Libxc

a library of exchange and correlation functionals

Miguel A. L. Marques

[1]LPMCN, Université Claude Bernard Lyon 1 and CNRS, France
[2]European Theoretical Spectroscopy Facility

Octopus Workshop

Why the need for libxc?

- The xc functional is at the heart of DFT
- There are many approximations for the xc (probably of the order of **250–300**)
- Most computer codes only include a very limited quantity of functionals, typically around **10–15**
- 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 the need for libxc?

- The xc functional is at the heart of DFT
- There are many approximations for the xc (probably of the order of **250–300**)
- Most computer codes only include a very limited quantity of functionals, typically around **10–15**
- 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 the need for libxc?

- The xc functional is at the heart of DFT
- There are many approximations for the xc (probably of the order of **250–300**)
- Most computer codes only include a very limited quantity of functionals, typically around **10–15**
- 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 the need for libxc?

- The xc functional is at the heart of DFT
- There are many approximations for the xc (probably of the order of **250–300**)
- Most computer codes only include a very limited quantity of functionals, typically around **10–15**
- 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 the need for libxc?

- The xc functional is at the heart of DFT
- There are many approximations for the xc (probably of the order of **250–300**)
- Most computer codes only include a very limited quantity of functionals, typically around **10–15**
- 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 the need for libxc?

- The xc functional is at the heart of DFT
- There are many approximations for the xc (probably of the order of **250–300**)
- Most computer codes only include a very limited quantity of functionals, typically around **10–15**
- 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 the need for libxc?

- The xc functional is at the heart of DFT
- There are many approximations for the xc (probably of the order of **250–300**)
- Most computer codes only include a very limited quantity of functionals, typically around **10–15**
- 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

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:

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

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 $v_{\mathrm{xc}}(r)$.

Jacob's ladder

Local density approximation:

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

Generalized gradient approximation:

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

Meta-generalized gradient approximation:

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

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:

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

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

Lyon 1

Jacob's ladder

Local density approximation:

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

Generalized gradient approximation:

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

Meta-generalized gradient approximation:

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

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

Jacob's ladder

Local density approximation:

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

Generalized gradient approximation:

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

Meta-generalized gradient approximation:

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

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

What do we need? - I

The energy is usually written as:

$$E_{\mathrm{xc}} = \int \mathrm{d}^3 r\, e_{\mathrm{xc}}(r) = \int \mathrm{d}^3 r\, n(r) \epsilon_{\mathrm{xc}}(r)$$

The potential in the LDA is:

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

In the GGA:

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

What do we need? - II

For response properties we also need higher derivatives of e_{xc}

- 1st-order response (polarizabilities, phonon frequencies, etc.):

$$f_{\mathrm{xc}}^{\mathrm{LDA}}(r) = \frac{d^2}{d^2 n} e_{\mathrm{xc}}^{\mathrm{LDA}}(n)\Big|_{n=n(r)}$$

- 2st-order response (hyperpolarizabilities, etc.):

$$k_{\mathrm{xc}}^{\mathrm{LDA}}(r) = \frac{d^3}{d^3 n} e_{\mathrm{xc}}^{\mathrm{LDA}}(n)\Big|_{n=n(r)}$$

And let's not forget spin...

What do we need? - II

For response properties we also need higher derivatives of e_{xc}

- 1st-order response (polarizabilities, phonon frequencies, etc.):

$$f_{\mathrm{xc}}^{\mathrm{LDA}}(r) = \frac{d^2}{d^2 n} e_{\mathrm{xc}}^{\mathrm{LDA}}(n) \bigg|_{n=n(r)}$$

- 2st-order response (hyperpolarizabilities, etc.):

$$k_{\mathrm{xc}}^{\mathrm{LDA}}(r) = \frac{d^3}{d^3 n} e_{\mathrm{xc}}^{\mathrm{LDA}}(n) \bigg|_{n=n(r)}$$

And let's not forget spin...

An example: Perdew & Wang 91 (an LDA)

Perdew and Wang parametrized the correlation energy per unit particle:

$$e_c(r_s,\zeta) = e_c(r_s,0) + \alpha_c(r_s)\frac{f(\zeta)}{f''(0)}(1-\zeta^4) + [e_c(r_s,1) - e_c(r_s,0)]f(\zeta)\zeta^4$$

The function $f(\zeta)$ is

$$f(\zeta) = \frac{[1+\zeta]^{4/3} + [1-\zeta]^{4/3} - 2}{2^{4/3} - 2},$$

while its second derivative $f''(0) = 1.709921$. The functions $e_c(r_s,0)$, $e_c(r_s,1)$, and $-\alpha_c(r_s)$ are all parametrized by the function

$$g = -2A(1+\alpha_1 r_s)\log\left\{1 + \frac{1}{2A(\beta_1 r_s^{1/2} + \beta_2 r_s + \beta_3 r_s^{3/2} + \beta_4 r_s^2)}\right\}$$

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 at the moment **33** LDA functionals, **142** GGA functionals, **36** hybrids, and **14** mGGAs
- Contains functionals for exchange, correlation, and kinetic energy
- Contains functionals for 1D, 2D, and 3D calculations
- Returns $\varepsilon_{\rm xc}$, $v_{\rm xc}$, $f_{\rm xc}$, and $k_{\rm xc}$
- Quite mature: included in 16 codes (including abinit, espresso, cp2k, etc.)

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 at the moment **33** LDA functionals, **142** GGA functionals, **36** hybrids, and **14** mGGAs
- Contains functionals for exchange, correlation, and kinetic energy
- Contains functionals for 1D, 2D, and 3D calculations
- Returns $\varepsilon_{\rm xc}$, $v_{\rm xc}$, $f_{\rm xc}$, and $k_{\rm xc}$
- Quite mature: included in 16 codes (including abinit, espresso, cp2k, etc.)

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 at the moment **33** LDA functionals, **142** GGA functionals, **36** hybrids, and **14** mGGAs
- Contains functionals for exchange, correlation, and kinetic energy
- Contains functionals for 1D, 2D, and 3D calculations
- Returns $\varepsilon_{\rm xc}$, $v_{\rm xc}$, $f_{\rm xc}$, and $k_{\rm xc}$
- Quite mature: included in 16 codes (including abinit, espresso, cp2k, etc.)

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 at the moment **33** LDA functionals, **142** GGA functionals, **36** hybrids, and **14** mGGAs
- Contains functionals for exchange, correlation, and kinetic energy
- Contains functionals for 1D, 2D, and 3D calculations
- Returns $\varepsilon_{\rm xc}$, $v_{\rm xc}$, $f_{\rm xc}$, and $k_{\rm xc}$
- Quite mature: included in 16 codes (including abinit, espresso, cp2k, etc.)

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 at the moment **33** LDA functionals, **142** GGA functionals, **36** hybrids, and **14** mGGAs
- Contains functionals for exchange, correlation, and kinetic energy
- Contains functionals for 1D, 2D, and 3D calculations
- Returns $\varepsilon_{\rm xc}$, $v_{\rm xc}$, $f_{\rm xc}$, and $k_{\rm xc}$
- Quite mature: included in 16 codes (including abinit, espresso, cp2k, etc.)

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 at the moment **33** LDA functionals, **142** GGA functionals, **36** hybrids, and **14** mGGAs
- Contains functionals for exchange, correlation, and kinetic energy
- Contains functionals for 1D, 2D, and 3D calculations
- Returns $\varepsilon_{\rm xc}$, $v_{\rm xc}$, $f_{\rm xc}$, and $k_{\rm xc}$
- Quite mature: included in 16 codes (including abinit, espresso, cp2k, etc.)

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 at the moment **33** LDA functionals, **142** GGA functionals, **36** hybrids, and **14** mGGAs
- Contains functionals for exchange, correlation, and kinetic energy
- Contains functionals for 1D, 2D, and 3D calculations
- Returns $\varepsilon_{\rm xc}$, $v_{\rm xc}$, $f_{\rm xc}$, and $k_{\rm xc}$
- Quite mature: included in 16 codes (including abinit, espresso, cp2k, etc.)

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 at the moment **33** LDA functionals, **142** GGA functionals, **36** hybrids, and **14** mGGAs
- Contains functionals for exchange, correlation, and kinetic energy
- Contains functionals for 1D, 2D, and 3D calculations
- Returns $\varepsilon_{\rm xc}$, $v_{\rm xc}$, $f_{\rm xc}$, and $k_{\rm xc}$
- Quite mature: included in 16 codes (including abinit, espresso, cp2k, etc.)

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 at the moment **33** LDA functionals, **142** GGA functionals, **36** hybrids, and **14** mGGAs
- Contains functionals for exchange, correlation, and kinetic energy
- Contains functionals for 1D, 2D, and 3D calculations
- Returns $\varepsilon_{\rm xc}$, $v_{\rm xc}$, $f_{\rm xc}$, and $k_{\rm xc}$
- Quite mature: included in 16 codes (including abinit, espresso, cp2k, etc.)

What is working!

	$\varepsilon_{\mathrm{xc}}$	v_{xc}	f_{xc}	k_{xc}
LDA	OK	OK	OK	OK
GGA	OK	OK	OK	NO
HYB_GGA	OK	OK	OK	NO
mGGA	TEST	TEST	TEST	NO

An example in C

```
switch(xc_family_from_id(xc.functional))
{
case XC_FAMILY_LDA:
  if(xc.functional == XC_LDA_X)
    xc_lda_x_init(&lda_func, xc.nspin, 3, 0);
  else
    xc_lda_init(&lda_func, xc.functional, xc.nspin);
  xc_lda_vxc(&lda_func, xc.rho, &xc.zk, xc.vrho);
  xc_lda_end(&lda_func);
  break;
case XC_FAMILY_GGA:
  xc_gga_init(&gga_func, xc.functional, xc.nspin);
  xc_gga_vxc(&gga_func, xc.rho, xc.sigma, &xc.zk, xc.vrho, xc.vsigma
  xc_gga_end(&gga_func);
  break;
default:
  fprintf(stderr, "Functional '%d' not found\n", xc.functional);
  exit(1);
}
```

Another example in Fortran

```
program lxctest
  use libxc

  implicit none

  real(8) :: rho, e_c, v_c

  TYPE(xc_func) :: xc_c_func
  TYPE(xc_info) :: xc_c_info

  CALL xc_f90_lda_init(xc_c_func, xc_c_info, &
    XC_LDA_C_VWN, XC_UNPOLARIZED)
  CALL xc_f90_lda_vxc(xc_c_func, rho, e_c, v_c)
  CALL xc_f90_lda_end(xc_c_func)

end program lxctest
```

The info structure

```
typedef struct{
  int    number;    /* indentifier number */
  int    kind;      /* XC_EXCHANGE or XC_CORRELATION */

  char *name;       /* name of the functional, e.g. "PBE" */
  int    family;    /* type of the functional, e.g. XC_FAMILY_GGA */
  char *refs;       /* references                         */

  int    provides;  /* e.g. XC_PROVIDES_EXC | XC_PROVIDES_VXC */
  ...
} xc_func_info_type;
```

This is an example on how you can use it:

```
  xc_gga_type b88;

  xc_gga_init(&b88, XC_GGA_X_B88, XC_UNPOLARIZED);
  printf("The functional '%s' is defined in the reference(s):\n%s",
    b88.info->name, b88.info->refs);
  xc_gga_end(&b88);
```

The future

- More functionals!
- More derivatives!
- More codes using it!

The future

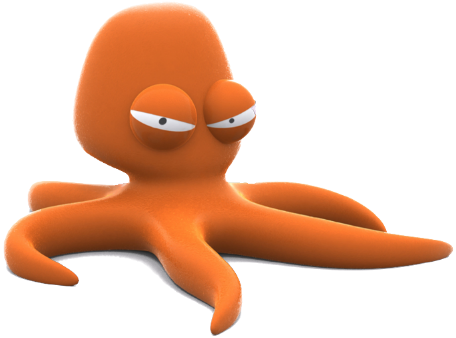
- More functionals!
- More derivatives!
- More codes using it!

The future

- More functionals!
- More derivatives!
- More codes using it!

Where to find us!

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



Libxc: a library of exchange and correlation functionals for DFT
M.A.L. Marques, M.J.T. Oliveira, and T. Burnus

Comput. Phys. Commun. **183**, 2272-2281 (2012)