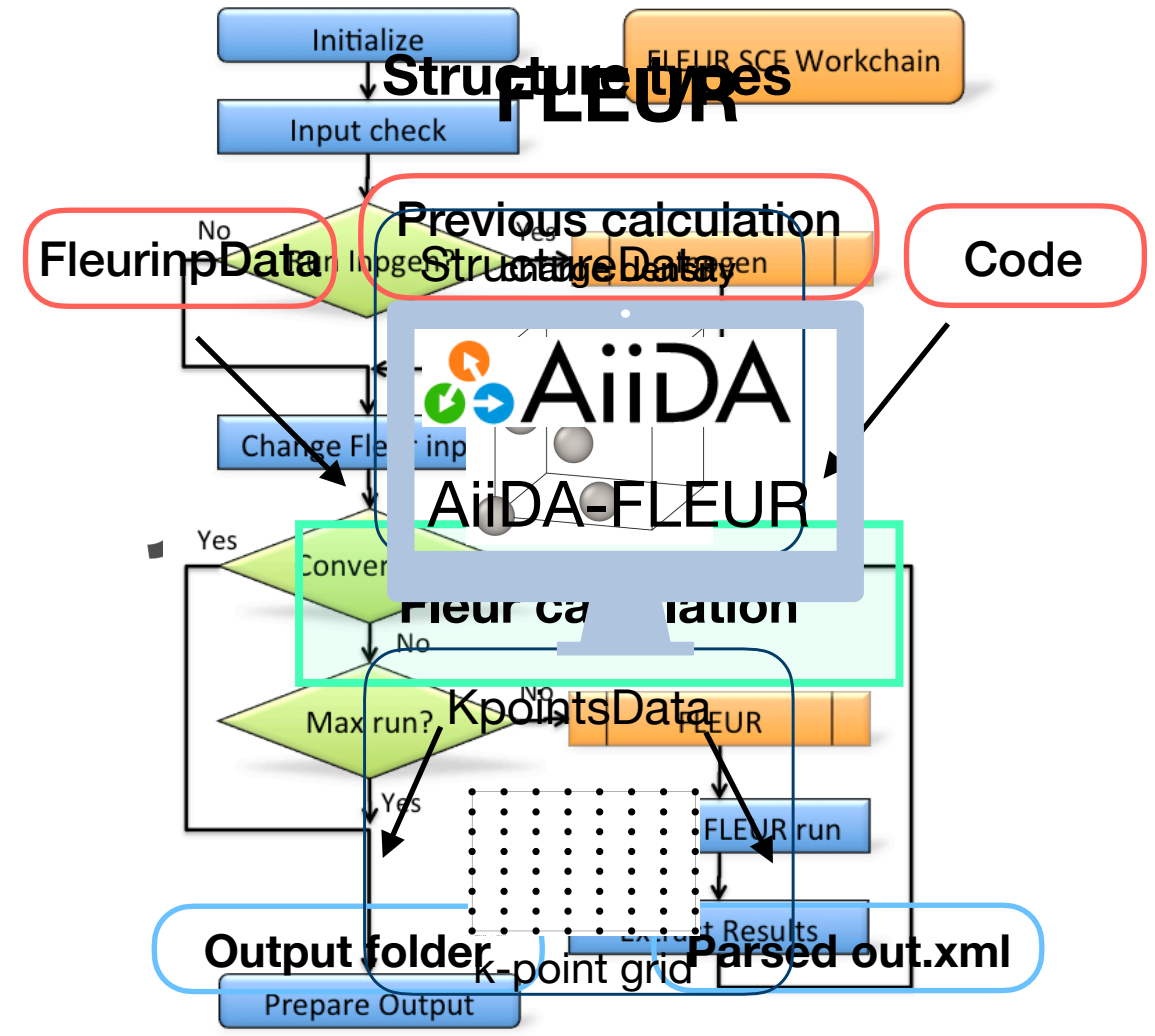
AIIDA-FLEUR PLUGIN

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



Automatisation using AiiDA

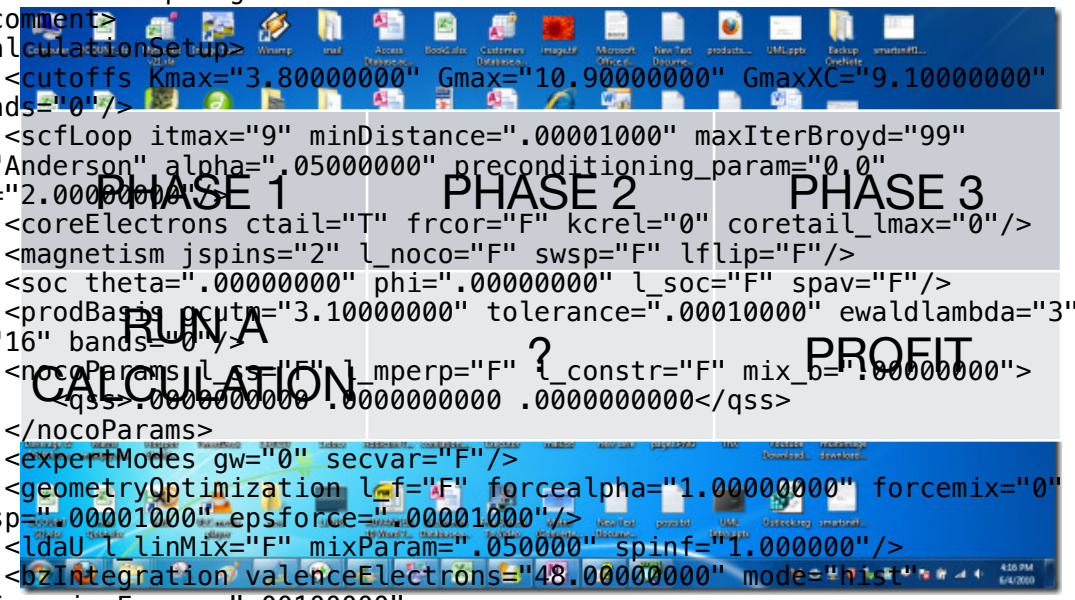
Main problems of a computational scientist:

Not automatised ➜ human mistakes

Data badly organised ➜ easy to loose

Exact steps are not clear ➜ hardly reproducible by others

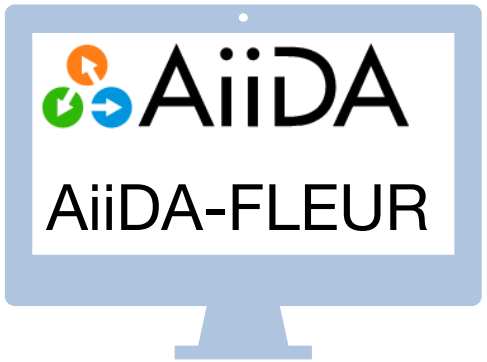
```
<fleurInput fleurInputVersion="0.29">
   <comment>
      A Fleur input generator calulation with aiida
   </comment>
   <calculationSetup>
      <cutoffs Kmax="3.80000000" Gmax="10.90000000" GmaxXC="9.10000000"
numbands="0"/>
      <scfLoop itmax="9" minDistance=".00001000" maxIterBroyd="99"
imix="Anderson" alpha=".05000000" preconditioning_param="0.0"
spinf="2.000000000"/>
      <coreElectrons ctail="T" frcor="F" kcrel="0" coretail_lmax="0"/>
      <magnetism jspins="2" l_noco="F" swsp="F" lflip="F"/>
      <soc theta=".00000000" phi=".00000000" l_soc="F" spav="F"/>
      <prodBasis gcutm="3.10000000" tolerance=".00010000" ewaldlambda="3"
lexp="16" bands="0"/>
      <nocoParams l_ss="F" l_mperp="F" l_constr="F" mix_b=".00000000">
         <qss>.0000000000 .0000000000 .0000000000</qss>
      </nocoParams>
      <expertModes gw="0" secvar="F"/>
      <geometryOptimization l_f="F" forcealpha="1.00000000" forcemix="0"
epsdisp=".00001000" epsforce=".00001000"/>
      <ldaU l_linMix="F" mixParam=".050000" spinf="1.000000"/>
      <bzIntegration valenceElectrons="48.00000000" mode="hist"
fermiSmearingEnergy=".00100000">
         <kPointList posScale="1.00000000" weightScale="1.00000000"
count="30">
         </kPointList>
      </bzIntegration>
      <energyParameterLimits ellow="-1.80000000" elup=".50000000"/>
   </calculationSetup>
   <cell>
```



MAX DRIVING THE EXASCALE TRANSITION

JÜLICH Forschungszentrum

Automatisation using AiiDA



Submit and parse calculations

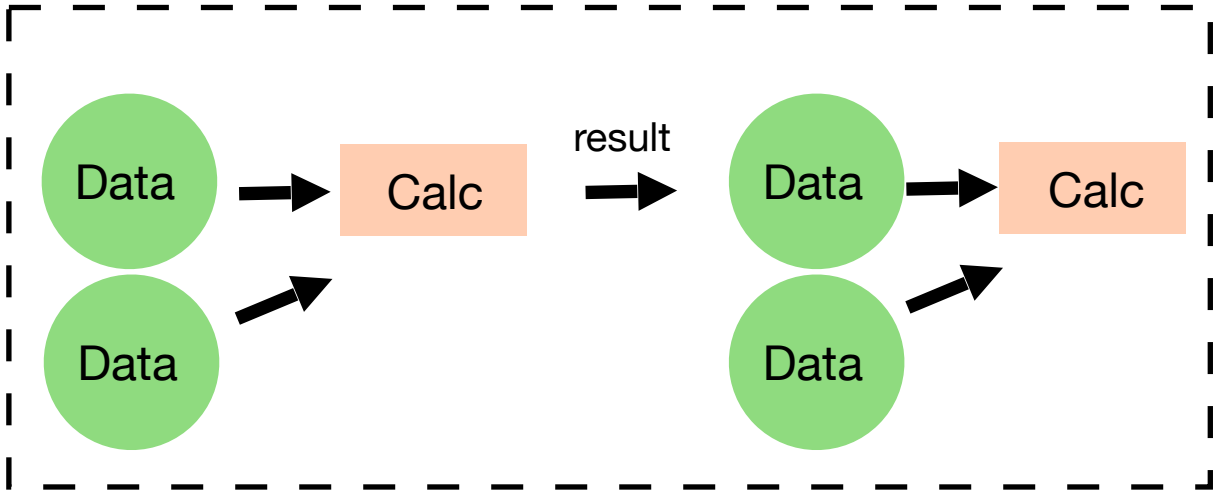
Supercomputer

Keep provenance

Store data

Repository

Database



MAX DRIVING THE EXASCALE TRANSITION

JÜLICH Forschungszentrum

Plugin Structure

Data	Calculation	WorkChain
store and work with input/output data	run and parse a calculation	complex algorithms to get particular result
• inp.xml • out.xml	• Inpgen • Fleur	• Equation of state • Magnetic anisotropy

AiiDA data types

Python-like types

Int

Float

Str

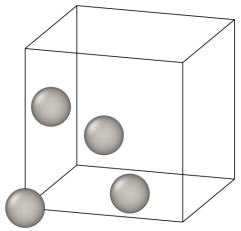
Dict

a nested set of key: value pairs

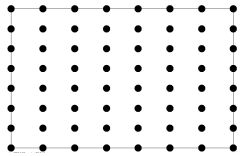
```
{'conv_mode': 'density',
 'loop_count': 1,
 'total_energy': -90511.187617666,
 'force_largest': 0.0,
 'workflow_name': 'FleurScfWorkChain'}
```

Structure types

StructureData



KpointsData



k-point grid

File types

FolderData

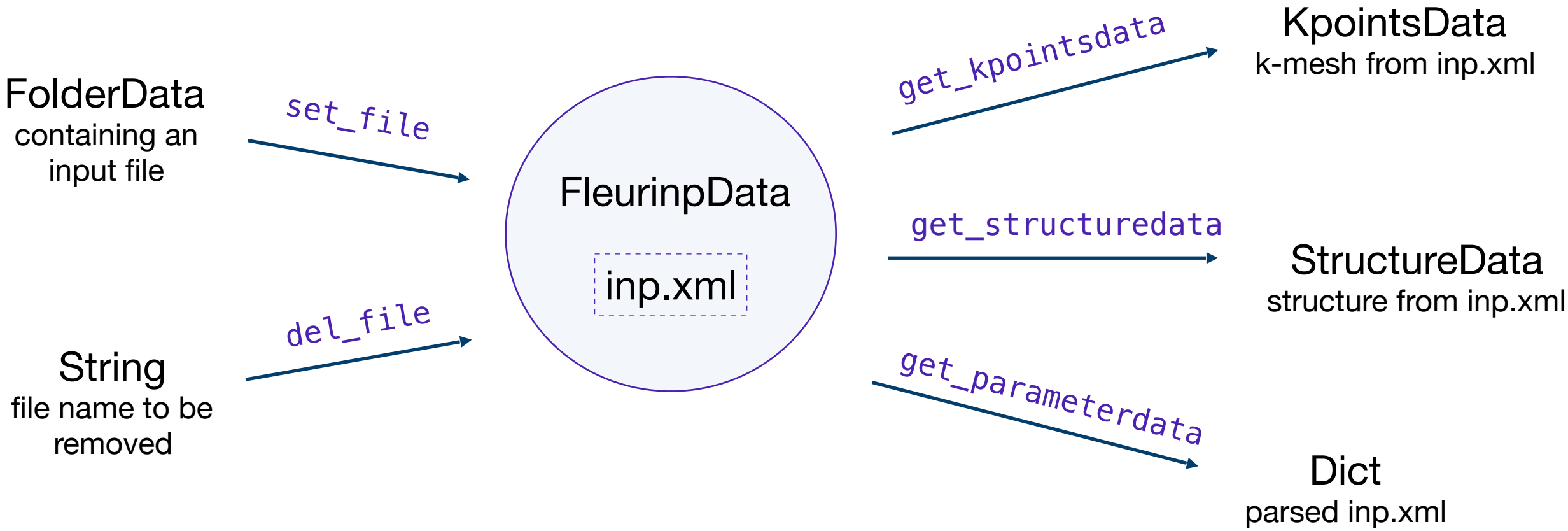
a local folder

RemoteData

a remote folder

SinglefileData

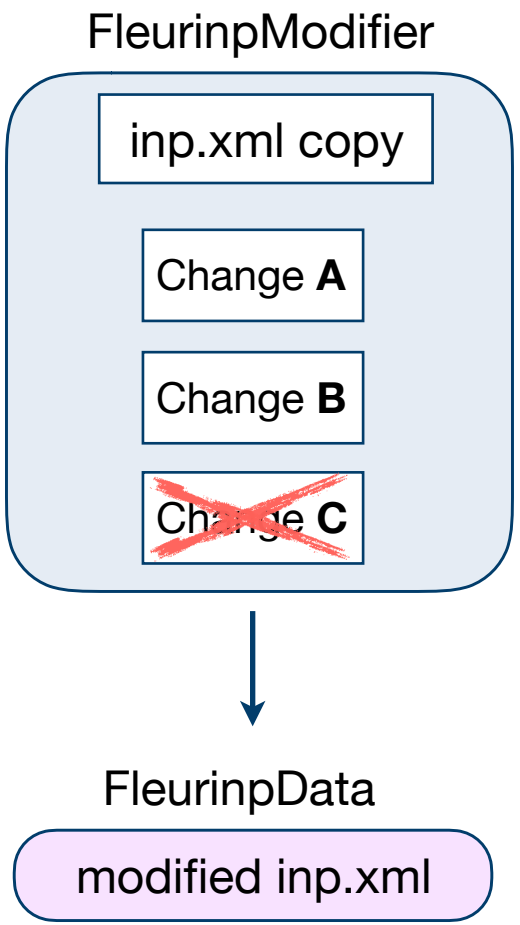
AiiDA-Fleur input files



No methods changing the content of inp.xml

Inp.xml modifications

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



1. Initialise FleurinpModifier object:

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

2. Register modifications:

```
modification.set_inpchanges({'phi': 0.33079})

modification.set_inpchanges({'theta': 1.57079})

modification.set_species('W-1', {'mtSphere' : {'radius' : 3.5}})
modification.undo()
```

3. Apply modifications:

```
modified_fleurinp = modification.freeze()
```

Registation methods

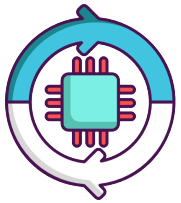
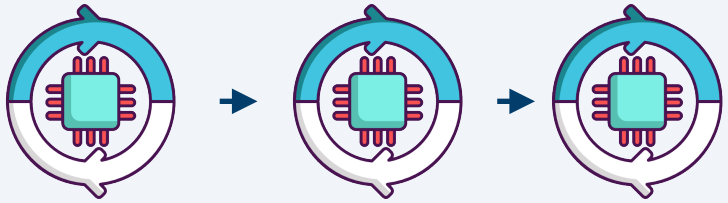
One can make any modification to inp.xml: change/delete/create a tag

XML methods	Shortcuts
Total number of iterations	
`xml_set_first_attribv('/fleurInput/calculationSetup/scfLoop', 'itmax', 29)`	`set_inpchanges({'itmax': 29})`
Muffin tin radius	
`xml_set_first_attribv('/fleurInput/atomSpecies/' 'species[@name = "W-1"]/mtSphere', 'radius', 2.2)`	`set_specie('W-1', {'mtSphere' : {'radius' : 2.2}})`
beta noco parameter	
`xml_set_first_attribv('/fleurInput/atomGroups/atomGroup/' 'atomGroup[@species = "W-1"]/nocoParams', 'beta', 1.57)`	`set_atomgr_att({'nocoParams': [('beta', 1.57)]}, species='W-1')`
Flexible but not convenient	Convenient but not flexible

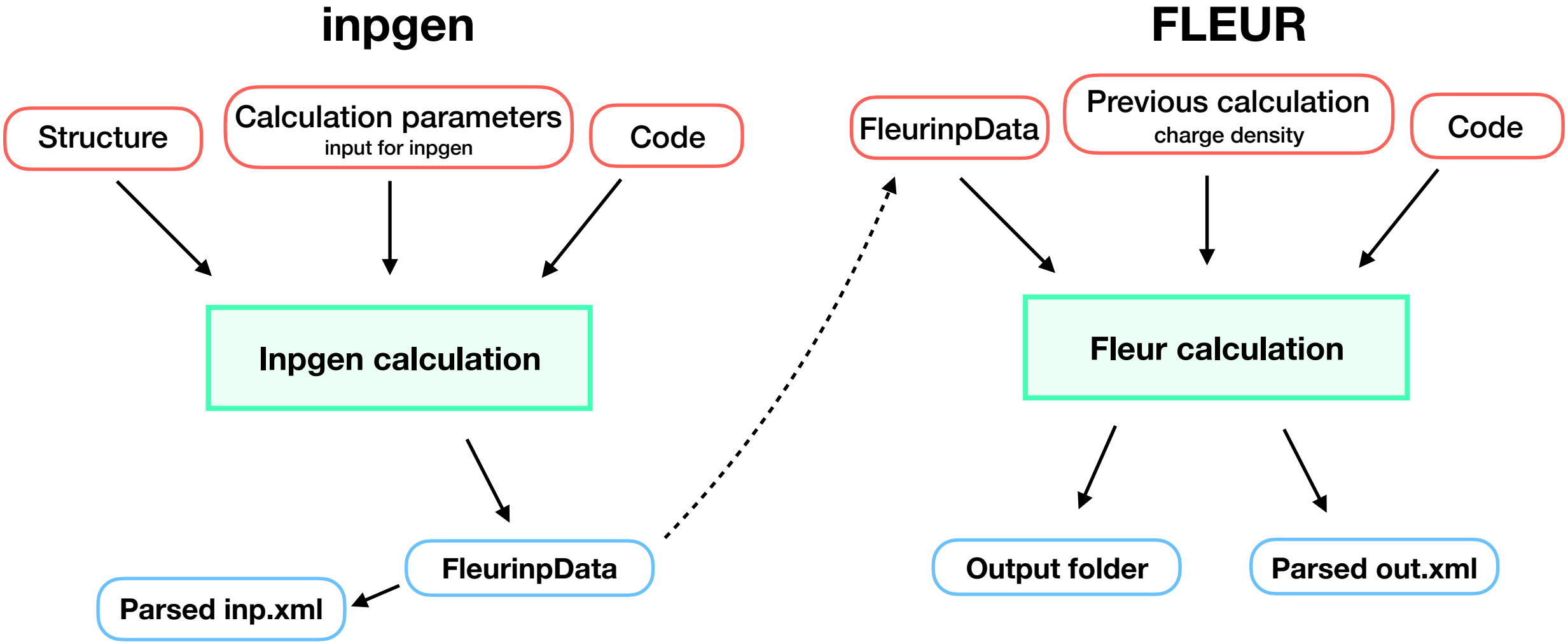
Plugin Structure

Data	Calculation	WorkChain
store and work with input/output data	run and parse a calculation	Complex algorithms to get particular result
• inp.xml • out.xml	• Inpgen • Fleur	• Equation of state • Magnetic anisotropy

AiiDA engine types

Code	Executable www.flapw.de fleur	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)



Example of calculation outputs

inpgen

Parsed inp.xml

(a part is shown)

```
'files': ['inp.xml'],
'inp_dict': {
   'cell': {'filmLattice': {
     'dVac': 10.31,
     'scale': 1.0,
     'dTilda': 13.62,
     'latnam': 'any',
     'bravaisMatrix': {
       'row-1': '5.3011797029 .0000000000 .0000000000',
       'row-2': '.0000000000 7.4970000330 .0000000000',
       'row-3': '.0000000000 .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)

```
'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
```

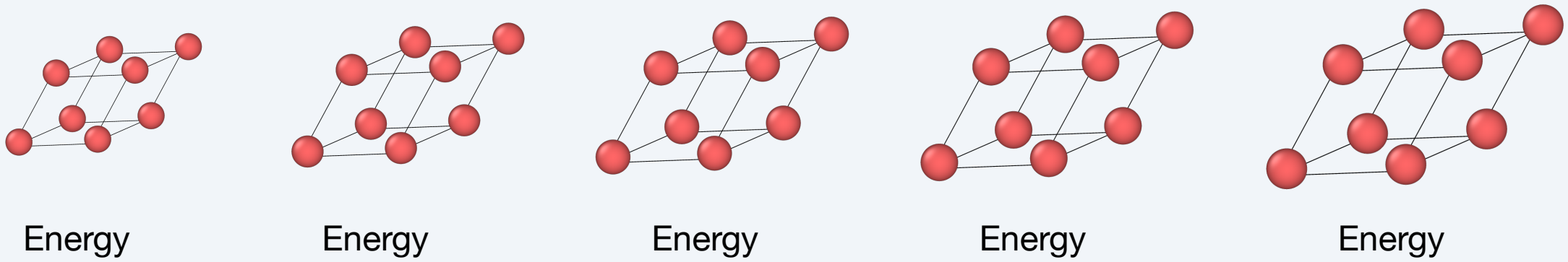
Plugin Structure

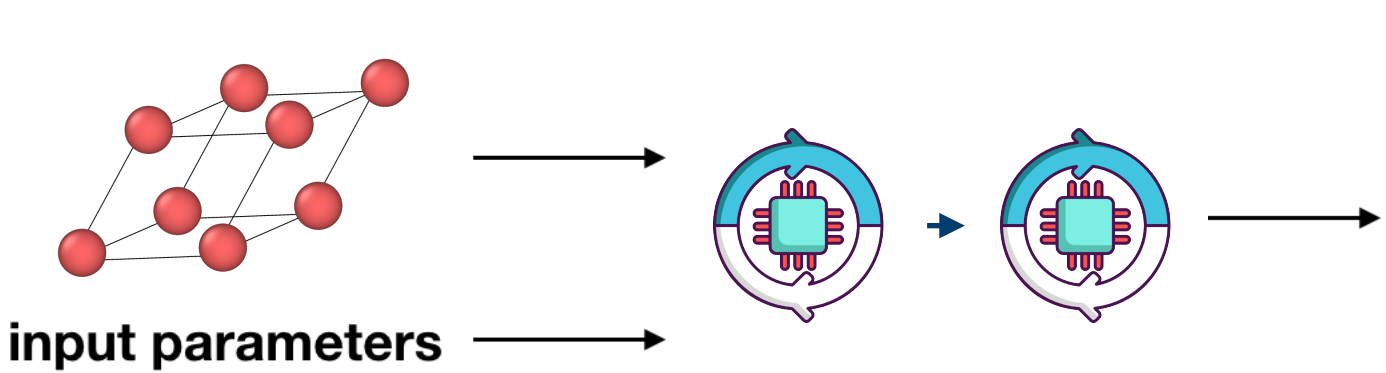
Data	Calculation	WorkChain
store and work with input/output data	run and parse a calculation	Complex algorithms to get particular result
• inp.xml • out.xml	• Inpgen • Fleur	• Equation of state • Magnetic anisotropy

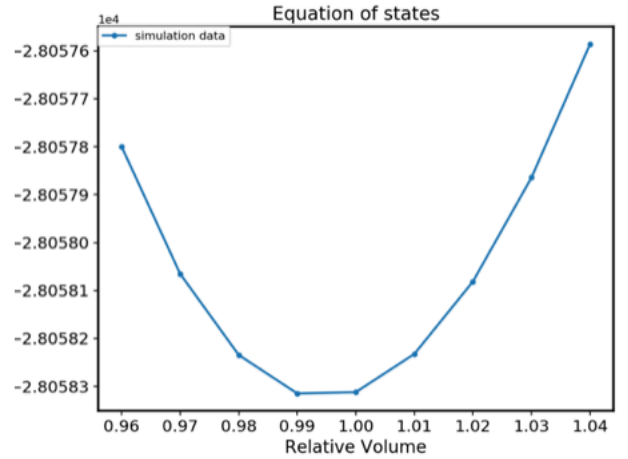
Workchain

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

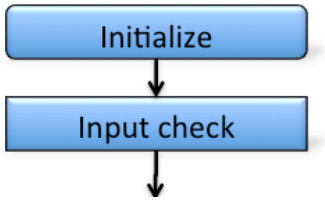
Equation of States:

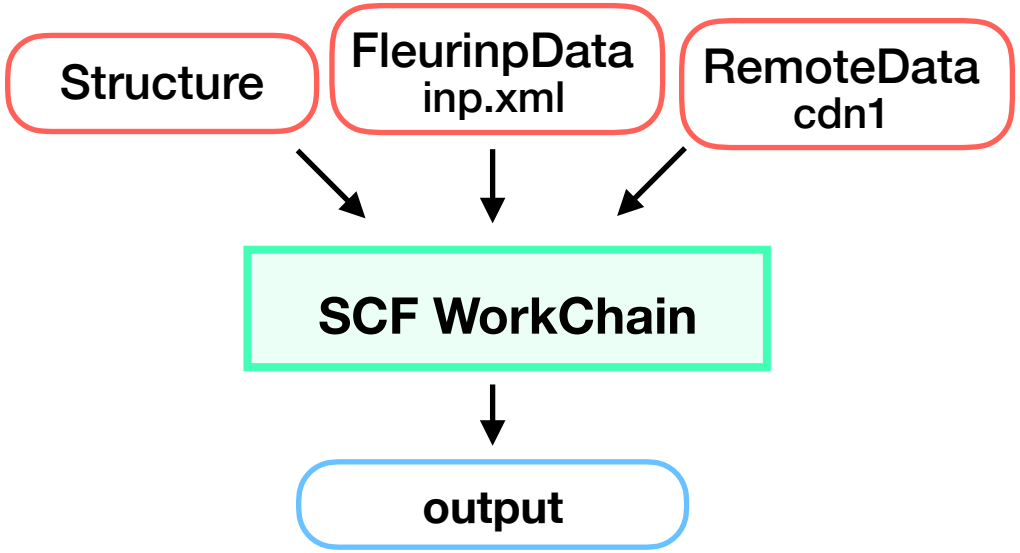






SCF WorkChain





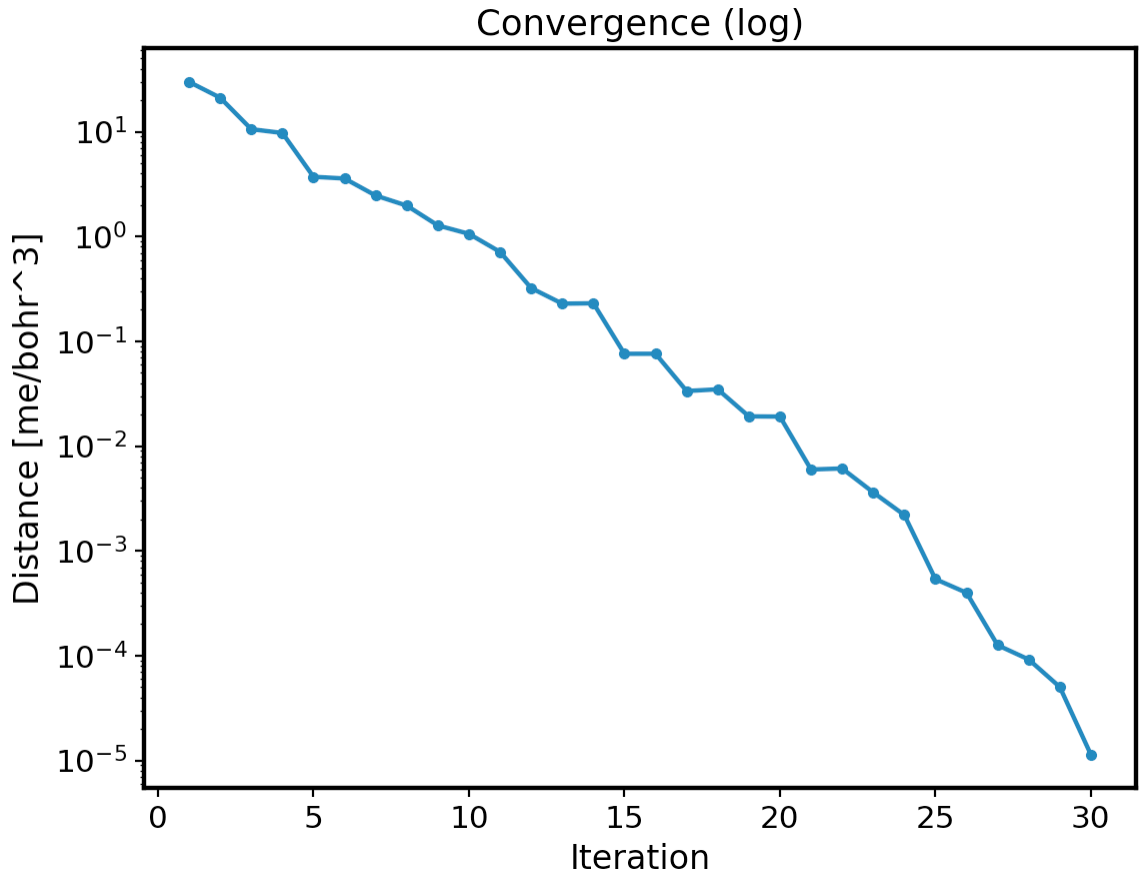
1. Run inpgen
2. Change FleurinpData
3. Check convergence
   (density, energy or force)
4. Run Fleur

SCF WorkChain

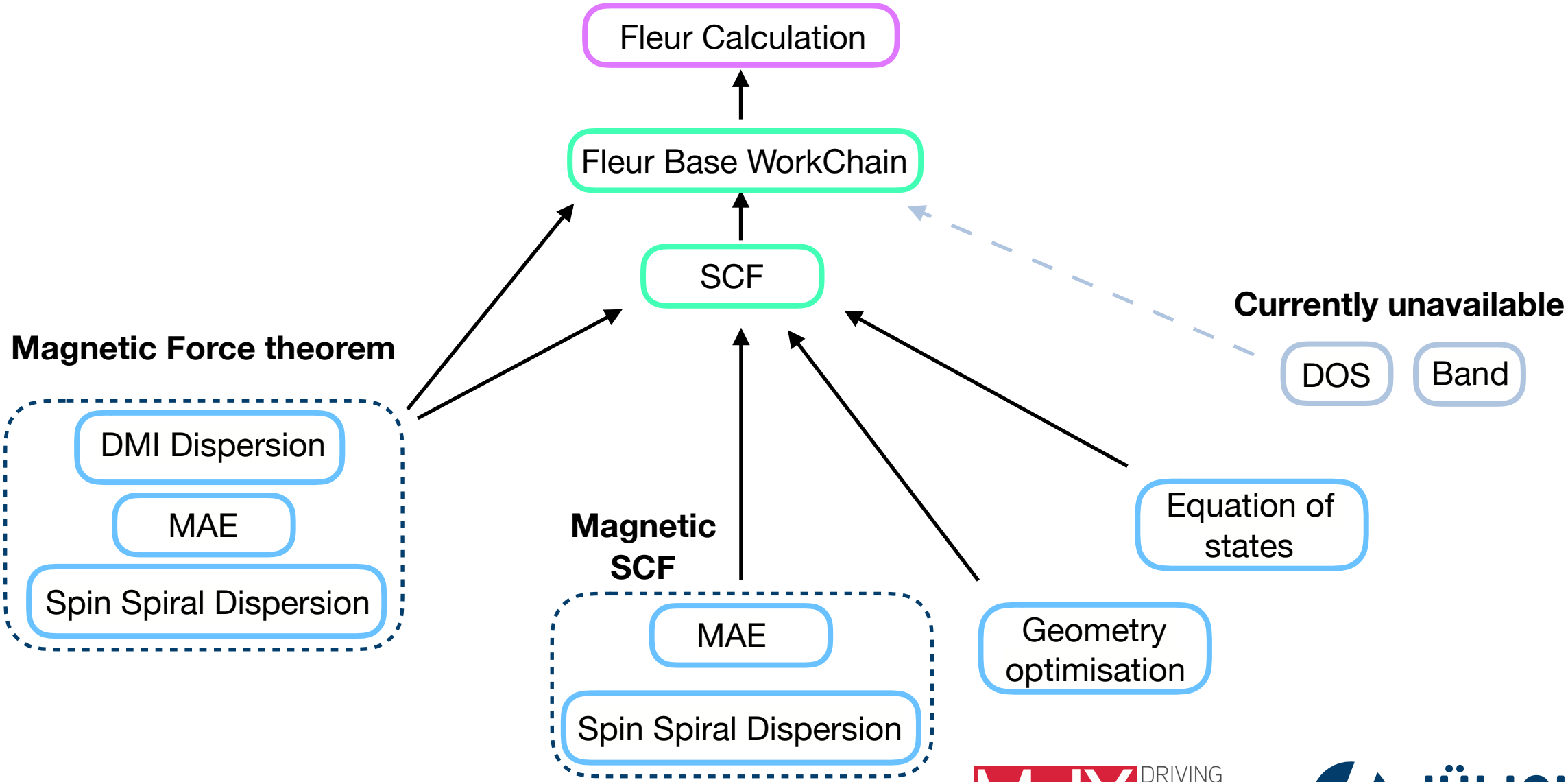
output


```
"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_uuid": "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"
```


plot_fleur()

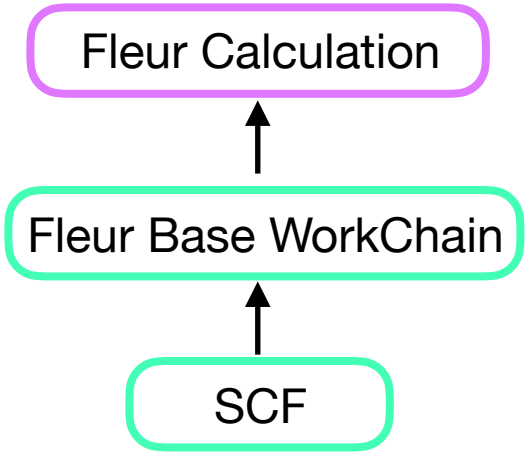


Workchain Hierarchy



Base workchain

SCF workchain does not call Fleur Calculation directly

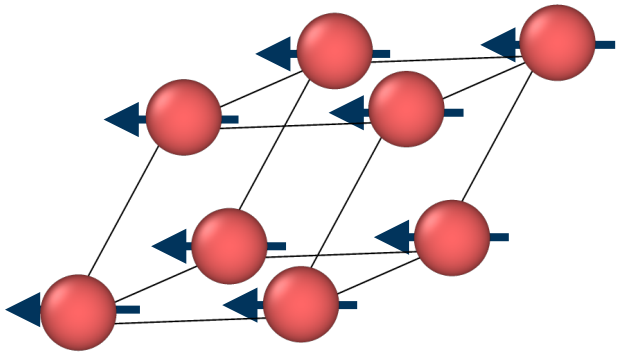


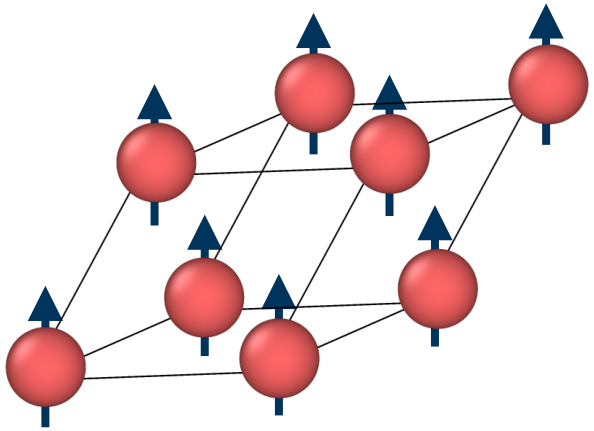
FleurBaseWorkChain wraps FleurCalculation and deals with possible failures:
technical problems and memory issues

HIGHER-LEVEL WORKCHAINS

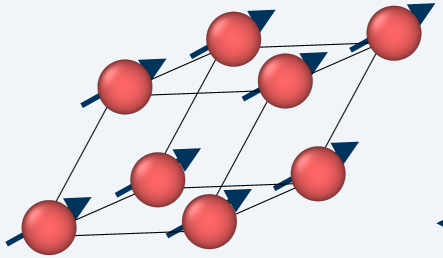
MAE Workchain

Energies are not the same for:

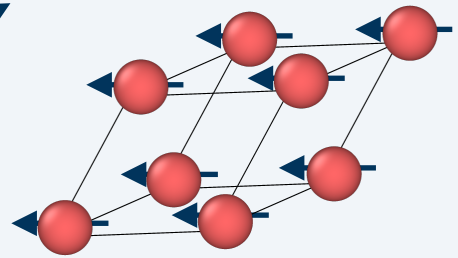




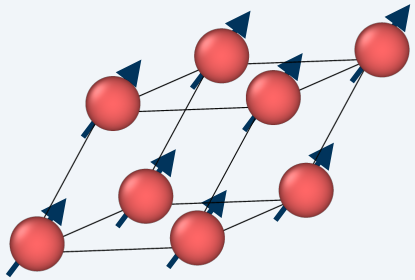
What is a qualitative energy difference?



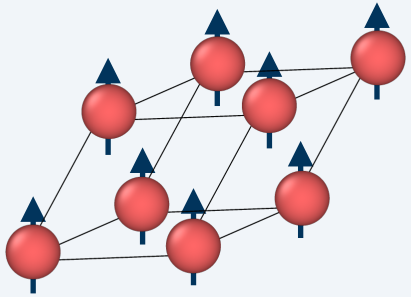
Energy



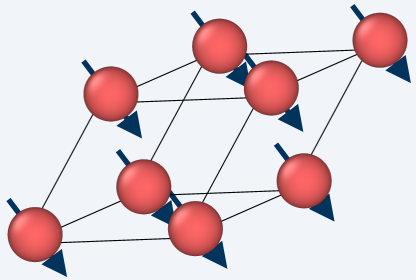
Energy



Energy

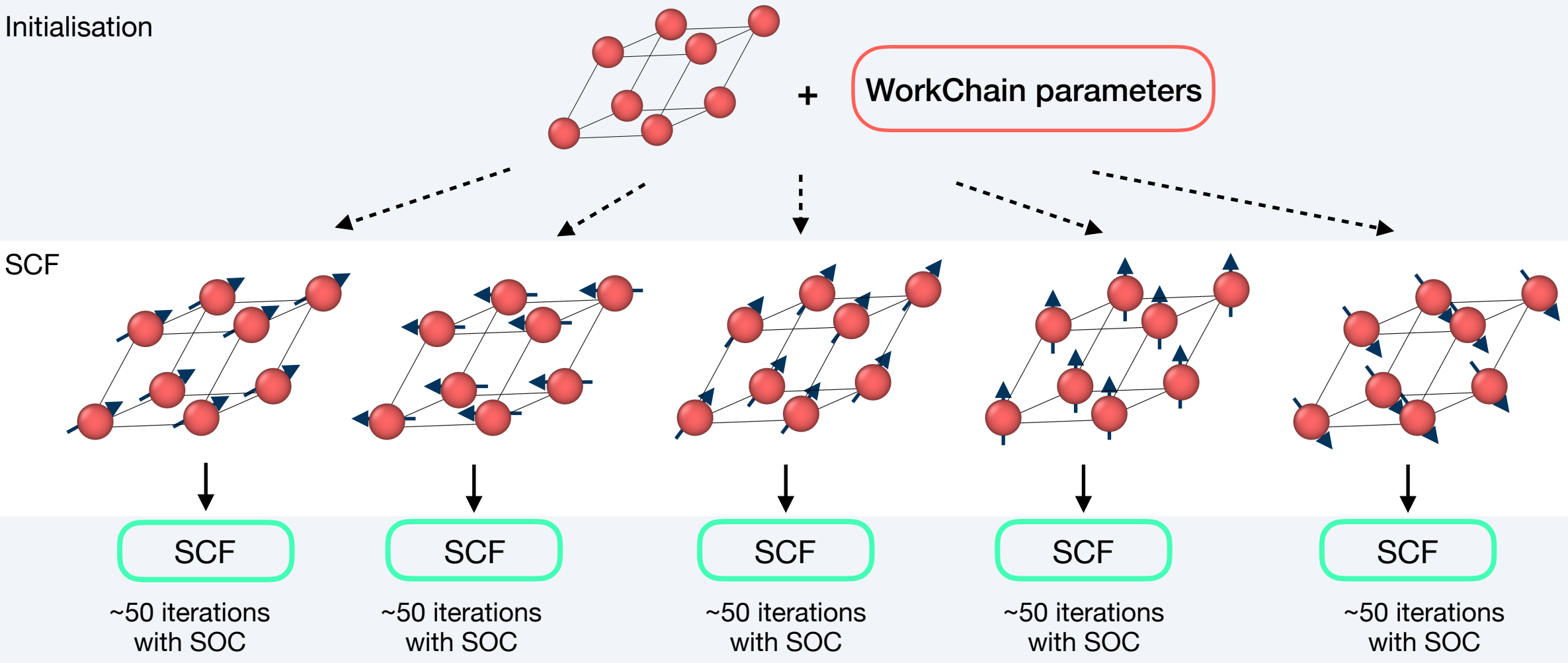


Energy



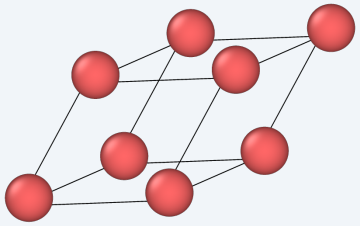
Energy

MAE_conv Workchain



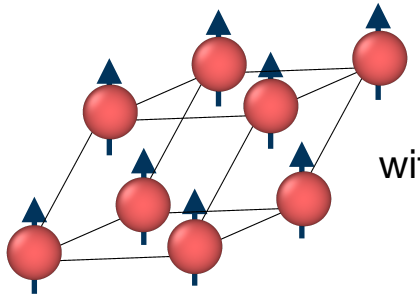
MAE Workchain

Initialisation



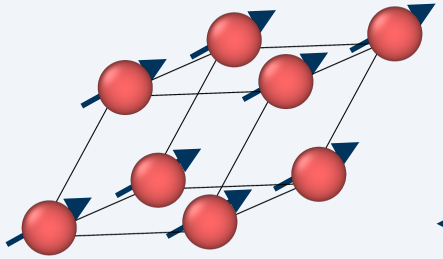
+ WorkChain parameters

Reference calculation

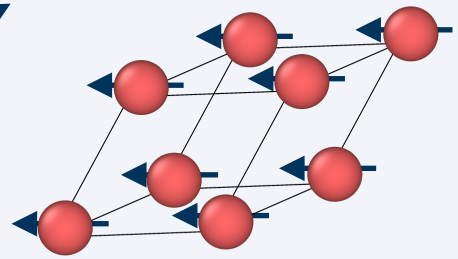


~50 iterations
with or without SOC

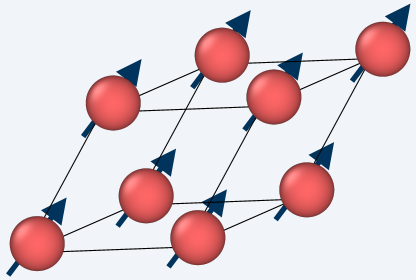
Force theorem



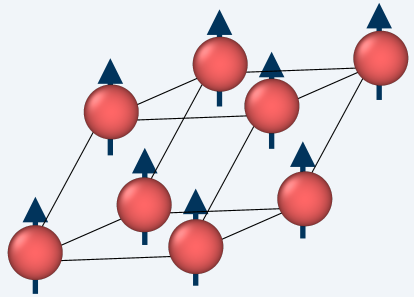
1 iteration
with SOC



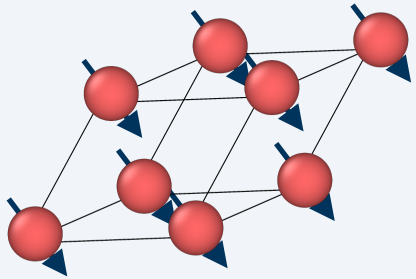
1 iteration
with SOC



1 iteration
with SOC



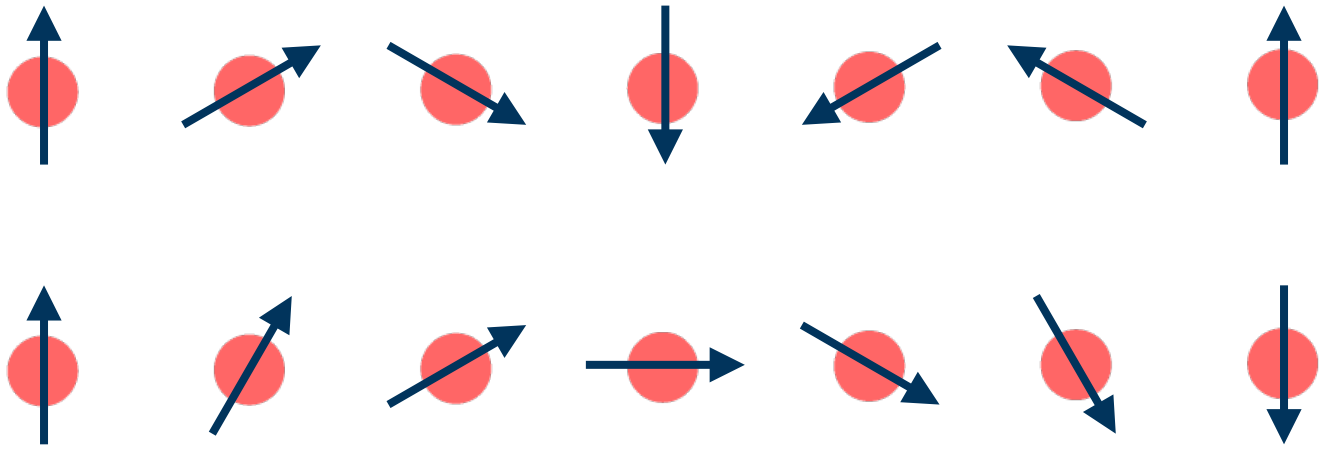
1 iteration
with SOC



1 iteration
with SOC

output

Spin spiral dispersion workchain



q_1

q_2

Find E(q)

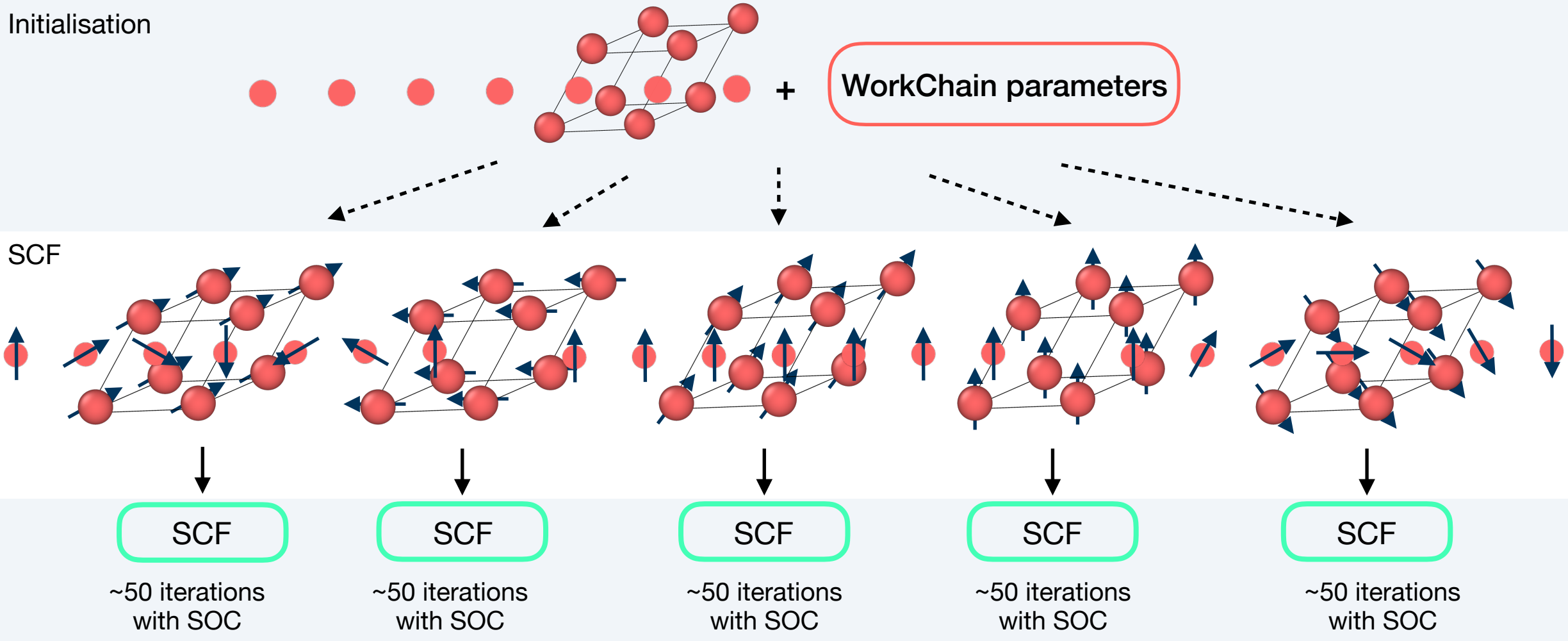
Converge all q-vectors

SSDisp_conv

Force theorem

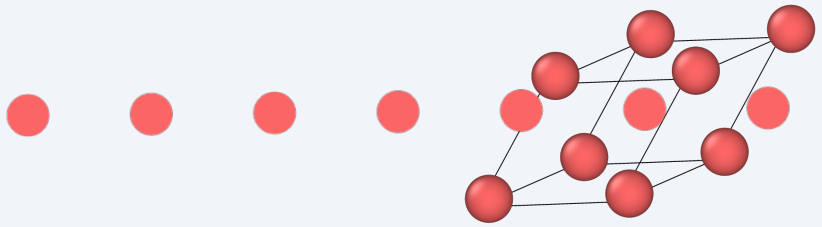
SSDisp

SSDisp_conv Workchain



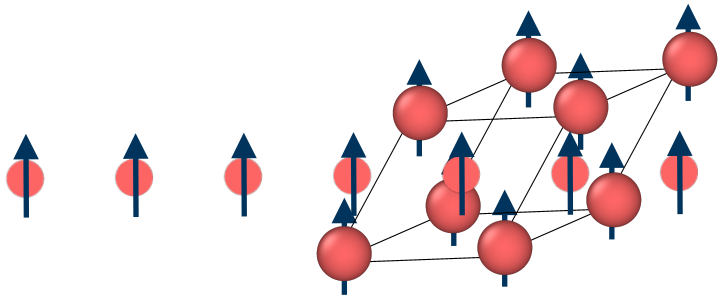
SSDisp Workchain

Initialisation



+ **WorkChain parameters**

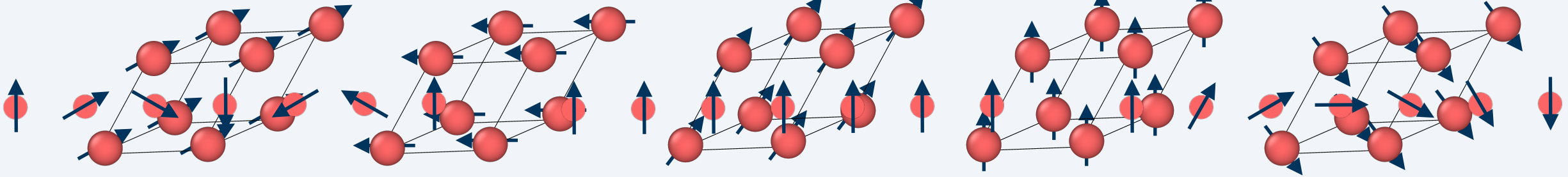
Reference calculation



~50 iterations with SOC

~50 iterations

Force theorem



1 iteration with SOC

1 iteration with SOC

1 iteration with SOC

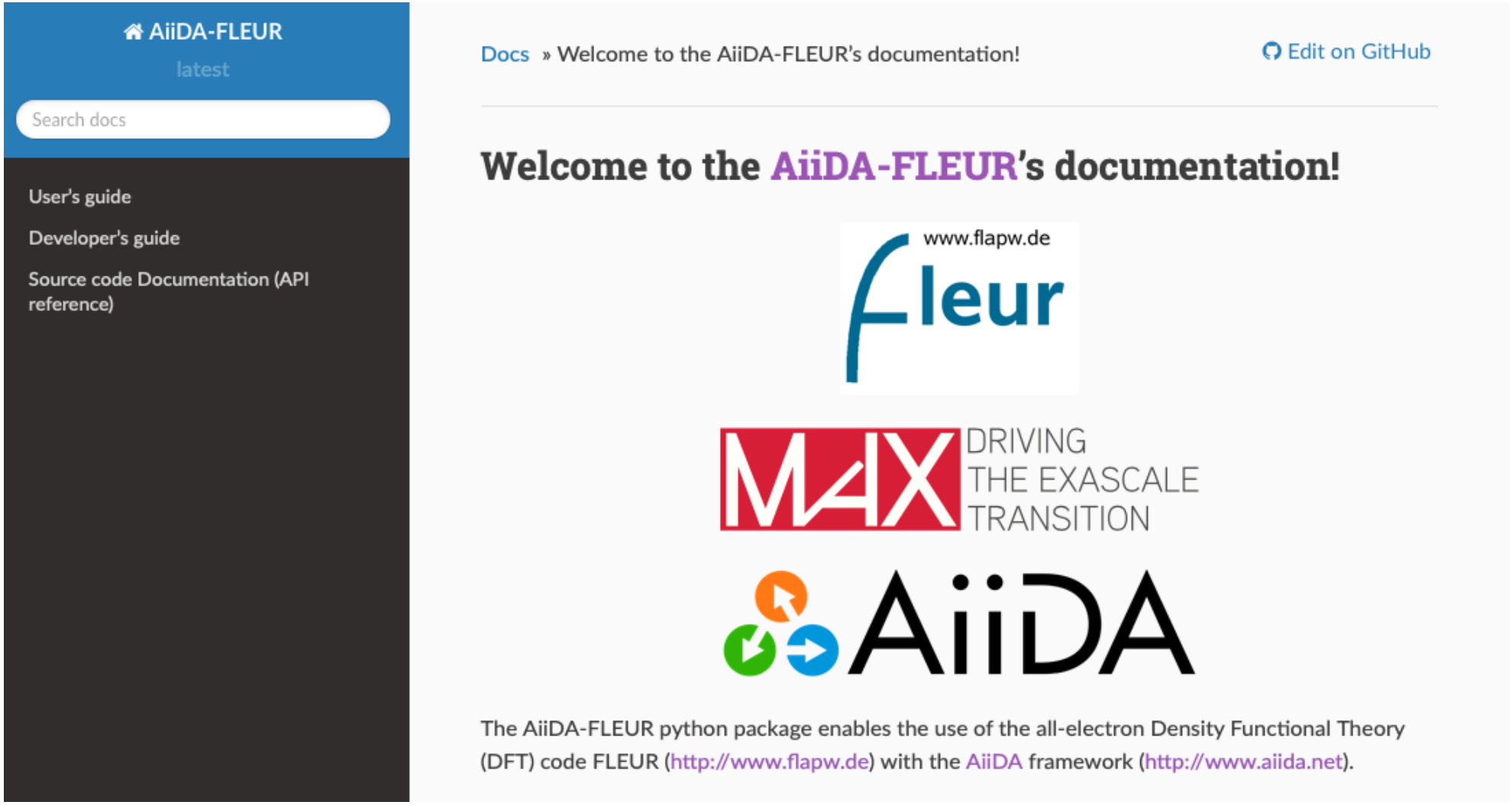
1 iteration with SOC

1 iteration with SOC

output

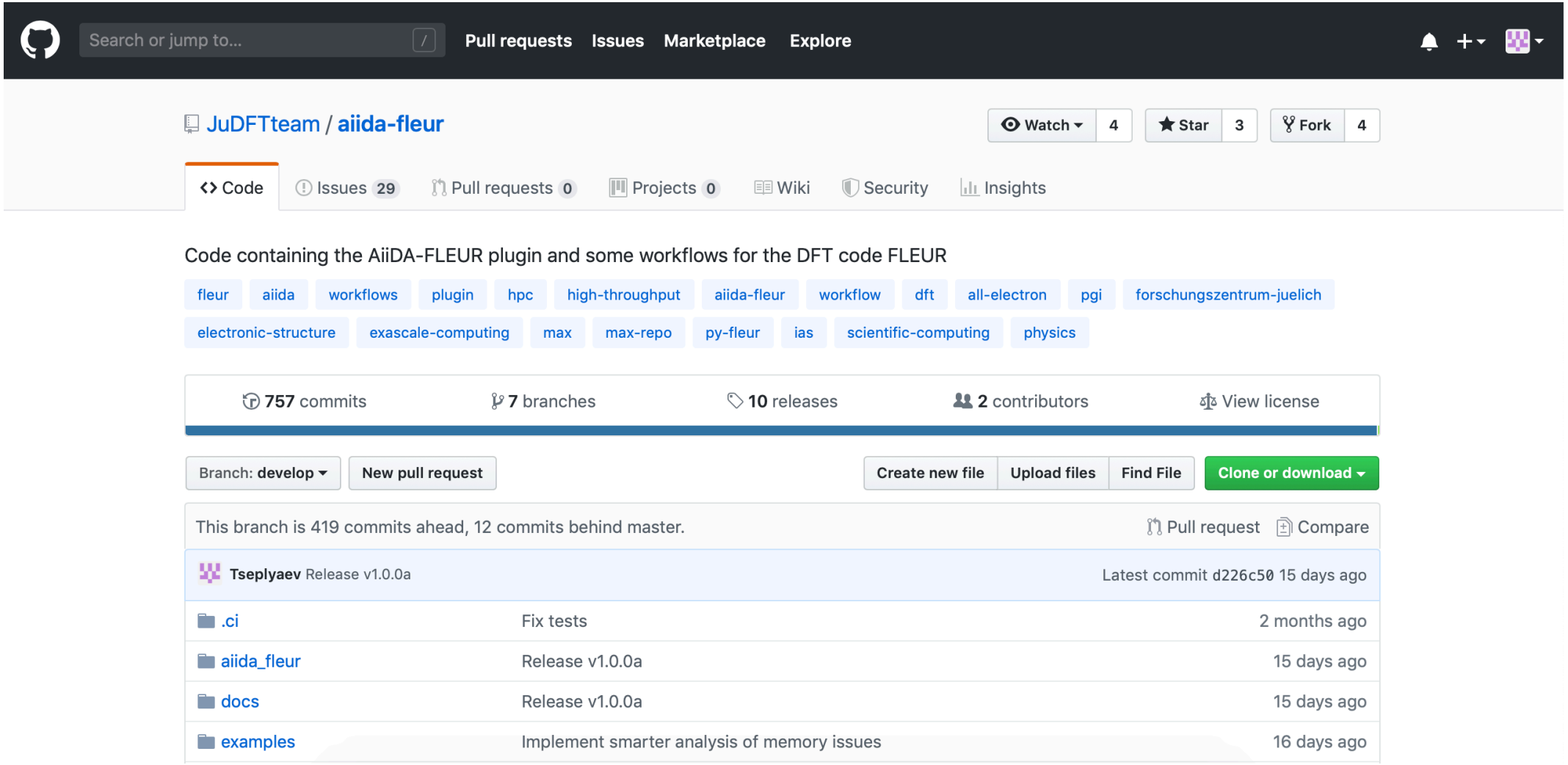
Documentation

aiida-fleur.readthedocs.io



GitHub

github.com/JuDFTteam/aiida-fleur



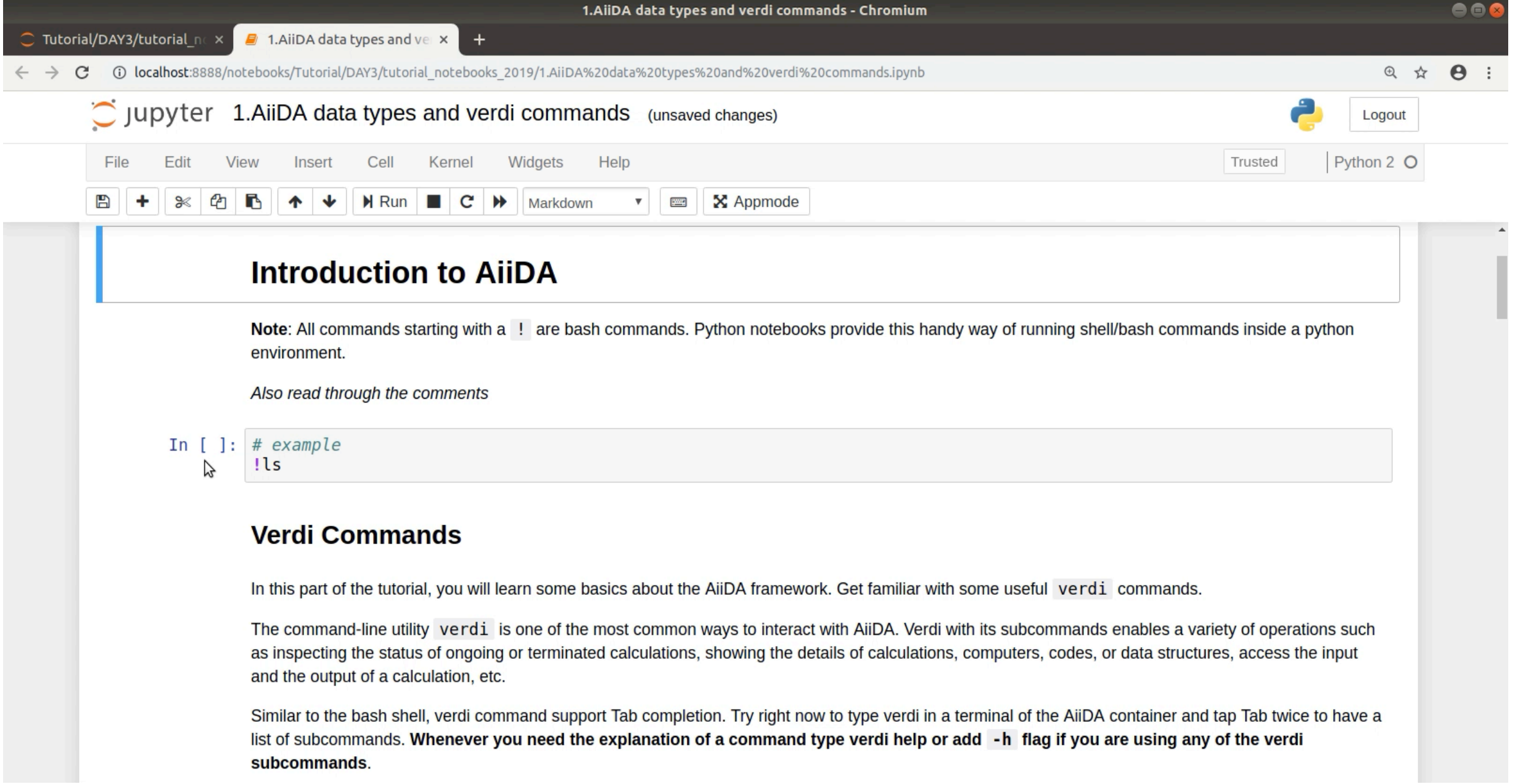
Tutorial

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


1.AiiDA data types and verdi commands - Chromium
Tutorial/DAY3/tutorial_n
1.AiiDA data types and ve
localhost:8888/notebooks/Tutorial/DAY3/tutorial_notebooks_2019/1.AiiDA%20data%20types%20and%20verdi%20commands.ipynb
jupyter 1.AiiDA data types and verdi commands (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 2
Run
Markdown
Appmode
Introduction to AiiDA
Note: All commands starting with a ! 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.

3) FleurinpData modifications - Chromium

Tutorial/DAY3/tutorial_n | 3) FleurinpData modificat | 2) AiiDA-Fleur input file | 1) AiiDA data types and ve

localhost:8888/notebooks/Tutorial/DAY3/tutorial_notebooks_2019/3)%20FleurinpData%20modifications.ipynb

jupyter 3) FleurinpData modifications Last Checkpoint: 19 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 2

Run Markdown Appmode

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` can not 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.

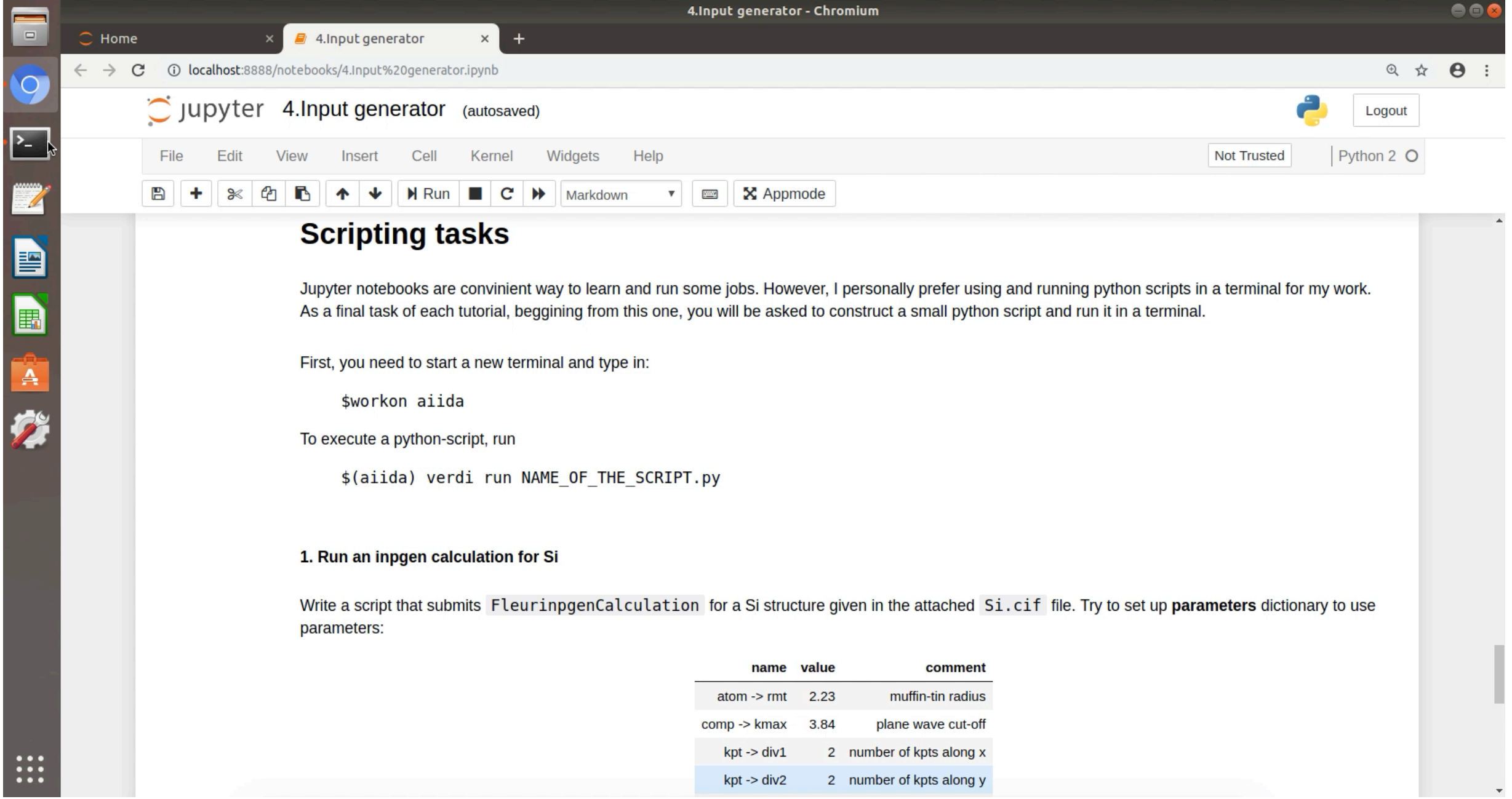
To make changes and store the result in the database, AiiDA-Fleur contains a `FluerinpModifier` class.

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 aiida.plugins import DataFactory
# FleurinpModifier = DataFactory('fleur.fleurinpmodifier')

from aiida_fleur.data.fleurinpmodifier import FleurinpModifier

fleurmode = FleurinpModifier(fleurinp)
```


4.Input generator - Chromium
Home
4.Input generator
localhost:8888/notebooks/4.Input%20generator.ipynb
jupyter 4.Input generator (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 2
Run
Markdown
Appmode
Scripting tasks
Jupyter notebooks are convinient 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, beggining 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:
$workon aiida
To execute a python-script, run
$(aiida) verdi run NAME_OF_THE_SCRIPT.py
1. Run an inpgen 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:
name value comment
atom -> rmt 2.23 muffin-tin radius
comp -> kmax 3.84 plane wave cut-off
kpt -> div1 2 number of kpts along x
kpt -> div2 2 number of kpts along y

Tutorial

Good luck with tutorials!

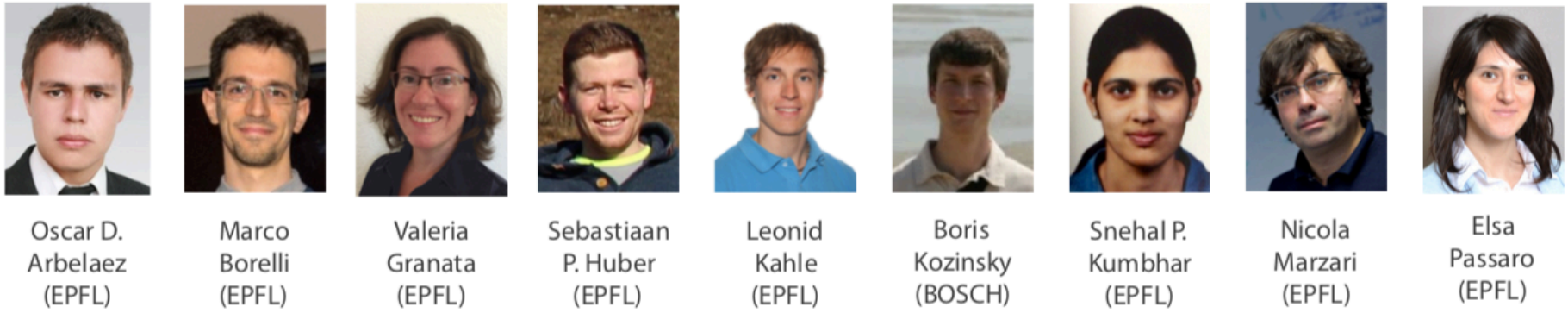
Any questions ➜ ask me or

Anoop

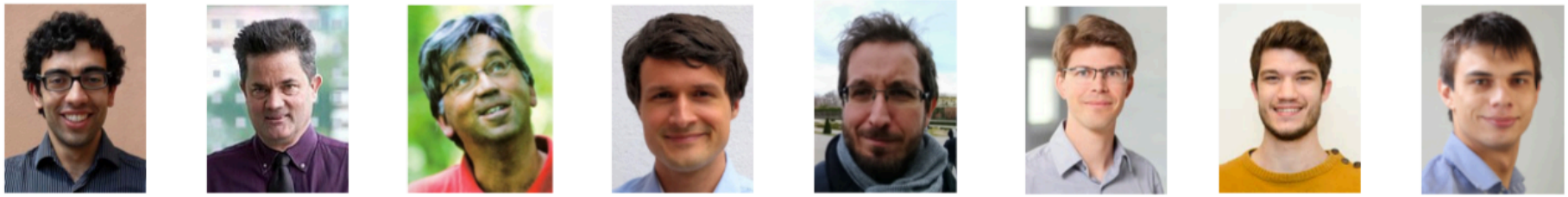
Acknowledgements

MAX DRIVING THE EXASCALE TRANSITION

AiiDA & Materials Cloud Teams



Oscar D. Arbelaez (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 FleruinpData
- A set of key-turn WorkChains is available