CPPE Documentation

Release 0.0.0

Maximilian Scheurer

Jan 11, 2019

Table of Contents

CPPE has been interfaced to the follow quantum chemical program packages:

- Psi4
- PYSCF
- Q-Chem

CHAPTER 1

C++ API (Doxygen)

enum libcppe::**BorderType**
Values:

rem

redist
{"Electrostatic", {"Electronic", "Nuclear", "Multipoles"}}, {"Polarization", {"Electronic", "Nuclear", "Multipoles"}}}]

std::vector<*Potential*> libcppe::**get_polarizable_sites**(std::vector<*Potential*> *potentials*)

Eigen::Vector3d libcppe::**smat_vec**(**const** Eigen::VectorXd &*mat*, **const** Eigen::Vector3d &*vec*, double *alpha*)

Eigen::VectorXd libcppe::**Tk_tensor**(int *k*, **const** Eigen::Vector3d &*Rij*, std::vector<Eigen::MatrixXi> &*Tk_coeffs*)

Eigen::VectorXd libcppe::**multipole_derivative**(int *k*, int *l*, **const** Eigen::Vector3d &*Rji*, Eigen::VectorXd *Mkj*, std::vector<Eigen::MatrixXi> &*Tk_coeffs*)

int libcppe::**xyz2idx**(int *x*, int *y*, int *z*)

double libcppe::**T**(**const** Eigen::Vector3d &*Rij*, int *x*, int *y*, int *z*, std::vector<Eigen::MatrixXi> &*Cijn*)

std::vector<Eigen::MatrixXi> libcppe::**Tk_coefficients**(int *max_order*)

double libcppe::**factorial**(int *n*)

void libcppe::**make_df**(unsigned *k*, std::vector<double> &*df*)

int libcppe::**trinom**(int *i*, int *j*, int *k*)

std::vector<double> libcppe::**symmetry_factors**(unsigned *k*)

std::vector<double> libcppe::**prefactors**(unsigned *k*)

std::vector<double> libcppe::**prefactors_nuclei**(unsigned *k*)

int libcppe::**multipole_components**(int *k*)

bool libcppe::**sortbysec**(**const** std::pair<int, double> &*a*, **const** std::pair<int, double> &*b*)

struct Atom

#include <molecule.hh>

Public Functions

Atom(int *an*)

Atom(int *an*, double *x*, double *y*, double *z*)

Eigen::Vector3d **get_pos**()

Public Members

int **atomic_number**

int **charge**

double **m_x**

double **m_y**

double **m_z**

struct BorderOptions

#include <pe_options.hh>

Public Members

BorderType **border_type** = {*rem*}

double **rmin** = 2.2

int **nredist** = 1

int **redist_order** = 1

bool **redist_pol** = false

class CPPE

#include <libcppe.hh>

Public Functions

CPPE()

~CPPE()

std::vector<*Potential*> **read_potfile**(std::string *potfile_name*)

Private Members

bool **m_gen1int_initialized**

bool **m_pe_initialized**

int **m_nbas**

int **m_nnbas**

int **m_natoms**

class CppeState

#include <cppe_state.hh>

Public Functions

CppeState()

CppeState(*PeOptions* *options*, *Molecule* *mol*, std::ostream &*output_stream* = std::cout)

~CppeState()

void **set_options**(*PeOptions* *options*)

void **set_molecule**(*Molecule* *mol*)

void **set_potentials**(std::vector<*Potential*> *potentials*)

std::vector<*Potential*> **get_potentials**()

PeEnergy &**get_energies**()

void **set_energies**(*PeEnergy* *energy*)

void **calculate_static_energies_and_fields**()

std::vector<double> **get_induced_moments**() **const**

Eigen::VectorXd **get_induced_moments_vec**() **const**

void **update_induced_moments**(Eigen::VectorXd *elec_fields*, bool *elec_only* = false)

size_t **get_polarizable_site_number**()

std::vector<double> **get_static_fields**()

void **print_summary**()

Private Members

Eigen::MatrixXd **m_es_operator**
PE electrostatics operator.

PeEnergy **m_pe_energy**
PE Energy Container.

Molecule **m_mol**
core region molecule

std::vector<*Potential*> **m_potentials**
vector with all site potentials

size_t **m_polarizable_sites**
number of polarizable sites

Eigen::VectorXd **m_nuc_fields**
electric fields from nuclei

Eigen::VectorXd **m_multipole_fields**
electric fields from multipole moments

Eigen::VectorXd **m_induced_moments**
Vector with induced moments.

PeOptions **m_options**

std::ostream &**m_output_stream** = std::cout
Output stream for printing.

bool **m_make_guess** = true

class InducedMoments
#include <electric_fields.hh>

Public Functions

InducedMoments(std::vector<*Potential*> *potentials*, *PeOptions* *options*)

~InducedMoments()

void **compute**(**const** Eigen::VectorXd &*total_fields*, Eigen::VectorXd &*induced_moments*, bool *make_guess*, std::ostream &*output_stream* = std::cout)

Eigen::VectorXd **compute**(Eigen::VectorXd &*total_fields*, bool *make_guess*)
overloads the compute method for induced moments and returns a copy of the induced moments vector

Private Members

std::vector<*Potential*> **m_potentials**
vector with all site potentials

std::vector<*Potential*> **m_polsites**
vector with all potentials of polarizable sites

size_t **m_n_polsites**
number of polarizable sites

PeOptions **m_options**

struct Molecule : **public** std::vector<*Atom*>
#include <molecule.hh>

Public Functions

Eigen::Vector3d **get_atom_position**(int *atom*)

~Molecule()

Molecule &**operator=**(**const** *Molecule*&)

class Multipole
#include <multipole.hh>

Public Functions

Multipole(unsigned *k*)

~Multipole()

void **add_value**(double *val*)

void **remove_trace**()

std::vector<double> &**get_values**()

Eigen::VectorXd **get_values_vec**()

Public Members

unsigned **m_k**

Private Members

std::vector<double> **m_values**

class MultipoleExpansion
#include <multipole_expansion.hh>

Public Functions

MultipoleExpansion(*Molecule core*, std::vector<*Potential*> *potentials*)

~MultipoleExpansion()

double **calculate_interaction_energy**()

Private Members

Molecule **m_mol**
core region molecule

std::vector<*Potential*> **m_potentials**
vector with all site potentials

class MultipoleFields
#include <electric_fields.hh>

Public Functions

MultipoleFields(std::vector<*Potential*> *potentials*)

~MultipoleFields()

Eigen::VectorXd **compute**(bool *damp* = false)

Private Members

std::vector<*Potential*> **m_potentials**
vector with all site potentials

std::vector<*Potential*> **m_polsites**
vector with all potentials of polarizable sites

size_t **m_n_polsites**
number of polarizable sites

class NuclearFields
#include <electric_fields.hh>

Public Functions

NuclearFields(*Molecule mol*, std::vector<*Potential*> *potentials*)

~NuclearFields()

Eigen::VectorXd **compute**(bool *damp_core* = false)

Private Members

std::vector<*Potential*> **m_potentials**
vector with all site potentials

std::vector<*Potential*> **m_polsites**
vector with all potentials of polarizable sites

size_t **m_n_polsites**
number of polarizable sites

Molecule **m_mol**
core region molecule

struct PeEnergy
#include <pe_energies.hh> PE Energy Container

Public Functions

PeEnergy()

double **get**(std::string *energy_string*)
returns energy contribution from given string

void **set**(std::string *energy_string*, double *energy*)
sets the energy titled energy_string to energy

Return void

Parameters

- `energy_string`: name of the energy contribution
- `energy`: value

double **get_total_energy**()
returns the total PE energy

Private Members

std::vector<*PeEnergyContribution*> **m_energies**

struct PeEnergyContribution
#include <pe_energies.hh> PE Energy Contribution

Public Functions

PeEnergyContribution(std::string *cat*, std::string *name*, double *val*)

Public Members

std::string **m_category**
category of the energy, either "Electrostatic" or "Polarization"

std::string **m_name**
name of the energy, "Electronic", "Nuclear", or "Multipoles"

double **m_value**
energy value

struct PeOptions
#include <pe_options.hh>

Public Members

std::string **potfile** = {"potential.pot"}

int **print_level** = 1

bool **damp_induced** = false

bool **damp_multipoles** = false

bool **damp_core** = false

double **damp_coeff_ind** = 2.1304

double **damp_coeff_mult** = 2.1304

double **damp_coeff_core** = 2.1304

bool **zero_pol** = false

bool **zero_mul** = false

int **zero_mul_order** = 1

int **induced_thresh** = 8

bool **do_diis** = true

int **diis_maxiter** = 50

double **diis_start_norm** = 1.0

bool **pe_border** = false

BorderOptions **border_options** = {}

class Polarizability

#include <multipole.hh>

Public Functions

Polarizability()

~Polarizability()

void **add_value**(double *val*)

Eigen::VectorXd **get_values_vec**()

std::vector<double> &**get_values**()

Private Members

std::vector<double> **m_values**

class Potential

#include <multipole.hh>

Public Functions

Potential(double *x*, double *y*, double *z*, int *idx*)

~Potential()

void **add_multipole**(*Multipole* *mul*)

void **add_polarizability**(*Polarizability* *pol*)

void **add_exclusion**(int *excl*)

bool **excludes_site**(int *other_site*)

std::vector<int> &**get_exclusions**()

std::vector<*Multipole*> &**get_multipoles**()

std::vector<*Polarizability*> &**get_polarizabilities**()

bool **is_polarizable**()

Eigen::Vector3d **get_site_position**()

Public Members

double **m_x**

double **m_y**

double **m_z**

int **index**

Private Members

std::vector<*Multipole*> **m_multipoles**

std::vector<*Polarizability*> **m_polarizabilities**

std::vector<int> **m_exclusions**

class PotfileReader

#include <potfile_reader.hh>

Public Functions

PotfileReader(std::string *potfile_name*)

~PotfileReader()

std::vector<*Potential*> **read**()

Private Members

std::string **m_potfile**

class PotManipulator

#include <pot_manipulation.hh>

Public Functions

PotManipulator(std::vector<*Potential*> *potentials*, *Molecule* *mol*, std::ostream &*output_stream* = std::cout)

~PotManipulator()

std::vector<*Potential*> **manipulate**(*PeOptions* &*pe_options*)

Private Members

std::vector<*Potential*> **m_potentials**

Molecule **m_mol**

std::ostream &**m_output_stream**

struct Site

Public Members

double **x**

double **y**

double **z**

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Index

L