5th International ABINIT Developer Worshop

HAN SUR LESSE

Modularisation of ABINIT

Damien Caliste and the ABINIT community

L_Sim - CEA Grenoble (France)

April 11th 2011

Context and reasons of modularisation

Modularisation, pro & cons

- ✗ higher constraints when programming:
  - use minimal interfaces and isolate consistent parts;
  - clearly separate private / public;
- ✓ code tends to be clearer:
  - easier to maintain and to debug;
- ✓ code can be reused:
  - more use cases means better debugging;
  - give and receive contributions.

Steps of modularisation:

- Document purpose and I/O of routines (done);
- Advertise entries to the code with modules / interfaces (partially done);
- Allow isolated compilation and linkage (to be done).

Context and reasons of modularisation

Modularisation, pro & cons

- ✗ higher constraints when programming:
- ✓ code tends to be clearer:
- ✓ code can be reused:

Steps of modularisation:

- Document purpose and I/O of routines (done);
- Advertise entries to the code with modules / interfaces (partially done);
- Allow isolated compilation and linkage (to be done).

ABINIT is already taking advantages of modularisation

- FFT: use Goedecker implementation or FFTW;
- XC: internal or Marquès implementation;
- Wannier: use Wannier90;
- ...

Playing with symmetries (formerly `42_geometry`)

Symmetry detection routines

- Space group, primitive cell, ...
- Equivalent atoms, symmetry operators, ...

The public API

- Symmetries are an object, operating on a crystal with external constraints;
- Object is all private;
- Conditions are set from public methods;
- Properties are get through public methods;
- Lazy evaluation (*i.e.* on-the-fly update).

```
call ab6_symmetry_new(obj)  ! integer :: obj
call ab6_symmetry_set_lattice(obj, rprimd, ierr)
call ab6_symmetry_set_structure(obj, nat, iatype, xRed, ierr)
call ab6_symmetry_get_group(obj, spgrp, id, magn, afm, ierr)
```

Symmetry API Fortran/C, `ab6_symmetry` namespace

`new()`, `free()` to set or free symmetry objects.

Set conditions

All routines have a `ierr` argument (or a return value) for error reporting.

```
set_tolerance   (double tolsym)
set_lattice     (double rprimd[3][3])
set_structure   (integer nat,
                 integer types[nat],
                 double xred[3][nat])
set_spin        (double magn[3][nat])
set_spin_orbit  (boolean status)
set_field       (double field[3])
set_jellium     (boolean status)
set_periodicity(boolean per[3])
```

Symmetry API Fortran/C, `ab6_symmetry` namespace

Get properties

Get basic information:

```
get_n_atoms       (integer nat)
get_n_sym         (integer nsym)
get_multiplicity(integer mult)
```

Get symmetry information:

```
get_bravais (integer brav[3][3], ...)
get_matrices(integer nsym,
             integer syms[3][3][nsym], ...)
```

Get extra information:

```
get_group      (string name, integer id, ...)
get_equivalent(integer iat, integer equiv[])
get_type       (integer isym, string name)
```

Creating Monkhorst-Pack (and others) *k*-grids

Mainly a wrapper around `testkgrid()` and `getkgrid()`.

C/Fortran API

```
get_mp_k_grid  (Obj sym, integer nkpts,
                double kpts[3][nkpts],
                double wkpt[nkpts],
                integer ngkpt[3],
                integer nshifts,
                double shifts[3][nshifts])
get_auto_k_grid(Obj sym, integer nkpts,
                double kpts[3][nkpts],
                double wkpt[nkpts],
                double kptrlen)
```

Should be completed with reciprocal space handling functions.

Mixing library for iterations of diagonalisation

Full ABINIT support

- Memory / disk possibility;
- PAW and response function;
- Moving atoms inside, ...

Fortran only API, `ab6_mixing` namespace

```
new (integer den/pot, integer iscf,
     integer space, integer nspden,
     integer nfft, integer npawmix, ...)
eval(double arr[], integer istep, ...)
```

Question? A Fortran issue.

Would it be relevant to hide all internals, especially `f_fftgr`, `f_paw and f_atm` arrays? How to avoid memory copies and grant access to these buffers?

ABINIT code is spreading

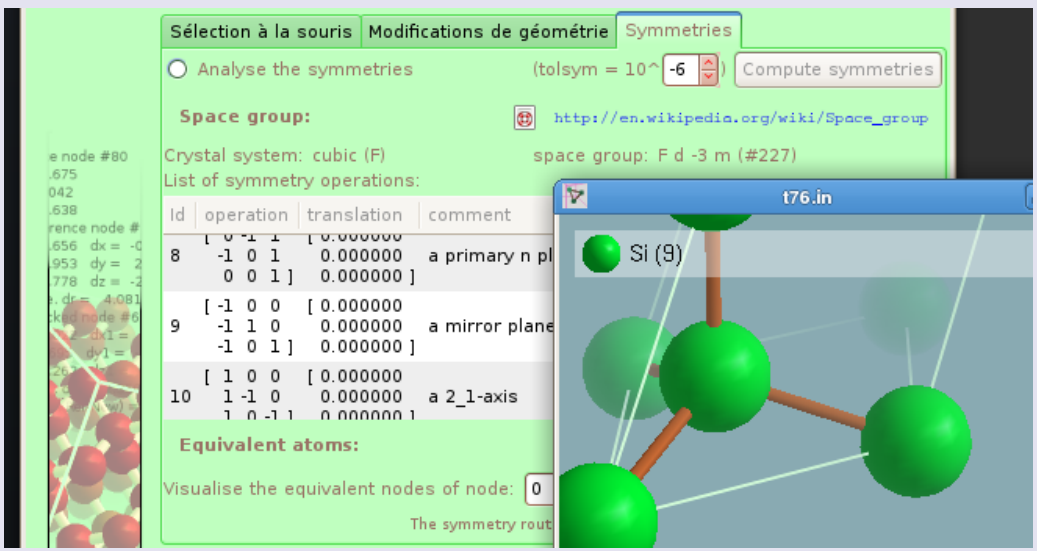
In other(s?) physic codes

BigDFT is using all modularisation, *e.g.* MD, mixing, symmetry detection and *k*-points mesh.

→ Future, send back specific BigDFT geometry optimisation capabilities into ABINIT.

Availability for {pre-,post-}processing

Read input ABINIT file and display sym. data (C bindings):



Make ABINIT code base a common lib for physics

Medium term objective

Advantage: keep control of code, get improvements from others, . . .

Build a library for basis-independent atomistic physic:

- Symmetry;
- Brillouin zone;
- Mixing;
- Density analysis;
- Poisson solver

✓ exchange-correlation;

✓ pseudo-potentials;

Or even go further with top-of-DFT algorithms:

- Geometry optimisation, MD;
- BSE, GW;
- PAW, . . .

Make ABINIT code base a common lib for physics

Medium term objective

Advantage: keep control of code, get improvements from others, . . .

Build a library for basis-independent atomistic physic:

- 42_geometry;
- 42_nlstrain;
- 43_ptgroups;
- 42_geomoptim;
- 49_gw_toolbox;
- 56_mixing;
- ✘ *53_spacepar*;
- ✘ *56_recipspace*;
- ✘ *62_occeig*;
- ✘ *62_poisson*;
- ✘ *63_bader*;

Based on low-level tools:

- ✘ *10_defs*;
- 12_hide_mpi;
- 14_hide_write;
- 15_gpu_toolbox;
- 18_profiling;
- 27_toolbox_oop;
- 28_numeric;
- 32_contract;
- 32_util;

```
mpif90 -o abinit abinit.F90 lib67_common.a ... -labtool
-lxc -lpsp ...
```

Propositions of modifications below `67_common`

- Libraries can not stop, change `leave_new()` calls to error reporting.
- `mpi_enreg` is ABINIT specific. Do a quick survey for MPI requirements (communicator only, FFT descriptors?).
- Advertise public routines through interfaces or modules. Do a list of exportable routines and modify `abilint` to export them.
- `defs_basis` has namespace issues. We should choose a global namespace, why not `ab_`? Rename public parameters (`defs_basis`).
- `use m_module` is not nice, prefix names with the namespace only.
- Move `50_abitypes` up to `67_abitypes`.
- Build the previous list of directories outside `src/`.

New possible abirules

- No `leave_new()` calls below `67_common`;
- Report errors:
  - with error ids described in `defs_basis`;
  - or with a simple error structure, allowing strings...
- Naming scheme for public routines (library prefix, object prefix, accessors names,...)
- don't use *magic* numbers but named parameters instead.