Third order DFPT via "$2n+1$", ph-ph interaction and thermal transport

Lorenzo Paulatto

IMPMC - Université Pierre et Marie Curie, Paris 6/CNRS - Paris (France)

lorenzo.paulatto@impmc.upmc.fr

January 18, 2016

Overview

Introduction

This work is divided in two parts, one that overlap strongly with the DFPT phonon code, another that is related to Phonon post-processing tools (q2r, matdyn, qha, ...)

1. The $2n + 1$ theorem and the D3 code
   - What does it do?
   - How is it done?
   - What can be backported into phonon?
   - What would I like from upstream?
2. The ph-ph interaction and thermal transport codes
   - What do they do?
   - Should it go back to phonon?
   - What do they do?

The $2n + 1$ theorem and the D3 code

Theorem (The $2n + 1$ theorem)

The $(2n + 1)$th derivative of the eigenenergy of a Hamiltonian may be determined with only a knowledge of the change in the eigenfunctions up to order n.

Kohn-Sham formulation: Gonze and Vigneron 1989, Gonze 1995

What does it do?

The D3 code coputes the third derivative of the total energy with respect to three harmonic perturbations using this theorem.

How is it done/1: workflow

Step 1

We do a normal self-consisted calculation with `pw.x`

Step 2

We do a phonon dispersion calculation on a regular symmetry-reduced grid with `ph.x`; the variantion of charge density $n^1(\mathbf{q})$ is rotated on the star of $\mathbf{q}$ and stored to files (a few 100MB)

Step 3

- We build a grid of $\mathbf{q}_1$, $\mathbf{q}_2$, $\mathbf{q}_3 = -\mathbf{q}_1 - \mathbf{q}_2$ reduce it with symmetry and permutations
- For every triplet, compute the D^3 matrix: $\partial^3 E / \partial u_i(\mathbf{q}_1) \partial u_j(\mathbf{q}_2) \partial u_k(\mathbf{q}_3)$

How is it done/1b: comments

Why do we store only $n^1(\mathbf{q})$ from `ph.x`?

- the perturbation of the wavefunctions $\psi^1(\mathbf{q})$ take orders of magnitude more disk space
- we would still need to transform $\mathbf{k} \rightarrow \mathbf{k} + \mathbf{G}$ (annoying in parallel)
- there is a subtle difference in the case of metals

Why don't you recompute $n^1(\mathbf{q})$/do this inside phonon?

- it is several orders of magnitude more complicated
- it is probably slower

How is it done/2: inside d3

For every triplet $\mathbf{q}_1$, $\mathbf{q}_2$, $\mathbf{q}_3$:

- Restart from file, do an nscf calculation for every $\mathbf{k}$, $\mathbf{k}+\mathbf{q}_x$, $\mathbf{k}-\mathbf{q}_x$, x=1,2,3

$\Rightarrow$ code detects special cases to reduce the grid ($\mathbf{q}=\Gamma$, $\mathbf{q}_x=\mathbf{q}_y$)

$\Rightarrow$ $\mathbf{k}+\mathbf{q}=\mathbf{k}'+\mathbf{G}$ not implemented (too complex)

- Find the files containing $n^1(\mathbf{q}_x)+\mathbf{G}$, read it and eventually apply a phase
- Solve 12 times the non-self consistently the Sternheimer equation:
  $(H_{scf}+\alpha P_v-\epsilon_n)\Delta^{\mathbf{q}}\psi_n(\mathbf{k})=P_c\Delta^{\mathbf{q}}V\psi_n\mathbf{k}$

$\Rightarrow$ $\Delta^{\mathbf{q}}\psi_n(\mathbf{k}-\mathbf{q})$, $\Delta^{-\mathbf{q}}\psi_n(\mathbf{k}+\mathbf{q})$, $\Delta^{\mathbf{q}}\psi_n(\mathbf{k})$, $\Delta^{-\mathbf{q}}\psi_n(\mathbf{k})$

How is it done/2: inside d3

For every triplet $\mathbf{q}_1$, $\mathbf{q}_2$, $\mathbf{q}_3$:

$\Rightarrow$ $\Delta^{\mathbf{q}}\psi_n(\mathbf{k}-\mathbf{q})$, $\Delta^{-\mathbf{q}}\psi_n(\mathbf{k}+\mathbf{q})$, $\Delta^{\mathbf{q}}\psi_n(\mathbf{k})$, $\Delta^{-\mathbf{q}}\psi_n(\mathbf{k})$

- $\left\langle \Delta^{-\mathbf{q}_1}\psi_{\mathbf{k}+\mathbf{q}_1} \left| \Delta^{-\mathbf{q}_2} V \right| \psi_{\mathbf{k}+\mathbf{q}_2} \right\rangle$
- $\left\langle \Delta^{-\mathbf{q}_3}\psi_{\mathbf{k}} \left| \Delta^{\mathbf{q}_2} V \right| \delta^{\mathbf{q}_1}\psi_{\mathbf{k}} \right\rangle$
- $\left\langle \delta^{-\mathbf{q}_3}\psi_{\mathbf{k}+\mathbf{q}_3} \left| \delta^{\mathbf{q}_1}\psi_{\mathbf{k}-\mathbf{q}_1} \right\rangle \left\langle \psi_{\mathbf{k}-\mathbf{q}_1} \right| \Delta^{\mathbf{q}_2} V \left| \psi_{\mathbf{k}+\mathbf{q}_3} \right\rangle \right.$
- $\left\langle \Delta^{-\mathbf{q}_1}\psi_{\mathbf{k}} \left| \Delta^{\mathbf{q}_2,\mathbf{q}_3} V \right| \psi_{\mathbf{k}} \right\rangle$
- $\left\langle \psi_{\mathbf{k}} \left| \Delta^{\mathbf{q}_1,\mathbf{q}_2,\mathbf{q}_3} V \right| \psi_{\mathbf{k}} \right\rangle$
- Ewald (Ion-Ion) term
- terms depending on the Fermi energy
- derivative of the functional $\delta E_{xc}/\delta n^1(\mathbf{q}_1)\delta n^1(\mathbf{q}_2)\delta n^1(\mathbf{q}_3)$

What do we use from `ph.x`

What we use

- `cg_solve_all` is used as-is, with custom (aka *ancient*) `ch_psi_all` and `cg_psi`; it uses the same `h_psi` as phonon
- custom version of `solve_linter` (non-scf, partial list of **k**-points)
- simplified `dq_vscf`

Can we do bettter?

- For some subroutines it is just a matter of testing: `cg_psi`
- For some a bit more work is required `dv_of_drho` (but I do not want to make the original subroutine a mess)
- For others the additional complexity outside the subroutine would hinder the simplification (es. `ch_psi_all`)

Why we do not use more subroutines from phonon.

example: `cg_psi`

```
IF(tddfpt) THEN
   ikq = ik
   evq => evc
ELSE
   ikq = ikqs(ik)
ENDIF
```

- We would love to use `dv_of_vdrho` & company but they got way too complicated: too many global variables, used very deeply (i.e. **q**-point, n^1, `evc`/`evq`) makes the outcome practically unpredictable.
- One especially offending variable: ikqs

Is there a tradeoff between having two similar subroutines or one subroutine that nobody can read?

No: but you will have to use subroutine arguments, instead of global variables. This requires writing a few more lines.

How to fix it

Why is it difficult

I can easily generalize the subroutines in `ph.x` and my code, but will it break everyone else code? Will everybody hate me?

A possible workaround

```
subroutine old_style()
  use module, only : variable
  call new_style(variable)
end subroutine

subroutine new_style(variable)
  kind :: variable
  ! old body is actually here
end subroutine
```

I would rather get rid of `old_style` immediately

What can be backported into phonon?

Already backported:

- The subroutines to rotate n^1 (or V^1) and write it to a **q**-defined file

To do:

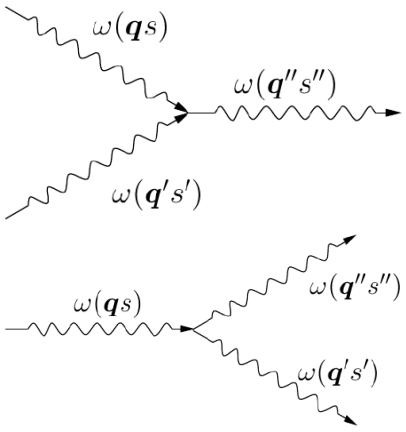
- Use a different scratch directory per **q**-point, no name conflicts
- Open all `pw.x` files read-only to prevent accidental corruption of scf data
- Keep track of all opened files

$\Rightarrow$ Difference instances of the code can work on the same prefix/outdir as long as they have difference **q**-points

What would I like from upstream?

- The possibility to do a read-only restart
- Read a wafunction by **k**-point, instead of **k**-point number (*analogous* to what I do in d3 for **q**-points)
- more "bottom-up" and self-contained basic subroutines, especially
- symmetry
- derived types that contain big chuncks of the global variables (i.e. lattice+basis+symmetry, k-points and g vectors,...)

Phonon-Phonon interaction



What do they do

Integrate the ph-ph interaction equation: footnotesize

$$\gamma_{\mathbf{q}j}(T) = 1/\tau_{\mathbf{q}j}(T) = \frac{\pi}{\hbar^2 N_q} \sum_{\mathbf{q}',j',j''} \left| V^{(3)}_{\mathbf{q}j,\mathbf{q}'j',\mathbf{q}''j''} \right|^2 \times \Big[(1 + n_{\mathbf{q}'j'} + n_{\mathbf{q}''j''}))\delta(\omega_{\mathbf{q}j} - \omega_{\mathbf{q}'j'} - \omega_{\mathbf{q}''j''}) + 2(n_{\mathbf{q}'j'} - n_{\mathbf{q}''j''})\delta(\omega_{\mathbf{q}j} + \omega_{\mathbf{q}'j'} - \omega_{\mathbf{q}''j''}) \Big]$$

How is it done

D^3 and D^2 matrices need to be Fourier-interpolated on a fine double-grid of $\mathbf{q}$-points.

Thermal transport (SMA)

$$\kappa_{\mathrm{L}}^{\alpha,\beta} = \frac{\hbar^2}{N_q \Omega K_{\mathrm{B}} T^2} \sum_{\mathbf{q}j} c_{\mathbf{q}j}^{\alpha} c_{\mathbf{q}j}^{\beta} \omega_{\mathbf{q}j}^2 n_{\mathbf{q}j}(n_{\mathbf{q}j}+1)\tau_{\mathbf{q}j}$$

- We have two nested sums over double q-point grid
- We have libraries to quiclky and reliably compute phonon frequencies, group velocities, occupations...
- Everything has to be parallelised with MPI

What do they do?

- A set of small specialized codes and libraries that:

(1) Take care of complicate input/output, initial setup, sum rules, etc.

(2) Perform one and only one core function (Fourier interpolate, sum integral, BE occupation, ...)

- In total 13k lines, around 3.9k are rewrites of q2r and matdyn
- q2r+matdyn is 3.4k lines

$\Rightarrow$ but the "matdyn" equivalent code is only about 100 lines

Effective charges are still not implemented.

Should I put them back in phonon?

pros

- More people may use them
- Some people may find bugs and complain
- Some bug may even get fixed
- Profit

pros

- I may not like some fixes

$\Rightarrow$ Will my code rot?

$\Rightarrow$ Will it rot faster if I keep a tight grip on it?

The End