BigDFT hands-on 2012

OAK RIDGE, TENNESSEE, USA

All you need to know about BigDFT

Damien Caliste, Luigi Genovese & Thierry Deutsch

L_Sim - CEA Grenoble

8 February 2012
Outline

1 Introduction to the wavelet calculations
   - Definition of the basis-set
   - Structure of the calculation
     - The electronic minimisation loop
     - The input guess
     - Possible add-on calculations after fix point is reached

2 A quick look from the user point of view
   - The input parameters and atomic data
   - How to compile BigDFT
   - How to run BigDFT
A DFT code based on Daubechies wavelets

BigDFT: a PSP Kohn-Sham code

A Daubechies wavelets basis has unique properties for DFT usage

- Systematic, Orthogonal
- Localised, Adaptive
- Kohn-Sham operators are analytic

Short, Separable convolutions

\[ \tilde{c}_\ell = \sum_j a_j c_{\ell-j} \]

- Peculiar numerical properties

Real space based, highly flexible

Big & inhomogeneous systems

Daubechies Wavelets
Wavelet properties: multi-resolution

Example of two resolution levels: step function (Haar Wavelet)

**Scaling Functions**: Multi-Resolution basis

Low and High resolution functions related each other

\[
\begin{array}{c}
\text{Low resolution function} \\
= \text{Scaling Function} + \text{Wavelet}
\end{array}
\]

Scaling Function + Wavelet = High resolution

We increase the resolution without modifying the grid spacing
Going to a 3D basis-set

On a discretised mesh \([i, j, k]\) for one resolution:

\[ f(i, j, k) = \sum_{l,m,n} c_{l,m,n} \Phi_{N_x}^l(i) \Phi_{N_y}^m(j) \Phi_{N_z}^n(k) \]

Schematic view for a multi-resolution grid in 2D
Wavelet properties: adaptivity

Adaptivity
Resolution can be refined following the grid point.

The grid is divided in Low (1 DoF) and High (8 DoF) resolution points. Points of different resolution belong to the same grid. Empty regions must not be “filled” with basis functions.

Localization property, real space description
Optimal for big & inhomogeneous systems, highly flexible
Overview of wavelet families used in BigDFT code

**Daubechies** $f(x) = \sum_\ell c_\ell \phi_\ell(x)$

Orthogonal set

$$c_\ell = \int dx \phi_\ell(x) f(x)$$

No need to calculate overlap matrix of basis functions
Used for wavefunctions, scalar products

**Interpolating** $f(x) = \sum_j f_j \varphi_j(x)$

Dual to dirac deltas

$$f_j = f(j)$$

The expansion coefficients are the point values on a grid
Used for charge density, function products

**Magic Filter method (A. Neelov, S. Goedecker)**

The passage between the two basis sets can be performed without losing accuracy
Basis set features

BigDFT features in a nutshell

- Arbitrary absolute precision can be achieved
  Good convergence ratio for real-space approach ($O(h^{14})$)

- Optimal usage of the degrees of freedom (adaptivity)
  Optimal speed for a systematic approach (less memory)

- Hartree potential accurate for various boundary conditions
  Free and Surfaces BC Poisson Solver
  (present also in CP2K, ABINIT, OCTOPUS)

- Data repartition is suitable for optimal scalability
  Simple communications paradigm, multi-level parallelisation possible (and implemented)

Improve and develop know-how
Optimal for advanced DFT functionalities in HPC framework

Laboratoire de Simulation Atomistique  http://inac.cea.fr/L_Sim  Damien Caliste, Luigi Genovese & Thierry Deutsch
The DFT framework

The ground state is obtained by solving:

\[ H\psi = E\psi \]

- Expressed only for the electrons, using the Born-Oppenheimer approximation;
- Solved in the framework of Kohn-Sham, using one particule wavefunctions;
- Use of a self-consistency loop since the Hamiltonian \( H \) depends on \( \psi \).

What about wavelets?

Wavelets are the basis set for the representation of the wavefunctions. Thus things like XC approximation or Hellmann-Feynman theorem are still usable.
Elements of the Hamiltonian $H$

All classical parts present in plane-waves, can also be treated with wavelets.

**The kinetic operator $\nabla^2 \psi$**

Computed from the wavefunctions $\psi$, applying a local filter. Length for Daubechies is 28, see isf_to_daub_kinetic().

Finite differences in 1D:

$$\Delta f(x_n) = \frac{f(x_{n-1}) - 2f(x_n) + f(x_{n+1})}{h^2}$$

is a filter of $(1, -2, 1)$.

Precision is $O(h^{14})$ for arbitrary function. Even exact for linear combination of Daubechies functions:

$$f(x) = \sum_{\ell} c_\ell \phi_\ell(x), \quad \nabla^2 f(x) = \sum_{\ell} \tilde{c}_\ell \phi_\ell(x),$$

$$\tilde{c}_\ell = \sum_j c_j a_{\ell-j}, \quad a_\ell \equiv \int \phi_0(x) \partial_x^2 \phi_\ell(x),$$
Elements of the Hamiltonian $H$

**The non-local part of pseudo-potentials**

Computed from the **scalar product** of wavefunctions $\Psi$ and the projectors, in routine `applyprojectorsonthefly()`.

$$V_{\text{nonloc}} = \sum_{\ell} \sum_{ij} \sum_{m=-\ell}^{\ell} h_{ij}^{(\ell)} |p_i^\ell,m\rangle \langle p_j^\ell,m|$$

$$\langle r|p_i^\ell,m\rangle = Y_{\ell,m}(\theta, \varphi) f_i^{(\ell)}(r_\ell) e^{-\frac{1}{2}\left(\frac{r}{r_\ell}\right)}$$

This is possible for GTH and HGH pseudo-potentials because of the spatial **separability** in their analytic expression.

Ongoing developments are done to include PAW pseudo-potentials.
Elements of the Hamiltonian $H$

The local potentials, $V_H$, $V_{\text{xc}}$, $V_{\text{psploc}}$ & $V_{\text{ionic}}$

Hartree and ionic potentials are computed with a Poisson solver from the density $\rho(j)$.

\[ V_H(j) = \int d\vec{x} \frac{\rho(\vec{x})}{|\vec{x} - j|}, \quad V_{\text{xc}}(j) \text{ with LibXC routines and} \]
\[ V_{\text{ionic}}(j) = \sum_\kappa \int d\vec{x} \frac{Z_\kappa}{|\vec{x} - j|} \]

This Poisson solver is:

- very fast and accurate, with optimal parallelisation ;
- can be used independently from the DFT code ;
- integrated quantities (energies) are easy to extract ;
- correct treatment of isolated boundary conditions and of charge offset ;
- non-adaptive, needs data uncompression.

Laboratoire de Simulation Atomistique  
http://inac.cea.fr/L_Sim  
Damien Caliste, Luigi Genovese & Thierry Deutsch
Finding the fix point in Schrödinger equation

Two possible methods: the diagonalisation and the **direct minimisation**. The latter is **fast** and **simple** but restricted to systems with a non-zero HOMO-LUMO gap.

**Direct minimisation (systems with gap)**

Classical gradient algorithms are applied for electronic minimisation: steepest descent or DIIS.

**Diagonalisation (metallic systems)**

Done within two loops, the inner one, non self consistent to find wavefunctions of a given hamiltonian, the outer one mixing either potential or density.

To improve convergency, **preconditioning** is done on the gradients $H\Psi$ using the routine `preconditionall()`. $(\frac{1}{2}\nabla^2 + C)\vec{p} = \vec{g}$, $\vec{p}$ is computed in an iterative way (CG).
The input guess, take a good start!

On the contrary to plane waves, wavelet representation requires a good input start for the minimisation loop.

Using atomic orbitals

The input guess is based on atomic orbitals, using the following scheme:

- project gaussian atomic orbitals in a Daubechies wavelet basis set (see the `inputguess_gaussian_orbitals()` routine);
- compute the associated hamiltonian;
- diagonalise it in this basis set;
- use the lowest states as input for the calculation.

This is done in `input_wf_diag()` routine.
Additional computation after minimisation

Getting the Kohn-Sham wavefunctions

A diagonalisation is done on output of the minimisation loop by the `last_orthon()` routine.

Compute the forces

The forces are sum of three parts:

- the local contribution, computed with the Poisson solver, see `local_forces()` routine;
- the non-local part, coming from the first derivatives of projectors, see `nonlocal_forces()` routine;
- an optional contribution to the non-linear core correction, see `rhocore_forces()` routine.

Finite-size effects - an error estimation

After convergency, the basis set can easily be expanded to estimate these effects.
Introduction to BigDFT

Overview (code)

Self-consistent loop

- Input guess in LCAO
  - project atomic orb. in Daubechies
  - diagonalise + take lowest eigen-values.
- Restart from disk or memory
- Start from random wavefunctions
- Restart from gaussians on disk or memory

Calculate density.

Kinetic operator

Preconditioner

- force calculation;
- empty bands calculation;
- band structure;
- mulliken analysis...

Calculate energy and gradient() & hpsitopsi()
Outline

1 Introduction to the wavelet calculations
   - Definition of the basis-set
   - Structure of the calculation
     - The electronic minimisation loop
     - The input guess
     - Possible add-on calculations after fix point is reached

2 A quick look from the user point of view
   - The input parameters and atomic data
   - How to compile BigDFT
   - How to run BigDFT
The BigDFT input files (or not)

- **input.dft**: DFT related parameters;
- **input.geopt**: geometrie relaxations and MD;
- **input.kpt**: the \( k \)-point mesh and band structure path;
- **input.mix**: diagonalisation mechanism;
- **input.sic, tddft, occ, perf**: other parameters;
- **psppar.Element**: pseudo-potentials for each element;
- **posinp.xyz**: the atomic coordinates and box definition.

Only the coordinates are mandatory.

All lines of different input files are mandatory and output on BigDFT log. When BigDFT is run, all default values are output in files called **default.xxx**.

**Naming scheme**

All input files may be renamed by providing a `name` argument to the command line.
BigDFT uses \textit{XYZ} format plus additional modifications:

- **physical unit**, after the number of atoms on the first line put either bohr or angstrom.
- **boundary conditions**, on second line, start with one of freeBC, surface $X_{\text{lat}}$ 0 $Z_{\text{lat}}$ or periodic $X_{\text{lat}}$ $Y_{\text{lat}}$ $Z_{\text{lat}}$.
- **frozen positions**, add a f, fy or fxz at the end of an atom line.
- **input polarisation**, add values at the end of an atom line.

On output, forces are added for each element at the end of the XYZ file:

\begin{verbatim}
2 angstrom
freeBC
Na -1 0 0
Cl +1 0 0
forces
Na 2.654367E-2 0 0
Cl -2.654367E-2 0 0
\end{verbatim}
Defining the basis set in `input.dft`

**Main specific variables to control the basis set**

- `$hgrid$`: the grid size
- `$frmult$`: the expansion around atoms for fine grid
- `$crmult$`: the expansion around atoms for coarse grid

**Convergence properties**

- **Decreasing** `$hgrid$` → more degrees of freedom.
- **Increasing** `$crmult$` → less box constraint.

It is important to improve both parameters together to avoid error on one hiding improvement made on the other one.
How to compile BigDFT

Build system of BigDFT is based on *Autotools*. BigDFT provides uncommon libraries (LibXC, LibABINIT) in its tar file and relies on *installed Lapack implementations*.

**Compiling with Intel MKL**

```bash
./configure -with-ext-linalg="-lmkl_intel_lapack
-lmkl_intel_thread -lmkl_core"
-with-ext-linalg-path="-L$MKLPATH" FC=mpif90
```

**Compiling with OpenCL (or CUDA) support**

```bash
./configure -enable-opencl
-with-ocl-path="/opt/cuda/4.1" FC=mpif90
```

CUDA is activated with equivalent options. Both can be activated at the same time.
How to run BigDFT

Some pre-processing

\texttt{bigdft-tool -n nprocs [name]}

Help to find the optimal number of processors for the run that fit into the computer memory. Also useful to check that all input files are without errors.

The main run

\texttt{mpirun -np 6 bigdft [name] | tee [name].log}

Number of MPI processes 5
MPI process does not use OpenMP
No material acceleration (iproc=0)

The outputs

- \texttt{log} is output on stdout, so redirect it.
- generated \texttt{data} are stored in a subdirectory called \texttt{data[-name]}.