# Introduction to Octopus: a real-space (TD)DFT code

David A. Strubbe
and the Octopus development team

MIT IAP, Jan 2016

## Time-dependent Kohn-Sham equation

$$
\begin{aligned}
i\frac{\partial}{\partial t}\varphi_n(\boldsymbol{r},t) &= -\nabla^2\varphi_n + V_{\text{eff}}\left[\rho\right](\boldsymbol{r},t)\varphi_n(\boldsymbol{r},t) \\
\rho(\boldsymbol{r},t) &= \sum_n \varphi_n^*(\boldsymbol{r},t)\varphi_n(\boldsymbol{r},t)
\end{aligned}
$$

- Solve the equations numerically.
- Represent functions and other objects.
- Calculate derivatives and integrals.

# Introduction

## Time-dependent Kohn-Sham equation

$$
\begin{aligned}
i\frac{\partial}{\partial t}\varphi_n(\boldsymbol{r},t) &= -\nabla^2\varphi_n + V_{\text{eff}}\left[\rho\right](\boldsymbol{r},t)\varphi_n(\boldsymbol{r},t) \\
\rho(\boldsymbol{r},t) &= \sum_n \varphi_n^*(\boldsymbol{r},t)\varphi_n(\boldsymbol{r},t)
\end{aligned}
$$

- Solve the equations numerically.
- Represent functions and other objects.
- Calculate derivatives and integrals.

# Introduction

## Time-dependent Kohn-Sham equation

$$
\begin{aligned}
i\frac{\partial}{\partial t}\varphi_n(\boldsymbol{r},t) &= -\nabla^2\varphi_n + V_{\text{eff}}\left[\rho\right](\boldsymbol{r},t)\varphi_n(\boldsymbol{r},t) \\
\rho(\boldsymbol{r},t) &= \sum_n \varphi_n^*(\boldsymbol{r},t)\varphi_n(\boldsymbol{r},t)
\end{aligned}
$$

- Solve the equations numerically.
- Represent functions and other objects.
- Calculate derivatives and integrals.

# Pseudo-potentials

- The atomic potential is very strong and “hard” (small spacing or high plane-wave cutoff required).
- Core electrons are almost independent of the environment.
- Replace the potential and core electrons by a pseudo-potential.

$$V = V_{loc} + \sum_{lm} |lm\rangle \left( V_l - V_{loc} \right) \langle lm|$$

# Pseudo-potentials

- The atomic potential is very strong and “hard” (small spacing or high plane-wave cutoff required).
- Core electrons are almost independent of the environment.
- Replace the potential and core electrons by a pseudo-potential.

**Norm-conserving pseudo-potentials in Kleinman-Bylander form**

$$V = V_{\mathrm{loc}} + \sum_{lm} |lm\rangle \left(V_l - V_{\mathrm{loc}}\right) \langle lm|$$

# Pseudo-potentials

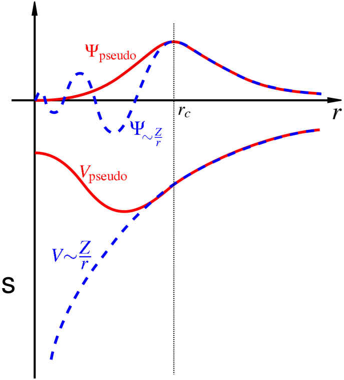
- The atomic potential is very strong and "hard" (small spacing or high plane-wave cutoff required).
- Core electrons are almost independent of the environment.
- Replace the potential and core electrons by a pseudo-potential.

**Norm-conserving pseudo-potentials in Kleinman-Bylander form**

$$V = V_{\text{loc}} + \sum_{lm} |lm\rangle \left(V_l - V_{\text{loc}}\right) \langle lm|$$

# Pseudo-potentials

- The atomic potential is very strong and “hard” (small spacing or high plane-wave cutoff required).
- Core electrons are almost independent of the environment.
- Replace the potential and core electrons by a pseudo-potential.




## Norm-conserving pseudo-potentials in Kleinman-Bylander form

$$V = V_{\mathrm{loc}} + \sum_{lm} |lm\rangle \left(V_l - V_{\mathrm{loc}}\right) \langle lm|$$

# Real-space grid

- Partial differential equation with infinite degrees of freedom.
- Reduce to a finite number.
- Functions are represented by values on a set of points.
- Point distribution:
  - Uniformly spaced grid.
  - Distance between points is constant: *Spacing*.
  - Non-uniform grids also possible.
- Finite region of the space: *Box*

# Real-space grid

- Partial differential equation with infinite degrees of freedom.
- Reduce to a finite number.
- Functions are represented by values on a set of points.
- Point distribution:
  - Uniformly spaced grid.
  - Distance between points is constant: *Spacing*.
  - Non-uniform grids also possible.
- Finite region of the space: *Box*

# Real-space grid

- Partial differential equation with infinite degrees of freedom.
- Reduce to a finite number.
- Functions are represented by values on a set of points.
- Point distribution:
  - Uniformly spaced grid.
  - Distance between points is constant: *Spacing*.
  - Non-uniform grids also possible.
- Finite region of the space: *Box*

# Real-space grid

- Partial differential equation with infinite degrees of freedom.
- Reduce to a finite number.
- Functions are represented by values on a set of points.
- Point distribution:
  - Uniformly spaced grid.
  - Distance between points is constant: *Spacing*.
  - Non-uniform grids also possible.
- Finite region of the space: *Box*

# Real-space grid

- Partial differential equation with infinite degrees of freedom.
- Reduce to a finite number.
- Functions are represented by values on a set of points.
- Point distribution:
  - Uniformly spaced grid.
  - Distance between points is constant: *Spacing*.
  - Non-uniform grids also possible.
- Finite region of the space: *Box*

# Real-space grid

- Partial differential equation with infinite degrees of freedom.
- Reduce to a finite number.
- Functions are represented by values on a set of points.
- Point distribution:
  - Uniformly spaced grid.
  - Distance between points is constant: *Spacing*.
  - Non-uniform grids also possible.
- Finite region of the space: *Box*

# Real-space grid

- Partial differential equation with infinite degrees of freedom.
- Reduce to a finite number.
- Functions are represented by values on a set of points.
- Point distribution:
  - Uniformly spaced grid.
  - Distance between points is constant: *Spacing*.
  - Non-uniform grids also possible.
- Finite region of the space: *Box*

# Real-space grid

- Partial differential equation with infinite degrees of freedom.
- Reduce to a finite number.
- Functions are represented by values on a set of points.
- Point distribution:
  - Uniformly spaced grid.
  - Distance between points is constant: *Spacing*.
  - Non-uniform grids also possible.
- Finite region of the space: *Box*

# Boundary conditions

- For finite systems, functions go to zero.
- Force functions to go to zero on the border of the box.
- The box has to be large enough to contain the functions.
- Other BCs are possible: periodic, zero derivative, open.

# Boundary conditions

- For finite systems, functions go to zero.
- Force functions to go to zero on the border of the box.
- The box has to be large enough to contain the functions.
- Other BCs are possible: periodic, zero derivative, open.

# Boundary conditions

- For finite systems, functions go to zero.
- Force functions to go to zero on the border of the box.
- The box has to be large enough to contain the functions.
- Other BCs are possible: periodic, zero derivative, open.

# Boundary conditions

- For finite systems, functions go to zero.
- Force functions to go to zero on the border of the box.
- The box has to be large enough to contain the functions.
- Other BCs are possible: periodic, zero derivative, open.

# Boundary conditions

- Optimize the shape of the box to minimize the number of points needed.
- Available box shapes:
  - Minimum box: union of spheres around each atom.
  - Sphere.
  - Cylinder.
  - Parallelepiped.
  - Arbitrary (e.g. 2D image!)

# Boundary conditions

- Optimize the shape of the box to minimize the number of points needed.
- Available box shapes:
  - Minimum box: union of spheres around each atom.
  - Sphere.
  - Cylinder.
  - Parallelepiped.
  - Arbitrary (*e.g.* 2D image!)

# Boundary conditions

- Optimize the shape of the box to minimize the number of points needed.
- Available box shapes:
  - Minimum box: union of spheres around each atom.
  - Sphere.
  - Cylinder.
  - Parallelepiped.
  - Arbitrary (*e.g.* 2D image!)

# Boundary conditions

- Optimize the shape of the box to minimize the number of points needed.
- Available box shapes:
  - Minimum box: union of spheres around each atom.
  - Sphere.
  - Cylinder.
  - Parallelepiped.
  - Arbitrary (*e.g.* 2D image!)

# Boundary conditions

- Optimize the shape of the box to minimize the number of points needed.
- Available box shapes:
  - Minimum box: union of spheres around each atom.
  - Sphere.
  - Cylinder.
  - Parallelepiped.
  - Arbitrary (*e.g.* 2D image!)

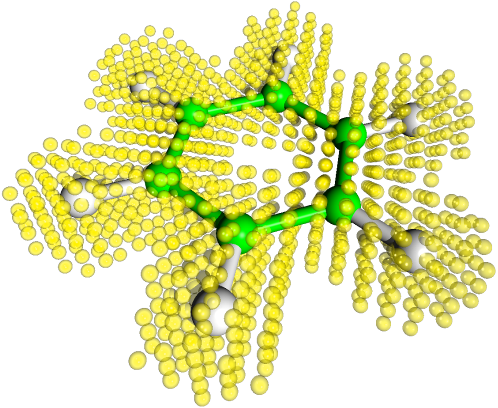
# Boundary conditions

- Optimize the shape of the box to minimize the number of points needed.
- Available box shapes:
  - Minimum box: union of spheres around each atom.
  - Sphere.
  - Cylinder.
  - Parallelepiped.
  - Arbitrary (*e.g.* 2D image!)

# Boundary conditions

- Optimize the shape of the box to minimize the number of points needed.
- Available box shapes:
  - Minimum box: union of spheres around each atom.
  - Sphere.
  - Cylinder.
  - Parallelepiped.
  - Arbitrary (*e.g.* 2D image!)

# Example: benzene molecule in minimal box



# Real-space grid characteristics

- Natural boundary conditions for different problems:
  zero, one, two, or three periodic dimensions for molecules, wires, sheets, and solids.
- Representation used for calculating $V_{\mathrm{xc}}[\rho]$ even with other bases.
- Can systematically improve discretization quality:
  - Decrease the spacing (like increasing plane-wave cutoff).
  - Increase the box size (in finite directions).
- Orthogonal “basis set”.
- Unbiased, independent of atomic positions (no Pulay forces).
- Problems:
  - Breaking of translational invariance: egg-box effect
  - Breaking of rotational invariance.
  - (Decreasing spacing helps both.)

# Real-space grid characteristics

- Natural boundary conditions for different problems:
  zero, one, two, or three periodic dimensions for molecules, wires, sheets, and solids.
- Representation used for calculating $V_{\mathrm{xc}}[\rho]$ even with other bases.
- Can systematically improve discretization quality:
  - Decrease the spacing (like increasing plane-wave cutoff).
  - Increase the box size (in finite directions).
- Orthogonal “basis set”.
- Unbiased, independent of atomic positions (no Pulay forces).
- Problems:

# Real-space grid characteristics

- Natural boundary conditions for different problems:
  zero, one, two, or three periodic dimensions for molecules, wires, sheets, and solids.
- Representation used for calculating $V_{\mathrm{xc}}[\rho]$ even with other bases.
- Can systematically improve discretization quality:
  - Decrease the spacing (like increasing plane-wave cutoff).
  - Increase the box size (in finite directions).
- Orthogonal "basis set".
- Unbiased, independent of atomic positions (no Pulay forces).
- Problems:

# Real-space grid characteristics

- Natural boundary conditions for different problems:
  zero, one, two, or three periodic dimensions for molecules, wires, sheets, and solids.
- Representation used for calculating $V_{\mathrm{xc}}\left[\rho\right]$ even with other bases.
- Can systematically improve discretization quality:
  - Decrease the spacing (like increasing plane-wave cutoff).
  - Increase the box size (in finite directions).
- Orthogonal "basis set".
- Unbiased, independent of atomic positions (no Pulay forces).
- Problems:
  - Breaking of translational invariance: egg-box effect.
  - Breaking of rotational invariance.
  - (Decreasing spacing helps both.)

# Real-space grid characteristics

- Natural boundary conditions for different problems:
  zero, one, two, or three periodic dimensions for molecules, wires, sheets, and solids.
- Representation used for calculating $V_{\mathrm{xc}}[\rho]$ even with other bases.
- Can systematically improve discretization quality:
  - Decrease the spacing (like increasing plane-wave cutoff).
  - Increase the box size (in finite directions).
- Orthogonal “basis set”.
- Unbiased, independent of atomic positions (no Pulay forces).
- Problems:
  - Breaking of translational invariance: egg-box effect.
  - Breaking of rotational invariance.
  - (Decreasing spacing helps both.)

# Real-space grid characteristics

- Natural boundary conditions for different problems:
  zero, one, two, or three periodic dimensions for molecules, wires, sheets, and solids.
- Representation used for calculating $V_{\mathrm{xc}}[\rho]$ even with other bases.
- Can systematically improve discretization quality:
  - Decrease the spacing (like increasing plane-wave cutoff).
  - Increase the box size (in finite directions).
- Orthogonal "basis set".
- Unbiased, independent of atomic positions (no Pulay forces).
- Problems:
  - Breaking of translational invariance: egg-box effect.
  - Breaking of rotational invariance.
  - (Decreasing spacing helps both.)

# Real-space grid characteristics

- Natural boundary conditions for different problems:
  zero, one, two, or three periodic dimensions for molecules, wires, sheets, and solids.
- Representation used for calculating $V_{\mathrm{xc}}[\rho]$ even with other bases.
- Can systematically improve discretization quality:
  - Decrease the spacing (like increasing plane-wave cutoff).
  - Increase the box size (in finite directions).
- Orthogonal "basis set".
- Unbiased, independent of atomic positions (no Pulay forces).
- Problems:
  - Breaking of translational invariance: egg-box effect.
  - Breaking of rotational invariance.
  - (Decreasing spacing helps both.)

# Real-space grid characteristics

- Natural boundary conditions for different problems:
  zero, one, two, or three periodic dimensions for molecules, wires, sheets, and solids.
- Representation used for calculating $V_{\mathrm{xc}}\left[\rho\right]$ even with other bases.
- Can systematically improve discretization quality:
  - Decrease the spacing (like increasing plane-wave cutoff).
  - Increase the box size (in finite directions).
- Orthogonal "basis set".
- Unbiased, independent of atomic positions (no Pulay forces).
- Problems:
  - Breaking of translational invariance: egg-box effect.
  - Breaking of rotational invariance.
  - (Decreasing spacing helps both.)

# Real-space grid characteristics

- Natural boundary conditions for different problems:
  zero, one, two, or three periodic dimensions for molecules, wires, sheets, and solids.
- Representation used for calculating $V_{\mathrm{xc}}\left[\rho\right]$ even with other bases.
- Can systematically improve discretization quality:
  - Decrease the spacing (like increasing plane-wave cutoff).
  - Increase the box size (in finite directions).
- Orthogonal "basis set".
- Unbiased, independent of atomic positions (no Pulay forces).
- Problems:
  - Breaking of translational invariance: egg-box effect.
  - Breaking of rotational invariance.
  - (Decreasing spacing helps both.)

# Real-space grid characteristics

- Natural boundary conditions for different problems:
  zero, one, two, or three periodic dimensions for molecules, wires, sheets, and solids.
- Representation used for calculating $V_{\mathrm{xc}}[\rho]$ even with other bases.
- Can systematically improve discretization quality:
  - Decrease the spacing (like increasing plane-wave cutoff).
  - Increase the box size (in finite directions).
- Orthogonal “basis set”.
- Unbiased, independent of atomic positions (no Pulay forces).
- Problems:
  - Breaking of translational invariance: egg-box effect.
  - Breaking of rotational invariance.
  - (Decreasing spacing helps both.)

# Real-space grid characteristics

- Natural boundary conditions for different problems:
  zero, one, two, or three periodic dimensions for molecules, wires, sheets, and solids.
- Representation used for calculating $V_{\mathrm{xc}}[\rho]$ even with other bases.
- Can systematically improve discretization quality:
  - Decrease the spacing (like increasing plane-wave cutoff).
  - Increase the box size (in finite directions).
- Orthogonal "basis set".
- Unbiased, independent of atomic positions (no Pulay forces).
- Problems:
  - Breaking of translational invariance: egg-box effect.
  - Breaking of rotational invariance.
  - (Decreasing spacing helps both.)

# Derivatives

- Derivative at a point: sum over neighboring points.
- The coefficients $c_{ij}$ depend on the mesh and number of points used: *the stencil*.
- General form for Laplacian:

$$\nabla^2 f(n_x h, n_y h) = \sum_i^n \sum_j^n \frac{c_{ij}}{h} f(n_x h + ih, n_y h + jh)$$

- Compare definition of derivative:

$$f'(x_0) = \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{\Delta x}$$

- More points $\rightarrow$ more precision.
- Semi-local operation.

# Derivatives

- Derivative at a point: sum over neighboring points.
- The coefficients $c_{ij}$ depend on the mesh and number of points used: *the stencil*.
- General form for Laplacian:

$$\nabla^2 f(n_x h, n_y h) = \sum_i^n \sum_j^n \frac{c_{ij}}{h} f(n_x h + ih, n_y h + jh)$$

- Compare definition of derivative:

$$f'(x_0) = \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{\Delta x}$$

- More points $\rightarrow$ more precision.
- Semi-local operation.

# Derivatives

- Derivative at a point: sum over neighboring points.
- The coefficients $c_{ij}$ depend on the mesh and number of points used: *the stencil*.
- General form for Laplacian:

$$\nabla^2 f(n_x h, n_y h) = \sum_i^n \sum_j^n \frac{c_{ij}}{h} f(n_x h + ih, n_y h + jh)$$

- Compare definition of derivative:

$$f'(x_0) = \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{\Delta x}$$

- More points $\rightarrow$ more precision.
- Semi-local operation.

# Derivatives

- Derivative at a point: sum over neighboring points.
- The coefficients $c_{ij}$ depend on the mesh and number of points used: *the stencil*.
- General form for Laplacian:

$$\nabla^2 f(n_x h, n_y h) = \sum_i^n \sum_j^n \frac{c_{ij}}{h} f(n_x h + ih, n_y h + jh)$$

- Compare definition of derivative:

$$f'(x_0) = \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{\Delta x}$$

- More points $\rightarrow$ more precision.
- Semi-local operation.

# Derivatives

- Derivative at a point: sum over neighboring points.
- The coefficients $c_{ij}$ depend on the mesh and number of points used: *the stencil*.
- General form for Laplacian:

$$\nabla^2 f(n_x h, n_y h) = \sum_i^n \sum_j^n \frac{c_{ij}}{h} f(n_x h + ih, n_y h + jh)$$

- Compare definition of derivative:

$$f'(x_0) = \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{\Delta x}$$

- More points $\rightarrow$ more precision.
- Semi-local operation.

# Derivatives

- Derivative at a point: sum over neighboring points.
- The coefficients $c_{ij}$ depend on the mesh and number of points used: *the stencil*.
- General form for Laplacian:

$$\nabla^2 f(n_x h,\, n_y h) = \sum_i^n \sum_j^n \frac{c_{ij}}{h} f(n_x h + ih,\, n_y h + jh)$$

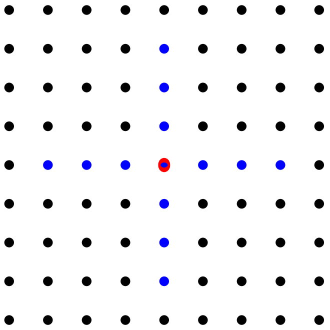
- Compare definition of derivative:

$$f'(x_0) = \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{\Delta x}$$

- More points $\rightarrow$ more precision.
- Semi-local operation.

# Example of stencil for Laplacian

Symmetric third-order in 2D.



# Integration

## Trapezoidal rule

$$\int f(x,y)\,dx\,dy = h^2 \sum_{ij} f(ih, jh)$$

- Sum over grid points.

# Integration

## Trapezoidal rule

$$\int f(x,y)\,dx\,dy = h^2 \sum_{ij} f(ih, jh)$$

- Sum over grid points.

- What we want to solve:

**Kohn-Sham equations**

$$-\nabla^2 \varphi_n + V_{\text{eff}}[\rho](\boldsymbol{r})\varphi_n = \epsilon_n \varphi_n$$

- We use a self-consistency scheme to treat non-linearity.
- Solve for eigenstates at fixed $V_{\text{eff}}$, then update $\rho$ and $V_{\text{eff}}$.

# Ground-state calculations

- What we want to solve:

**Kohn-Sham equations**

$$-\nabla^2\varphi_n + V_{\text{eff}}\left[\rho\right](\boldsymbol{r})\varphi_n = \epsilon_n\varphi_n$$

- We use a self-consistency scheme to treat non-linearity.
- Solve for eigenstates at fixed $V_{\text{eff}}$, then update $\rho$ and $V_{\text{eff}}$.

- What we want to solve:

**Kohn-Sham equations**

$$-\nabla^2 \varphi_n + V_{\text{eff}}\left[\rho\right](\boldsymbol{r})\varphi_n = \epsilon_n \varphi_n$$

- We use a self-consistency scheme to treat non-linearity.
- Solve for eigenstates at fixed $V_{\text{eff}}$, then update $\rho$ and $V_{\text{eff}}$.

# Ground-state calculations

- What we want to solve:

**Kohn-Sham equations**

$$-\nabla^2\varphi_n + V_{\text{eff}}[\rho](\boldsymbol{r})\varphi_n = \epsilon_n\varphi_n$$

- We use a self-consistency scheme to treat non-linearity.
- Solve for eigenstates at fixed $V_{\text{eff}}$, then update $\rho$ and $V_{\text{eff}}$.

# Discretization of the Hamiltonian

- For the Laplacian (kinetic energy) we use finite differences.
- The local part of the potential can be applied directly.
- The non-local potential is applied in a small spherical grid around the atoms.
- The Hamiltonian becomes a finite-size matrix.

# Discretization of the Hamiltonian

- For the Laplacian (kinetic energy) we use finite differences.
- The local part of the potential can be applied directly.
- The non-local potential is applied in a small spherical grid around the atoms.
- The Hamiltonian becomes a finite-size matrix.

# Discretization of the Hamiltonian

- For the Laplacian (kinetic energy) we use finite differences.
- The local part of the potential can be applied directly.
- The non-local potential is applied in a small spherical grid around the atoms.
- The Hamiltonian becomes a finite-size matrix.

- For the Laplacian (kinetic energy) we use finite differences.
- The local part of the potential can be applied directly.
- The non-local potential is applied in a small spherical grid around the atoms.
- The Hamiltonian becomes a finite-size matrix.

# The eigenproblem

- Find the eigenvectors and eigenvalues of a matrix.
- Very large matrix with lots of zero components (*Sparse*).
- Use iterative solvers where only the action of the matrix is required (various options available in the code).

# The eigenproblem

- Find the eigenvectors and eigenvalues of a matrix.
- Very large matrix with lots of zero components (*Sparse*).
- Use iterative solvers where only the action of the matrix is required (various options available in the code).

# The eigenproblem

- Find the eigenvectors and eigenvalues of a matrix.
- Very large matrix with lots of zero components (*Sparse*).
- Use iterative solvers where only the action of the matrix is required (various options available in the code).

## The eigensolver

- We minimize (using conjugate gradient or other method):

**Rayleigh-Ritz quotient**

$$\epsilon(\psi) = \frac{\langle\psi|H|\psi\rangle}{\langle\psi|\psi\rangle}$$

- Works for the first state.
- For higher-energy states, it is necessary to orthogonalize against the lower ones.

# The eigensolver

- We minimize (using conjugate gradient or other method):

**Rayleigh-Ritz quotient**

$$\epsilon(\psi) = \frac{\langle\psi|H|\psi\rangle}{\langle\psi|\psi\rangle}$$

- Works for the first state.
- For higher-energy states, it is necessary to orthogonalize against the lower ones.

# The eigensolver

- We minimize (using conjugate gradient or other method):

**Rayleigh-Ritz quotient**

$$\epsilon(\psi) = \frac{\langle\psi|H|\psi\rangle}{\langle\psi|\psi\rangle}$$

- Works for the first state.
- For higher-energy states, it is necessary to orthogonalize against the lower ones.

# The eigensolver

- We minimize (using conjugate gradient or other method):

**Rayleigh-Ritz quotient**

$$\epsilon(\psi) = \frac{\langle\psi|H|\psi\rangle}{\langle\psi|\psi\rangle}$$

- Works for the first state.
- For higher-energy states, it is necessary to orthogonalize against the lower ones.

# Time-propagation

- Given an initial condition, solve the:

**Time-dependent Kohn-Sham equation**

$$i\frac{\partial \varphi_k}{\partial t} = -\nabla^2 \varphi_k + V_{\mathrm{eff}}\left[\rho\right](\boldsymbol{r}, t)\varphi_k$$

- Various numerical schemes of doing the time-propagation.
- Many properties can be obtained.
- Response to time-dependent fields: lasers.

# Time-propagation

- Given an initial condition, solve the:

## Time-dependent Kohn-Sham equation

$$i\frac{\partial \varphi_k}{\partial t} = -\nabla^2 \varphi_k + V_{\text{eff}}\left[\rho\right](\boldsymbol{r}, t)\varphi_k$$

- Various numerical schemes of doing the time-propagation.
- Many properties can be obtained.
- Response to time-dependent fields: lasers.

# Time-propagation

- Given an initial condition, solve the:

**Time-dependent Kohn-Sham equation**

$$i\frac{\partial \varphi_k}{\partial t} = -\nabla^2 \varphi_k + V_{\text{eff}}\left[\rho\right](\boldsymbol{r}, t)\varphi_k$$

- Various numerical schemes of doing the time-propagation.
- Many properties can be obtained.
- Response to time-dependent fields: lasers.

# Time-propagation

- Given an initial condition, solve the:

**Time-dependent Kohn-Sham equation**

$$i\frac{\partial \varphi_k}{\partial t} = -\nabla^2 \varphi_k + V_{\text{eff}}\left[\rho\right](\boldsymbol{r}, t)\varphi_k$$

- Various numerical schemes of doing the time-propagation.
- Many properties can be obtained.
- Response to time-dependent fields: lasers.

# Time-propagation

- Given an initial condition, solve the:

**Time-dependent Kohn-Sham equation**

$$i\frac{\partial \varphi_k}{\partial t} = -\nabla^2 \varphi_k + V_{\text{eff}}\left[\rho\right](\boldsymbol{r}, t)\varphi_k$$

- Various numerical schemes of doing the time-propagation.
- Many properties can be obtained.
- Response to time-dependent fields: lasers.

# Absorption spectra from time-propagation

- Start from the ground state, with a ‘kick.’

**Time-dependent potential**

$$V(\boldsymbol{r},t) = \boldsymbol{\kappa}\delta(t) \qquad \Rightarrow \qquad \psi \rightarrow \psi e^{i\boldsymbol{k}\cdot\boldsymbol{r}}$$

- Time-propagate and get the dipole $\boldsymbol{d}(t)$ as a function of time.

# Absorption spectra from time-propagation

- Start from the ground state, with a 'kick.'

## Time-dependent potential

$$V(\boldsymbol{r}, t) = \boldsymbol{\kappa}\delta(t) \qquad \Rightarrow \qquad \psi \to \psi e^{i\boldsymbol{k}\cdot\boldsymbol{r}}$$

- Time-propagate and get the dipole $d(t)$ as a function of time.

## Polarizability tensor

$$\alpha_{ij}(\omega) = -\frac{1}{\kappa_j}\int dt\, e^{i\omega t} d_j(t)$$

# Absorption spectra from time-propagation

- Start from the ground state, with a 'kick.'

**Time-dependent potential**

$$V(\boldsymbol{r}, t) = \boldsymbol{\kappa}\delta(t) \qquad \Rightarrow \qquad \psi \to \psi e^{i\boldsymbol{k}\cdot\boldsymbol{r}}$$

- Time-propagate and get the dipole $\boldsymbol{d}(t)$ as a function of time.

**Polarizability tensor**

$$\alpha_{ij}(\omega) = -\frac{1}{\kappa_i}\int dt\, e^{i\omega t} d_j(t)$$

**Absorption cross section**

$$\sigma(\omega) = \frac{4\pi\omega}{c}\Im\left[\alpha(\omega)\right]$$

# Absorption spectra from time-propagation

- Start from the ground state, with a 'kick.'

**Time-dependent potential**

$$V(\boldsymbol{r}, t) = \boldsymbol{\kappa}\delta(t) \quad \Rightarrow \quad \psi \to \psi e^{i\boldsymbol{k}\cdot\boldsymbol{r}}$$

- Time-propagate and get the dipole $\boldsymbol{d}(t)$ as a function of time.

**Polarizability tensor**

$$\alpha_{ij}(\omega) = -\frac{1}{\kappa_i}\int dt\, e^{i\omega t} d_j(t)$$

**Absorption cross section**

$$\sigma(\omega) = \frac{4\pi\omega}{c}\Im\left[\alpha(\omega)\right]$$

# Absorption spectra from time-propagation

- Start from the ground state, with a 'kick.'

## Time-dependent potential

$$V(\boldsymbol{r}, t) = \boldsymbol{\kappa}\delta(t) \qquad \Rightarrow \qquad \psi \to \psi e^{i\boldsymbol{k}\cdot\boldsymbol{r}}$$

- Time-propagate and get the dipole $\boldsymbol{d}(t)$ as a function of time.

## Polarizability tensor

$$\alpha_{ij}(\omega) = -\frac{1}{\kappa_i}\int dt\, e^{i\omega t} d_j(t)$$

## Absorption cross section

$$\sigma(\omega) = \frac{4\pi\omega}{c}\Im\left[\alpha(\omega)\right]$$

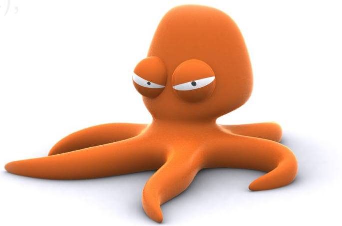
# Octopus[1]

- Fortran 95 and C (+ some Perl utilities).
- Focused on finite systems (periodic systems possible too).
- Norm-conserving pseudopotentials.
- Real-space grid representation.
- Current version is 5.0.1
- DFT with many functionals (from `libxc`), Hartree-Fock, Hartree

[1] http://www.tddft.org/programs/octopus

# Octopus[1]

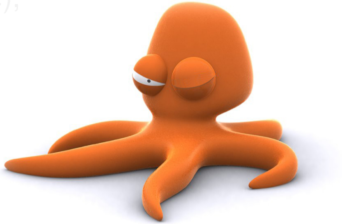
- Fortran 95 and C (+ some Perl utilities).
- Focused on finite systems (periodic systems possible too).
- Norm-conserving pseudopotentials.
- Real-space grid representation.
- Current version is 5.0.1
- DFT with many functionals (from `libxc`), Hartree-Fock, Hartree



[1] http://www.tddft.org/programs/octopus

# Octopus[1]

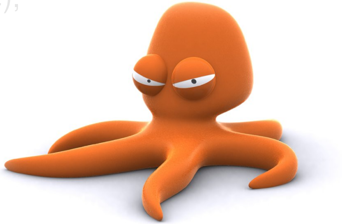
- Fortran 95 and C (+ some Perl utilities).
- Focused on finite systems (periodic systems possible too).
- Norm-conserving pseudopotentials.
- Real-space grid representation.
- Current version is 5.0.1
- DFT with many functionals (from `libxc`), Hartree-Fock, Hartree



[1] `http://www.tddft.org/programs/octopus`

# Octopus[1]

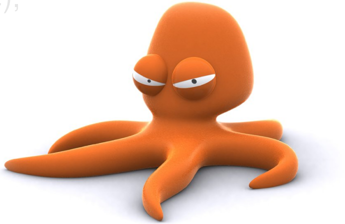
- Fortran 95 and C (+ some Perl utilities).
- Focused on finite systems (periodic systems possible too).
- Norm-conserving pseudopotentials.
- Real-space grid representation.
- Current version is 5.0.1
- DFT with many functionals (from `libxc`), Hartree-Fock, Hartree



[1] `http://www.tddft.org/programs/octopus`

# Octopus[1]

- Fortran 95 and C (+ some Perl utilities).
- Focused on finite systems (periodic systems possible too).
- Norm-conserving pseudopotentials.
- Real-space grid representation.
- Current version is 5.0.1
- DFT with many functionals (from `libxc`), Hartree-Fock, Hartree



[1] `http://www.tddft.org/programs/octopus`

# Octopus[1]

- Fortran 95 and C (+ some Perl utilities).
- Focused on finite systems (periodic systems possible too).
- Norm-conserving pseudopotentials.
- Real-space grid representation.
- Current version is 5.0.1
- DFT with many functionals (from `libxc`), Hartree-Fock, Hartree

[1] `http://www.tddft.org/programs/octopus`

# References


- Xavier Andrade, David A. Strubbe, Umberto De Giovannini, Ask Hjorth Larsen, Micael J. T. Oliveira, Joseba Alberdi-Rodríguez, Alejandro Varas, Iris Theophilou, Nicole Helbig, Matthieu Verstraete, Lorenzo Stella, Fernando Nogueira, Alán Aspuru-Guzik, Alberto Castro, Miguel A. L. Marques, and Ángel Rubio, "Real-space grids and the Octopus code as tools for the development of new simulation approaches for electronic systems," *Phys. Chem. Chem. Phys.* **17**, 31371-31396 (2015).
- A. Castro, H. Appel, Micael Oliveira, C.A. Rozzi, X. Andrade, F. Lorenzen, M.A.L. Marques, E.K.U. Gross, and A. Rubio, "octopus: a tool for the application of time-dependent density functional theory," *Phys. Stat. Sol. B* **243**, 2465-2488 (2006).
- M.A.L. Marques, Alberto Castro, George F. Bertsch, and Angel Rubio, "octopus: a first-principles tool for excited electron-ion dynamics," *Comput. Phys. Commun.* **151**, 60-78 (2003).

# Pulpo a feira (pulpo a la gallega)

The origin of the name Octopus. (Recipe available in code.)

# Octopus[2]

- Ground-state DFT.
- Time-propagation.
- Molecular dynamics (Ehrenfest, Born-Oppenheimer).
- Casida linear response.
- Sternheimer linear response for electromagnetic response, phonons, Van der Waals coefficients.
- Optimal control theory.
- Photoemission spectroscopy.
- (Other experimental features.)

[2] http://www.tddft.org/programs/octopus

# Octopus[2]

- Ground-state DFT.
- Time-propagation.
- Molecular dynamics (Ehrenfest, Born-Oppenheimer).
- Casida linear response.
- Sternheimer linear response for electromagnetic response, phonons, Van der Waals coefficients.
- Optimal control theory.
- Photoemission spectroscopy.
- (Other experimental features.)

[2] http://www.tddft.org/programs/octopus

# Octopus[2]

- Ground-state DFT.
- Time-propagation.
- Molecular dynamics (Ehrenfest, Born-Oppenheimer).
- Casida linear response.
- Sternheimer linear response for electromagnetic response, phonons, Van der Waals coefficients.
- Optimal control theory.
- Photoemission spectroscopy.
- (Other experimental features.)

[2] `http://www.tddft.org/programs/octopus`

# Octopus[2]

- Ground-state DFT.
- Time-propagation.
- Molecular dynamics (Ehrenfest, Born-Oppenheimer).
- Casida linear response.
- Sternheimer linear response for electromagnetic response, phonons, Van der Waals coefficients.
- Optimal control theory.
- Photoemission spectroscopy.
- (Other experimental features.)

[2] http://www.tddft.org/programs/octopus

# Octopus[2]

- Ground-state DFT.
- Time-propagation.
- Molecular dynamics (Ehrenfest, Born-Oppenheimer).
- Casida linear response.
- Sternheimer linear response for electromagnetic response, phonons, Van der Waals coefficients.
- Optimal control theory.
- Photoemission spectroscopy.
- (Other experimental features.)

[2] http://www.tddft.org/programs/octopus

# Octopus[2]

- Ground-state DFT.
- Time-propagation.
- Molecular dynamics (Ehrenfest, Born-Oppenheimer).
- Casida linear response.
- Sternheimer linear response for electromagnetic response, phonons, Van der Waals coefficients.
- Optimal control theory.
- Photoemission spectroscopy.
- (Other experimental features.)

[2] http://www.tddft.org/programs/octopus

# Octopus[2]

- Ground-state DFT.
- Time-propagation.
- Molecular dynamics (Ehrenfest, Born-Oppenheimer).
- Casida linear response.
- Sternheimer linear response for electromagnetic response, phonons, Van der Waals coefficients.
- Optimal control theory.
- Photoemission spectroscopy.
- (Other experimental features.)

[2] http://www.tddft.org/programs/octopus

# Octopus[2]

- Ground-state DFT.
- Time-propagation.
- Molecular dynamics (Ehrenfest, Born-Oppenheimer).
- Casida linear response.
- Sternheimer linear response for electromagnetic response, phonons, Van der Waals coefficients.
- Optimal control theory.
- Photoemission spectroscopy.
- (Other experimental features.)

[2] `http://www.tddft.org/programs/octopus`

# Parallelization

- Parallelization in domains:
  - Each processor handles points in a region of space.
  - Points in the boundaries of each region must be copied to other nodes.
  - Integrals are performed locally and summed over all domains.
  - Efficient and scalable scheme.
- Parallelization in states:
  - Each processor handles a group of states
  - Efficient scheme for time propagation
  - Also applicable for the ground state
- Parallelization in $\mathbf{k}$-points/spin.
- Parallelization in electron-hole pairs (for Casida linear response).
- Combined parallelization.
- Scales to thousands of processors.

# Parallelization

- Parallelization in domains:
  - Each processor handles points in a region of space.
  - Points in the boundaries of each region must be copied to other nodes.
  - Integrals are performed locally and summed over all domains.
  - Efficient and scalable scheme.
- Parallelization in states:
  - Each processor handles a group of states.
  - Efficient scheme for time propagation.
  - Also applicable for the ground state.
- Parallelization in $\mathbf{k}$-points/spin.
- Parallelization in electron-hole pairs (for Casida linear response).
- Combined parallelization.
- Scales to thousands of processors.

# Parallelization

- Parallelization in domains:
  - Each processor handles points in a region of space.
  - Points in the boundaries of each region must be copied to other nodes.
  - Integrals are performed locally and summed over all domains.
  - Efficient and scalable scheme.
- Parallelization in states:
  - Each processor handles a group of states
  - Efficient scheme for time propagation
  - Also applicable for the ground state
- Parallelization in $\mathbf{k}$-points/spin.
- Parallelization in electron-hole pairs (for Casida linear response).
- Combined parallelization.
- Scales to thousands of processors.

# Parallelization

- Parallelization in domains:
  - Each processor handles points in a region of space.
  - Points in the boundaries of each region must be copied to other nodes.
  - Integrals are performed locally and summed over all domains.
  - Efficient and scalable scheme.
- Parallelization in states:
  - Each processor handles a group of states.
  - Efficient scheme for time propagation.
  - Also applicable for the ground state.
- Parallelization in $\mathbf{k}$-points/spin.
- Parallelization in electron-hole pairs (for Casida linear response).
- Combined parallelization.
- Scales to thousands of processors.

- Parallelization in domains:
  - Each processor handles points in a region of space.
  - Points in the boundaries of each region must be copied to other nodes.
  - Integrals are performed locally and summed over all domains.
  - Efficient and scalable scheme.
- Parallelization in states:
  - Each processor handles a group of states.
  - Efficient scheme for time propagation.
  - Also applicable for the ground state.
- Parallelization in $\mathbf{k}$-points/spin.
- Parallelization in electron-hole pairs (for Casida linear response).
- Combined parallelization.
- Scales to thousands of processors.

# Parallelization

- Parallelization in domains:
  - Each processor handles points in a region of space.
  - Points in the boundaries of each region must be copied to other nodes.
  - Integrals are performed locally and summed over all domains.
  - Efficient and scalable scheme.
- Parallelization in states:
  - Each processor handles a group of states.
  - Efficient scheme for time-propagation.
  - Also applicable for the ground state.
- Parallelization in $\mathbf{k}$-points/spin.
- Parallelization in electron-hole pairs (for Casida linear response).
- Combined parallelization.
- Scales to thousands of processors.

# Parallelization

- Parallelization in domains:
  - Each processor handles points in a region of space.
  - Points in the boundaries of each region must be copied to other nodes.
  - Integrals are performed locally and summed over all domains.
  - Efficient and scalable scheme.
- Parallelization in states:
  - Each processor handles a group of states.
  - Efficient scheme for time-propagation.
  - Also applicable for the ground state.
- Parallelization in $\mathbf{k}$-points/spin.
- Parallelization in electron-hole pairs (for Casida linear response).
- Combined parallelization.
- Scales to thousands of processors.

# Parallelization

- Parallelization in domains:
  - Each processor handles points in a region of space.
  - Points in the boundaries of each region must be copied to other nodes.
  - Integrals are performed locally and summed over all domains.
  - Efficient and scalable scheme.
- Parallelization in states:
  - Each processor handles a group of states.
  - Efficient scheme for time-propagation.
  - Also applicable for the ground state.
- Parallelization in $\mathbf{k}$-points/spin.
- Parallelization in electron-hole pairs (for Casida linear response).
- Combined parallelization.
- Scales to thousands of processors.

# Parallelization

- Parallelization in domains:
  - Each processor handles points in a region of space.
  - Points in the boundaries of each region must be copied to other nodes.
  - Integrals are performed locally and summed over all domains.
  - Efficient and scalable scheme.
- Parallelization in states:
  - Each processor handles a group of states.
  - Efficient scheme for time-propagation.
  - Also applicable for the ground state.
- Parallelization in $\mathbf{k}$-points/spin.
- Parallelization in electron-hole pairs (for Casida linear response).
- Combined parallelization.
- Scales to thousands of processors.

# Parallelization

- Parallelization in domains:
  - Each processor handles points in a region of space.
  - Points in the boundaries of each region must be copied to other nodes.
  - Integrals are performed locally and summed over all domains.
  - Efficient and scalable scheme.
- Parallelization in states:
  - Each processor handles a group of states.
  - Efficient scheme for time-propagation.
  - Also applicable for the ground state.
- Parallelization in $\mathbf{k}$-points/spin.
- Parallelization in electron-hole pairs (for Casida linear response).
- Combined parallelization.
- Scales to thousands of processors.

# Parallelization

- Parallelization in domains:
  - Each processor handles points in a region of space.
  - Points in the boundaries of each region must be copied to other nodes.
  - Integrals are performed locally and summed over all domains.
  - Efficient and scalable scheme.
- Parallelization in states:
  - Each processor handles a group of states.
  - Efficient scheme for time-propagation.
  - Also applicable for the ground state.
- Parallelization in $\mathbf{k}$-points/spin.
- Parallelization in electron-hole pairs (for Casida linear response).
- Combined parallelization.
- Scales to thousands of processors.

# Parallelization

- Parallelization in domains:
  - Each processor handles points in a region of space.
  - Points in the boundaries of each region must be copied to other nodes.
  - Integrals are performed locally and summed over all domains.
  - Efficient and scalable scheme.
- Parallelization in states:
  - Each processor handles a group of states.
  - Efficient scheme for time-propagation.
  - Also applicable for the ground state.
- Parallelization in $\mathbf{k}$-points/spin.
- Parallelization in electron-hole pairs (for Casida linear response).
- Combined parallelization.
- Scales to thousands of processors.

# Parallelization

- Parallelization in domains:
  - Each processor handles points in a region of space.
  - Points in the boundaries of each region must be copied to other nodes.
  - Integrals are performed locally and summed over all domains.
  - Efficient and scalable scheme.
- Parallelization in states:
  - Each processor handles a group of states.
  - Efficient scheme for time-propagation.
  - Also applicable for the ground state.
- Parallelization in $\mathbf{k}$-points/spin.
- Parallelization in electron-hole pairs (for Casida linear response).
- Combined parallelization.
- Scales to thousands of processors.

# License

- Octopus is free open-source software (GNU Public License v2).
  - Free to use it.
  - Study the code and modify it.
  - Contribute back your changes.
- New developers are welcome.

# License

- Octopus is free open-source software (GNU Public License v2).
  - Free to use it.
  - Study the code and modify it.
  - Contribute back your changes.
- New developers are welcome.

# License

- Octopus is free open-source software (GNU Public License v2).
  - Free to use it.
  - Study the code and modify it.
  - Contribute back your changes.
- New developers are welcome.

# License

- Octopus is free open-source software (GNU Public License v2).
  - Free to use it.
  - Study the code and modify it.
  - Contribute back your changes.
- New developers are welcome.

# License

- Octopus is free open-source software (GNU Public License v2).
  - Free to use it.
  - Study the code and modify it.
  - Contribute back your changes.
- New developers are welcome.