Advances in Libxc

Micael Oliveira

5th International ABINIT Developer Workshop

April 11-14, 2011, Han-sur-Lesse

Outline

Kohn-Sham equations

The main equations of DFT are the Kohn-Sham equations:

$$\left[-\frac{1}{2}\nabla^2 + v_{\mathrm{ext}}(r) + v_{\mathrm{H}}(r) + v_{\mathrm{xc}}(r)\right] \varphi_i(r) = \epsilon_i \varphi_i(r)$$

where the exchange-correlation potential is defined as

$$v_{\mathrm{xc}}(r) = \frac{\delta E_{\mathrm{xc}}}{\delta n(r)}$$

In any practical application of the theory, we have to use an approximation to E_{xc}, or $v_{\mathrm{xc}}(r)$.

Kohn-Sham equations

The main equations of DFT are the Kohn-Sham equations:

$$\left[-\frac{1}{2}\nabla^2 + v_{\rm ext}(r) + v_{\rm H}(r) + v_{\rm xc}(r)\right]\varphi_i(r) = \epsilon_i\varphi_i(r)$$

where the exchange-correlation potential is defined as

$$v_{\rm xc}(r) = \frac{\delta E_{\rm xc}}{\delta n(r)}$$

In any practical application of the theory, we have to use an approximation to $E_{\rm xc}$, or $v_{\rm xc}(r)$.

Jacob's ladder

Local density approximation:

$$E_{\mathrm{xc}}^{\mathrm{LDA}}(r) = E_{\mathrm{xc}}^{\mathrm{LDA}}[n]\Big|_{n=n(r)}$$

Generalized gradient approximation:

$$E_{\mathrm{xc}}^{\mathrm{GGA}}(r) = E_{\mathrm{xc}}^{\mathrm{GGA}}[n, \nabla n]\Big|_{n=n(r)}$$

Meta-generalized gradient approximation:

$$E_{\mathrm{xc}}^{\mathrm{MGGA}}(r) = E_{\mathrm{xc}}^{\mathrm{MGGA}}[n, \nabla n, \nabla^2 n, \tau]\Big|_{n=n(r), \tau=\tau(r)}$$

And more: orbital functionals, hybrid functionals, hyper-GGAs, etc.

Jacob's ladder

Local density approximation:

$$E_{\mathrm{xc}}^{\mathrm{LDA}}(r) = E_{\mathrm{xc}}^{\mathrm{LDA}}[n]\Big|_{n=n(r)}$$

Generalized gradient approximation:

$$E_{\mathrm{xc}}^{\mathrm{GGA}}(r) = E_{\mathrm{xc}}^{\mathrm{GGA}}[n, \nabla n]\Big|_{n=n(r)}$$

Meta-generalized gradient approximation:

$$E_{\mathrm{xc}}^{\mathrm{MGGA}}(r) = E_{\mathrm{xc}}^{\mathrm{MGGA}}[n, \nabla n, \nabla^2 n, \tau]\Big|_{n=n(r),\tau=\tau(r)}$$

And more: orbital functionals, hybrid functionals, hyper-GGAs, etc.

Jacob's ladder

Local density approximation:

$$E_{\mathrm{xc}}^{\mathrm{LDA}}(r) = E_{\mathrm{xc}}^{\mathrm{LDA}}[n]\Big|_{n=n(r)}$$

Generalized gradient approximation:

$$E_{\mathrm{xc}}^{\mathrm{GGA}}(r) = E_{\mathrm{xc}}^{\mathrm{GGA}}[n, \nabla n]\Big|_{n=n(r)}$$

Meta-generalized gradient approximation:

$$E_{\mathrm{xc}}^{\mathrm{MGGA}}(r) = E_{\mathrm{xc}}^{\mathrm{MGGA}}[n, \nabla n, \nabla^2 n, \tau]\Big|_{n=n(r),\tau=\tau(r)}$$

And more: orbital functionals, hybrid functionals, hyper-GGAs, etc.

Jacob's ladder

Local density approximation:

$$E_{\mathrm{xc}}^{\mathrm{LDA}}(r) = E_{\mathrm{xc}}^{\mathrm{LDA}}[n]\big|_{n=n(r)}$$

Generalized gradient approximation:

$$E_{\mathrm{xc}}^{\mathrm{GGA}}(r) = E_{\mathrm{xc}}^{\mathrm{GGA}}[n, \nabla n]\big|_{n=n(r)}$$

Meta-generalized gradient approximation:

$$E_{\mathrm{xc}}^{\mathrm{MGGA}}(r) = E_{\mathrm{xc}}^{\mathrm{MGGA}}[n, \nabla n, \nabla^2 n, \tau]\big|_{n=n(r),\tau=\tau(r)}$$

And more: orbital functionals, hybrid functionals, hyper-GGAs, etc.

What do we need to compute $v_{\rm xc}$?

The energy is usually written as:

$$E_{\rm xc} = \int {\rm d}r \, e_{\rm xc}(r) = \int {\rm d}r \, n(r) \epsilon_{\rm xc}(r)$$

The potential in the LDA is:

$$v_{\rm xc}^{\rm LDA}(r) = \frac{d}{dn} e_{\rm xc}^{\rm LDA}(n) \Big|_{n=n(r)}$$

In the GGA:

$$v_{\rm xc}^{\rm GGA}(r) = \frac{\partial}{\partial n} e_{\rm xc}^{\rm LDA}(n, \nabla n) \Big|_{n=n(r)} - \nabla \, \frac{\partial}{\partial(\nabla n)} e_{\rm xc}^{\rm LDA}(n, \nabla n) \Big|_{n=n(r)}$$

Why a library of xc functionals?

- The xc functional is at the heart of DFT.
- There are many approximations for the xc (probably of the order of **150–200**).
- Most computer codes only include a very limited quantity of functionals, typically around **10–15**.
- Implementation of functionals is a time consuming task.
- Chemist and Physicists do not use the same functionals.

- Difficult to reproduce older calculations with older functionals.
- Difficult to reproduce calculations performed with other codes.
- Difficult to perform calculations with the newest functionals.

Why a library of xc functionals?

- The xc functional is at the heart of DFT.
- There are many approximations for the xc (probably of the order of **150–200**).
- Most computer codes only include a very limited quantity of functionals, typically around **10–15**.
- Implementation of functionals is a time consuming task.
- Chemist and Physicists do not use the same functionals.

- Difficult to reproduce older calculations with older functionals.
- Difficult to reproduce calculations performed with other codes.
- Difficult to perform calculations with the newest functionals.

Why a library of xc functionals?

- The xc functional is at the heart of DFT.
- There are many approximations for the xc (probably of the order of **150–200**).
- Most computer codes only include a very limited quantity of functionals, typically around **10–15**.
- Implementation of functionals is a time consuming task.
- Chemist and Physicists do not use the same functionals.

- Difficult to reproduce older calculations with older functionals.
- Difficult to reproduce calculations performed with other codes.
- Difficult to perform calculations with the newest functionals.

Why a library of xc functionals?

- The xc functional is at the heart of DFT.
- There are many approximations for the xc (probably of the order of **150–200**).
- Most computer codes only include a very limited quantity of functionals, typically around **10–15**.
- Implementation of functionals is a time consuming task.
- Chemist and Physicists do not use the same functionals.
- Difficult to reproduce older calculations with older functionals.
- Difficult to reproduce calculations performed with other codes.
- Difficult to perform calculations with the newest functionals.

Why a library of xc functionals?

- The xc functional is at the heart of DFT.
- There are many approximations for the xc (probably of the order of **150–200**).
- Most computer codes only include a very limited quantity of functionals, typically around **10–15**.
- Implementation of functionals is a time consuming task.
- Chemist and Physicists do not use the same functionals.

- Difficult to reproduce older calculations with older functionals.
- Difficult to reproduce calculations performed with other codes.
- Difficult to perform calculations with the newest functionals.

Why a library of xc functionals?

- The xc functional is at the heart of DFT.
- There are many approximations for the xc (probably of the order of **150–200**).
- Most computer codes only include a very limited quantity of functionals, typically around **10–15**.
- Implementation of functionals is a time consuming task.
- Chemist and Physicists do not use the same functionals.
- Difficult to reproduce older calculations with older functionals.
- Difficult to reproduce calculations performed with other codes.
- Difficult to perform calculations with the newest functionals.

Why a library of xc functionals?

- The xc functional is at the heart of DFT.
- There are many approximations for the xc (probably of the order of **150–200**).
- Most computer codes only include a very limited quantity of functionals, typically around **10–15**.
- Implementation of functionals is a time consuming task.
- Chemist and Physicists do not use the same functionals.

- Difficult to reproduce older calculations with older functionals.
- Difficult to reproduce calculations performed with other codes.
- Difficult to perform calculations with the newest functionals.

Why a library of xc functionals?

- The xc functional is at the heart of DFT.
- There are many approximations for the xc (probably of the order of **150–200**).
- Most computer codes only include a very limited quantity of functionals, typically around **10–15**.
- Implementation of functionals is a time consuming task.
- Chemist and Physicists do not use the same functionals.

- Difficult to reproduce older calculations with older functionals.
- Difficult to reproduce calculations performed with other codes.
- Difficult to perform calculations with the newest functionals.

Libxc

- Written in C from scratch.
- Bindings both in C and in Fortran.
- Lesser GNU general public license (v. 3.0).
- Automatic testing of the functionals.
- Contains functionals for 1D, 2D, and 3D calculations.
- Returns $\varepsilon_{\rm xc}$, $v_{\rm xc}$, $f_{\rm xc}$, and $k_{\rm xc}$.

Libxc

- Written in C from scratch.
- Bindings both in C and in Fortran.
- Lesser GNU general public license (v. 3.0).
- Automatic testing of the functionals.
- Contains functionals for 1D, 2D, and 3D calculations.
- Returns $\varepsilon_{\rm xc}$, $v_{\rm xc}$, $f_{\rm xc}$, and $k_{\rm xc}$.

Libxc

- Written in C from scratch.
- Bindings both in C and in Fortran.
- Lesser GNU general public license (v. 3.0).
- Automatic testing of the functionals.
- Contains functionals for 1D, 2D, and 3D calculations.
- Returns $\varepsilon_{\rm xc}$, $v_{\rm xc}$, $f_{\rm xc}$, and $k_{\rm xc}$.

Libxc

- Written in C from scratch.
- Bindings both in C and in Fortran.
- Lesser GNU general public license (v. 3.0).
- Automatic testing of the functionals.
- Contains functionals for 1D, 2D, and 3D calculations.
- Returns $\varepsilon_{\rm xc}$, $v_{\rm xc}$, $f_{\rm xc}$, and $k_{\rm xc}$.

Libxc

- Written in C from scratch.
- Bindings both in C and in Fortran.
- Lesser GNU general public license (v. 3.0).
- Automatic testing of the functionals.
- Contains functionals for 1D, 2D, and 3D calculations.
- Returns ε_{xc}, v_{xc}, f_{xc}, and k_{xc}.

Libxc

- Written in C from scratch.
- Bindings both in C and in Fortran.
- Lesser GNU general public license (v. 3.0).
- Automatic testing of the functionals.
- Contains functionals for 1D, 2D, and 3D calculations.
- Returns $\varepsilon_{\rm xc}$, $v_{\rm xc}$, $f_{\rm xc}$, and $k_{\rm xc}$.

Calling Libxc: a simple example

```
program lxctest
  use xc_f90_types_m
  use xc_f90_lib_m

  implicit none

  TYPE(xc_f90_pointer_t) :: xc_func, xc_info
  real(8) :: rho(4) = (/0.1, 0.2, 0.3, 0.4/), sigma(4) = (/0.2, 0.3, 0.4, 0.5/), zk(4)
  integer :: i
  character(len=120) :: s

  call xc_f90_func_init(xc_func, xc_info, XC_LDA_C_PW, XC_UNPOLARIZED)

  select case (xc_f90_info_family(xc_info))
  case(XC_FAMILY_LDA)
    call xc_f90_lda_exc(xc_func, 4, rho(1), zk(1))
  case(XC_FAMILY_GGA)
    call xc_f90_gga_exc(xc_func, 4, rho(1), sigma(1), zk(1))
  end select

  call xc_f90_info_name(xc_info, s)
  write(*, '(A)') trim(s)

   do i = 1, 4
     write(*,"(F8.6,1X,F9.6)") rho(i), zk(i)
   end do

  call xc_f90_func_end(xc_func)

end program lxctest
```

Where to find Libxc

http://www.tddft.org/programs/octopus/wiki/index.php/Libxc

Comput. Phys. Commun. **151**, 60–78 (2003)

Phys. Stat. Sol. B **243**, 2465–2488 (2006)

News

- Version 1.0 released on 2010-07-09.
- Stable API.
- Updated manual available on the web page.
- Packages available for Fedora 13/14/15 in the extras repository.
- Experimental Debian and Ubuntu packages available.
- More functionals.
- More derivatives.
- More codes using it.
- New type of functionals (kinetic energy density functionals)

News

- Version 1.0 released on 2010-07-09.
- Stable API.
- Updated manual available on the web page.
- Packages available for Fedora 13/14/15 in the extras repository.
- Experimental Debian and Ubuntu packages available.
- More functionals.
- More derivatives.
- More codes using it.
- New type of functionals (kinetic energy density functionals)

News

- Version 1.0 released on 2010-07-09.
- Stable API.
- Updated manual available on the web page.
- Packages available for Fedora 13/14/15 in the extras repository.
- Experimental Debian and Ubuntu packages available.
- More functionals.
- More derivatives.
- More codes using it.
- New type of functionals (kinetic energy density functionals)

News

- Version 1.0 released on 2010-07-09.
- Stable API.
- Updated manual available on the web page.
- Packages available for Fedora 13/14/15 in the extras repository.
- Experimental Debian and Ubuntu packages available.
- More functionals.
- More derivatives.
- More codes using it.
- New type of functionals (kinetic energy density functionals)

News

- Version 1.0 released on 2010-07-09.
- Stable API.
- Updated manual available on the web page.
- Packages available for Fedora 13/14/15 in the extras repository.
- Experimental Debian and Ubuntu packages available.
- More functionals.
- More derivatives.
- More codes using it.
- New type of functionals (kinetic energy density functionals)

News

- Version 1.0 released on 2010-07-09.
- Stable API.
- Updated manual available on the web page.
- Packages available for Fedora 13/14/15 in the extras repository.
- Experimental Debian and Ubuntu packages available.
- More functionals.
- More derivatives.
- More codes using it.
- New type of functionals (kinetic energy density functionals)

News

- Version 1.0 released on 2010-07-09.
- Stable API.
- Updated manual available on the web page.
- Packages available for Fedora 13/14/15 in the extras repository.
- Experimental Debian and Ubuntu packages available.
- More functionals.
- More derivatives.
- More codes using it.
- New type of functionals (kinetic energy density functionals)

News

- Version 1.0 released on 2010-07-09.
- Stable API.
- Updated manual available on the web page.
- Packages available for Fedora 13/14/15 in the extras repository.
- Experimental Debian and Ubuntu packages available.
- More functionals.
- More derivatives.
- More codes using it.
- New type of functionals (kinetic energy density functionals)

News

- Version 1.0 released on 2010-07-09.
- Stable API.
- Updated manual available on the web page.
- Packages available for Fedora 13/14/15 in the extras repository.
- Experimental Debian and Ubuntu packages available.
- More functionals.
- More derivatives.
- More codes using it.
- New type of functionals (kinetic energy density functionals)

More functionals

	2009	2011
LDA	19	26
GGA	55	93
Hybrids	24	24
MGGA	7	13

What is working

2009

	$\varepsilon_{\rm xc}$	$v_{\rm xc}$	$f_{\rm xc}$	$k_{\rm xc}$
LDA	OK	OK	OK	OK
GGA	OK	OK	PARTIAL	NO
HYB_GGA	OK	OK	PARTIAL	NO
MGGA	TEST	TEST	NO	NO

2011

	$\varepsilon_{\rm xc}$	$v_{\rm xc}$	$f_{\rm xc}$	$k_{\rm xc}$
LDA	OK	OK	OK	OK
GGA	OK	OK	PARTIAL	NO
HYB_GGA	OK	OK	PARTIAL	NO
MGGA	OK	OK	PARTIAL	NO

Codes using Libxc

- Octopus - real-space (TD)DFT code
- APE - atomic DFT code and pseudopotential generator
- GPAW - grid-based projector-augmented wave method
- ABINIT - plane-wave code
- BigDFT - wavelet code
- DP - Dielectric Properties, a linear response TDDFT code
- AtomPAW - projector augmented wave functions generator
- Elk - FP-LAPW code
- Yambo - solid state and molecular physics many-body calculations code
- Atomistix ToolKit - numerical orbitals code

Libxc in ABINIT

- Available for production since version 5.7.
- Interface done through the `libxc_functionals` module (`src/56_xc/m_libxc_functionals.F90`).
- What is working:
  - LDA and GGA functionals (ϵ_{xc}, v_{xc}, and f_{xc})
  - MGGA functionals for v_{xc} (NCPP only).

Libxc in ABINIT

- Available for production since version 5.7.
- Interface done through the `libxc_functionals` module (`src/56_xc/m_libxc_functionals.F90`).
- What is working:
  - LDA and GGA functionals for [illegible] and [illegible])
  - MGGA functionals for v_{xc} (NCPP only).

Libxc in ABINIT

- Available for production since version 5.7.
- Interface done through the `libxc_functionals` module (`src/56_xc/m_libxc_functionals.F90`).
- What is working:
  - LDA and GGA functionals ($e_{\rm xc}$, $v_{\rm xc}$, and $f_{\rm xc}$)
  - MGGA functionals for $v_{\rm xc}$ (NCPP only).

Libxc in ABINIT

- Available for production since version 5.7.
- Interface done through the `libxc_functionals` module (`src/56_xc/m_libxc_functionals.F90`).
- What is working:
  - LDA and GGA functionals ($e_{\rm xc}$, $v_{\rm xc}$, and $f_{\rm xc}$)
  - MGGA functionals for $v_{\rm xc}$ (NCPP only).

Libxc in ABINIT

- Available for production since version 5.7.
- Interface done through the `libxc_functionals` module (`src/56_xc/m_libxc_functionals.F90`).
- What is working:
  - LDA and GGA functionals (e_{xc}, v_{xc}, and f_{xc})
  - MGGA functionals for v_{xc} (NCPP only).

Using Libxc functionals in Abinit

- Compile ABINIT with Libxc support.
- Libxc functionals are accessed by negative values of **ixc**.
- Functionals are identified by a three-digit number.
- Combination of exchange and correlation functionals done by concatenation (**ixc** = -XXXCCC).
- MGGA functionals require the kinetic energy density (**usekedn** = 1)

Using Libxc functionals in Abinit

- Compile ABINIT with Libxc support.
- Libxc functionals are accessed by negative values of **ixc**.
- Functionals are identified by a three-digit number.
- Combination of exchange and correlation functionals done by concatenation (**ixc** = -XXXCCC).
- MGGA functionals require the kinetic energy density (**usekedn** = 1)

Using Libxc functionals in Abinit

- Compile ABINIT with Libxc support.
- Libxc functionals are accessed by negative values of **ixc**.
- Functionals are identified by a three-digit number.
- Combination of exchange and correlation functionals done by concatenation (**ixc** = -XXXCCC).
- MGGA functionals require the kinetic energy density (**usekedn** = 1)

Using Libxc functionals in Abinit

- Compile ABINIT with Libxc support.
- Libxc functionals are accessed by negative values of **ixc**.
- Functionals are identified by a three-digit number.
- Combination of exchange and correlation functionals done by concatenation (**ixc** = -XXXCCC).
- MGGA functionals require the kinetic energy density (**usekedn** = 1)

Using Libxc functionals in Abinit

- Compile ABINIT with Libxc support.
- Libxc functionals are accessed by negative values of **ixc**.
- Functionals are identified by a three-digit number.
- Combination of exchange and correlation functionals done by concatenation (**ixc** = -XXXCCC).
- MGGA functionals require the kinetic energy density (**usekedn** = 1)

Work in progress and future developments

- MGGA functionals for E_{xc} (A. Lherbier). This requires an extra term in the hamiltonian:

$$-\frac{1}{2}\boldsymbol{\nabla}\cdot\left[\frac{\partial e_{\mathrm{xc}}}{\partial\tau}\boldsymbol{\nabla}\varphi_i\right]$$

- PAW + MGGA functionals.
- ?

Work in progress and future developments

- MGGA functionals for $E_{\rm xc}$ (A. Lherbier). This requires an extra term in the hamiltonian:

$$-\frac{1}{2}\boldsymbol{\nabla}\cdot\left[\frac{\partial e_{\rm xc}}{\partial\tau}\boldsymbol{\nabla}\varphi_i\right]$$

- PAW + MGGA functionals.
- ?

Work in progress and future developments

- MGGA functionals for $E_{\rm xc}$ (A. Lherbier). This requires an extra term in the hamiltonian:

$$-\frac{1}{2}\boldsymbol{\nabla}\cdot\left[\frac{\partial e_{\rm xc}}{\partial\tau}\boldsymbol{\nabla}\varphi_i\right]$$

- PAW + MGGA functionals.
- ?

Acknowledgments

- Miguel Marques and the OCTOPUS developers
- Yann Pouillon
- Xavier Gonze
- Aurélien Lherbier
- Marc Torrent