D3Q

D3Q manual

Foreword

The thermal2 suite of codes has been written starting in 2010 by Lorenzo Paulatto1. It is based on a previous code developed by Michele Lazzeri and Stefano de Gironcoli2 which has been distributed with the Quantum−ESPRESSO suite of codes.

Copyright

Citing

We would greatly appreciate if when using the d3q code you cite the following papers where the underlying theory is described in detail: – *All applications*: L. Paulatto, F. Mauri, and M. Lazzeri, Phys. Rev. B 87, 214303 (2013) – *Exact BTE solution*: G. Fugallo, M. Lazzeri, L. Paulatto, and F. Mauri, Phys. Rev. B 88, 045430 (2013) – *Spectral functions*: L. Paulatto, I. Errea, M. Calandra, and F. Mauri, Phys. Rev. B 91, 054304 (2015) – *Finite size effects*: L. Paulatto, D. Fournier, M. Marangolo, M. Eddrief, P. Atkinson, M. Calandra. Phys. Rev. B 101 (20), 205419 (2020)

Table of contents

- D3Q manual
- Foreword
  - Copyright
  - Citing
- Table of contents
- Capabilities of the d3q code
- Compiling the code
- Preparing a D3Q calculation
  - Running the pw.x calculation
  - Running the ph.x calculation
  - Example pw.x and ph.x input files
    - Input of pw.x
    - Input of ph.x
- D3Q Input File
  - QPOINTS or GRID specifications
  - &inputd3q namelist * mode (CHARACTER, default: “single”) * prefix (CHARACTER, no default) * outdir (CHARACTER, default $ESPRESSO_TMPDIR) * d3dir

(CHARACTER, default outdir, or $ESPRESSO_D3DIR) * fildrho_dir (CHARACTER, default outdir, or $ESPRESSO_FILDRHO_DIR) * fild3dyn (CHARACTER, default: “anh”) * ethr_ph (REAL, default: 1.d−8) * amass (REAL, array, default: same as pw.x) * fildrho (CHARACTER, default: “drho”) * first, last, only (INTEGER, default: 1, 0) * offset, step (INTEGER, default: 0, 1) * restart (LOGICAL, default: .true.) * max_seconds (no default, in seconds) * max_time (format hh.mmss) * nk1, nk2, nk3 (3x INTEGER, default: same as pw.x) * k1, k2, k3 (3x INTEGER, default: same as pw.x) * degauss (REAL, default: same as pw.x) * print_star (LOGICAL, default: .true.) * print_perm (LOGICAL, default: .false.) * print_trev (LOGICAL, default: .false. * safe_io (LOGICAL, default: .false.)

- GRID and QPOINTS
- &d3_debug namelist

- Running D3Q * Note on pools parallelisation
- Output format
- Data flow
- Bibliography

Capabilities of the d3q code

The d3q code computes the third derivative of the Density Functional Theory ground−state energy with respect to three harmonic perturbations, identified by their wavevectors q1, q2 and q3=−q1−q2. The code can use a certain number of methods: − Norm Conserving pseudopotentials − Local Density Approximation (LDA) and Generalized Gradient Approximation (GGA) functionals. − Insulators and Metals (i.e. partial occupation of the electronic bands)

On the other hand, the codes *does not* implements the following features: − Ultrasoft pseudopotentials and PAW datasets − Advanced functionals, e.g. meta−GGA, Grimme vdW corrections and non−local vdW functionals − Hybrid functionals − LDA+U, or self−interaction correction − External electric field, via Berry phase or with a sawtooth potential − spin−polarized systems − non−collinear spin and spin−orbit interaction

Please be aware of these limitations! Adding any of these features would take a huge amount of work for developing the theory, implementing the code and validating the results. There is very little chance that we will implement any of them (perhaps with the exception of spin−polarized systems) in the near future. On the other hand, if you are willing to provide a consequent amount of workforce, we are available to provide as much assistance as possible.

In most cases, we have have been able to obtain meaningful results despite these limitation, sometimes combining phonons computed with more advanced methods with 3rd order calculation computed on the same systems but with a simpler method, assuming that more advanced correction would not affect the 3rd order too much. If you want to take this approach we recommend that you validate the assumption on a simpler test, where the entire calculation can be performed with the methods available in d3q.x. Alternatively, you can use one of the compatible real−space codes.

Compiling the code

The d3q.x code is distributed in a package bundled with the thermal2 codes for computing phonon−phonon interactions. The package has to be decompressed inside the main directory of a Quantum−ESPRESSO source distribution (which is usually called espresso−X.Y.Z, where X, Y and Z identify the version). Please be aware that each release of the D3Q code is developed and tested on top of a specific version of Quantum−ESPRESSO and it will probably not work on any other version: Take care to choose the correct package.

In order to decompress the package move inside the espresso−X.Y.Z directory and issue the command tar xzvf /where/it/is/d3q−A.B.C−qeX.Y.Z.tgz then move to the D3Q directory and type "make" to compile the code. D3Q will use the same compilation parameters as Quantum−ESPRESSO, as specified in the file make.inc generated by the configure script of QE. The make command will build both the D3Q code and the thermal2 codes.

Preparing a D3Q calculation

The d3q.x code works in combination with the ph.x code from the Quantum−ESPRESSO distribution, which in turn has to be run on top of a total energy calculation performed with the pw.x code. As it is impractical to store the wavefunctions and the wavefunction perturbations, the d3q code recomputes them from the ground−state charge density (produced by pw.x) and its variation with respect to a harmonic phonon perturbation (computed by ph.x). The latter is, per default, not stored to disk, here we will see the correct workflow to prepare all the quantities required for a d3q calculation.

Running the pw.x calculation

There is no specific keyword for the pw.x calculation, just a couple of notes: − Be sure to use a very tight value for conv_thr, we recommend 1.d−9 or smaller, down to 1.d−12. It is important to start from a ground−state solution which is as close as possible to the variational minimum in order to ensure that the assumptions of the perturbation theory hold − Check the convergence of the plane−wave kinetic energy cutoff (ecutwfc) against a phonon calculation, not just the total energy. For example, you can check the convergence of the optical modes at Γ = (0,0,0), which is rather inexpensive, down to 0.1 cm−1. Phonon calculation can converge much slower than the total energy, especially when using GGA functionals.

Running the ph.x calculation

Per default, the ph.x code does not save the variation of the charge density (δn) to a file. In order to save it you can use the fildrho="filename" keyword in the &inputph namelist of phonon, which will store δn to filename in the outdir directory. This can be sufficient, but note that it cannot be used when doing a phonon dispersion calculation, as the δn of different q−points would overwrite each other. Furthermore, if you are doing a phonon dispersion calculation on a regular grid, only the points in the irreducible wedge of the Brillouin zone will be computed explicitly, while the others are obtained using symmetry. However, when d3q.x computes a regular grid of (q1, q2, q3) triplets only q1 can be in the irreducible wedge: q2 will span all the grid and q3 may not even be in the grid at all! In order to solve these two problems, you have to add the drho_star keywords to the &inputph namelist , with the following syntax:

```
drho_star%open = .true.
drho_star%dir  = "directory_name"
drho_star%ext  = "file_extension"
```

Which will use symmetry to rotate δn from the irreducible wedge to all the q−points in the regular grid; δn will be saved to a different file for each q−point, in the directory drho_star%dir. We refer you to the phonon documentation for mode details about these variables.

All this mechanism is performed automatically by the code: you do not need to worry about the details. However, if you are interested, the file names will be prefix.drho_star%ext.qpoint1. Where prefix and drho_star%ext are the values of the corresponding variables the input of ph.x, and qpoint1 is an automatically generated extension which depend on the fractional coordinates of the q−point. In addition to each δn file, there will be a file with the ".pat" extension which contains the phonon polarization vector (also known as phonon pattern or phonon eigenvector). Furthermore, the code will generate an index file called prefix.drho_star%ext.dfile_dir which contains

a list of the q−points; this index will be used by d3q.x to find the δn it needs: if you are moving these files, do not forget to copy it along with the rest.

Example pw.x and ph.x input files

These are two minimalist input files for pw.x and ph.x for a crystal of Silicon, which will allow you to proceed with a d3q.x calculation.

Input of pw.x

```
 &control
    calculation='scf'
    prefix='silicon',
    pseudo_dir = '../',
    outdir='/tmp'
 /
 &system
    ibrav=  2,
    celldm(1) = 10.20
    nat=  2,
    ntyp= 1,
    ecutwfc = 24
 /
 &electrons
    conv_thr =  1.0d-12
 /
ATOMIC_SPECIES
 Si  0.0  Si.pz-rrkj.UPF
ATOMIC_POSITIONS
 Si 0.00 0.00 0.00
 Si 0.25 0.25 0.25
K_POINTS AUTOMATIC
 4 4 4  0 0 0
```

Input of ph.x

```
phonon_disp_of_silicon
 &inputph
   outdir='/tmp',
   prefix='silicon',
   ldisp=.true.
   nq1=4,   nq2=4,   nq3=4,
   tr2_ph=1.0d-16,
    drho_star%open = .true.
    drho_star%ext  = 'drho',
    drho_star%dir  = './FILDRHO'
 /
```

D3Q Input File

The d3q.x code reads its input from a file, and a few environment variables. If a parameter is specified both in the input file and in a variable, it is the value in the file that is used. The input file has the following structure:

```
title_line
&inputd3q
   ...
/
```

QPOINTS or GRID specifications

[debug instructions] The title line can be any title you wish to use for your calculation, it will be printed in output but it has no effect on the results. After the title line, you have to specify the &inputd3q namelist, which contain all the parameters of the calculation. After the namelist, and depending on the value of the parameters therein, you will have to specify either a series of q−points or the dimension of the regular grid that you want to compute.

&inputd3q namelist

The namelist start with the "&inputd3q" keyword and ends with a "/" in an otherwise empty line. It can contain the following variables:

mode (CHARACTER, default: "single")

This variable specify the mode in which d3q will operate, it can take the following values: − "single": compute the 3rd order dynamical matrix for a single triplet (q1, q2, q3) of q−points. You will have to specify q1 and q2 points, on two separate lines, after the namelist (units 2π/alat); q3 is just −(q1+q2). − "gamma−only": compute the D3 matrix of (0, 0, 0) − "gamma−q": compute the matrix at (0,−q,q), like the old d3.x code; you have to specify the point after the namelist (units of 2π/alat) − "partial": compute a full grid of triplets around a specific point q0: (q0, q0+q, q0−q) with q = k1 b1/nq1 + k2 b2/nq2 + k3 b3/nq3 with ki = 0, 1, ... , nqi. After the namelist, you will have to specify the point (units of 2π/alat) and then, on a new line, the size of the grid nq1 nq2 nq3. This method is in principle useful, as to compute the scattering matrix of phonon q0 you only need this set of D3 matrices, which can be especially useful for limiting the computational cost . However, using these matrices requires a specialized code, which is not implemented: the thermal2 suite of codes expects a full grid. If you badly need to use this feature, please contact us. − "full": compute a full grid of triplets (q1, q2, q3), where q1 and q2 span the grid and q3 =−(q1+q2). Symmetry is aggressively used to reduce the total number of triplets to compute, which makes it difficult to estimate; it is usually smaller than the square of the number of points in the irreducible wedge of the BZ, i.e. if the phonon calculation included N q−points, the D3 calculation will probably include less than N2 triplets.

If you want to use the D3 matrices to do linewidth or thermal transport calculations with the thermal2 codes, you definitely have to use the "full" mode, although the other can be useful for testing convergence or in other special cases.

prefix (CHARACTER, no default)

Prefix must be the same as in the input of pw.x and ph.x

outdir (CHARACTER, default $ESPRESSO_TMPDIR)

Outdir must be the same as in the input of pw.x and ph.x. Its default value is read from the ESPRESSO_TMPDIR environment variable, the current directory is used if the variable is not set.

d3dir (CHARACTER, default outdir, or $ESPRESSO_D3DIR)

If you wish to write the d3 temporary data to a different place than the pw data, you can use d3dir. Its default value is read from the ESPRESSO_D3DIR environment variable, and will eventually fallback to the value of outdir. A further sub−directory will be created which depends on the

coordinates of the q−points, it is hence safe to use the same value of d3dir even when running several d3q.x calculations at the same time and with the same prefix, as long as they work on different triplets. Data in d3dir is normally deleted after each triplet is computed. If for some reason you need this data, you will have to change d3q.f90 to call d3_reset with parameter cleanup = .false., and recompile the code.

fildrho_dir (CHARACTER, default outdir, or $ESPRESSO_FILDRHO_DIR)

The directory where the files with the variation of the charge density have been stored. For the code to work, this must be the same as the value of drho_star%dir in the input of ph.x. Note that only files inside d3dir will be opened read/write, files inside outdir and fildrho_dir will be opened read−only, which ensure that even in case of a catastrophic crash you should not need to repeat the pw.x and ph.x calculations.

fild3dyn (CHARACTER, default: "anh")

The prefix of the files containing the D3 matrices. Note that the d3q.x code will create a lot of output files (one for each of the explicitly compute triplets, and one for each triplet obtained by symmetry); we recommend that you insert a directory name in this variable to keep your working directory tidy. E.g. you can use fild3dyn="OUTPUT/anh", the "OUTPUT" directory will be created automatically.

ethr_ph (REAL, default: 1.d−8)

The threshold for solving the non−self consistent Sternheimer equation. The default value should work for everyone. Please note that this threshold is not the same as the variable tr2_ph from phonon input. It is related to the threshold of the conjugate−gradient solver of the Sternheimer equation; it is printed in the output of phonon, i.e. the value of "thresh" marked in bold in the following snippet from ph.x:

```
Representation #  5 mode #   5

Self-consistent Calculation
    ...
 iter #   7 total cpu time : 4.0 secs   av.it.:   7.4
 thresh= 1.076E-08 alpha_mix =  0.700 |ddv_scf|^2 =  4.004E-17
```

You can check the last iteration of your phonon calculation and use a similar value for your d3 input.

amass (REAL, array, default: same as pw.x)

With amass you change the mass of atomic species. You can change it again later if you wish, no part of the DFPT calculation depends on ionic mass. The default value is read from the pw.x restart file.

fildrho (CHARACTER, default: "drho")

fildrho must be the same as drho_star%ext in ph.x. When this variable is used, the d3q.x code will automatically search the file it needs inside the fildrho_dir directory. When computing a grid of triplets calculation this is the only possible way to supply the fildrho files.

Alternatively, when computing single triplets, you can use fild1rho, fild2rho and fild3rho to specify the rho variation at q point 1, 2 and 3. In special cases, i.e. q2=−q3 and q2=q3, you only need to specify fild1rho and fild2rho. In some other case, i.e. q1=−q2 and q1=q2, you only need fild1rho and fild3rho. In the Gamma−only case only fild1rho file is used.

first, last, only (INTEGER, default: 1, 0)

If you only want to compute a subset of the q−point you can use these two variables. The d3q.x code will then compute all the triplets between first and last. If you set last to zero, all the triplet starting from first will be computed. Setting "only=x" is a shortcut for "first=x" and "last=x".

offset, step (INTEGER, default: 0, 1)

Another way to select a subset of the triplets is to use the offset and step variables. In this case the code will compute one triplet every step, starting from 1+offset. Note that offset must be between 0 and step−1. You can use first and last together with offset and step, but note that first and last will always refer to the global triplet number.

restart (LOGICAL, default: .true.)

When restart is true, the code will check existing fild3dyn files and skip the calculation of any triplet that has already been computed. This option can also be used to refine a grid calculation, e.g. if you have already computed a 2x2x2 grid and want to compute a 4x4x4 one, simply restart with the finer grid and the same input parameters, and the triplets from the coarser grid will be skipped automatically. Restart of partially−computed q−points is currently implemented but disabled because 1) it is still unreliable 2) it requires that all input/output is safely written to disk, which makes the calculation much slower. You can activate it modifying the code at your own risk!

max_seconds (no default, in seconds)

max_time (format hh.mmss)

The maximum running time after which the code will stop, you can only set one of the two. The code will only check this condition after finishing to compute each triplet.

nk1, nk2, nk3 (3x INTEGER, default: same as pw.x)

k1, k2, k3 (3x INTEGER, default: same as pw.x)

degauss (REAL, default: same as pw.x)

You can change the k−points grid and smearing used in d3q, do it with care and only if you understand what you are doing, or for testing purpose. nkX is the grid size along direction X, kX can be 1 or 0, it means shift or no shift. degauss is the smearing in Ry.

print_star (LOGICAL, default: .true.)

print_perm (LOGICAL, default: .false.)

print_trev (LOGICAL, default: .false.

The code saves the D3 matrix of each triplet computed and of the star of those triplet. It does not compute the permutations (q1,q2,q3) −> (q2,q3,q1); etc... (although some symmetry operations may be equivalent to permutations) It also does not use time−reversal symmetry to send q−>−q. You can control this behavior with these variables. Please note that the subsequent codes in the thermal2 require print_star=.true. in order to work correctly.

safe_io (LOGICAL, default: .false.)

If you set safe_io to true, every file is closed and reopened after each write. This can make the code much slower, but can solve some problems with parallel filesystem not syncing or corrupted