Advances in Libxc

Micael Oliveira

5th International ABINIT Developer Workshop

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

1. Brief review of Libxc
2. Recent developments
3. Libxc in ABINIT
The main equations of DFT are the Kohn-Sham equations:

\[
\left[-\frac{1}{2} \nabla^2 + v_{\text{ext}}(r) + v_H(r) + v_{\text{xc}}(r) \right] \psi_i(r) = \epsilon_i \psi_i(r)
\]

where the exchange-correlation potential is defined as

\[
v_{\text{xc}}(r) = \frac{\delta E_{\text{xc}}}{\delta n(r)}
\]

In any practical application of the theory, we have to use an approximation to \(E_{\text{xc}}\), or \(v_{\text{xc}}(r)\).
The main equations of DFT are the Kohn-Sham equations:

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

where the exchange-correlation potential is defined as

\[
v_{\text{xc}}(r) = \frac{\delta E_{\text{xc}}}{\delta n(r)}
\]

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

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

Generalized gradient approximation:

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

Meta-generalized gradient approximation:

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

And more: orbital functionals, hybrid functionals, hyper-GGAs, etc.
Jacob’s ladder

Local density approximation:

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

Generalized gradient approximation:

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

Meta-generalized gradient approximation:

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

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

April 11, 2011, Han-sur-Lesse

Advances in Libxc
Jacob’s ladder

Local density approximation:

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

Generalized gradient approximation:

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

Meta-generalized gradient approximation:

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

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

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

Generalized gradient approximation:

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

Meta-generalized gradient approximation:

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

And more: orbital functionals, hybrid functionals, hyper-GGAs, etc.
What do we need to compute $v_{xc}$?

The energy is usually written as:

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

The potential in the LDA is:

$$v^{LDA}_{xc}(r) = \left. \frac{d}{dn} e^{LDA}_{xc}(n) \right|_{n=n(r)}$$

In the GGA:

$$v^{GGA}_{xc}(r) = \left. \frac{\partial}{\partial n} e^{LDA}_{xc}(n, \nabla n) \right|_{n=n(r)} - \nabla \left. \frac{\partial}{\partial (\nabla n)} e^{LDA}_{xc}(n, \nabla n) \right|_{n=n(r)}$$
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.

April 11, 2011, Han-sur-Lesse  Advances in Libxc
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 $\varepsilon_{xc}$, $v_{xc}$, $f_{xc}$, 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 $\varepsilon_{xc}$, $v_{xc}$, $f_{xc}$, 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 $\epsilon_{xc}$, $v_{xc}$, $f_{xc}$, 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 $\varepsilon_{xc}$, $v_{xc}$, $f_{xc}$, 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 $\varepsilon_{xc}$, $v_{xc}$, $f_{xc}$, 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 $\varepsilon_{xc}$, $v_{xc}$, $f_{xc}$, and $k_{xc}$. 
Calling Libxc: a simple example

```fortran
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
```

April 11, 2011, Han-sur-Lesse  Advances in Libxc
Where to find Libxc


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)
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

<table>
<thead>
<tr>
<th></th>
<th>2009</th>
<th>2011</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA</td>
<td>19</td>
<td>26</td>
</tr>
<tr>
<td>GGA</td>
<td>55</td>
<td>93</td>
</tr>
<tr>
<td>Hybrids</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>MGGA</td>
<td>7</td>
<td>13</td>
</tr>
</tbody>
</table>
## What is working

### 2009

<table>
<thead>
<tr>
<th></th>
<th>$\varepsilon_{xc}$</th>
<th>$v_{xc}$</th>
<th>$f_{xc}$</th>
<th>$k_{xc}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
</tr>
<tr>
<td>GGA</td>
<td>OK</td>
<td>OK</td>
<td>PARTIAL</td>
<td>NO</td>
</tr>
<tr>
<td>HYB_GGA</td>
<td>OK</td>
<td>OK</td>
<td>PARTIAL</td>
<td>NO</td>
</tr>
<tr>
<td>MGGA</td>
<td>TEST</td>
<td>TEST</td>
<td>NO</td>
<td>NO</td>
</tr>
</tbody>
</table>

### 2011

<table>
<thead>
<tr>
<th></th>
<th>$\varepsilon_{xc}$</th>
<th>$v_{xc}$</th>
<th>$f_{xc}$</th>
<th>$k_{xc}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
</tr>
<tr>
<td>GGA</td>
<td>OK</td>
<td>OK</td>
<td>PARTIAL</td>
<td>NO</td>
</tr>
<tr>
<td>HYB_GGA</td>
<td>OK</td>
<td>OK</td>
<td>PARTIAL</td>
<td>NO</td>
</tr>
<tr>
<td>MGGA</td>
<td>OK</td>
<td>OK</td>
<td>PARTIAL</td>
<td>NO</td>
</tr>
</tbody>
</table>
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
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 ($v_{xc}$, $f_{xc}$, and $j_{xc}$)
- MGGA functionals for $v_{xc}$ (NCPP only).
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 (\(e_{xc}\) and \(v_{xc}\))
- MGGA functionals for \(\tau_{xc}\) (NCPP only).
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).
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).
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 ($e_{\text{xc}}$, $v_{\text{xc}}$, and $f_{\text{xc}}$)
- MGGA functionals for $v_{\text{xc}}$ (NCPP only).
Using Libxc functionals in Abinit

- Compile ABINIT with Libxc support.
- Libxc functionals are accessed by negative values of \(i_{xc}\).
- Functionals are identified by a three-digit number.
- Combination of exchange and correlation functionals done by concatenation (\(i_{xc} = -XXXCCC\)).
- 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 $\text{ixc}$.
- Functionals are identified by a three-digit number.
- Combination of exchange and correlation functionals done by concatenation ($\text{ixc} = -\text{XXXCCC}$).
- MGGA functionals require the kinetic energy density ($\text{usekgedn} = 1$)
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 $\textit{ixc}$.
- Functionals are identified by a three-digit number.
- Combination of exchange and correlation functionals done by concatenation ($\textit{ixc} = -\text{XXXCCC}$).
- MGGA functionals require the kinetic energy density ($\textit{usekedn} = 1$)
Using Libxc functionals in Abinit

- Compile ABINIT with Libxc support.
- Libxc functionals are accessed by negative values of $i_{xc}$.
- Functionals are identified by a three-digit number.
- Combination of exchange and correlation functionals done by concatenation ($i_{xc} = -XXXCCC$).
- MGGA functionals require the kinetic energy density ($usekedn = 1$)
MGGA functionals for $E_{xc}$ (A. Lherbier). This requires an extra term in the Hamiltonian:

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

- PAW + MGGA functionals.
- ?
MGGA functionals for $E_{xc}$ (A. Lherbier). This requires an extra term in the Hamiltonian:

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

PAW + MGGA functionals.
MGGA functionals for $E_{xc}$ (A. Lherbier). This requires an extra term in the Hamiltonian:

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

PAW + MGGA functionals.

?
Acknowledgments

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