Siesta interface to AiiDA

Alberto García (ICMAB-CSIC, Barcelona)

Victor M. García-Suárez
Vladimir Dikan
Emanuele Bosoni

AiiDA workflows tutorial
Lausanne, 21-24 May 2019

AiiDA-Siesta package

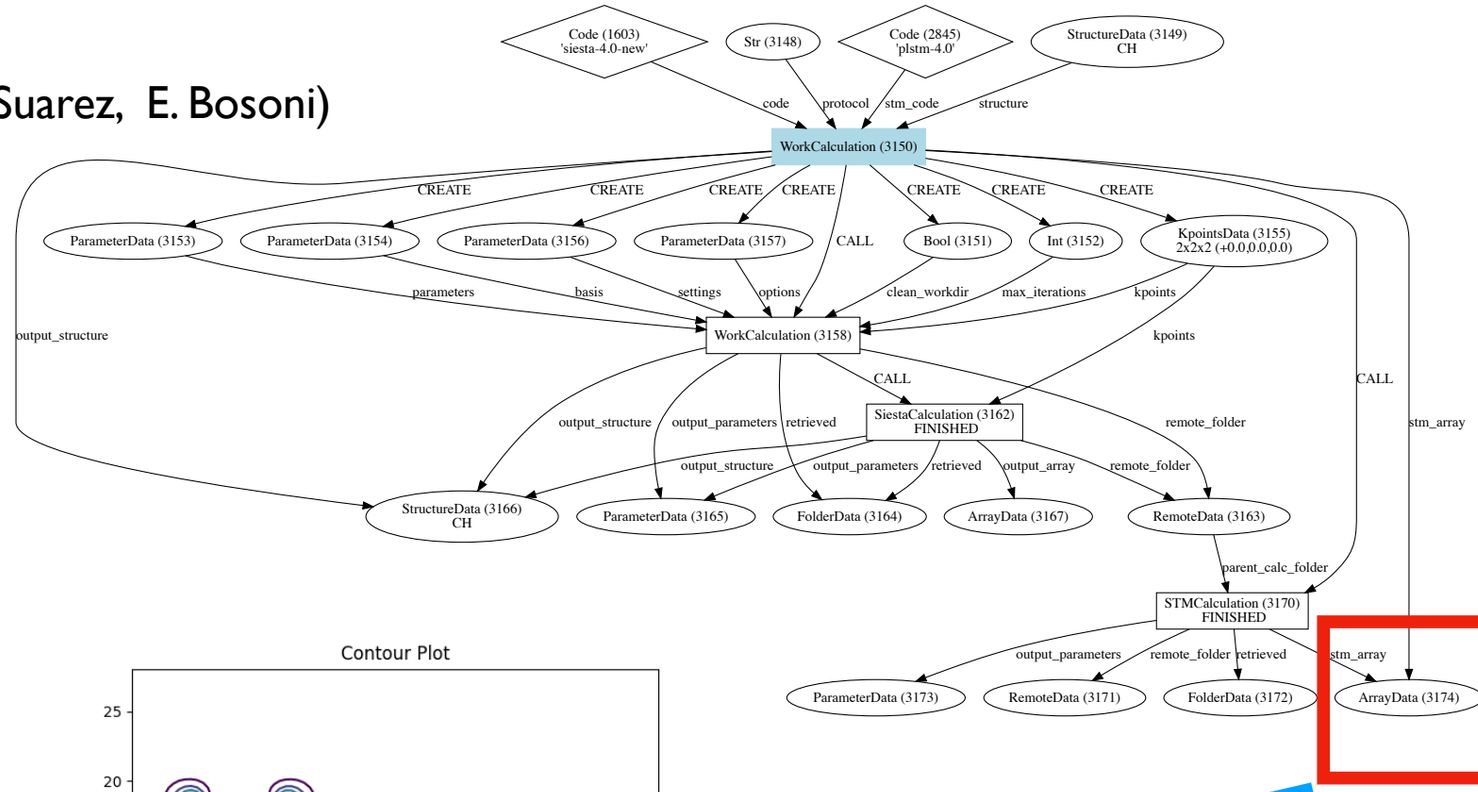
(A. Garcia, V. Dikan, V.M. Garcia-Suarez, E. Bosoni)

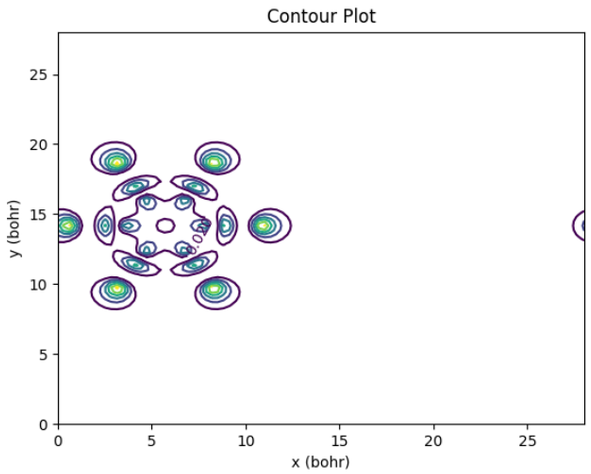
Plugin

- Total energies, forces, stresses, bands, spin polarization.
- Geometry relaxations (including variable cell), but no MD yet (no trajectories in the plugin).
- Output plugin uses existing CML I/O framework in XML.
- Basic mechanism for error reporting and restarts has been implemented.
- Incorporates new “Psfdata” class for pseudopotentials.
- All of Siesta FDF constructs can be passed to the plugin in the ‘parameters’ dictionary. So most of Siesta is in principle covered, except that the parsing is limited.

Workflows

- **BaseWorkChain**: Supports the basic features of SIESTA. It can recover from lack of convergence or out-of-time conditions.
- **BandsWorkChain**: Harnesseses the Base workflow to compute the band structure.
- **STMWorkChain**: Computes the LDOS as a proxy for STM images,





STM image of benzene molecule

```
"entry_points": {
    "aiida.calculations": [
        "siesta.siesta = aiida_siesta.calculations.siesta:SiestaCalculation",
        "siesta.stm = aiida_siesta.calculations.stm:STMCalculation",
        "siesta.vibra = aiida_siesta.calculations.vibra:VibraCalculation"
    ],
    "aiida.parsers": [
        "siesta.parser = aiida_siesta.parsers.siesta:SiestaParser",
        "siesta.stm = aiida_siesta.parsers.stm:STMParser",
        "siesta.vibra = aiida_siesta.parsers.vibra:VibraParser"
    ],
    "aiida.data": [
        "siesta.psf = aiida_siesta.data.psf:PsfData"
    ],
    "aiida.cmdline.data": [
        "psf = aiida_siesta.commands.data_psf:psfdata"
    ]
}
```

```
Inputs
                  code:  required  Code           Input code
            parameters:  required  Dict           Input parameters
               pseudos:  required  PsfData        Input pseudo potentials
             structure:  required  StructureData  Input structure
          bandskpoints:  optional  KpointsData    Input kpoints for bands
                 basis:  optional  Dict           Input basis
               kpoints:  optional  KpointsData    Input kpoints
              metadata:  optional
    parent_calc_folder:  optional  RemoteData     Parent folder
              settings:  optional  Dict           Input settings
Outputs
     output_parameters:  required  Dict           The calculation results
         remote_folder:  required  RemoteData     Input files necessary to run the process will
der node ...
             retrieved:  required  FolderData     Files that are retrieved by the daemon will be
 By defa ...
          bands_array:  optional  BandsData      Optional band structure
     bands_parameters:  optional  Dict           Optional parameters of bands
    forces_and_stress:  optional  ArrayData      Optional forces and stress
     output_structure:  optional  StructureData  Optional relaxed structure
Exit codes
                  10:  The process returned an invalid output
                  11:  The process did not register a required output
                 100:  The retrieved folder data node could not be accessed.
                 120:  Calculation did not reach scf convergence!
                 130:  Calculation did not reach geometry convergence!
```

```
alat = 15. # angstrom
cell = [[alat, 0., 0.,],
  [0., alat, 0.,],
  [0., 0., alat,],
 ]

 # Benzene molecule with a special carbon atom
 s = StructureData(cell=cell)
 s.append_atom(position=(0.000,0.000,0.468),symbols=['H'])
 s.append_atom(position=(0.000,0.000,1.620),symbols=['C'])
 s.append_atom(position=(0.000,-2.233,1.754),symbols=['H'])
 s.append_atom(position=(0.000,2.233,1.754),symbols=['H'])
 s.append_atom(position=(0.000,-1.225,2.327),symbols='C',name="Cred")
 s.append_atom(position=(0.000,1.225,2.327),symbols=['C'])
 s.append_atom(position=(0.000,-1.225,3.737),symbols=['C'])
 s.append_atom(position=(0.000,1.225,3.737),symbols=['C'])
 s.append_atom(position=(0.000,-2.233,4.311),symbols=['H'])
 s.append_atom(position=(0.000,2.233,4.311),symbols=['H'])
 s.append_atom(position=(0.000,0.000,4.442),symbols=['C'])
 s.append_atom(position=(0.000,0.000,5.604),symbols=['H'])
```

```
pseudo_file_to_species_map = [ ("C.psf", ['C', 'Cred']),
                               ("H.psf", 'H')
                               ]
```

```
basis_dict = {
'pao-basistype': 'split',
'pao-splitnorm': 0.150,
'pao-energyshift': '0.020 Ry',
'%block pao-basis-sizes' :"""
C    SZP
Cred SZ
H    SZP
%endblock pao-basis-sizes""",
}
```

Output


```
{
  "siesta:Version": "siesta-4.0.2",
  "E_fermi": -3.24,
  "E_fermi_units": "eV",
  "FreeE": -6656.2343
  "FreeE_units": "eV",
  "global_time": 55.213,
  "timing_decomposition": {
    "compute_DM": 33.208,
    "nlefsm-1": 0.582,
    "nlefsm-2": 0.045,
    "post-SCF": 2.556,
    "setup_H": 16.531,
    "setup_H0": 2.351,
    "siesta": 55.213,
    "state_init": 0.171
  },
  "warnings": [ "INFO: Job Completed"]
}
```

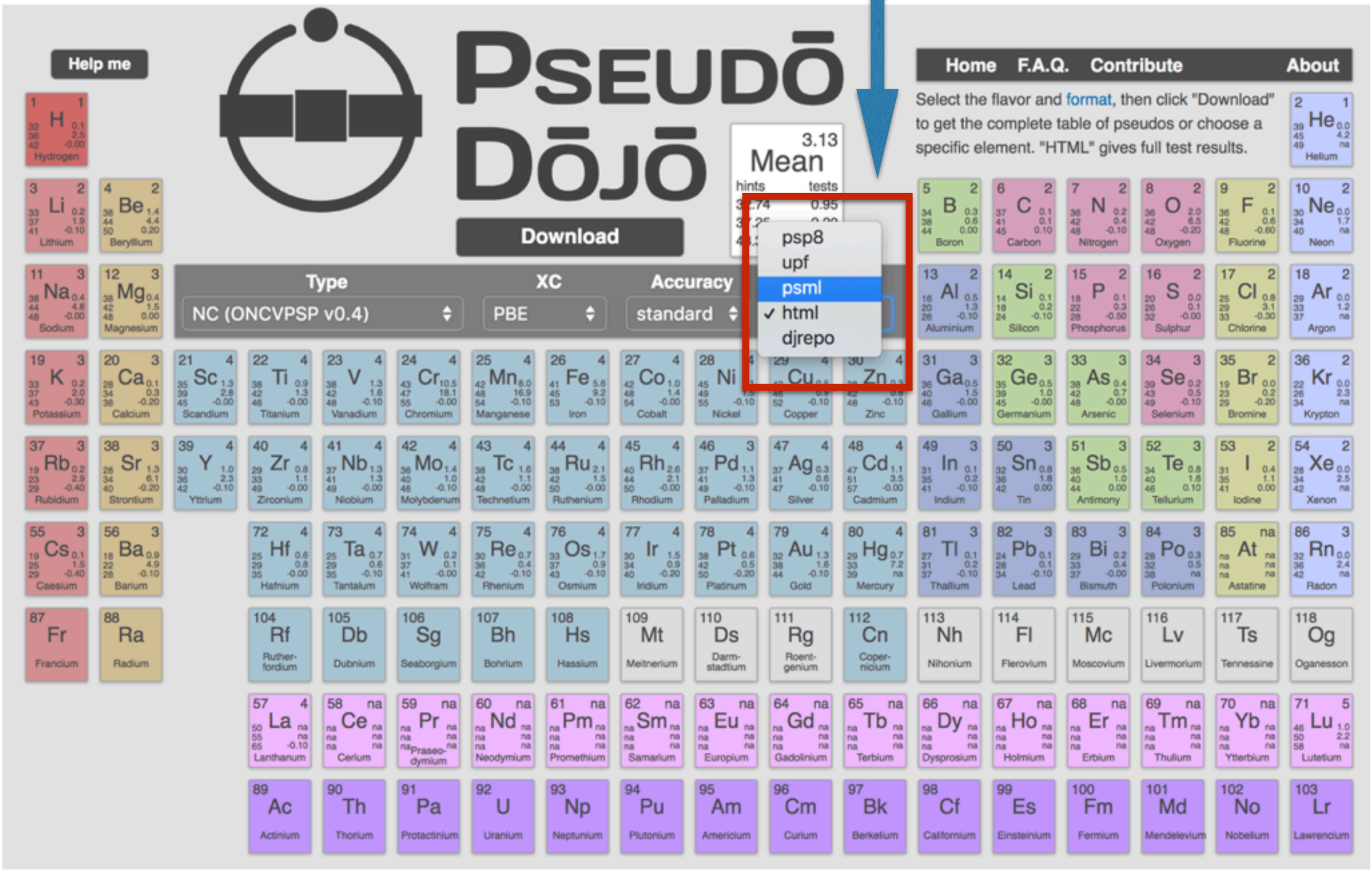
Pre-packaged quality protocols

Work-in-Progress
Example from QE:

```
    self.ctx.protocol = {
                'kpoints_mesh_offset': [0., 0., 0.],
                'kpoints_mesh_density': 0.2,
                'convergence_threshold': 2.E-06,
                'smearing': 'marzari-vanderbilt',
                'degauss': 0.02,
                'occupations': 'smearing',
                'tstress': True,
                'pseudo_data': {
                'H':  {'cutoff': 55,  'dual': 8,  'pseudo': '031US'},
                'He': {'cutoff': 55,  'dual': 4,  'pseudo': 'SG15'},
                'Li': {'cutoff': 30,  'dual': 8,  'pseudo': 'GBRV-1.4'},
                'Be': {'cutoff': 40,  'dual': 8,  'pseudo': 'GBRV-1.4'},
                'B':  {'cutoff': 40,  'dual': 8,  'pseudo': '031PAW'},
```

We need to provide a more comprehensive pseudopotential database

PSML format available



Other Issues

- **Basis sets** are further away from being treated as a “black-box”. We have some high-level knobs in Siesta, but some more heuristics are needed.
- **More features** need to be implemented and tested in the plugins and the workflows.
- **Migration** to 1.0 almost completed.
- A proper taxonomy of failure modes is needed

http://aiida-siesta-plugin.readthedocs.io

From Siesta to AiiDA and back

- While designing the plugin and workflows, **new ideas** for SIESTA operation appeared:
  - Proper **error reporting**
  - Production of more files, easier to **parse.**
  - Re-modularization; new basic objects

- New **insights**:
  - Codes are already implementing **crude workflows**
  - Separation of functionality is a key issue

Thank you