PWDFT.jl Documentation

Fadjar Fathurrahman

October 30, 2019

Contents

Part I
Implementation Notes

This document is a work in progress

In this part I will describe my design choices in implementing PWDFT.jl. This design is by no means perfect and it might change in the future to accomodate more complex use cases.

1 Overview

The design of PWDFT.jl is intended to be rather simple. One constraint that is set to the code is that it should be possible to perform application of Hamiltonian operator to wave function as simple as:

```
Hpsi = Ham*psi # or
Hpsi = op_H(Ham, psi)
```

where `psi` is, currently, of type `Array{ComplexF64,2}` [1]. This comes with an important consequences: all other pieces of information about how this operation is done should be present in the type of `Ham`. [2].
In PWDFT.jl, the type of `Ham` is `Hamiltonian`. Several important fields of `Hamiltonian` are instances of the following types (please refer to the source code for more details about this):

- `Atoms`: contains information about atomic structure: cell vectors, atomic species and atomic coordinates.
- `PsPot_GTH`: contains information about atomic pseudopotentials.
- `Electrons`: contains information about electronic states.
- `PWGrid`: contains information about plane wave basis set.
- `Potentials`: contains information about local potentials such as local pseudopotential, Hartree and exchange-correlation potential.
- `PsPotNL`: contains information about nonlocal pseudopotential terms.
- `Energies`: contains information about components of Kohn-Sham energy.
- `SymmetryInfo`: contains information about symmetry operations.

2 Atomic structure

The type `Atoms` contains the following information:

- Number of atoms: `Natoms::Int64`
- Number of atomic species: `Nspecies::Int64`
- Atomic coordinates: `positions::Array{Float64,2}`
- Unit cell vectors (lattice vectors): `LatVecs::Array{Float64,2}`

`Atoms` also contains several other fields such as `Zvals` which will be set according to the pseudopotentials assigned to the instance of `Atoms`. [3]
`LatVecs` is a 3×3 matrix. The vectors are stored column-wise which is opposite to PWSCF input convention. Convenience functions to calculate lattice vectors for several types of Bravais lattice are provided in PWDFT.jl. These functions adapt PWSCF definition. Several of these functions are listed below:

- `gen_lattice_sc` or `gen_lattice_cubic` for generating simple cubic lattice vectors.
- `gen_lattice_fcc`: for fcc structure
- `gen_lattice_bcc`: for bcc structure
- `gen_lattice_hcp`: for hcp structure

Please see file `gen_lattice.jl` for more information.
There are several ways to initialize an instance of `Atoms`. The following are typical cases.

- From xyz file. We need to supply the path to xyz file as string and set the lattice vectors:

[1] This function may be extended take other types other that plain Julia array for more complex case.
[2] We will also see some quirks related to this design choice later, such as applying Hamiltonian to several k-points or spin-polarized case
[3] Maybe we should include pseudopotential information under the `Atoms` type. However this would make `Atoms` "heavier".

```
atoms = Atoms(xyz_file="file.xyz", LatVecs=gen_lattice_sc(16.0))
```

- For crystalline systems, using keyword argument `xyz_string_frac` is sometimes convenient:

```
atoms = Atoms(xyz_string_frac=
        """
        2

        Si  0.0  0.0  0.0
        Si  0.25  0.25  0.25
        """, in_bohr=true,
        LatVecs=gen_lattice_fcc(10.2631))
```

  IMPORTANT We need to be careful to also specify `in_bohr` keyword to get the correct coordinates in bohr (which is used internally in `PWDFT.jl`).

- From extended xyz file, the lattice vectors information is included along with several others information, if any, however they are ignored):

```
atoms = Atoms(ext_xyz_file="file.xyz")
```

3 Pseudopotentials

Currently, PWDFT.jl supports a subset of GTH (Goedecker-Teter-Hutter) pseudopotentials. This type of pseudopotential is analytic and contains few parameters. PWDFT.jl distribution contains several parameters of GTH pseudopotential

4 Plane wave basis set

The type `PWGrid` wraps various variables related to plane wave basis set.

Part II
Howtos

This part contains miscellaneous info.

5 Referring or including files in **sandbox** (or other dirs in **PWDFT.jl**)

```
using PWDFT
const DIR_PWDFT = joinpath(dirname(pathof(PWDFT)),"..")
const DIR_PSP = joinpath(DIR_PWDFT,"pseudopotentials","pade_gth")
const DIR_STRUCTURES = joinpath(DIR_PWDFT, "structures")

pspfiles = [joinpath(DIR_PSP,"Ag-q11.gth")]
```

6 Generating lattice vectors

Lattice vectors are simply 3x3 array. We can set it manually or use one of functions defined in `gen_lattice_pwscf.jl` for generating lattice vectors for Bravais lattices that used in Quantum ESPRESSO's PWSCF.

7 Using Babel to generate xyz file from SMILES

```
babel file.smi file.sdf
babel file.sdf file.xyz
```

Use `babel -h` to autogenerate hydrogens.

8 Setting up pseudopotentials

One can use the function `get_default_psp(::Atoms)` to get default pseudopotentials set for a given instance of `Atoms`.

Currently, it is not part of main `PWDFT.jl` package. It is located under `sandbox` subdirectory of `PWDFT.jl` distribution.

```
using PWDFT

DIR_PWDFT = jointpath(dirname(pathof(PWDFT)),"..")
include(jointpath(DIRPWDFT,"sandbox","get_default_psp.jl"))

atoms = Atoms(ext_xyz_file="atoms.xyz")
pspfiles = get_default_psp(atoms)
```

Alternatively, one can set `pspfiles` manually because it is simply an array of `String`:

```
pspfiles = ["Al-q3.gth", "O-q6.gth"]
```

IMPORTANT Be careful to set the order of species to be same as `atoms.SpeciesSymbols`. For example, if

```
atoms.SpeciesSymbols = ["Al", "O", "H"]
```

then

```
pspfiles = ["Al-q3.gth", "O-q6.gth", "H-q1.gth"]
```

9 Initializing Hamiltonian

For molecular systems:

```
Ham = Hamiltonian( atoms, pspfiles, ecutwfc )
```

For insulator and semiconductor solids:

```
Ham = Hamiltonian( atoms, pspfiles, ecutwfc, meshk=[3,3,3] )
```

For metallic systems:

```
Ham = Hamiltonian( atoms, pspfiles, ecutwfc, meshk=[3,3,3],
 ↪  extra_states=4 )
```

Empty extra states can be specified by using `extra_states` keyword.

For spin-polarized systems, `Nspin` keyword can be used.

10 Iterative diagonalization of Hamiltonian

```
evals =  diag_LOBPCG!( Ham, psiks, verbose=false, verbose_last=false,
                       Nstates_conv=Nstates_occ )
```

11 Calculating electron density

Several ways:

```
Rhoe = calc_rhoe( Nelectrons, pw, Focc, psiks, Nspin )
# or
Rhoe = calc_rhoe( Ham, psiks )
# or
calc_rhoe!( Ham, psiks, Rhoe )
```

12 Read and write array (binary file)

Write to binary files:

```
for ikspin = 1:Nkpt*Nspin
    wfc_file = open("WFC_ikspin_"*string(ikspin)*".data","w")
    write( wfc_file, psiks[ikspin] )
    close( wfc_file )
end
```

Read from binary files:

```
psiks = BlochWavefunc(undef,Nkpt)
for ispin = 1:Nspin, ik = 1:Nkpt
    ikspin = ik + (ispin-1)*Nkpt
    # Don't forget to use read mode
    wfc_file = open("WFC_ikspin_"*string(ikspin)*".data","r")
    psiks[ikspin] = Array{ComplexF64}(undef,Ngw[ik],Nstates)
    psiks[ikspin] = read!( wfc_file, psiks[ikspin] )
    close( wfc_file )
end
```

Subspace rotation

In case need sorting:

```
Hr = psiks[ikspin]' * op_H( Ham, psiks[ikspin] )
evals, evecs = eigen(Hr)
evals = real(evals[:])

# Sort in ascending order based on evals
idx_sorted = sortperm(evals)

# Copy to Hamiltonian
Ham.electrons.ebands[:,ikspin] = evals[idx_sorted]

# and rotate
psiks[ikspin] = psiks[ikspin]*evecs[:,idx_sorted]
```

Usually we don't need to sort the eigenvalues if we use Hermitian matrix. We can calculate the subspace Hamiltonian by:

```
evals, evecs = eigen(Hermitian(Hr))
```

Status

29 July 2019 Total energy results are now similar to ABINIT and Quantum ESPRESSO. A rather comprehensive test has been added for SCF and Emin PCG for several simple systems.

28 May 2018 The following features are working now:

- LDA and GGA, spin-paired and spin polarized calculations
- Calculation with k-points (for periodic solids). SPGLIB is used to reduce the Monkhorst-Pack grid points for integration over Brillouin zone.

Band structure calculation is possible in principle as this can be done by simply solving Schrodinger equation with converged Kohn-Sham potentials, however there is currently no tidy script or function to do that.

Total energy result for isolated systems (atoms and molecules) agrees quite well with ABINIT and PWSCF results.

~~Total energy result for periodic solid is quite different from ABINIT and PWSCF. I suspect that this is related to treatment of electrostatic terms in periodic system.~~

These discrepancies have been minimized. For several systems the agreement is very good even though I did not use the same algorithm as ABINIT.

~~SCF is rather shaky for several systems, however it is working in quite well in nonmetallic system.~~

SCF stability has been improved with Pulay mixing and its variants.