[2]p()*1/2-ˆ [1]p1-ˆ TJ
3 10000pt

0.00.5em

0.0.00.5em

0.0.0.00.5em

0.0.0.0.00.5em

0.0.0.0.0.00.5em

FC-DFT

Release 1.0.0

Jun-Hyeong Kim

Jun 12, 2025

Contents

1 Installation **1**

1.1 Requirements 1

1.2 How to install 1

2 Examples **3**

2.1 Wide-Band Limit 3

2.2 Non-Linear Poisson-Boltzmann Solvation Model 3

2.3 Geometry Optimization 4

2.4 Thermochemistry 5

3 About **7**

3.1 Developers 7

3.2 Citation 7

3.3 Bug reports and feature requests 7

4 Indices and tables **9**

Chapter 1

Installation

This page explains how to install FC-DFT code.

1.1 Requirements

- GeomeTRIC
- PySCF
- PyAMG
- PyAMGCL
- GPU4PySCF (optional)

1.2 How to install

Download the lastest version of FC-DFT from the repository:

1. Clone the repository:

   ```
   $ git clone https://github.com/Yang-Laboratory/FC-DFT.git
   ```

2. Change the directory to the cloned repository:

   ```
   $ cd FC-DFT
   ```

3. Install the package using *pip*:

   ```
   $ pip install .
   ```

4. Change directory to *$PYTHONPATH/fcdft/lib* and create build directory.
5. Go into *build* and compile the C shared libraries by *cmake ..* and *make*.

Chapter 2

Examples

This chapter introduces brief exmaples of running FC-DFT and Poisson-Boltzmann solvation calculations

2.1 Wide-Band Limit

FC-DFT uses the wide-band limit (WBL) approximation. In particular, the WBL-Molecule approximation is adopted for simple but powerful implementation. WBL-Molecule requires users to provide the imaginary part of the self-energy in WBLMolecule object. Currently, spin-restricted version of FC-DFT (fcdft.wbl.rks) is supported. Below is a sample code, where the self-energy of 0.01 eV is attached to the sulfur atom of methanethiol with 25.95 electrons.

```
>>> from pyscf import gto
>>> from pyscf.dft import RKS
>>> mol = gto.M(atom='''
...         C       -1.718553971      -0.000000250      -0.626147715
...         H       -2.739245971      -0.008907250      -0.227127715
...         H       -1.200493971      -0.879491250      -0.227127715
...         H       -1.215921971       0.888398750      -0.227127715
...         S       -1.718553971      -0.000000250      -2.396147715
...         H       -2.150082583       0.805150681      -2.710667448''',
...     charge=0, basis='6-31g**')
>>> mf = RKS(mol, xc='pbe')
>>> mf.kernel()
>>> from fcdft.wbl.rks import *
>>> wblmf = WBLMolecule(mf, broad=0.01, nelectron=25.95)
>>> wblmf.kernel()
```

2.2 Non-Linear Poisson-Boltzmann Solvation Model

We provide the Poisson-Boltzmann solver for general purpose. fcdft.solvent.pbe module supports usual solvation energy calculations as what polarizable continuum model does. To do so, a few attributes of PBE needs to be controlled since it was originally intended to solve the electrostatic potential under the Gouy-Chapman-Stern theory:

```
>>> from pyscf import gto
>>> from pyscf.dft import RKS
>>> mol = gto.M(atom='''
...         O        0.152427064       0.959723218      -2.275350162
```

(continues on next page)

(continued from previous page)

```
...          H       0.152427064     1.719060218     -1.679307162
...          H       0.152427064     0.200386218     -1.679307162''',
...     charge=0, basis='6-31g**')
>>> mf = RKS(mol, xc='b3lyp')
>>> from fcdft.solvent.pbe import *
>>> cm = PBE(mol, cb=1.0, length=15, ngrids=41)
>>> cm.eps = 78.3553
>>> cm.atom_bottom = 'center'
>>> cm.nelectron = mol.nelectron
>>> cm.bias = 0.0e0
>>> cm.surf = 0.0e0
>>> solmf = pbe_for_scf(mf, cm)
>>> solmf.kernel()
```

The following example introduces how to run the Poisson-Boltzmann solver under the Gouy-Chapman-Stern boundary values:

```
>>> from pyscf import gto
>>> from pyscf.dft import RKS
>>> mol = gto.M(atom='''
...          C      -1.718553971     -0.000000250     -0.626147715
...          H      -2.739245971     -0.008907250     -0.227127715
...          H      -1.200493971     -0.879491250     -0.227127715
...          H      -1.215921971      0.888398750     -0.227127715
...          S      -1.718553971     -0.000000250     -2.396147715
...          H      -2.150082583      0.805150681     -2.710667448''',
...     charge=0, basis='6-31g**')
>>> mf = RKS(mol, xc='pbe')
>>> mf.kernel()
>>> from fcdft.wbl.rks import *
>>> wblmf = WBLMolecule(mf, broad=0.01, nelectron=25.95)
>>> wblmf.kernel()
>>> dm = wblmf.make_rdm1()
>>> from fcdft.solvent.pbe import *
>>> cm = PBE(mol, cb=1.0, length=15, ngrids=41, stern_sam=3.0)
>>> cm.eps = 78.3553
>>> cm.eps_sam = 2.284
>>> cm._dm = dm
>>> solmf = pbe_for_scf(wblmf, cm)
>>> solmf.kernel()
```

2.3 Geometry Optimization

Our code supports analytic nuclear gradients of FC-DFT as well as the Poisson-Boltzmann solvation model. We have tested geometry optimization using GeomeTRIC, an external geometry optimizer implemented in PySCF:

```
>>> from pyscf.geomopt.geometric_solver import optimize
>>> moleq = optimize(solmf, maxstep=100)
```

2.4 Thermochemistry

We provide a code for numerical Hessian matrix constructed by analytic forces due to the non-Hermitian Hamiltonian resulted by the self-energy. Thermochemical properties can be calculated by utilizing pyscf.hessian.thermo module. Hessian offers three-point (default) and five-point finite difference method for calculating the Hessian matrix. The following code introduces how to obtain thermochemical properties using harmonic_analysis function:

```
>>> from fcdft.hessian numhess import *
>>> hessmf = Hessian(solmf)
>>> hess = hessmf.kernel()
>>> from pyscf.hessian.thermo import harmonic_analysis
>>> freq_info = harmonic_analysis(moleq, hess)
```

Once the quantities are obtained, these can be saved into a molden format as implemented in fcdft.tools.molden:

```
>>> from fcdft.tools.molden import dump_freq
>>> dump_freq(moleq, freq_info, 'freq.molden')
```

Chapter 3

About

Fractional Charge Density Functional Theory (FC-DFT) is a theory that reformulates open quantum systems in terms of the canonical ensemble. The prototype of FC-DFT is linear interpolation FC-DFT (LI-FC-DFT), which enforces the Perdew-Parr-Levy-Balduz (PPLB) condition to DFT calculations through linear interpolation and bypasses the delocalization error.

3.1 Developers

Jun-Hyeong Kim, Duke University

Weitao Yang, Duke University

3.2 Citation

Jun-Hyeong Kim, Dongju Kim, Weitao Yang, and Mu-Hyun Baik. Fractional Charge Density Functional Theory and Its Application to the Electro-inductive Effect. *J. Phys. Chem. Lett.* **2023**, *14*, 3329-3334

Jun-Hyeong Kim and Weitao Yang. Fractional Charge Density Functional Theory Elucidates Electro-Inductive and Electric Field Effects at Electrochemical Interfaces. *To be submitted*

3.3 Bug reports and feature requests

Please create a posting on Issues tab.

Chapter 4

Indices and tables

- genindex
- modindex
- search