GHF

Release 0.1

Xeno De Vriendt

Oct 03, 2019

CONTENTS:

CHAPTER

ONE

RESTRICTED HARTREE FOCK, BY MEANS OF SCF PROCEDURE

This function calculates the RHF energy of a given molecule and the number of occupied orbitals. The molecule has to be created in pySCF: molecule = gto.M(atom = geometry, spin = diff. in alpha and beta electrons, basis = basis set)

number of occupied orbitals: number of doubly occupied orbitals (1 for H_2, 2 for H_4,...)

The function prints the number of iterations and the converged SCF energy, while also returning the energy value for eventual subsequent calculations

ghf.RHF.**RHF**(*molecule*, *number_occupied_orbitals*)

Input is a molecule and the number of occupied orbitals.

Molecules are made in pySCF, eg.:

```
>>> h_2 = gto.M(atom = 'h 0 0 0; h 0 0 1', spin = 0, basis = 'cc-pvdz')
>>> RHF(h_2, 1)
```

prints and returns RHF energy of h_2

CHAPTER

TWO

UNRESTRICTED HARTREE FOCK, BY MEANS OF SCF PROCEDURE

This function calculates the UHF energy for a given molecule and the number of occupied alpha and beta orbitals. The molecule has to be created in pySCF: molecule = gto.M(atom = geometry, spin = diff. in alpha and beta electrons, basis = basis set)

number of occupied orbitals: first the number of occupied alpha orbitals second the number of occupied beta orbitals

There are two extra options added into the function:

- extra_e_coeff: when true, the program will add two extra alpha electrons to the system and calculate when the new system's energy converges. From the last density matrix in this calculation, new coefficients for the alpha and beta orbitals will be calculated and used as new guess density for the original system.
- internal_stability_analysis: when true, an internal stability analysis will be performed. When the wave finction is deemed unstable, new coefficients will be calculated that are closer in value to the stable condition, which is done by varying the Hessian by means of a trial vector. These will then be used as a guess density for the energy calculation.

The function prints the number of iterations and the converged SCF energy, while also returning the energy value for eventual subsequent calculations

ghf.UHF.**UHF**(*molecule*, *occ_a*, *occ_b*, *extra_e_coeff=False*, *internal_stability_analysis=False*)

```
>>> h3 = gto.M(atom = 'h 0 0 0; h 0 0.86602540378 0.5; h 0 0 1', spin = 1, basis␣
↪= 'cc-pvdz')
>>> UHF(h3, 2, 1)
```

```
>>> h3 = gto.M(atom = 'h 0 0 0; h 0 0.86602540378 0.5; h 0 0 1', spin = 1, basis␣
↪= 'cc-pvdz')
>>> UHF(h3, 2, 1, extra_e_coeff=True)
```

```
>>> h3 = gto.M(atom = 'h 0 0 0; h 0 0.86602540378 0.5; h 0 0 1', spin = 1, basis␣
↪= 'cc-pvdz')
>>> UHF(h3, 2, 1, internal_stability_analysis=True)
```

prints and returns UHF energy of h3

CHAPTER

THREE

USEFUL FUNCTIONS FOR SCF PROCEDURE

A number of functions used throughout the UHF and RHF calculations are summarised here.

ghf.SCF_functions.**density_matrix**(*f_matrix*, *occ*, *trans*)

- density() creates a density matrix from a fock matrix and the number of occupied orbitals.
- Input is a fock matrix, the number of occupied orbitals, which can be separate for alpha and beta in case of UHF. And

a transformation matrix X.

ghf.SCF_functions.**get_integrals**(*molecule*)
A function to calculate your integrals & nuclear repulsion with pyscf.

ghf.SCF_functions.**spin**(*occ_a*, *occ_b*, *coeff_a*, *coeff_b*, *overlap*)

Parameters

- **occ_a** – number of occupied alpha orbitals
- **occ_b** – number of occupied beta orbitals
- **coeff_a** – MO coefficients of alpha orbitals
- **coeff_b** – MO coefficients of beta orbitals
- **overlap** – overlap matrix of the molecule

Returns S^2, S_z and spin multiplicity

ghf.SCF_functions.**trans_matrix**(*overlap*)

- Define a transformation matrix X, used to orthogonalize different matrices throughout the calculation.
- Input should be an overlap matrix.

ghf.SCF_functions.**uhf_fock_matrix**(*density_matrix_1*, *density_matrix_2*, *one_electron*, *two_electron*)

- calculate a fock matrix from a given alpha and beta density matrix
- fock alpha if 1 = alpha and 2 = beta and vice versa
- input is the density matrix for alpha and beta, a one electron matrix and a two electron tensor.

ghf.SCF_functions.**uhf_scf_energy**(*density_matrix_a*, *density_matrix_b*, *fock_a*, *fock_b*, *one_electron*)

- calculate the scf energy value from a given density matrix and a given fock matrix for both alpha and beta, so 4 matrices in total
- then calculate the initial electronic energy and put it into an array

- input is the density matrices for alpha and beta, the fock matrices for alpha and beta and lastly a one electron matrix

PYTHON MODULE INDEX

g

INDEX