sphinx-example Documentation

Release

Firstname Lastname

Feb 25, 2018

How to use

SlowQuant is a molecular quantum chemistry program written in python. Even the computational demanding parts are written in python, so it lacks speed, thus the name SlowQuant. The program is run as:

```
python SlowQuant.py MOLECULE SETTINGS
```

As a ready to run example:

```
python SlowQuant.py H2O.csv settingExample.csv
```

SlowQuant have the following requirements:

- Python 3.5 or above
- numpy 1.13.1
- scipy 0.19.1
- numba 0.34.0
- cython 0.25.2
- gcc 5.4.0

CHAPTER 1

Installation and Running

How to install SlowQaunt, and run the program.

1.1 Installation Ubuntu

This installation guide is made for Ubuntu.

Python 3.6 can be installed with conda running the following command lines:

```
wget https://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.sh
```

Then run:

```
bash Miniconda3-latest-Linux-x86_64.sh
```

The python packages can now be installed by:

```
conda install numpy scipy numba cython
```

The gcc compiler can be installed by:

```
sudo apt-get install gcc
```

The SlowQuant program can be downloaded by:

```
git clone https://github.com/Melisius/SlowQuant.git
```

Inside the SlowQuant folder run:

```
python setup.py build_ext --inplace
```

To test the installation install pytest:

```
conda install pytest
pip install hypothesis
```

And then run:

```
pytest tests.py
```

All of the tests should succeed.

1.2 Installation Windows 10

For Windows 10, an Ubuntu can be installed following the guide in the following link:

https://www.howtogeek.com/249966/how-to-install-and-use-the-linux-bash-shell-on-windows-10/

Here after all the above steps can be followed.

1.3 Running SlowQuant

The program is run as:

```
python SlowQuant.py MOLECULE SETTINGS
```

As a ready to run example:

```
python SlowQuant.py H2O.csv settingExample.csv
```

1.4 Compiling documentation

The documentation is compiled using sphinx:

```
conda install sphinx
```

To make the equations LaTeX is needed, and can be installed by:

```
sudo apt-get install texlive-full
```

The theme is downloaded by:

```
pip install sphinx_rtd_theme
```

The documentation is now compiled by:

```
sphinx-build data/documentation docs
```

CHAPTER 2

List of keywords

Settings are given in the setting inputfile by using the below keywords. The order of the keywords in the setting inputfile does not matter at all.

- basisset;x

x=bassiset used in calculation. STO2G, STO3G, DZ, DZP, 3-21G, 6-31ppGss (6-31++G**), 6-31Gs (6-31G*), 6-31pGs (6-31+G*).

2.1 Integrals

- None

2.2 Initial Method

- Initial Method;x

Method to use during SCF. x=HF for Hartree-Fock calculation. x=UHF for unrestricted Hartree-Fock calculation.

- UHF mix guess;x

Mix initial coefficients to break spin symmetry

2.3 SCF

- SCF Energy Threshold;x

Threshold for convergence of the SCF, given as x

- SCF RMSD Threshold;x

Threshold for convergence of the SCF, given as x

- SCF Max iterations;x

Maximum SCF iterations

- DIIS;x

Activation of DISS, x=Yes

- Keep Steps;x

Number of steps saved in the DIIS algorithm

2.4 Perturbation

- MPn;x

Møller-Plesset energy correction, x=MP2

2.5 Properties

- Charge;x

Calculation of atomic charges. x=Mulliken gives Mulliken charges. x=Lowdin gives Lowdin charges.

- Dipole;x

Calcultion of molecular dipolemomemnt. x=Yes for calcultion.

- Excitation energy;x

Calculation of excitation energies. x=RPA for using TDHF/RPA.

2.6 Geomoetry Optimization

- GeoOpt;x

Turns on geometry optimization. x=Yes to turn it on. Only works for Hartree-Fock.

- Max iteration GeoOpt;x

Maximum geometry optimization steps.

- Geometry Tolerance;x

Tolerance for convergence, given as x

- Gradient Descent Step;x

Gradient scaling factor in Gradient Descent algorithm

- Force Numeric;x

Choose to evaluate Forces numerically. x=Yes to activate. Only works for Hartree-Fock.

2.7 Configuration Interaction

- CI;x

To get exciation energies with CI singles. x=CIS

2.8 Couple Cluster

- CC;x

To run CCSD, x=CCSD. To run CCSD(T), x=CCSD(T)

- CC Max iterations;x

Maximaum CC amplitudes iterations.

- CC RMSD Threshold;10

RMSD Threshold for T1 and T2, given as x

- CC Energy Threshold;10

Energy change Threshold for CC, given as x

2.9 Ab-Inition Molecular Dynamics

- stepsize;x

Integration stepsize for the movement of the nuclei.

- steps;20

Number of steps for the dynamics simulation

CHAPTER 3

Example input files

```
10;;;
1;1.63803684;1.136548823;0
8;0;-0.143225817;0
1;-1.63803684;1.136548823;0
```

Example of inputfile. The molecule is water.

3.1 Hartree-Fock calculation

```
basisset;3-21G
Initial Method;HF
```

Example of settingsfile. Performs a Hartree-Fock calculation with the basisset 3-21G

3.2 MP2 calculation

```
basisset;3-21G
Initial Method;HF
MPn;MP2
```

Example of settingsfile. Performs a MP2 calculation with the basisset 3-21G

3.3 Geometry optimization

```
basisset;STO3G
Initial Method;HF
GeoOpt;Yes
```

Example of settingsfile. Performs a geometry optimization calculation with the basisset STO3G

3.4 BOMD simulation

```
basisset;3-21G
Initial method;BOMD
SCF Energy Threshold;1e-12
SCF RMSD Threshold;1e-12
steps;100
stepsize;1.0
SCF Max iterations;1000
```

Example of settingsfile. Perfoms a BOMD simulation of 100 steps, with a time step of 1.0 a.u.

CHAPTER 4

Issues

In this section known problems with the code can be found

4.1 General

- No checking of input files at all at any stages of the code
- Basisset object has a bad structure
- Loops should be replaced with np.einsum() in many cases
- Documentation of integral code is out-of-date

4.2 Performance

- Alot of recalculations in R, when calculating higher order angular momentum integrals
- Not sure variables are passed around in integral code the right way with respect to Cython
- The SCF seem to have trouble converging if BOMD time steps are above 1.0 a.u.
- All information is stored directly in memory; Integrals scale as N^4 for memory

4.3 Need testing

- DCPT2 not tested, no case in paper, where cartesian integrals was used (all had D orbitals)
- Numerical Forces are not tested yet

4.4 Broken

- Self overlap integrals is non-one for contracted basisfunctions
- No check for singularity in DIIS, H2/STO3G breaks the code if used with DIIS
- Changing Cython directive_defaults seem to make the code break sometimes, not very reproduceable

CHAPTER 5

Illustrative calculations

Here some illustrative calculations can be seen

5.1 SCF with and without DIIS

The used geometry can be seen below:

```
10;;;
1;1.63803684;1.136548823;0
8;0;-0.143225817;0
1;-1.63803684;1.136548823;0
```

The used settings can be seen below:

```
basisset;6-31ppGss
DIIS;Yes
SCF Energy Threshold;1e-5
SCF RMSD Threshold;1e-5
```

```
basisset;6-31ppGss
DIIS;No
SCF Energy Threshold;1e-5
SCF RMSD Threshold;1e-5
```

First a SCF calculation was performed with DIIS enabled giving:

```
Iter          Eel                   Etot                  dE
↪ rmsD                    DIIS
0             -134.1449183618       -126.1425513008
1             -61.4018313732        -53.3994643122         7.27430870e+01
↪ 1.37140022e+01
2             -82.1748278956        -74.1724608346        -2.07729965e+01
↪ 1.34573050e+01          7.81843771e-01
```

```
3              -83.5297539977          -75.5273869367          -1.35492610e+00
↪ 1.33204239e-01          7.81843771e-01
4              -84.0658119442          -76.0634448831          -5.36057946e-01
↪ 4.78581548e-02          7.81843771e-01
5              -84.0523274324          -76.0499603713           1.34845118e-02
↪ 8.49130680e-03          7.81843771e-01
6              -84.0148299835          -76.0124629225           3.74974488e-02
↪ 2.96875516e-03          7.81843771e-01
7              -83.9935874709          -75.9912204099           2.12425126e-02
↪ 1.76964818e-03          7.81843771e-01
8              -83.9947637830          -75.9923967219          -1.17631206e-03
↪ 2.26414594e-04          1.16112494e-01
9              -83.9947955055          -75.9924284444          -3.17225082e-05
↪ 1.66077078e-05          3.21096922e-02
10             -83.9948061725          -75.9924391115          -1.06670263e-05
↪ 6.54369763e-06          3.03280249e-03
```

Then a SCF calculation was performed without DIIS giving:

```
Iter           Eel                     Etot                    dE
↪rmsD
0              -134.1449183618         -126.1425513008
1              -61.4018313732          -53.3994643122           7.27430870e+01
↪ 1.37140022e+01
2              -104.9615819231         -96.9592148621          -4.35597505e+01
↪ 1.35548186e+01
3              -64.7628958717          -56.7605288107           4.01986861e+01
↪ 1.25781385e+01
4              -102.8062267026         -94.8038596416          -3.80433308e+01
↪ 1.25633760e+01
5              -65.5183313906          -57.5159643295           3.72878953e+01
↪ 1.23251505e+01
6              -102.3075211689         -94.3051541079          -3.67891898e+01
↪ 1.23221102e+01
7              -65.7017602918          -57.6993932308           3.66057609e+01
↪ 1.22666435e+01
8              -102.1836707293         -94.1813036683          -3.64819104e+01
↪ 1.22659169e+01
9              -65.7473338650          -57.7449668040           3.64363369e+01
↪ 1.22524386e+01
......
96             -102.1421225328         -94.1397554718          -3.63795182e+01
↪ 1.22477235e+01
97             -65.7626043489          -57.7602372879           3.63795182e+01
↪ 1.22477235e+01
98             -102.1421225328         -94.1397554718          -3.63795182e+01
↪ 1.22477235e+01
99             -65.7626043489          -57.7602372879           3.63795182e+01
↪ 1.22477235e+01
```

As it can bee seen the one with DIIS converges in 10 iterations, whereas without DIIS it never converges

5.2 Hartee-Fock H2 dissociation

Under the Hartee-Fock approximation the dissociation of dihydrogen can be considered. To investigate this the following geometry input was used:

```
2;;;
1;0.0;0.0;0.0
1;x;0.0;0.0
```

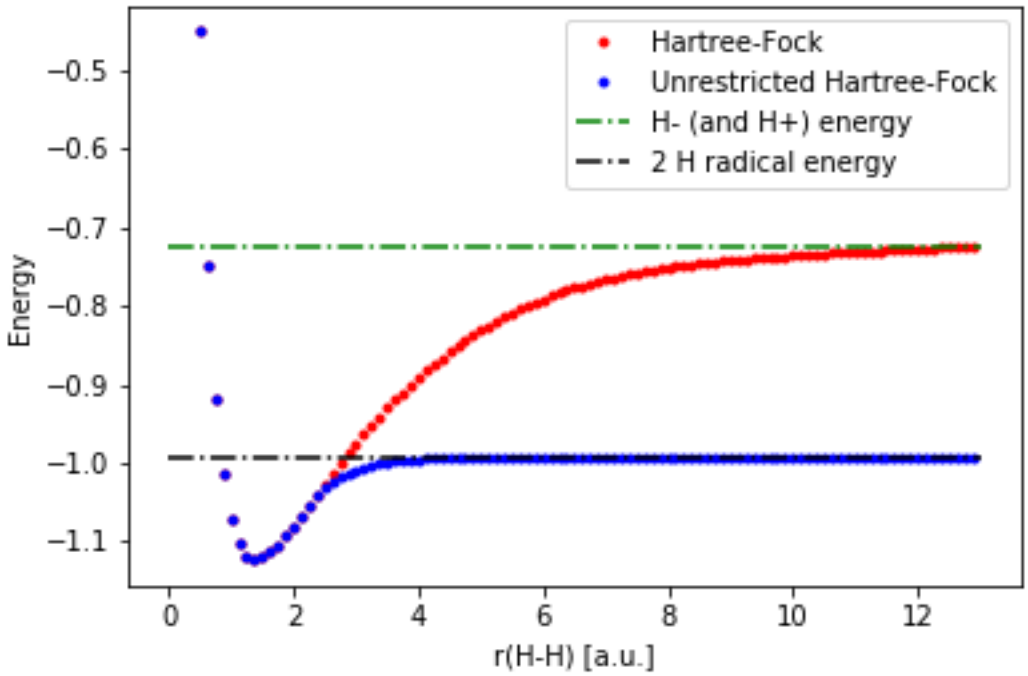
Here x is varied to make the dissociation curve. The following settings were used:

```
Initial method;HF
basisset;3-21G
```

and,

```
Initial method;UHF
basisset;3-21G
```

For x varried from 0.5 to 13 a.u. the following curve can be made:



For a script to calculate the curves see SlowQuant/data/notebooks/H2 dissociation HF vs UHF.ipynb

It can be seen from the curve that (R)HF and UHF converges to two different values. By calculating H- and H radical it can be seen that RHF converges to H- and H+, whereas UHF converges towards 2x H radicals. This can be understood since RHF is a closed-shell method. When H2 dissociates it is expected that each hydrogen will be associated with one electron, this cannot be descirbed with RHF. UHF can describe open-shells and therefore, seem to have a better describtion of the dissocation. UHF have other problems like spin contamination to be aware of.

CHAPTER 6

General Considerations

Inside the code are four different objects that ties the different moduels together. The basisset obejct "basis", the inputfile object "input", the settingsfile object "settings" and the results object "results".

6.1 Basisset object

The basisset object is a list of list with the following structure:

$$\begin{bmatrix} Idx & x & y & z & \#CONTR & CONTR & Atomidx \end{bmatrix}$$

Idx is the AO index. Atomidx, is the index of the associated atom. #CONTR contains the number of primitive functions in the contracted.

CONTR contains all the information about the primitive functions and have the form:

$$\begin{bmatrix} N & \zeta & c & l & m & n \end{bmatrix}$$

Inside the integral call, the basisset file is reconstructed into three different arrays, containing the basisset information. The first one is basisidx that have the following form:

$$\begin{bmatrix} \#\text{primitives} & \text{loop start idx} \end{bmatrix}$$

It thus contains the number of primitives in each basisfunction, and what start index it have for loop inside the integral code.

The second array is basisint, that have the following forms:

$$\begin{bmatrix} l & m & n \end{bmatrix}$$

$$\begin{bmatrix} l & m & n & \text{atom idx} \end{bmatrix}$$

The first one is for regular integrals and the second one is for derivatives. Both contains all the angular momentum quantum numbers, and the derivative also contains the atom index (used in derivative of VNe).

The last array is basisfloat and have the following forms:

$$\begin{bmatrix} N & \zeta & c & x & y & z \end{bmatrix}$$

$$\begin{bmatrix} N & \zeta & c & x & y & z & N_{x,+} & N_{x,-} & N_{y,+} & N_{y,-} & N_{z,+} & N_{z,-} \end{bmatrix}$$

basisfloat contains the normalization constants, Gaussian exponent and prefacor and the coordinates of the atoms. The second one is again for the derivatives, it contains normalization constants of the differentiated primitives.

6.2 Inputfile obejct

The inputfile obejct is a numpy array containing the following informations:

$$\begin{bmatrix} \#electrons & None & None & None \\ Atom_1\, nr. & Atom_1\, x & Atom_1\, y & Atom_1\, z \\ Atom_2\, nr. & Atom_2\, x & Atom_2\, y & Atom_2\, z \\ \dots & \dots & \dots & \dots \\ Atom_i\, nr. & Atom_i\, x & Atom_i\, y & Atom_i\, z \end{bmatrix}$$

6.3 Settings obejct

The settings are parsed around in the code as a dictionary.

6.4 Results object

The results are parsed around in the code as a dictionary. It contains the results from different calculatations that are either used in other calculations or given as an output.

CHAPTER 7

Molecular Integrals

Contains the information about how the integrals are calculated. In the equations in this section the following definitions is used.

$$p = a + b$$

$$\mu = \frac{ab}{a+b}$$

$$P_x = \frac{aA_x + bB_x}{p}$$

$$X_{AB} = A_x - B_x$$

Here a and b are Gaussian exponent factors. Ax and Bx are the position of the Gaussians in one dimension. Further the basisset functions is of Gaussian kind and given as:

$$\phi_A(r) = N(x - A_x)^l (y - A_y)^m (z - A_z)^n \exp\left(-\zeta\left(\vec{r} - \vec{A}\right)^2\right)$$

with a normalization constant given as:

$$N = \left(\frac{2\alpha}{\pi}\right)^{3/4} \left[\frac{(8\alpha)^{l+m+n} l!m!n!}{(2l)!(2m)!(2n)!}\right]$$

7.1 Boys Function

The Boys function is given as:

$$F_n(x) = \int_0^1 \exp\left(-xt^2\right) t^{2n} dt$$

FUNCTION:

- MIcython.boys(m,T)

- return value

Input:

- m, subscript of the Boys function
- T, argument of the Boys function

Output:

- value, value corrosponding to given m and T

References:

- Trygve Helgaker, Poul Jorgensen and Jeppe Olsen, Molecular Electronic-Structure Theory

7.2 Expansion coefficients

The expansion coefficient is found by the following recurrence relation:

$$E_t^{i,j} = 0, \quad t < 0 \text{ or } t > i + j$$

$$E_t^{i+1,j} = \frac{1}{2p} E_{t-1}^{i,j} + X_{\mathrm{PA}} E_t^{i,j} + (t+1)\, E_{t+1}^{i,j}$$

$$E_t^{i,j+1} = \frac{1}{2p} E_{t-1}^{i,j} + X_{\mathrm{PB}} E_t^{i,j} + (t+1)\, E_{t+1}^{i,j}$$

With the boundary condition that:

$$E_0^{0,0} = \exp\left(-p X_{\mathrm{AB}}^2\right)$$

FUNCTION:

- MolecularIntegrals.E(i,j,t,Qx,a,b,XPA,XPB,XAB)
- return val

Input:

- i, input values
- j, input values
- t, input values
- Qx, input values
- a, input values
- b, input values
- XPA, input values
- XPB, input values
- XAB, input values

Output:

- val, value corrosponding to the given input

References:

- Trygve Helgaker, Poul Jorgensen and Jeppe Olsen, Molecular Electronic-Structure Theory
- http://joshuagoings.com/assets/integrals.pdf

7.3 Hermite coulomb integral

The hermite coulomb integrals is given as the following recurrence relations:

$$R^n_{t+1,u,v}\left(p, R_{\mathrm{PC}}\right) = tR^{n+1}_{t-1,u,v}\left(p, R_{\mathrm{PC}}\right) + X_{\mathrm{PC}}R^{n+1}_{t,u,v}\left(p, R_{\mathrm{PC}}\right)$$

$$R^n_{t,u+1,v}\left(p, R_{\mathrm{PC}}\right) = uR^{n+1}_{t,u-1,v}\left(p, R_{\mathrm{PC}}\right) + Y_{\mathrm{PC}}R^{n+1}_{t,u,v}\left(p, R_{\mathrm{PC}}\right)$$

$$R^n_{t,u,v+1}\left(p, R_{\mathrm{PC}}\right) = vR^{n+1}_{t,u,v-1}\left(p, R_{\mathrm{PC}}\right) + Z_{\mathrm{PC}}R^{n+1}_{t,u,v}\left(p, R_{\mathrm{PC}}\right)$$

With the boundary condition:

$$R^n_{0,0,0}\left(p, R_{\mathrm{PC}}\right) = (-2p)^n F_n\left(pR^2_{\mathrm{PC}}\right)$$

FUNCTION:

- MIcython.R2(t,u,v,n,p,PCx,PCy,PCz,RPC)
- return val

Input:

- t, input value
- u, input value
- v, input value
- n, input value
- p, input value
- PCx, input value
- PCy, input value
- PCz, input value
- RPC, input value

Output:

- val, value corrosponding to the given input

References:

- Trygve Helgaker, Poul Jorgensen and Jeppe Olsen, Molecular Electronic-Structure Theory
- http://joshuagoings.com/assets/integrals.pdf

7.4 Overlap

The overlap integrals are solved by the following recurrence relation:

$$S_{i+1,j} = X_{PA}S_{ij} + \frac{1}{2p}\left(iS_{i-1,j} + jS_{i,j-1}\right)$$

$$S_{i,j+1} = X_{PB}S_{ij} + \frac{1}{2p}\left(iS_{i-1,j} + jS_{i,j-1}\right)$$

With the boundary condition that:

$$S_{00} = \sqrt{\frac{\pi}{p}}\exp\left(-\mu X^2_{AB}\right)$$

FUNCTION:

- MolecularIntegrals.Overlap(a, b, la, lb, Ax, Bx)
- return Sij

Input:

- a, Gaussian exponent factor
- b, Gaussian exponent factor
- la, angular momentum quantum number
- lb, angular momentum quantum number
- Ax, position along one axis
- Bx, position along one axis

Output:

- Sij, non-normalized overlap element in one dimension

References:

- Trygve Helgaker, Poul Jorgensen and Jeppe Olsen, Molecular Electronic-Structure Theory

7.5 Kinetic energy

The kinetic energy integrals are solved by the following recurrence relation:

$$T_{i+1,j} = X_{\mathrm{PA}} T_{i,j} + \frac{1}{2p}\left(iT_{i-1,j} + jT_{i,j-1}\right) + \frac{b}{p}\left(2aS_{i+1,j} - iS_{i-1,j}\right)$$

$$T_{i,j+1} = X_{\mathrm{PB}} T_{i,j} + \frac{1}{2p}\left(iT_{i-1,j} + jT_{i,j-1}\right) + \frac{a}{p}\left(2bS_{i,j+1} - iS_{i,j-1}\right)$$

With the boundary condition that:

$$T_{00} = \left[a - 2a^2\left(X_{\mathrm{PA}}^2 + \frac{1}{2p}\right)\right] S_{00}$$

FUNCTION:

- Kin(a, b, Ax, Ay, Az, Bx, By, Bz, la, lb, ma, mb, na, nb, N1, N2, c1, c2)
- return Tij, Sij

Input:

- a, Gaussian exponent factor
- b, Gaussian exponent factor
- Ax, position along the x-axis
- Bx, position along the x-axis
- Ay, position along the y-axis
- By, position along the y-axis
- Az, position along the z-axis
- Bz, position along the z-axis
- la, angular momentum quantum number

- lb, angular momentum quantum number
- ma, angular momentum quantum number
- mb, angular momentum quantum number
- na, angular momentum quantum number
- nb, angular momentum quantum number
- N1, normalization constant
- N2, normalization constant
- c1, Gaussian prefactor
- c2, Gaussian prefactor

Output:

- Tij, normalized kinetic energy matrix element
- Sij, normalized overlap matrix element

References:

- Trygve Helgaker, Poul Jorgensen and Jeppe Olsen, Molecular Electronic-Structure Theory

7.6 Electron-nuclear attraction

The electron-nuclear interaction integral is given as:

$$V_{ijklmn}^{000} = \frac{2\pi}{p} \sum_{t}^{i+j} E_t^{ij} \sum_{u}^{k+l} E_u^{kl} \sum_{v}^{m+n} E_v^{mn} R_{tuv}$$

FUNCTION:

- MolecularIntegrals.elnuc(P, p, l1, l2, m1, m2, n1, n2, N1, N2, c1, c2, Zc, Ex, Ey, Ez, R1)
- return Vij

Input:

- P, Gaussian product
- p, exponent from Guassian product
- l1, angular momentum quantum number
- l2, angular momentum quantum number
- m1, angular momentum quantum number
- n1, angular momentum quantum number
- n2, angular momentum quantum number
- N1, normalization constant
- N2, normalization constant
- c1, Gaussian prefactor
- c2, Gaussian prefactor
- Zc, Nuclear charge

- Ex, expansion coefficients
- Ey, expansion coefficients
- Ez, expansion coefficients
- R1, hermite coulomb integrals

Output:

- Vij, normalized electron-nuclei attraction matrix element

References:

- Trygve Helgaker, Poul Jorgensen and Jeppe Olsen, Molecular Electronic-Structure Theory
- http://joshuagoings.com/assets/integrals.pdf

7.7 Electron-nuclear field

The electron-nuclear interaction integral is given as:

$$V_{ijklmn}^{efg} = (-1)^{e+f+g} \frac{2\pi}{p} \sum_{t}^{i+j} E_t^{ij} \sum_{u}^{k+l} E_u^{kl} \sum_{v}^{m+n} E_v^{mn} R_{t+e,u+f,v+g}$$

Here e, f and g detones the order of derivate with respect to x, y and z

FUNCTION:

- MolecularIntegrals.electricfield(p, Ex, Ey, Ez, Zc, l1, l2, m1, m2, n1, n2, N1, N2, c1, c2, derivative, R1)
- return VijA

Input:

- p, Gaussian exponent form Gaussian product
- Ex, expansion coefficient
- Ey, expansion coefficient
- Ez, expansion coefficient
- l1, angular momentum quantum number
- l2, angular momentum quantum number
- m1, angular momentum quantum number
- n1, angular momentum quantum number
- n2, angular momentum quantum number
- N1, normalization constant
- N2, normalization constant
- c1, Gaussian prefactor
- c2, Gaussian prefactor
- derivative, axis of derivative (dx,dy or dz)
- R1, hermite coulomb integral

Output:

- VijA, normalized electron-nuclei field of nuclei A matrix element

References:

- Trygve Helgaker, Poul Jorgensen and Jeppe Olsen, Molecular Electronic-Structure Theory

7.8 Electron-electron repulsion

The electron-electron repulsion integral is calculated as:

$$g_{abcd} = \sum_{t}^{l1+l2} E_t^{ab} \sum_{u}^{m1+m2} E_u^{ab} \sum_{v}^{n1+n2} E_v^{ab} \sum_{\tau}^{l3+l4} E_\tau^{cd} \sum_{\nu}^{m3+m4} E_\nu^{cd} \sum_{\phi}^{n3+n4} E_\phi^{cd} (-1)^{\tau+\nu+\phi} \frac{2\pi^{5/2}}{pq\sqrt{p+q}} R_{t+\tau,u+\nu,v+\phi}(\alpha, R_{\mathrm{PQ}})$$

FUNCTION:

- MIcython.elelrep(p, q, l1, l2, l3, l4, m1, m2, m3, m4, n1, n2, n3, n4, N1, N2, N3, N4, c1, c2, c3, c4, E1, E2, E3, E4, E5, E6, Rpre)
- return Veeijkl

Input:

- p, Gaussian exponent factor from Gaussian product
- q, Gaussian exponent factor from Gaussian product
- l1, angular momentum quantum number
- l2, angular momentum quantum number
- l3, angular momentum quantum number
- l4, angular momentum quantum number
- m1, angular momentum quantum number
- m2, angular momentum quantum number
- m3, angular momentum quantum number
- m4, angular momentum quantum number
- n1, angular momentum quantum number
- n2, angular momentum quantum number
- n3, angular momentum quantum number
- n4, angular momentum quantum number
- N1, normalization constant
- N2, normalization constant
- N3, normalization constant
- N4, normalization constant
- c1, Gaussian prefactor
- c2, Gaussian prefactor
- c3, Gaussian prefactor
- c4, Gaussian prefactor

- E1, expansion coefficient
- E2, expansion coefficient
- E3, expansion coefficient
- E4, expansion coefficient
- E5, expansion coefficient
- E6, expansion coefficient
- Rpre, hermite coulomb integral

Output:

- Veeijkl, normalized electron-electron repulsion matrix element

References:

- Trygve Helgaker, Poul Jorgensen and Jeppe Olsen, Molecular Electronic-Structure Theory
- http://joshuagoings.com/assets/integrals.pdf

7.9 Nuclear-nuclear repulsion

The nucleus-nucleus repulsion term is calculated with classical nuclie as follows:

$$V_{NN} = \sum_{A} \sum_{B<A} \frac{Z_A Z_B}{r_{AB}}$$

FUNCTION:

- MolecularIntegrals.nucrep(input)
- return Vnn

Input:

- input, inputfile object

Output:

- Vnn, nuclear repulsion energy

References:

- None

7.10 Nuclear-nuclear field

The nucleus-nucleus field term is calculated with classical nuclie as follows:

$$\frac{\partial V_{NN}}{\partial X_A} = -Z_A \sum_{B \neq A} \frac{Z_B \left(X_B - X_A\right)}{r_{AB}^3}$$

FUNCTION:

- MolecularIntegrals.nucdiff(input, atomidx, direction)
- return Vnn

Input:

- input, inputfile object
- atomidx, atom which is differentiated with respect to
- direction, axis of differentiation (1 = dx, 2 = dy, 3 = dz)

Output:

- Vnn, nucleus-nucleus field

References:

- None

7.11 Dipole moment integral

The dipole moment integral is calculated by using the following relations:

$$S^e_{i+1,j} = X_{\mathrm{PA}} S^e_{i,j} + \frac{1}{2p}\left(iS^e_{i-1,j} + jS^e_{i,j-1} + eS^{e-1}_{ij}\right)$$

$$S^e_{i,j+1} = X_{\mathrm{PB}} S^e_{i,j} + \frac{1}{2p}\left(iS^e_{i-1,j} + jS^e_{i,j-1} + eS^{e-1}_{ij}\right)$$

$$S^{e+1}_{i,j} = X_{\mathrm{PC}} S^e_{i,j} + \frac{1}{2p}\left(iS^e_{i-1,j} + jS^e_{i,j-1} + eS^{e-1}_{ij}\right)$$

Here e is the order of multipole moment, e=1 is dipole moment.

FUNCTION:

- MolecularIntegrals.u_ObaraSaika(a1, a2, Ax, Ay, Az, Bx, By, Bz, la, lb, ma, mb, na, nb, N1, N2, c1, c2, input)
- return muxij, muyij, muzij

Input:

- a1, Gaussian exponent factor
- a2, Gaussian exponent factor
- Ax, position along x axis
- Ay, position along y axis
- Az, position along z axis
- Bx, position along x axis
- By, position along y axis
- Bz, position along z axis
- la, angular momentum quantum number
- lb, angular momentum quantum number
- ma, angular momentum quantum number
- mb, angular momentum quantum number
- na, angular momentum quantum number
- nb, angular momentum quantum number

- N1, normalization constant
- N2, normalization constant
- c1, Gaussian prefactor
- c2, Gaussian prefactor
- input, inputfile object

Output:

- muxij, dipolemoment integral matrix element for x axis
- muyij, dipolemoment integral matrix element for y axis
- muzij, dipolemoment integral matrix element for z axis

References:

- Trygve Helgaker, Poul Jorgensen and Jeppe Olsen, Molecular Electronic-Structure Theory

CHAPTER 8

Integral Transformations

In this section the integral transformations will be described.

8.1 AO to MO basis 2 electron integrals

The integral transformation of the AO two electron integrals to the MO two electron integrals is given as:

$$(ij \mid kl) = \sum_{\mu} \sum_{\nu} \sum_{\lambda} \sum_{\sigma} C_{\mu}^{i} C_{\nu}^{j} (\mu\nu \mid \lambda\sigma) C_{\lambda}^{k} C_{\sigma}^{l}$$

FUNCTION:

- IntegralTransform.Transform2eMO(C, Vee)
- return VeeMO

Input:

- C, MO coefficients from SCF calculation
- Vee, two electron integrals in AO basis

Output:

- VeeMO, two electron integrals in MO basis

References:

- http://sirius.chem.vt.edu/wiki/doku.php?id=crawdad:programming:project4

8.2 MO spatial to MO spin basis 2 electron integrals

The integral transformation from MO spatial to MO spin orbitals is given by the following equation:

$$\langle ij \mid kl \rangle = (\sigma_1 i \sigma_2 k \mid \sigma_3 j \sigma_4 l) \, \delta_{\sigma_1 \sigma_2} \delta_{\sigma_3 \sigma_4}$$

FUNCTION:

- IntegralTransform.Transform2eSPIN(Vee)
- return VeeSpin

Input:

- Vee, two electron integrals

Output:

- VeeSpin, two electron integrals in spinbasis

References:

- None

CHAPTER 9

Hartree-Fock methods

In this section the Hartree-Fock methods will be described.

9.1 Restricted Hartree-Fock

For the Roothan-Hartee-Fock equations an orthogonal basis is needed, first the first orthogonalization matrix is constructed from the overlap matrix. First by a diagonalization:

$$SL_S = L_S \Lambda_S$$

and then the orthogonalization matrix is constructed:

$$S_{\mathrm{ortho}} = L_S \Lambda_S^{-1/2} L_S^T$$

The SCF iterations requires an initial Fock matrix given as:

$$F_0 = H^{\mathrm{core}}$$

The SCF procedure is calulated as the following equations. First the fock matrix is constructed:

$$F_{n,ij}|_{n \neq 0} = H_{ij}^{\mathrm{core}} + \sum_{kl}^{\mathrm{AO}} D_{n-1,kl} \left(2 \left(ij \,||\, kl \right) - \left(ik \,||\, jl \right) \right)$$

Then the Fock matrix is brought into the orthogonal basis:

$$F_n^{'} = S_{\mathrm{ortho}}^T H^{\mathrm{core}} S_{\mathrm{ortho}}$$

The F' is then diagonalized:

$$F_n^{'} C_n^{'} = C_n^{'} \epsilon_n$$

The coefficients are transformed back:

$$C_n = S_{\mathrm{ortho}} C_n^{'}$$

A density matrix can be made from the coefficients:

$$D_{n,ij} = \sum_{k}^{\text{occ}} C_{n,ki} C_{n,kj}$$

The electronic energy of system can be found as:

$$E_{n,\text{elec}} = \sum_{ij}^{\text{AO}} D_{0,ij} \left(H_{ij}^{\text{core}} + F_{n,ij} \right)$$

The above SCF procedure will is stopped at certain tresholds. The change in energy and the RMSD of the density matrix can be found as:

$$\Delta E_n = E_{n,\text{elec}} - E_{n-1,\text{elec}}$$

$$\text{RMSD}_n = \sqrt{\sum_{ij} D_{n,ij} - D_{n-1,ij}}$$

FUNCTION:

- HartreeFock.HartreeFock(input, VNN, Te, S, VeN, Vee, deTHR=10**-6,rmsTHR=10**-6,Maxiter=100, DO_DIIS='Yes', DIIS_steps=6, print_SCF='Yes')
- return EHF, C, F, D, iter

Input:

- input, inputfile object
- VNN, nuclear-nuclear repulsion
- Te, electronic energy matrix
- S, overlap matrix
- VeN, nuclear-electron attraction matrix
- Vee, ERI matrix
- deTHR, energy change threshold for convergence
- rmsTHR, RMS threshold for convergence
- Maxiter, max SCF iterations
- DO_DIIS, enable DIIS, if = 'Yes'
- DIIS_steps, how many steps are stored in DIIS
- print_SCF, wether to print SCF to output

Output:

- EHF, Total Hartree-Fock energy
- C, MO coefficients
- F, Fock matrix
- D, density matrix
- iter, SCF iterations used

References:

- http://sirius.chem.vt.edu/wiki/doku.php?id=crawdad:programming:project3

9.2 Unrestricted Hartee-Fock

The unrestricted Hartee-Fock method uses the same SCF procedure as as the restricted Hartree-Fock, but with the Fock matrix coupling the alpha and beta spins:

$$F_{n,\alpha,ij} = H_{ij}^{\text{core}} + \sum_{kl}^{\text{AO}} D_{n-1,\alpha,kl} \left((ij \,||\, kl) - (ik \,||\, jl) \right) + \sum_{kl}^{\text{AO}} D_{n-1,\beta,kl} \left(ij \,||\, kl \right)$$

FUNCTION:

- UHF.HartreeFock(input, VNN, Te, S, VeN, Vee, deTHR=10**-6,rmsTHR=10**-6,Maxiter=100, UHF_mix=0.15, print_SCF='Yes')
- return EUHF, C_alpha, F_alpha, D_alpha, C_beta, F_beta, D_beta, iter

Input:

- input, inputfile object
- VNN, nuclear-nuclear repulsion
- Te, electronic energy matrix
- S, overlap matrix
- VeN, nuclear-electron attraction matrix
- Vee, ERI matrix
- deTHR, energy change threshold for convergence
- rmsTHR, RMS threshold for convergence
- Maxiter, max SCF iterations
- UHF_mix, how much beta and alpha are mixed to break symmetry
- print_SCF, wether to print SCF to output

Output:

- EUHF, total UHF energy
- C_alpha, MO coefficients
- F_alpha, Fock matrix
- D_alpha, density matrix
- C_beta, MO coefficients
- F_beta, Fock matrix
- D_beta, density matrix

References:

- Szabo and Ostlund, Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory

In unrestricted Hartree-Fock for a closed shell system the spin-symmetry needs to be broken else restricted Hartree-Fock is restored. This is done by the following method, after the first MO coefficients have been made:

$$C_{i,\text{HOMO}}^{\text{new}} = \frac{1}{\sqrt{1+k^2}} \left(C_{i,\text{HOMO}}^{\text{old}} + k C_{i,\text{LUMO}}^{\text{old}} \right)$$

$$C_{i,\text{LUMO}}^{\text{new}} = \frac{1}{\sqrt{1+k^2}} \left(-k C_{i,\text{HOMO}}^{\text{old}} + C_{i,\text{LUMO}}^{\text{old}} \right)$$

CHAPTER 10

DIIS

Direct Inversion in the Iterative Subspace (DIIS). Makes new F' guesses bassed on previous guesses.

The error vector in DIIS is given as:

$$e_i = F_i D_i S - S D_i F_i$$

It is wanted that the sum of error vectors is zero:

$$e' = \sum_i c_i e_i = 0$$

And now with the requirement that the sum of all c is zero, the following matrix eqution is solwed:

$$B_{i,j} c_i = b0$$

Here:

$$B_{ij} = \mathrm{tr}\,(e_i \cdot e_j)$$

and,

$$b0 = \begin{pmatrix} 0 \\ 0 \\ \cdots \\ -1 \end{pmatrix}$$

Finally the new F' is constructed as:

$$F' = \sum_i c_i F_i$$

FUNCTION:

- DIIS.DIIS(F,D,S,Efock,Edens,numbF)
- return Fprime, Efock, Edens, Emax

Input:

- F, Fock matrix
- D, Density matrix
- S, overlap matrix
- Efock, saved fock matrices
- Edens, saved density matrices
- numbF, number of matrixes to save

Output:

- Fprime, F' guess
- Efock, saved fock matrices
- Edens, saved density matrices
- Emax, Maximum error in error vector

References:

- http://sirius.chem.vt.edu/wiki/doku.php?id=crawdad:programming:project8
- 16. Pulay, Chem. Phys. Lett. 73, 393 (1980).

CHAPTER 11

Perturbations

Contains information about perturbative methods.

11.1 Møller-Plesset, second order

The second order Møller-Plesset correction to the energy is given as:

$$E_{\mathrm{MP2}} = \sum_{i,j}^{\mathrm{occ}} \sum_{a,b}^{\mathrm{unocc}} \frac{(ia\,|jb)\,[2\,(ia\,|jb) - (ib\,|ja)]}{\epsilon_i + \epsilon_j - \epsilon_a - \epsilon_b}$$

The above equation is valid for closed shell systems. The integrals is in Mulliken notation, and the denominator is the orbital energies in MO, obtained from an SCF procedure.

FUNCTION:

- MPn.MP2(occ, F, C, VeeMO)
- return EMP2

Input:

- occ, number of occupied MOs
- F, Fock matrix
- C, MO coefficient matrix
- VeeMO, twoelecron integral matrix

Output:

- EMP2, MP2 correction to the Hartree-Fock energy

References:

- http://sirius.chem.vt.edu/wiki/doku.php?id=crawdad:programming:project4
- Szabo and Ostlund, Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory

11.2 Møller-Plesset, third order

The third order Møller-Plesset correction to the energy is given as:

$$E^{(3)} = \frac{1}{8}\sum_{abcdrs}\frac{\langle ab\,||\,rs\rangle\,\langle cd\,||\,ab\rangle\,\langle rs\,||\,cd\rangle}{(\epsilon_a+\epsilon_b-\epsilon_r-\epsilon_s)\,(\epsilon_c+\epsilon_d-\epsilon_r-\epsilon_s)} + \frac{1}{8}\sum_{abrstu}\frac{\langle ab\,||\,rs\rangle\,\langle rs\,||\,tu\rangle\,\langle tu\,||\,ab\rangle}{(\epsilon_a+\epsilon_b-\epsilon_r-\epsilon_s)\,(\epsilon_a+\epsilon_b-\epsilon_t-\epsilon_u)} + \sum_{abcrst}\frac{\langle ab\,||\,rs\rangle\,\langle cs\,||\,tb\rangle\,\langle rt\,||\,ac\rangle}{(\epsilon_a+\epsilon_b-\epsilon_r-\epsilon_s)\,(\epsilon_a+\epsilon_c-\epsilon_r-\epsilon_t)}$$

FUNCTION:

- MPn.MP3(occ, F, C, VeeMO)
- return EMP3

Input:

- occ, number of occupied MOs in spinbasis
- F, Fock matrix
- C, MO coefficient matrix
- VeeMOspin, twoelecron integral matrix in spinbasis

Output:

- EMP3, MP3 correction to the energy

References:

- Szabo and Ostlund, Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory

11.3 Degeneracy-corrected perturbation, second order

The second order Degeneracy-corrected correction to the energy is given as:

$$E^{(2)} = \frac{1}{2}\left(D_{abij} - \sqrt{D_{abij}^2 + 4\,\langle\, ij|\, ab|\rangle^2}\right) + \frac{1}{4}\left(D_{abij} - \sqrt{D_{abij}^2 + 4\,(\langle\, ij|\, ab|\rangle - \langle\, ij|\, ba|\rangle)^2}\right)$$

with:

FUNCTION:

- MPn.DCPT2(occ, F, C, VeeMO)
- return EDCPT2

Input:

- occ, number of occupied MOs
- F, Fock matrix
- C, MO coefficient matrix
- VeeMO, twoelecron integral matrix

Output:

- EDCPT2, DCPT2 correction to the Hartree-Fock energy

References:

- Xavier Assfeld, Jan E Almlöf, and Donald G Truhlar. Degeneracy-corrected perturbation theory for electronic structure calculations. Chemical physics letters, 241(4):438–444, 1995

CHAPTER 12

Properties

This section containts information about atomic and molecular properties that can be calculated.

12.1 Molecular dipole

The molecular dipole in one dimension is found as:

$$\mu_x = -\sum_i \sum_j D_{i,j} \left(i \left|x\right| j\right) + \sum_A Z_A X_A$$

FUNCTION:

- Properties.dipolemoment(input, D, mux, muy, muz)
- return ux, uy, uz, u

Input:

- input, inputfile object
- D, density matrix
- mux, dipolemoment integrals in x direction
- muy, dipolemoment integrals in y direction
- muz, dipolemoment integrals in z direction

Output:

- ux, dipolemoment in x direction
- uy, dipolemoment in y direction
- uz, dipolemoment in z direction
- u, total dipolemoment

Refrence:

- Szabo and Ostlund, Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory

12.2 Mulliken charges

The atomic charge of the A'th atom can be found as:

$$q_A = Z_A - \sum_{i \in A} (D \cdot S)_{i,i}$$

FUNCTION:

- Properties.MulCharge(basis, input, D, S)
- return qvec

Input:

- basis, basisset object
- input, inputfile object
- D, density matrix
- S, overlap matrix

Output:

- qvec, vector of Mulliken charges

Refrence:

- Szabo and Ostlund, Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory

12.3 Lowdin charges

The atomic charge of the A'th atom can be found as:

$$q_A = Z_A - \sum_{i \in A} \left(S^{1/2} \cdot D \cdot S^{1/2} \right)_{i,i}$$

FUNCTION:

- Properties.LowdinCharge(basis, input, D, S)
- return qvec

Input:

- basis, basisset object
- input, inputfile object
- D, density matrix
- S, overlap matrix

Output:

- qvec, vector of Lowdin charges

Refrence:

- Szabo and Ostlund, Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory

12.4 Random-Phase Approximation Excitation energy

The excitation energies can be calculated by using the random-phase approximation also known as time dependent Hartree-Fock. The exciation energy is found by diagonalizing the following equation:

$$(A+B)(A-B)X = E^2 X$$

with the elements given as:

$$A_{ia,jb} = f_{ab}\delta_{ij} - f_{ij}\delta_{ab} + \langle aj \,||\, ib \rangle$$

$$B_{ia,jb} = \langle ab \,||\, ij \rangle$$

All of the elements are in spin basis.

FUNCTION:

- Properties.RPA(occ, F, C, VeeMOspin)
- return Exc

Input:

- occ, number of occupied MOs in spinbasis
- F, fock matrix in spatial basis
- C, MO coeffcients in spatial basis
- VeeMOspin, MO integrals in spin basis

Output:

- Exc, single excitation energies

References:

- http://sirius.chem.vt.edu/wiki/doku.php?id=crawdad:programming:project12

CHAPTER 13

Geometry optimization

In this section the geometry optimization are described. In general the derivative of the Gaussian basisset functions is given as:

$$\frac{\partial \phi_{l,m,n}}{\partial X_A} = \left[(2l+1)\, a\right]^{1/2} \phi_{l+1,m,n} - 2l \left(\frac{a}{2l-1}\right)^{1/2} \phi_{l-1,m,n}$$

13.1 Gradient Descent

The molecular structure is propagated using the Gradient Descent method:

$$X_{A,i+1} = X_{A,i} - \zeta \frac{\partial E_{\mathrm{HF}}}{\partial X_A}$$

FUNCTION:

- None

References:

- None

13.2 Analytic Hartree-Fock

The analytic Hartree-Fock derivative is given as:

$$\frac{\partial E_{\mathrm{HF}}}{\partial X_A} = \sum_{i,j} D_{i,j} \frac{H_{i,j}^{\mathrm{core}}}{\partial X_A} + \frac{1}{2} \sum_{i,j,k,l} D_{j,i} D_{k,l} \frac{\partial \left(ij \,||\, kl\right)}{\partial X_A} - 2 \sum_{i,j} \sum_{a}^{\mathrm{occ}} \varepsilon_a C_{j,a} C_{i,a} \frac{\partial S_{i,j}}{\partial X_A} + \frac{\partial V_{NN}}{\partial X_A}$$

FUNCTION:

- GeometryOptimization.run_analytic(input, set, results)

Input:

- input, inputfile object
- set, settingsfile object
- results, results object

Output:

- input, inputfile object with opdated coordinates

References:

- Szabo and Ostlund, Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory

13.3 Finite difference

The numerical gradient is calculated by the use of finite difference, and is found as:

$$\frac{\partial E_{\mathrm{HF}}}{\partial X_A} = \frac{E_{\mathrm{HF}}\left(X_A + \epsilon\right) - E_{\mathrm{HF}}\left(X_A - \epsilon\right)}{2\epsilon}$$

FUNCTION:

- GeometryOptimization.run_numeric(input, set, results)
- return input

Input:

- input, inputfile object
- set, settingsfile object
- results, results object

Output:

- input, inputfile object with opdated coordinates

References:

- None

CHAPTER 14

Configuration Interaction

In this section configuration interaction methods will be described.

14.1 CI Singles

The CI singles are done by building a Hamiltonian and diagonalizing it. The Hamiltonian build accourding to:

$$H_{ia,jb} = f_{ab}\delta_{ij} - f_{ij}\delta_{ab} + \langle aj \,||\, ib \rangle$$

Here everything is in spin orbital basis.

FUNCTION:

- CI.CIS(occ, F, C, VeeMOspin)
- return Exc

Input:

- occ, number of occupied MOs in spinbasis
- F, fock matrix in spatial basis
- C, MO coeffcients in spatial basis
- VeeMOspin, MO integrals in spin basis

Output:

- Exc, single excitation energies

References:

- http://sirius.chem.vt.edu/wiki/doku.php?id=crawdad:programming:project12

CHAPTER 15

Coupled Cluster

In this section the working equations for Coupled Cluster can be found.

15.1 Coupled Cluster Singles Double

For CCSD the T1 (tia) and T2 (tijab) amplitudes is constructed to calculate the energy. These are constructed by the fisrt creating some intermediates:

$$F_{ae} = (1 - \delta_{ae}) f_{ae} - \frac{1}{2} \sum_{m}^{occ} f_{me} t_m^a + \sum_{m,f}^{occ,virt} t_m^f \langle ma \,||\, fe \rangle - \frac{1}{2} \sum_{m,n,f}^{occ,occ,virt} \tilde{\tau}_{mn}^{af} \langle mn \,||\, ef \rangle$$

$$F_{mi} = (1 - \delta_{mi}) f_{mi} + \frac{1}{2} \sum_{e}^{virt} t_i^e f_{me} + \sum_{e,n}^{virt,occ} \langle mn \,||\, ie \rangle + \frac{1}{2} \sum_{n,e,f}^{occ,virt,virt} \tilde{\tau}_{in}^{ef} \langle mn \,||\, ef \rangle$$

$$F_{me} = f_{me} + \sum_{n,f}^{occ,virt} t_n^f \langle mn \,||\, ef \rangle$$

$$W_{mnij} = \langle mn \,||\, ij \rangle + P_{-}(ij) \sum_{e}^{virt} t_j^e \langle mn \,||\, ie \rangle + \frac{1}{4} \sum_{e,f}^{virt,virt} \tau_{ij}^{ef} \langle mn \,||\, ef \rangle$$

$$W_{abef} = \langle ab \,||\, ef \rangle - P_{-}(ab) \sum_{m}^{occ} t_m^b \langle ma \,||\, ef \rangle + \frac{1}{4} \sum_{m,n}^{occ,occ} \tau_{mn}^{ab} \langle mn \,||\, ef \rangle$$

$$W_{mbej} = \langle mb \,||\, ej \rangle + \sum_{f}^{virt} \langle mb \,||\, ef \rangle - \sum_{n}^{occ} t_n^b \langle mn \,||\, ej \rangle - \sum_{n,f}^{occ,virt} \left(\frac{1}{2} t_{jn}^{fb} + t_j^f t_n^b \right) \langle mn \,||\, ef \rangle$$

In the above equations the following definitions is used:

$$\tilde{\tau}_{ij}^{ab} = t_{ij}^{ab} + \frac{1}{2} \left(t_i^a t_j^b - t_i^b t_j^a \right)$$

$$\tau = t_{ij}^{ab} + t_i^a t_j^b - t_i^b t_j^a$$

$$P_-(ij) = 1 - P(ij)$$

The T1 and T2 is the constructed as:

$$t_i^a D_i^a = f_{ia} + \sum_e^{occ} t_i^e F_{ae} - \sum_m^{occ} t_m^a F_{mi} + \sum_{m,e}^{occ,virt} t_{im}^{ae} F_{me} - \sum_{n,f}^{occ,virt} t_n^f \langle na \,||\, if \rangle$$
$$-\frac{1}{2} \sum_{m,e,f}^{occ,virt,virt} t_{im}^{ef} \langle ma \,||\, ef \rangle - \frac{1}{2} \sum_{m,e,n}^{occ,virt,occ} t_{mn}^{ae} \langle mn \,||\, ei \rangle$$

$$t_{ij}^{ab} D_{ij}^{ab} = \langle ij \,||\, ab \rangle + P_-(ab) \sum_e^{virt} t_{ij}^{ae} \left(F_{be} - \frac{1}{2} \sum_m^{occ} t_m^b F_{me} \right) + \frac{1}{2} \sum_{e,f}^{virt,virt} \tau_{ij}^{ef} W_{abef}$$
$$-P_-(ij) \sum_m^{occ} t_{im}^{ab} \left(F_{mj} + \frac{1}{2} \sum_e^{virt} t_j^e F_{me} \right) + \frac{1}{2} \sum_{m,n}^{occ,occ} \tau_{mn}^{ab} W_{mnij}$$
$$+P_-(ij) P_-(ab) \sum_{m,e}^{occ,virt} (t_{im}^{ae} W_{mbej} - t_i^e t_m^a \langle mb \,||\, ej \rangle)$$
$$+P_-(ij) \sum_e^{virt} t_i^e \langle ab \,||\, ej \rangle - P_-(ab) \sum_m^{occ} t_m^a \langle mb \,||\, ij \rangle$$

Here:

$$D_i^a = f_{ii} - f_{aa}$$

$$D_{ij}^{ab} = f_{ii} + f_{jj} - f_{aa} - f_{bb}$$

It can be noted that the T1 and T2 equations depends on T1 and T2. Thus it have to be solver iteratively. The initial guess is given as:

$$t_i^a = 0$$

$$t_{ij}^{ab} = \frac{\langle ij \,||\, ab \rangle}{D_{ij}^{ab}}$$

The CCSD energy is then found as:

$$E_{\mathrm{CCSD}} = \sum_{i,a}^{occ,virt} f_{ia} t_i^a + \sum_{i,j,a,b}^{occ,occ,virt,virt} \langle ij \,||\, ab \rangle \left(\frac{1}{4} t_{ij}^{ab} + \frac{1}{2} t_i^a t_j^b \right)$$

FUNCTION:

- CC.CCSD(occ, F, C, VeeMOspin, maxiter, deTHR, rmsTHR, runCCSDT=0)
- return EMP2, ECCSD

Input:

- F, fock matrix in spatial basis
- C, MO coeffcients in spatial basis
- VeeMOspin, two electron integrals in spinbasis
- deTHR, change in energy check for convergence

- rmsTHR, check for change in T1 and T2
- runCCSDT, 0 for CCSD and 1 for CCSD(T)

Output:

- EMP2, MP2 energy
- ECCSD, CCSD energy

References:

- http://sirius.chem.vt.edu/wiki/doku.php?id=crawdad:programming:project5
- J.F. Stanton, J. Gauss, J.D. Watts, and R.J. Bartlett, J. Chem. Phys. volume 94, pp. 4334-4345 (1991)

15.2 Perturbative Triples Correction

To find the perturbative triples correction, the disconnected and connected T3 have to be calculated. The disconnected is found as:

$$D_{ijk}^{abc} t_{ijk,\text{disconnected}}^{abc} = P\left(i/jk\right) P\left(a/bc\right) t_i^a \left\langle jk \,||\, bc \right\rangle$$

And the connected is found as:

$$D_{ijk}^{abc} t_{ijk,\text{connected}}^{abc} = P\left(i/jk\right) P\left(a/bc\right) \left[\sum_e^{virt} t_{jk}^{ae} \left\langle ei \,||\, bc \right\rangle - \sum_m^{occ} t_{im}^{bc} \left\langle ma \,||\, jk \right\rangle \right]$$

In the above equations the following definitions is used:

$$D_{ijk}^{abc} = f_{ii} + f_{jj} + f_{kk} - f_{aa} - f_{bb} - f_{cc}$$

$$P\left(i/jk\right) f\left(i,j,k\right) = f\left(i,j,k\right) - f\left(j,i,k\right) - f\left(k,j,i\right)$$

The energy correction can now be found as:

$$E_{(\mathrm{T})} = \frac{1}{36} \sum_{i,j,k,a,b,c}^{occ,occ,occ,virt,virt,virt} t_{ijk,\text{connected}}^{abc} D_{ijk}^{abc} \left(t_{ijk,\text{connected}}^{abc} + t_{ijk,\text{disconnected}}^{abc} \right)$$

FUNCTION:

- see CCSD function above
- return EMP2, ECCSD, ET

Input:

- runCCSDT=1

Output:

- EMP2, MP2 energy
- ECCSD, CCSD energy
- ET, perturbative triples corrections

References:

- http://sirius.chem.vt.edu/wiki/doku.php?id=crawdad:programming:project6

CHAPTER 16

Ab-Initio Molecular Dynamics

In this section, working equations for AIMD is presented. For the Forces see Geometry Optimization section.

16.1 Velocity Verlet integrator

The Velocity Verlet is given is three steps. First the position is updated:

$$x\left(t+\Delta t\right)=x\left(t\right)+v\left(t\right)\Delta t+\frac{1}{2}a\left(t\right)\Delta t^{2}$$

Then the forces are calculated. At last the velocties are updated:

$$v\left(t+\Delta t\right)=v\left(t\right)+\frac{1}{2}\left[a\left(t\right)+a\left(t+\Delta t\right)\right]\Delta t$$

FUNCTION:

- runBOMD.VelocityVerlet(inputBOMD, dt, results, set)

Input:

- inputBOMD, inputfile object for BOMD
- dt, integration step size
- results, results object
- set, settings object

Output:

- inputBOMD, inputfile object for BOMD
- results, results object

References:

- None