AIIDA-FLEUR PLUGIN
A robust way to perform your Fleur calculations

11. SEPTEMBER 2019  I  VASILY TSEPLYAEV, JENS BRÖDER, DANIEL WORTMANN
Outline

1. Why use AiiDA
2. AiiDA Data types
3. Calculations
4. Workchains
5. Tutorial instructions
Main problems of a computational scientist:

- Not automatised ➔ human mistakes
- Data badly organised ➔ easy to loose
- Exact steps are not clear ➔ hardly reproducible by others

### PHASE 1
RUN A CALCULATION ? PROFIT

### PHASE 2

### PHASE 3

---

Fleur input generator calculation with AiiDA

```xml
<fleurInput fleurInputVersion="0.29">
  <comment>
    A Fleur input generator calculation with aiida
  </comment>
  <calculationSetup>
    <cutoffs Kmax="3.80000000" Gmax="10.90000000" GmaxXC="9.10000000" numbands="40"/>
    <scfLoop itmax="9" minDistance=".00001000" maxIterBroyd="99" imix="Anderson" alpha=".05000000" preconditioning_param="0.0" spinf="2.00000000"/>
    <coreElectrons ctail="T" frcor="F" kcrel="0" coretail_lmax="0"/>
    <magnetism jspins="2" l_noco="F" swsp="F" lflip="F"/>  
    <soc theta="0.00000000" phi="0.00000000" l_soc="F" spav="F"/>
    <prodBasis gcutm="3.10000000" tolerance=".0000100000" ewaldlambda="3.00000000" lexp="16" bands="0"/>
    <nocoParams l_mperp="F" l_constr="F" mix_b="0.00000000"/>
    <ldaU l_linMix="F" mixParam=".05000000" spinf="1.00000000"/>  
    <bzIntegration valenceElectrons="48.00000000" mode="hist" fermiSmearingEnergy="0.00100000">  
      <kPointList posScale="1.0000000000" weightScale="1.0000000000" count="30"/>
    </kPointList>
    </bzIntegration>
    <energyParameterLimits ellow="-1.80000000" elup=".50000000"/>  
  </calculationSetup>
</fleurInput>
```
Automatisation using AiiDA

Submit and parse calculations

Supercomputer

Keep provenance

Database

Data → Calc → Data → Calc

Data

Data

result

AiiDA-FLEUR

Store data

Repository
Plugin Structure

Data
- store and work with input/output data
  - inp.xml
  - out.xml

Calculation
- run and parse a calculation
  - Inpgen
  - Fleur

WorkChain
- complex algorithms to get particular result
  - Equation of state
  - Magnetic anisotropy
**AiiDA data types**

### Python-like types

- Int
- Float
- Str
- Dict

A nested set of key: value pairs:

```python
{'conv_mode': 'density',
 'loop_count': 1,
 'total_energy': -90511.187617666,
 'force_largest': 0.0,
 'workflow_name': 'FleurScfWorkChain'}
```

### Structure types

- StructureData
- KpointsData

A k-point grid:

### File types

- FolderData
- RemoteData
- SinglefileData

A local folder

An remote folder
AiiDA-Fleur input files

FolderData containing an input file

String file name to be removed

FleurinpData

get_kpointsdata
KpointsData
k-mesh from inp.xml

get_structuredata
StructureData
structure from inp.xml

get_parameterdata
Dict
parsed inp.xml

del_file
set_file

No methods changing the content of inp.xml

Member of the Helmholtz Association
11 September 2019
Inp.xml modifications

Stored FleurinpData is sealed → one needs to create a new one

1. Initialise FleurinpModifier object:

   ```python
   from aiida_fleur.data.fleurinpmodifier import FleurinpModifier
   modification = FleurinpModifier(fleurinp)
   ```

2. Register modifications:

   ```python
   modification.set_inpchanges({'theta': 1.57079})
   modification.set_inpchanges({'phi': 0.33079})
   modification.set_species('W-1', {'mtSphere': {'radius': 3.5}})
   modification.undo()
   ```

3. Apply modifications:

   ```python
   modified_fleurinp = modification.freeze()
   ```
# Registration methods

One can make any modification to inp.xml: change/delete/create a tag

<table>
<thead>
<tr>
<th>XML methods</th>
<th>Shortcuts</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Total number of iterations</strong></td>
<td></td>
</tr>
<tr>
<td><code>xml_set_first_attribv('fleurInput/calculationSetup/scfLoop', 'itmax', 29)</code></td>
<td><code>set_inpchanges({'itmax': 29})</code></td>
</tr>
<tr>
<td><strong>Muffin tin radius</strong></td>
<td></td>
</tr>
<tr>
<td><code>xml_set_first_attribv('fleurInput/atomSpecies/species[@name = &quot;W-1&quot;]/mtSphere', 'radius', 2.2)</code></td>
<td><code>set_specie('W-1', {'mtSphere' : {'radius' : 2.2}})</code></td>
</tr>
<tr>
<td><strong>beta noco parameter</strong></td>
<td></td>
</tr>
<tr>
<td><code>xml_set_first_attribv('fleurInput/atomGroups/atomGroup/', 'atomGroup[@species = &quot;W-1&quot;]/nocoParams', 'beta', 1.57)</code></td>
<td><code>set_atomgr_att({'nocoParams': [{'beta', 1.57}]], species='W-1')</code></td>
</tr>
</tbody>
</table>

**Flexible but not convenient** | **Convenient but not flexible**
Plugin Structure

Data
- store and work with input/output data
  - inp.xml
  - out.xml

Calculation
- run and parse a calculation
  - Inpgen
  - Fleur

WorkChain
- Complex algorithms to get particular result
  - Equation of state
  - Magnetic anisotropy
AiiDA engine types

**Code**

Executable

Used by CalcJobs

**CalcJob**

Calculation process

Runs a single Fleur calculation parse results

**WorkChain**

Chain of processes

Runs a sequence of Fleur calculations
Calculations (CalcJob)

**inpgen**

- Structure
- Calculation parameters: input for inpgen
- Code

**inpgen calculation**

- Parsed inp.xml

**FLEUR**

- FleurinpData
- Previous calculation: charge density
- Code

**Fleur calculation**

- Output folder
- Parsed out.xml
Example of calculation outputs

**inpgen**

**Parsed inp.xml**

(a part is shown)

```python
'files': ['inp.xml'],
'inp_dict': {
    'cell': {'filmLattice': {
        'dVac': 10.31,
        'scale': 1.0,
        'dTilda': 13.62,
        'latnam': 'any',
        'bravaisMatrix': {
            'row-1': '5.3011797029 0.0000000000 0.0000000000',
            'row-2': '0.0000000000 7.4970000330 0.0000000000',
            'row-3': '0.0000000000 0.0000000000 11.3011800234'},
        'vacuumEnergyParameters': {
            'spinUp': '-.25000000',
            'vacuum': '2',
            'spinDown': '-.25000000'}},
    'symmetryOperations': {
        'symOp': {
            'row-1': '-1 0 0 .0000000000',
            'row-2': '0 -1 0 .0000000000',
            'row-3': '0 0 1 .0000000000'}}
}
```

**FLEUR**

**Parsed out.xml**

(a part is shown)

```python
'energy': -536262.57517656,
'bandgap': 0.0014428048,
'end_date': {'date': '2019/08/26', 'time': '13:19:38'},
'unparsed': [],
'walltime': 276,
'start_date': {'date': '2019/08/26', 'time': '13:15:02'},
'CalcJob_uuid': 'e9522c11-3b38-41c5-ad86-53bba78add85',
'energy_units': 'eV',
'fermi_energy': 0.2355843143,
'spin_density': 1.10861e-05,
'bandgap_units': 'eV',
'force_largest': 0.0,
'energy_hartree': -19707.286309577
```
Plugin Structure

Data

- store and work with input/output data
- inp.xml
- out.xml

Calculation

- run and parse a calculation
- Inpgen
- Fleur

WorkChain

- Complex algorithms to get particular result
- Equation of state
- Magnetic anisotropy

11 September 2019
Workchain

Sometimes to achieve the result one needs to perform a chain of calculations

Equation of States:

Energy
Energy
Energy
Energy
Energy

input parameters

Equation of states

Relative Volume
SCF WorkChain

1. Run inpigen
2. Change FleurinpData
3. Check convergence
   (density, energy or force)
4. Run Fleur
"conv_mode": "density",
"distance_charge": 1.13714e-05,
"distance_charge_all": [
  29.9211648204,

],
"distance_charge_units": "me/bohr^3",
"force_diff_last": "can not be determined",
"force_largest": 0.0,
"iterations_total": 30,
"last_calc_uid": "4c0a1cf4-932b-4b88-96ea-8ff443bad13c",
"loop_count": 1,
"material": "FePt2",
"total_energy": -38143.906981714,
"total_energy_all": [
  -38144.008129573,

],
"total_energy_units": "Htr",
"total_wall_time": 545,
"total_wall_time_units": "s",
"workflow_name": "FleurScfWorkChain",
"workflow_version": "0.4.0"
Workchain Hierarchy

Fleur Calculation

Fleur Base WorkChain

SCF

DMI Dispersion

MAE

Spin Spiral Dispersion

Magnetic Force theorem

Currently unavailable

DOS

Band

Equation of states

Geometry optimisation

Spin Spiral Dispersion

Magnetic SCF

MAE
Base workchain

SCF workchain does not call Fleur Calculation directly

FleurBaseWorkChain wraps FleurCalculation and deals with possible failures: technical problems and memory issues
HIGHER-LEVEL WORKCHAINS
Energies are not the same for:

What is a qualitative energy difference?
MAE_conv Workchain

Initialisation

WorkChain parameters

SCF

~50 iterations with SOC
~50 iterations with SOC
~50 iterations with SOC
~50 iterations with SOC
~50 iterations with SOC

output
MAE Workchain

Initialisation

Reference calculation

Force theorem

output
Spin spiral dispersion workchain

\[\text{SSDisp}\]

**Find E(q)**

**Converge all q-vectors**

**SSDisp\_conv**

**Force theorem**

**SSDisp**
SSDisp_conv Workchain

Initialisation

WorkChain parameters

SCF

~50 iterations with SOC

~50 iterations with SOC

~50 iterations with SOC

~50 iterations with SOC

~50 iterations with SOC

output
SSDisp Workchain

Initialisation

+ WorkChain parameters

Reference calculation

~50 iterations with SOC

50 iterations

Force theorem

1 iteration with SOC

1 iteration with SOC

1 iteration with SOC

1 iteration with SOC

1 iteration with SOC

Output
Welcome to the AiiDA-FLEUR’s documentation!

The AiiDA-FLEUR python package enables the use of the all-electron Density Functional Theory (DFT) code FLEUR (http://www.flapw.de) with the AiiDA framework (http://www.aiida.net).
Tutorial

6 tutorial chapters:

1. verdi commands and AiiDA data types
2. AiiDA-Fleur input file
3. FleurinpData modifications
4. inpgen code
5. Fleur code
6. Workchains
Jupyter notebook

Learn by pressing Ctrl+Enter
Introduction to AiiDA

Note: All commands starting with a `v` are bash commands. Python notebooks provide this handy way of running shell/bash commands inside a python environment.

Also read through the comments

```
In [ ]:

# example
ls
```

Verdi Commands

In this part of the tutorial, you will learn some basics about the AiiDA framework. Get familiar with some useful `verdi` commands.

The command-line utility `verdi` is one of the most common ways to interact with AiiDA. Verdi with its subcommands enables a variety of operations such as inspecting the status of ongoing or terminated calculations, showing the details of calculations, computers, codes, or data structures, access the input and the output of a calculation, etc.

Similar to the bash shell, verdi command support Tab completion. Try right now to type `verdi` in a terminal of the AiiDA container and tap Tab twice to have a list of subcommands. Whenever you need the explanation of a command type `verdi help` or add `-h` flag if you are using any of the verdi subcommands.
Modifying a Fleur input file

To modify an existing `FleurinpData`, we need to load it first. Please, do in via `load_node()` function and proceed to the next section:

```
In []:
# you need to modify this - insert a PK of any FleurinpData
fleurinp = load_node('FLEURINP_PK')
```

**FleurinpModifier**

The stored `FleurinpData` cannot be modified in-place because it was sealed when it was stored in the database. Therefore we always need to create a new `FleurinpData` object to change an existing one.

To make changes and store the result in the database, AiiDA-Fleur contains a `FleurinpModifier` class.

To start a process of `FleurinpData` modification, we need to import the `FleurinpModifier` class first and initialise an instance:

```
In []:
# we can also import it using DataFactory:
# from aida.plugins import DataFactory
# FleurinpModifier = DataFactory('fleur.fleurinpmodifier')

from aida_fleur.data.fleurinpmodifier import FleurinpModifier

fleurmofde = FleurinpModifier(fleurinp)
```
Scripting tasks

Jupyter notebooks are a convenient way to learn and run some jobs. However, I personally prefer using and running Python scripts in a terminal for my work. As a final task of each tutorial, beginning from this one, you will be asked to construct a small Python script and run it in a terminal.

First, you need to start a new terminal and type in:

```bash
$workon aida
```

To execute a Python script, run

```bash
$(aida) verdi run NAME_OF_THE_SCRIPT.py
```

1. Run an inp genotype calculation for Si

Write a script that submits `FleurinpgenCalculation` for a Si structure given in the attached `Si.cif` file. Try to set up `parameters` dictionary to use parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>atom-rmt</td>
<td>2.23</td>
<td>muffin-tin radius</td>
</tr>
<tr>
<td>comp-kmax</td>
<td>3.84</td>
<td>plane wave cut-off</td>
</tr>
<tr>
<td>kpt-div1</td>
<td>2</td>
<td>number of kpts along x</td>
</tr>
<tr>
<td>kpt-div2</td>
<td>2</td>
<td>number of kpts along y</td>
</tr>
</tbody>
</table>
Good luck with tutorials!

Any questions ➔ ask me or

Anoop
Acknowledgements

AiiDA & Materials Cloud Teams

- Oscar D. Arbeleaz (EPFL)
- Marco Borelli (EPFL)
- Valeria Granata (EPFL)
- Sebastiaan P. Huber (EPFL)
- Leonid Kahle (EPFL)
- Boris Kozinsky (BOSCH)
- Snehal P. Kumbhar (EPFL)
- Nicola Marzari (EPFL)
- Elsa Passaro (EPFL)

- Giovanni Pizzi (EPFL)
- Thomas Schulthess (ETHZ, CSCS)
- Berend Smit (EPFL)
- Leopold Talirz (EPFL)
- Daniele Tomerini (EPFL)
- Joost VandeVondele (ETHZ, CSCS)
- Casper Welzel (EPFL)
- Aliaksandr Yakutovich (EPFL)
Summary

• inp.xml is represented as FleurinpData

• There is a special class to modify FleurinpData

• inpgen calculation generates FleruinData

• A set of key-turn WorkChains is available