange the two surfaces, and therefore the point co-group a **k**-point $\mathrm{k} = (\mathrm{k}_\parallel, 0)$ on the surface is a subgroup of the point co-group of $\mathrm{k} = (\mathrm{k}_\parallel, 0)$ on the slab. The point co-group $\mathrm{k} = (\mathrm{k}_\parallel, k_z)$ in the bulk does not contain operations that exchange k_z with $-k_z$ but it might be larger than the point co-group of $\mathrm{k} = (\mathrm{k}_\parallel, 0)$ on the surface because it might contain symmetries of the bulk that require fractional translations perpendicular to the surface that are not symmetries neither of the slab nor of the surface.

The point co-group of a given **k**-point $\mathrm{k} = (\mathrm{k}_\parallel, 0)$ on the surface can be found by removing from the corresponding slab point co-group the operations that exchange the two surfaces. It is also the group formed from the intersection of the point co-group of $\mathrm{k} = (\mathrm{k}_\parallel, 0)$ in the slab and of the point $\mathrm{k} = (\mathrm{k}_\parallel, k_z)$ in the bulk. It is the user responsibility to specify the same number of panels for

the PBS and for the slab calculation and to assure that the representations plotted in each panel correspond to each other. Returning to the example of the (111) surface of an fcc, in the direction $\bar{\Gamma} - \bar{K}$ the slab has C_2 symmetry about the x-axis, a symmetry that the surface has not. Therefore you can plot with two different colors the bands that belong to the A or B representations of the slab, (states even or odd with respect to a 180° rotation about the x axis, an operation that exchanges the two surfaces) but you cannot separate the PBS into even or odd states with respect to the C_2 symmetry. You might specify two different panels with the A or B bands in each, but the PBS in the two panels will be the same. On the contrary, for a **k**-point along the $\bar{\Gamma} - \bar{M}$ direction, the point co-group has the C_s symmetry both for the slab and for the surface, so you can separate both the PBS and the surface states in two different panels.

There is no input variable to control or change the colors or style of the plot. To change the defaults you can modify directly the `gnuplot` script, it is written in such a way that a change to a few variables can control the entire plot.

4.8 WHAT='SCF_DOS'

With this option the code makes a self-consistent calculation followed by a band structure calculation on a uniform mesh of **k**-points and computes and plots the electronic density of states.
There is no image parallelization and no advantage to use several images.
This option is controlled by the following variables:

```
deltae         : energy interval for electron dos plot (in Ry).
                 Default: real 0.01 Ry.
ndose          : number of energy points in the dos plot.
                 Default: determined from previous data
nk1_d, nk2_d, nk3_d : thick mesh for dos calculation.
                 Default: integer 16, 16, 16
k1_d, k2_d, k3_d : the shift of the k point mesh
                 Default: integer 1, 1, 1
sigmae         : the smearing used for dos calculation (in eV).
                 If 0.0 uses the degauss of the electronic
                 structure calculation in metals and 0.01 Ry
                 in insulators.
                 Default: real 0.0
legauss        : When .TRUE. computes the electronic dos using
                 a gaussian smearing. When .false. uses the same
                 smearing of the electronic structure calculation
                 in metals or gaussian smearing in insulators.
                 Default: logical .false.
fleldos        : name of the file that contains the electron dos
                 data
                 Default: character output_eldos.dat
flpseldos      : name of the postscript file that contains the
```

```
                 electron dos picture
                 Default: character output_eldos
fleltherm      : name of the file that contains the electron
                 thermodynamic data
                 Default: character output_eltherm.dat
flpseltherm    : name of the postscript file that contains the
                 picture of the electron thermodynamic quantities
                 Default: character 'output_eltherm'
```

The minimum and maximum energy, as well as the number of bands, are specified as with the option `what='scf_bands'`. However with the present option no energy shift is applied to the bands and the minimum and maximum energies refer to the unshifted eigenvalues. Note that after a calculation with `what='scf_dos'` you can run the tool code `epsilon_tpw.x` to evaluate the frequency dependent dielectric constant (for insulators only).
With this option, in the metallic case, the code computes the electronic thermodynamic quantities of a gas of independent electrons whose energy levels give the calculated density of states and produces a postscript file with the electronic excitation energy, free energy, entropy, and constant strain heat capacity as a function of temperature. The zero of the electronic energy is the energy at the smallest temperature required in input when it is lower than 4 K or 4 K.

Number of tasks for this option: 1.

4.9 WHAT='PLOT_BZ'

With this option the code writes a script to make a plot of the Brillouin zone (BZ) and of the path (the default one or the one given in input). The script must be read by the `asymptote` code, available at `http://asymptote.sourceforge.net/`. In many Linux distributions this code is available as a separate package, but it is not installed by default.
The following variables control the plot:

```
lasymptote   : if .TRUE. asymptote is called from within the pro-
               gram and the pdf file with a plot of the BZ is
               produced
               Default: logical .FALSE.
flasy        : initial part of the name of the file where the
               asymptote script is written and of the name of the
               pdf file.
               Default: character(len=*) 'asy_tmp'
asymptote_command  : the command that invokes asymptote and
               produces the pdf file of the BZ.
               Default: character(len=*) 'asy -f pdf -noprc
               flasy.asy'
npx          : used only in the monoclinic cell, this parameter
               is needed to determine the shape of the Brillouin
```

```
              zone. The default value is usually large enough,
              but for particular shapes of the monoclinic
              Brillouin zone it could be small. If the code
              stops with an error asking to increase npx, double
              it until the error disappears.
              Default: integer 8
```

Starting from version 1.5.0 thermo_pw writes also a script to plot the BZ using the freeCAD software. In order to plot the BZ you need to download the code (from the site https://www.freecadweb.org/, version 0.18 or higher) and, after opening it, choose Macro/Macros to execute the freeCAD macro that you find in the working directory (the default name is tpw_freecad.FCMacro). In the Doc directory there is a file called tpw_bz.svg that contains the template for the PDF file with the BZ plot. This file should be copied in the directory where you run the freeCAD executable before running the script. In the same directory there is a file called brillouin_view.cam in the Doc directory that can be used with the option View/Freeze Display to obtain the standard view of the Brillouin zone. This feature is still experimental.
The following variables control the plot:

```
fcfact      : factor used to convert from the 2\pi/a units used
              for the Brillouin zone plot to the millimeters
              units needed to make a plot in freecad.
              Default: real 100.0
fc_red,
fc_green,
fc_blue     : color of the BZ in the rgb format (between 0.0
              and 1.0). By default the BZ will be yellow.
              Default: real 1.0, 1.0, 0.0
fc_transparency : transparency of the BZ between 0 (opaque)
              and 100 (transparent).
              Default: integer 15.
freecadfile : name of the file that contains the freecad
              script to plot the Brillouin zone.
              Default: character(len=*) 'tpw_freecad'
```

The structure of the solid can be seen using the XCrySDen code that can read the input file of pw.x. You can find the code at http://www.xcrysden.org/. thermo_pw produces also a file in the xsf format called prefix.xsf, where the variable prefix is given in the input of pw.x. This can be useful when the nonequivalent atomic positions and the space group are given in the input of pw.x. To see an xsf file, give the command xcrysden -xsf file.xsf.

With this option the code produces also a file with the X-ray powder diffraction intensities for the solid. A plot shows the scattering angles and the relative intensity of each peak. Note that this plot is made using a superposition of atomic charges, not the self-consistent charge. By setting the flag lformf=.TRUE. the atomic form factors of all the atomic types used to calculate the intensities are plotted. By setting the flag lxrdp=.TRUE. the intensities plot is done also after the cell optimization and after a self-consistent

calculation for the options that support it. The variables that control these plots are:

```
lambda      : The X-ray wavelength (in A) used to calculate the
              scattering angles.
              Default: Cu alpha line 1.541838 A if lambda_element
              is empty
lambda_elem : The anode element, used to set the X-ray
              wavelength. Supported elements 'Cr', 'Fe', 'Co',
              'Cu', 'Mo'. NB: lambda must be zero to use
              lambda_elem, otherwise the value of lambda given
              in input is used.
              Default: character(len=2) ' '
flxrdp      : name of the file where the scattering angles and
              intensities are written.
              Default: character 'output_xrdp.dat'
flpsxrdp    : name of the postscript file with the X-ray
              diffraction spectrum.
              Defaults: character 'output_xrdp'
lxrdp       : if .TRUE. compute the xrdp also after the cell
              optimization with all the options mur_lc_...
              with the uniformly strained atomic positions
              and after the scf calculation if supported by
              the option.
              Default: logical .FALSE.
lformf      : if .TRUE. plot also the form factor of each
              atom type present in the solid. Note that the
              atom type is recognized from the atom name in
              the thermo_pw input. The name must coincide
              with the symbols in the periodic table. (Cu, H,
              Li, Li1, ... are correct, CU, LI, H1 ...  are
              wrong).
              Default: logical .FALSE.
smin        : minimum value of s used in the atomic form
              factor plot.
              Default: real 0.0
smax        : maximum value of s used in the atomic form
              factor plot.
              Default: real 1.0
nspoint     : number of points in which the atomic form
              factor is calculated.
              Default: integer 200
lcm         : when .TRUE. the code uses the Cromer-Mann
              coefficients form the International Tables of
              Crystallography to compute the atomic form
              factors, otherwise uses the Doyle-Turner or
              Smith-Burge parameters.
```

```
                Default: logical .FALSE.
flformf      : name of the file in which the atomic form
                factor is written. The code adds a number to
                each file name and creates a file per atom type.
                Defaults: character 'output_formf.dat'
flpsformf    : name of the postscript file with the atomic
                form factor. The code adds a number to each
                file name and creates a file per atom type.
                Defaults: character 'output_formf'
```

4.10 WHAT='SCF_PH'

With this option the code makes a self-consistent calculation followed by a phonon calculation. The phonon calculation is controlled by the file `ph_control` and can be at a single **q** point or on a mesh of **q** points. The different representations are calculated in parallel when several images are available. No other input variable is necessary. The outputs of this calculation are the dynamical matrices files.
`thermo_pw` adds to the `ph.x` code the ability to compute the complex dielectric constant tensor of insulators as a function of a complex frequency for the study of optical properties within time-dependent density functional perturbation theory (TD-DFPT). The code produces also the complex index of refraction for all systems except monoclinic and triclinic. For cubic solids it makes also a plot of the reflectivity for normal incidence and of the adsorption coefficient. As a default the TD-DFPT algorithm uses the Sternheimer equation and a self-consistent loop, but it is also possible to use a Lanczos chain. The option is activated in the `ph.x` input by setting `epsil=.TRUE.` and `fpol=.TRUE.`, but at variance with the `ph.x` code, the frequencies must be specified as complex numbers. The following additional variables can be put in the input of the `ph.x` code, to select the frequency range and the number of frequencies to compute:

```
freq_line : if this variable is .TRUE., after the FREQUENCY
            keyword the code expects the number of frequency
            points and the starting and final frequencies.
            If .FALSE. the number of frequencies and a list of
            frequencies are given. The frequencies are complex
            numbers and are given with a real and an imaginary
            part (in Ry), without parenthesis.
            Default: .FALSE.
delta_freq : When freq_line is .TRUE. instead of giving the
            last frequency of the line one can give the distance
            between two frequency points delta_freq as a complex
            number. The last point of the line is calculated using
            the number of frequencies nfs and the first frequency.
            When delta_freq is not zero the last frequency is not
            used and can be omitted.
```

Default: complex, (0.0, 0.0).
start_freq : Number of the initial frequency calculated in the job in the sequence of frequencies.
Default: integer 1
last_freq : Number of the final frequency calculated in the job in the sequence of frequencies.
Default: integer nfs (total number of frequencies)
lfreq_ev : If .TRUE. the units of the frequencies are eV instead of the default Ry units.
Default: logical .FALSE.
linear_im_freq: This option is used only when freq_line=.TRUE. When linear_freq_im is .TRUE., the imaginary part of each frequency is calculated as eta * freq where eta is the imaginary part of the first frequency on the frequency line.
Default: logical .FALSE.
llanczos: When this flag is .TRUE. at finite frequencies a Lanczos algorithm is used to solve the linear system. Can be very fast but might require much more memory than the standard algorithm. Presently it is incompatible with images.
Default : .FALSE.
lanczos_steps: steps of the Lanczos chain.
Default : interger 2000
lanczos_steps_ext: steps of the extrapolated Lanczos chain
Default: integer 10000
lanczos_restart_steps: number of steps between saving of the Lanczos status. If 0 the status is saved only at the end of the run. Use recover=.TRUE. to resume an interrupted Lanczos chain or to increase the number of steps.
Default: integer 0
extrapolation : extrapolation type. Presently only 'no' or 'average' are available. In the first case no extrapolation is applied, in the second the average of the beta and gamma is used.
Default: character 'average'
pseudo_hermitian : when .TRUE. a pseudo-hermitian algorithm is used to make the Lanczos steps. Should be twice faster than the default non hermitian algorithm.
Default: .TRUE.
only_spectrum : Computes only the spectrum assuming that the Lanczos chain coefficients are in a file. It gives error if the number of requested Lanczos steps is larger than those available on file.
Default: logical .FALSE.
lcg: When this flag is .TRUE. a global conjugate

```
          gradient algorithm is used to compute the
          dielectric constant and the phonon frequencies.
          It will not require mixing, but will use more
          memory than the standard algorithm (for insulators
          only). It is not available for the frequency
          dependent case.
          Default : .FALSE.
```

When in the input of the phonon code a non zero wave-vector **q** is specified, the previous options produce the inverse of the dielectric constant as a function of the frequency at the wave-vector **q** (this option can be used both for insulators and metals).
Additional variables can be specified in the `thermo_pw` input to control where the frequency dependent dielectric constant is written and plotted and how the images divide the work:

```
flepsilon : beginning of the name of the file where the
          frequency dependent dielectric constant is
          written at finite q (the code adds the
          extensions _re and _im)
          Default: character(len=*) 'epsilon'
flpsepsilon : name of the postscript file where the frequency
          dependent dielectric constant is plotted at finite q.
          Default: character(len=*) 'output_epsilon'
floptical : beginning of the name of the file where the
          frequency dependent dielectric constant and the
          complex index of refraction are written (the code
          adds the extensions _xx, _yy, and _zz for the solids
          that need to distinguish the different directions.
          Default: character(len=*) 'optical'
flpsoptical : name of the postscript file where the frequency
          dependent dielectric constant, the complex index of
          refraction and for cubic solid also the reflectivity
          and the absorption coefficient are plotted.
          Default: character(len=*) 'output_optical'
omega_group : number of frequencies calculated together by
          each image. This variables is used only with images.
          Default: integer 1.
```

An example for this option can be found in `example03`, `example16`, `example17`, `example20`, and `example21`.
Number of tasks for this option: for a phonon calculation the number of parallelizable tasks of the phonon code (smaller but of the order of the number of **q** points times $3N_{at}$, where N_{at} is the number of atoms in the unit cell), for a dielectric constant calculation using Sternheimer equation `nfs/omega_group`, number of frequencies divided by the number of frequencies in each group, for a dielectric constant calculation using Lanczos 1 (images not allowed).

It is also possible to separate the self-consistent and the phonon calculation, by running first `thermo_pw.x` using `what='scf'` and then running, on the same directory, `thermo_pw.x` using `what='scf_ph'`. The same input can be used in the two calculations, only the `thermo_control` file need to be changed. The number or processors/pools/images can be changed in the same cases in which this is possible in Quantum ESPRESSO.

Using images in a phonon calculation with the master/slave approach has an overhead because each image must recalculate the initialization and the band structure at each task, or check if the bands are already on disk, calculated previously by the same image. On some systems with slow disks it could be faster to recalculate the bands instead of reading them from disk. It is also possible to use the image breaking suggested by the `ph.x` code that keeps, as much as possible, on the same image the tasks that require the same initialization without recomputing it. The input variables that control this part of the calculation are:

```
force_band_calculation : if .TRUE. the bands are never read
              from disk but recalculated when needed.
              Default: logical .FALSE.
use_ph_images : if .TRUE. each image makes a set of tasks so
              as to minimize the number of band calculations
              and phonon initialization.
              Default: logical .FALSE. if nimage>1 .TRUE.
              nimage=1.
sym_for_diago : When .TRUE. use symmetry to calculate the
              bands and the unperturbed wavefunctions instead
              of diagonalizing the Hamiltonian.
              Default: logical .FALSE.
```

4.11 WHAT='SCF_DISP'

With this option the code makes a self-consistent calculation followed by a phonon dispersion calculation at a fixed geometry. The geometry is given in the input of `pw.x`. The dynamical matrices are used to calculate the interatomic force constants. Using these interatomic force constants the code calculates the dynamical matrices and hence the phonon frequencies along a path in the Brillouin zone and on a much thicker mesh of **q** points. The path can be generated automatically or given in input as in a band structure calculation (see above `what='scf_bands'`). The uniform mesh of **q** points is specified in `thermo_control`. The code uses the phonon frequencies calculated on the thick mesh of **q** points to get the phonon density of states using a smearing approach. The density of states is used to calculate the harmonic thermodynamic properties: vibrational energy, vibrational free energy, vibrational entropy, and constant strain heat capacity. The same thermodynamic quantities are calculated also by direct integration over the Brillouin zone and compared in the plots. When `with_eigen=.TRUE.` the atomic B-factors are calculated as a function of temperature by the generalized vibrational density

of states or by a direct integration over the Brillouin zone. Note that presently no interpolation formula is used at low temperatures so `thermo_pw` can not be used to obtain thermodynamic properties at very low temperatures. The plotted numerical values of the extensive quantities refer to an Avogadro number of unit cells. If you need them per mole you have to divide by the number of formula units in a unit cell.
The input variables that control this option are:

```
freqmin_input : minimum frequency for phonon dos plot.
                Default: real determined from phonon frequencies
freqmax_input : maximum frequency for phonon dos plot.
                Default: real determined from phonon frequencies
deltafreq     : frequency interval for phonon dos plot.
                Default: real 1 cm^{-1}
ndos_input    : number of frequency points in the dos plot.
                Default: determined from previous data
nq1_d, nq2_d, nq3_d : thick mesh for phonon dos calculation.
                Default: integer 192, 192, 192
phdos_sigma   : the smearing used for phonon dos calculation
                (in cm^-1).
                Default: real 2. cm^-1
after_disp    : if .TRUE. the dynamical matrices are supposed
                to be already available in files in the current
                directory. This option is needed to restart when
                the outdir directory has been erased and ph.x
                cannot be run without redoing the scf calculation.
                The exact restart point depends on the files
                already available on the current directory.
                Default: logical .FALSE.
fildyn        : the name of the dynamical matrix file, as
                would be specified in the input of ph. To be used
                when after_disp is .TRUE..
                Default: character ' '
zasr          : type of acoustic sum rule applied to the ifc.
                Default: character(len=*) 'Simple'
ltherm_dos    : if .TRUE. the thermal properties are calculated
                from the phonon dos.
                Default: logical .TRUE.
ltherm_freq   : if .TRUE. the thermal properties are calculated
                from the direct integration using the phonon
                frequencies.
                Default: logical .TRUE.
flfrc         : file where the interatomic force constants are
                written.
                Default: character(len=*) 'output_frc.dat.g1'
flfrq         : file where matdyn writes the interpolated
                frequencies.
```

```
                Default: character(len=*) 'output_frq.dat.g1'
flvec         : file where the eigenvectors of the dynamical
                matrix are written.
                Default: character(len=*) 'matdyn.modes'
fldosfrq      : file where the frequencies used to calculate
                the phonon density of states are saved.
                Default: character(len=*) 'save_frequencies.dat'
fldos         : file where the phonon dos is written.
                Default: character(len=*) 'output_dos.dat.g1'
fltherm       : file where the harmonic thermodynamic.
                quantities are written
                Default: character(len=*) 'output_therm.dat.g1'
flpsdisp      : postscript file of the phonon dispersions.
                Default: character(len=*) 'output_disp'
flpsdos       : postscript file of the phonon dos.
                Default: character(len=*) 'output_dos'
flpstherm     : postscript file of the harmonic thermodynamic
                quantities.
                Default: character(len=*) 'output_therm'
```

This option requires `ldisp=.TRUE.` in the phonon input.
An example for this option can be found in `example04`.
Number of tasks for this option: number of parallelizable tasks of the phonon code (smaller but of the order of number of **q** points times $3N_{at}$, where N_{at} is the number of atoms in the unit cell).

4.12 WHAT='SCF_ELASTIC_CONSTANTS'

With this option the code calculates the elastic constants of the solid at the geometry given as input to `pw.x`. There are four different algorithms that at convergence should give the same results. In two of them, depending on the Laue class, the code calculates the nonzero components of the stress tensor for a set of strains and obtains the elastic constants from the numerical first derivative of the stress with respect to strain. The two algorithms `standard` and `advanced` differ only in the choice of the unit cell. In the `standard` method the code applies the strain to the primitive vectors of the unstrained solid and uses `ibrav=0` and the strained vectors to compute the stress tensor. The `advanced` method, available only for selected Bravais lattices, try to optimize the calculation by choosing strains for which the number of needed **k**-points is reduced. Moreover it identifies the Bravais lattice of the strained solid and recalculates the primitive vectors with the conventions of QUANTUM ESPRESSO. When available this should be the most efficient method. The other two algorithms are called `energy_std` and `energy`. Using the `energy_std` or `energy` algorithm the elastic constants are calculated from a polynomial fit of the total energy as a function of strain without computing stress. This option usually requires more independent strains. It can be used when stress calculation is not implemented in QUANTUM ESPRESSO. The difference between

energy_std and energy is the same between the algorithms that use the stress. With energy_std the code applies the strain using ibrav=0, while with energy an optimized cell is used. As the advanced algorithm the energy algorithm is available only for selected Bravais lattices.
For all methods the number of strains is ngeo_strain for each independent strain. For each strain, the code relaxes the ions to their equilibrium positions if frozen_ions=.FALSE. or keeps them in the strained positions if frozen_ions=.TRUE.. Note that elastic constant calculations with frozen_ions=.FALSE. might require smaller force convergence threshold than standard calculations. The default value of forc_conv_thr must be changed in the pw.x input. At finite pressure all methods give the elastic constants that relate linearly stress and strain.
The input variables that control this option are:

```
frozen_ions: if .TRUE. the elastic constants are calculated
             keeping the ions frozen in the strained positions.
             Default: logical .FALSE.
ngeo_strain: the number of strained configurations used to
             calculate each derivative.
             Default: integer 4 ('standard' and 'advanced'),
             6 ('energy')
elastic_algorithm: 'standard', 'advanced', 'energy_std' or
             'energy'. See discussion above.
             Default: character 'standard'
delta_epsilon: the interval of strain values between two
             geometries. To avoid a zero strain geometry
             that might have a different symmetry ngeo_strain
             must be even.
             Default: real 0.005
epsilon_0:   a minimum strain. For small strains the ionic
             relaxation routine requires a very small threshold
             to give the correct internal relaxations and
             sometimes fail to converge. In this case you
             can increase delta_epsilon, but if delta_epsilon
             becomes too large you can reach the nonlinear
             regime. In this case you can use a small
             delta_epsilon and a minimum strain. (To be used
             only for difficult systems).
             Default: real 0.0
poly_degree: degree of the polynomial used to interpolate
             stress or energy. ngeo_strain must be larger
             than poly_degree+1
             Default: 3 ('standard', 'advanced', 2
             if ngeo_strain < 6), 4 ('energy', 3 if
             ngeo_strain < 6).
fl_el_cons:  the name of the file that contains the elastic
             constants
```

Default: character(len=*) 'output_el_con.dat'

The three algorithms are equivalent only at convergence both with **k**-point sampling and with the kinetic-energy cut-off, but large differences between the elastic constants obtained with the standard and advanced algorithms might point to insufficient **k**-point sampling. Large differences between the elastic constants obtained with the energy algorithm with respect to the other two might point to insufficient kinetic-energy cut-off.
Number of tasks for this option: ngeo_strain times the number of independent strains.

Using the elastic constants tensor the code can calculate and print a few auxiliary quantities: the bulk modulus, the poly-crystalline averages of the Young modulus, of the shear modulus, and of the Poisson ratio. Both the Voigt and the Reuss averages are printed together with the Hill average. The Voigt-Reuss-Hill average of the shear modulus and of the bulk modulus are used to compute average sound velocities. The average of the Poisson ratio and the bulk modulus allow the estimation of the Debye temperature. The Debye temperature is calculated also with the exact formula evaluating the average sound velocity from the angular average of the sound velocities calculated for each propagation direction solving the Christoffel wave equation. The exact Debye temperature is used within the Debye model to calculate the Debye's vibrational energy, free energy, entropy, and constant strain heat capacity. These quantities are plotted in a postscript file as a function of temperature.

4.13 WHAT='MUR_LC'

With this option the code runs several self-consistent calculations at different geometries. The runs can be done in parallel when several images are available. This option has two working modes controlled by the logical variable lmurn. When lmurn=.TRUE. the total energy as a function of the volume is interpolated by a Murnaghan equation and a plot of the energy as a function of the volume and of the pressure as a function of the volume is produced. The volume is changed by changing only celldm(1). celldm(2)...celldm(6) remain fixed at the values given as input of pw.x. When lmurn=.FALSE. the energy is calculated in a uniform grid of parameters composed of ngeo(1) $\times$ ngeo(2)... $\times$ ngeo(6) points. The energies are fitted with a quadratic or quartic polynomial of N_k variables, where N_k is the number of independent crystal parameters for the given crystal system. A plot of the energy as a function of the lattice constant is produced for cubic systems. For solids of the hexagonal, tetragonal, and trigonal systems contour plots of the energy as a function of the two crystal parameters (a and c/a or a and $\cos\alpha$) are plotted. For orthorhombic systems contour plots of the energy as a function of a and b/a are plotted for each value of c/a. Presently no graphical tool is implemented to plot the energy for monoclinic and triclinic crystal systems. When lmurn=.FALSE. the bulk modulus is not calculated. To obtain it, you can calculate the elastic constants at the minimum geometry (see the

option what='mur_lc_elastic_constants'). With this option the pressure control is active. You can specify a finite pressure and the enthalpy is minimized instead of the energy. Note however that if the minimum is distant from the starting configuration its associated error can be large, larger for the quadratic than for the Murnaghan interpolation. For this reason the present option should be used starting from the minimum found by pw.x using the vc-relax option and the pressure should not be too different from the pressure used for vc-relax. Note that with this option the atomic coordinates are relaxed at each geometry even if you specified calculation='scf' in the pw.x input. Use frozen_ion=.TRUE. if you want to keep them fixed. To increase the maximum number of ionic iterations use calculation='relax' and give nstep (otherwise the default is 20). Only the bfgs relaxation is supported by this option. When lel_free_energy=.TRUE. the code makes also an electronic bands dos calculation at each geometry, computes the electronic thermodynamic quantities as a function of temperature and writes them in separate files. These files can be used to add the electronic contribution to the anharmonic properties with the option mur_lc_t.
This option can be controlled by the following variables:

```
ngeo(1),...,ngeo(6) : the number of geometries to use for each
             celldm parameter. The lattice constant of these
             geometries is calculated from the input of pw.x.
             celldm(1),...,celldm(6) of this input is used
             for the central geometry. For the others celldm(1),
             ...,celldm(6), are changed in steps of step_ngeo(1),
             ...,step_ngeo(6). ngeo(1) must be odd. Only the
             values of celldm relevant for each Bravais lattice
             are actually changed.
             Default: integer 1,1,1,1,1,1 for what=scf_*,
             9,1,1,1,1,1 for what=mur_lc_* and lmurn=.TRUE. or
             for cubic systems, 5 on all the relevant celldm
             parameters when lmurn=.FALSE. and the system
             is not cubic.
step_ngeo(1),...,step_ngeo(6) : The step between the lattice
             constants at different geometries. step_ngeo(1) is,
             in atomic units, the change of a, step_ngeo(2),
             step_ngeo(3) are dimensionless and are the changes
             of the ratios b/a, c/a, step_ngeo(4), step_ngeo(5), &
             step_ngeo(6) are the changes in degree of the
             angles alpha, beta, and gamma. The cosine of the
             angle is calculated by the program.
             Default: real 0.05 a.u., 0.02, 0.02, 0.5, 0.5, 0.5
lmurn       : if .TRUE. the Murnaghan fit is done. Only ngeo(1)
             values of the energy are fitted, the other values
             of ngeo are not used. if .FALSE. use a quadratic
             or quartic function to interpolate the energy as
             a function of all celldm parameters. The number of
```

```
              self-consistent calculations is ngeo(1) x ngeo(2)
              x ngeo(3) x ngeo(4) x ngeo(5) x ngeo(6). In this
              case only the minimum energy and the optimal celldm
              are given in output.
              Default: .TRUE.
show_fit    : if .TRUE. show the contour plot of the fitted
              energy instead of the energy. Used by default
              when reduced_grid is .TRUE.
              Default: logical .FALSE.
frozen_ions: if .TRUE. the atomic coordinates are obtained by
              straining the coordinates given in the pw.x input
              to the new cell parameters (equivalent to keep the
              crystal coordinates fixed) and kept fixed. If .FALSE.
              the atomic coordinates are relaxed at each geometry.
              Default: logical .FALSE.
vmin_input : minimum volume for the plot of the energy as a
              function of volume.
              Default: real 0.98 times the volume of the first
              geometry.
vmax_input : maximum volume for the plot of the energy as a
              function of volume.
              Default: real 1.02 times the volume of the last
              geometry.
deltav      : distance between two volumes in the plot of the
              energy as a function of the volume.
              Default: real calculated from nvol.
nvol        : number of volumes in Murnaghan plot.
              Default : integer 51
lquartic    : if .TRUE. fit the energy with a quartic polynomial.
              Default : logical .TRUE.
lsolve      : choose the algorithm used to fit the quartic
              polynomial parameters.
              Allowed values:
              1 explicitly minimize chi^2 (usually less accurate
              than the other two. Should be used only for tests).
              2 Use the QR algorithm to minimize chi^2 (lapack
              routine dgels) 3 Use the SVD algorithm to minimize
              chi^2 (lapack routine dgelss).
              Default: integer 2
flevdat     : file where the Murnaghan equation is written. The
              results of the Murnaghan fit are then written in
              flevdat.ev.out.
              Default: character(len=*) 'output_ev.dat'
flpsmur     : postscript file of the Murnaghan plot.
              Default: character(len=*) 'output_mur'
lel_free_energy : if .TRUE. computes the electronic thermodynamic
              properties (energy, free energy, entropy, and constant
```

```
              strain heat capacity) at each temperature and plots
              them. See the scf_dos option for the parameters that
              control the calculation.
              Default: .FALSE.
ncontours   : the number of contours in the energy plot. These
              levels can be determined automatically by the code
              or defined by the user. The energy levels can be
              defined after the INPUT_THERMO namelist but before
              the path, as a list:
              energy_level(1)      color(1)
              ...
              energy_level(ncontours)   color(ncontours)
              Color is a string of the type color_red, color_green,
              etc.
              The list of available colors is at the beginning of
              each gnuplot script. energy_level is in Ry units.
              Default: integer 9
do_scf_relax : if .TRUE. the code makes a self-consistent relax
              calculation at the equilibrium geometry to find
              the optimized atomic coordinates. This step is
              needed only for solids that have internal degrees
              of freedom in the unstrained configuration.
              If .FALSE. the coordinates of the input geometry
              are strained uniformly to the equilibrium geometry.
              Default: logical .FALSE.
flenergy    : name of the file that contains the energy in a
              form that can be used by gnuplot to make contour
              plots.
              Defaults: character(len=*) 'output_energy'
flpsenergy  : file with the contour plots of the energy as a
              function of the crystal parameters.
              Default: character(len=*) 'output_energy'
```

An example for this option can be found in `example05`.
Number of tasks for this option:
`ngeo(1)` when `lmurn=.TRUE.`,
`ngeo(1)×ngeo(2)×ngeo(3)×ngeo(4)×ngeo(5)×ngeo(6)` when `lmurn=.FALSE.`.

4.14 WHAT='MUR_LC_BANDS'

With this option the code computes the band structure at the geometry that minimizes the energy. See `what='scf_bands'` and `what='mur_lc'` for a list of the variables that control these two options. With this option the pressure control is active. You can specify a finite pressure and the enthalpy is minimized instead of the energy to find the equilibrium geometry. The bands are calculated at the geometry that corresponds to the external pressure. An example for this option can be found in `example06`.

Number of tasks for this option: see what='mur_lc' and what='scf_bands'.

4.15 WHAT='MUR_LC_DOS'

With this option the code computes the electronic dos at the geometry that minimizes the energy. See what='scf_dos' and what='mur_lc' for a list of the variables that control these two options. With this option the pressure control is active. You can specify a finite pressure and the enthalpy is minimized instead of the energy to find the equilibrium geometry. The dos is calculated at the geometry that corresponds to the external pressure.
Number of tasks for this option: see what='mur_lc' and what='scf_dos'.

4.16 WHAT='MUR_LC_PH'

This option is similar to what='scf_ph' but the phonon calculation is made at the geometry that minimizes the energy. See what='scf_ph' and what='mur_lc' for a list of the variables that control these two options. With this option the pressure control is active. You can specify a finite pressure and the enthalpy is minimized instead of the energy to find the equilibrium geometry. The phonons are calculated at the geometry that corresponds to the external pressure. An example for this option can be found in example07.
Number of tasks for this option: Maximum between the number of tasks needed by the what='mur_lc' option and the number of tasks of the phonon code (see above the option what='scf_ph').

4.17 WHAT='MUR_LC_DISP'

This option is similar to what='scf_disp' but the phonon calculation is made at the geometry that minimizes the energy. See what='scf_disp' and what='mur_lc' for a list of the variables that control these two options. With this option the pressure control is active. You can specify a finite pressure and the enthalpy is minimized instead of the energy to find the equilibrium geometry. The phonons are calculated at the geometry that corresponds to the external pressure. An example for this option can be found in example08.
Number of tasks for this option: Maximum between the number of tasks needed by the what='mur_lc' option and the number of tasks of the phonon code (see above the option what='scf_ph').

4.18 WHAT='MUR_LC_ELASTIC_CONSTANTS'

As what='scf_elastic_constants' but the calculation is made at the geometry that minimizes the energy. The energy minimization is done as described for what='mur_lc'. With this option the pressure control is active. You can specify a finite pressure and the enthalpy is minimized instead of the energy to find the equilibrium structure. Note however that if the minimum is distant from the starting configuration its associated error can be large, larger for the

quadratic than for the Murnaghan interpolation. An example for this option can be found in example13.
Number of tasks for this option: Maximum between the number of tasks needed by the what='mur_lc' option and the number of tasks needed for the what='scf_elastic_constants' option.

4.19 WHAT='MUR_LC_T'

With this option the code calculates the anharmonic properties within the quasi-harmonic approximation. The outputs of the code are the values of crystal parameters (celldm) as a function of temperature. This calculation is done by computing the phonon dispersions on all the geometries specified as in what='mur_lc' (or on a subset of these geometries) and minimizing the Helmholtz free energy. Separate plots of the phonon dispersions are obtained for all the calculated geometries. For each geometry the code produces also plots of the phonon density of states and of the harmonic thermodynamic quantities. From celldm as a function of temperature the code computes the thermal expansion tensor, the volume, and the volume thermal expansion as a function of temperature. The frequencies at all the calculated geometries are interpolated by quadratic or quartic polynomials of the crystal parameters and can be shown at crystal parameters given in input or at those that minimize the free energy at a temperature given in input. The interpolated frequencies are shown on the same path used for the phonon dispersions. In addition to the frequencies the code produces also several plots of the derivatives of the frequencies with respect to the crystal parameters multiplied by the crystal parameters.
When the Murnaghan equation is used to interpolate the Helmholtz free energy (lmurn=.TRUE.), in addition to the volume, the bulk modulus and the pressure derivative of the bulk modulus are plotted as a function of temperature. Moreover the isobaric heat capacity, the isoentropic bulk modulus, and the average Grüneisen parameter are calculated as a function of temperature. The mode Grüneisen parameters are calculated with cubic interpolations of the phonon frequencies. Using the variable with_eigen one can calculate these parameters as derivatives of the phonon frequencies (default) or as expectation values of the derivatives of the dynamical matrix on the central geometry eigenvectors (might require a lot of RAM). The mode Grüneisen parameters are used to calculate the volume thermal expansion and the result is compared with the volume thermal expansion derived from the numerical derivative of the equilibrium volume obtained from the minimization of the Helmholtz free energy. When the Helmholtz free energy is interpolated with a quadratic or cubic polynomial (lmurn=.FALSE.), by default, the code computes only the temperature dependence of the lattice parameters and of the volume, the volume thermal expansion, and the thermal expansion tensor. However if a file with the elastic constants is found in the elastic_constants directory and lb0_t=.FALSE. the bulk modulus is calculated and assumed independent from the temperature so that also the isobaric specific heat, the isoentropic bulk modulus, and the average Grüneisen

parameter are calculated as a function of temperature. The derivatives of the frequencies with respect to the crystal parameters are used to calculate the thermal expansion tensor which is compared with that obtained from the numerical derivatives of the crystal parameters obtained from the minimization of the Helmholtz free energy. If many files with the elastic constants, one for each geometry, as produced with the option `elastic_constants_t`, are found in the `elastic_constants` directory and `lb0_t=.TRUE.`, the bulk modulus and the elastic constants are computed as a function of temperature within the “quasi-static approximation" and are used to calculate the other thermodynamic properties. If one or many elastic constants files are found in the `anhar_files` directory and `lb0_t=.TRUE.` the bulk modulus and elastic constants are computed as a function of temperature within the “quasi-harmonic approximation" and are used to calculate the other thermodynamic properties (see a more detailed discussion in the option `what=elastic_constants_t`).
In addition to the quantities plotted for cubic solids, the code plots also the elastic constants and the bulk modulus as a function of the temperature and the elastic compliances and the compressibility as a function of temperature. The elastic constants are interpolated with a quadratic (`lquartic=.FALSE.`) or quartic (`lquartic=.TRUE.`) polynomial of the crystal parameters. Moreover the thermal stresses and the generalized average Grüneisen parameters are plotted. These possibilities are implemented only for cubic, tetragonal, hexagonal, trigonal, and orthorhombic systems.
With this option the pressure control is active. You can specify a finite pressure and the Gibbs energy is minimized instead of the Helmholtz free energy. Note however that if the minimum is distant from the starting configuration its associated error can be large, larger for the quadratic than for the Murnaghan interpolation.
The input variables that control these plots are those described in the option `what='mur_lc'` and `what='mur_lc_disp'` in addition to the following:

```
grunmin_input : minimum y coordinate for the Gruneisen
                parameter plot.
                Default: real, calculated from the Gruneisen
                parameters.
grunmax_input : maximum y coordinate for the Gruneisen
                parameter plot.
                Default: real, calculated from the Gruneisen
                parameters.
volume_ph     : The frequencies and Gruneisen parameters inter-
                polated at this volume are plotted on a postscript
                file. When volume_ph=0.0 the volume is calculated
                from temp_ph. This option is available only for
                cubic solids. Otherwise use celldm_ph.
                Default : 0.0 (in (a.u.)**3)
celldm_ph     : The frequencies and Gruneisen parameters
                interpolated at this crystal parameters are
                plotted on a postscript file.
```

```
                If this is 0.0 the celldm are calculated from
                temp_ph. To have accurate Gruneisen parameters
                and interpolated frequencies set the central
                geometry as close as possible to celldm_ph. When
                all nstep are odd, the central geometry is the one
                given in the input of pw.x.
                Default : 0.0 (celldm(1) in a.u., celldm(2-6)
                dimensionless)
temp_ph       : The frequencies and Gruneisen parameters inter-
                polated at the volume (cubic systems) or at
                celldm (anisotropic systems) that minimize the
                energy at this temperature are plotted on a
                postscript file (only when volume_ph=0.0 or
                celldm_ph(1)=0.0).
                Default : real tmin (in K)
with_eigen    : if .TRUE. use the eigenvectors of the
                dynamical matrix to calculate the Gruneisen
                parameters used for anharmonic properties.
                Could require a lot of RAM. Note however that
                eigenvectors are always used to calculate
                the plotted Gruneisen bands (both in cubic
                and anisotropic solids).
                Default: logical .FALSE.
poly_degree_ph : degree of the polynomial used to interpolate
                the vibrational free energy. Presently only the
                values 1, 2, 3, or 4 are available for anisotropic
                solids.
                Default: integer 4
poly_degree_cv : degree of the polynomial used to interpolate
                the heat capacity. Presently only the values
                1, 2, 3, or 4 are available for anisotropic
                solids.
                Default: integer 4
poly_degree_bfact : degree of the polynomial used to interpolate
                the b factor. Presently only the values 1, 2, 3,
                or 4 are available for anisotropic solids.
                Default: integer 4
poly_degree_elc : degree of the polynomial used to interpolate
                the elastic constants. Presently only the values
                1, 2, 3, or 4 are available for anisotropic solids.
                Default: integer 4
lv0_t         : if .TRUE.  the calculation of the thermal
                expansion with Gruneisen parameters uses the
                equilibrium geometry as a function of temperature
                computed from the free energy minimization,
                otherwise the equilibrium geometry at T=0 K.
                If reduced_grid=.TRUE. or both ltherm_freq=.FALSE.
```

and ltherm_dos=.FALSE. the input geometry is used when lv0_t=.FALSE.
Default: logical .TRUE.

lb0_t : if .TRUE. the calculation of the thermal expansion with Gruneisen parameters uses the bulk modulus as a function of temperature computed from the free energy minimization, otherwise the bulk modulus computed at T=0 K (lmurn=.TRUE.).
For lmunrn=.FALSE. the code expects a single elastic constant file when lb0_t=.FALSE. and an elastic constants file for each geometry when lb0_t=.TRUE.. Note that if lb0_t=.FALSE. and there are many elastic constants files the code use a constant bulk modulus computed with the elastic constants found in the file of the central geometry. If lb0_t=.TRUE. and there is a single elastic constants file all the quantities that depend on elastic properties are not computed.
Default: logical .TRUE.

poly_degree_grun : degree of the polynomial used to interpolate the frequencies (Used only when lmurn=.TRUE. otherwise it is 2).
Default: 4

flgrun : file where the Gruneisen parameters are written.
Default: character(len=*) 'output_pgrun.dat'

flpgrun : file where the Gruneisen parameters in a plotable form are written.
Default: character(len=*) 'output_grun.dat'

flpsgrun : name of the postscript file with the Gruneisen parameters plot. The frequencies are written in a file with the same name plus the string _freq.
Default: character(len=*) 'output_grun'

flanhar : file where the anharmonic thermodynamic quantities are written.
Default: character(len=*) 'output_anhar.dat'

flpsanhar : postscript file of the anharmonic quantities.
Default: character(len=*) 'output_anhar'

fact_ngeo(1)...fact_ngeo(6) : With these factors the vibrational free energy is interpolated using a smaller number of geometries with respect to the total energy. The phonons are always calculated at geometry 1, then fact_ngeo(i)-1 geometries are not calculated and so on. The last calculated geometry must be ngeo(i). This happens when fact_ngeo(i) divides ngeo(i)-1. For even ngeo(i), fact_ngeo(i) must be 1. For odd ngeo(i) the following table gives a few examples
ngeo fact_ngeo calculated geometries

```
                3         2            1,3
                5         2            1,3,5
                7         2            1,3,5,7
                7         3            1,4,7
                9         2            1,3,5,7,9
                9         4            1,5,9
                11        2            1,3,5,7,9,11
                11        5            1,6,11
                Defaults: integer 1,1,1,1,1,1
                This option is not active when one of the
                ngeo_ph(i) is different from ngeo(i).
ngeo_ph(1),...,ngeo_ph(6) These variables are set to compute
                the phonon dispersions in a subset of the
                geometries used to compute the total energy.
                All values must be smaller than the corresponding
                ngeo and even or odd as the corresponding ngeo.
                step_ngeo remains the same for the two meshes.
                The following table gives a few examples:
                ngeo  ngeo_ph     phonon calculated in geometries
                5        3            2,3,4
                6        2            3,4
                6        4            2,3,4,5
                7        3            3,4,5
                7        5            2,3,4,5,6
                9        3            4,5,6
                9        5            3,4,5,6,7
                Defaults: integer ngeo(1),...,ngeo(6)
reduced_grid: if .TRUE. the computed geometries are only along
              one dimensional lines. So each parameter is varied
              independently keeping the others fixed at the input
              values. This option sets ltherm_freq=.FALSE.,
              ltherm_dos=.FALSE., lv0_t=.FALSE. and lb0_t=.FALSE..
              With this option the thermal expansion is
              calculated only using the Gruneisen parameters
              at the input geometry. The multidimensional fit
              of the free energy is not done so this method
              should be faster than the default one, but it
              is less precise. This option is used only with
              lmurn=.FALSE. and requires a file with the
              elastic constants at the input geometry.
              Default: logical .FALSE.
all_geometries_together : if .TRUE. all the phonon calculation
              for all the geometries are used for the image
              parallelization. To be used only if you have many
              images (and CPUs) available.
              Default : logical .FALSE.
```

The output files corresponding to different geometries can be identified by the presence of the letters g1, g2, ... in the filename. To exploit all the features of this option please write the dynamical matrices in .xml format (using a fildyn with the .xml extension). An example for this option can be found in example09.
Number of tasks for this option: Maximum between the number of tasks needed by the what='mur_lc' option and the number of tasks of the phonon code (see above the option what='scf_ph').
When all_geometries_together=.TRUE.: number of tasks of the phonon code times the number of geometries.

4.20 WHAT='ELASTIC_CONSTANTS_T'

With this option the code can compute the elastic constants and elastic compliances as a function of temperature using the quasi-harmonic approximation. For the same geometries that are used to compute the elastic constants with the elastic_algorithm='energy_std' or elastic_algorithm='energy', the code can compute the phonon dispersions and compute the elastic constants at each temperature as the second derivatives of the Helmholtz free energy with respect to strain. The second derivatives are corrected so that the stress-strain elastic constants are shown in the plots and in output. The temperature dependent elastic constants are calculated on a regular grid of unperturbed geometries, the same geometries chosen by the option what='mur_lc', and written on separate files, one for each unperturbed geometry, inside the directory anhar_files. In order to plot the elastic constants as a function of temperature within the 'quasi-harmonic' approximation, it is necessary to make another calculation with what='mur_lc_t' having on files the elastic constants calculated for each geometry with the present option. In this case thermo_pw will be able to calculate the anharmonic properties using temperature dependent elastic constants and bulk moduli obtained by interpolating the "fixed-geometry quasi-harmonic" elastic constants computed by this option at the crystal parameters found at each temperature from the minimization of the free energy. The variables fact_ngeo and ngeo_ph are not available with this option. Using start_geometry_qha and last_geometry_qha it is possible to compute the temperature dependent elastic constants for selected or for a single unperturbed configuration. The use of start_geometry and last_geometry is also allowed but it refers to the global number of geometries necessary to compute the elastic constants in all the grid.

Since the calculation of the Helmholtz free energy derivatives is quite heavy, it has to be requested explicitly using the flag use_free_energy=.TRUE..
By default, the code computes only the elastic constants at $T = 0$ as second derivatives of the energy and writes them on files in the directory elastic_constants. A run of thermo_pw using what='mur_lc_t' having on files the $T = 0$ elastic constants contained in the directory elastic_constants allows the calculation of the anharmonic properties using temperature de-

pendent elastic constants and bulk moduli obtained by interpolating (within the “quasi-static approximation") the elastic constants computed by this option at the crystal parameters that, at each temperature, minimize the free energy. When both the $T = 0$ and the temperature dependent elastic constants are on file in the directory `elastic_constants` and `anhar_files` respectively, the latter are used. When both `use_free_energy=.TRUE.` and `lel_free_energy=.TRUE.` the electronic free energy is added to the free energy before computing the elastic constants. In this case the code expects to find on file (in `therm_files`) the electronic thermodynamic properties for each perturbed geometry. When `use_free_energy=.FALSE.` and `lel_free_energy=.TRUE.` the code computes the electronic thermodynamic properties at each perturbed geometry and writes them on file. In this case the electronic free energy is not added to the free energy. Note that the user must be careful to use the same value for the `lel_free_energy` flag with this option and in the following `mur_lc_t` calculation that interpolates the elastic constants. The variables that control this run are:

```
use_free_energy : when .TRUE. computes the elastic constants
              as second derivatives of the Helmholtz free
              energy with respect to strain. When .FALSE.
              the elastic constants are computed as second
              derivatives of the energy or using the
              stress-strain algorithms.
              Default .FALSE.
start_geometry_qha : Among the geometries considered by the
              option mur_lc_t the calculations of elastic
              constants are done starting from this geometry.
              Default, integer 1
last_geometry_qha : Among the geometries considered by the
              option mur_lc_t the calculations of elastic
              constants are done only up to this geometry.
              Default, integer total number of geometries.
```

An example for this option with `use_free_energy=.FALSE.` can be found in `example22` while an example with `use_free_energy=.TRUE.` can be found in `example23`.
Number of tasks for this option: The product of the number of tasks needed by the `what='scf_elastic_constants'` option and the number of geometries used with `mur_lc_t` when `use_free_energy=.FALSE.`. When `use_free_energy=.TRUE.` and `all_geometries_together=.TRUE.` the number of tasks of the previous case is further multiplied by the number of tasks needed to compute a phonon dispersion (see above the option `what='scf_ph'`).
When `use_free_energy=.TRUE.` and `all_geometries_together=.FALSE.` the number of tasks of this option is equal to the number of tasks needed to compute a phonon dispersion.

5 Restarting an interrupted run

There are several situations that might require the restart of the thermo_pw code. We must distinguish two different cases: thermo_pw stopped while running QUANTUM ESPRESSO routines, because the code reached the maximum cpu time or because some external event stopped the run, or thermo_pw stopped after doing some post-processing task. This second case comprises also the normal termination of thermo_pw and the necessity to change some details of the plot rerunning the post-processing tools without redoing the QUANTUM ESPRESSO calculations.

Support for the first case is based on the recover features provided by QUANTUM ESPRESSO routines and usually works when images are not used. This restarting method needs files in the outdir directory. In this case thermo_pw behaves as QUANTUM ESPRESSO except for the fact that max_seconds in the input of pw.x or of ph.x is not active. To run thermo_pw for a fixed number of seconds max_seconds must be set in the THERMO_CONTROL namelist. If the code stopped inside pw.x, restart_mode must be set to 'restart' in the input of pw.x while if the code stopped inside ph.x routines recover must be set to .TRUE. in the input of ph.x.

When running thermo_pw with several images and calculating a phonon dispersion or using the what='mur_lc_t' option, max_seconds is controlled by the image driver of thermo_pw. Presently, after max_seconds a signal is sent to the asynchronous driver and the master stops sending new works to the images. It stops the code when all the images have terminated their current task. Recovering from this point is possible without loosing any previous work by keeping the outdir directory and by setting recover=.TRUE. in the ph.x input. Note however that when thermo_pw is stopped by the operating system in an unclean way this restart method could not work.

As a last resource you can remove the outdir directory, and thermo_pw will not recalculate the quantities contained in files that are already in the working directory. Completed phonon calculations at a given geometry for which the dynamical matrices are available are not redone if you put the fildyn name in the thermo_control namelist. It is possible to stop thermo_pw after the calculation of the phonon dispersions for a fixed number of geometries by setting the input variable max_geometries in the THERMO_CONTROL namelist or also specify exactly which geometries to do in a given run using the variables start_geometry and last_geometry or start_geometry_qha and last_geometry_qha.

In general the restart of thermo_pw from a post-processing task is much easier. Each routine checks if a file with the same name as the file that it would produce is already in the working directory, and if this happens, it reads its content and returns. This feature cannot be disabled from input. In order to recalculate a given quantity, just remove the file that contains it from the working directory. For instance in an anharmonic calculation, if you have already all the dynamical matrices for all the geometries and you do not have any more the outdir directory, it is possible to skip entirely the phonon calculations and the reading of the files produced by pw.x by setting the variable

after_disp=.TRUE. and giving the name of the dynamical matrices file using the variable fildyn in the THERMO_CONTROL namelist. In this case thermo_pw can compute the anharmonic properties with a different set of temperatures, or with a different sampling on the phonon frequencies, etc.. You need to erase the output files that contain the phonon dos, or the thermal properties from a previous calculation, keeping the dynamical matrices files and the restart directory and rerun thermo_pw. Similarly if the files containing the bands energy eigenvalues are already in the working directory, it is possible to set the input variable only_bands_plot to change the bands plot without redoing the bands calculation. Note however that in this case it is not possible to change the Brillouin zone path.
The following variables can be used to stop thermo_pw before it concludes all the calculations:

```
max_seconds : the code stops after max_seconds have elapsed.
              Note that the check is not done continuously so
              the clean stop might occur a few minutes after
              max_seconds.
              Default: real 10E8

max_geometries : the code stops after computing the dispersions
              in max_geometries.
              Default: integer 1000000

start_geometry : the code starts doing the phonons for
              start_geometry.
              Default: integer 1

last_geometry : the code does only the phonons for geometries
               with index lower than last_geometry.
               Default: integer total number of geometries.
```

Note that the first time that you use start_geometry and last_geometry the codes makes a self-consistent pw.x run for all the geometries and saves the results in the outdir directory. If for any reason the outdir directory is removed after computing some geometries, just remove also the restart directory and the code will recreate the information in outdir but will not recalculate the dynamical matrices already available.

Finally we consider some typical runs of the most time consuming option what='mur_lc_t', but what we say is valid also for the computation of the phonon dispersions at a single geometry. With a single processor or with a personal computer with a small number of processors you can run the code without interruption. In general in these cases it is not useful to use the image parallelization before exploiting all the parallelization levels of QUANTUM ESPRESSO, since images have an overhead due to the necessity of reinitialize the phonon calculation and recalculate the bands. If you cannot complete the calculation in a single run you need to send several times the thermo_pw run. In this case you can set start_geometry=last_geometry, max_seconds in

the `thermo_pw` input, and `recover=.TRUE.` in the `ph.x` input. You need to repeat this for all the geometries and finally you can collect the results and compute the anharmonic properties.

6 Tools

The directory `tools` contains a few tools that can be useful to build structures of solids, surfaces, ribbons, and nanowires. Moreover it contains some miscellaneous codes that give additional information on the internal conventions of `thermo_pw` or further process its output. Currently it contains the following programs:

- `bravais_lattices.x` tests the module `lattices.f90` of the library. Presently it can read three primitive lattice vectors of a Bravais lattice and find the `ibrav` code and the `celldm` parameters of the input lattice. It can also read two sets of primitive vectors and decide if they describe the same Bravais lattice. In the positive case it gives the orientation of one lattice with respect to the other. For an example of its use see `tools_input/bravais_lattice.in`.

- `crystal_point_group.x` is a crystal point group calculator. It can give several information about the crystallographic point groups, such as the list of symmetry operations, the product of two rotation matrices, the product table, the class structure, the character tables of the irreducible representations of the point group and of the double point group and the projective representations of the point group. It gives the list of subgroups and supergroups of a given group and the compatibility tables of a given group with its subgroups. It can also decompose the Kronecker product representations. Finally it can list the conjugate groups and calculate the intersection of two point groups. For an example of its use see `tools_input/crystal_point_group.in`.

- `debye.x` reads from input the Debye temperature and the number of atoms per unit cell and writes in a file the thermodynamic quantities (vibrational energy, free energy, entropy, and heat capacity) as a function of temperature computed using the Debye model. When the system has only one atomic type it writes the atomic B factor as a function of temperature computed using the Debye model. For an example of its use see `tools_input/debye.in`.

- `elastic.x` reads the elastic constants of a solid and computes the elastic compliances, the bulk modulus, and a few poly-crystalline averages. It uses the `thermo_pw` library so the output is the same. For an example of its use see `tools_input/elastic.in`.

- `epsilon_tpw.x` generalizes the routine `epsilon.f90` of the QUANTUM ESPRESSO distribution. It calculates the complex dielectric constant of a solid as a function of the frequency for independent electrons using the LDA or GGA eigenvalues. It is limited to insulators, but supports norm-conserving, ultrasoft, and PAW pseudopotentials. It supports both scalar relativistic and fully relativistic pseudopotentials and it uses the point group symmetry of the solid to reduce the number of **k**-points. For an example of its use see `example14`.

- `gener_nanowire.x` reads a two dimensional (2D) Bravais lattice index and atomic coordinates an generates a sheet of type (m, n). A sheet contained between the two vectors **C**$= m\ \mathbf{a}_1 + n\mathbf{a}_2$ and **T** $= p\mathbf{a}_1 + q\mathbf{a}_2$ can be also generated and wrapped about **C** in a nanotube form ($\mathbf{a}_1$ and $\mathbf{a}_2$ are the primitive lattices of the 2D Bravais lattice). For lattices that allow it, p and q can be determined automatically so that **T** is perpendicular to **C**. For an example of its use see `tools_input/gener_nanowire.in`.

- `gener_2d_slab.x` reads a two dimensional Bravais lattice index and atomic coordinates an generates an infinite ribbon perpendicular to **G** $= m\ \mathbf{b}_1 + n\mathbf{b}_2$, where $\mathbf{b}_1$ and $\mathbf{b}_2$ are the primitive reciprocal lattice vectors of the 2D Bravais lattice. The number of rows of the ribbon, and the number of atoms per row are given as input variables. For an example of its use see `tools_input/gener_2d_slab.in`.

- `gener_3d_slab.x` reads a three dimensional Bravais lattice index and atomic coordinates and generates an infinite slab perpendicular to **G** $= m\mathbf{b}_1 + n\mathbf{b}_2 + o\mathbf{b}_3$, where $\mathbf{b}_1$, $\mathbf{b}_2$ and $\mathbf{b}_3$ are the primitive reciprocal lattice vectors of the Bravais lattice. The number of layers of each slab, and the size of the surface unit cell are given as input parameters. For an example of its use see `tools_input/gener_3d_slab.in`.

- `hex_trig.x` reads the values of a and c of the conventional hexagonal cell of a rhombohedral lattice (in Ångstrom), and gives as output the size a_r (in a.u.) and the cosine of the angle α of the rhombohedral cell. This information can be written in the input of `pw.x` for this type of cells. It is used to convert the structural information contained in a CIF file to the `pw.x` input.

- `kovalev.x` writes the correspondence between point group symmetry operations defined in the Kovalev tables and those used by QUANTUM ESPRESSO.

- `mag_point_group.x` gives a few information on the magnetic point groups.

- `optical.x` contains a few utilities for optical properties calculations. It transforms a complex dielectric constant into a complex index of refraction and computes the reflectivity or the absorption coefficient for cubic system. It converts also from energy of the photon in eV to the frequency in Hz or the wavelength in nm.

- `pdec.x` reads the temperature dependent elastic constants files calculated for several pressures and makes a plot of the elastic constants as a function of pressure at several temperatures.

- `plot_sur_states.x` reads the dump file produced by `thermo_pw` in a `what='scf_2d_bands'` calculation that contains the planar averages of all the states, and plots the states with the **k** point and the band numbers requested in input. For an example of its use see `tools_input/plot_sur_states.in`.

- `rotate_tensors.x` applies a rotation to a tensor of rank 1, 2, 3, or 4 defined in a coordinate system 1 and finds the form of the tensor in a new coordinate system 2.

- `space_groups.x` gives several information on space groups. It can give the names of the space group given the number reported in the International Tables for Crystallography (ITA), or the number given one of the names, translate the names between different editions of the ITA tables or the Shönflies name. It gives the list of coset representatives of each space group and the list of symmorphic space groups. For an example of its use see `tools_input/space_groups.in`.

- `supercell.x` reads a three dimensional Bravais lattice index and the atomic coordinates of the atoms inside a unit cell and produces a supercell with $n1 \times n2 \times n3$ cells of the original unit cell or a supercell delimited by three arbitrary Bravais lattice vectors given in crystal or cartesian coordinates. For centered cells there is the option to consider $n1$, $n2$, and $n3$ for the primitive or for the centered Bravais lattices. The input unit cell can be specified also by giving the space group and the coordinates of the nonequivalents atoms. It can be useful to study defects or to calculate all the atomic positions starting from the space-group and the nonequivalent positions. It is also possible to give the input Bravais lattice by using `ibrav=0` and the three principal vectors. In this case, before generating the supercell, the code rotates the Bravais lattice so that it has the same orientation of the vectors described in the `thermo.pdf` guide. For an example of its use see `tools_input/supercell.in`.

- `test_colors.x` produces a postscript file with the `gnuplot` colors that can be used in the plots.

- `translate.x` reads a set of atomic positions and a translation vector and translates the atomic positions. It can read also a rotation matrix and roto-translate the atomic positions.

- `units.x` writes on output the numerical constants used to write the guide `units.pdf` and `equilibrium.pdf`. It computes also the error associated to each conversion factor.

For a detailed description of the input variables please look at the beginning of the `fortran` sources of each code.

7 Examples, examples_qe, inputs, pseudo_test, space_groups, tools_inputs

The directories `examples`, `examples_qe`, `inputs`, `pseudo_tests`, `space_groups` and `tools_inputs` contain a set of examples that can be studied in order to learn how to use the `thermo_pw` package. The `examples` directory contains inputs that run in a few seconds but do not give converged results. These examples can be studied to see how the `thermo_pw` code works in the different cases. The reference directory of each example contains all the output files produced by the run. A one-to-one comparison with the output produced by running the example script is however not possible due to the asynchronous nature of the runs. Only the plotted physical quantities should be the same.
The directory `examples_qe` is used by developers. It contains examples mostly imported from QUANTUM ESPRESSO that are used to check that QUANTUM ESPRESSO functionalities are not spoiled by `thermo_pw`.
The directory `inputs` contains a set of realistic inputs and reasonably converged results. Not all output files are reported in the reference directory of each run. The `inputs` examples are divided according to the structure type and many material properties are calculated for each structure. This directory can be seen as a gallery of the results that can be obtained by the `thermo_pw` code, or as a source of information for the construction of a particular input geometry.
The directory `pseudo_test` contains a set of inputs that can be used to test a pseudopotential library. It illustrates how to use `thermo_pw` for high-throughput calculations.
The directory `space_groups` contains a collection of structures ordered by the space group number that are used to test the space groups routines. These inputs are also examples for the keyword `space_group` and of the Wyckoff positions to give the atomic coordinates in the `pw.x` input. You can also use these structures for your calculations, but note that the cut-off energies and the **k**-point meshes are not converged.
The directory `tools_inputs` give some examples of the inputs of the auxiliary `tools` programs.

8 Color codes

In this section we briefly summarize the color codes of some of the figures that can be obtained from `thermo_pw`.

- Total energy versus kinetic energy. This is a single figure of the total energy versus wave-functions kinetic energy cut-offs. When the test requires several charge density cut-offs there is a different curve for each charge density cut-off. The curve corresponding to the lowest charge density cut-off is `red`, the one corresponding to the highest is `blue`, all the others are `green`. Note that the total energy of the last configuration (highest wave function and charge density cut offs) is subtracted from all energies.

- Total energy versus size of the **k**-point mesh. This is a single figure of the total energy as a function of the size of the **k**-point mesh. When the test requires several values of `degauss`, there is a curve for each `degauss`. The curve corresponding to the first `degauss` is `red`, the one corresponding to the last is `blue`, all the others are `green`. Note that the total energy of the last configuration (highest number of points and lowest `degauss`) is subtracted from all energies.

- Total energy as a function of volume (`lmurn=.TRUE.`). This plot is composed of two figures. Total energy as a function of volume and pressure as a function of volume. Both curves are `red`. The points on the first curve are the energies calculated by `pw.x`, the continuous curve is the fit.

- Total energy as a function of one or two crystallographic parameters (`lmurn=.FALSE.`). When there is a single parameter the curve is `red` as in the case `lmurn=.TRUE.`. When there are two parameters a contour plot of the energy as a function of two parameters is shown. The contour levels, their number and their colors can be given in input. By default the code shows nine levels with three colors. From the lowest to the highest levels, the colors are `red`, `green`, and `blue`. The energy value of each level is written on output. When the user requests more levels without specifying their colors, the code continues with three `yellow` levels, then `pink`, `cyan`, `orange`, `black`, and when more than 24 levels are requested the sequence of colors is repeated. For orthorhombic solids the code produces many postscript figures, one for each value of c/a on the grid. In each figure there is a contour plot of the energy as a function of a and b/a. The colors of the levels follow the same conventions of the previous case. When the levels are chosen by the code the entire energy range (for all c/a) is divided into nine levels so each figure might have less that nine curves. For crystal systems with more crystallographic parameters, this figure is not available.

- Energy bands. In this figure the bands have the color of their irreducible representation. Each line of the path can have a different point group

and set of representations. See the `point_groups.pdf` file for the list of representations and their color code. When the symmetry analysis is not done all the bands are `red`.

- Energy bands with `enhance_plot=.TRUE.`. In this case the background color of the panels with lines at the zone border are gray, yellow, or pink. Gray means that the point group (or double point group) representations are used, yellow or pink means that a gauge transformation was applied and projective representations might have been used. A yellow background indicates that no switch from the point group to the double point group or viceversa was made, while a pink background means that such a switch was necessary.

- Electron density of states. This is a plot composed of two figures, the first contains the electron density of states, the second the integral of the density of states up to that energy. The dos is `red`. In the local spin density case, the dos for spin up is `red` the one for spin down is `blue` and with a negative sign. The integrated density of states is `blue`. In the spin polarized case, the curve shows the integral of the sum of the up and down density of states.

- Electronic energy, free energy, entropy, and isochoric heat capacity (metals only). This plot is composed of four pictures one for each quantity. There is a single `blue` curve per plot.

- Dielectric constant as a function of frequency ($\mathbf{q} = 0$). There are two plots, one for the real part and one for the imaginary part. Other two plots contain the real and imaginary part of the complex index of refraction. For cubic solids other two plots show the reflectivity for normal incidence and the absorption coefficient. All curves are in red. For hexagonal, trigonal, and tetragonal systems the xx component is in red, while the zz component is in green. For orthorombic systems the xx component is in red, the yy component in green and the zz component in blue. For monoclinic and triclinic systems the plot is not done.

- Inverse of the dielectric constant as a function of frequency ($\mathbf{q} \neq 0$). There are four plots: the real and imaginary part of $\varepsilon(\mathbf{q},\omega)$ and the real and imaginary part of $1/\varepsilon(\mathbf{q},\omega)$. They are all in red. Note that the latter is really calculated, while the first is just its inverse.

- Phonon dispersions. In this figure the phonon dispersions have the color of their irreducible representations. The same comments made for the plot of the band structure apply here.

- Phonon dos. There is one picture with a single `red` curve.

- Vibrational energy, free energy, entropy, and isochoric heat capacity. This plot is composed of four figures each one showing one quantity. In `red` the quantities obtained using the phonon density of states, in `blue`

those obtained from integration over the Brillouin zone. In some cases the `red` curve is not visible because it is exactly below the `blue` one.

- Atomic B factors as a function of temperature. This plot is composed of one figure for each atom for cubic solids and of two figures for each atom in the other cases. One figure contains B_{xx} (red, pink), B_{yy} (blue, light_blue) and B_{zz} (dark_green, green) as a function of temperature. The first color refers to quantities calculated from generalized phonon density of states while the second refers to quantities calculated by Brillouin zone integration. If the curves coincide, only the last one (green) will be visible. The second figure, when plotted shows B_{xy} (red, pink), B_{xz} (blue, light_blue), and B_{yz} (dark_green, green).

- Debye vibrational energy, free energy, entropy, and isochoric heat capacity. This plot is composed of four figures, one for each quantity. The curves are in `blue` and the word Debye appears in the y axis label.

- Equilibrium volume, Helmholtz (or Gibbs at finite pressure) free energy, bulk modulus, pressure derivative of the bulk modulus, volume thermal expansion, isochoric heat capacity, isobaric heat capacity, isobaric-isochoric difference of the heat capacity, isoentropic-isothermal difference of the bulk modulus, and average Grüneisen parameter as a function of temperature (`lmurn=.TRUE.`). This plot is composed of ten figures, one for each quantity in the order given above. The quantities calculated using the phonon dos are in `red`, those obtained by an integral over the Brillouin zone are in `blue`. The volume thermal expansion can be calculated using the mode Grüneisen parameters and is plotted in `green`. The isobaric-isochoric difference of the heat capacity, the isoentropic-isothermal difference of the bulk modulus, and average Grüneisen parameter calculated with this volume thermal expansion are plotted in `green`. The volume and the bulk modulus used to compute the volume thermal expansion via the mode Grüneisen parameters are the `blue` ones if `ltherm_freq=.TRUE.` or the `red` ones if `ltherm_freq=.FALSE.` and `ltherm_dos=.TRUE.`. When both `ltherm_freq=.FALSE.` and `ltherm_dos=.FALSE.` the equilibrium values of the volume and of the bulk modulus calculated at $T = 0$ K are used at all temperatures. The equilibrium volume at $T = 0$ K is used at all temperatures also when `lv0_t=.FALSE.`. The equilibrium bulk modulus at $T = 0$ K is used all temperatures also when `lb0_t=.FALSE.`

- Crystallographic parameters, volume, Helmholtz (or Gibbs at finite pressure) free energy, thermal expansion tensor, volume thermal expansion, constant strain heat capacity (C_ϵ), isobaric heat capacity (C_P), difference $C_P - C_V$ of isobaric and isochoric heat capacities, difference $C_\sigma - C_\epsilon$ of constant stress and constant strain heat capacities (note that $C_\sigma = C_P$), difference $C_V - C_\epsilon$ of isochoric and constant strain heat capacities, difference $B_S - B_T$ of the isoentropic and isothermal bulk modulus, and average Grüneisen parameter as a function of temperature (`lmurn=.FALSE.`).

The number of figures in this plot depends on the crystal system and on the presence of one or more files with the elastic constants. It shows a as a function of temperature for cubic solids, a, c/a, and c for tetragonal and hexagonal solids. For orthorhombic solids it shows also b/a and b while for trigonal solids it shows a and $\cos\alpha$. For monoclinic solids it shows a, b/a, b, c/a, c, and $\cos\alpha$ (c-unique) or $\cos\beta$ (b-unique). All the six crystallographic parameters as a function of temperature are shown for triclinic solids. All quantities calculated using the phonon density of states are in `red`, those calculated integrating over the Brillouin zone are in `blue` with the exception of the thermal expansion tensor. When this tensor is diagonal with all identical components it follows the above rules while the tensor computed from mode Grüneisen parameters is in `green`. For hexagonal, tetragonal and trigonal solids α_{xx} follows the above rules while α_{zz} is `pink`, `cyan`, and `orange` when computed from phonon density of states, Brillouin zone integration, or mode Grüneisen parameters, respectively. In the orthorhombic case α_{xx} and α_{zz} have the same colors, while α_{yy} is `gold`, `olive`, and `light-blue` in the three cases, respectively. For the other crystal systems the thermal expansion tensor is not given. The thermal expansion tensor from the mode Grüneisen parameters is calculated only when the `elastic_constants` directory contains at least one file with the elastic constants. In this case also $C_P - C_V$, $C_\sigma - C_\epsilon$, $C_V - C_\epsilon$, and the average Grüneisen parameters are calculated using this thermal expansion tensor and plotted in `green`. The volume used in these calculations is the `blue` curve if `ltherm_freq=.TRUE.` or as in the `red` curve if `ltherm_freq=.FALSE.` and `ltherm_dos=.TRUE.`. When both `ltherm_freq=.FALSE.` and `ltherm_dos=.FALSE.` the volume is kept fixed at the equilibrium volume at $T = 0$ K. The same applies for the bulk modulus calculated from a single elastic constant file when the flag `lb0_t=.FALSE.` or computed within the "quasi-static" approximation when `lb0_t=.TRUE.`. The C_P, $C_\sigma - C_\epsilon$, $C_V - C_\epsilon$, and the average Grüneisen parameter are plotted only in presence of one or more elastic constants file.

- Thermal stresses as a function of temperature. This plot is composed of one figure in cubic solids and of two figures in the other cases. One figure contains b_{xx} (red, pink), b_{yy} (blue, light_blue) and b_{zz} (dark_green, green) as a function of temperature. The first color refers to quantities calculated from phonon density of states while the second refers to quantities calculated by Brillouin zone integration. If the curves coincide, only the last one (green) will be visible. The second figure, when plotted, shows b_{xy} (red, pink), b_{xz} (blue, light_blue), and b_{yz} (dark_green, green).

- Mode Grüneisen parameters. In this plot the mode Grüneisen parameters have the color of the irreducible representation of the phonon dispersion curve of which they are the derivative. The same comments made for the band structure plot apply here.

- Generalized average Grüneisen parameters as a function of temperature.

This plot is composed of one figure in cubic solids and of two figures in the other cases. One figure contains γ_{xx} (red, pink), γ_{yy} (blue, light_blue) and γ_{zz} (dark_green, green) as a function of temperature. The first color refers to quantities calculated from phonon density of states while the second color refers to quantities calculated by Brillouin zone integration. If the curves coincide, only the last one (green) will be visible. The second figure, when plotted shows γ_{xy} (red, pink), γ_{xz} (blue, light_blue), and γ_{yz} (dark_green, green).

- Phonon dispersions at the geometry that corresponds to a given temperature. The colors are assigned on the basis of the irreducible representation of each mode. The same comments made for the band structure plot apply here.

- Temperature dependence of the isothermal and isoentropic elastic constants within the 'Quasi-static', 'Fixed geometry quasi-harmonic' or 'Quasi-harmonic approximation'. There is a plot for each non-zero elastic constant and a plot of the bulk modulus. The number of plots depends on the Laue class. Elastic constants interpolated at the geometry computed using the phonon density of states are in `red` (isothermal) and `green` (isoentropic), those calculated from integration over the Brillouin zone are in `blue` (isothermal) and `orange` (isoentropic).

- Temperature dependence of the isothermal and isoentropic elastic compliances within the 'Quasi-static', 'Fixed geometry quasi-harmonic' or 'Quasi-harmonic approximation'. There is a plot for each non-zero elastic compliance and a plot of the compressibility. The number of plots depends on the Laue class. Elastic compliances interpolated at the geometry computed using the phonon density of states are in `red` (isothermal) and `green` (isoentropic), those calculated from integration over the Brillouin zone are in `blue` (isothermal) and `orange` (isoentropic).

- Temperature dependence of the isothermal elastic constants within the 'Fixed geometry quasi-harmonic approximation' for all the geometries of the mesh. There is a plot for each non-zero elastic constant and a plot of the bulk modulus. The number of plots depends on the Laue class. Elastic constants of the different geometries are in the sequence `red`, `green`, `blue`, `yellow`, `pink`, `cyan`, `orange`, `black`. When there are more than eight geometries the sequence is repeated. The same colors are used for the elastic constants obtained with the phonon density of states or from the integration over the Brillouin zone.

- Temperature dependence of the isothermal elastic compliances within the 'Fixed geometry quasi-harmonic approximation' for all the geometries of the mesh. There is a plot for each non-zero elastic compliance and a plot of the compressibility. The number of plots depends on the Laue class. Elastic compliances of the different geometries are in the sequence `red`, `green`, `blue`, `yellow`, `pink`, `cyan`, `orange`, `black`. When there are more than eight geometries the sequence is repeated. The same colors are used

for the elastic compliances obtained with the phonon density of states or from the integration over the Brillouin zone.

9 Documentation

In addition to this user's guide, this directory contains the following documents:

- `tutorial.pdf` : a short guide that indicates where to find the information needed to compute a given quantity.
- `point_groups.pdf` : a description of the crystallographic point groups, character tables of the irreducible representations of groups and of the double point groups and tables of the projective representations, for the interpretation of the color codes in the band and phonon dispersion plots.
- `thermo.pdf` : some notes on the thermodynamic expressions implemented in `thermo_pw`.
- `developer_guide.pdf` : some notes on the internal logic of `thermo_pw`.