Automatize a DFT code: high-throughput workflows for Abinit

Guido Petretto

MODL, Institute of Condensed Matter and Nanosciences, Université catholique de Louvain, 1348 Louvain-la-neuve, Belgium

24/05/2019

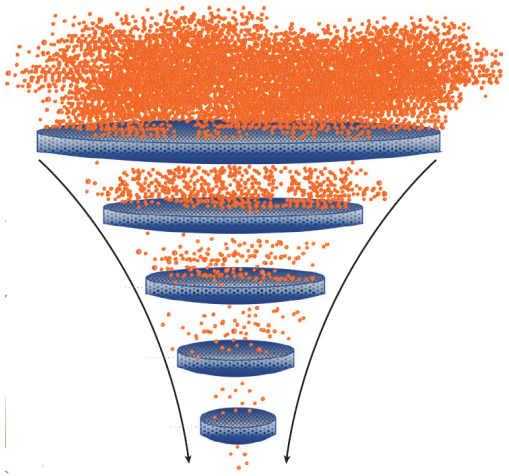
Aim of the talk

High-throughput workflows in DFT

Aim of the talk

High-throughput workflows in DFT

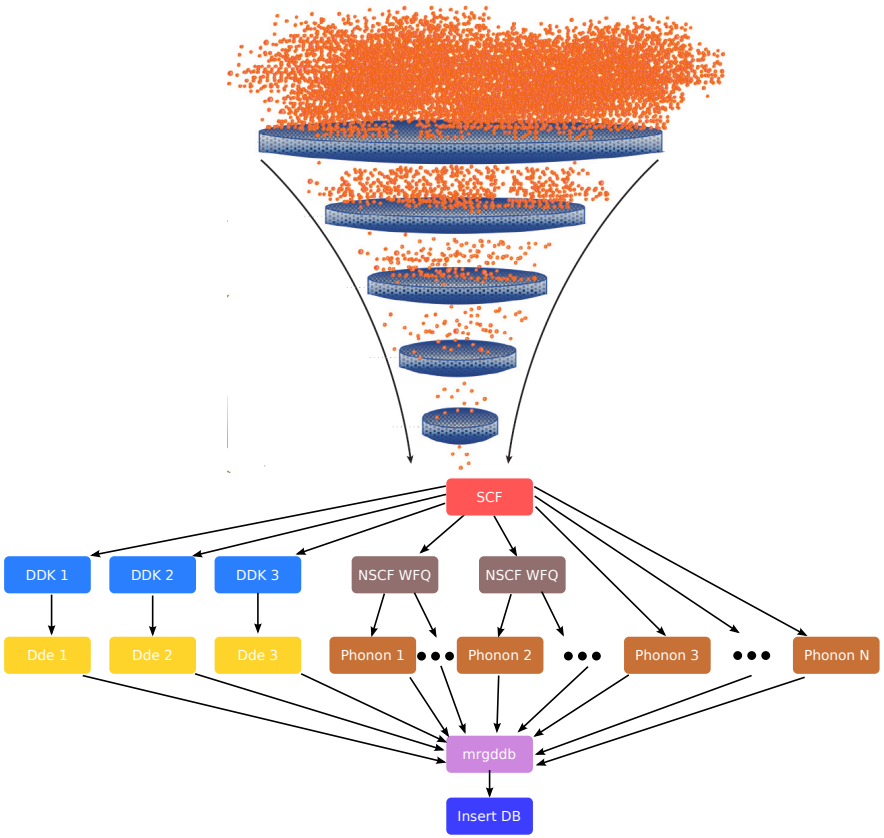
- **Why?**
  - Screening based on some property
  - Create large databases
  - Indetifying trends



Aim of the talk

High-throughput workflows in DFT

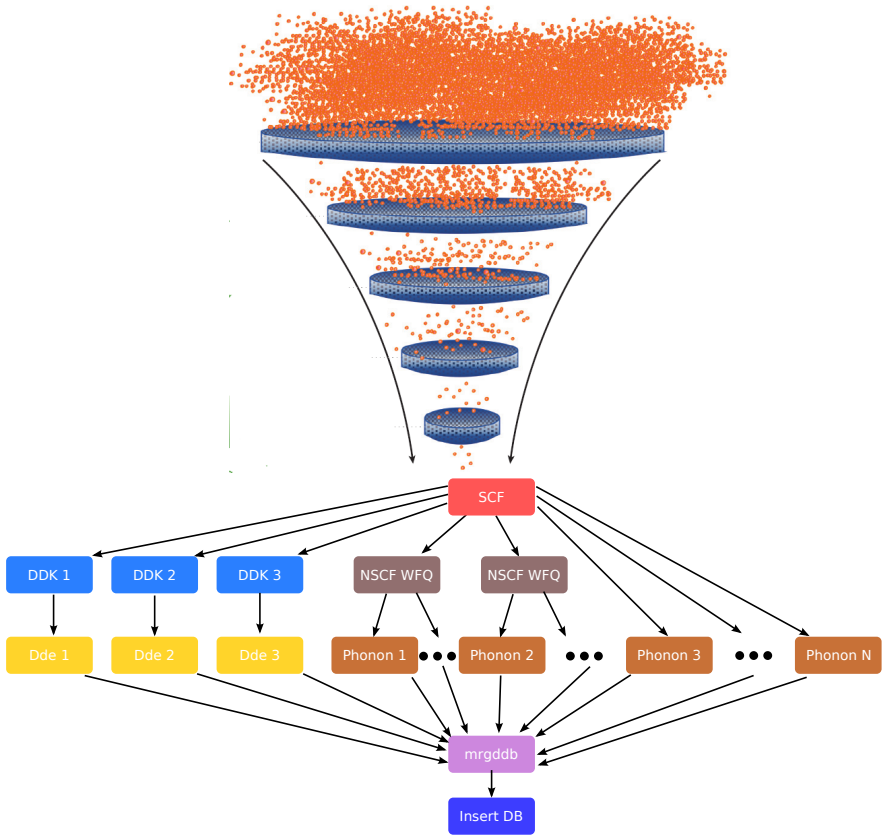
- **Why?**
  - Screening based on some property
  - Create large databases
  - Indetifying trends
- **What?**
  - Automation
  - Workflows
  - Reproducibility



Aim of the talk

High-throughput workflows in DFT

- **Why?**
  - Screening based on some property
  - Create large databases
  - Indetifying trends
- **What?**
  - Automation
  - Workflows
  - Reproducibility
- **How?**
  - What is needed?
  - Implementation
  - Challanges



Abinit workflows: goal and time frame

Main development at UCLouvain

Goals:

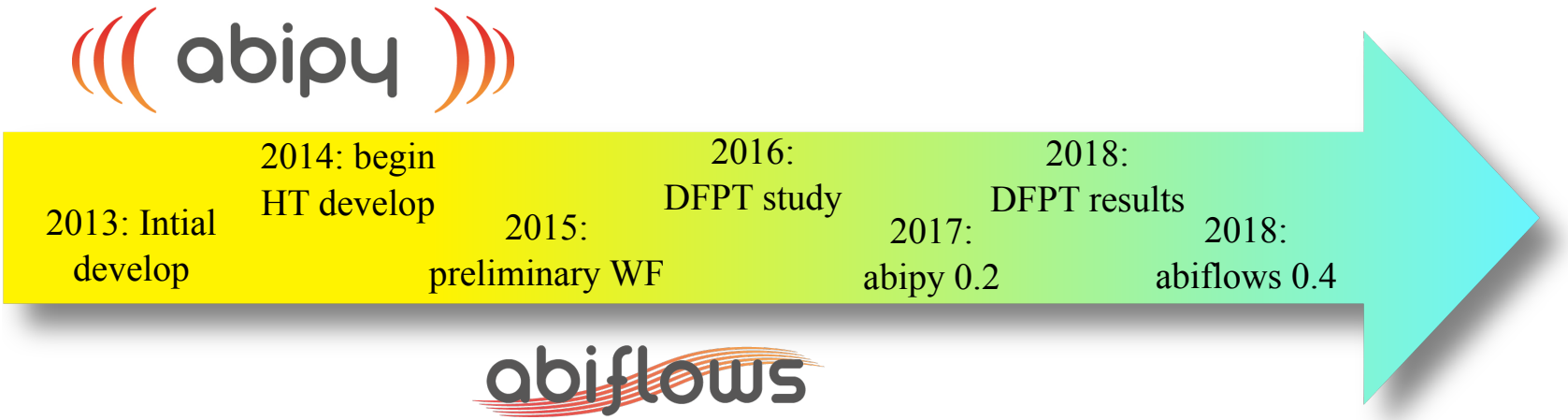
- Customizable high-throughput workflows
- Support for advanced calculations: DFPT
- Phonons for Materials project

Abinit workflows: goal and time frame

Main development at UCLouvain

Goals:

- Customizable high-throughput workflows
- Support for advanced calculations: DFPT
- Phonons for Materials project



Outline

Outline

The road to a high-throughput framework

What do we need to run high-throughput workflows?

- DFT code
- Inputs
- Error handling
- Database interface
- Workflow management
- Data analysis

The road to a high-throughput framework

What do we need to run high-throughput workflows?

- DFT code
- Inputs
- Error handling
- Database interface
- Workflow management
- Data analysis

Use existing frameworks when possible

Abinit

- Density-functional theory (DFT), DFPT, GW, DMFT
- Open source
- NetCDF output
  - array-oriented format for scientific data
- Strong contributions from Louvain-la-neuve

$\Rightarrow$ Connections between the Fortan and Python codes.

The road to a high-throughput framework

What do we need to run high-throughput workflows?

- ✔ DFT code 

  - Inputs
  - Error handling
  - Database interface
  - Workflow management
  - Data analysis

Input: crystal definition

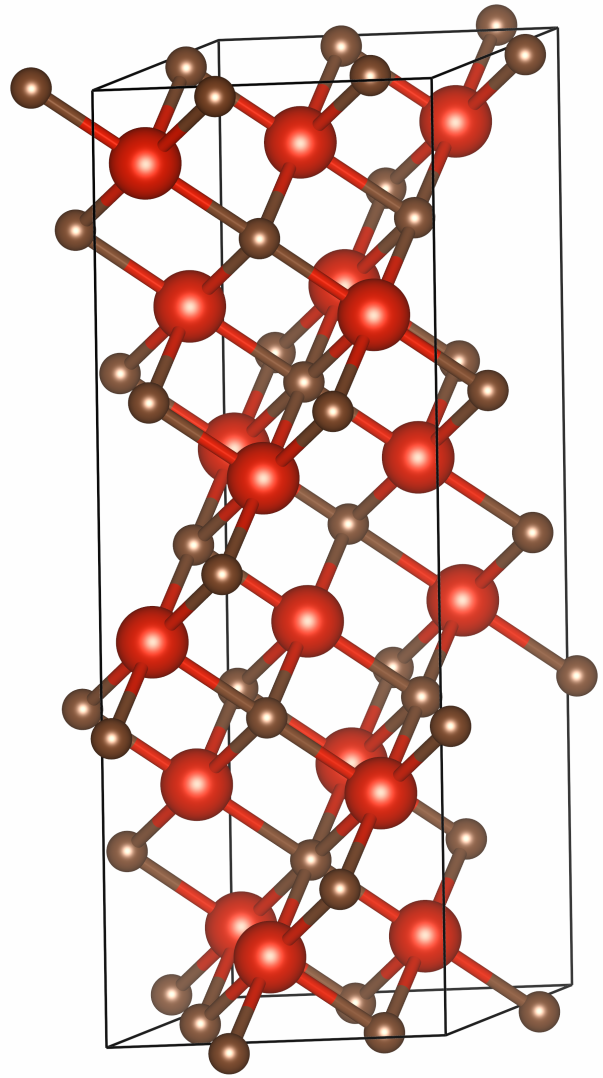
Interface to define and modify a crystal:

pymatgen

`Structure` object

```
s = Structure.from_file("Si.xyz")
s.to_abivars()
```

Easier interaction with all the tools defined in pymatgen



Input: pseudopotentials

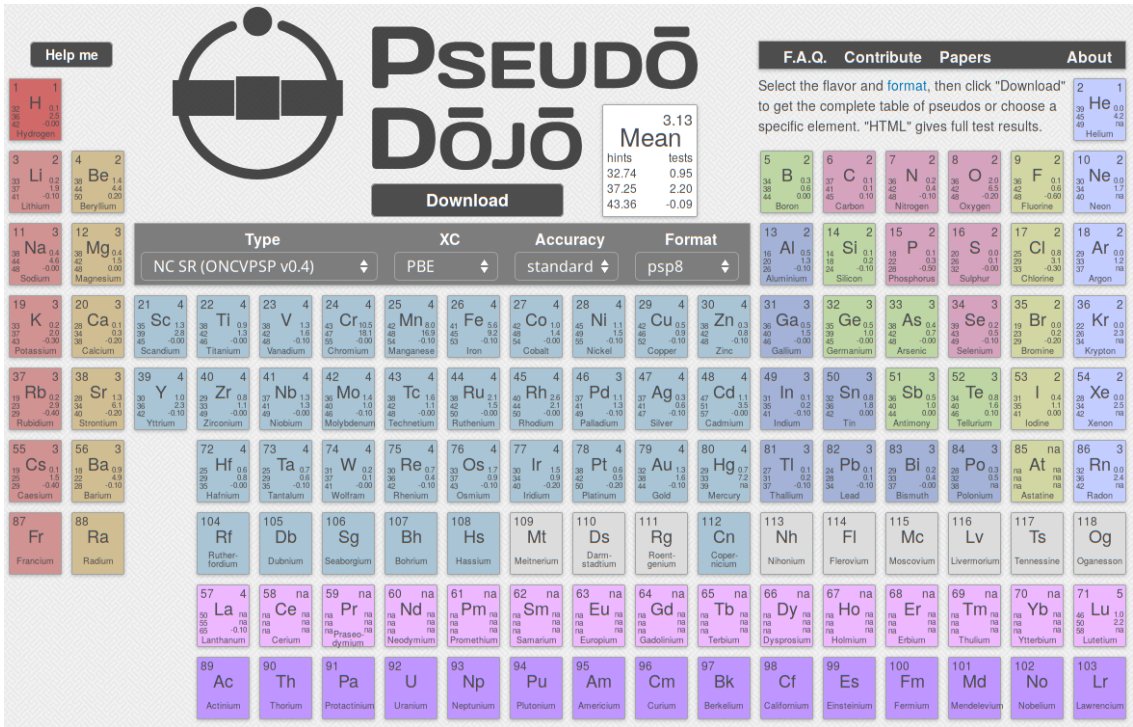
- High-quality pseudopotentials
  - Validation (Δ-factor)

  $\Rightarrow$ Reproducibility
- Periodic table
- Different XC functionals
- Suggested values for cutoff
- Integration with python code

Input: pseudopotentials

- High-quality pseudopotentials
  - Validation (Δ-factor)

  $\Rightarrow$ Reproducibility
- Periodic table
- Different XC functionals
- Suggested values for cutoff
- Integration with python code



M.J. van Setten *et al.*, Comp. Phys. Commun. **226**, 39 (2018)

`www.pseudo-dojo.org`

FHI98pp/ABINIT	13.5	13.4	13.4	13.2	13.0	13.2	13.4	13.3
HGH/ABINIT	2.2	2.2	2.2	2.0	2.3	2.2	2.1	2.2
HGH-NLCC/BigDFT	1.3	1.2	1.2	1.1	1.3	1.3	1.2	1.2
MBK2013/OpenMX	2.1	2.1	2.1	1.9	1.8	1.8	2.0	2.0
ONCVPSP/ABINIT	0.7	0.7	0.7	0.6	1.0	0.8	0.6	0.7
ONCVPSP(SG15)1/QE	1.4	1.3	1.3	1.3	1.6	1.5	1.3	1.4
ONCVPSP(SG15)2/CASTEP	1.4	1.4	1.4	1.3	1.6	1.5	1.4	1.4

Input: Abinit input files. `AbinitInput` object.

Programmatic interface to generate input files:

- Dict-like object storing ABINIT variables
- Methods to set variables (e.g. k-path from structure)
- Validation

Input: Abinit input files. `AbinitInput` object.

Programmatic interface to generate input files:

- Dict-like object storing ABINIT variables
- Methods to set variables (e.g. k-path from structure)
- Validation

Can invoke ABINIT to get important parameters such as:

- list of k-points in the IBZ
- list of irreducible DFPT perturbations
- list of possible configurations for MPI jobs (npkpt, npfft, npband ...)

```
inp = abilab.AbinitInput(structure="si.cif", pseudos="Si.psp8")
inp["ecut"] = 30
inp["ngkpt"] = [8, 8, 8]
```

Input: Abinit input files. `AbinitInput` object.

Programmatic interface to generate input files:

- Dict-like object storing ABINIT variables
- Methods to set variables (e.g. k-path from structure)
- Validation

Can invoke ABINIT to get important parameters such as:

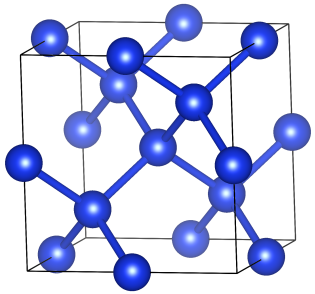
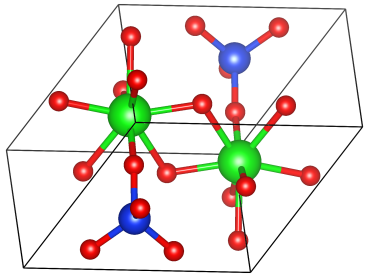
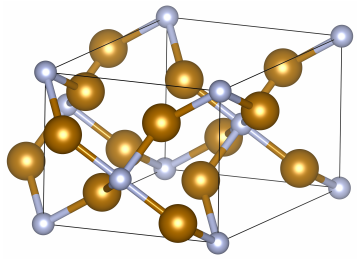
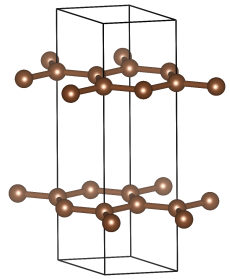
- list of k-points in the IBZ
- list of irreducible DFPT perturbations
- list of possible configurations for MPI jobs (npkpt, npfft, npband ...)

```
inp = abilab.AbinitInput(structure="si.cif", pseudos="Si.psp8")
inp["ecut"] = 30
inp["ngkpt"] = [8, 8, 8]
```

Good for low throughput calculations

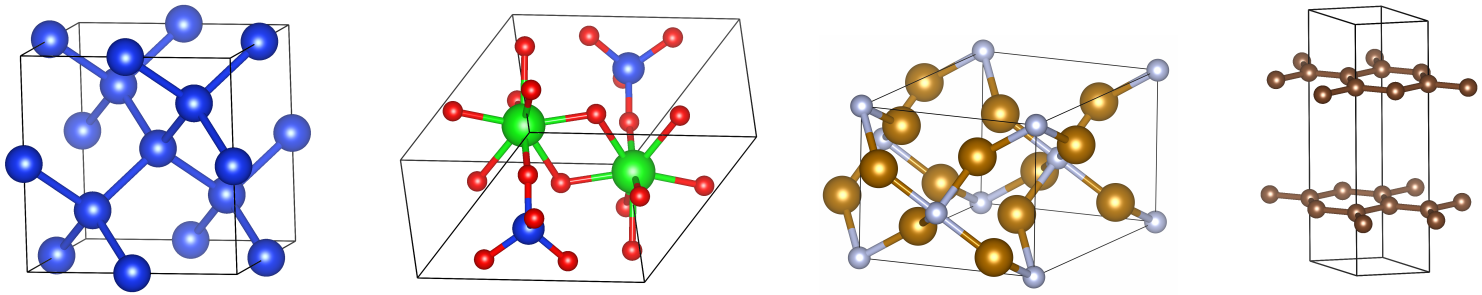
Input: Abinit input files. Factory functions

High-throughput input structures may be very different

Input: Abinit input files. Factory functions

High-throughput input structures may be very different



⇓

Need for an easier interface: factory functions

- Minimal input from user (structure, pseudopotential table)
- Metavariables. e.g. kppa for the BZ sampling
- Default values designed to cover the most common scenarios
- Optional arguments to change the default behaviour

Input: Abinit input files. Factory functions

```
def ebands_input(structure, pseudos, kppa=None, nscf_nband=None, ndivsm=15,
                 ecut=None, pawecutdg=None, scf_nband=None, accuracy="normal",
                 spin_mode="polarized", smearing="fermi_dirac:0.1 eV",
                 charge=0.0, scf_algorithm=None, dos_kppa=None):
```

Input: Abinit input files. Factory functions

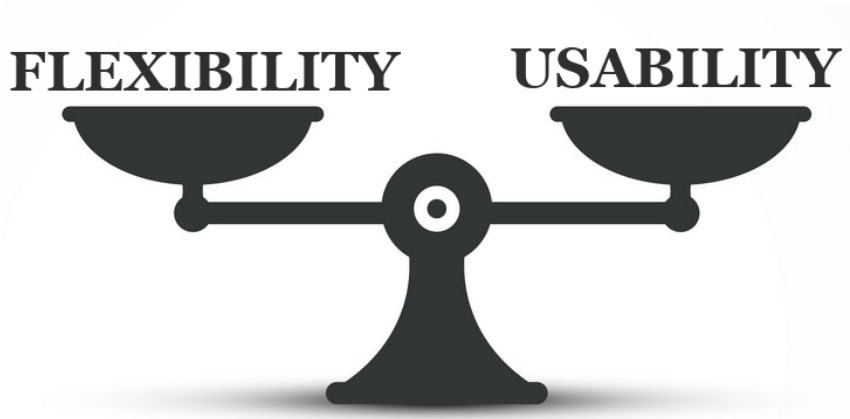
```
def ebands_input(structure, pseudos, kppa=None, nscf_nband=None, ndivsm=15,
                 ecut=None, pawecutdg=None, scf_nband=None, accuracy="normal",
                 spin_mode="polarized", smearing="fermi_dirac:0.1 eV",
                 charge=0.0, scf_algorithm=None, dos_kppa=None):
```

Designed to be the entry point for our workflows: 2 challenges

Input: Abinit input files. Factory functions

```
def ebands_input(structure, pseudos, kppa=None, nscf_nband=None, ndivsm=15,
                 ecut=None, pawecutdg=None, scf_nband=None, accuracy="normal",
                 spin_mode="polarized", smearing="fermi_dirac:0.1 eV",
                 charge=0.0, scf_algorithm=None, dos_kppa=None):
```

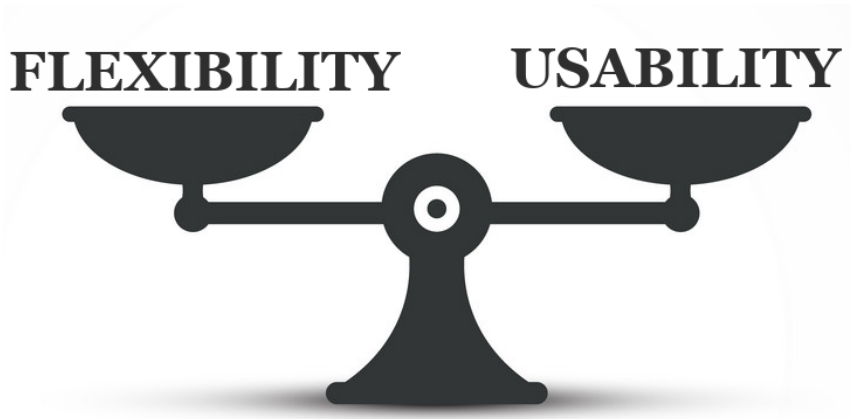
Designed to be the entry point for our workflows: 2 challenges

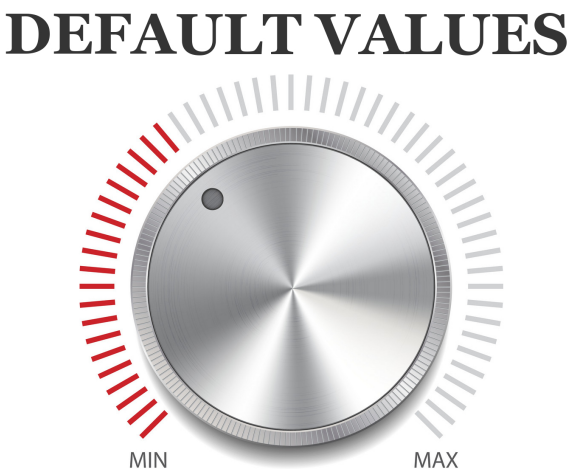


Input: Abinit input files. Factory functions

```
def ebands_input(structure, pseudos, kppa=None, nscf_nband=None, ndivsm=15,
                 ecut=None, pawecutdg=None, scf_nband=None, accuracy="normal",
                 spin_mode="polarized", smearing="fermi_dirac:0.1 eV",
                 charge=0.0, scf_algorithm=None, dos_kppa=None):
```

Designed to be the entry point for our workflows: 2 challenges





The road to a high-throughput framework

What do we need to run high-throughput workflows?

- ✔ DFT code 
- ✔ Inputs
  - Structures pymatgen
  - Pseudopotentials and cutoffs 
  - Automatic generation abipy
- Error handling
- Database interface
- Workflow management
- Data analysis

Error handling

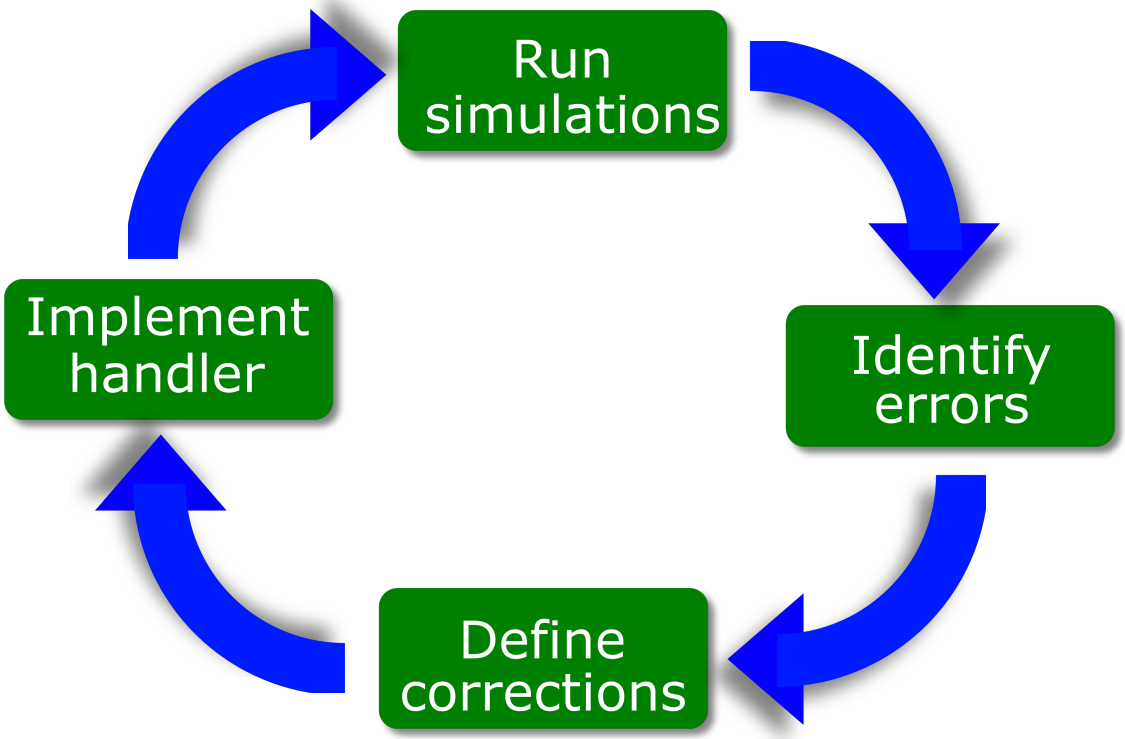
- Define a standard interface for handlers
  - Identify a type of error
  - Apply a correction
- Error messages identofied by tags and encoded in yaml

```
--- !DilatmxError
src_file: mover.F90
message: |
    Dilatmx has been exceeded too many times (4)
    Restart your calculation from larger lattice vectors and/or a larger dilatmx
...
```

- Parsing errors for a generic code might be difficult (text messages)
  - Message may be not self-explanatory
  - Messages may evolve from one version to another

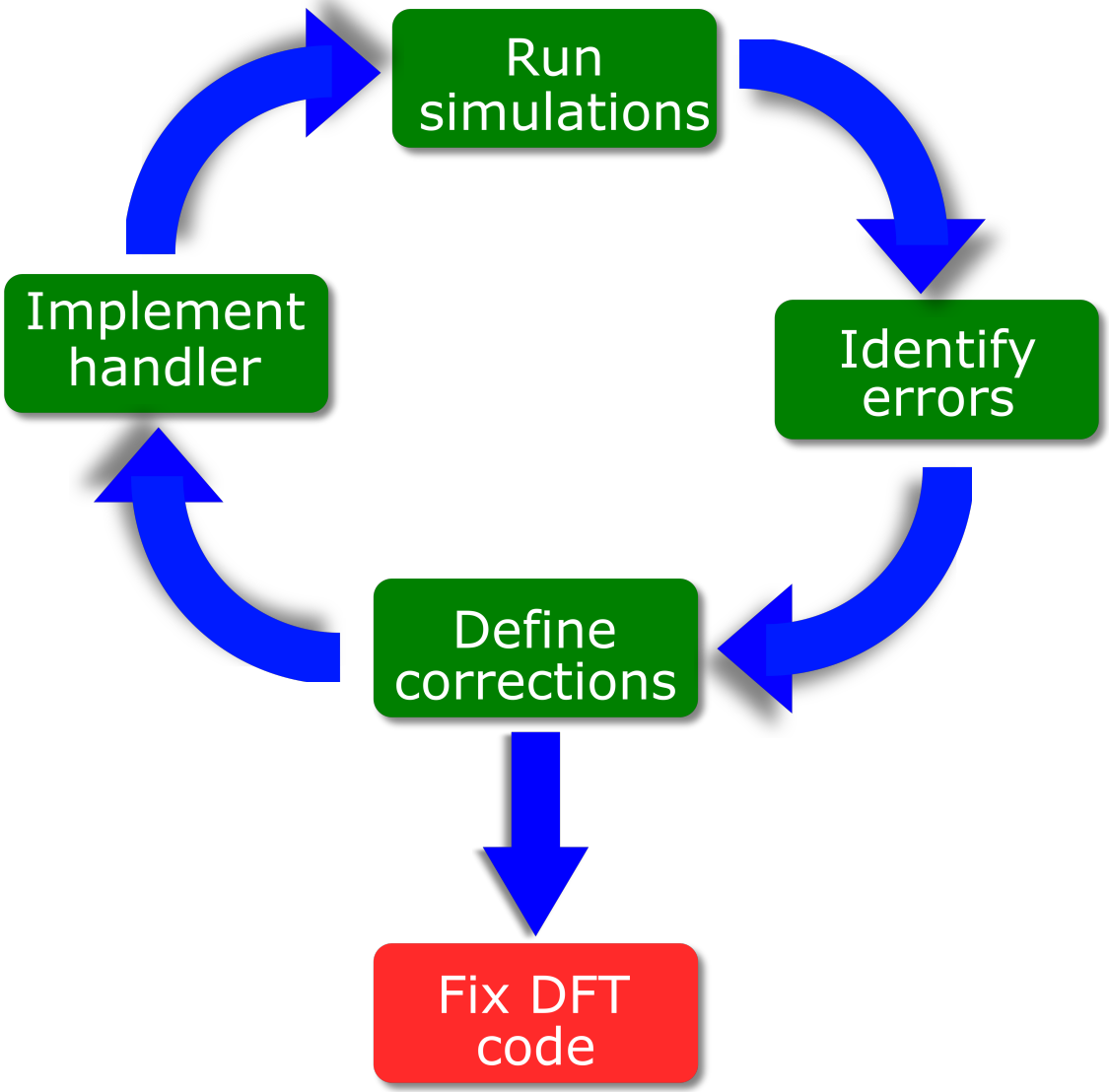
Error handling

Iterative procedure of findind an fixing error



Error handling

Iterative procedure of findind an fixing error



The road to a high-throughput framework

What do we need to run high-throughput workflows?

- ✔ DFT code 

- ✔ Inputs
  - Structures pymatgen
  - Pseudopotentials and cutoffs 

  - Automatic generation abipy
- ✔ Error handling abipy
- Database interface
- Workflow management
- Data analysis

Database interface

- Document base database + Document-Object Mapper
- Mixins for standard quantities
- Store netcdf output files
- Only properties to be queried in the main document

$\Rightarrow$ Standardized output fields

```
class MaterialMixin(object):
    """
    fields describing the material examined in the calculation
    """
class PhononResult(MaterialMixin, DateMixin, DirectoryMixin, Document):
    """
    results for a phonon workflow
    """
```

The road to a high-throughput framework

What do we need to run high-throughput workflows?

- ✔ DFT code 

- ✔ Inputs
  - Structures pymatgen
  - Pseudopotentials and cutoffs 

  - Automatic generation 

- ✔ Error handling 

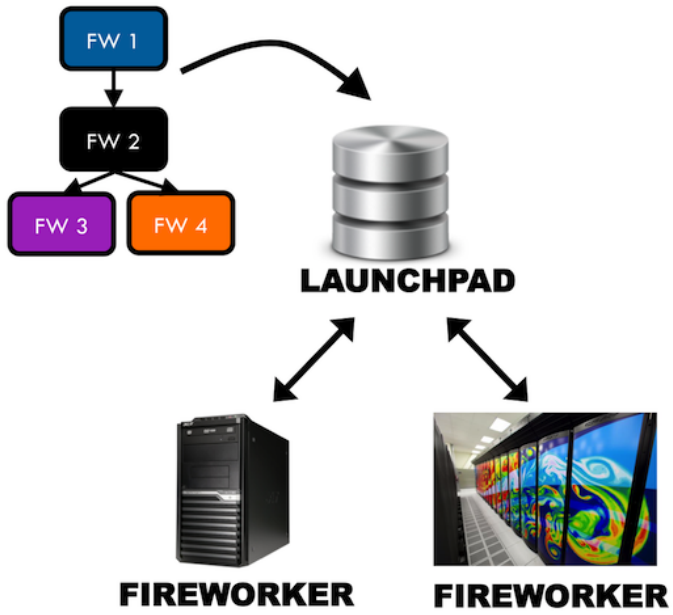
- ✔ Database interface 

  - Workflow management
  - Data analysis

Managing workflows

FireWorks

- General purpose workflow manager
- Database backend (MongoDB)
- Support for dynamic workflows
- Support for several queueing systems such as PBS, SLURM...
- Workflow logic should be implemented

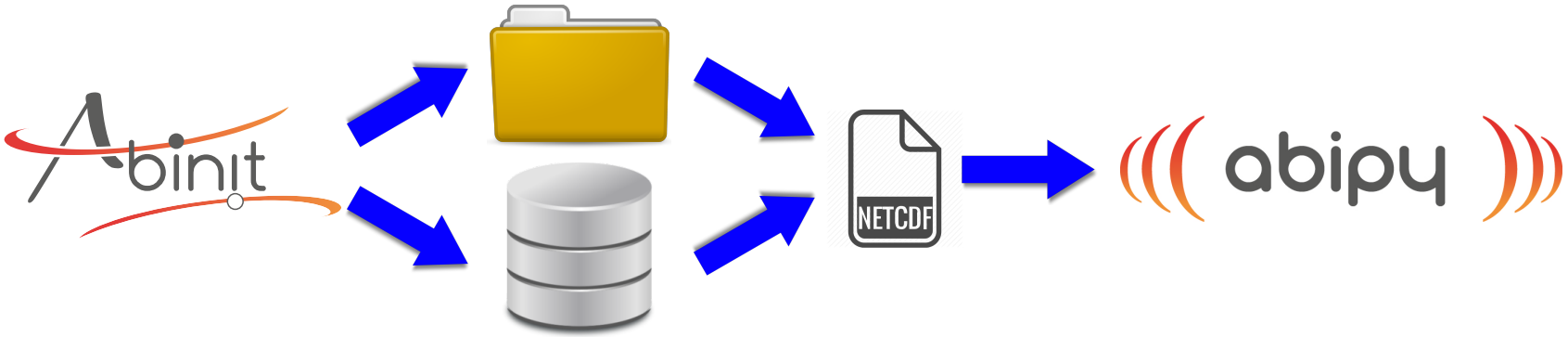


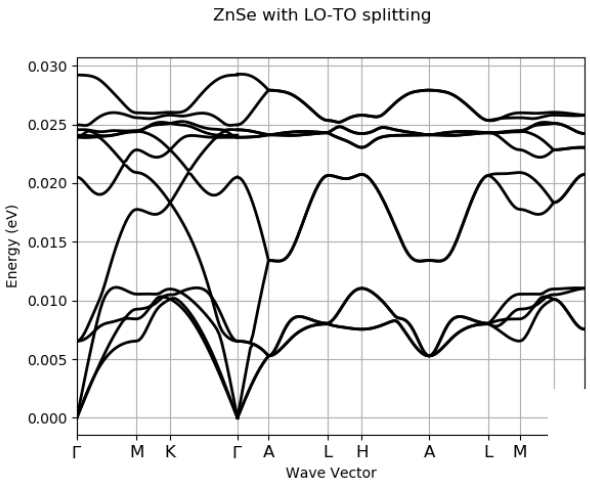
The road to a high-throughput framework

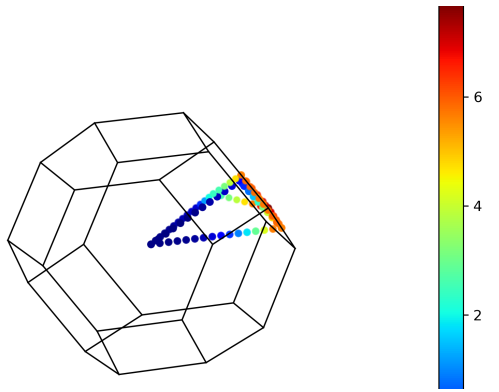
What do we need to run high-throughput workflows?

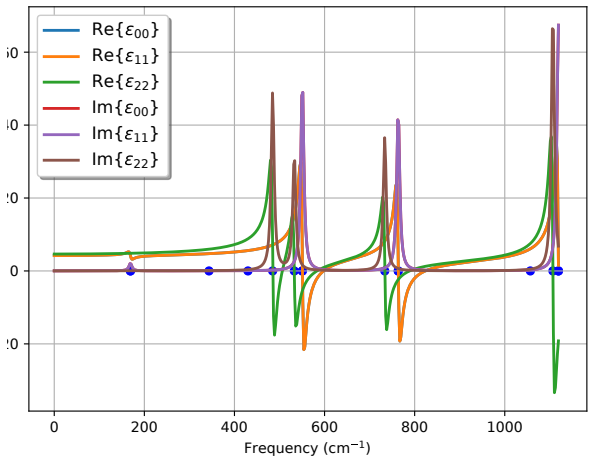
- ✔ DFT code 
- ✔ Inputs
  - Structures pymatgen
  - Pseudopotentials and cutoffs 
  - Automatic generation 
- ✔ Error handling 
- ✔ Database interface 
- ✔ Workflow management 
  - Data analysis

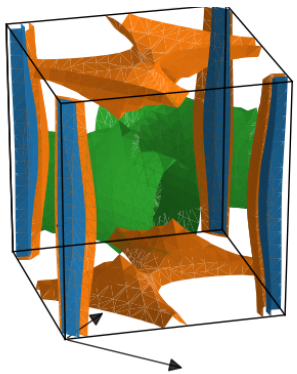
Data analysis

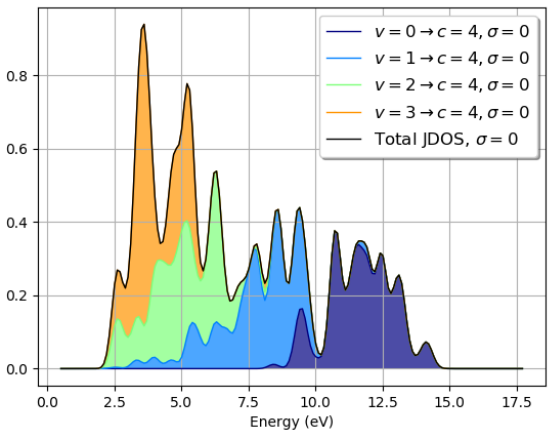


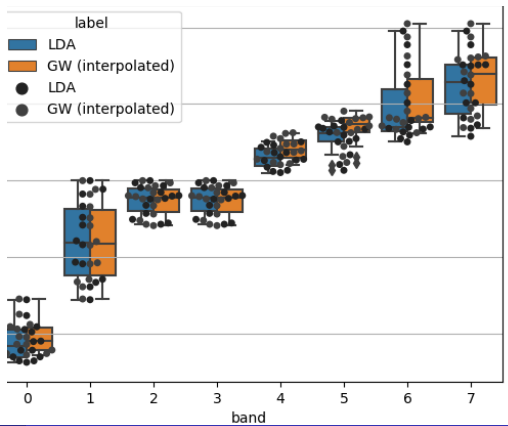












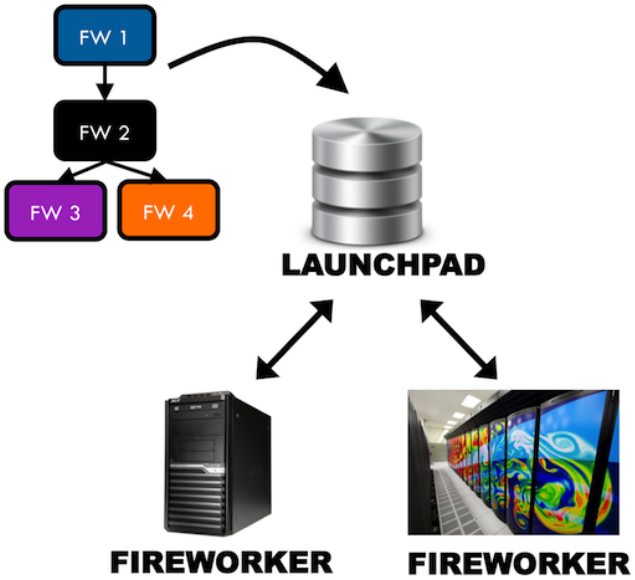
The road to a high-throughput framework

What do we need to run high-throughput workflows?

- ✔ DFT code 
- ✔ Inputs
  - Structures pymatgen
  - Pseudopotentials and cutoffs 
  - Automatic generation abipy
- ✔ Error handling abipy
- ✔ Database interface mongoengine
- ✔ Workflow management FireWorks
- ✔ Data analysis abipy

Outline

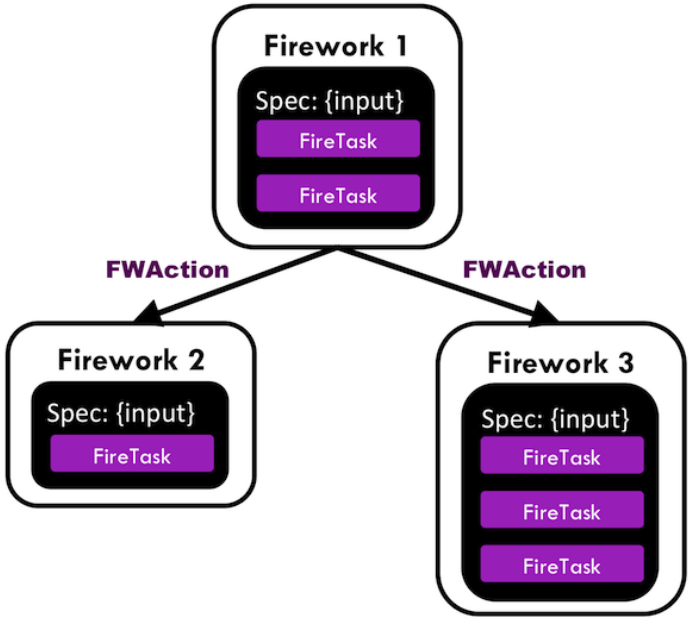
Fireworks features



Potential differences wrt Aiida

- No central control center. Central storage of the status: the DB
- Workers pull the jobs to be executed
- Python code wraps the execution of the DFT code on the node

- 1 firework $\simeq$ 1 job in the queue
- Logic implemented in the Tasks



Abiflows

⇓

Implementation of the logic in the Abinit tasks

⇓

Different logic required for different types of calculations

⇓

Building blocks to be composed to write workflows

Abiflows

abiflows

⇓

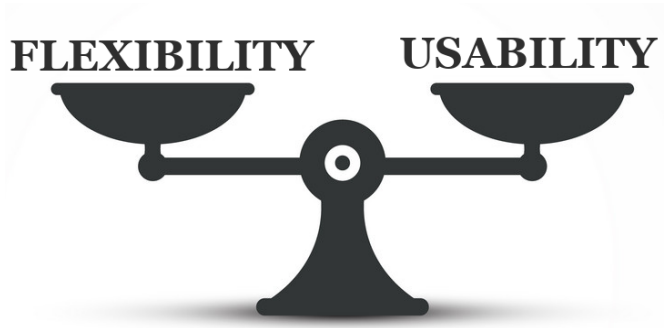
Implementation of the logic in the Abinit tasks

⇓

Different logic required for different types of calculations

⇓

Building blocks to be composed to write workflows



Generation

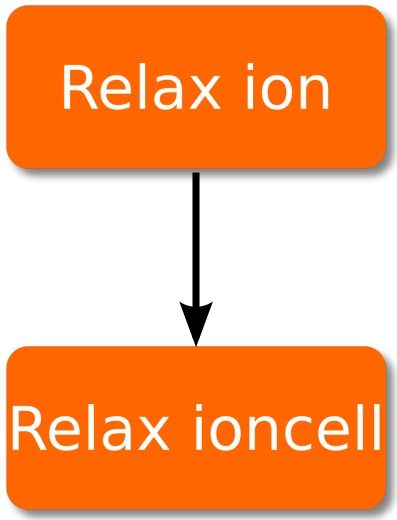
Based on Abipy factory functions. Relax workflow generation

```
wf = RelaxFWWorkflow.from_factory(structure, pseudos, kppa=None, nband=None,
  ecut=None, accuracy="normal", spin_mode="polarized", smearing=None,
  scf_algorithm=None, extra_abivars=None, autoparal=False, charge=0,
  initialization_info=None, target_dilatmx=None, shift_mode="Gamma")
```

Definition of the arguments

- Abinit inputs
- Additional arguments to manage the workflow

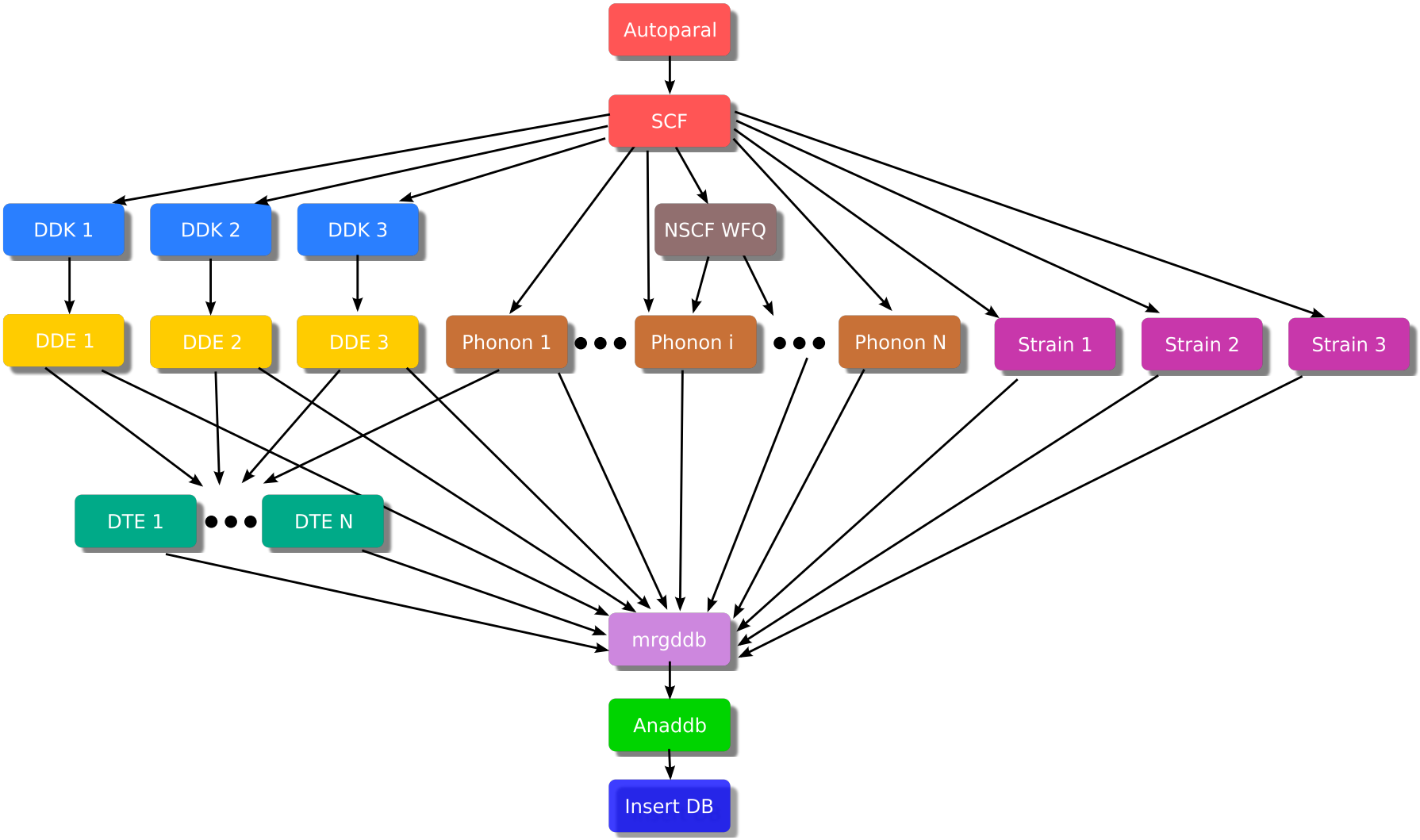
$\Rightarrow$ Requires balancing



Properties that do not affect directly the main workflow structure can be set with ad-hoc functions and shared among the different workflows

DFPT workflow

Complex workflow: all the possible DFPT calculations available in Abinit



Complexity

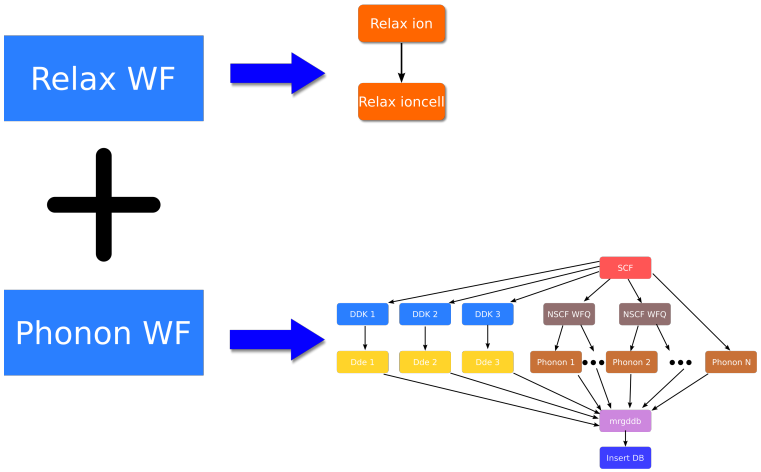
More control variables:

```
wf = DfptFWWorkflow.from_factory(structure, pseudos, ...,
   qppa=None, do_ddk=True, do_dde=True, do_strain=True, do_dte=False,
   scf_tol=None, ph_tol=None, ddk_tol=None, dde_tol=None, wfq_tol=None,
   strain_tol=None, extra_abivars=None, autoparal=False)
```

Complexity

More control variables:

```
wf = DfptFWWorkflow.from_factory(structure, pseudos, ...,
   qppa=None, do_ddk=True, do_dde=True, do_strain=True, do_dte=False,
   scf_tol=None, ph_tol=None, ddk_tol=None, dde_tol=None, wfq_tol=None,
   strain_tol=None, extra_abivars=None, autoparal=False)
```

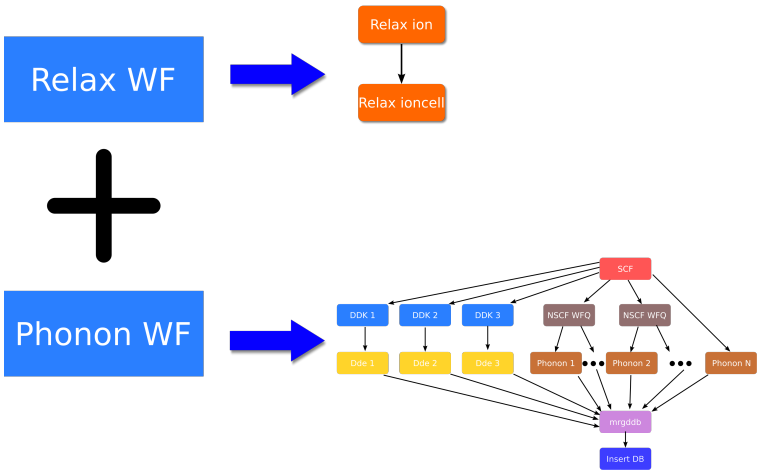


Workflows can also be composed to generate more complex ones

Complexity

More control variables:

```
wf = DfptFWWorkflow.from_factory(structure, pseudos, ...,
    qppa=None, do_ddk=True, do_dde=True, do_strain=True, do_dte=False,
    scf_tol=None, ph_tol=None, ddk_tol=None, dde_tol=None, wfq_tol=None,
    strain_tol=None, extra_abivars=None, autoparal=False)
```



Workflows can also be composed to generate more complex ones

As a rule of thumb we do not go beyond this level of complexity in a single workflow for high-throughput

Complexity

Potential issues:

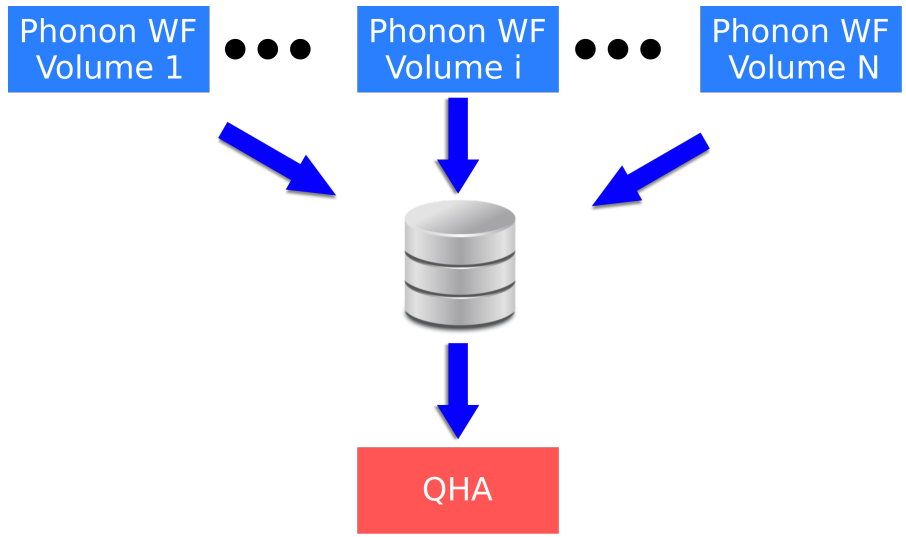
- More difficult to recover in the middle of a complex workflows
- Find default values suitable for large set of materials
- Less user friendly

Complexity

Potential issues:

- More difficult to recover in the middle of a complex workflows
- Find default values suitable for large set of materials
- Less user friendly

Alternatively a simpler solution usually adopted in our group



Connecting files

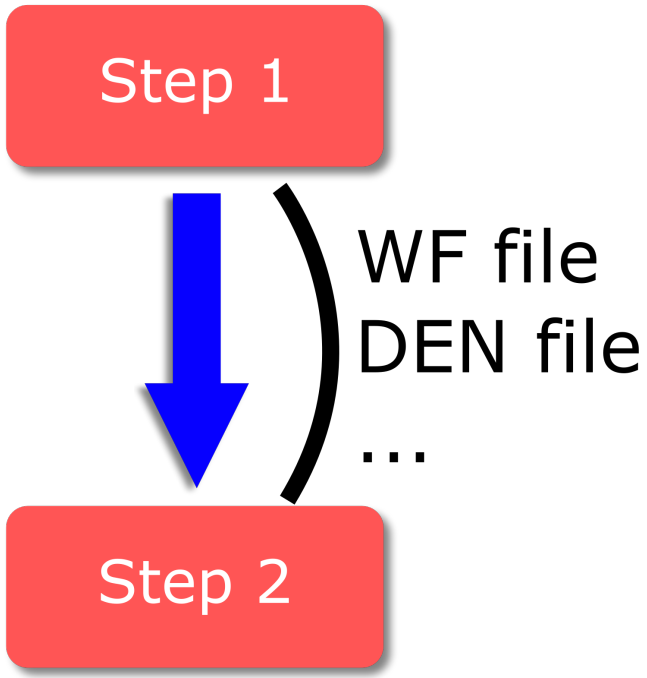
Different files need to be available for different steps of the workflows

E.g. NSCF calculation needs density from SCF step.

$\Downarrow$

Dependencies grow with the number of steps

- DFT code have their own conventions
- Copying all the files is a partial solution
  - Multiple sources
  - Input names $\neq$ output names
- Options may enable/require additional files
- Restarting needs files



Connecting files

Different files need to be available for different steps of the workflows

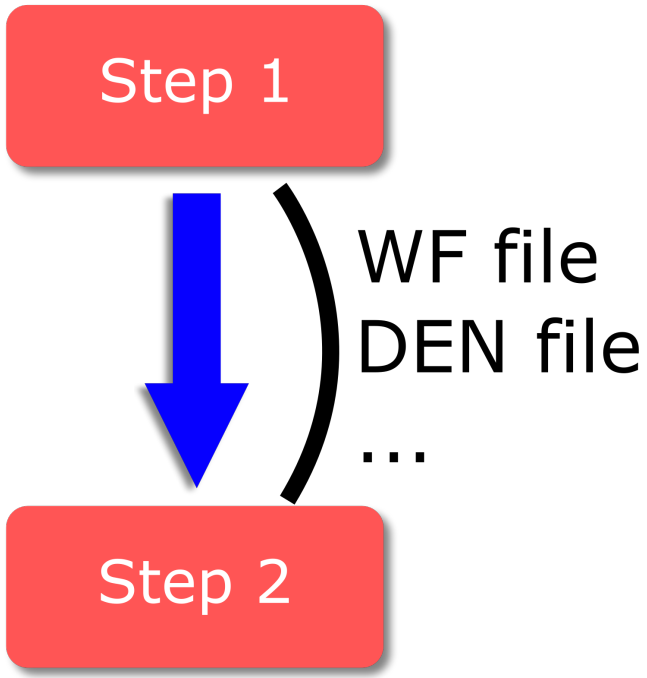
E.g. NSCF calculation needs density from SCF step.

$\Downarrow$

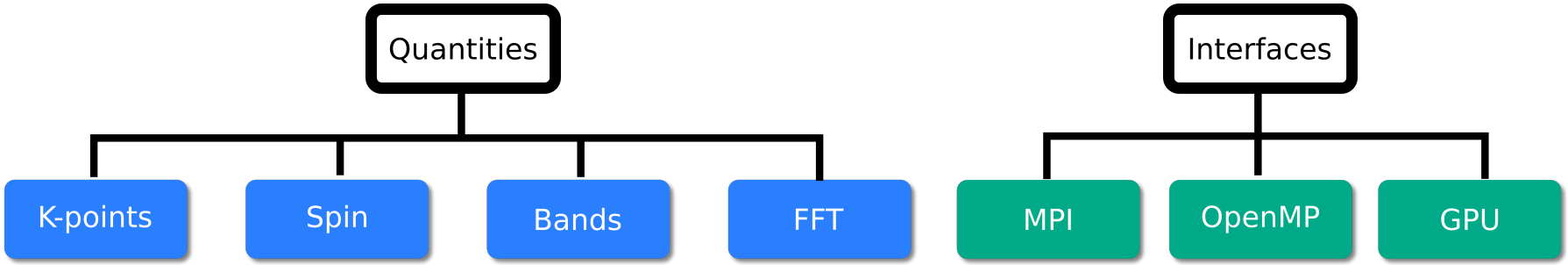
Dependencies grow with the number of steps

Strategies:

- Define conventions
- Customizable internal selection
- Profiles for different kind of calculations
- If needed an option for the user



Parallelization



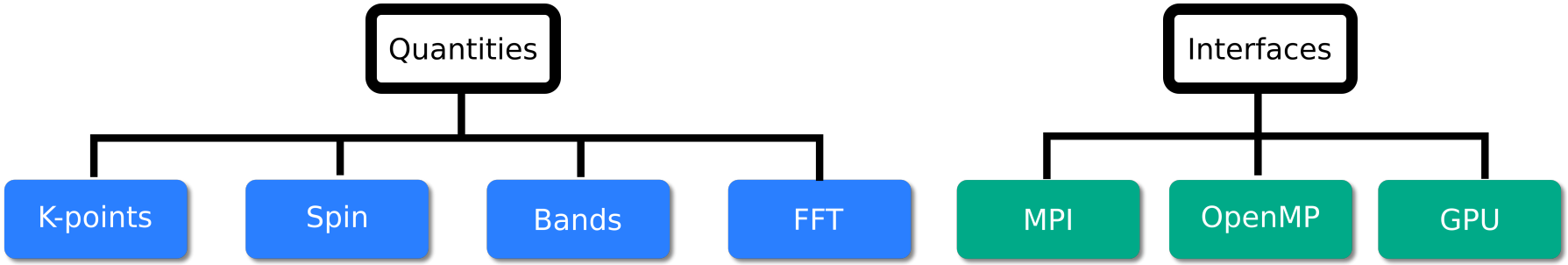
Optimizing parallelization is tricky:

- Many parallelization levels
- Cluster constraints (cores, nodes, . . .)
- Memory

Different approaches to automation:

- Same configuration for all the calculations
- User defined

Parallelization



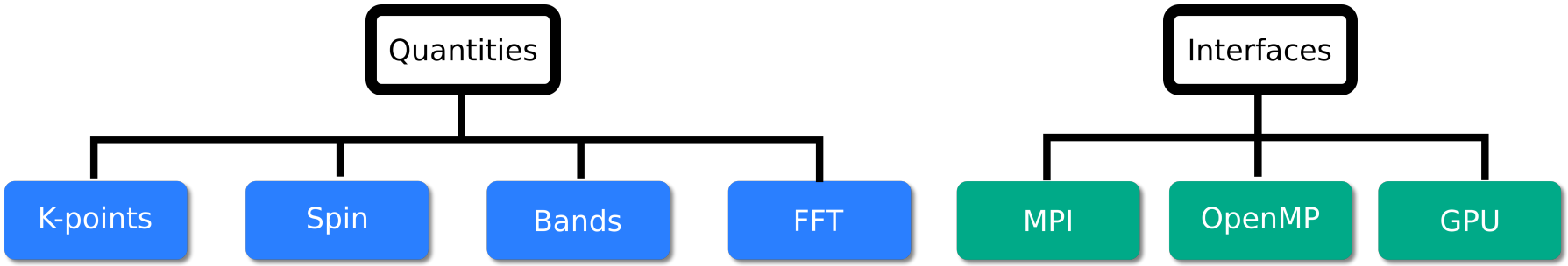
Optimizing parallelization is tricky:

- Many parallelization levels
- Cluster constraints (cores, nodes, ...)
- Memory

Different approaches to automation:

- Same configuration for all the calculations
- User defined
- Automatic parallelization

Parallelization



First workflow step: Abinit autoparal

- Effective basic parallelization
- Not always satisfactory
- Poor with several cluster constraints

$\Rightarrow$ Room for improvements

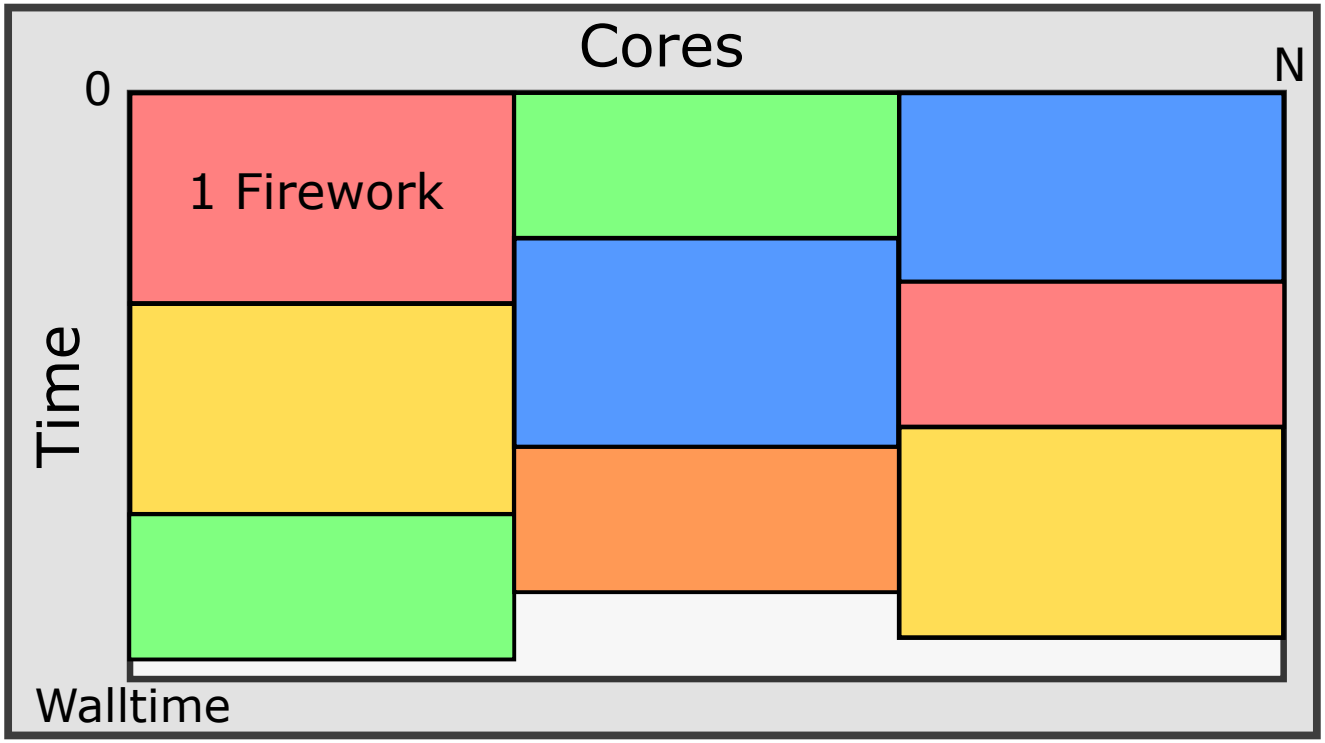
Parallelization: packing jobs

Small systems and cluster constraints: parallelization may be less effective

$\Downarrow$

Job Packing in FireWorks

1 Job in the queue

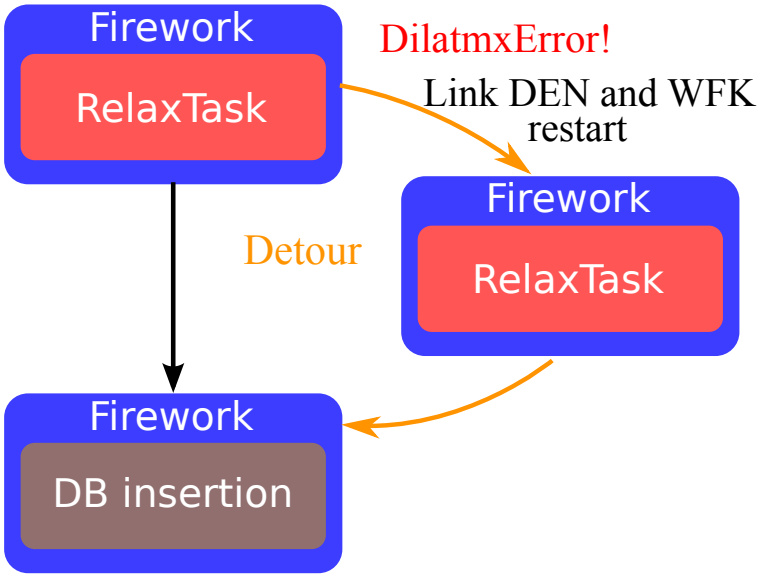


Requires settings to avoid hitting the walltime

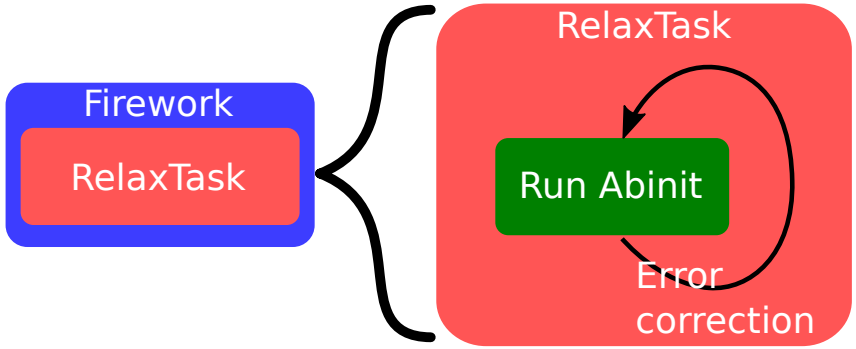
Dealing with failures: soft failures

Soft failures: error in the DFT code ⇒ error handlers

Restart in a new job



Restart locally



Preference for restart in a new job, in connection with the handling of the walltime and of the automatic parallelization.

Strong dependence on the cluster and DFT code features

Dealing with failures: hard failures

Hard failures: job killed by the queue manager

Potential causes:

- Walltime
- Memory
- Cluster failure

Difficult to deal with, especially in Fireworks

$\Downarrow$

Implementation of ad-hoc error handlers to identify the cause of the failure

Dealing with failures: hard failures

Hard failures: job killed by the queue manager

Potential causes:

- Walltime
- Memory
- Cluster failure

Difficult to deal with, especially in Fireworks

⇓

Implementation of ad-hoc error handlers to identify the cause of the failure

⇓

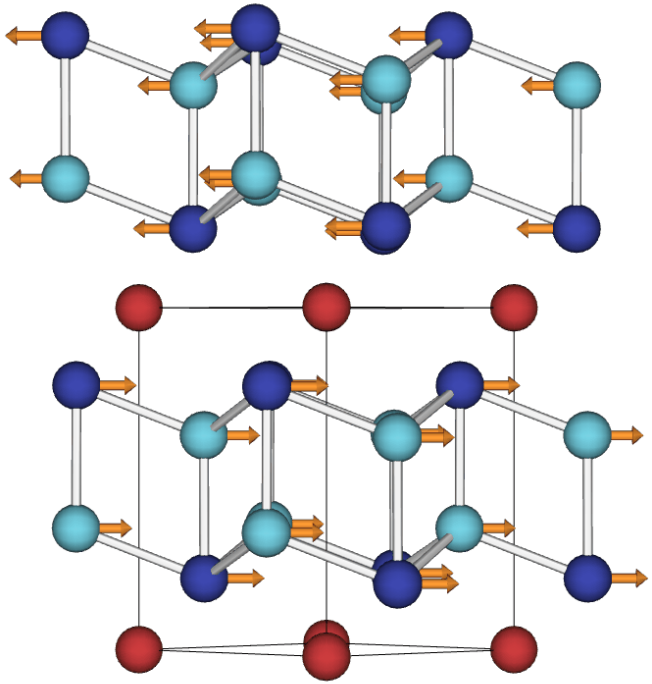
Difficulties in handling the different error messages

Currently disabled 

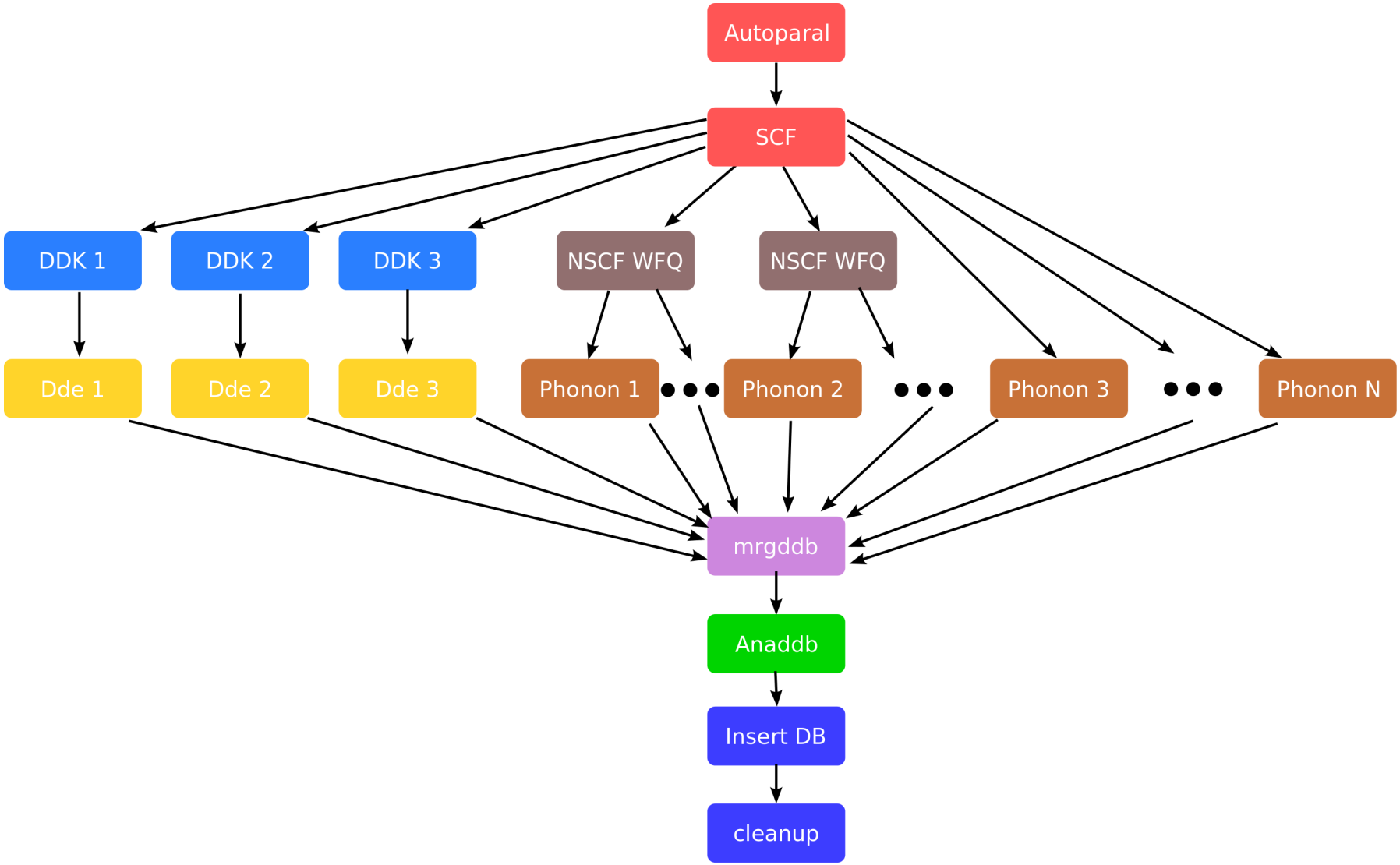
Outline

Why high throughput phonons?

- Superconductivity
- Optical properties (Raman, Indirect absorption)
- Transport (electronic and thermal)
- Ferroelectricity
- Carrier thermalization



DFPT workflow



Input parameters

Strategies for input parameters:

- Separate convervence study for each material
- Find good overall parameters

Input parameters

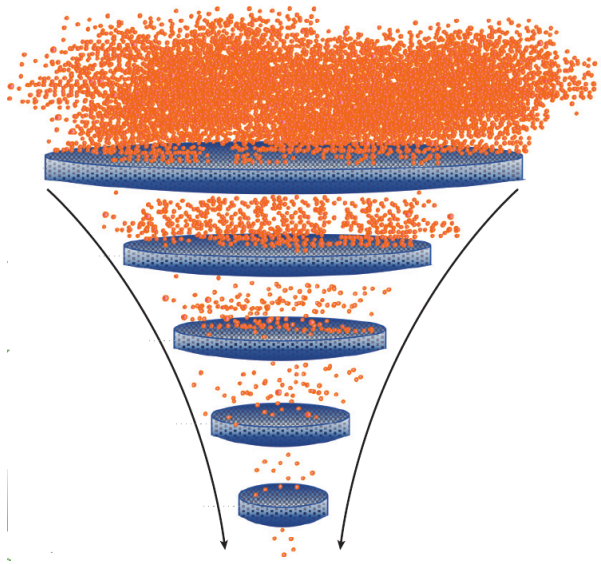
Strategies for input parameters:

- Separate convervence study for each material
- Find good overall parameters

For high-throughput and screening procedures finding good overall parameters is fine

- Cutoff
- K-points
- Q-points

What are "good" converged values?



Input parameters

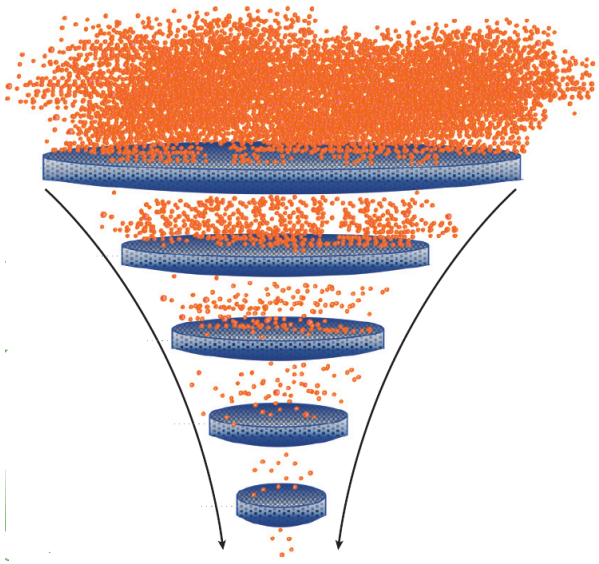
Strategies for input parameters:

- Separate convervence study for each material
- Find good overall parameters

For high-throughput and screening procedures finding good overall parameters is fine

- Cutoff
- K-points
- Q-points

What are "good" converged values?



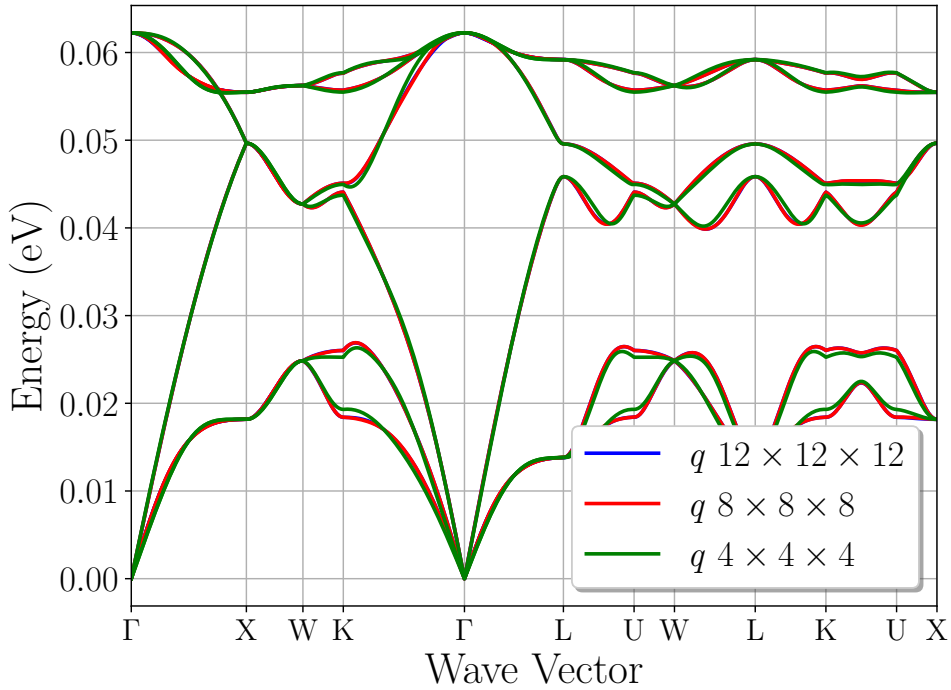
$\Downarrow$

Convergence study for K and Q points

Convergence study

Find optimal k-points and q-points sampling for high-throughput

- 48 semiconductors varying
  - elements
  - size
  - crystal symmetries
  - gaps
- Referencing dense grids:
  - relative and absolute error
  - mean and maximum error

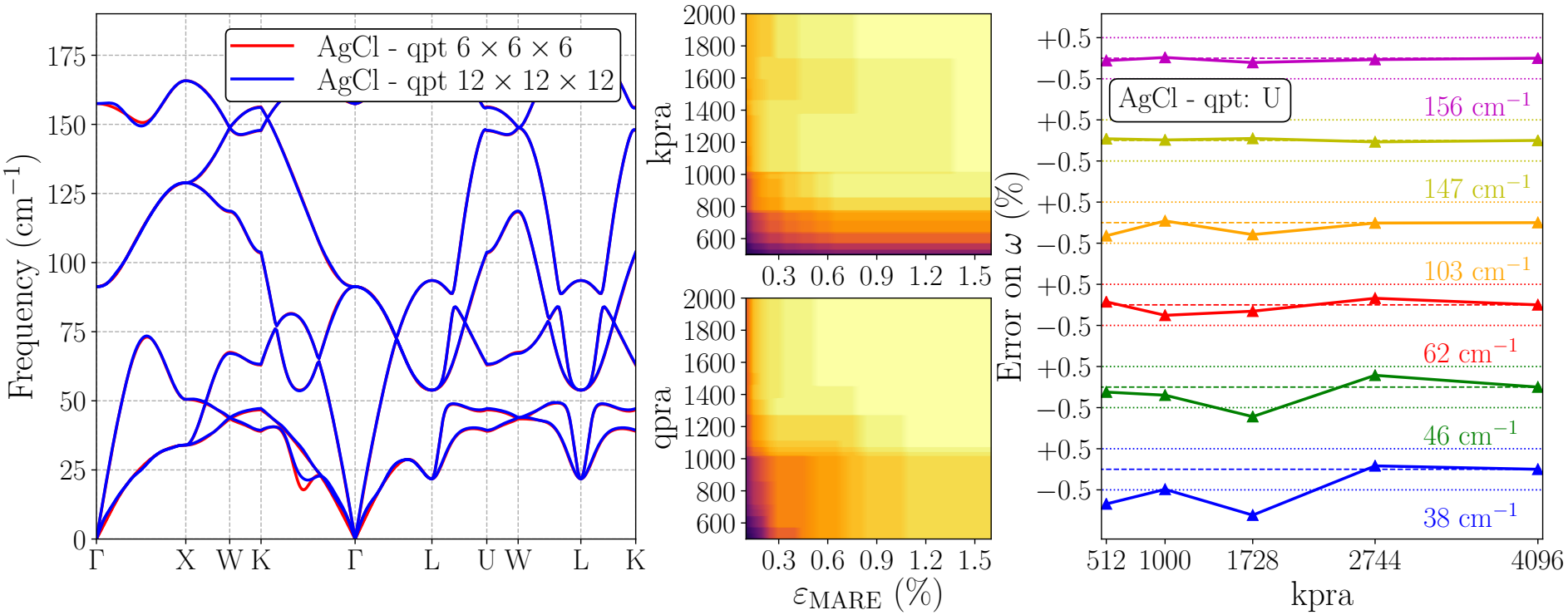


NaLi2Sb	Ca(CdP)2	CdS	SrLiP	InS	GaN	RbYO2
SiO2	BP	AlSb	LiZnP	MgCO3	ScF3	ZnGeN2
LiMgAs	P2Ir	Si	Li3Sb	K2O	Ga3Os	Be3P2
ZnSe	MgO	AgCl	SiC	YWN3	SrO	PbF2
MgSiP2	SiO2	GaP	Be2C	SnHgF6	MgMoN2	ZnO
ZrSiO4	Ba(MgP)2	Ba(MgAs)2	Ca(MgAs)2	C	RbI	FeS2
Si4P4Ru	CsBr	CsAu	RuS2	VCu3S4	ZnTe	

Petretto, Gonze, Hautier, Rignanese, *Comp. Mat. Sci.*, 144, 331 (2018)

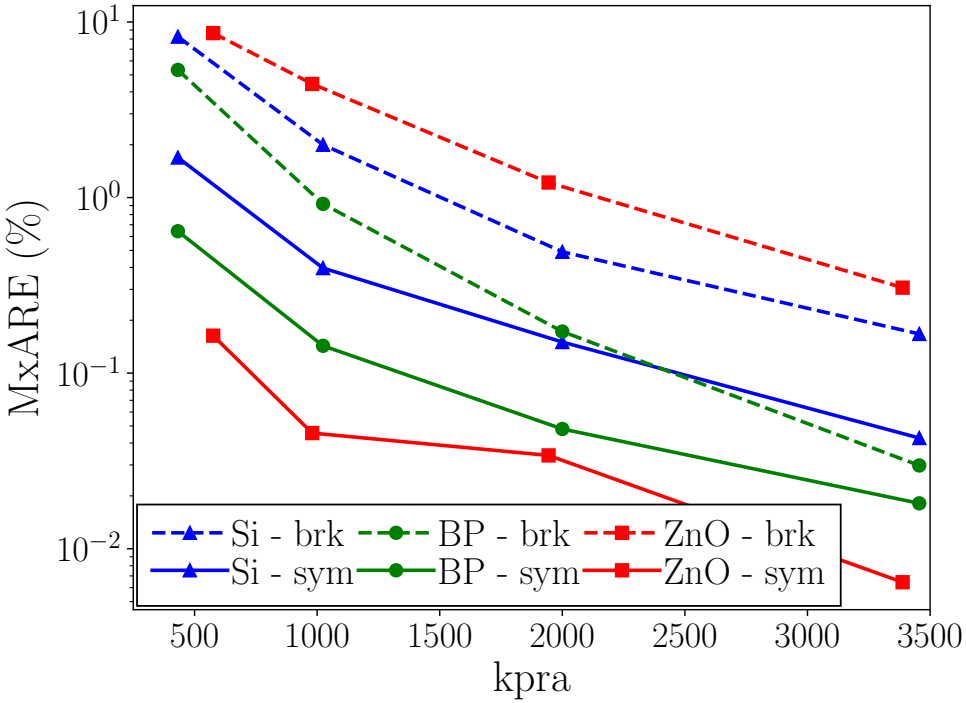
Convergence study: grids density

- absolute and relative errors on ω, E_{at}, Z^* and ϵ
- 1500 points per reciprocal atom $\Rightarrow$ $\mathrm{N_{kpt}}$*$\mathrm{N_{atoms}} \backsimeq 1500$
  - $\Rightarrow$ All materials converged with 0.5 cm^{-1} MAE and 0.6% MARE
- Better using a Q-grid commensurable with K-grid
  - $\Rightarrow$ Smoother close to Γ



Convergence study: symmetry of the grid

Convergence rate of phonon frequencies ω for symmetric versus non-symmetric grids



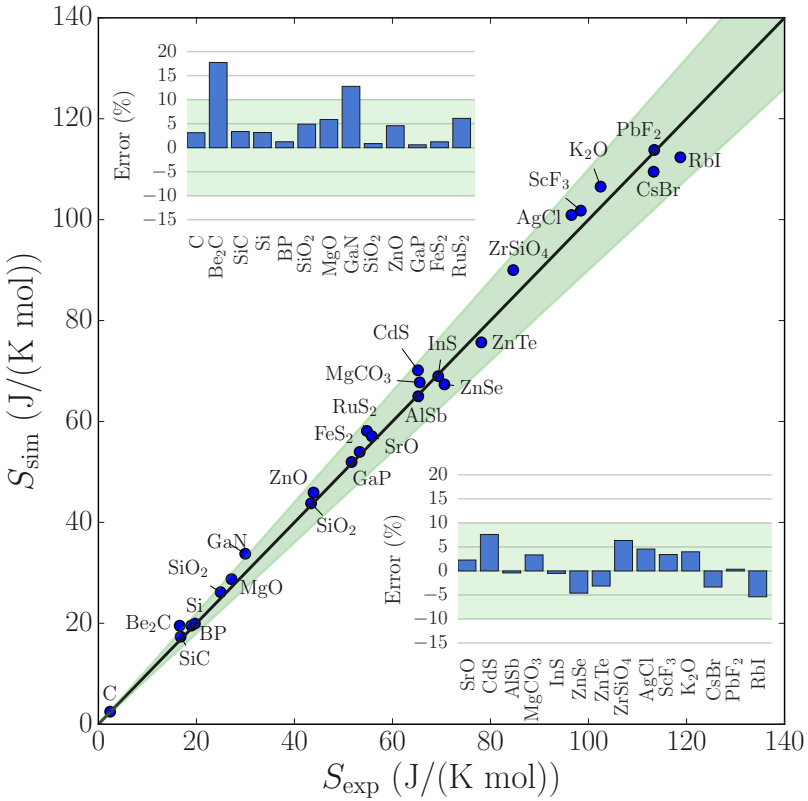
$\Downarrow$

Grids should preserve the symmetries of the system

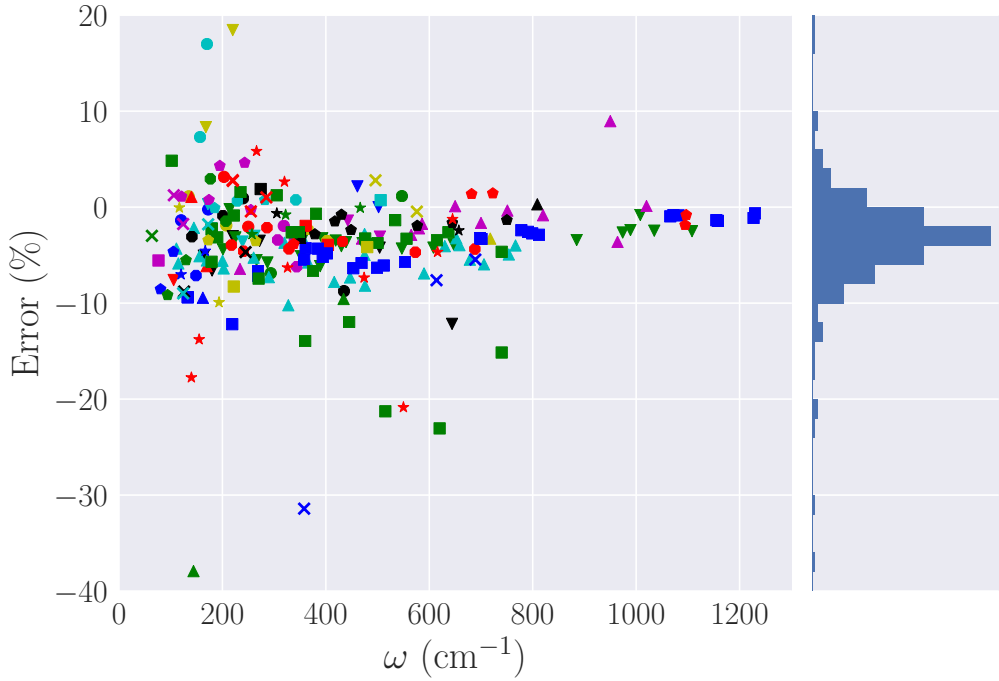
Results validation

Validation versus experimental data

Vibrational entropy at 300K



Γ phonon frequencies



Materials Project phonons database

Open access database: 1521 semiconductor materials (and growing...)

1508 of those materials with less than 13 atomic sites

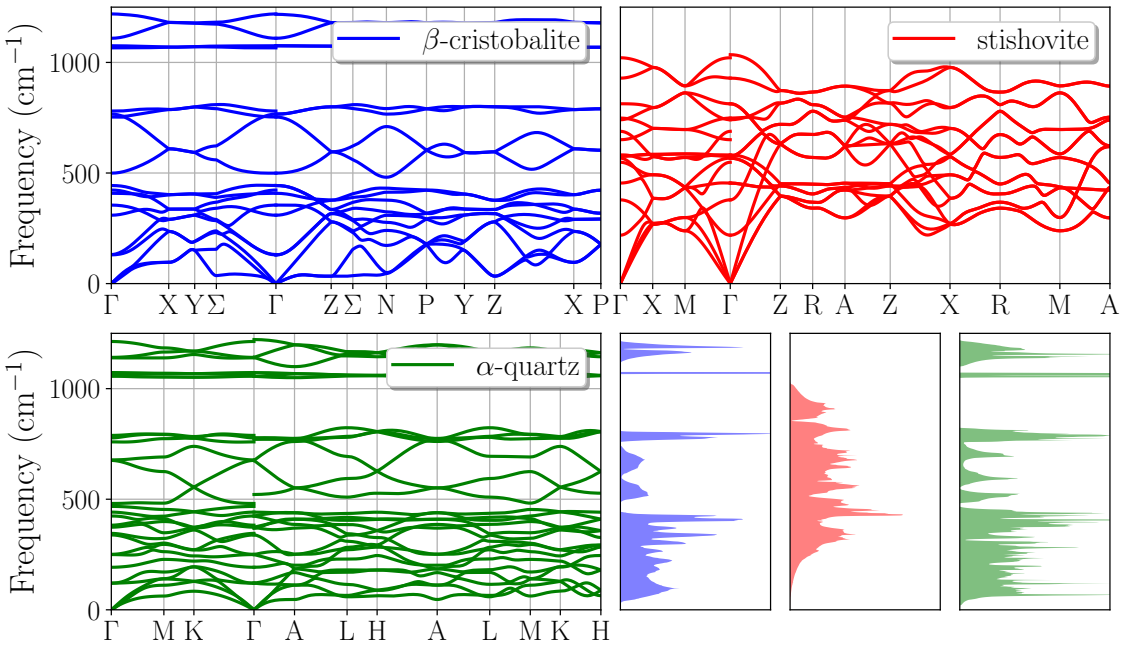
$\sim$ 5M CPU-hours

Materials Project phonons database

Open access database: 1521 semiconductor materials (and growing...)
1508 of those materials with less than 13 atomic sites
$\sim$ 5M CPU-hours

SCIENTIFIC DATA

- Interatomic force constants (DDB files)
- Phonon dispersion
- Born effective charges
- Dielectric tensor
- Thermodynamic prop.:
  - ΔF, ΔE_{ph}, C_v, S



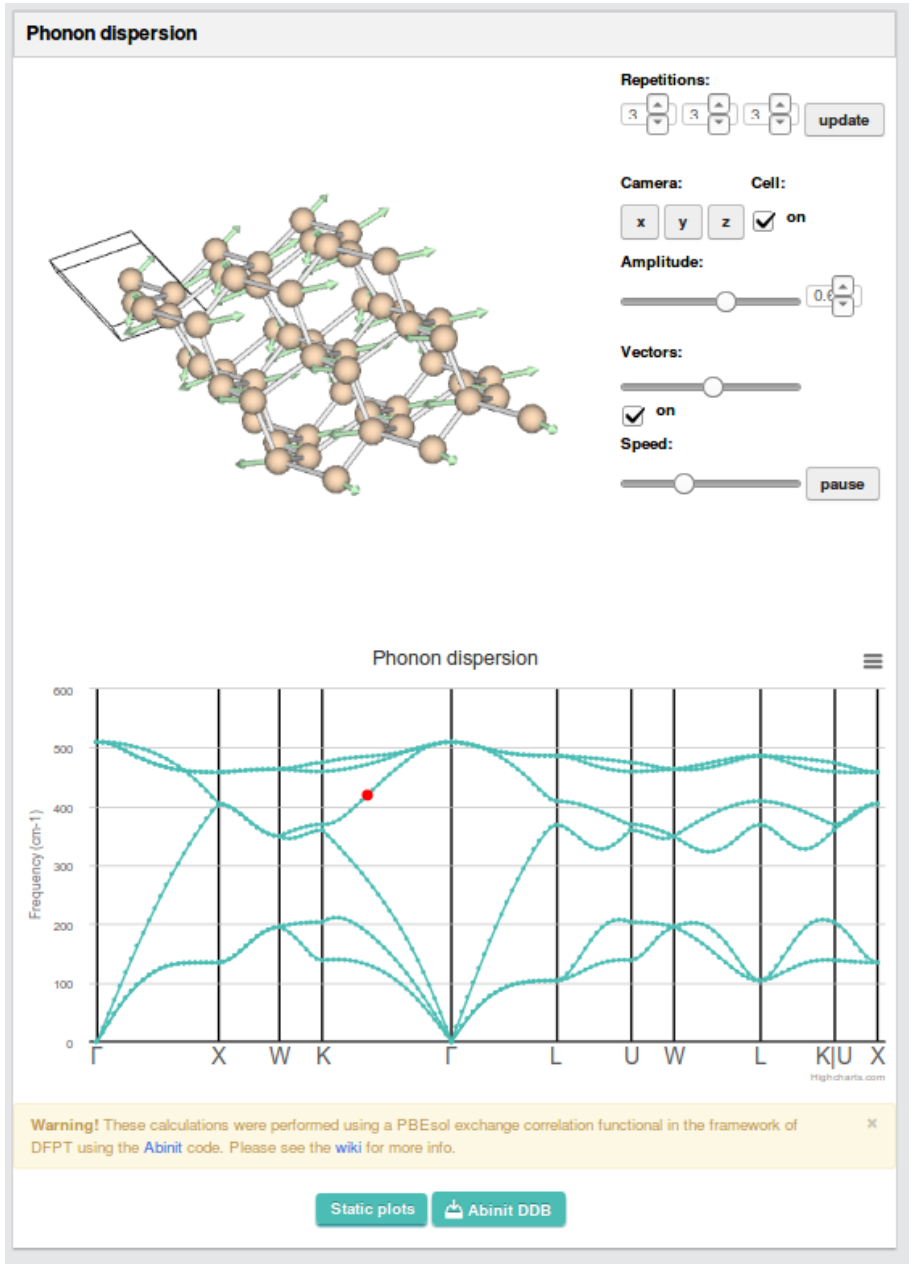
Petretto, Dwaraknath, Miranda, Winston, Giantomassi, Van Setten, Gonze, Persson, Hautier, Rignanese, Scientific Data (2018)

Materials Project phonons database

All the data available on the website and through REST service.

Interactive visualization of the phonon dispersion using the phononwebsite

`http://henriquemiranda.github.io/phononwebsite/`



Reliability: workflows

For the 1521 phonon band structures:

- Relax workflow
- Phonon workflow (+ anaddb)

Out of all the submitted workflows only ~ 30 did not complete successfully

Reliability: workflows

For the 1521 phonon band structures:

- Relax workflow
- Phonon workflow (+ anaddb)

Out of all the submitted workflows only ~ 30 did not complete successfully

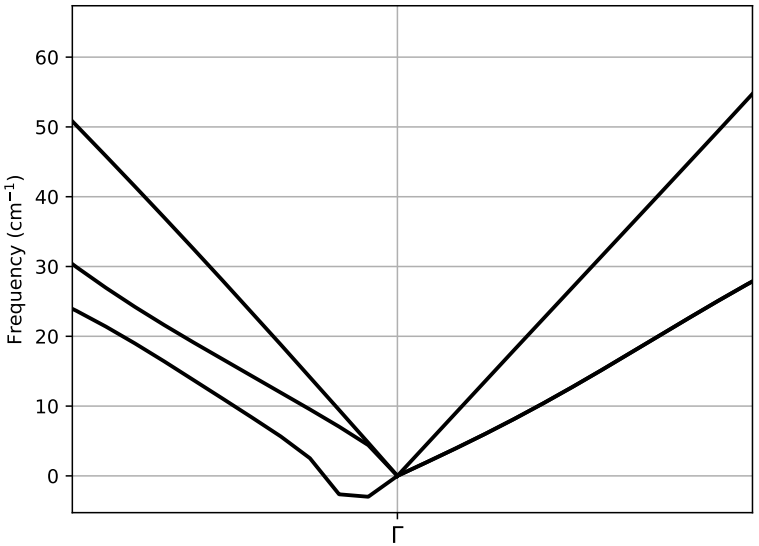
Main reasons:

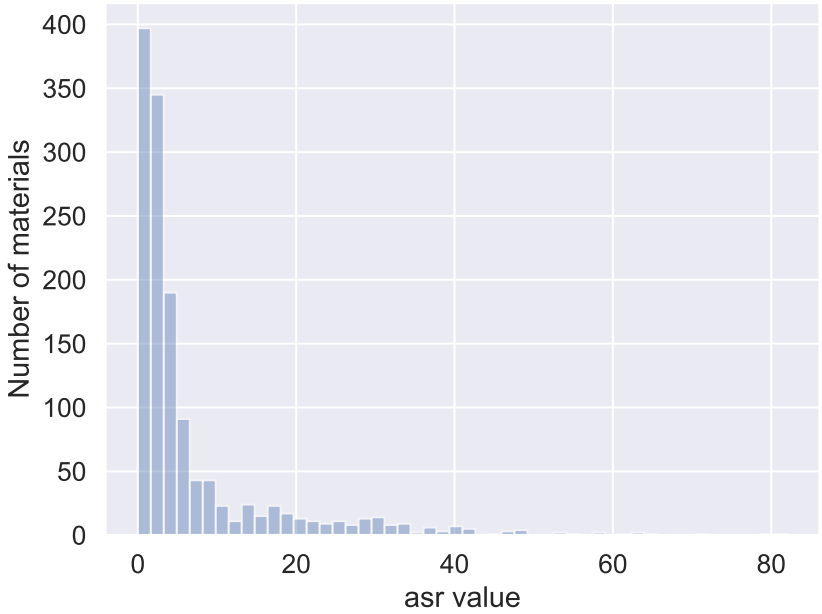
- Too slow relaxation/relaxation did not converge
- Too small gap (switched to metallic)
- Presence of La
- Poor choice of materials

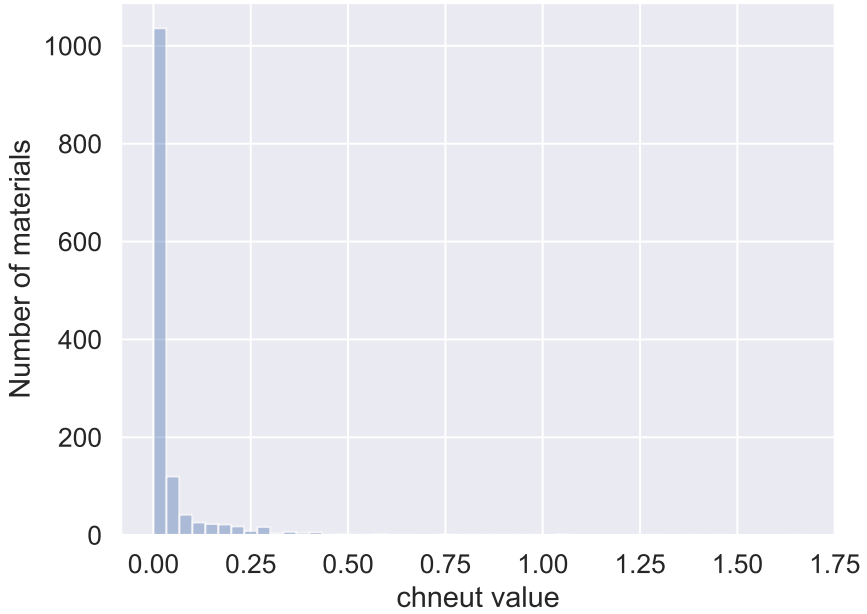
Reliability: results

Warnings available in the database:

- Negative ω close to Γ: 24 materials
- ASR break $>$30 cm^{-1}: 72 materials
- CNSR break $>0.2e$: 92 materials

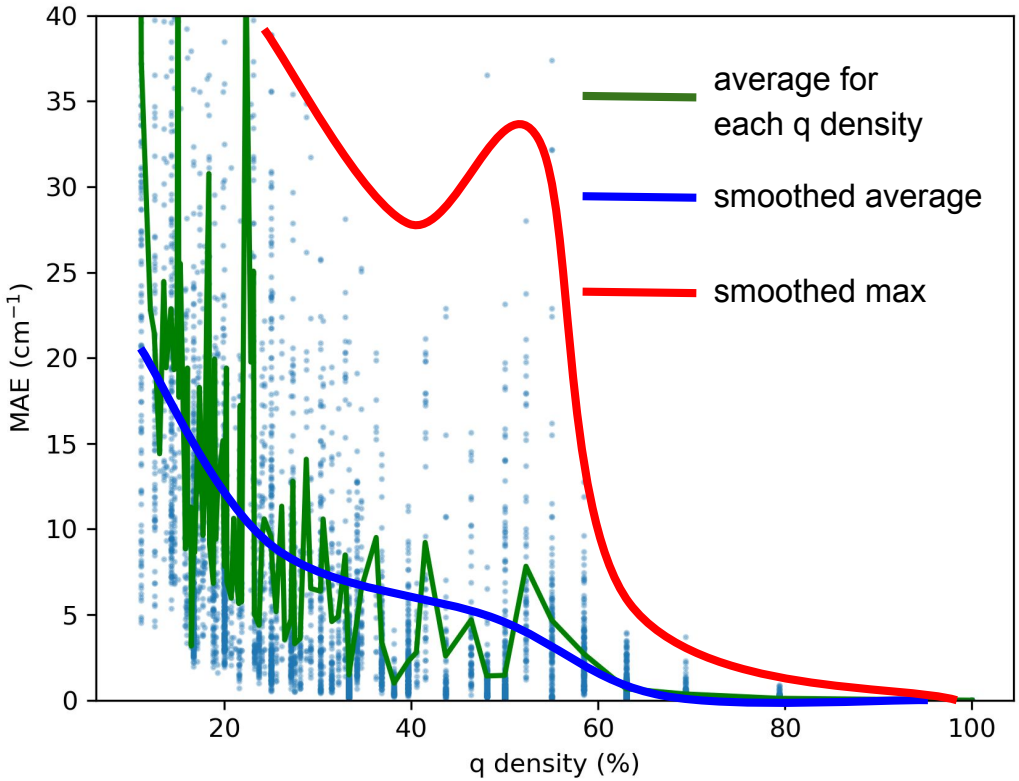






q-points convergence

Use subgrids of the q-point grid to check the convergence w.r.t. qpt



- Material dependent
- Suggest good convergence level at 1500 qppa
- reducing by a factor 2 may lead to sizeable errors on average

Acknowledgments

- Matteo Giantomassi
- Henrique Miranda
- Michiel Van Setten
- David Waroquiers
- Yannick Gillet
- Xavier Gonze
- Geoffroy Hautier
- Gian-Marco Rignanese

UCLouvain

- Shyam Dwaraknath
- Donald Winston
- Kristin Persson

MATERIALS PROJECT

Thank you for your attention