Contents

1 Installation 1

2 Documentation 3
  2.1 Optimiser ......................................................... 3
    2.1.1 In non-redundant internal coordinates (Zmatrix) ............ 3
    2.1.1.1 optimise .................................................. 3
  2.2 Interfaces for electronic structure calculation .................. 4
    2.2.1 Generic Interface ........................................... 4
      2.2.1.1 calculate .............................................. 5
    2.2.2 Molcas Interface ........................................... 5
      2.2.2.1 calculate .............................................. 5
      2.2.2.2 generate_input_file .................................. 6
      2.2.2.3 parse_output .......................................... 7
    2.2.3 Molpro Interface ........................................... 7
      2.2.3.1 calculate .............................................. 7
      2.2.3.2 generate_input_file .................................. 8
      2.2.3.3 parse_output .......................................... 8
For the installation checkout the git repository and execute the following command:

```bash
pip install .
```
2.1 Optimiser

2.1.1 In non-redundant internal coordinates (Zmatrix)

\texttt{optimise}(zmolecule, hamiltonian, basis[, ...])

Optimize a molecule.

\texttt{optimise}

\texttt{chemopt.zmat_optimisation.optimise}(zmolecule, hamiltonian, basis, symbols=None, md_out=None, el_calc_dir=None, molden_out=None, etol=1e-06, gtol=0.0006, max_iter=100, backend='molcas', charge=0, multiplicity=1, num_procs=None, mem_per_proc=None, start_orb=None, coord_fmt='.4f', **kwargs)

Optimize a molecule.

\textbf{Parameters}

- \texttt{zmolecule} (chemcoord.Zmat) –
- \texttt{hamiltonian} (str) – The hamiltonian to use for calculating the electronic energy. The allowed values are {'SCF', 'RASSCF', 'CCSD', 'CASPT2', 'MP2', 'B3LYP', 'CCSD(T)'}.
- \texttt{basis} (str) – The basis set to use for calculating the electronic energy.
- \texttt{symbols} (list) – A list of tuples. Each tuple consists of a sympy symbolic expression and a starting value. An example is: [(\textit{x}, 1.1), (\textit{alpha}, 120)]. Has exactly the same format as the multi-parameter substitution in sympy.
- \texttt{el_calc_dir} (str) – Specify the input filename for electronic calculations. If it is None, the filename of the calling python script is used (With the suffix .inp instead of .py) and the files for the electronic calculations will reside in their own directory.
• `md_out (str)` – Specify the output filename for chemopt output files. If it is None, the filename of the calling python script is used (With the suffix `.md` instead of `.py`). The output will be `os.path.splitext(inputfile)[0] + '.md'`.

• `molden_out (str)` – Specify the output filename for the molden file from a geometry optimisation. If it is None, the filename of the calling python script is used (With the suffix `.molden` instead of `.py`). The output will be `os.path.splitext(inputfile)[0] + '.molden'`.

• `backend (str)` – Specify which QM program suite should be used. Allowed values are `{'molpro', 'molcas'}`, the default is `molcas`.

• `charge (int)` – The overall charge of the molecule. The default is 0.

• `title (str)` – The title to be printed in input and output.

• `multiplicity (int)` – The spin multiplicity. The default is 1.

• `etol (float)` – Convergence criterium for the energy.

• `gtol (float)` – Convergence criterium for the gradient.

• `max_iter (int)` – Maximum number of iterations. The default is ‘100’.

• `num_procs (int)` – The number of processes to spawn.

• `mem_per_proc (str)` – Memory per process. This is a string with a number and a unit like ‘800 MB’. SI and binary prefixes are supported. Uses the `datasize` library for parsing.

• `start_orb (str)` – Path to an orbital file, if starting orbitals should be used.

• `coord_fmt (str)` – A string as float formatter for the coordinates in the output file of chemopt. The default is ‘.4f’

Returns

A list of dictionaries. The last one is the optimised structure. The keys of each dictionary depend on the used optimisation. In any case each dictionary has at least two keys:

• ‘energy’: The energy in Hartree.

• ‘structure’: The Zmatrix.

If `symbols` was `None` a generic optimisation was performed and the following keys are available:

• ‘grad_energy’: The energy gradient (‘grad_energy’) in internal coordinates. The units are Hartree /Angstrom for bonds and Hartree / radians for angles and dihedrals.

If `symbols` was not `None` an optimisation with reduced degrees of freedom was performed and the following keys are available:

• ‘symbols’: A list of tuples containing the symbol and its value.

Return type list

2.2 Interfaces for electronic structure calculation

2.2.1 Generic Interface
calculate(molecule, hamiltonian, basis[,...]) Calculate the energy of a molecule.

2.2.1.1 calculate

chemopt.interface.generic.calculate(molecule, hamiltonian, basis, el_calc_input=None, backend=None, charge=0, forces=False, title=", multiplicity=1, **kwargs)

Calculate the energy of a molecule.

Parameters

- **molecule** (chemcoord.Cartesian or chemcoord.Zmat or str) – If it is a string, it has to be a valid xyz-file.
- **hamiltonian** (str) – The hamiltonian to use for calculating the electronic energy. The allowed values are {‘SCF’, ‘RASSCF’, ‘CCSD’, ‘CASPT2’, ‘MP2’, ‘B3LYP’, ‘CCSD(T)’}.
- **basis** (str) – The basis set to use for calculating the electronic energy.
- **el_calc_input** (str) – Specify the input filename for electronic calculations. If it is None, the filename of the calling python script is used (With the suffix .inp instead of .py). The output will be os.path.splitext(inputfile)[0] + '.inp'.
- **backend** (str) – Specify which QM program suite should be used. Allowed values are {‘molpro’, ‘molcas’}, the default is ‘molcas’.
- **charge** (int) – The overall charge of the molecule. The default is 0.
- **forces** (bool) – Specify if energy gradients should be calculated. The default is False.
- **title** (str) – The title to be printed in input and output.
- **multiplicity** (int) – The spin multiplicity. The default is 1.

Returns A dictionary with at least the keys 'structure' and 'energy' which contains the energy in Hartree. If forces were calculated, the key 'gradient' contains the gradient in Hartree / Angstrom.

Return type dict

2.2.2 Molcas Interface

2.2.2.1 calculate

chemopt.interface.molcas.calculate(molecule, hamiltonian, basis, molcas_exe=None, el_calc_input=None, sym_group=None, charge=0, forces=False, title=", multiplicity=1, start_orb=None, num_procs=None, mem_per_proc=None)

Calculate the energy of a molecule using Molcas.

Parameters

- **el_calc_input** (str) – Specify the input filename for electronic calculations. If it is
None, the filename of the calling python script is used (With the suffix .inp instead of .py). The output will be os.path.splitext(inputfile)[0] + '.inp'.

- **molecule (chemcoord.Cartesian or chemcoord.Zmat or str)** – If it is a string, it has to be a valid xyz-file.
- **hamiltonian (str)** – The hamiltonian to use for calculating the electronic energy. The allowed values are {'SCF', 'RASSCF', 'CCSD', 'CASPT2', 'MP2', 'B3LYP', 'CCSD(T)'}.
  But ‘CCSD’ and ‘CCSD(T)’ are not yet implemented.
- **basis (str)** – The basis set to use for calculating the electronic energy.
- **molcas_exe (str)** – Specify the command to invoke molcas. The default is ‘molcas’.
- **charge (int)** – The overall charge of the molecule. The default is 0.
- **forces (bool)** – Specify if energy gradients should be calculated. The default is False.
- **title (str)** – The title to be printed in input and output.
- **multiplicity (int)** – The spin multiplicity. The default is 1.
- **start_orb (str)** – Path to an orbital file, if starting orbitals should be used.
- **num_procs (int)** – The number of processes to spawn.
- **mem_per_proc (str)** – Memory per process. This is a string with a number and a unit like ‘800 MB’. SI and binary prefixes are supported. Uses the datasize library for parsing.

**Returns** A dictionary with at least the keys 'structure' and 'energy' which contains the energy in Hartree. If forces were calculated, the key 'gradient' contains the gradient in Hartree / Angstrom.

**Return type** dict

### 2.2.2.2 generate_input_file

```python
cemopt.interface.molcas.generate_input_file(molecule, hamiltonian, basis, el_calc_input, charge=0, forces=False, title='', start_orb=None, sym_group=None, multiplicity=1)
```

Generate a molcas input file.

**Parameters**

- **molecule (chemcoord.Cartesian or chemcoord.Zmat or str)** – If it is a string, it has to be a valid xyz-file.
- **hamiltonian (str)** – The hamiltonian to use for calculating the electronic energy. The allowed values are {'SCF', 'RASSCF', 'CCSD', 'CASPT2', 'MP2', 'B3LYP', 'CCSD(T)'}.
- **basis (str)** – The basis set to use for calculating the electronic energy.
- **charge (int)** – The overall charge of the molecule. The default is 0.
- **forces (bool)** – Specify if energy gradients should be calculated. The default is False.
- **title (str)** – The title to be printed in input and output.
- **multiplicity (int)** – The spin multiplicity. The default is 1.
• `wfn_symmetry (int)` – The symmetry of the wavefunction specified with the molpro notation.

Returns molcas input.

Return type `str`

### 2.2.2.3 parse_output

```python
cheompt.interface.molcas.parse_output(output_path)
```

Parse a molcas output file.

**Parameters**

- `output_path (str)` –

**Returns**

A dictionary with at least the keys 'structure' and 'energy' which contains the energy in Hartree. If forces were calculated, the key 'gradient' contains the gradient in Hartree / Angstrom.

Return type `dict`

### 2.2.3 Molpro Interface

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>calculate</code></td>
<td>Calculate the energy of a molecule using Molpro.</td>
</tr>
<tr>
<td><code>generate_input_file</code></td>
<td>Generate a molpro input file.</td>
</tr>
<tr>
<td><code>parse_output</code></td>
<td>Parse a molpro output file.</td>
</tr>
</tbody>
</table>

#### 2.2.3.1 calculate

```python
chemopt.interface.molpro.calculate(molecule, hamiltonian, basis, molpro_exe=None, el_calc_input=None, charge=0, forces=False, title='-', multiplicity=1, wfn_symmetry=1, num_procs=None, mem_per_proc=None)
```

Calculate the energy of a molecule using Molpro.

**Parameters**

- `el_calc_input (str)` – Specify the input filename for electronic calculations. If it is None, the filename of the calling python script is used (With the suffix .inp instead of .py). The output will be os.path.splitext(inputfile)[0] + '.inp'.
- `molecule (chemcoord.Cartesian or chemcoord.Zmat or str)` – If it is a string, it has to be a valid xyz-file.
- `hamiltonian (str)` – The hamiltonian to use for calculating the electronic energy. The allowed values are {'SCF', 'RASSCF', 'CCSD', 'CASPT2', 'MP2', 'B3LYP', 'CCSD(T)'}.
  But 'RASSCF' and 'CASPT2' not yet implemented.
- `basis (str)` – The basis set to use for calculating the electronic energy.
- `molpro_exe (str)` – Specify the command to invoke molpro. The default is 'molpro'.
- `charge (int)` – The overall charge of the molecule. The default is 0.
- `forces (bool)` – Specify if energy gradients should be calculated. The default is False.
- `title (str)` – The title to be printed in input and output.
• **multiplicity** *(int)* – The spin multiplicity. The default is 1.

• **wfn_symmetry** *(int)* – The symmetry of the wavefunction specified with the molpro notation.

• **num_procs** *(int)* – The number of processes to spawn.

• **mem_per_proc** *(str)* – Memory per process. This is a string with a number and a unit like ‘800 MB’. SI and binary prefixes are supported. Uses the datasize library for parsing.

**Returns** A dictionary with at least the keys 'structure' and 'energy' which contains the energy in Hartree. If forces were calculated, the key 'gradient' contains the gradient in Hartree / Angstrom.

**Return type** *dict*

### 2.2.3.2 generate_input_file

chemopt.interface.molpro.generate_input_file *(molecule, hamiltonian, basis, charge=0, forces=False, title='', multiplicity=1, wfn_symmetry=1, mem_per_proc=None)*

Generate a molpro input file.

**Parameters**

• **molecule** *(chemcoord.Cartesian or chemcoord.Zmat or str)* – If it is a string, it has to be a valid xyz-file.

• **hamiltonian** *(str)* – The hamiltonian to use for calculating the electronic energy. The allowed values are {'SCF', 'RASSCF', 'CCSD', 'CASPT2', 'MP2', 'B3LYP', 'CCSD(T)'}.  

• **basis** *(str)* – The basis set to use for calculating the electronic energy.

• **charge** *(int)* – The overall charge of the molecule. The default is 0.

• **forces** *(bool)* – Specify if energy gradients should be calculated. The default is False.

• **title** *(str)* – The title to be printed in input and output.

• **multiplicity** *(int)* – The spin multiplicity. The default is 1.

• **wfn_symmetry** *(int)* – The symmetry of the wavefunction specified with the molpro notation.

• **mem_per_proc** *(str)* – Memory per process. This is a string with a number and a unit like ‘800 MB’. SI and binary prefixes are supported. Uses the datasize library for parsing.

**Returns** Molpro input.

**Return type** *str*

### 2.2.3.3 parse_output

chemopt.interface.molpro.parse_output *(output_path)*

Parse a molpro output file.

**Parameters**

output_path *(str)*

**Returns** A dictionary with at least the keys 'structure' and 'energy' which contains the energy in Hartree. If forces were calculated, the key 'gradient' contains the gradient in Hartree / Angstrom.
Return type: `dict`
Index

C
calculate() (in module chemopt.interface.generic), 5
calculate() (in module chemopt.interface.molcas), 5
calculate() (in module chemopt.interface.molpro), 7

G
generate_input_file() (in module chemopt.interface.molcas), 6
generate_input_file() (in module chemopt.interface.molpro), 8

O
optimise() (in module chemopt.zmat_optimisation), 3

P
parse_output() (in module chemopt.interface.molcas), 7
parse_output() (in module chemopt.interface.molpro), 8