PWDFT.jl: Density Functional Theory Calculations with Julia

Introduction and Current Status

Fadjar Fathurrahman

Engineering Physics Department
Research Center for Nanoscience and Nanotechnology
Institut Teknologi Bandung

24 September 2019

Software packages for DFT calculations

There are a lot of software packages for DFT calculations, for examples:

- Quantum ESPRESSO
- VASP
- ABINIT
- Gaussian series: G03, G09, G16
- NWchem

More extensive list:

`https://en.wikipedia.org/wiki/List_of_quantum_chemistry_and_solid-state_physics_software`

Problems

- These packages are very helpful for doing various calculations based on DFT. These packages are suitable for black-box-type calculations where we are only concerned about the results.
- However they are generally rather difficult to extend.
  - New development on the DFT functionals: users generally need to wait for the next release of the package to use them (if these functionals are to be implemented at all).
  - Custom calculations or post-processing steps: Users need to know in some detail about how the data they are interested in is represented or implemented in the package's source code.

Introducing PWDFT.jl

- A package (more or less like a library, no executable) for electronic structure calculations based on DFT.
- Available at `https://github.com/f-fathurrahman/PWDFT.jl`
- Using plane wave basis functions.
- Implemented using Julia programming language.
- My latest attempt to implement DFT softwares, previous attempts:
  - `ffr-LFDFT`: `https://github.com/f-fathurrahman/ffr-LFDFT`, using Lagrange basis functions, implemented in Fortran.
  - `ffr-PWDFT`: `https://github.com/f-fathurrahman/ffr-PWDFT` also using plane wave basis functions, implemented in Fortran.
- Starting point is my Julia implementation of the code by Prof. Tomas Arias described in his Practical DFT course. I extended the code to handle nonlocal pseudopotentials and multiple k-points.
- Taking many ideas from Quantum ESPRESSO, KSSOLV, ELK, etc.
- No logo yet.

Why I write another DFT package?

Writing a DFT package from scratch is not the solution for all problems.

[Rant mode ON]

Several questions appeared during my early days with DFT:

- Why my calculations are slow? What makes my calculations slow?
- Is this slowness justified? How can I make it faster?
- What are actually calculated by the DFT packages?

Learn by doing: how DFT or Kohn-Sham equations are solved (beyond described in textbooks or research papers)

Frustation when trying to extend functionalities of available packages.

Educational purpose: the secret art of writing a DFT code is not yet documented extensively in a book.

Programming languages for DFT

Programming languages used:

- Fortran and/or C/C++: ABINIT, VASP, Quantum Espresso, ...
- Python: GPAW
- MATLAB: KSSOLV, RESCU

Static languages: Fortran, C/C++

Dynamic languages: Python and MATLAB

Julia programming language

A rather new programming language (2012)

Syntax is familar to MATLAB or Python users

support for multidimensional array and linear algebra

Loop is fast!

Aims of PWDFT.jl

- Friendly-to-developers DFT package: enables quick implementation of various algorithms
- educational purpose: simple yet powerful enough to carry out practical DFT calculations for molecular and crystalline systems.

Julia installation

I am assuming familiarity with command line.

Download Julia the current stable of Julia at `https://julialang.org/downloads` for your operating system.

Unpack the tarball. Usually after unpacking the tarball you should see a new directory created: `julia-1.x.x`, where `1.x.x` is the version of Julia.

```
julia-1.x.x/
├── bin
├── etc
├── include
├── lib
├── libexec
├── LICENSE.md
└── share
```

The Julia binary resides within the `bin` directory. You can launch Julia interpreter by executing the `bin/julia` executable. After executing the binary you can see the the following output:

```
               _
   _       _ _(_)_     |  Documentation: https://docs.julialang.org
  (_)     | (_) (_)    |
   _ _   _| |_  __ _   |  Type "?" for help, "]?" for Pkg help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 1.3.1 (2019-12-30)
 _/ |\__'_|_|_|\__'_|  |  Official https://julialang.org/ release
|__/                   |

julia>
```

Title

Examples

Typical steps

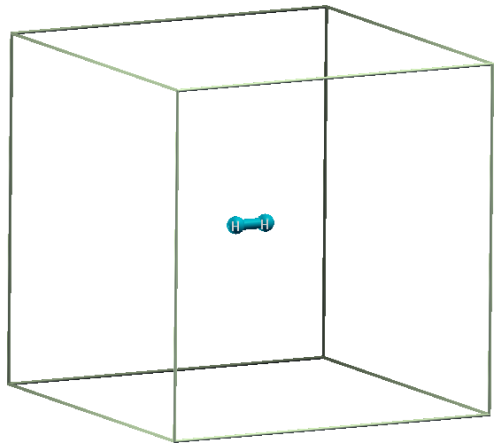
Write Julia code not an input file.

Typical steps:

- Initialize Atoms
- Initialize Hamiltonian with the given Atoms
- Solve the Hamiltonian

Atomic units are used (energy: Hartree, length: bohr)

Example: hydrogen molecule in a box



```
using PWDFT # activate PWDFT package

# Initialize an H2 molecule in cubic box (simple cubic lattice)
# The coordinates are read from
atoms = Atoms( xyz_file="H2.xyz",
               LatVecs=gen_lattice_sc(16.0) )

pspfiles = ["H-q1.gth"] # pseudopotential parameters

ecutwfc = 15.0 # cutoff energy for wave function expansion

# initialize Hamiltonian
Ham = Hamiltonian( atoms, pspfiles, ecutwfc )

# solve the Kohn-Sham problem (using SCF algorithm)
KS_solve_SCF!( Ham )
```

Output

```
$ julia run.jl

Self-consistent iteration begins ...
update_psi = LOBPCG
mix_method = simple
Density mixing with betamix =    0.20000

----------------------------------------------------------
       iter            E            ΔE           Δρ
----------------------------------------------------------
SCF:     1      -0.9088890376  9.08889e-01  1.36287e-04
SCF:     2      -0.9197780666  1.08890e-02  1.17877e-04
SCF:     3      -0.9589404606  3.91624e-02  9.53668e-05
SCF:     4      -0.9975511159  3.86107e-02  7.60953e-05
.... # snipped
```

Converged Kohn-Sham energy components

PWDFT.jl's result:

```
-------------------------
Final Kohn-Sham energies:
-------------------------

Kinetic    energy:       1.0100082069
Ps_loc     energy:      -2.7127851088
Ps_nloc    energy:       0.0000000000
Hartree    energy:       0.9015229089
XC         energy:      -0.6314259148
PspCore    energy:      -0.0000012675
------------------------------------
Electronic energy:      -1.4326811753
NN         energy:       0.3131700043
------------------------------------
Total      energy:      -1.1195111709
```

ABINIT's result:

```
Components of total free energy (in Hartree) :

   Kinetic energy  =  1.01004059294567E+00
   Hartree energy  =  9.01545039301481E-01
   XC energy       = -6.31436384237843E-01
   Ewald energy    =  3.13170052325859E-01
   PspCore energy  = -1.26742500464741E-06
   Loc. psp. energy= -2.71283243086241E+00
   NL   psp  energy=  0.00000000000000E+00
   >>>>>>>>> Etotal= -1.11951439795224E+00
```

SCF solvers

Kohn-Sham problem is solved using self-consistent field (SCF) iterations. This is the most popular method.

In PWDFT.jl we can set various options to SCF algorithm:

- `betamix`: linear mixing parameters (between 0 and 1)
- `mix_method`: linear, adaptive linear, Anderson, Pulay, restarted Pulay, periodic Pulay, and Broyden.
- `update_psi`: how to update the wave functions (iterative diagonalization or Chebyshev subspace filtering).

Direct energy minimization

For systems with band gaps:

Conjugate gradient

Direct minimization