oepdev

0.0.0

Generated by Doxygen 1.8.6

Fri Jan 19 2018 14:42:26

Contents

Chapter 1

Main Page

oep-dev

Generalized One-Electron Potentials: Development Platform.

Contact: Bartosz Błasiak (blasiak.bartosz@gmail.com)

Overview

Test various models of the intermolecular interaction that is based on the application of the **One-Electron Potentials (OEP's)** technique.

Currently, the interaction between two molecules described by the Hartree-Fock-Roothaan-Hall theory or the configuration interaction with singles theory is considered. In particular, the plugin tests the models of:

1. the Pauli exchange-repulsion interaction energy (Project II)
2. the Induction interaction energy (Project III)
3. the excitation energy transfer couplings (Project I)

against reference solutions (exact or other approximations).

Places to go:

- https://github.com/globulion/oepdev/blob/master/doc/git/doc_oep_design.md "OEP Design"
- https://github.com/globulion/oepdev/blob/master/doc/git/doc_implemented_models.md "Implemented Models"
- https://github.com/globulion/oepdev/blob/master/doc/git/doc_programming_etiquette.md "Programming Etiquette"
- Current Issues

References

Chapter 2

License

Chapter 3

OEP Design.

OEP (One-Electron Potential) is associated with certain quantum one-electron operator $\hat{v}^A$ that defines the ability of molecule A to interact in a particular way with other molecules.

Technically, OEP can be understood as a **container object** (associated with the molecule in question) that stores the information about the above mentioned quantum operator. Here, it is assumed that similar OEP object is also defined for all other molecules in a molecular aggregate.

In case of interaction between molecules A and B, OEP object of molecule A interacts directly with wavefunction object of the molecule B. Defining a Solver class that handles such interaction Wavefunction class and OEP class

the universal design of OEP-based approaches can be established and developed.

> **Important:** OEP and Wavefunction classes should not be restricted to Hartree-Fock; in generall any correlated wavefunction and derived OEP's should be allowed to work with each other.

3.1 OEP Classes

There are many types of OEP's, but the underlying principle is the same and independent of the type of intermolecular interaction. Therefore, the OEP's should be implemented by using a multi-level class design. In turn, this design depends on the way OEP's enter the mathematical expressions, i.e., on the types of matrix elements of the one-electron effective operator $\hat{v}^A$.

3.1.1 Structure of possible OEP-based expressions and their unification

Structure of OEP-based mathematical expressions is listed below:

| Type | Matrix Element | Comment | |-------—|-—|—|

In the above table, I, J and K indices correspond to basis functions or molecular orbitals. Basis functions can be primary or auxiliary OEP-specialized density-fitting. Depending on the type of function and matrix element, there are many subtypes of resulting matrix elements that differ in their dimensionality. Examples are given below:

| Matrix Element | DF-based form | ESP-based form | |-—|—|—|

In the formulae above, the OEP-part (stored by OEP instances) is shown in blue whereas the Solver-part (to be computed by the Solver) is shown in brown. It is apparent that all OEP-parts have the form of 2nd- or 3rd-rank tensors with different class of axes (molecular orbitals, primary/auxiliary basis, atomic space). Therefore, they can be uniquely defined by a unified *tensor object* (storing double precision numbers) and unified *dimension object* storing the information of the axes classes.

In Psi4, a perfect candidate for the above is `psi4::Tensor` class declared in `psi4/libthce/thce.h`. Except from the numeric content its instances also store the information of the dimensions in a form of a vector of `psi4::Dimension` instances.

Another possibility is to use `psi::Matrix` objects, instead of `psi4::Tensor` objects, possibly putting them into a `std::vector` container in case there is more than two axes.

Chapter 4

List of One-Electron Potentals

Here I provide the list of OEP's that have been already derived within the scope of the OEPDev project.

4.1 Electrostatic Energy OEP's

For electrostatic energy calculations, OEP is simply the electrostatic potential due to nuclei and electrons.

3D form:

$$v(\mathbf{r}) = \sum_x \frac{Z_x}{|\mathbf{r}-\mathbf{r}_x|} + \sum_{\mu\nu\in A} P_{\nu\mu} \int d\mathbf{r}' \frac{\phi_\mu^*(\mathbf{r}')\phi_\nu(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}$$

Matrix form:

$$v_{ik} = \sum_{x\in A} Z_x V_{ik}^{(x)} + \sum_{\mu\nu\in A} P_{\nu\mu}\,(\mu\nu|ik)$$

4.2 Pauli Repulsion OEP's

The following potentials are derived for the evaluation of the Pauli repulsion energy based on Murrel's expressions.

4.2.1 First-order contribution in overlap matrix expansion.

This contribution is simply the electrostatic potential coming from all nuclei and electron density except$*$ from electron density from molecular orbital i that interacts with the generalized overlap density between i of molecule A and j of molecule B.

3D forms:

$$v(\mathbf{r})^{A[i]}_{S^{-1}} = -\sum_{x\in A} \frac{Z_x}{|\mathbf{r}-\mathbf{r}_x|} + \sum_{\mu\nu\in A} \left\{D_{\nu\mu} - C^*_{\mu i}C_{\nu i}\right\} \int d\mathbf{r}' \frac{\phi_\mu^*(\mathbf{r}')\phi_\nu(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}$$

Matrix forms:

$$v_{\xi i}(S^{-1}) = \sum_{\kappa \in A} C_{i\kappa} \left\{ -\sum_{x \in A} V^{(x)}_{\kappa\xi} + \sum_{\mu\nu \in A} \left\{ D_{\nu\mu} - C^*_{\mu i} C_{\nu i} \right\} (\mu\nu|\xi\kappa) \right\}$$

4.2.2 Second-order contribution in overlap matrix expansion.

To be added here!

4.3 Excitonic Energy Transfer OEP's

The following potentials are derived for the evaluation of the short-range EET couplings based on Fujimoto's TDFI-TI method.

4.3.1 ET contributions.

3D forms:

$$v(\mathbf{r})^{A[\mu]}_1 = -C^*_{\mu L} \sum_{x \in A} \frac{Z_x}{|\mathbf{r} - \mathbf{r}_x|} + \sum_{\nu\kappa \in A} \left\{ C^*_{\mu L} D_{\nu\kappa} - \frac{1}{2} C^*_{\nu L} D_{\mu\kappa} \right\} \int d\mathbf{r}' \frac{\phi^*_\nu(\mathbf{r}')\phi_\kappa(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|}$$
$$v(\mathbf{r})^{A[\mu]}_2 = C_{\kappa H} \sum_{\nu\kappa \in A} \left\{ 2C^*_{\nu L} C^*_{\mu H} - C^*_{\nu H} C^*_{\mu L} \right\} \int d\mathbf{r}' \frac{\phi^*_\nu(\mathbf{r}')\phi_\kappa(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|}$$
$$v(\mathbf{r})^{A[\mu]}_3 = v(\mathbf{r})^{A[\mu]}_1 + v(\mathbf{r})^{A[\mu]}_1$$

Matrix forms:

$$v_{\mu\xi}(1) = -C^*_{\mu L} \sum_{x \in A} V^x_{\mu\xi} + \sum_{\nu\kappa \in A} \left\{ C^*_{\mu L} D_{\nu\kappa} - \frac{1}{2} C^*_{\nu L} D_{\mu\kappa} \right\} (\nu\kappa|\mu\xi)$$
$$v_{\mu\xi}(2) = C_{\kappa H} \sum_{\nu\kappa \in A} \left\{ 2C^*_{\nu L} C^*_{\mu H} - C^*_{\nu H} C^*_{\mu L} \right\} (\nu\kappa|\mu\xi)$$
$$v_{\mu\xi}(3) = v_{\mu\xi}(1) + v_{\mu\xi}(2)$$

4.3.2 HT contributions.

Do be derived.

4.3.3 CT contributions.

To be derived.

4.4 Full HF Interaction OEP's

The following potentials are derived for the evaluation of the full Hartree-Fock interaction energy based on the OEPDev equations.

Chapter 5

Namespace Index

5.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 6

Hierarchical Index

6.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 7

Class Index

7.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 8

Namespace Documentation

8.1 oepdev Namespace Reference

oepdev module namespace.

Classes

- class OEPotential

  Generalized One-Electron Potential: Abstract base.
- class ElectrostaticEnergyOEPotential

  Generalized One-Electron Potential for Electrostatic Energy calculations.
- class RepulsionEnergyOEPotential

  Generalized One-Electron Potential for Pauli repulsion energy calculations.
- class EETCouplingOEPotential

  Generalized One-Electron Potential for EET coupling calculations.
- class PotentialInt

  Computes potential integrals.
- class CPHF

  CPHF solver class.
- class DIISManager

  DIIS manager.
- class ESPSolver

  Charges from Electrostatic Potential (ESP). A solver-type class.
- class AllAOShellCombinationsIterator

  Loop over all possible ERI shells.
- class AllAOIntegralsIterator

  Loop over all possible ERI within a particular shell.
- class Points3DIterator

  Iterator over a collection of points in 3D space. Abstract base.
- class CubePoints3DIterator

  Iterator over a collection of points in 3D space. g09 Cube-like order.
- class RandomPoints3DIterator

  Iterator over a collection of points in 3D space. Random collection.
- class PointsCollection3D

  Collection of points in 3D space. Abstract base.
- class RandomPointsCollection3D

  Collection of random points in 3D space.

- class CubePointsCollection3D

  G09 cube-like ordered collection of points in 3D space.
- class ScalarField3D

  Scalar field in 3D space. Abstract base.
- class ElectrostaticPotential3D

  Electrostatic potential of a molecule.
- class OEPotential3D

  Class template for OEP scalar fields.
- class WavefunctionUnion

  Union of two Wavefunction objects.

Typedefs

- using **SharedWavefunction** = std::shared_ptr< Wavefunction >
- using **SharedBasisSet** = std::shared_ptr< BasisSet >
- using **SharedTensor** = std::shared_ptr< Tensor >
- using **SharedMatrix** = std::shared_ptr< Matrix >
- using **SharedVector** = std::shared_ptr< Vector >
- using **SharedScalarField3D** = std::shared_ptr< ScalarField3D >
- using **SharedIntegralFactory** = std::shared_ptr< IntegralFactory >
- using **SharedTwoBodyAOInt** = std::shared_ptr< TwoBodyAOInt >
- using **SharedMolecule** = std::shared_ptr< Molecule >
- using **SharedSuperFunctional** = std::shared_ptr< SuperFunctional >
- using **SharedMOSpace** = std::shared_ptr< MOSpace >
- using **SharedMOSpaceVector** = std::vector< std::shared_ptr< MOSpace >>
- using **SharedIntegralTransform** = std::shared_ptr< IntegralTransform >
- using **SharedLocalizer** = std::shared_ptr< Localizer >

Functions

- void preambule (void)

  Print preambule for module OEPDEV.
- std::shared_ptr< SuperFunctional > create_superfunctional (std::string name, Options &options)

  Set up DFT functional.
- std::shared_ptr< Molecule > extract_monomer (std::shared_ptr< const Molecule > molecule_dimer, int id)

  Extract molecule from dimer.
- std::shared_ptr< Wavefunction > solve_scf (std::shared_ptr< Molecule > molecule, std::shared_ptr< BasisSet > primary, std::shared_ptr< SuperFunctional > functional, Options &options, std::shared_ptr< PSIO > psio)

  Solve RHF-SCF equations for a given molecule in a given basis set.

8.1.1 Detailed Description

oepdev module namespace. Contains:

8.1.2 Function Documentation

8.1.2.1 std::shared_ptr< SuperFunctional > oepdev::create_superfunctional (std::string *name,* Options & *options*)

Set up DFT functional.

Now it accepts only pure HF functional.

Parameters

name	name of the functional ("HF" is now only available)
options	psi::Options object

Returns

psi::SharedSuperFunctional object with functional.

8.1.2.2 std::shared_ptr< Molecule > oepdev::extract_monomer (std::shared_ptr< const Molecule > *molecule_dimer,* int *id*)

Extract molecule from dimer.

Parameters

molecule_dimer	psi::SharedMolecule object with dimer
id	index of a molecule (starts from 1)

Returns

psi::SharedMolecule object with indicated monomer

8.1.2.3 std::shared_ptr< Wavefunction > oepdev::solve_scf (std::shared_ptr< Molecule > *molecule,* std::shared_ptr< BasisSet > *primary,* std::shared_ptr< SuperFunctional > *functional,* Options & *options,* std::shared_ptr< PSIO > *psio*)

Solve RHF-SCF equations for a given molecule in a given basis set.

Parameters

molecule	psi::SharedMolecule object with molecule
primary	shared primary basis set
functional	DFT functional
options	psi::Options object
psio	psi::PSIO object

Returns

psi::SharedWavefunction SCF wavefunction of the molecule

8.2 psi Namespace Reference

Psi4 package namespace.

Typedefs

- using **SharedVetor** = std::shared_ptr< Vector >
- using **SharedBasisSet** = std::shared_ptr< BasisSet >
- using **SharedMolecule** = std::shared_ptr< Molecule >
- using **SharedMatrix** = std::shared_ptr< Matrix >
- using **SharedWavefunction** = std::shared_ptr< Wavefunction >

8.2.1 Detailed Description

Psi4 package namespace. Contains all Psi4 functionalities.

Chapter 9

Class Documentation

9.1 oepdev::AllAOIntegralsIterator Class Reference

Loop over all possible ERI within a particular shell.

`#include <integrals_iter.h>`

Public Member Functions

- **AllAOIntegralsIterator** (const AllAOShellCombinationsIterator &shellIter)
- **AllAOIntegralsIterator** (std::shared_ptr< AllAOShellCombinationsIterator > shellIter)
- void **first** ()
- void **next** ()
- bool **is_done** ()
- int index () const

- int i () const
- int **j** () const
- int **k** () const
- int **l** () const

9.1.1 Detailed Description

Loop over all possible ERI within a particular shell.

Constructed by providing a const reference or shared pointer to an AllAOShellCombinationsIterator object.

Suggested usage:

```
SharedIntegralFactory ints(bs1, bs2, bs3, bs4);
SharedTwoBodyAOInt tei = std::make_shared<TwoBodyAOInt>(ints->eri());
AllAOShellCombinationsIterator shellIter(ints);
const doube * buffer = tei->buffer();
for (shellIter.first(); shellIter.is_done()==false; shellIter.next())
{
     shellIter.compute_shell(tei);
     AllAOIntegralsIterator intsIter(shellIter);
     for (intsIter.first(); intsIter.is_done()==false; intsIter.next())
     {
          // Grab (ij|kl) integrals and indices here
          int i = intsIter.i();
          int j = intsIter.j();
          int k = intsIter.k();
          int l = intsIter.l();
          double integral = buffer[intsIter.index()];
     }
}
```

9.1.2 Member Function Documentation

9.1.2.1 int oepdev::AllAOIntegralsIterator::i ( ) const [inline]

Grab the current integral indices

9.1.2.2 int oepdev::AllAOIntegralsIterator::index ( ) const [inline]

Grab the current index of integral value stored in the buffer

The documentation for this class was generated from the following files:

- oepdev/libutil/integrals_iter.h
- oepdev/libutil/integrals_iter.cc

9.2 oepdev::AllAOShellCombinationsIterator Class Reference

Loop over all possible ERI shells.

`#include <integrals_iter.h>`

Public Member Functions

- **AllAOShellCombinationsIterator** (SharedBasisSet bs_1, SharedBasisSet bs_2, SharedBasisSet bs_3, SharedBasisSet bs_4)
- **AllAOShellCombinationsIterator** (SharedIntegralFactory integrals)
- void **first** ()
- void **next** ()
- bool **is_done** ()
- SharedBasisSet **bs_1** () const
- SharedBasisSet **bs_2** () const
- SharedBasisSet **bs_3** () const
- SharedBasisSet **bs_4** () const
- void **compute_shell** (SharedTwoBodyAOInt tei) const

- int P () const
- int **Q** () const
- int **R** () const
- int **S** () const

9.2.1 Detailed Description

Loop over all possible ERI shells.

Constructed by providing shared pointer to IntegralFactory object or shared pointers to four basis set spaces.

Suggested usage:

```
SharedIntegralFactory ints(bs1, bs2, bs3, bs4);
SharedTwoBodyAOInt tei = std::make_shared<TwoBodyAOInt>(ints->eri());
AllAOShellCombinationsIterator shellIter(ints);
const doube * buffer = tei->buffer();
for (shellIter.first(); shellIter.is_done()==false; shellIter.next())
{
     shellIter.compute_shell(tei);
     AllAOIntegralsIterator intsIter(shellIter);
     for (intsIter.first(); intsIter.is_done()==false; intsIter.next())
     {
```

```
            // Grab (ij|kl) integrals and indices here
            int i = intsIter.i();
            int j = intsIter.j();
            int k = intsIter.k();
            int l = intsIter.l();
            double integral = buffer[intsIter.index()];
        }
}
```

9.2.2 Member Function Documentation

9.2.2.1 int oepdev::AllAOShellCombinationsIterator::P ( ) const [inline]

Grab the current shell indices

The documentation for this class was generated from the following files:

- oepdev/libutil/integrals_iter.h
- oepdev/libutil/integrals_iter.cc

9.3 oepdev::CPHF Class Reference

CPHF solver class.

`#include <cphf.h>`

Public Member Functions

- CPHF (SharedWavefunction ref_wfn, Options &options)
  orbital-associated polarizabilities tensors
- ~CPHF ()
  Desctructor.
- void compute (void)
  run the calculations
- void print (void) const
  print to output file
- std::shared_ptr< Matrix > get_molecular_polarizability (void) const
  retrieve the molecular (total) polarizability

Protected Attributes

- const int _no
  Number of occupied orbitals.
- const int _nv
  Number of virtual orbitals.
- const int _nn
  Number of basis functions.
- long int _memory
  Memory.
- int _maxiter
  Maximum number of iterations.
- double _conv
  CPHF convergence threshold.
- bool _with_diis

whether use DIIS or not

- const int _diis_dim

  Size of subspace.

- std::shared_ptr< BasisSet > _primary

  Primary Basis Set.

- std::shared_ptr< Matrix > _cocc

  Occupied orbitals.

- std::shared_ptr< Matrix > _cvir

  Virtual orbitals.

- std::shared_ptr< Vector > _eps_occ

  Occupied orbital energies.

- std::shared_ptr< Vector > _eps_vir

  Virtual orbital energies.

- std::vector< std::shared_ptr < oepdev::DIISManager > > _diis

  the DIIS managers for each perturbation operator x, y and z

- Options _options

  Options.

- std::shared_ptr< Matrix > _molecular_polarizability

  Total (molecular) polarizability tensor.

9.3.1 Detailed Description

CPHF solver class.

Solves CPHF equations (now only for RHF wavefunction). Computes molecular and orbital-associated polarizabilities.

Suggested usage:

std::shared_ptr<CPHF> cphf(new CPHF(ref_wfn, options)); cphf->compute(); std::shared_ptr<Matrix> polarizability = cphf->get_molecular_polarizability(); std::shared_ptr<Tensor> orbital_polars = cphf->get_orbital_polarizabilities();

9.3.2 Constructor & Destructor Documentation

9.3.2.1 oepdev::CPHF::CPHF (SharedWavefunction *ref_wfn*, Options & *options*)

orbital-associated polarizabilities tensors

Constructor

Parameters

ref_wfn	reference HF wavefunction
options	set of Psi4 options

The documentation for this class was generated from the following files:

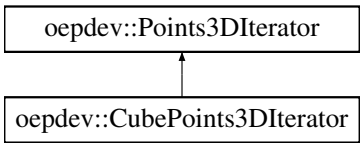
- oepdev/libutil/cphf.h
- oepdev/libutil/cphf.cc

9.4 oepdev::CubePoints3DIterator Class Reference

Iterator over a collection of points in 3D space. g09 Cube-like order.

`#include <space3d.h>`

Inheritance diagram for oepdev::CubePoints3DIterator:



Public Member Functions

- **CubePoints3DIterator** (const int &nx, const int &ny, const int &nz, const double &dx, const double &dy, const double &dz, const double &ox, const double &oy, const double &oz)
- virtual void first ()

  Initialize first iteration.
- virtual void next ()

  Step to next iteration.

Protected Attributes

- const int **nx_**
- const int **ny_**
- const int **nz_**
- const double **dx_**
- const double **dy_**
- const double **dz_**
- const double **ox_**
- const double **oy_**
- const double **oz_**
- int **ii_**
- int **jj_**
- int **kk_**

Additional Inherited Members

9.4.1 Detailed Description

Iterator over a collection of points in 3D space. g09 Cube-like order.

Note: Always create instances by using static factory method from Points3DIterator. Do not use constructor of this class.

The documentation for this class was generated from the following files:

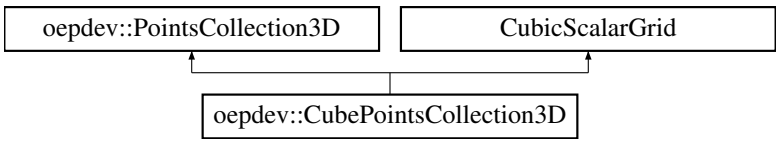
- oepdev/libutil/space3d.h
- oepdev/libutil/space3d.cc

9.5 oepdev::CubePointsCollection3D Class Reference

G09 cube-like ordered collection of points in 3D space.

`#include <space3d.h>`

Inheritance diagram for oepdev::CubePointsCollection3D:



Public Member Functions

- **CubePointsCollection3D** (Collection collectionType, const int &nx, const int &ny, const int &nz, const double &px, const double &py, const double &pz, psi::SharedBasisSet bs, psi::Options &options)
- virtual void print () const

  Print the information to Psi4 output file.
- virtual void **write_cube_file** (psi::SharedMatrix v, const std::string &name)

Additional Inherited Members

9.5.1 Detailed Description

G09 cube-like ordered collection of points in 3D space.

Note: Do not use constructors of this class explicitly. Instead, use static factory methods of the superclass to create instances.

The documentation for this class was generated from the following files:

- oepdev/libutil/space3d.h
- oepdev/libutil/space3d.cc

9.6 oepdev::DIISManager Class Reference

DIIS manager.

`#include <diis.h>`

Public Member Functions

- DIISManager (int dim, int na, int nb)
- ~DIISManager ()

  Destructor.
- void put (const std::shared_ptr< const Matrix > &error, const std::shared_ptr< const Matrix > &vector)
- void compute (void)
- void update (std::shared_ptr< Matrix > &other)

9.6.1 Detailed Description

DIIS manager.

Instance can interact directly with the process of solving vector quantities in iterative manner. One needs to pass the dimensions of solution vector as well as the DIIS subspace size. The iterative procedure requires providing the current vector and also an estimate of the error vector. The updated DIIS vector can be copied to an old vector through the Instance.

9.6.2 Constructor & Destructor Documentation

9.6.2.1 oepdev::DIISManager::DIISManager (int *dim,* int *na,* int *nb*)

Constructor.

Parameters

dim	Size of DIIS subspace
na	Number of solution rows
nb	Number of solution columns

9.6.3 Member Function Documentation

9.6.3.1 void oepdev::DIISManager::compute (void)

Perform DIIS interpolation.

9.6.3.2 void oepdev::DIISManager::put (const std::shared_ptr< const Matrix > & *error,* const std::shared_ptr< const Matrix > & *vector*)

Put the current solution to the DIIS manager.

Parameters

error	Shared matrix with current solution error
vector	Shared matrix with current solution vector

9.6.3.3 void oepdev::DIISManager::update (std::shared_ptr< Matrix > & *other*)

Update solution vector. Pass the Shared pointer to current solution. Then it will be overriden by the updated DIIS solution.

The documentation for this class was generated from the following files:

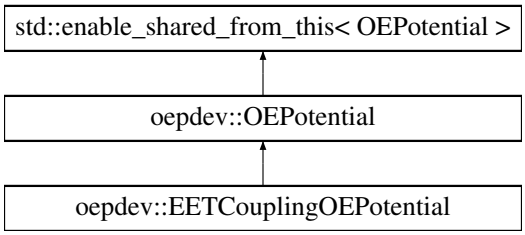
- oepdev/libutil/diis.h
- oepdev/libutil/diis.cc

9.7 oepdev::EETCouplingOEPotential Class Reference

Generalized One-Electron Potential for EET coupling calculations.

`#include <oep.h>`

Inheritance diagram for oepdev::EETCouplingOEPotential:



Public Member Functions

- **EETCouplingOEPotential** (SharedWavefunction wfn, SharedBasisSet auxiliary, Options &options)
- **EETCouplingOEPotential** (SharedWavefunction wfn, Options &options)
- virtual void **compute** (const std::string &oepType) override
- virtual void compute_3D (const std::string &oepType, const double &x, const double &y, const double &z, double &v) override
- virtual void **print_header** () const override

Additional Inherited Members

9.7.1 Detailed Description

Generalized One-Electron Potential for EET coupling calculations.

Contains the following OEP types: "ET1" "ET2" "HT1" "HT1" "HT2" "CT1" "CT2"

9.7.2 Member Function Documentation

9.7.2.1 void oepdev::EETCouplingOEPotential::compute_3D (const std::string & *oepType*, const double & *x*, const double & *y*, const double & *z*, double & *v*) [override], [virtual]

Compute value of potential in point x, y, z and save at v

Implements oepdev::OEPotential.

The documentation for this class was generated from the following files:

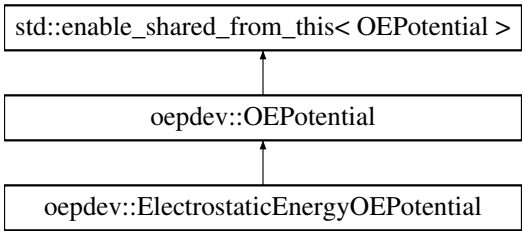
- oepdev/liboep/oep.h
- oepdev/liboep/oep.cc

9.8 oepdev::ElectrostaticEnergyOEPotential Class Reference

Generalized One-Electron Potential for Electrostatic Energy calculations.

`#include <oep.h>`

Inheritance diagram for oepdev::ElectrostaticEnergyOEPotential:



Public Member Functions

- ElectrostaticEnergyOEPotential (SharedWavefunction wfn, Options &options)

  Only ESP-based potential is worth implementing.
- virtual void **compute** (const std::string &oepType) override
- virtual void compute_3D (const std::string &oepType, const double &x, const double &y, const double &z, double &v) override
- virtual void **print_header** () const override

Additional Inherited Members

9.8.1 Detailed Description

Generalized One-Electron Potential for Electrostatic Energy calculations.

Contains the following OEP types: "V"

9.8.2 Member Function Documentation

9.8.2.1 void oepdev::ElectrostaticEnergyOEPotential::compute_3D (const std::string & *oepType*, const double & *x*, const double & *y*, const double & *z*, double & *v*) [override], [virtual]

Compute value of potential in point x, y, z and save at v

Implements oepdev::OEPotential.

The documentation for this class was generated from the following files:

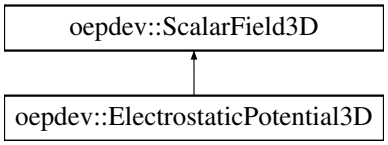
- oepdev/liboep/oep.h
- oepdev/liboep/oep.cc

9.9 oepdev::ElectrostaticPotential3D Class Reference

Electrostatic potential of a molecule.

`#include <space3d.h>`

Inheritance diagram for oepdev::ElectrostaticPotential3D:



Public Member Functions

- **ElectrostaticPotential3D** (const int &np, const double &radius, const double &cx, const double &cy, const double &cz)
- **ElectrostaticPotential3D** (const int &np, const double &padding, psi::SharedWavefunction wfn, psi::Options &options)
- **ElectrostaticPotential3D** (const int &nx, const int &ny, const int &nz, const double &px, const double &py, const double &pz, psi::SharedWavefunction wfn, psi::Options &options)
- virtual double compute_xyz (const double &x, const double &y, const double &z)

  Compute a value of scalar field at point (x, y, z)

- virtual void print () const

  Print information of the object to Psi4 output.

Additional Inherited Members

9.9.1 Detailed Description

Electrostatic potential of a molecule.

Computes the electrostatic potential $v(\mathbf{r})$ at point $\mathbf{r}$ is computed from the following formula:

$$v(\mathbf{r}) = v(\mathbf{r})_{\mathrm{nuc}} + v(\mathbf{r})_{\mathrm{el}}$$

where the nuclear and electronic contributions are defined accordingly as

$$v(\mathbf{r})_{\mathrm{nuc}} = \sum_x \frac{Z_x}{|\mathbf{r} - \mathbf{r}_x|}$$
$$v(\mathbf{r})_{\mathrm{el}} = \sum_{\mu\nu} \left\{ D_{\mu\nu}^{(\alpha)} + D_{\mu\nu}^{(\beta)} \right\} V_{\nu\mu}(\mathbf{r})$$

In the above equations, Z_x denotes the charge of xth nucleus, $D_{\mu\nu}^{(\omega)}$ is the one-particle (relaxed) density matrix element in AO basis associated with the ω electron spin, and $V_{\mu\nu}(\mathbf{r})$ is the potential one-electron integral defined by

$$V_{\nu\mu}(\mathbf{r}) \equiv \int d\mathbf{r}' \varphi_\nu^*(\mathbf{r}') \frac{1}{|\mathbf{r} - \mathbf{r}'|} \varphi_\mu(\mathbf{r}')$$

The documentation for this class was generated from the following files:

- oepdev/libutil/space3d.h
- oepdev/libutil/space3d.cc

9.10 oepdev::ESPSolver Class Reference

Charges from Electrostatic Potential (ESP). A solver-type class.

`#include <esp.h>`

Public Member Functions

- ESPSolver (SharedScalarField3D field)

  Construct from scalar field.
- ESPSolver (SharedScalarField3D field, psi::SharedMatrix centres)

  Construct from scalar field.
- virtual ~ESPSolver ()

  Destructor.
- virtual psi::SharedVector charges () const

  Get the (fit) charges.
- virtual psi::SharedMatrix centres () const

  Get the charge distribution centres.
- virtual void compute ()

  Perform fitting of effective charges.

Protected Attributes

- const int nCentres_

  Number of fit centres.
- SharedScalarField3D field_

  Scalar field.
- psi::SharedVector charges_

  Charges to be fit.
- psi::SharedMatrix centres_

  Centres, at which fit charges will reside.

9.10.1 Detailed Description

Charges from Electrostatic Potential (ESP). A solver-type class.

Solves the least-squares problem to fit the generalized charges q_m, that reproduce the reference generalized potential $v^{\text{ref}}(\mathbf{r})$ supplied by the `ScalarField3D` object:

$$\int d\mathbf{r}' \left[v^{\text{ref}}(\mathbf{r}') - \sum_m \frac{q_m}{|\mathbf{r}' - \mathbf{r}_m|} \right]^2 \rightarrow \text{minimize}$$

The charges are subject to the following constraint:

$$\sum_m q_m = 0$$

Method description.

M generalized charges is found by solving the matrix equation

$$\begin{pmatrix} \mathbf{A} & 0 \\ 0 & 1 \end{pmatrix}^{-1} \cdot \begin{pmatrix} \mathbf{b} \\ 0 \end{pmatrix} = \begin{pmatrix} \mathbf{q} \\ \lambda \end{pmatrix}$$

where the $\mathbf{A}$ matrix of dimension $M \times M$ and b} vector or length M are given as

$$A_{mn} = \sum_i \frac{1}{r_{im} r_{in}}$$

$$b_m = \sum_i \frac{v^{\text{ref}}(\mathbf{r}_m)}{r_{im}}$$

In the above equation, summations run over all sample points, at which reference potential is known.

9.10.2 Constructor & Destructor Documentation

9.10.2.1 oepdev::ESPSolver::ESPSolver (SharedScalarField3D *field*)

Construct from scalar field.

Assume that the centres are on atoms associated with the scalar field.

Parameters

field	- oepdev scalar field object

9.10.2.2 oepdev::ESPSolver::ESPSolver (SharedScalarField3D *field*, psi::SharedMatrix *centres*)

Construct from scalar field.

Solve ESP equations for a custom set of charge distribution centres.

Parameters

field	- oepdev scalar field object
centres	- matrix with coordinates of charge distribution centres

The documentation for this class was generated from the following files:

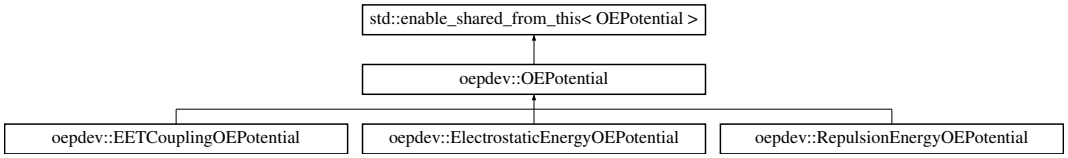
- oepdev/libutil/esp.h
- oepdev/libutil/esp.cc

9.11 oepdev::OEPotential Class Reference

Generalized One-Electron Potential: Abstract base.

`#include <oep.h>`

Inheritance diagram for oepdev::OEPotential:



Public Member Functions

- OEPotential (SharedWavefunction wfn, Options &options)

  ESP-based OEP object.
- OEPotential (SharedWavefunction wfn, SharedBasisSet auxiliary, Options &options)

  DF-based OEP object.
- virtual ~OEPotential ()

  Destructor.
- virtual void rotate (const Matrix &rotmat)

  Rotate.
- virtual void translate (const Vector &trans)

  Translate.
- virtual void superimpose (const Matrix &refGeometry, const std::vector< int > &supList, const std::vector< int > &reordList)

  Superimpose.
- std::string name () const

  Retrieve name of this OEP.
- SharedMatrix matrix (const std::string &oepType) const

  Retrieve matrix potential.
- SharedWavefunction wfn () const

  Retrieve wavefunction object.
- void **set_name** (const std::string &name)
- virtual void **print_header** () const =0

- virtual void **compute** (const std::string &oepType)=0
- virtual void **compute** (void)

- virtual void write_cube (const std::string &oepType, const std::string &fileName)
- virtual void compute_3D (const std::string &oepType, const double &x, const double &y, const double &z, double &v)=0

Static Public Member Functions

- static std::shared_ptr < OEPotential > build (const std::string &category, SharedWavefunction wfn, Options &options)

  Build ESP-based OEP object.
- static std::shared_ptr < OEPotential > build (const std::string &category, SharedWavefunction wfn, SharedBasisSet auxiliary, Options &options)

  Build DF-based OEP object.

Public Attributes

- const bool is_density_fitted

  Is this OEP density-fitted?
- const bool is_esp_based

  Is this OEP ESP-based?

Protected Attributes

- Options options_

  Psi4 options.
- SharedWavefunction wfn_

  Wavefunction.
- SharedBasisSet primary_

  Promary Basis set.
- SharedBasisSet auxiliary_

  Auxiliary Basis set.
- std::string name_

  Name of this OEP;.
- std::vector< std::string > oepTypes_

  Types of OEP's within the scope of this object.
- std::map< std::string, SharedMatrix > oepMatrices_

  OEP's matrix forms for each OEP type.
- std::shared_ptr < psi::IntegralFactory > intsFactory_

  Integral factory.
- std::shared_ptr< psi::Matrix > potMat_

  Matrix of potential one-electron integrals.
- std::shared_ptr < psi::OneBodyAOInt > OEInt_

  One-electron integral shared pointer.
- std::shared_ptr< PotentialInt > potInt_

  One-electron potential shared pointer.

9.11.1 Detailed Description

Generalized One-Electron Potential: Abstract base.

Manages OEP's in matrix and 3D forms.

9.11.2 Constructor & Destructor Documentation

9.11.2.1 oepdev::OEPotential::OEPotential (SharedWavefunction *wfn,* Options & *options*)

ESP-based OEP object.

Parameters

wfn	- wavefunction
options	- Psi4 options

9.11.2.2 oepdev::OEPotential::OEPotential (SharedWavefunction *wfn,* SharedBasisSet *auxiliary,* Options & *options*)

DF-based OEP object.

Parameters

wfn	- wavefunction
auxiliary	- basis set for density fitting of OEP's
options	- Psi4 options

9.11.3 Member Function Documentation

9.11.3.1 std::shared_ptr< OEPotential > oepdev::OEPotential::build (const std::string & *category,* SharedWavefunction *wfn,* Options & *options*) [static]

Build ESP-based OEP object.

Parameters

type	- OEP category
wfn	- wavefunction
options	- Psi4 options

9.11.3.2 std::shared_ptr< OEPotential > oepdev::OEPotential::build (const std::string & *category,* SharedWavefunction *wfn,* SharedBasisSet *auxiliary,* Options & *options*) [static]

Build DF-based OEP object.

Parameters

type	- OEP category
wfn	- wavefunction
auxiliary	- basis set for density fitting of OEP's
options	- Psi4 options

9.11.3.3 void oepdev::OEPotential::compute_3D (const std::string & *oepType,* const double & *x,* const double & *y,* const double & *z,* double & *v*) [pure virtual]

Compute value of potential in point x, y, z and save at v

Implemented in oepdev::EETCouplingOEPotential, oepdev::RepulsionEnergyOEPotential, and oepdev::ElectrostaticEnergyOEPotential.

9.11.3.4 void oepdev::OEPotential::write_cube (const std::string & *oepType,* const std::string & *fileName*) [virtual]

Write potential to a cube file

The documentation for this class was generated from the following files:

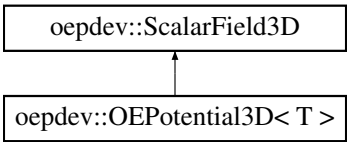
- oepdev/liboep/oep.h
- oepdev/liboep/oep.cc

9.12 oepdev::OEPotential3D< T > Class Template Reference

Class template for OEP scalar fields.

`#include <space3d.h>`

Inheritance diagram for oepdev::OEPotential3D< T >:



Public Member Functions

- OEPotential3D (const int &np, const double &padding, std::shared_ptr< T > oep, const std::string &oepType)

  Construct random spherical collection of scalar field of type T.
- OEPotential3D (const int &nx, const int &ny, const int &nz, const double &px, const double &py, const double &pz, std::shared_ptr< T > oep, const std::string &oepType, psi::Options &options)

  Construct ordered 3D collection of scalar field of type T.
- virtual ~OEPotential3D ()

  Destructor.
- virtual double compute_xyz (const double &x, const double &y, const double &z)

  Compute a value of scalar field at point (x, y, z)
- virtual void print () const

  Print information of the object to Psi4 output.

Protected Attributes

- std::shared_ptr< T > oep_

  Shared pointer to the instance of class `T`
- std::string oepType_

  Descriptor of the scalar field type stored in instance of `T`

Additional Inherited Members

9.12.1 Detailed Description

template<class T>class oepdev::OEPotential3D< T >

Class template for OEP scalar fields.

Used for special type of classes T that contain following public member functions:

```
class T : public std::enable_shared_from_this<T> {

  public:
    void compute_3D(const std::string& descriptor,
                    const double& x, const double& y, const double& z,
                    double& v);

    shared_ptr<psi::Wavefunction> wfn() const {return wfn_;}

};
```

with the `descriptor` of a certain scalar field type, `x`, `y`, `z` the points in 3D space in which the scalar field has to be computed and stored at `v`. Instances of `T` should store shared pointer to wavefunction object. List of classes `T` that are compatible with this class template and are currently implemented in oepdev is given below:

- `oepdev::OEPotential` abstract base (do not use derived classes as `T`)

Template parameters:

Template Parameters

T	the compatible class (e.g. `oepdev::OEPotential`)

9.12.2 Constructor & Destructor Documentation

9.12.2.1 template<class T > oepdev::OEPotential3D< T >::OEPotential3D (const int & *np*, const double & *padding*, std::shared_ptr< T > *oep*, const std::string & *oepType*)

Construct random spherical collection of scalar field of type T.

The points are drawn according to uniform distrinution in 3D space.

Parameters

np	- number of points to draw
padding	- spherical padding distance (au)
oep	- OEP object of type T
oepType	- type of OEP

9.12.2.2 template<class T > oepdev::OEPotential3D< T >::OEPotential3D (const int & *nx*, const int & *ny*, const int & *nz*, const double & *px*, const double & *py*, const double & *pz*, std::shared_ptr< T > *oep*, const std::string & *oepType*, psi::Options & *options*)

Construct ordered 3D collection of scalar field of type T.

The points are generated according to Gaussian cube file format.

Parameters

nx	- number of points along x direction
ny	- number of points along y direction
nz	- number of points along z direction
px	- padding distance along x direction
py	- padding distance along y direction
pz	- padding distance along z direction
oep	- OEP object of type T
oepType	- type of OEP
options	- Psi4 options object

The documentation for this class was generated from the following file:

- oepdev/libutil/space3d.h

9.13 oepdev::Points3DIterator::Point Struct Reference

Public Attributes

- double **x**

- double **y**
- double **z**
- int **index**

The documentation for this struct was generated from the following file:

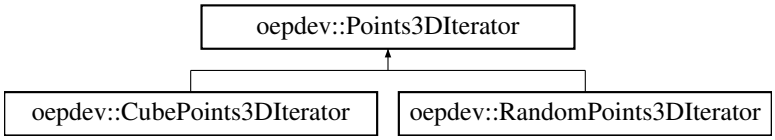
- oepdev/libutil/space3d.h

9.14 oepdev::Points3DIterator Class Reference

Iterator over a collection of points in 3D space. Abstract base.

`#include <space3d.h>`

Inheritance diagram for oepdev::Points3DIterator:



Classes

- struct Point

Public Member Functions

- Points3DIterator (const int &np)

  Plain constructor. Initializes the abstract features.
- virtual ~Points3DIterator ()

  Destructor.
- virtual bool is_done ()

  Check if iteration is finished.
- virtual void first ()=0

  Initialize first iteration.
- virtual void next ()=0

  Step to next iteration.

- virtual double **x** () const
- virtual double **y** () const
- virtual double **z** () const
- virtual int **index** () const

Static Public Member Functions

- static shared_ptr
  < Points3DIterator > build (const int &nx, const int &ny, const int &nz, const double &dx, const double &dy, const double &dz, const double &ox, const double &oy, const double &oz)

  Build G09 Cube collection iterator.

- static shared_ptr
  < Points3DIterator > build (const int &np, const double &radius, const double &cx, const double &cy, const double &cz)

  Build random collection iterator.
- static shared_ptr
  < Points3DIterator > build (const int &np, const double &pad, psi::SharedMolecule mol)

  Build random collection iterator.

Protected Attributes

- const int np_

  Number of points.
- bool done_

  Status of the iterator.
- int index_

  Current index.

- Point **current_**

9.14.1 Detailed Description

Iterator over a collection of points in 3D space. Abstract base.

Points3DIterators are constructed either as iterators over:

- a random collections or
- an ordered (g09 cube-like) collections. **Note:** Always create instances by using static factory methods.

9.14.2 Constructor & Destructor Documentation

9.14.2.1 oepdev::Points3DIterator::Points3DIterator (const int & *np*)

Plain constructor. Initializes the abstract features.

Parameters

np	- number of points this iterator is constructed for

9.14.3 Member Function Documentation

9.14.3.1 std::shared_ptr< Points3DIterator > oepdev::Points3DIterator::build (const int & *nx*, const int & *ny*, const int & *nz*, const double & *dx*, const double & *dy*, const double & *dz*, const double & *ox*, const double & *oy*, const double & *oz*) [static]

Build G09 Cube collection iterator.

The points are generated according to Gaussian cube file format.

Parameters

nx	- number of points along x direction

ny	- number of points along y direction
nz	- number of points along z direction
dx	- spacing distance along x direction
dy	- spacing distance along y direction
dz	- spacing distance along y direction
ox	- coordinate x of cube origin
oy	- coordinate y of cube origin
oz	- coordinate z of cube origin

9.14.3.2 std::shared_ptr< Points3DIterator > oepdev::Points3DIterator::build (const int & *np*, const double & *radius*, const double & *cx*, const double & *cy*, const double & *cz*) [static]

Build random collection iterator.

The points are drawn according to uniform distrinution in 3D space.

Parameters

np	- number of points to draw
radius	- sphere radius inside which points are to be drawn
cx	- coordinate x of sphere's centre
cy	- coordinate y of sphere's centre
cz	- coordinate z of sphere's centre

9.14.3.3 shared_ptr< Points3DIterator > oepdev::Points3DIterator::build (const int & *np*, const double & *pad*, psi::SharedMolecule *mol*) [static]

Build random collection iterator.

The points are drawn according to uniform distrinution in 3D space enclosing a molecule given. All drawn points lie outside the van der Waals volume.

Parameters

np	- number of points to draw
pad	- radius padding of a minimal sphere enclosing the molecule
mol	- Psi4 molecule object

The documentation for this class was generated from the following files:

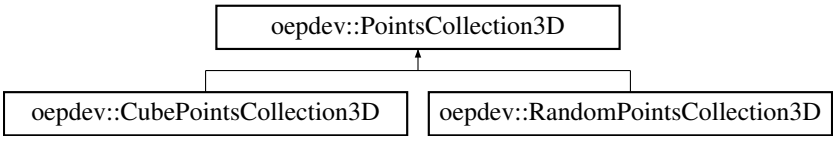
- oepdev/libutil/space3d.h
- oepdev/libutil/space3d.cc

9.15 oepdev::PointsCollection3D Class Reference

Collection of points in 3D space. Abstract base.

`#include <space3d.h>`

Inheritance diagram for oepdev::PointsCollection3D:



Public Types

- enum Collection { **Random**, **Cube** }

  Public descriptior of collection type.

Public Member Functions

- PointsCollection3D (Collection collectionType, int &np)

  Initialize abstract features.
- **PointsCollection3D** (Collection collectionType, const int &np)
- virtual ~PointsCollection3D ()

  Destructor.
- virtual int npoints () const

  Get the number of points.
- virtual shared_ptr < Points3DIterator > points_iterator () const

  Get the iterator over this collection of points.
- virtual Collection get_type () const

  Get the collection type.
- virtual void print () const =0

  Print the information to Psi4 output file.

Static Public Member Functions

- static shared_ptr < PointsCollection3D > build (const int &npoints, const double &radius, const double &cx=0.0, const double &cy=0.0, const double &cz=0.0)

  Build random collection of points.
- static shared_ptr < PointsCollection3D > build (const int &npoints, const double &padding, psi::SharedMolecule mol)

  Build random collection of points.
- static shared_ptr < PointsCollection3D > build (const int &nx, const int &ny, const int &nz, const double &px, const double &py, const double &pz, psi::SharedBasisSet bs, psi::Options &options)

  Build G09 Cube collection of points.

Protected Attributes

- const int np_

  Number of points.
- Collection collectionType_

  Collection type.
- shared_ptr< Points3DIterator > pointsIterator_

  iterator over points collection

9.15.1 Detailed Description

Collection of points in 3D space. Abstract base.

Create random or ordered (g09 cube-like) collections of points in 3D space.

Note: Always create instances by using static factory methods.

9.15.2 Constructor & Destructor Documentation

9.15.2.1 oepdev::PointsCollection3D::PointsCollection3D (Collection *collectionType,* int & *np*)

Initialize abstract features.

Parameters

np	- number of points to be created

9.15.3 Member Function Documentation

9.15.3.1 std::shared_ptr< PointsCollection3D > oepdev::PointsCollection3D::build (const int & *npoints,* const double & *radius,* const double & *cx* = 0.0, const double & *cy* = 0.0, const double & *cz* = 0.0) [static]

Build random collection of points.

Points uniformly span a sphere.

Parameters

npoints	- number of points to draw
radius	- sphere radius inside which points are to be drawn
cx	- coordinate x of sphere's centre
cy	- coordinate y of sphere's centre
cz	- coordinate z of sphere's centre

9.15.3.2 std::shared_ptr< PointsCollection3D > oepdev::PointsCollection3D::build (const int & *npoints,* const double & *padding,* psi::SharedMolecule *mol*) [static]

Build random collection of points.

Points uniformly span space inside a sphere enclosing a molecule. exluding the van der Waals volume.

Parameters

np	- number of points to draw
padding	- radius padding of a minimal sphere enclosing the molecule
mol	- Psi4 molecule object

9.15.3.3 std::shared_ptr< PointsCollection3D > oepdev::PointsCollection3D::build (const int & *nx,* const int & *ny,* const int & *nz,* const double & *px,* const double & *py,* const double & *pz,* psi::SharedBasisSet *bs,* psi::Options & *options*) [static]

Build G09 Cube collection of points.

The points span a parallelpiped according to Gaussian cube file format.

Parameters

nx	- number of points along x direction
ny	- number of points along y direction
nz	- number of points along z direction
px	- padding distance along x direction

py	- padding distance along y direction
pz	- padding distance along z direction
bs	- Psi4 basis set object
options	- Psi4 options object

The documentation for this class was generated from the following files:

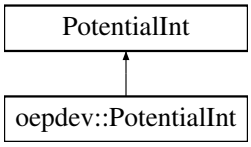
- oepdev/libutil/space3d.h
- oepdev/libutil/space3d.cc

9.16 oepdev::PotentialInt Class Reference

Computes potential integrals.

`#include <potential.h>`

Inheritance diagram for oepdev::PotentialInt:



Public Member Functions

- PotentialInt (std::vector< psi::SphericalTransform > &st, std::shared_ptr< psi::BasisSet > bs1, std::shared_ptr< psi::BasisSet > bs2, int deriv=0)

  Constructor. Initialize identically like in psi::PotentilInt.
- PotentialInt (std::vector< psi::SphericalTransform > &st, std::shared_ptr< psi::BasisSet > bs1, std::shared_ptr< psi::BasisSet > bs2, std::shared_ptr< psi::Matrix > Qxyz, int deriv=0)

  Constructor. Takes an arbitrary collection of charges.
- PotentialInt (std::vector< psi::SphericalTransform > &, std::shared_ptr< psi::BasisSet >, std::shared_ptr< psi::BasisSet >, const double &x, const double &y, const double &z, const double &q=1.0, int deriv=0)

  Constructor. Computes potential for one point x, y, z for a test particle of charge q.
- void set_charge_field (const double &x, const double &y, const double &z, const double &q=1.0)

  Mutator. Set the charge field to be a x, y, z point of charge q.

9.16.1 Detailed Description

Computes potential integrals.

9.16.2 Constructor & Destructor Documentation

9.16.2.1 oepdev::PotentialInt::PotentialInt (std::vector< psi::SphericalTransform > & *st*, std::shared_ptr< psi::BasisSet > *bs1*, std::shared_ptr< psi::BasisSet > *bs2*, int *deriv* = 0)

Constructor. Initialize identically like in psi::PotentilInt.

Parameters

st	- Spherical transform object
bs1	- basis set for first space
bs2	- basis set for second space
deriv	- derivative level

9.16.2.2 oepdev::PotentialInt::PotentialInt (std::vector< psi::SphericalTransform > & *st*, std::shared_ptr< psi::BasisSet > *bs1*, std::shared_ptr< psi::BasisSet > *bs2*, std::shared_ptr< psi::Matrix > *Qxyz*, int *deriv* = 0)

Constructor. Takes an arbitrary collection of charges.

Parameters

st	- Spherical transform object
bs1	- basis set for first space
bs2	- basis set for second space
Qxyz	- matrix with charges and their positions
deriv	- derivative level

9.16.2.3 oepdev::PotentialInt::PotentialInt (std::vector< psi::SphericalTransform > & *st*, std::shared_ptr< psi::BasisSet > *bs1*, std::shared_ptr< psi::BasisSet > *bs2*, const double & *x*, const double & *y*, const double & *z*, const double & *q* = 1.0, int *deriv* = 0)

Constructor. Computes potential for one point x, y, z for a test particle of charge q.

Parameters

st	- Spherical transform object
bs1	- basis set for first space
bs2	- basis set for second space
x	- x coordinate of q
y	- y coordinate of q
z	- z coordinate of q
q	- value of the probe charge
deriv	- derivative level

9.16.3 Member Function Documentation

9.16.3.1 void oepdev::PotentialInt::set_charge_field (const double & *x*, const double & *y*, const double & *z*, const double & *q* = 1.0)

Mutator. Set the charge field to be a x, y, z point of charge q.

Parameters

x	- x coordinate of q
y	- y coordinate of q
z	- z coordinate of q
q	- value of the probe charge

The documentation for this class was generated from the following files:

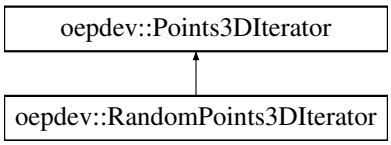
- oepdev/libpsi/potential.h
- oepdev/libpsi/potential.cc

9.17 oepdev::RandomPoints3DIterator Class Reference

Iterator over a collection of points in 3D space. Random collection.

`#include <space3d.h>`

Inheritance diagram for oepdev::RandomPoints3DIterator:



Public Member Functions

- **RandomPoints3DIterator** (const int &np, const double &radius, const double &cx, const double &cy, const double &cz)
- **RandomPoints3DIterator** (const int &np, const double &pad, psi::SharedMolecule mol)
- virtual void first ()

  Initialize first iteration.
- virtual void next ()

  Step to next iteration.

Protected Member Functions

- virtual double **random_double** ()
- virtual void **draw_random_point** ()
- virtual bool **is_in_vdWsphere** (double x, double y, double z) const

Protected Attributes

- double **cx_**
- double **cy_**
- double **cz_**
- double **radius_**
- double **r_**
- double **phi_**
- double **theta_**
- double **x_**
- double **y_**
- double **z_**
- psi::SharedMatrix **excludeSpheres_**
- std::map< std::string, double > **vdwRadius_**
- std::default_random_engine **randomNumberGenerator_**
- std::uniform_real_distribution < double > **randomDistribution_**

Additional Inherited Members

9.17.1 Detailed Description

Iterator over a collection of points in 3D space. Random collection.

Note: Always create instances by using static factory method from Points3DIterator. Do not use constructors of this class.

The documentation for this class was generated from the following files:

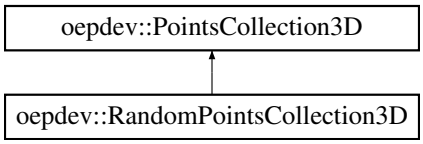
- oepdev/libutil/space3d.h
- oepdev/libutil/space3d.cc

9.18 oepdev::RandomPointsCollection3D Class Reference

Collection of random points in 3D space.

`#include <space3d.h>`

Inheritance diagram for oepdev::RandomPointsCollection3D:



Public Member Functions

- **RandomPointsCollection3D** (Collection collectionType, const int &npoints, const double &radius, const double &cx, const double &cy, const double &cz)
- **RandomPointsCollection3D** (Collection collectionType, const int &npoints, const double &padding, psi::SharedMolecule mol)
- virtual void print () const

  Print the information to Psi4 output file.

Additional Inherited Members

9.18.1 Detailed Description

Collection of random points in 3D space.

Note: Do not use constructors of this class explicitly. Instead, use static factory methods of the superclass to create instances.

The documentation for this class was generated from the following files:

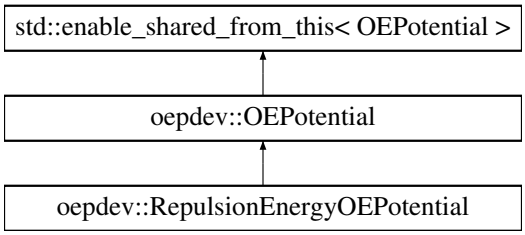
- oepdev/libutil/space3d.h
- oepdev/libutil/space3d.cc

9.19 oepdev::RepulsionEnergyOEPotential Class Reference

Generalized One-Electron Potential for Pauli repulsion energy calculations.

`#include <oep.h>`

Inheritance diagram for oepdev::RepulsionEnergyOEPotential:



Public Member Functions

- **RepulsionEnergyOEPotential** (SharedWavefunction wfn, SharedBasisSet auxiliary, Options &options)
- **RepulsionEnergyOEPotential** (SharedWavefunction wfn, Options &options)
- virtual void **compute** (const std::string &oepType) override
- virtual void compute_3D (const std::string &oepType, const double &x, const double &y, const double &z, double &v) override
- virtual void **print_header** () const override

Additional Inherited Members

9.19.1 Detailed Description

Generalized One-Electron Potential for Pauli repulsion energy calculations.

Contains the following OEP types:

9.19.2 Member Function Documentation

9.19.2.1 void oepdev::RepulsionEnergyOEPotential::compute_3D (const std::string & *oepType,* const double & *x,* const double & *y,* const double & *z,* double & *v*) [override], [virtual]

Compute value of potential in point x, y, z and save at v

Implements oepdev::OEPotential.

The documentation for this class was generated from the following files:

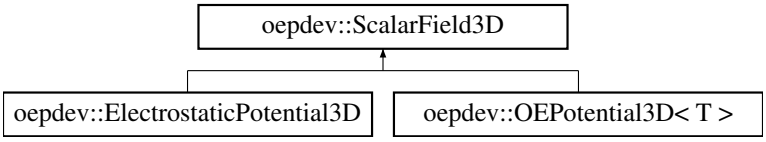
- oepdev/liboep/oep.h
- oepdev/liboep/oep.cc

9.20 oepdev::ScalarField3D Class Reference

Scalar field in 3D space. Abstract base.

`#include <space3d.h>`

Inheritance diagram for oepdev::ScalarField3D:



Public Member Functions

- **ScalarField3D** (const int &np, const double &radius, const double &cx, const double &cy, const double &cz)
- **ScalarField3D** (const int &np, const double &pad, psi::SharedWavefunction wfn, psi::Options &options)
- **ScalarField3D** (const int &nx, const int &ny, const int &nz, const double &px, const double &py, const double &pz, std::shared_ptr< psi::Wavefunction > wfn, psi::Options &options)
- virtual ~ScalarField3D ()

  Destructor.
- virtual int npoints () const

  Get the number of points at which the scalar field is defined.
- virtual std::shared_ptr < PointsCollection3D > points_collection () const

  Get the collection of points.
- virtual std::shared_ptr < psi::Matrix > data () const

  Get the data matrix in a form { [x, y, z, f(x, y, z)] }.
- virtual std::shared_ptr < psi::Wavefunction > wfn () const

  Get the wavefunction.
- virtual bool is_computed () const

  Get the information if data is already computed or not.
- virtual void compute ()

  Compute the scalar field in each point from the point collection.
- virtual double compute_xyz (const double &x, const double &y, const double &z)=0

  Compute a value of scalar field at point (x, y, z)
- virtual void write_cube_file (const std::string &name)

  Write the cube file (only for Cube collections, otherwise does nothing)
- virtual void print () const =0

  Print information of the object to Psi4 output.

Static Public Member Functions

- static shared_ptr< ScalarField3D > **build** (const std::string &type, const int &np, const double &radius, const double &cx, const double &cy, const double &cz)
- static shared_ptr< ScalarField3D > **build** (const std::string &type, const int &np, const double &pad, psi::SharedWavefunction wfn, psi::Options &options)
- static shared_ptr< ScalarField3D > **build** (const std::string &type, const int &nx, const int &ny, const int &nz, const double &px, const double &py, const double &pz, psi::SharedWavefunction wfn, psi::Options &options)

Protected Attributes

- std::shared_ptr < PointsCollection3D > pointsCollection_

  Collection of points at which the scalar field is to be computed.
- std::shared_ptr< psi::Matrix > data_

The data matrix in a form { [x, y, z, f(x, y, z)] }.

- std::shared_ptr < psi::Wavefunction > wfn_

  Wavefunction.
- psi::Matrix geom_

  Geometry of a molecule.
- std::shared_ptr < psi::IntegralFactory > fact_

  Integral factory.
- std::shared_ptr< psi::Matrix > pot_

  Matrix of potential one-electron integrals.
- std::shared_ptr < psi::OneBodyAOInt > oneInt_

  One-electron integral shared pointer.
- std::shared_ptr< PotentialInt > potInt_

  One-electron potential shared pointer.
- std::shared_ptr< psi::BasisSet > primary_

  Basis set.
- int nbf_

  Number of basis functions.
- bool isComputed_

  Has data already computed?

9.20.1 Detailed Description

Scalar field in 3D space. Abstract base.

Create scalar field defined at points distributed randomly or as an ordered g09 cube-like collection. Currently implemented scalar fields are:

- Electrostatic potential - computes electrostatic potential (requires wavefunction)
- Template of generic classes - compute custom scalar fields (requires generic object that is able to compute the field in 3D space)

Note: Always create instances by using static factory methods `build`.

The documentation for this class was generated from the following files:

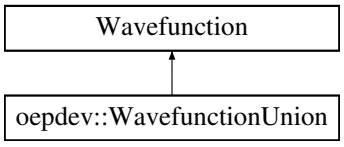
- oepdev/libutil/space3d.h
- oepdev/libutil/space3d.cc

9.21 oepdev::WavefunctionUnion Class Reference

Union of two Wavefunction objects.

`#include <wavefunction_union.h>`

Inheritance diagram for oepdev::WavefunctionUnion:



Public Member Functions

- WavefunctionUnion (SharedWavefunction ref_wfn, Options &options)

  Constructor.
- virtual ~WavefunctionUnion ()

  Destructor.
- virtual double compute_energy ()

  Compute Energy (now blank)
- void localize_orbitals ()

  Localize Molecular Orbitals.
- void transform_integrals ()

  Transform Integrals (2- and 4-index transformations)
- int **l_nmo** (int n) const
- int **l_nso** (int n) const
- int **l_nbf** (int n) const
- int **l_ndocc** (int n) const
- int **l_nvir** (int n) const
- int **l_noffs_ao** (int n) const
- int **l_nalpha** (int n) const
- int **l_nbeta** (int n) const
- double **l_energy** (int n) const
- SharedMolecule **l_molecule** (int n) const
- SharedBasisSet **l_primary** (int n) const
- SharedBasisSet **l_auxiliary** (int n) const
- SharedWavefunction **l_wfn** (int n) const
- SharedMOSpace **l_mospace** (int n, const std::string &label) const
- SharedLocalizer **l_localizer** (int n) const
- SharedIntegralTransform **integrals** (void) const
- bool **has_localized_orbitals** (void) const
- void **print_header** (void)
- void **print_mo_integrals** (void)

Protected Attributes

- int nIsolatedMolecules_

  Number of isolated molecules.
- SharedWavefunction dimer_wavefunction_

  The wavefunction for a dimer (electrons relaxed in the field of monomers)
- SharedIntegralTransform integrals_

  Integral transform object (2- and 4-index transformations)
- bool hasLocalizedOrbitals_

  whether orbitals of the union were localized (or not)
- std::vector< SharedMolecule > l_molecule_

  List of molecules.
- std::vector< SharedBasisSet > l_primary_

  List of primary basis functions per molecule.
- std::vector< SharedBasisSet > l_auxiliary_

  List of auxiliary basis functions per molecule.
- std::vector< SharedWavefunction > l_wfn_

  List of original isolated wavefunctions (electrons unrelaxed)
- std::vector< std::string > l_name_

  List of names of isolated wavefunctions.

- std::vector< int > l_nbf_

  List of basis function numbers per molecule.
- std::vector< int > l_nmo_

  List of numbers of molecular orbitals (MO's) per molecule.
- std::vector< int > l_nso_

  List of numbers of SO's per molecule.
- std::vector< int > l_ndocc_

  List of numbers of doubly occupied orbitals per molecule.
- std::vector< int > l_nvir_

  List of numbers of virtual orbitals per molecule.
- std::vector< int > l_noffs_ao_

  List of basis set offsets per molecule.
- std::vector< double > l_energy_

  List of energies of isolated wavefunctions.
- std::vector< double > l_efzc_

  List of frozen-core energies per isolated wavefunction.
- std::vector< bool > l_density_fitted_

  List of information per wfn whether it was obtained using DF or not.
- std::vector< int > l_nalpha_

  List of numbers of alpha electrons per isolated wavefunction.
- std::vector< int > l_nbeta_

  List of numbers of beta electrons per isolated wavefunction.
- std::vector< int > l_nfrzc_

  List of numbers of frozen-core orbitals per isolated molecule.
- std::vector< SharedLocalizer > l_localizer_

  List of orbital localizers.
- std::vector< std::map< const std::string, SharedMOSpace > > l_mospace_

  List of dictionaries of MO spaces.

9.21.1 Detailed Description

Union of two Wavefunction objects.

The WavefunctionUnion is the union of two unperturbed Wavefunctions.

Notes:

1. Works only for C1 symmetry! Therefore `this->nirrep() = 1`.
2. Does not set `reference_wavefunction_`
3. Sets `oeprop_` for the union of uncoupled molecules
4. Performs Hadamard sums on `H_`, `Fa_`, `Da_`, `Ca_` and `S_` based on uncoupled wavefunctions.
5. Since it is based on shallow copy of the original Wavefunction, it **changes** contents of this wavefunction. Reallocate and copy if you want to keep the original wavefunction.

Warnings:

1. Gradients, Hessians and frequencies are not touched, hence they are **wrong**!
2. Lagrangian (if present) is not touched, hence its **wrong**!

The following variables are *shallow* copies of variables inside the Wavefunction object, that is created for the *whole* molecule cluster:

- basissets_ (DF/RI/F12/etc basis sets)_
- basisset_ (ORBITAL basis set)
- sobasisset_ (Primary basis set for SO integrals)
- AO2SO_ (AO2SO conversion matrix (AO in rows, SO in cols)
- molecule_ (Molecule that this wavefunction is run on)
- options_ (Options object)
- psio_ (PSI file access variables)
- integral_ (Integral factory)
- factory_ (Matrix factory for creating standard sized matrices)
- memory_ (How much memory you have access to)
- nalpha_, nbeta_ (Total alpha and beta electrons)
- nfrzc_ (Total frozen core orbitals)
- doccpi_ (Number of doubly occupied per irrep)
- soccpi_ (Number of singly occupied per irrep)
- frzcpi_ (Number of frozen core per irrep)
- frzvpi_ (Number of frozen virtuals per irrep)
- nalphapi_ (Number of alpha electrons per irrep)
- nbetapi_ (Number of beta electrons per irrep)
- nsopi_ (Number of so per irrep)
- nmopi_ (Number of mo per irrep)
- nso_ (Total number of SOs)
- nmo_ (Total number of MOs)
- nirrep_ (Number of irreps; must be equal to 1 due to symmetry reasons)
- same_a_b_dens_ and same_a_b_orbs_ The rest is altered so that the Wavefunction parameters reflect a cluster of non-interacting (uncoupled, isolated, unrelaxed) molecular electron densities.

9.21.2 Constructor & Destructor Documentation

9.21.2.1 oepdev::WavefunctionUnion::WavefunctionUnion (SharedWavefunction *ref_wfn,* Options & *options*)

Constructor.

Provide wavefunction with molecule containing at least 2 fragments.

Parameters

ref_wfn	- reference wavefunction
options	- Psi4 options

The documentation for this class was generated from the following files:

- oepdev/libutil/wavefunction_union.h
- oepdev/libutil/wavefunction_union.cc

Index