THERMO_PW Tutorial (v.2.0.0)

Andrea Dal Corso (SISSA - Trieste)

Contents

Introduction

This guide gives a brief overview of the THERMO_PW package. It gives you the minimum information needed to accomplish a given task. It assumes that you have already downloaded and compiled QUANTUM ESPRESSO and THERMO_PW and you want to calculate some material property but you do not know where to start. The most complete reference for the use of THERMO_PW is its user's guide. However the capabilities of the code are rapidly expanding and the user's guide is getting long so, if you do not want to read it entirely, here you find where to search the relevant information. There might be some limitations on the type of functionals that you can use or the electronic structure scheme available to compute a given quantity. Please refer to the QUANTUM ESPRESSO documentation for this.

0.1 People

This tutorial has been written by Andrea Dal Corso (SISSA - Trieste).

0.2 Overview

In order to make a calculation with THERMO_PW you need an input file for the `pw.x` code of QUANTUM ESPRESSO. This input file requires mainly five information:

- The Bravais lattice.
- The position of the atoms inside the unit cell.
- The type of atoms and the pseudopotentials files that you want to use.
- The cut-off energies.
- The **k**-point mesh used to make integration over the Brillouin zone. The smearing parameter for metals.

The Bravais lattice is specified by an integer number `ibrav` and by the crystal parameters `celldm` (up to six numbers). The `ibrav` codes and the required crystal parameters are described in the file `PW/Doc/INPUT_PW` of the QUANTUM ESPRESSO distribution. In QUANTUM ESPRESSO you can use `ibrav=0` and give the primitive lattice vectors of the Bravais lattice. Presently THERMO_PW needs to know the Bravais lattice number so this form of input is not recommended. If you use it, THERMO_PW writes on output `ibrav`, `celldm` and the atomic coordinates needed to simulate the same cell and stops. You can just cut and paste these quantities in the input of `pw.x` or you can set the flag `find_ibrav=.TRUE.` in the THERMO_PW input and THERMO_PW will make the conversion for you before runnig the job. After setting the correct `ibrav` and `celldm`, THERMO_PW might still tell you that the Bravais lattice is not compatible with the point group. This can happen, for instance, if you have isolated molecules, amorphous solids, defects, or supercells. In these cases you can still continue but symmetry will not be used to reduce the number of components of the physical quantities tensors. In order to use the residual symmetry, you have to use one of the suggested `ibrav`, adjusting the `celldm` to the parameters of your cell. For instance if you have a cubic cell, but the symmetry requires a tetragonal lattice, you have to use a tetragonal lattice with `celldm(3)=1.0`. In rare cases, with lattices such as the face-centered orthorhombic some symmetry operations might be incompatible with the FFT grid found by `pw.x`. The choice made in QUANTUM ESPRESSO is to discard these symmetries making the lattice incompatible with the point group. In these cases the code needs `nr1=nr2=nr3`. Set these three parameters in the `pw.x` input equal to the largest one.

The positions of the atoms inside the unit cell are defined by an integer number `nat` (the number of atoms) and by `nat` three-dimensional vectors as explained in the file PW/Doc/INPUT_PW. You can use several units, give the coordinates in the Cartesian or in the crystal basis or you can give the space group number and the crystal coordinates of the nonequivalent atoms. Note that in centered lattices the crystal basis is the conventional one when using the space group number and the primitive one when not using it. These options are supported by THERMO_PW. See the pw.x manual for details.

The number of different types of atoms is given by an integer number ntyp and for each atomic type you need to specify a pseudopotential file. Pseudopotential files depend on the exchange and correlation functional and can be found in many different places. There is a pseudopotential page in the QUANTUM ESPRESSO website, or you can consider generating your pseudopotentials with the pslibrary inputs. You can consult the web page

https://dalcorso.github.io/pslibrary/

for more information.

The kinetic energies cut-offs depend on the pseudopotentials and on the accuracy of your calculation. You can find some hints about the required cut-offs inside the pseudopotentials files, but you need to check the convergence of your results with the cut-off energies. Many tests of the kinetic energy cut-offs can be found also at https://www.materialscloud.org/discover/sssp/.

The **k**-point mesh is given by three integer numbers and possible shifts (0 or 1) in the three directions. The convergence of the results with this mesh should be tested. For metals you have also to specify a smearing method (for instance occupations='smearing', smearing='mp') and a value of the smearing parameter (in Ry) (see the variable degauss in the file PW/Doc/INPUT_PW). Note that the convergence with respect to the **k**-points depends on the value of degauss and must be tested for each degauss.

Once you have an input for pw.x, in order to run THERMO_PW you have to write a file called thermo_control that contains a namelist called INPUT_THERMO. This namelist contains a keyword what that controls the calculation performed by THERMO_PW. Ideally you need to set only what and call thermo_pw.x instead of pw.x, giving as input the input prepared for pw.x. In practice each what is controlled by several variables described in the user's guide. These variables have default values that are usually sufficient to give a first estimate of a given quantity but that must be fine tuned to obtain publication quality results.

Chapter 1

Howtos

1.1 How do I make a self-consistent calculation?

Use `what='scf'`. See `example01`. The calculation is equivalent to a call to `pw.x` and is controlled by its input. In particular in the input of `pw.x` you can choose a single self-consistent calculation using `calculation='scf'`, an atomic relaxation using `calculation='relax'`, or a cell relaxation using `calculation='vc-relax'`. The use of `calculation='nscf'` and `calculation ='bands'` is not supported by THERMO_PW and could give unpredictable results. There is no advantage to use THERMO_PW to do a molecular dynamic calculation.

1.2 How do I plot the band structure?

Use `what='scf_bands'`. See `example02`. With this option THERMO_PW calls `pw.x` twice, making first a self-consistent calculation with the parameters given in the `pw.x` input and then a band calculation along a path chosen by THERMO_PW, or along a path given by the user after the `INPUT_THERMO` namelist. In this case the path is given as in the `pw.x` input (see the user's guide for additional details). There are a few parameters that you can give in the `INPUT_THERMO` namelist to control the band plot. The most useful are `emin_input` and `emax_input` that allow you to plot the bands in a given energy range. At the end of the run, the figure of the bands is in a file called by default `output_band.ps`, a name that can be changed in the THERMO_PW input. Check also the option `what='scf_2d_bands'` to plot the surface band structure.

1.3 How do I plot the electronic density of states?

Use `what='scf_dos'`. See `example18`. With this option THERMO_PW calls `pw.x` twice, making first a self-consistent calculation followed by a non self-consistent calculation on a uniform **k**-point mesh. This mesh can be specified in the THERMO_PW input (if none is given THERMO_PW uses the default values). At the end of the run, the figure of the density of states is in a file called by default `output_eldos.ps`, a name that can be changed in the THERMO_PW input.

1.4 How can I see the crystal structure?

THERMO_PW is not a crystal structure viewer, but you can use the `XCrySDen` code, which reads the `pw.x` input, to see the crystal structure. If you use `what='plot_bz'`, THERMO_PW produces a `.xsf` file with the input structure that can be read by `XCrySDen`. This could be useful when you give the space group number and the nonequivalent atomic positions since this input is presently not readable by `XCrySDen`. The generated `.xsf` file contains all the symmetry equivalent atomic positions. For the same purpose you could also use the output of `pw.x`.

1.5 How can I see the Brillouin zone?

Use `what='plot_bz'`. See `example12`. With this option THERMO_PW does not call `pw.x`, it just produces a script for the `asymptote` code with the instructions to plot the Brillouin zone and the standard path (or the path that you have given in the THERMO_PW input).

1.6 How can I plot the X-ray powder diffraction spectrum?

Use `what='plot_bz'` to see the spectrum corresponding to the geometry given in the `pw.x` input. You can also see the spectrum corresponding to a relaxed structure using for instance `what='scf'`, asking for an atomic (cell) relaxation in the `pw.x` input and using `lxrdp=.TRUE.` variable in the THERMO_PW input. The X-ray powder diffraction spectrum is shown in a file called by default `output_xrdp.ps`, a name that can be changed in the THERMO_PW input. The scattering angles and intensities are also written in a file called by default `output_xrdp.dat`, which can also be changed in the THERMO_PW input.

1.7 How can I find the space group of my crystal?

Use `what='plot_bz'` and look at the output. The space group is identified. In the case you have a structure with `ibrav=0` and the primitive lattice vectors use the option `find_ibrav=.TRUE.` in the THERMO_PW input (see the THERMO_PW user's guide in the subsection *Coordinates and structure*). This option has presently some limitations. It does not work for noncollinear magnetic system, or for supercells, or when the Bravais lattice and the point group are incompatible.

1.8 How do I plot the phonon dispersions?

Use `what='scf_disp'`. See `example04`. In this case you have to prepare an input for the `ph.x` code that must be called `ph_control`. The required information in this input is the **q**-point mesh on which the dynamical matrices are computed and the name of the files where the dynamical matrices are written. See the `ph.x` guide if you need information on this point. At the end of the run, the phonon dispersions are found in a file called by default `output_disp.g1.ps`, a name that can be changed in the THERMO_PW input. The vibrational density of states is found in a file called by default `output_dos.g1.ps`, which can also be changed in the THERMO_PW input.

1.9 How do I calculate the vibrational energy, free energy, entropy, and heat capacity?

Use `what='scf_disp'`. See `example04`. These quantities are calculated after the phonon dispersion for the default temperature range (1 K - 800 K) or for the range given in the THERMO_PW input. The figure of these quantities is in the file `output_therm.g1.ps`. Note that they are calculated at the geometry given in the input of `pw.x`.

1.10 How do I calculate the atomic B-factor of a solid?

Use `what='scf_disp'` as in `example04` and add the flag `with_eigen=.TRUE.`. These quantities are calculated after the phonon dispersions for the default temperature range (1 K - 800 K) or for the range given in the THERMO_PW input. The figure of these quantities is in the file `output_therm.g1_dw.p` Note that they are calculated at the geometry given in the input of `pw.x`.

1.11 How do I calculate the elastic constants?

Use `what='scf_elastic_constants'`. See `example13`. The elastic constants appear in the output of THERMO_PW and also in a file called by default `output_ el_cons.dat`, a name that can be changed in the THERMO_PW input. This file can be read by THERMO_PW for the options that require the knowledge of the elastic constants.

1.12 How do I calculate the Debye temperature?

Use `what='scf_elastic_constants'`. See `example13`. The Debye temperature appears in the output of THERMO_PW. A file called `output_therm_debye.g1.ps` contains plots of the vibrational energy, free energy, entropy, and heat capacity computed within the Debye model.

1.13 How do I calculate the equilibrium structure?

Use `what='mur_lc'`. See `example05` for the cubic case and refer to the user's guide for anisotropic solids. The crystal parameters are written in the THERMO_PW output file. Note that the structure is searched interpolating with a polynomial or with the Murnaghan equation the energy calculated for several geometries close to the geometry given in the input of `pw.x` so the closer this structure to the actual equilibrium structure the better the fit and the closer the value found by THERMO_PW to the real minimum. You can check on the file `output_mur.ps` (when `lmur=.true.`) or `output_energy.ps` (when `lmur=.false.`) if the minimum is within the range of calculated structures. If it is not, the calculated minimum is probably inaccurate and it is better to repeat the calculation starting from it. Note also that almost all options can be specified using `what='mur_lc_...'` instead of `what='scf_...'`. In this case the calculations are carried out at the equilibrium geometry instead of the geometry given in the `pw.x` input. Setting a finite pressure in the THERMO_PW input, the equilibrium geometry is the one at the given pressure (see below) and the calculations are carried out at this geometry.

1.14 How do I calculate the equilibrium structure at a given pressure?

Use `what='mur_lc'` and specify `pressure=value` in kbar in the THERMO_PW input. Note that in the input of `pw.x` you should give a geometry which is as close as possible to the equilibrium value found at the given pressure (see above).

1.15 How do I specify the temperature range

See the subsection *Temperature and Pressure* in the THERMO_PW user's guide.

1.16 How do I calculate the crystal parameters as a function of temperature?

Use `what='mur_lc_t'`. See `example09`. Note that for this option you need to give also the `ph.x` input. For anisotropic solids using `lmurn=.TRUE.` you calculate the volume as a function of temperature varying `celldm(1)` but all the other crystal parameters are kept constant, while using `lmurn=.FALSE.` you can calculate all the crystal parameters as a function of temperature. The crystal parameters are plotted as a function of temperature in the standard range ($T = 1$ K - $T = 800$ K) or in the range requested in input, in the file `output_anharm.ps`, a name that can be changed in the THERMO_PW input. Presently no temperature dependence is calculated for the atomic coordinates, so the present approach is applicable to solids in which equilibrium atomic positions are fixed by symmetry, while it is an approximation in the other cases.

1.17 How do I calculate the thermal expansion?

Use `what='mur_lc_t'`. See `example09`. The components of the thermal expansion tensor are shown as a function of temperature in the file `output_anharm.ps`, a name that can be changed in the THERMO_PW input.

1.18 How do I calculate the Helmholtz (or Gibbs) free energy as a function of temperature keeping into account the thermal expansion?

Use `what='mur_lc_t'`. The Helmholtz (or Gibbs at finite pressure) free energy is shown as a function of temperature in the file `output_anharm.ps`. Note that the absolute value of this energy depends on the pseudopotentials as the total energy. You can however compare the free energies for different crystal structures and predict if a phase transition occurs and at which temperature (also as a function of pressure).

1.19 How do I calculate the bulk modulus as a function of temperature?

Use `what='mur_lc_t'` and the option `lmurn=.TRUE.`. This approach is rigorously valid only for cubic solids, for anisotropic solid it is an approximation in which only `celldm(1)` is changed while the other crystal parameters are kept constant. For the general case, see the elastic constants as a function of temperature. This calculation gives also the bulk modulus as a function of temperature.

1.20 How do I calculate the isobaric heat capacity?

Use `what='mur_lc_t'` and the option `lmurn=.TRUE.`. This approach is rigorously valid only for cubic solids, for anisotropic solid it is an approximation in which only `celldm(1)` is changed while the other crystal parameters are kept constant. For the general case, see the elastic constants as a function of temperature. This calculation gives also the isobaric heat capacity as a function of temperature.

1.21 How do I calculate the elastic constants as a function of temperature?

There are two ways. Both of them are a two step calculation. The fastest, but less accurate, method uses the "quasi-static" approximation. First use what='elastic_constants_geo'. This option computes the elastic constants at $T = 0$ K for all the geometries used by what='mur_lc_t' and saves them in the directory elastic_constants. In the second step, using the same input, run again THERMO_PW with what='mur_lc_t' after copying in your working directory the directory elastic_constants obtained in the previous step. Elastic constants are read only when lmurn=.FALSE.. The "quasi-static" approximation means that the code interpolates the $T = 0$ K elastic constants found in the first step at the geometry that minimizes the Helmholtz (or Gibbs at finite pressure) free energy at temperature T.
The second method uses the "quasi-harmonic" approximation and requires many phonon calculations at many geometries. First use what= 'elastic_constants_geo' and set use_free_energy= .TRUE.. This option computes the temperature dependent elastic constants taking as unperturbed geometries those used by what='mur_lc_t' and saves them in the directory anhar_files. In the second step, using the same input, run again THERMO_PW with what= 'mur_lc_t' after copying in your working directory the directory anhar_files obtained in the previous step. Elastic constants are read only when lmurn= .FALSE.. The "quasi-harmonic" approximation means that the code interpolates the temperature dependent elastic constants found in the first step as second derivatives of the Helmholtz free energy (corrected to give the stress-strain elastic constants), at the geometry that minimizes the Helmholtz free energy.
The plot of the elastic constants will be found in a file whose default name is output_anhar.el_cons.ps. Note that when the elastic constant are available in a file, with the option lmurn=.FALSE. the thermodynamic properties are calculated also for non-cubic solids. The main approximation of the present implementation is that the atomic coordinates are relaxed only at $T = 0$ K minimizing the energy and the free energy is not calculated as a function of the atomic positions. Please be careful to use the same tmin, tmax, and deltat in these calculations. Only a tmax larger in the what='elastic_constants_geo' calculation than in what='mur_lc_t' is allowed.

1.22 How do I calculate the electronic heat capacity of a metal?

Use `what='scf_dos'`. See `example18`. In the metallic case in addition to a plot of the density of states this option produces also a plot of the electronic energy, free energy, entropy, heat capacity, and chemical potential in the standard temperature range ($T = 1$ K - $T = 800$ K) or in the range requested in input. These quantities are found in the file called by default `output_eltherm.ps`, a name that can be changed in the THERMO_PW input. Please be careful with the value of `deltae` or the electronic thermodynamic properties could be wrong at low temperatures.

1.23 How do I calculate the thermal expansion of a metal accounting for the free energy due to the electronic excitations?

This is a two step calculation. First use `what='mur_lc'` and the flag `lel_ free_energy=.TRUE.` to write on files the electronic thermodynamic properties for each geometry used by `what='mur_lc'`. Then copy the files with the electronic thermodynamic properties in the `therm_files` directory and run THERMO_PW with `what='mur_lc_t'` and the flag `lel_free_energy=.TRUE.`. The electronic free energy is read from files and added to the vibrational free energy before computing the equilibrium crystal structure. See `example24` for this case.

1.24 How do I calculate the elastic constants of a metal accounting for the free energy due to the electronic excitations?

This is a four step calculation. First use `what='mur_lc'` and the flag `lel_ free_energy=.TRUE.` as in the previous point in order to get the electronic thermodynamic properties for all the unperturbed geometries. Then use `what= 'elastic_constants_geo'` with the flag `lel_free _energy=.TRUE.` and the flag `use_free_energy=.FALSE.` to write on file the electronic thermodynamic properties for all strained geometries. Then use `what='elastic_constants_geo'` with `lel_free _energy=.TRUE.` and `use_free_energy=.TRUE.` after copying the electronic thermodynamic properties obtained at previous point in the `therm_files` directory, so the free energy derived in this step to obtain the elastic constants has also the electronic contribution. Finally use `what='mur_lc_t'` with `lel_free_energy=.TRUE.` and `use_free_energy=.TRUE.` after copying the elastic constants obtained at previous step in the `anhar_files` directory and the electronic thermodynamic properties obtained in the first step in the `therm_files` directory.

1.25 How do I calculate the temperature dependent elastic constants of a solid relaxing the internal degree of freedom?

This is possible only within the ZSISA approximation. This means that for each value of the crystal parameters the ions are relaxed (with `frozen_ions=.FALSE.`) to their $T = 0$ K equilibrium positions. The phonons (and the free energy) are computed only in this configuration. For solids in which internal relaxations are described by one degree of freedom, it is possible to compute the phonons (and the free energy) for several values of this parameter. The free energy as a function of this parameter is then minimized at each strain and each temperature (See the parameter `stype` in `what='scf_elastic_constants'`).

1.26 How do I calculate the frequency dependent dielectric constant of a material?

Use `what='scf_ph'`. See `example16` and `example20`. Please note that this calculation is made by the extended version of the phonon code available in THERMO_PW, so you have to provide specific flags to the phonon input. See the user's guide for details. The dielectric constant is in the file `output_epsilon.ps`. Note that for metals you need to specify a finite wave-vector **q** and you cannot plot the dielectric constant when **q** is the Γ point. See `example17` and `example21` for this case.

1.27 How do I calculate the frequency dependent reflectivity of a material?

The calculation of this quantity is implemented only for isotropic (cubic) solids. It is plotted together with the frequency dependent dielectric constant in insulating cubic solids when $\mathbf{q}$ is the Γ point. In this case the code plots also the absorption coefficient. Both quantities can be plotted as a function of the frequency or of the wavelength.

1.28 Which is the meaning of the colors in the electronic bands and phonon dispersions plots?

Different colors correspond to different irreducible representations of the point co-group of the **k** or **q** wavevector. To see the correspondence color-representation see the `point_groups.pdf` manual. The point group used for each **k** or **q** point is written in the THERMO_PW output and also in the plot if you set `enhance_plot=.TRUE.`. In the output you can also find, close to each band energy or phonon frequency value, the name of the irreducible representation. Relevant character tables are given in the `point_groups.pdf` manual, in the THERMO_PW output, or by the `crystal_point_group.x` tool.

1.29 How do I specify a custom path for plotting the electronic bands and phonon dispersions?

You have to give an explicit path after the `INPUT_THERMO` namelist (see `what='scf_bands'` and `what='scf_2d_bands'`). There are two possibilities to specify a path in the Brillouin zone. You can use explicit coordinates of the first and last points of each line (in cartesian or crystal coordinates) and a weight that indicates the number of points in the line, or you can use labels to indicate the first and/or the last points. The available labels depend on the Brillouin zone and are illustrated in the file `Doc/brillouin_zones.pdf` in the main QE documentation. Labels for points given by explicit coordinates can be added after the weights. Usually, the same label that indicates the coordinates of the point is written in the dispersion plot, but it is also possible to override this label, adding a letter after the weight as for the points given by explicit coordinates. Note that labels are given without quotes, while the additional letter after the weight must be written between single quotes. An example of a path for the fcc lattice:

```
7
gG   20
X    20
W    20  'gS1'
1.0 1.0 0.0 20 'X'
gG    0
gG   20
L     0
```

This path has five lines, the first from Γ to X with 20 points, the second from X to W (that in the plot will have the label Σ_1) with 20 points, the third from W to the point of coordinates $(1.0, 1.0, 0.0)$ (that in the plot will have the label X), the fourth from $(1.0, 1.0, 0.0)$ to Γ with 20 points and a final line from Γ to L with 20 points. Until version `1.8.1` labels are read as three characters, so it is important not to write the weights too close to the labels. Grüneisen parameters are not analytic at the Γ point so to obtain a correct plot, the Γ label in the middle of a plot must be repeated twice, as in the example.

1.30 How do I specify the plane of k-points when I set the flag `q2d=.TRUE.`?

In this case you must give three **k** points: $\mathbf{k}_1$, $\mathbf{k}_2$, and $\mathbf{k}_3$ after the `INPUT_THERMO` namelist. $\mathbf{k}_1$ is the origin and the two vectors $\mathbf{a}_1 = \mathbf{k}_2 - \mathbf{k}_1$ and $\mathbf{a}_2 = \mathbf{k}_3 - \mathbf{k}_1$ determine the plane

$$\mathbf{k} = \mathbf{k}_1 + \lambda \mathbf{a}_1 + \mu \mathbf{a}_2, \tag{1.1}$$

where $0 \leq \lambda \leq 1$, $0 \leq \mu \leq 1$. The interval of λ is divided into n_1 points, while μ is divided n_2 points, where n_1 and n_2 are the weights of the points $\mathbf{k}_2$ and $\mathbf{k}_3$. An example of the input for determining the plane is:

```
3
k_{x,1}  k_{y,1}  k_{z,1}  n0
k_{x,2}  k_{y,2}  k_{z,2}  n1
k_{x,3}  k_{y,3}  k_{z,3}  n2
```

where the **k** points are given in cartesian coordinates in units of $\frac{2\pi}{a}$ (a is the lattice constant `celldm(1)` or in crystal coordinates using `q_in_cryst_coord=.TRUE.` in the `INPUT_THERMO` namelist. n_0 is not used and can be set to any integer value.