GHF

Release 0.1

Xeno De Vriendt

Nov 11, 2019

CONTENTS:

CHAPTER

ONE

RESTRICTED HARTREE FOCK, BY MEANS OF SCF PROCEDURE

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

class `ghf.RHF.`**`RHF`**(*molecule*, *number_of_electrons*)

Input is a molecule and the number of electrons.

Molecules are made in pySCF and calculations are performed as follows, eg.: The following snippet prints and returns RHF energy of h_2 and the number of iterations needed to get this value.

```
>>> h_2 = gto.M(atom = 'h 0 0 0; h 0 0 1', spin = 0, basis = 'cc-pvdz')
>>> x = RHF(h_2, 2)
>>> x.get_scf_solution()
Number of iterations: 109
Converged SCF energy in Hartree: -1.9403598392831243 (RHF)
```

`get_last_dens`()

Returns the last density matrix of the converged solution.

Returns The last density matrix.

`get_last_fock`()

Returns the last fock matrix of the converged solution.

Returns The last Fock matrix.

`get_mo_coeff`()

Returns mo coefficients of the converged solution.

Returns The mo coefficients

`get_one_e`()

Returns The one electron integral matrix: T + V

`get_ovlp`()

Returns The overlap matrix

`get_scf_solution`()

Prints the number of iterations and the converged scf energy.

Returns The converged scf energy.

`get_two_e`()

Returns The electron repulsion interaction tensor

`nuc_rep`()

Returns The nuclear repulsion value

scf()

Performs a self consistent field calculation to find the lowest RHF energy.

Returns number of iterations, scf energy, mo coefficients, last density matrix, last fock matrix

CHAPTER

TWO

UNRESTRICTED HARTREE FOCK, BY MEANS OF SCF PROCEDURE

This class is used to calculate the UHF energy for a given molecule and the number of electrons of that molecule. Several options are available to make sure you get the lowest energy from your calculation, as well as some usefull functions to get intermediate values such as MO coefficients, density and fock matrices.

class ghf.UHF.**UHF**(*molecule*, *number_of_electrons*)

Input is a molecule and the number of electrons.

Molecules are made in pySCF and calculations are performed as follows, eg.: The following snippet prints and returns UHF energy of h_3 and the number of iterations needed to get this value.

For a normal scf calculation your input looks like the following example:

```
>>> h3 = gto.M(atom = 'h 0 0 0; h 0 0.86602540378 0.5; h 0 0 1', spin = 1, basis␣
↪= 'cc-pvdz')
>>> x = UHF(h3, 3)
>>> x.get_scf_solution()
Number of iterations: 62
Converged SCF energy in Hartree: -1.5062743202681235 (UHF)
<S^2> = 0.7735672504295973, <S_z> = 0.5, Multiplicity = 2.023430009098014
```

extra_electron_guess()

This method adds two electrons to the system in order to get coefficients that can be used as a better guess for the scf procedure. This essentially forces the system into it's <S_z> = 0 state.

To perform a calculation with this method, you will have to work as follows:

```
>>> h4 = gto.M(atom = 'h 0 0 0; h 1 0 0; h 0 1 0; h 1 1 0' , spin = 2, basis␣
↪= 'cc-pvdz')
>>> x = UHF(h4, 4)
>>> guess = x.extra_electron_guess()
>>> x.get_scf_solution(guess)
Number of iterations: 74
Converged SCF energy in Hartree: -2.0210882477030547 (UHF)
<S^2> = 1.0565277001056579, <S_z> = 0.0, Multiplicity = 2.2860688529487976
```

Returns A new guess matrix to use for the scf procedure.

get_last_dens()

Gets the last density matrix of the converged solution. Alpha density in the first matrix, beta density in the second.

Returns The last density matrix.

get_last_fock()

Gets the last fock matrix of the converged solution. Alpha Fock matrix first, beta Fock matrix second.

Returns The last Fock matrix.

get_mo_coeff()
Gets the mo coefficients of the converged solution. Alpha coefficients in the first matrix, beta coefficients in the second.

Returns The mo coefficients

get_one_e()

Returns The one electron integral matrix: T + V

get_ovlp()

Returns The overlap matrix

get_scf_solution(*guess=None*)
Prints the number of iterations and the converged scf energy. Also prints the expectation value of S_z, S^2 and the multiplicity.

Returns The converged scf energy.

get_two_e()

Returns The electron repulsion interaction tensor

nuc_rep()

Returns The nuclear repulsion value

scf(*initial_guess=None*)
Performs a self consistent field calculation to find the lowest UHF energy.

Parameters **initial_guess** – A tuple of an alpha and beta guess matrix. If none, the core hamiltonian will be used.

Returns The scf energy, number of iterations, the mo coefficients, the last density and the last fock matrices

stability()
Performing a stability analysis checks whether or not the wave function is stable, by checking the lowest eigen- value of the Hessian matrix. If there's an instability, the MO's will be rotated in the direction of the lowest eigenvalue. These new MO's can then be used to start a new scf procedure.

To perform a stability analysis, use the following syntax:

```
>>> h4 = gto.M(atom = 'h 0 0 0; h 1 0 0; h 0 1 0; h 1 1 0' , spin = 2, basis␣
↪= 'cc-pvdz')
>>> x = UHF(h4, 4)
>>> guess = x.stability()
>>> x.get_scf_solution(guess)
There is an internal instability in the UHF wave function.
Number of iterations: 78
Converged SCF energy in Hartree: -2.0210882477030716 (UHF)
<S^2> = 1.056527700105677, <S_z> = 0.0, Multiplicity = 2.2860688529488145
```

Returns New and improved MO's.

CHAPTER

THREE

REAL GENERALISED HARTREE FOCK, BY MEANS OF SCF PROCEDURE

This class creates a generalised Hartree-Fock object which can be used for scf calculations. Different initial guesses are provided as well as the option to perform a stability analysis. The molecule has to be created in pySCF: molecule = gto.M(atom = geometry, spin = diff. in alpha and beta electrons, basis = basis set)

class `ghf.real_GHF.`**`RealGHF`**(*molecule*, *number_of_electrons*)

Input is a molecule and the number of electrons.

Molecules are made in pySCF and calculations are performed as follows, eg.: The following snippet prints and returns UHF energy of h3 and the number of iterations needed to get this value.

For a normal scf calculation your input looks like the following example:

```
>>> h3 = gto.M(atom = 'h 0 0 0; h 0 0.86602540378 0.5; h 0 0 1', spin = 1, basis␣
↪= 'cc-pvdz')
>>> x = RealGHF(h3, 3)
>>> x. get_scf_solution()
Number of iterations: 82
Converged SCF energy in Hartree: -1.5062743202607725 (Real GHF)
```

`get_last_dens`()

Gets the last density matrix of the converged solution.

Returns The last density matrix.

`get_last_fock`()

Gets the last fock matrix of the converged solution.

Returns The last Fock matrix.

`get_mo_coeff`()

Gets the mo coefficients of the converged solution.

Returns The mo coefficients

`get_one_e`()

Returns The one electron integral matrix: T + V

`get_ovlp`()

Returns The overlap matrix

`get_scf_solution`(*guess=None*)

Prints the number of iterations and the converged scf energy.

Returns The converged scf energy.

`get_two_e`()

Returns The electron repulsion interaction tensor

nuc_rep()

Returns The nuclear repulsion value

random_guess()

A function that creates a matrix with random values that can be used as an initial guess for the SCF calculations.

To use this guess:

```
>>> h3 = gto.M(atom = 'h 0 0 0; h 0 0.86602540378 0.5; h 0 0 1', spin = 1,␣
↪basis = 'cc-pvdz')
>>> x = RealGHF(h3, 3)
>>> guess = x.random_guess()
>>> x.get_scf_solution(guess)
:return: A random hermitian matrix.
```

scf(*guess=None*)

This function performs the SCF calculation by using the generalised Hartree-Fock formulas. Since we're working in the real class, all values throughout are real. For complex, see the "complex_GHF" class. :param guess: The initial guess to start the calculation. Different options are integrated within the class. If no guess is specified, the core hamiltonian will be used. :return: The scf energy, number of iterations, the mo coefficients, the last density and the last fock matrices.

stability()

Performing a stability analysis checks whether or not the wave function is stable, by checking the lowest eigen- value of the Hessian matrix. If there's an instability, the MO's will be rotated in the direction of the lowest eigenvalue. These new MO's can then be used to start a new scf procedure.

To perform a stability analysis, use the following syntax, this will continue the analysis until there is no more instability:

```
>>> h4 = gto.M(atom = 'h 0 0 0; h 1 0 0; h 0 1 0; h 1 1 0' , spin = 2, basis␣
↪= 'cc-pvdz')
>>> x = RealGHF(h4, 4)
>>> x.scf()
>>> guess = x.stability()
>>> while x.instability:
>>>     new_guess = x.stability()
>>>     x.get_scf_solution(new_guess)
```

Returns New and improved MO's.

unitary_rotation_guess()

A function that creates an initial guess matrix by performing a unitary transformation on the core Hamiltonian matrix.

To use this guess:

```
>>> h3 = gto.M(atom = 'h 0 0 0; h 0 0.86602540378 0.5; h 0 0 1', spin = 1,␣
↪basis = 'cc-pvdz')
>>> x = RealGHF(h3, 3)
>>> guess = x.unitary_rotation_guess()
>>> x.get_scf_solution(guess)
:return: A rotated guess matrix.
```

CHAPTER

FOUR

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.`**`expand_matrix`**(*matrix*)

Parameters **`matrix`** –

Returns a matrix double the size, where blocks of zero's are added top right and bottom left.

`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.`**`spin_blocked`**(*block_1*, *block_2*, *block_3*, *block_4*)

When creating the blocks of the density or fock matrix separately, this function is used to add them together, and create the total density or Fock matrix in spin Blocked notation. :return: a density matrix in the spin-blocked notation

`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.

CHAPTER

FIVE

TESTING THE RHF AND UHF METHODS

Simple tests to check whether or not the functions return the correct value.

`ghf.tests.test_auth.`**`test_RHF`**`()`

test_RHF will test whether or not the RHF method returns the wanted result. The accuracy is 10^11.

`ghf.tests.test_auth.`**`test_UHF`**`()`

test_UHF will test the regular UHF method, by checking whether or not it returns the expected result. The accuracy is 10^-6.

`ghf.tests.test_auth.`**`test_extra_e`**`()`

test_extra_e will test the UHF method, with the added option of first adding 2 electrons to the system and using those coefficients for the actual system, by checking whether or not it returns the expected result. The accuracy is 10^-6.

`ghf.tests.test_auth.`**`test_stability`**`()`

test_stability will test the UHF method, with stability analysis, by checking whether or not it returns the expected result. The accuracy is 10^-6.

PYTHON MODULE INDEX

INDEX