Octopus: a package for simulation of electronic dynamics

Xavier Andrade <xavier@tddft.org>
Harvard University
and European Theoretical Spectroscopy Facility
Outline

The Octopus code
- simulation strategy
- physics
- parallelization and GPUs

Related projects

\texttt{Lib_{xc}}
exchange and correlation functionals library

\texttt{LibPSP}
a project for a pseudo-potentials library
Octopus

Electronic dynamics in time-dependent density functional theory (TDDFT)

Started in 2003 in San Sebastián

Several developers in different groups

An ETSF code

http://www.tddft.org/programs/octopus/
Octopus simulation strategy

Pseudopotential approximation
norm-conserving Troullier-Martins

Real-space grids
arbitrary shape
uniform spacing
Physics in Octopus

1D, 2D, 3D and 4D systems

LDA, GGA, mGGA, EXX, Hartree-Fock and hybrids

Finite and periodic systems

Ground state

Excited states: Real-time, Casida and Sternheimer
Real-time TDDFT

Optical absorption spectra

Response to lasers and strong fields

Quantum optimal control

Adiabatic and non-adiabatic molecular dynamics
The code

Free software: GPL license

150k lines of code

Mainly Fortran 95 and some C

Object oriented

Extensive use of libraries: GSL, spglib, Metis...
Multi-level parallelization

- K-points / Spin
- Kohn-Sham states
- Real-space domains
  - OpenMP threads
  - Vectorization
  - OpenCL tasks
Domain parallelization

Only region boundaries need to be communicated
Vectorization

- Perform the same operation on several values
- Modern processors have vectorial FPUs
  Accessible by C compiler directives
- GPUs behave as vector processors
  OpenCL framework
- Vectors formed of blocks of KS orbitals
Comparison in performance of the finite difference Laplacian operator

CPU uses 4 threads
GPU is 4 times faster
Total speed-up: 3x
Large scale parallelization

Clorophyll molecule: 1350 atoms
Jugene - Blue Gene/P
States + domains + threads + vectorization
Libxc

a library of exchange and correlation functionals
Why an exchange and correlation functionals library?

XC functional: essential part of (TD)DFT

More than 150 functionals proposed

Codes include around 10-20 functionals

Difficult to reproduce calculations
About Libxc

Main author: M.A.L. Marques

Written from the scratch in C

C and Fortran interfaces

Free software: LGPL 3.0 license

Included in several codes: Octopus, APE, GPAW, Abinit, Elk, BigDFT, and RGWBS
# Libxc features

LDA, GGA, hybrid and mGGA functionals

1D, 2D and 3D functionals

Functionals for energy, potential and kernels

<table>
<thead>
<tr>
<th></th>
<th>number</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>23</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
</tr>
<tr>
<td>GGA</td>
<td>63</td>
<td>OK</td>
<td>OK</td>
<td>PARTIAL</td>
<td>NO</td>
</tr>
<tr>
<td>HYB_GGA</td>
<td>24</td>
<td>OK</td>
<td>OK</td>
<td>PARTIAL</td>
<td>NO</td>
</tr>
<tr>
<td>MGGA</td>
<td>11</td>
<td>OK</td>
<td>OK</td>
<td>NO</td>
<td>NO</td>
</tr>
</tbody>
</table>
LibPSP: a project for a pseudo-potentials library
Technical problems with pseudo-potentials

Almost each code uses a different format

Significant amount of code (replicated)

Difficult to use a pseudo-potential with different codes
Pseudo-potential converter

pspconvert (formerly known as pp2pp)

ETSF utility to convert between formats

Limited capabilities

Does not solve the problem of code replication

http://www.etsf.eu/resources/software/libraries_and_tools/
LibPSP: a pseudo-potential library

ETSF initiative

To be implemented in C from scratch

Major pseudo-potential formats

Return orbitals and potential in any point of space

PAW setups or other data?
Conclusion

Octopus: flexibility and performance

Independent and shared components

Other opportunities for code integration/sharing

Infrastructure to support components
Main Octopus developers

J. Alberdi, N. Helbig and A. Rubio
University of the Basque Country, Spain
M.A.L. Marques
Université Lyon I, France
A. Castro
Universidad de Zaragoza, Spain
D. Strubbe
UC Berkeley, USA
D. Nitsche
MPI Halle, Germany
M. Oliveira and F. Nogueira
Universidade de Coimbra, Portugal
M. Verstraete
Université de Liege, Belgium
program lxctest
    use xc_f90_types_m
    use xc_f90_lib_m

    implicit none

    real(8) :: rho, e_c, v_c

    TYPE(xc_f90_func_t) :: xc_c_func
    TYPE(xc_f90_info_t) :: xc_c_info

    CALL xc_f90_func_init(xc_c_func, xc_c_info, 
                          &
                          XC_LDA_C_VWN, XC_UNPOLARIZED)
    CALL xc_f90_lda_exc_vxc(xc_c_func, 1, rho, e_c, v_c)
    CALL xc_f90_func_end(xc_c_func)

end program lxctest