MultirefPredict Documentation

MultirefPredict

Nov 03, 2022

Contents:

Automated workflow to predict multireference character of molecules in quantum chemistry calculation

CHAPTER 1

Program Execution

A simple example of MultirefPredict's capabilities of as follows:

```
>>> import MultirefPredict
>>> import qcelemental
>>> mol = qcelemental.models.Molecule.from_data("""
O                  0.000000000000     0.000000000000    -0.068516245955
H                  0.000000000000    -0.790689888800     0.543701278274
H                  0.000000000000     0.790689888800     0.543701278274
""")
```

Here we just built a molecule object in qcelemental format. Various multireference diagnostics can be calculated for this molecule with the `computeDiagnostic` syntax of the diagnostic object, which is initialized with the `diagnostic_factory` function.

```
>>> b1 = MultirefPredict.diagnostic_factory("B1",molecule=mol, molname="water",␣
↪record=False).computeDiagnostic()
```

`b1` is the returned diagnostic value

```
>>> b1
0.006334860365228678
```

The I/O of the backend quantum chemistry package is totally hidden from the user. If one would like to get detailed information about the quantum chemistry calculation, it can be easily achieved by setting the `record` keyword to `True`.

```
>>> b1 = MultirefPredict.diagnostic_factory("B1",molecule=mol, molname="water",␣
↪record=True).computeDiagnostic()
```

Then the jason files recording the quantum chemistry calculations associated with this diagnostic calculation are automatically dumped in the working directory

```
>>> ls
B1_water_b1lyp_H.json
```

(continues on next page)

(continued from previous page)

```
B1_water_b1lyp_O.json
B1_water_b1lyp_whole.json
B1_water_blyp_H.json
B1_water_blyp_O.json
B1_water_blyp_whole.json
```

These files are in the qcelemental result format. For example:

```
>>> with open('B1_water_b1lyp_H.json') as f:
        data = json.load(f)
            print(json.dumps(data,indent=4))
{
    "molecule": {
            "symbols": [
                "H"
            ],
            "geometry": [
                 0.0,
                 0.0,
                 0.0
             ],
             "molecular_charge": 0.0,
             "molecular_multiplicity": 2
     },
     "driver": "energy",
     "model": {
          "method": "b1lyp",
           "basis": "6-31g"
      },
      ...
}
```

CHAPTER 2

Backends

Currently available compute backends for single results are as follow:

- Psi4
- TeraChem

All backends are driven by QCEngine

2.1 Installation

Currently MultirefPredict can only be installed from source code

1. Prerequisite: have Anaconda or miniconda installed on your system
2. Clone MultirefPredict source from gibhub

```
git clone https://github.com/hjkgrp/MultirefPredict.git
```

3. Go to the folder root folder for MultirefPredict, create the conda environment from yaml file

```
cd dev_tools/conda_envs
conda env create -f test_env.yaml
```

> **Important:** It is highly recommended to install the packages needed by creating environment from this yaml file. This help make sure that the version of package installed (e.g. psi4) is the same as the one tested by the developer. Installing the required packages from other distributions may result in unexpected behavior

4. Now you have created an environment called **test** for with the prerequisite packages for running MultirefPredict. You can rename the environment as you like by cloning this environment to a new name and remove the origional one. For example, to rename it to **multiref**:

```
conda create --name multiref --clone test
conda remove --name test --all
```

5. Activate the conda environment you just created. Go back to the root directory of MultirefPredict (where the setup.py file locates). Local install with pip

```
cd Your_root_path_to_MultirefPrect
pip install -e .
```

2.2 Usage

This page is still under construction

2.3 Examples

This page is still under construction