Introduction to the Octopus code

Xavier Andrade
and the Octopus development team

European Theoretical Spectroscopy Facility
and
Departamento de Física de Materiales
Universidad del País Vasco, Spain

Benasque, September 2008
Octopuses are *cephalopods* from the *octopoda* order.

They live in the sea (also in trees).

From few cm to 20 m.

Very smart animals.

Well equipped:
- Poison
- Ink
- Camouflage.
- Release their tentacles.

Very effective hunters:
they eat crabs, fishes, mussels, siestas, etc.
Octopus

- Fortran 95 and C.
- Focused on finite systems (periodic systems not mature yet).
- Norm-conserving pseudo-potentials.
- Real space grid representation.
- Features:
  - Ground state DFT.
  - TDDFT
  - Time propagation
  - Casida LR-TDDFT
  - Sternheimer linear response
  - Optimal control theory
  - Real-time quantum transport

http://www.tddft.org/programs/octopus
Fortran 95 and C.

Focused on finite systems (periodic systems not mature yet).

- Norm-conserving pseudo-potentials.
- Real space grid representation.
- Features:
  - Ground state DFT
  - TDDFT
  - Time propagation
  - Casida LR-TDDFT
  - Sternheimer linear response
  - Optimal control theory
  - Real-time quantum transport

1http://www.tddft.org/programs/octopus
Octopus

- Fortran 95 and C.
- Focused on finite systems (periodic systems not mature yet).
- Norm-conserving pseudo-potentials.
- Real space grid representation.
- Features:
  - Ground state DFT
  - TDDFT
  - Time propagation
  - Casida DR TDDFT
  - Sternheimer linear response
  - Optimal control theory
  - Real-time quantum transport

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

- Fortran 95 and C.
- Focused on finite systems (periodic systems not mature yet).
- Norm-conserving pseudo-potentials.
- Real space grid representation.

Features:
- Ground state DFT.
- TDDFT
- Time propagation
- Casida LR TDDFT
- Sternheimer linear response
- Optimal control theory
- Real-time quantum transport

1http://www.tddft.org/programs/octopus
Octopus

- Fortran 95 and C.
- Focused on finite systems (periodic systems not mature yet).
- Norm-conserving pseudo-potentials.
- Real space grid representation.
- **Features:**
  - Ground state DFT.
  - TDDFT
  - Time propagation:
    - Linear response and strong fields.
    - Ehrenfest molecular dynamics.
  - Casida LR-TDDFT.
  - Sternheimer linear response.
  - Optimal control theory.
  - Real-time quantum transport.

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

- Fortran 95 and C.
- Focused on finite systems (periodic systems not mature yet).
- Norm-conserving pseudo-potentials.
- Real space grid representation.
- Features:
  - Ground state DFT.
  - TDDFT
  - Time propagation:
    - Linear response and strong fields
    - Ehrenfest molecular dynamics
  - Casida LR-TDDFT.
  - Sternheimer linear response.
  - Optimal control theory.
  - Real-time quantum transport.

\[\text{http://www.tddft.org/programs/octopus}\]
Octopus

- Fortran 95 and C.
- Focused on finite systems (periodic systems not mature yet).
- Norm-conserving pseudo-potentials.
- Real space grid representation.
- Features:
  - Ground state DFT.
  - TDDFT
  - Time propagation:
    - Linear response and strong fields.
    - Ehrenfest molecular dynamics
  - Casida LR-TDDFT.
  - Sternheimer linear response.
  - Optimal control theory.
  - Real-time quantum transport.

1http://www.tddft.org/programs/octopus
Octopus

- Fortran 95 and C.
- Focused on finite systems (periodic systems not mature yet).
- Norm-conserving pseudo-potentials.
- Real space grid representation.
- Features:
  - Ground state DFT.
  - TDDFT
  - Time propagation:
    - Linear response and strong fields.
    - Ehrenfest molecular dynamics.
  - Casida LR-TDDFT.
  - Sternheimer linear response.
  - Optimal control theory.
  - Real-time quantum transport.

1http://www.tddft.org/programs/octopus
Octopus

- Fortran 95 and C.
- Focused on finite systems (periodic systems not mature yet).
- Norm-conserving pseudo-potentials.
- Real space grid representation.
- Features:
  - Ground state DFT.
  - TDDFT
  - Time propagation:
    - Linear response and strong fields.
    - Ehrenfest molecular dynamics.
  - Casida LR-TDDFT.
  - Sternheimer linear response.
  - Optimal control theory.
  - Real-time quantum transport.

1http://www.tddft.org/programs/octopus
Octopus

- Fortran 95 and C.
- Focused on finite systems (periodic systems not mature yet).
- Norm-conserving pseudo-potentials.
- Real space grid representation.
- Features:
  - Ground state DFT.
  - TDDFT
  - Time propagation:
    - Linear response and strong fields.
    - Ehrenfest molecular dynamics.
  - Casida LR-TDDFT.
  - Sternheimer linear response.
  - Optimal control theory.
  - Real-time quantum transport.

\[\text{http://www.tddft.org/programs/octopus}\]
Octopus

- Fortran 95 and C.
- Focused on finite systems (periodic systems not mature yet).
- Norm-conserving pseudo-potentials.
- Real space grid representation.
- Features:
  - Ground state DFT.
  - TDDFT
  - Time propagation:
    - Linear response and strong fields.
    - Ehrenfest molecular dynamics.
  - Casida LR-TDDFT.
  - Sternheimer linear response.
  - Optimal control theory.
  - Real-time quantum transport.

1http://www.tddft.org/programs/octopus
Octopus

- Fortran 95 and C.
- Focused on finite systems (periodic systems not mature yet).
- Norm-conserving pseudo-potentials.
- Real space grid representation.
- Features:
  - Ground state DFT.
  - TDDFT
  - Time propagation:
    - Linear response and strong fields.
    - Ehrenfest molecular dynamics.
  - Casida LR-TDDFT.
  - Sternheimer linear response.
  - Optimal control theory.
  - Real-time quantum transport.

Octopus

- Fortran 95 and C.
- Focused on finite systems (periodic systems not mature yet).
- Norm-conserving pseudo-potentials.
- Real space grid representation.
- Features:
  - Ground state DFT.
  - TDDFT
  - Time propagation:
    - Linear response and strong fields.
    - Ehrenfest molecular dynamics.
  - Casida LR-TDDFT.
  - Sternheimer linear response.
  - Optimal control theory.
  - Real-time quantum transport.

1http://www.tddft.org/programs/octopus
Octopus

- Fortran 95 and C.
- Focused on finite systems (periodic systems not mature yet).
- Norm-conserving pseudo-potentials.
- Real space grid representation.
- Features:
  - Ground state DFT.
  - TDDFT
  - Time propagation:
    - Linear response and strong fields.
    - Ehrenfest molecular dynamics.
  - Casida LR-TDDFT.
  - Sternheimer linear response.
  - Optimal control theory.
  - Real-time quantum transport.

1http://www.tddft.org/programs/octopus
Pseudo-potentials

- The atomic potential is very strong.
- Lots of core electrons.
- Core electrons are independent of the environment.
- Replace the potential and core electrons by a pseudo-potential.

\[ V = V_{\text{loc}} + \sum_{lm} |\psi_{lm}\rangle (V_l - V_{\text{loc}}) \langle \psi_{lm}| \]
Pseudo-potentials

- The atomic potential is very strong.
- **Lots of core electrons.**
- Core electrons are independent of the environment.
- Replace the potential and core electrons by a pseudo-potential.

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

- The atomic potential is very strong.
- Lots of core electrons.
- Core electrons are independent of the environment.
- Replace the potential and core electrons by a pseudo-potential.

**Norm-conserving pseudo-potentials**

\[ V = V_{\text{loc}} + \sum_{lm} |lm\rangle (V_l - V_{\text{loc}}) \langle lm| \]
Pseudo-potentials

- The atomic potential is very strong.
- Lots of core electrons.
- Core electrons are independent of the environment.
- Replace the potential and core electrons by a pseudo-potential.

\[
V = V_{loc} + \sum_{lm} |lm\rangle (V_l - V_{loc}) \langle lm| 
\]

Norm-conserving pseudo-potentials
Pseudo-potentials

- The atomic potential is very strong.
- Lots of core electrons.
- Core electrons are independent of the environment.
- Replace the potential and core electrons by a pseudo-potential.

Norm-conserving pseudo-potentials

\[ V = V_{loc} + \sum_{lm} |lm\rangle (V_l - V_{loc}) \langle lm| \]
Partial Differential Equation: infinite degrees of freedom.
- Reduce it to a finite number.
- Functions are represented by its value over a set of points.
- Point distribution:
  - Uniform space grid.
  - Distance between points is constant: Spacing.
  - Non-uniform grids.
- Finite region of the space: Box
Real space grid

- Partial Differential Equation: infinite degrees of freedom.
- Reduce it to a finite number.
- Functions are represented by its value over a set of points.
- Point distribution:
  - Uniform space grid.
  - Distance between points is constant: Spacing.
  - Non-uniform grids.
- Finite region of the space: Box
Real space grid

- Partial Differential Equation: infinite degrees of freedom.
- Reduce it to a finite number.
- Functions are represented by its value over a set of points.

- Point distribution:
  - Uniform space grid.
  - Distance between points is constant: Spacing.
  - Non-uniform grids.

- Finite region of the space: Box
Partial Differential Equation: infinite degrees of freedom.
Reduce it to a finite number.
Functions are represented by its value over a set of points.

Point distribution:
- Uniform space grid.
- Distance between points is constant: Spacing.
- Non-uniform grids.

Finite region of the space: Box
Real space grid

- Partial Differential Equation: infinite degrees of freedom.
- Reduce it to a finite number.
- Functions are represented by its value over a set of points.
- Point distribution:
  - Uniform space grid.
  - Distance between points is constant: *Spacing*.
  - Non-uniform grids.
- Finite region of the space: *Box*
Real space grid

- Partial Differential Equation: infinite degrees of freedom.
- Reduce it to a finite number.
- Functions are represented by its value over a set of points.
- Point distribution:
  - Uniform space grid.
  - Distance between points is constant: \textit{Spacing}.
  - Non-uniform grids.
- Finite region of the space: \textit{Box}
Real space grid

- Partial Differential Equation: infinite degrees of freedom.
- Reduce it to a finite number.
- Functions are represented by its value over a set of points.
- Point distribution:
  - Uniform space grid.
  - Distance between points is constant: *Spacing*.
  - Non-uniform grids.

- Finite region of the space: *Box*
Real space grid

- Partial Differential Equation: infinite degrees of freedom.
- Reduce it to a finite number.
- Functions are represented by its value over a set of points.
- Point distribution:
  - Uniform space grid.
  - Distance between points is constant: *Spacing*.
  - Non-uniform grids.

- Finite region of the space: *Box*
Boundary conditions

- For finite systems functions go to zero.
- Impose functions to be zero over 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.
- Impose functions to be zero over 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.
- Impose functions to be zero over 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.
- Impose functions to be zero over 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.
- General box shape:
  - Minimum box: a sphere around each atom.
  - Sphere.
  - Cylinder.
  - Parallelepiped.
  - Arbitrary.
Boundary conditions

- Optimize the shape of the box to minimize the number of points.
- General box shape:
  - Minimum box: a sphere around each atom.
  - Sphere.
  - Cylinder.
  - Parallelepiped.
  - Arbitrary.
Boundary conditions

- Optimize the shape of the box to minimize the number of points.
- General box shape:
  - Minimum box: a sphere around each atom.
  - Sphere.
  - Cylinder.
  - Parallelepiped.
  - Arbitrary.
Boundary conditions

- Optimize the shape of the box to minimize the number of points.
- General box shape:
  - Minimum box: a sphere around each atom.
  - Sphere.
  - Cylinder.
  - Parallelepiped.
  - Arbitrary.
Optimize the shape of the box to minimize the number of points.

General box shape:
- Minimum box: a sphere around each atom.
- Sphere.
- Cylinder.
- Parallelepiped.
- Arbitrary.
Boundary conditions

- Optimize the shape of the box to minimize the number of points.
- General box shape:
  - Minimum box: a sphere around each atom.
  - Sphere.
  - Cylinder.
  - Parallelepiped.
  - Arbitrary.
Optimize the shape of the box to minimize the number of points.

General box shape:
- Minimum box: a sphere around each atom.
- Sphere.
- Cylinder.
- Parallelepiped.
- Arbitrary.
Real space grid characteristics

- Natural boundary conditions for different problems.
- Systematically improve discretisation quality:
  - Decrease the spacing.
  - Increase the box size.
- Orthogonal “basis set”.
- Independent of atomic positions (no Pulay forces).
- Problems:
  - Breaking of translational invariance: egg-box effect.
  - Breaking of rotational invariance.
Real space grid characteristics

- Natural boundary conditions for different problems.
- **Systematically improve discretisation quality:**
  - Decrease the spacing.
  - Increase the box size.
- Orthogonal “basis set”.
- Independent of atomic positions (no Pulay forces).
- Problems:
  - Breaking of translational invariance: egg-box effect.
  - Breaking of rotational invariance.
Real space grid characteristics

- Natural boundary conditions for different problems.
- Systematically improve discretisation quality:
  - Decrease the spacing.
  - Increase the box size.
- Orthogonal “basis set”.
- Independent of atomic positions (no Pulay forces).
- Problems:
  - Breaking of translational invariance: egg-box effect.
  - Breaking of rotational invariance.
Real space grid characteristics

- Natural boundary conditions for different problems.
- Systematically improve discretisation quality:
  - Decrease the spacing.
  - Increase the box size.
- Orthogonal “basis set”.
- Independent of atomic positions (no Pulay forces).
- Problems:
  - Breaking of translational invariance: egg-box effect.
  - Breaking of rotational invariance.
Real space grid characteristics

- Natural boundary conditions for different problems.
- Systematically improve discretisation quality:
  - Decrease the spacing.
  - Increase the box size.
- Orthogonal "basis set".
  - Independent of atomic positions (no Pulay forces).
- Problems:
  - Breaking of translational invariance: egg-box effect.
  - Breaking of rotational invariance.
Real space grid characteristics

- Natural boundary conditions for different problems.
- Systematically improve discretisation quality:
  - Decrease the spacing.
  - Increase the box size.
- Orthogonal “basis set”.
- Independent of atomic positions (no Pulay forces).

Problems:
- Breaking of translational invariance: egg-box effect.
- Breaking of rotational invariance.
Real space grid characteristics

- Natural boundary conditions for different problems.
- Systematically improve discretisation quality:
  - Decrease the spacing.
  - Increase the box size.
- Orthogonal “basis set”.
- Independent of atomic positions (no Pulay forces).
- Problems:
  - Breaking of translational invariance: egg-box effect.
  - Breaking of rotational invariance.
Real space grid characteristics

- Natural boundary conditions for different problems.
- Systematically improve discretisation quality:
  - Decrease the spacing.
  - Increase the box size.
- Orthogonal “basis set”.
- Independent of atomic positions (no Pulay forces).
- Problems:
  - Breaking of translational invariance: egg-box effect.
  - Breaking of rotational invariance.
Real space grid characteristics

- Natural boundary conditions for different problems.
- Systematically improve discretisation quality:
  - Decrease the spacing.
  - Increase the box size.
- Orthogonal “basis set”.
- Independent of atomic positions (no Pulay forces).
- Problems:
  - Breaking of translational invariance: egg-box effect.
  - Breaking of rotational invariance.
Differential operations

Finite difference approach

\[ \nabla^2 f(n_x h, n_y h) = \sum_{i}^{n} \sum_{j}^{n} \frac{c_{ij}}{h} f(n_x h + ih, n_y h + jh) \]

- Derivative in a point: sum over neighbour points.
- \(c_{ij}\) depend on the points used: the stencil.
- More points \(\rightarrow\) more precision.
- Semi-local operation.
Differential operations

Finite difference approach

\[ \nabla^2 f(n_x h, n_y h) = \sum_{i}^{n} \sum_{j}^{n} \frac{c_{ij}}{h} f(n_x h + ih, n_y h + jh) \]

- Derivative in a point: sum over neighbour points.
- \( c_{ij} \) depend on the points used: the stencil.
- More points \( \rightarrow \) more precision.
- Semi-local operation.
Differential operations

Finite difference approach

\[ \nabla^2 f(n_x h, n_y h) = \sum_{i}^{n} \sum_{j}^{n} \frac{c_{ij}}{h} f(n_x h + ih, n_y h + jh) \]

- Derivative in a point: sum over neighbour points.
- \( c_{ij} \) depend on the points used: the stencil.
- More points \( \rightarrow \) more precision.
- Semi-local operation.

X. Andrade (EHU/UPV)
Differential operations

Finite difference approach

\[ \nabla^2 f(n_x h, n_y h) = \sum_{i} \sum_{j} \frac{c_{ij}}{h} f(n_x h + ih, n_y h + jh) \]

- Derivative in a point: sum over neighbour points.
- \( c_{ij} \) depend on the points used: *the stencil*.
- More points \( \rightarrow \) more precision.
- Semi-local operation.
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.
Ground state calculations

- What we want to solve:
  Kohn-Sham equations

\[-\nabla^2 \phi_k + V_{eff}[\rho](r)\phi_k = \epsilon_k \phi_k\]

- We use a self-consistency scheme to treat non-linearity.
What we want to solve:

Kohn-Sham equations

\[-\nabla^2 \phi_k + V_{\text{eff}} [\rho] (r) \phi_k = \epsilon_k \phi_k\]

We use a self-consistency scheme to treat non-linearity.
What we want to solve:

Kohn-Sham equations

\[-\nabla^2 \phi_k + V_{eff}[\rho](r)\phi_k = \epsilon_k \phi_k\]

We use a self-consistency scheme to treat non-linearity.
Discretisation of the Hamiltonian

- For the laplacian we use finite differences
  - High order schemes are needed.
- The local part of the potential is direct.
- The non-local potential is applied in small spherical grid around the atoms.
- The Hamiltonian becomes a finite size matrix.
Discretisation of the Hamiltonian

- For the laplacian we use finite differences
  - High order schemes are needed.
- The local part of the potential is direct.
- The non-local potential is applied in small spherical grid around the atoms.
- The Hamiltonian becomes a finite size matrix.
For the laplacian we use finite differences
- High order schemes are needed.

- The local part of the potential is direct.
- The non-local potential is applied in small spherical grid around the atoms.
- The Hamiltonian becomes a finite size matrix.
Discretisation of the Hamiltonian

- For the laplacian we use finite differences
  - High order schemes are needed.

- The local part of the potential is direct.

- The non-local potential is applied in small spherical grid around the atoms.

- The Hamiltonian becomes a finite size matrix.
Discretisation of the Hamiltonian

- For the laplacian we use finite differences
  - High order schemes are needed.
- The local part of the potential is direct.
- The non-local potential is applied in 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).
- Iterative solvers where only the action of the matrix is required.
The eigenproblem

- Find the eigenvectors and eigenvalues of a matrix.
- Very large matrix with lots of zero components (Sparse).
- Iterative solvers where only the action of the matrix is required.
The eigenproblem

- Find the eigenvectors and eigenvalues of a matrix.
- Very large matrix with lots of zero components (*Sparse*).
- Iterative solvers where only the action of the matrix is required.
The eigensolver

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

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

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

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

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

**Rayleigh-Ritz quotient**

- Works for the first state.
- For higher energy states it is necessary to orthogonalize against the lower ones.
Main feature of Octopus.

Given an initial condition, solve the:

\[
i \frac{\partial \phi_k}{\partial t} = -\nabla^2 \phi_k + V_{\text{eff}}[\rho](r,t) \phi_k
\]

Many properties can be obtained.

Response to time dependent fields: lasers.
Time propagation

- Main feature of Octopus.
- Given an initial condition, solve the:

\[ i \frac{\partial \phi_k}{\partial t} = -\nabla^2 \phi_k + V_{\text{eff}}[\rho](r, t) \phi_k \]

- Many properties can be obtained.
- Response to time dependent fields: lasers.
Time propagation

- Main feature of Octopus.
- Given an initial condition, solve the:

\[ i \frac{\partial \phi_k}{\partial t} = -\nabla^2 \phi_k + V_{eff}[\rho](r, t)\phi_k \]

- Many properties can be obtained.
- Response to time dependent fields: lasers.
Main feature of Octopus.

Given an initial condition, solve the:

\[ i \frac{\partial \phi_k}{\partial t} = -\nabla^2 \phi_k + V_{\text{eff}}[\rho](r, t) \phi_k \]

Many properties can be obtained.

Response to time dependent fields: lasers.
Main feature of Octopus.

- Given an initial condition, solve the:

\[
i \frac{\partial \phi_k}{\partial t} = -\nabla^2 \phi_k + V_{\text{eff}} [\rho] (r, t) \phi_k
\]

- Many properties can be obtained.

  - Response to time dependent fields: lasers.
Absorption spectra from time propagation

- Start from the ground state.

Time dependent potential

\[ V(\mathbf{r}, t) = \kappa \delta(t) \]

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

Polarizability

\[ \alpha(\omega) = -\frac{1}{\kappa} \int dt \, e^{i\omega t} d(t) \]

Absorption cross section

\[ \sigma(\omega) = 4\pi \omega c \Im \left[ \alpha(\omega) \right] \]
Absorption spectra from time propagation

- Start from the ground state.

**Time dependent potential**

\[ V(r, t) = \kappa \delta(t) \]

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

**Polarizability**

\[ \alpha(\omega) = -\frac{1}{\kappa} \int dt \, e^{i\omega t} d(t) \]

**Absorption cross section**

\[ \sigma(\omega) = 4\pi \omega c \text{Im} \left[ \alpha(\omega) \right] \]
Absorption spectra from time propagation

- Start from the ground state.

Time dependent potential

\[ V(\mathbf{r}, t) = \kappa \delta(t) \]

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

Polarizability

\[ \alpha(\omega) = -\frac{1}{\kappa} \int dt \, e^{i\omega t} d(t) \]

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.

**Time dependent potential**

\[ V(r, t) = \kappa \delta(t) \]

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

**Polarizability**

\[ \alpha(\omega) = -\frac{1}{\kappa} \int dt \ e^{i\omega t} d(t) \]

**Absorption cross section**

\[ \sigma(\omega) = \frac{4\pi\omega}{c} \Im [\alpha(\omega)] \]
Absorption spectra from time propagation

- Start from the ground state.

Time dependent potential

\[ V(r, t) = \kappa \delta(t) \]

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

Polarizability

\[ \alpha(\omega) = -\frac{1}{\kappa} \int dt \, e^{i\omega t} d(t) \]

Absorption cross section

\[ \sigma(\omega) = \frac{4\pi \omega}{c} \Im [\alpha(\omega)] \]
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.
  - Work in progress for the ground state.

- **Combined parallelization.**
- Scales to hundreds of processors (development version).
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.
  - Work in progress for the ground state.

- Combined parallelization.

- Scales to hundreds of processors (development version).
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.
  - Work in progress for the ground state.

- Combined parallelization.
- Scales to hundreds of processors (development version).
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.
  - Work in progress for the ground state.

- Combined parallelization.
- Scales to hundreds of processors (development version).
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.
- Work in progress for the ground state.

Combined parallelization.
- Scales to hundreds of processors (development version).
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.
  - Work in progress for the ground state.

- Combined parallelization.

- Scales to hundreds of processors (development version).
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.
  - Work in progress for the ground state.

- Combined parallelization.
- Scales to hundreds of processors (development version).
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.
  - Work in progress for the ground state.

- Combined parallelization.

- Scales to hundreds of processors (development version).
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.
  - Work in progress for the ground state.

- Combined parallelization.
- Scales to hundreds of processors (development version).
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.
  - Work in progress for the ground state.

- **Combined parallelization.**
  - Scales to hundreds of processors (development version).
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.
  - Work in progress for the ground state.

- Combined parallelization.
  - Scales to hundreds of processors (development version).
License

- Octopus is free software (GPL license).
  - Free to use it.
  - Study the code and modify it.
  - Contribute back your changes.
- New developers are welcome.
License

- Octopus is free software (GPL license).
  - Free to use it.
  - Study the code and modify it.
  - Contribute back your changes.
- New developers are welcome.
License

- Octopus is free software (GPL license).
  - Free to use it.
  - Study the code and modify it.
  - Contribute back your changes.

- New developers are welcome.
License

Octopus is free software (GPL license).
- Free to use it.
- Study the code and modify it.
- Contribute back your changes.

New developers are welcome.
License

- Octopus is free software (GPL license).
  - Free to use it.
  - Study the code and modify it.
  - Contribute back your changes.

- New developers are welcome.
Octopus developers

- Miguel Marques, Université Lyon I.
- Micael Oliveira and Fernando Nogueira, Universidade de Coimbra.
- David Strubbe, UC Berkeley.
- Carlo Andrea Rozzi, Università di Modena e Reggio Emilia.
- Xavier Andrade and Angel Rubio, Universidad del Pais Vasco, San Sebastian.