Guido Petretto, Matteo Giantomassi, David Waroquiers, Xavier Gonze, Geoffroy Hautier, Gian-Marco Rignanese

NAPS, Institute of Condensed Matter and Nanosciences, Université catholique de Louvain, 1348 Louvain-la-Neuve, Belgium

09/05/2017
Outline

1. Introduction
2. High-throughput environment
3. High-throughput DFPT
4. Conclusions
High-Throughput databases

Diffusion of large databases based on DFT calculations

Materials Project

AFlow

Materials Cloud

Open Materials Database

Energy materials
Big data methodologies
Supercomputers
High-Throughput computing
Density-functional Theory

Materials design
High-throughput frameworks

- Many high-throughput frameworks
  - Mostly focused on ground state calculations
  - No native support for abinit 😞

- What do we need for high-throughput (DFPT)?
  - Input generation
    - Generic input parameters for DFPT
  - Managing complicated workflows
    - Beyond relaxations or band-structures
  - Error handling
  - Insertion in the database
  - Analyze data
1. Introduction

2. High-throughput environment

3. High-throughput DFPT

4. Conclusions
Input generation

Handling and generating abinit inputs based on few input parameters:

- **Input objects** (import abipy.abio.inputs):
  - class AbinitInput
  - class AnaddbInput

- **Factory functions** (import abipy.abio.factories)

**Relax inputs:**

```python
ion_ioncell_relax_input(structure, pseudos, accuracy="normal",
                       kppa=1000, shift_mode='Monkhorst-pack',
                       smearing="fermi_dirac:0.1 eV",
                       spin_mode="polarized")
```
Input generation

Phonon perturbations inputs, parallelized over the perturbations:

```python
phonons_from_gsinput(gs_inp, ph_ngqpt=[6,6,6],
                     with_ddk=True, with_dde=True)
```

- List of `AbinitInput` objects
  - phonon perturbations
  - DDK
  - DDE
  - NSCF to get WFQ if needed

- Calls `abinit` to determine the list of irreducible perturbations.

- Tags to easily sort out the type of input
Managing workflows

- Many frameworks available
- Scales are important
  - How many systems? (Tens? Hundreds? Thousands?)
Managing workflows

- Many frameworks available
- Scales are important
  \[\Rightarrow\text{ How many systems? (Tens? Hundreds? Thousands?)}\]

![FireWorks](image-url)
Managing workflows

- Many frameworks available
- Scales are important
  ⇒ How many systems? (Tens? Hundreds? Thousands?)

- General purpose workflow manager
- Database backend (MongoDB)
- Support for dynamic workflows
- Support for several queueing systems such as PBS, SLURM…
- Web gui monitor
Managing workflows - Fireworks

- **Firetask**: The basic object defining an action
  
  ```python
  class HelloWorldTask(FiretaskBase):
      def run_task(self, fw_spec):
          print("Hello world!")
  ```

- **Firework**: A group of sequential tasks. 1 ↔ 1 with a job on a cluster

- **Workflow**: A collection of fireworks with dependencies
Implementing workflows - Abiflows

Managing workflows - Abiflows

Implementation of AbinitFireTask to handle several types of calculations.

Some of the available classes:

- GsTask
  - ScfTask
  - NscfTask
  - RelaxTask
- AbiFireTask
  - PhononTask
  - DdeTask
  - StrainTask
- DfptTask
  - AnaddbTask
  - FiretaskBase
Abiflows relies on Abipy, Abinit and pymatgen providing

- Tasks object to handle different types of calculations
- Automatized choice of number of cores (autoparal)
- Handling of the dependencies (wavefunction, densities, DDB, . . . )
- Workflows generators for common cases
  - Relaxation
  - Band structures
  - Phonons
  - . . .
- Error handling
- Templates for database insertion
Error handling

Firework

ScfTask

ScfConvergenceWarning!

Firework

NscfTask
Error handling

- **Firework**
  - **ScfTask**
    - `ScfConvergenceWarning!`
    - Link DEN and WFK restart
  - **Detour**
  - **Firework**
    - **ScfTask**
  - **Firework**
    - **NscfTask**
Error handling

Firework

ScfTask

ScfConvergenceWarning!
Link DEN and WFK restart

Detour

Firework

NscfTask

Firework

RelaxTask

Relax ions

Firework

RelaxTask

Relax ions and cell
DilatmxError!
Error handling

- **Firework**
  - **ScfTask**
    - `ScfConvergenceWarning!`
    - Link DEN and WFK restart
  - **Detour**
- **Firework**
  - **NscfTask**

- **Firework**
  - **RelaxTask**
    - **Relax ions**
    - DilatmxError!
    - Read structure from netcdf
    - Increase dilatmx
  - **Detour**
Database insertion

Document-Object Mapper (like ORM, but for document databases)

- Mixins for standard quantities
- Document associated with workflows

⇒ Standardized output

`abiflows.database.mongoengine` modules:

```python
class MaterialMixin(object):
    """
    fields describing the material examined in the calculation
    """

class AbinitPseudoData(EmbeddedDocument):
    """
    fields and function to save abinit pseudopotential data
    """

class PhononResult(MaterialMixin, DateMixin, DirectoryMixin, Document):
    """
    results for a phonon workflow
    """
```
Outline

1. Introduction
2. High-throughput environment
3. High-throughput DFPT
4. Conclusions
Relax workflow

Split the set of calculations as much as possible. First relax, then phonons.

- Less complicated workflows
- Easier error recovery
- Save intermediate results
Phonon workflow

DDK 1 -> Dde 1
DDK 2 -> Dde 2
DDK 3 -> Dde 3

SCF

NSCF WFQ

Phonon 1

mrgdddb

Anaddb

Insert DB

cleanup

Autoparal

Phonon 2

Phonon 3

Phonon N

G. Petretto (NAPS, UCL)
High-throughput DFPT
09/05/2017 17
Find optimal K-points and Q-points sampling for high-throughput

Set of 48 semiconductors. Calculations for several K and Q grids
Various sizes, crystal symmetries, gaps

\[ \sim 800 \text{ workflows run (relax + phonons)} \]

<table>
<thead>
<tr>
<th>NaLi2Sb</th>
<th>Ca(CdP)2</th>
<th>CdS</th>
<th>SrLiP</th>
<th>InS</th>
<th>GaN</th>
<th>RbYO2</th>
</tr>
</thead>
<tbody>
<tr>
<td>SiO2</td>
<td>BP</td>
<td>AlSb</td>
<td>LiZnP</td>
<td>MgCO3</td>
<td>ScF3</td>
<td>ZnGeN2</td>
</tr>
<tr>
<td>LiMgAs</td>
<td>P2Ir</td>
<td>Si</td>
<td>Li3Sb</td>
<td>K2O</td>
<td>Ga3Os</td>
<td>Be3P2</td>
</tr>
<tr>
<td>ZnSe</td>
<td>MgO</td>
<td>AgCl</td>
<td>SiC</td>
<td>YWN3</td>
<td>SrO</td>
<td>PbF2</td>
</tr>
<tr>
<td>MgSiP2</td>
<td>SiO2</td>
<td>GaP</td>
<td>Be2C</td>
<td>SnHgF6</td>
<td>MgMoN2</td>
<td>ZnO</td>
</tr>
<tr>
<td>ZrSiO4</td>
<td>Ba(MgP)2</td>
<td>Ba(MgAs)2</td>
<td>Ca(MgAs)2</td>
<td>C</td>
<td>Rbl</td>
<td>FeS2</td>
</tr>
</tbody>
</table>
Convergence study - results

- 1500 points per reciprocal atom $\Rightarrow N_{kpt} \cdot N_{\text{atoms}} \lesssim 1500$
- Better using a Q-grid commensurable with K-grid
- Improve weights for Fourier interpolation (set $\text{new\_wght}=1$ in m_ifc.F90)
Using the whole framework in production

- Screening for thermoelectric materials
  ⇒ 270 phonon band structures

- Set of small semiconducting systems
  - Running at NERSC computing center in the framework of the Materials Project
  ⇒ ~ 600 phonon band structures (and growing)
r = PhononResult.objects(mp_id='mp-8062')[0] # find result SiC
with r.abinit_output.ddb.abiopen() as ddb: # download ddb and open
    phbst, phdos = ddb.anaget_phbst_and_phdos_files(ngqpt=[8, 8, 5],
                                                    lo_to_splitting=True, asr=1) # run anaddd
    phb = phbst.phbands # get PhononBands object

phb.plot_with_phdos(phdos.phdos, units='cm-1')
Data analysis - plotting results

```python
r = PhononResult.objects(mp_id='mp-8062')[0] # find result SiC
with r.abinit_output.ddb.abiopen() as ddb: # download ddb and open
    phbst, phdos = ddb.anaget_phbst_and_phdos_files(ngqpt=[8,8,5],
        lo_to_splitting=True, asr=1) # run anaddb
    phb = phbst.phbands # get PhononBands object

    phb.plot_with_phdos(phdos.phdos, units='cm-1')
```

![Plot of phonon bands](image)
Data analysis - plotting results

```python
phb.plot_fatbands(units='eV')
```
Data analysis - plotting results

```python
phb.plot_fatbands(units='eV')
```
phb.plot_colored_matched(units='eV')
phb.plot_colored_matched(units='eV')
1. Introduction

2. High-throughput environment

3. High-throughput DFPT

4. Conclusions
Conclusion

- Generic high-throughput framework for abinit
  - Automatic input generation
  - Fireworks based framework
  - Error handling
  - Database insertion templates
  - Post-processing tools
- Convergence study for phonons
- Production environment
Thank you for your attention