SQ2EOM

Sahil Gulania[a]

[a] Department of Chemistry, University of Southern California, Los Angeles, California 90089, USA

February 1, 2021

1 From Second Quantization to Equation-of-Motion Coupled-Cluster using SymPy

1.1 Table of contents

1. Introduction
2. Second Quantization
3. Normal product
4. Contraction
5. Wicks theorem
6. Particle-Hole formalism
7. Hartree-Fock
8. Baker-Campbell-Hausdorff
9. Coupled-Cluster
10. Equation-of-motion Coupled-Cluster

1.1.1 Introduction

We will have hands on tutorial for the derivation of EOM-CCSD amplitudes, one and two particle density matrix. I have developed a symbolic library using SymPy for deriving analytical expressions which can be easily extended to any operator.

First, we will derive the fermionic algebra from first quantization and study their properties.

1.1.2 Second Quantization

In this section we will derive the fermionic algebra in second quantization representation from first quantization. Once the fermionic algebra is established, we will derive the relation between the its elements

$$|k\rangle = a_k^+ |vac\rangle$$

The opertor a_k^+ has created an electron on the vaccum state.

$$\phi_k(1) \longleftrightarrow a_k^+ |vac\rangle$$

$|vac\rangle$ is an abstract vaccum state and $\langle vac\rangle\, vac = 1$ If there are two electrons the wavefunction in first quantization is

$$\Phi(1,2) = \frac{1}{\sqrt{2}} \begin{vmatrix} \phi_i(1) & \phi_k(1) \\ \phi_i(2) & \phi_k(2) \end{vmatrix}$$

The same in second quantization follows as (keeping normalization included)

$$|ik\rangle = a_i^+ a_k^+ |vac\rangle$$

As $\Phi(1,2)$ is an antisymmetric function therefore the creation operator in second quantization should respect this antisymmetry.

$$\Phi(2,1) = -\Phi(1,2) \longleftrightarrow a_k^+ a_i^+ |vac\rangle = -a_i^+ a_k^+ |vac\rangle$$

One can see that the antisymmetry requirement is fulfilled by having the anticommutation relation between creation operators.

$$a_k^+ a_i^+ + a_i^+ a_k^+ = \{a_k^+, a_i^+\} = 0 \tag{QED 1}$$

Now we will repeat the same excercice for anhilation operators. Consider first one-electron system by creating an electron on ith orbital using a_i^+. We can define the anhilator operator a_i, which act on the former state and returns back the system to $|vac\rangle$.

$$a_i a_i^+ |vac\rangle = |vac\rangle$$

When there is no electron in a orbital i, then it should result to zero.

$$a_i |vac\rangle = 0$$

In order to establish the anticommutation of anhilation operators we will utilize anticommutation relation of creation operators.

$$a_i a_k |ki\rangle = |vac\rangle \quad \& \quad a_k a_i |ik\rangle = |vac\rangle = -a_k a_i |ki\rangle$$

Therefore,

$$a_i a_k + a_k a_i = \{a_k, a_i\} = 0 \tag{QED 2}$$

Now we already have two anticommutation relation and will try to derive the third, between creation and anhilation operators. Consider two orbitals i & k, where $i \neq k$

$$a_i a_k^+ |i\rangle = a_i |ki\rangle = -a_i |ik\rangle = -|k\rangle$$

Reversing the action,

$$a_k^+ a_i |i\rangle = |k\rangle$$

Comparing both

$$a_k^+ a_i + a_i a_k^+ = \{a_i, a_k^+\} = 0$$

But when $i = k$, then

$$a_i^+ a_i + a_i a_i^+ = \{a_i, a_i^+\} = 1$$

Therefore the final expression which captures both sitation is

$$a_i^+ a_k + a_k a_i^+ = \{a_i^+, a_k\} = \delta_{ik} \qquad \text{(QED 3)}$$

Last, we need to establish the adjoint relation between anhilation and creation operator. Consider From first quantization

$$\langle \phi_i \rangle \, \phi_k = \delta_{ik}$$

Using second quantization on right hand side where † indicates hermitian conjugate

$$\langle vac| \, (a_i^+)^\dagger a_k^+ \, |vac\rangle = \delta_{ik}$$

It is only true if

$$(a_i^+)^\dagger = a_i \qquad \text{(QED 4)}$$

Hence, $a_i^+ = a_i^\dagger$ which means creation operator for ith orbital is hermitian conjugate of anhilation operator for the same orbital. From now on for creation operator we will use † as superscript instead of $+$

1.1.3 Particle number operator

We succesfully showed the transformation of basis from first quantization to second quantization by creating fermionic algebra. Now we will do the same for operators. Lets look at number operator for N fermions in N orthogonal basis. The wavefuntion can be written as

$$|\Psi_N\rangle = a_N^\dagger a_{N-1}^\dagger ... a_2^\dagger a_1^\dagger \, |vac\rangle$$

Define number operator for kth orbital

$$a_k^\dagger a_k$$

Act this opeartor on $|\Psi_N\rangle$

$$a_k^\dagger a_k \, |\Psi_N\rangle = a_k^\dagger a_k (a_N^\dagger a_{N-1}^\dagger ... a_k^\dagger ... a_2^\dagger a_1^\dagger \, |vac\rangle)$$

Using eq.(1) and eq.(3), both $a_k^\dagger a_k$ can move next to $a_k^\dagger$ inside Ψ_N

$$a_k^\dagger a_k \, |\Psi_N\rangle = a_N^\dagger a_{N-1}^\dagger ... a_k^\dagger a_k a_k^\dagger ... a_2^\dagger a_1^\dagger \, |vac\rangle = a_N^\dagger a_{N-1}^\dagger ... a_k^\dagger ... a_2^\dagger a_1^\dagger \, |vac\rangle = +1 \, |\Psi_N\rangle$$

Particle number operator for N particles

$$\sum_{i=1}^{N} a_i^\dagger a_i \left|\Psi_N\right\rangle = N \left|\Psi_N\right\rangle$$

1.1.4 Parity of permutation

$$pqrs \xrightarrow{(12)(34)} qpsr$$

The total number of transposition required to decompose any permutation P is known as its parity,

$\surd$

If the total number is even then its called even parity

If the total number is odd then its called odd parity

Sign of permutation P is defined as

$$sgn(P) = -1^p$$

Hence,

$$sgn(P_{odd}) = -1 \qquad\qquad sgn(P_{even}) = 1$$

1.1.5 Normal product

Given the product of operators, the normal product is defined as rearrangment of operators such that all the creation operator are moved to left and all the anhilation operator to right, then multiply it by sign of permutation.

Example: Consiper $Q = a_p^\dagger a_q a_r^\dagger$

Now, the in order to write normal product of Q, one has to permute $a_q a_r^\dagger$ and multiply thw whole by -1 ($\because$ odd parity permutation).

$$n[a_p^\dagger a_q a_r^\dagger] = -a_p^\dagger a_r^\dagger a_q$$

Excercise: $Q = a_s a_p^\dagger a_r a_q^\dagger$

1) $n[Q]$
2) $n[n[Q]]$ (nomal product is idempotent operator)

Solution: $Q = a_s a_p^\dagger a_r a_q^\dagger$

1) $n[Q] = -a_q^\dagger a_p^\dagger a_r a_s = a_p^\dagger a_q^\dagger a_r a_s$
2) $n[n[Q]] = n[Q]$

Now, we will compute normal product using SymPy

```
[1]: from secondquant_gen import AnnihilateFermion, CreateFermion, NO
     from sympy import symbols

     p,q,r,s = symbols('p,q,r,s')

     Q = AnnihilateFermion(s)*CreateFermion(p)*AnnihilateFermion(r)*CreateFermion(q)

     display(Q)

     n = NO(Q)
     nn = NO(n)

     display(n,nn)
```

$a_s a_p^\dagger a_r a_q^\dagger$

$\left\{a_p^\dagger a_q^\dagger a_r a_s\right\}$

$\left\{a_p^\dagger a_q^\dagger a_r a_s\right\}$

1.1.6 Contraction

Contraction between two operators is defined as itself minus its normal product. Symbolically

$$\overbracket{M_1 M_2} = M_1 M_2 - n[M_1 M_2]$$

Example: Compute contraction of $Q = a_p a_q$

$$\overbracket{a_p a_q} = a_p a_q - n[a_p a_q] = 0$$

Excercise: Compute contraction of the following

1) $a_p a_q^\dagger$
2) $a_p^\dagger a_q$
3) $a_p^\dagger a_q^\dagger$

$$\begin{aligned}\overbracket{a_p a_q^\dagger} &= a_p a_q^\dagger - n[a_p a_q^\dagger] \\ &= a_p a_q^\dagger - (-a_q^\dagger a_p) \\ &= \{a_p a_q^\dagger\} = \delta_{pq}\end{aligned}$$

$$\begin{aligned}\overbracket{a_p^\dagger a_q} &= a_p^\dagger a_q - n[a_p^\dagger a_q] \\ &= a_p^\dagger a_q - a_p^\dagger a_q = 0\end{aligned}$$

$$\begin{aligned}\overbracket{a_p^\dagger a_q^\dagger} &= a_p^\dagger a_q^\dagger - n[a_p^\dagger a_q^\dagger] \\ &= a_p^\dagger a_q^\dagger - a_p^\dagger a_q^\dagger = 0\end{aligned}$$

Solution:

Now, we will compute normal product using SymPy

```
[2]: from secondquant_gen import contraction, F, Fd # F = AnnihilateFermion, Fd =␣
      ↪CreateFermion
     from sympy import symbols


     p,q = symbols('p q')

     C1 = contraction(F(p),F(q))
     C2 = contraction(F(p),Fd(q))
     C3 = contraction(Fd(p),F(q))
     C4 = contraction(Fd(p),Fd(q))

     display(C1,C2,C3,C4)
```

0

δ_{pq}

0

0

1.1.7 Normal products with Contractions

Given normal product of operators with contraction of operators inside the normal product.

Then take all the contractions out in pairs and same order, leave uncontracted operators inside normal product and multiply sign of permutation.

Example:
$$
\begin{aligned}
n[\wick{\c a_p a_q \c a_r^\dagger}] &= \wick{\c a_p \c a_r^\dagger} n[a_q] \\
&= -\delta_{pr} a_q
\end{aligned}
$$

$$n[\wick{\c1 a_p \c2 a_q \c1 a_r^\dagger a_s \c2 a_t^\dagger a_u^\dagger}] = sgn(P)\wick{\c a_p \c a_r^\dagger}\wick{\c a_q \c a_t^\dagger} n[a_s a_u^\dagger]$$

where P

$$pqrstu \xrightarrow{P} prqtsu$$

Therefore, sgn(P) $= 1$

Example:
$$
\begin{aligned}
n[\wick{\c1 a_p \c2 a_q \c1 a_r^\dagger a_s \c2 a_t^\dagger a_u^\dagger}] &= \delta_{pr}\delta_{qt} n[a_s a_u^\dagger] \\
&= -\delta_{pr}\delta_{qt} a_u^\dagger a_s
\end{aligned}
$$

1.1.8 Wicks Theorem

Given a product of operators then it can be rewritten as normal product of the same plus normal product with all possible contractions inside.

Example:

$$
\begin{aligned}
a_p a_q^\dagger a_r^\dagger &= n[a_p a_q^\dagger a_r^\dagger] + n[\wick{\c a_p \c a_q^\dagger a_r^\dagger}] + n[\wick{\c a_p a_q^\dagger \c a_r^\dagger}] + \cancel{n[\wick{a_p \c a_q^\dagger \c a_r^\dagger}]} \\
&= a_q^\dagger a_r^\dagger a_p + \wick{\c a_p \c a_q^\dagger} n[a_r^\dagger] + \wick{\c a_p \c a_r^\dagger} n[a_q^\dagger] \\
&= a_q^\dagger a_r^\dagger a_p + \delta_{pq} a_r^\dagger - \delta_{pr} a_q^\dagger
\end{aligned}
$$

Would you like to do it with hand or using computer?

$$a_p a_q a_r^\dagger a_s a_t^\dagger a_u^\dagger = ???$$

Now, we will show use of wicks theorem using SymPy

[3]:
```
from secondquant_gen import wicks, F, Fd
from sympy import symbols, Dummy


p,q,r,s,t,u = symbols('p q r s t u')

E1 = wicks(F(p)*Fd(q)*Fd(r))
display(E1)
```

$\delta_{pq} a_r^\dagger - \delta_{pr} a_q^\dagger + \left\{ a_q^\dagger a_r^\dagger a_p \right\}$

[4]:
```
E2 = wicks(F(p)*F(q)*Fd(r)*F(s)*Fd(t)*Fd(u))
display(E2)
```

$$\delta_{pr}\delta_{qt}\delta_{su} - \delta_{pr}\delta_{qt}\left\{a^\dagger_u a_s\right\} - \delta_{pr}\delta_{qu}\delta_{st} + \delta_{pr}\delta_{qu}\left\{a^\dagger_t a_s\right\} + \delta_{pr}\delta_{st}\left\{a^\dagger_u a_q\right\} - \delta_{pr}\delta_{su}\left\{a^\dagger_t a_q\right\} - \delta_{pr}\left\{a^\dagger_t a^\dagger_u a_q a_s\right\} - \delta_{pt}\delta_{qr}\delta_{su} + \delta_{pt}\delta_{qr}\left\{a^\dagger_u a_s\right\} - \delta_{pt}\delta_{qu}\left\{a^\dagger_r a_s\right\} + \delta_{pt}\delta_{su}\left\{a^\dagger_r a_q\right\} + \delta_{pt}\left\{a^\dagger_r a^\dagger_u a_q a_s\right\} + \delta_{pu}\delta_{qr}\delta_{st} - \delta_{pu}\delta_{qr}\left\{a^\dagger_t a_s\right\} + \delta_{pu}\delta_{qt}\left\{a^\dagger_r a_s\right\} - \delta_{pu}\delta_{st}\left\{a^\dagger_r a_q\right\} - \delta_{pu}\left\{a^\dagger_r a^\dagger_t a_q a_s\right\} - \delta_{qr}\delta_{st}\left\{a^\dagger_u a_p\right\} + \delta_{qr}\delta_{su}\left\{a^\dagger_t a_p\right\} + \delta_{qr}\left\{a^\dagger_t a^\dagger_u a_p a_s\right\} - \delta_{qt}\delta_{su}\left\{a^\dagger_r a_p\right\} - \delta_{qt}\left\{a^\dagger_r a^\dagger_u a_p a_s\right\} + \delta_{qu}\delta_{st}\left\{a^\dagger_r a_p\right\} + \delta_{qu}\left\{a^\dagger_r a^\dagger_t a_p a_s\right\} + \delta_{st}\left\{a^\dagger_r a^\dagger_u a_p a_q\right\} - \delta_{su}\left\{a^\dagger_r a^\dagger_t a_p a_q\right\} + \left\{a^\dagger_r a^\dagger_t a^\dagger_u a_p a_q a_s\right\}$$

1.1.9 Expectation value with $|vac\rangle$

Let $Q = M_1 M_2 M_3 ... M_{n-1} M_n$ product of n operators

1. $n[Q]\,|vac\rangle = 0$: unless all operators in Q are creation operator
2. $\langle vac|\, n[Q]\,|vac\rangle = 0$
3. $\langle vac|\, n[M_1 M_2 M_3 .... M_{n-1} M_n]\,|vac\rangle = 0$: unless all are contracted
4. If $n =$ odd, then $\langle vac|\, n[Q]\,|vac\rangle = 0$

From this one can easily see the importance of Wick's theorem. Lets do some checks using SymPy

Exercise:

$Q = a_p a^\dagger_q a^\dagger_r$

$Q = a_p a_q a^\dagger_r a_s a^\dagger_t a^\dagger_u$

Compute $\langle vac|\, Q\,|vac\rangle$?

Solution:

1. Conver Q to normal order using wicks theorem.
2. Only terms which are fully contracted will result in non-zero

[5]:
```
from secondquant_gen import wicks, F, Fd
from sympy import symbols


p,q,r,s,t,u = symbols('p q r s t u')

E1 = wicks(F(p)*Fd(q)*Fd(r),keep_only_fully_contracted=True)
display(E1)
```

0

[6]:
```
E2 = wicks(F(p)*F(q)*Fd(r)*F(s)*Fd(t)*Fd(u),keep_only_fully_contracted=True)
display(E2)
```

$\delta_{pr}\delta_{qt}\delta_{su} - \delta_{pr}\delta_{qu}\delta_{st} - \delta_{pt}\delta_{qr}\delta_{su} + \delta_{pu}\delta_{qr}\delta_{st}$

1.1.10 Particle Hole formalism

This procedure makes the evaluation of certain matrix elements easier like

$$\langle\Phi_0|\,\hat{H}\,|\Phi_0\rangle$$
$$\langle\Phi_i^a|\,\hat{H}\,|\Phi_0\rangle$$
$$\langle\Phi_{ij}^{ab}|\,\hat{H}\,|\Phi_0\rangle$$

where, Φ_0 is Hartree-Fock determinant. Given a Hartree-Fock solution we get occupied and virtual orbitals. Lets assign symbols $(i, j, k, ...)$ to occupied and $(a, b, c, ...)$ to virtual.

We will define new set of operators

$$b_p = \begin{cases} a_p^\dagger & \text{if } p = i \\ a_p & \text{if } p = a \end{cases}$$

$$b_p^\dagger = \begin{cases} a_p & \text{if } p = i \\ a_p^\dagger & \text{if } p = a \end{cases}$$

This makes the HF determinant a Fermi vaccum

$$b_p\,|\Phi_0\rangle = 0$$

This follows

$$\{b_p, b_q\} = 0 \qquad \{b_p^\dagger, b_q^\dagger\} = 0 \qquad \{b_p, b_q^\dagger\} = \delta_{pq}$$

1.1.11 Normal product of PHF

It is defined similarly, as rearrangment of operators (b_p & $b_q^\dagger$) such that all the creation operator are moved to left and all the anhilation operator to right, then multiply it by sign of permutation.

Example: Consiper $R = b_p^\dagger b_q b_r^\dagger$

Now, the in order to write normal product of R, one has to permute $b_q b_r^\dagger$ and multiply thw whole by -1 ($\because$ odd parity permutation).

$$N[b_p^\dagger b_q b_r^\dagger] = -b_p^\dagger b_r^\dagger b_q$$

The thing one has to focus is this definition shows different result that $n[]$. Lets do an excercise to explain this.

We are going to compute normal order of $a_p^\dagger a_q a_r^\dagger$ for $p = i, q = j, r = a$

$$N[a_i^\dagger a_j a_a^\dagger] = N[b_i b_j^\dagger b_a^\dagger] = b_j^\dagger b_a^\dagger b_i = -b_a^\dagger b_j^\dagger b_i = -a_a^\dagger a_j a_i^\dagger$$

Also compute

$$n[a_i^\dagger a_j a_a^\dagger] = -a_i^\dagger a_a^\dagger a_j = a_a^\dagger a_i^\dagger a_j = a_a^\dagger \delta_{ij} - a_a^\dagger a_j^\dagger a_i$$

```
[7]: from sympy.physics.secondquant import NO, F, Fd # F = AnnihilateFermion, Fd = CreateFermion
from sympy import symbols


i,j = symbols('i j', below_fermi=True)
a = symbols('a', above_fermi=True)

display(NO(Fd(i)*F(j)*Fd(a)))
```

$-\left\{a_a^\dagger a_j a_i^\dagger\right\}$

1.1.12 Contractions in PHF

Same way they are defined but the operators are from PHF

$$\wick{\c M_1 \c M_2} = M_1 M_2 - N[M_1 M_2]$$

Similarlry the contraction inside normal product, Wicks theorem are defined.

1.1.13 Expectation value with $|\Phi_0\rangle$

Let $R = M_1 M_2 M_3 ... M_{n-1} M_n$ product of n operators

1. $N[R]\,|\Phi_0\rangle = 0$: unless all operators in Q are creation operator
2. $\langle\Phi_0|\, N[R]\,|\Phi_0\rangle = 0$
3. $\langle\Phi_0|\, N[\wick{\c1 M_1 M_2 M_3 . \c2 . \c1 . ... \c2 M_{n-1} M_n}]\,|\Phi_0\rangle = 0$: unless all are contracted
4. If $n =$ odd, then $\langle\Phi_0|\, N[R]\,|\Phi_0\rangle = 0$

From this one can easily see the importance of Wick's theorem. Lets do some checks using SymPy and derive Hartree-Fock energy

1.1.14 Electronic Hamiltonian

$$H_{el} = \sum_{pq} h_{pq} a_p^\dagger a_q + \frac{1}{4} \sum_{pqrs} v_{rs}^{pq} a_p^\dagger a_q^\dagger a_s a_r$$

where,

$$h_{pq} = \langle \phi_p(x) | \left(-\frac{1}{2}\nabla^2 - \sum_\alpha \frac{Z_\alpha}{r_{\alpha x}}\right) | \phi_q(x) \rangle$$

$$v_{pqrs} = \langle \phi_p \phi_q | | \phi_r \phi_s \rangle = \langle \phi_p \phi_q \rangle \phi_r \phi_s - \langle \phi_p \phi_q \rangle \phi_s \phi_r$$

1.1.15 Hartree-Fock

Lets find Hartree-Fock energy

[8]:
```
from sympy.physics.secondquant import wicks, F, Fd, NO  # F = AnnihilateFermion,␣
 ↪Fd = CreateFermion
from sympy.physics.secondquant import substitute_dummies, contraction
from sympy.physics.secondquant import AntiSymmetricTensor, evaluate_deltas
from sympy import symbols, Rational, Dummy

p, q, r, s = symbols('p,q,r,s', cls=Dummy)

h = AntiSymmetricTensor('h', (p,), (q,))
pq = Fd(p)*F(q)
v = AntiSymmetricTensor('v', (p, q), (r, s))
pqsr = Fd(p)*Fd(q)*F(s)*F(r)

H = h*pq + Rational(1, 4)*v*pqsr
display(H) # Without summation                                                  ␣
 ↪

eq = wicks(H,keep_only_fully_contracted=True)
index_rule = {'below':  'ijklmno','above':  'abcde', 'general': 'pqrs'}
eq = substitute_dummies(eq, new_indices=True, pretty_indices=index_rule)
display(eq)

eq = evaluate_deltas(eq)
display(eq)
```

$$h_q^p a_p^\dagger a_q + \frac{v_{rs}^{pq} a_p^\dagger a_q^\dagger a_s a_r}{4}$$

$$\delta_{ip}\delta_{pq} h_p^q + \frac{\delta_{iq}\delta_{jp}\delta_{pr}\delta_{qs} v_{pq}^{rs}}{2}$$

$$h_i^i + \frac{v_{ij}^{ij}}{2}$$

1.1.16 Normal ordered Hamiltonian

$$H_N = \sum_{pq} h_{pq} N[a_p^\dagger a_q] + \frac{1}{4} \sum_{pqrs} v_{rs}^{pq} N[a_p^\dagger a_q^\dagger a_s a_r]$$

One can show that

$$H_{el} = H_N + E_{HF}$$

From now on we will use H_N or computing matrix elements, also H_N measures the correlation effect

1.1.17 Baker-Campbell-Hausdorff

$$e^{-B} A e^{B} = A + B + [A, B] + \frac{1}{2!}[[A, B], B] + \frac{1}{3!}[[[A, B], B], B] + ...$$

if A is electronic Hamiltonian or normal ordered Hamiltonian and B any excitation operator, then due to the stucture of Hamiltonian the series truncate after fourth expansion. This is the reason we call coupled cluster as coupled cluster.

1.1.18 Coupled-Cluster

Coupled Cluster Singles and Doubles (CCSD)

$$\Psi_{CCSD} = e^{T} |\Phi_0\rangle$$

where,

$$T_1 = \sum_{ia} t_i^a a^\dagger i$$

$$T_2 = \frac{1}{4} \sum_{ijab} t_{ij}^{ab} a^\dagger b^\dagger ji$$

Equations which need to be solved

$$E_{CCSD}^{cor} = \langle\Phi_0| e^{-T} H_N e^{T} |\Phi_0\rangle \qquad \text{(Correltion energy)}$$

$$0 = \langle\Phi_i^a| e^{-T} H_N e^{T} |\Phi_0\rangle \qquad (T_1 \text{ amplitude})$$

$$0 = \langle\Phi_{ij}^{ab}| e^{-T} H_N e^{T} |\Phi_0\rangle \qquad (T_2 \text{ amplitude})$$

Lets see the power of SymPy now,

[9]:
```
from sympy.physics.secondquant import wicks, F, Fd, NO   # F = AnnihilateFermion,␣
 ↪Fd = CreateFermion
from sympy.physics.secondquant import substitute_dummies, contraction
from sympy.physics.secondquant import AntiSymmetricTensor, evaluate_deltas
from sympy.physics.secondquant import PermutationOperator,␣
 ↪simplify_index_permutations
from sympy import symbols, Rational, Dummy
import BCH

p, q, r, s = symbols('p,q,r,s', cls=Dummy)

i,j = symbols('i,j' , below_fermi=True)
a,b = symbols('a,b' , above_fermi=True)

f = AntiSymmetricTensor('f', (p,), (q,))
pq = NO(Fd(p)*F(q))
v = AntiSymmetricTensor('v', (p, q), (r, s))
pqsr = NO(Fd(p)*Fd(q)*F(s)*F(r))

H = f*pq + Rational(1, 4)*v*pqsr

ccsd = BCH.level(H,"SD")

eq = wicks(ccsd,simplify_kronecker_deltas=True,keep_only_fully_contracted=True)
index_rule = {'below':  'ijklmno','above':  'abcdef', 'general': 'pqrs'}
e_ccsd = substitute_dummies(eq, new_indices=True, pretty_indices=index_rule)
display(e_ccsd)
```

$$f^{i}_{a} t^{a}_{i} - \frac{t^{a}_{j} t^{b}_{i} v^{ij}_{ab}}{2} + \frac{t^{ab}_{ij} v^{ij}_{ab}}{4}$$

[10]:
```
eq =␣
 ↪wicks(Fd(i)*F(a)*ccsd,simplify_kronecker_deltas=True,keep_only_fully_contracted=True)
index_rule = {'below':  'jklmno','above':  'bcdef', 'general': 'pqrs'}
t1 = (substitute_dummies(eq, new_indices=True, pretty_indices=index_rule))
display(t1)
```

$$-f^{j}_{b} t^{b}_{i} t^{a}_{j} + f^{j}_{b} t^{ab}_{ij} - f^{j}_{i} t^{a}_{j} + f^{a}_{b} t^{b}_{i} + f^{a}_{i} - t^{b}_{j} t^{c}_{i} t^{a}_{k} v^{jk}_{bc} - t^{b}_{j} t^{c}_{i} v^{aj}_{bc} + t^{b}_{j} t^{a}_{k} v^{jk}_{ib} + t^{b}_{j} t^{ac}_{ik} v^{jk}_{bc} + t^{b}_{j} v^{aj}_{ib} - \frac{t^{b}_{i} t^{ac}_{jk} v^{jk}_{bc}}{2} + \frac{t^{a}_{k} t^{bc}_{ij} v^{jk}_{bc}}{2} + \frac{t^{bc}_{ij} v^{aj}_{bc}}{2} - \frac{t^{ab}_{jk} v^{jk}_{ib}}{2}$$

[11]:
```
eq =␣
 ↪wicks(Fd(i)*Fd(j)*F(b)*F(a)*ccsd,simplify_kronecker_deltas=True,keep_only_fully_contracted=Tr
index_rule = {'below':  'klmno','above':  'cdef', 'general': 'pqrs'}
t2 = substitute_dummies(eq, new_indices=True, pretty_indices=index_rule)
P = PermutList = [PermutationOperator(i,j),PermutationOperator(a,b)]
t2 = simplify_index_permutations(t2,PermutList)
display(t2)
```

$$f^{k}_{c} t^{c}_{i} t^{ab}_{jk} P(ij) + f^{k}_{c} t^{a}_{k} t^{bc}_{ij} P(ab) + f^{k}_{i} t^{ab}_{jk} P(ij) - f^{a}_{c} t^{bc}_{ij} P(ab) + t^{c}_{k} t^{d}_{i} t^{ab}_{jl} v^{kl}_{cd} P(ij) + t^{c}_{k} t^{a}_{l} t^{bd}_{ij} v^{kl}_{cd} P(ab) - t^{c}_{k} t^{ad}_{ij} v^{bk}_{cd} P(ab) + t^{c}_{k} t^{ab}_{il} v^{kl}_{jc} P(ij) + t^{c}_{i} t^{d}_{j} t^{a}_{k} t^{b}_{l} v^{kl}_{cd} + t^{c}_{i} t^{d}_{j} t^{a}_{k} v^{bk}_{cd} P(ab) + \frac{t^{c}_{i} t^{d}_{j} t^{ab}_{kl} v^{kl}_{cd}}{2} + t^{c}_{i} t^{d}_{j} v^{ab}_{cd} - t^{c}_{i} t^{a}_{k} t^{b}_{l} v^{kl}_{jc} P(ij) - t^{c}_{i} t^{a}_{k} t^{bd}_{jl} v^{kl}_{cd} P(ab) P(ij) - t^{c}_{i} t^{a}_{k} v^{bk}_{jc} P(ab) P(ij) - t^{c}_{i} t^{ad}_{jk} v^{bk}_{cd} P(ab) P(ij) - \frac{t^{c}_{i} t^{ab}_{kl} v^{kl}_{jc} P(ij)}{2} - t^{c}_{i} v^{ab}_{jc} P(ij) + \frac{t^{a}_{k} t^{b}_{l} t^{cd}_{ij} v^{kl}_{cd}}{2} + t^{a}_{k} t^{b}_{l} v^{kl}_{ij} + \frac{t^{a}_{k} t^{cd}_{ij} v^{bk}_{cd} P(ab)}{2} + t^{a}_{k} t^{bc}_{il} v^{kl}_{jc} P(ab) P(ij) + t^{a}_{k} v^{bk}_{ij} P(ab) + \frac{t^{cd}_{ij} t^{ab}_{kl} v^{kl}_{cd}}{4} + \frac{t^{cd}_{ij} v^{ab}_{cd}}{2} + \frac{t^{cd}_{jk} t^{ab}_{il} v^{kl}_{cd} P(ij)}{2} + t^{ac}_{ik} t^{bd}_{jl} v^{kl}_{cd} P(ij) + t^{ac}_{ik} v^{bk}_{jc} P(ab) P(ij) - \frac{t^{ac}_{ij} t^{bd}_{kl} v^{kl}_{cd} P(ab)}{2} + \frac{t^{ab}_{kl} v^{kl}_{ij}}{2} + v^{ab}_{ij}$$

1.1.19 Equation-of-motion Coupled-Cluster

EOM-CC allows to compute excited and open-shell character electronic states. There are many flavor depending on the target state from reference.

Energy and amplitude equations of EOM-CC

$$\bar{H} = e^{-T} H_N e^{T}$$

$$\langle \Phi^{ab..}_{ij..} | \bar{H} \hat{R} | \Phi_0 \rangle = E_{EOM} \langle \Phi^{ab..}_{ij..} | \hat{R} | \Phi_0 \rangle$$

$$\langle \Phi^{ab..}_{ij..} | [\bar{H} - E_{cc}, \hat{R}] | \Phi_0 \rangle = \Delta E_{EOM} \langle \Phi^{ab..}_{ij..} | \hat{R} | \Phi_0 \rangle$$

Last equation is the one which we will derive using SymPy

Example - EOM-IP-CCSD using genralized Davidson method

$$\begin{pmatrix} \bar{H}_{SS} - E_{cc} & \bar{H}_{SD} \\ \bar{H}_{DS} & \bar{H}_{DD} - E_{cc} \end{pmatrix} \begin{pmatrix} R_1 \\ R_2 \end{pmatrix} = \omega \begin{pmatrix} R_1 \\ R_2 \end{pmatrix}$$

and

$$\begin{pmatrix} L_1 & L_2 \end{pmatrix} \begin{pmatrix} \bar{H}_{SS} - E_{cc} & \bar{H}_{SD} \\ \bar{H}_{DS} & \bar{H}_{DD} - E_{cc} \end{pmatrix} = \omega \begin{pmatrix} L_1 & L_2, \end{pmatrix}$$

Using matrix equations

Need right σ amplitudes of EOM-IP-CCSD amplitudes and for this we need

$$\langle \Phi_i | [\bar{H} - E_{cc}, \hat{R}_1] | \Phi_0 \rangle = ((\bar{H}_{SS} - E_{cc}) R_1)$$

$$\langle \Phi^{a}_{ij} | [\bar{H}, \hat{R}_1] | \Phi_0 \rangle = (\bar{H}_{DS} R_1)$$

$$\langle \Phi_i | [\bar{H}, \hat{R}_2] | \Phi_0 \rangle = (\bar{H}_{SD} R_2)$$

$$\langle \Phi_{ij}^{a} | [\bar{H} - E_{cc}, \hat{R}_2] | \Phi_0 \rangle = ((\bar{H}_{DD} - E_{cc}) R_2)$$

Therefore, trial vectors are defined as

$$\sigma_1 = ((\bar{H}_{SS} - E_{cc})R_1) + (\bar{H}_{SD}R_2)$$
$$\sigma_2 = (\bar{H}_{DS}R_1) + ((\bar{H}_{DD} - E_{cc})R_2)$$

Similarly, left vectors are defined. Lets use power of SymPy for right σ amplitudes of EOM-IP-CCSD

[12]:
```python
import EOM,SIGMA
flavor1 = "IP"
R0_f1 = EOM.R0(flavor1)
R1_f1 = EOM.R1(flavor1)
R2_f1 = EOM.R2(flavor1)
Rf1   = R0_f1 + R1_f1 + R2_f1

SIGMA.RVECTORS(R0_f1,R1_f1,R2_f1,flavor1)

L0_f1 = EOM.L0(flavor1)
L1_f1 = EOM.L1(flavor1)
L2_f1 = EOM.L2(flavor1)
Lf1   = L0_f1 + L1_f1 + L2_f1

SIGMA.LVECTORS(L0_f1,L1_f1,L2_f1,flavor1)
```

Computing right sigma amplitudes for IP (skipping summation for dummy variables)

$$((\overline{H}_{SS} - E_{cc})R_1) = -f_{a}^{j} r_{j} t_{i}^{a} - f_{i}^{j} r_{j} + r_{j} t_{k}^{a} t_{i}^{b} v_{ab}^{jk} - r_{j} t_{k}^{a} v_{ia}^{jk} - \frac{r_{j} t_{ik}^{ab} v_{ab}^{jk}}{2}$$

$$(\overline{H}_{SD}R_2) = f_{a}^{j} r_{ij}^{a} + \frac{r_{jk}^{a} t_{i}^{b} v_{ab}^{jk}}{2} - \frac{r_{jk}^{a} v_{ia}^{jk}}{2} + r_{ij}^{a} t_{k}^{b} v_{ab}^{jk}$$

$$(\overline{H}_{DS}R_1) = f_{b}^{k} r_{k} t_{ij}^{ab} - r_{k} t_{l}^{b} t_{ij}^{ac} v_{bc}^{kl} + r_{k} t_{i}^{b} t_{j}^{c} t_{l}^{a} v_{bc}^{kl} + r_{k} t_{i}^{b} t_{j}^{c} v_{bc}^{ak} - r_{k} t_{i}^{b} t_{l}^{a} v_{jb}^{kl} - r_{k} t_{i}^{b} t_{jl}^{ac} v_{bc}^{kl} - r_{k} t_{i}^{b} v_{jb}^{ak} + r_{k} t_{j}^{b} t_{l}^{a} v_{ib}^{kl} + r_{k} t_{j}^{b} t_{il}^{ac} v_{bc}^{kl} + r_{k} t_{j}^{b} v_{ib}^{ak} + \frac{r_{k} t_{l}^{a} t_{ij}^{bc} v_{bc}^{kl}}{2} + r_{k} t_{l}^{a} v_{ij}^{kl} + \frac{r_{k} t_{ij}^{bc} v_{bc}^{ak}}{2} + r_{k} t_{il}^{ab} v_{jb}^{kl} - r_{k} t_{jl}^{ab} v_{ib}^{kl} + r_{k} v_{ij}^{ak}$$

$$((\overline{H}_{DD} - E_{cc})R_2) = -f_{b}^{k} r_{ij}^{b} t_{k}^{a} - f_{b}^{k} r_{ik}^{a} t_{j}^{b} P(ij) + f_{i}^{k} r_{jk}^{a} P(ij) + f_{b}^{a} r_{ij}^{b} - \frac{r_{kl}^{b} t_{ij}^{ac} v_{bc}^{kl}}{2} - r_{ik}^{b} t_{j}^{c} t_{l}^{a} v_{bc}^{kl} P(ij) - r_{ik}^{b} t_{j}^{c} v_{bc}^{ak} P(ij) + r_{ik}^{b} t_{l}^{a} v_{jb}^{kl} P(ij) + r_{ik}^{b} t_{jl}^{ac} v_{bc}^{kl} P(ij) + r_{ik}^{b} v_{jb}^{ak} P(ij) + r_{ij}^{b} t_{k}^{c} t_{l}^{a} v_{bc}^{kl} + r_{ij}^{b} t_{k}^{c} v_{bc}^{ak} - \frac{r_{ij}^{b} t_{kl}^{ac} v_{bc}^{kl}}{2} + \frac{r_{kl}^{a} t_{i}^{b} t_{j}^{c} v_{bc}^{kl}}{2} - \frac{r_{kl}^{a} t_{i}^{b} v_{jb}^{kl} P(ij)}{2} + \frac{r_{kl}^{a} t_{ij}^{bc} v_{bc}^{kl}}{4} + \frac{r_{kl}^{a} v_{ij}^{kl}}{2} + r_{ik}^{a} t_{l}^{b} t_{j}^{c} v_{bc}^{kl} P(ij) - r_{ik}^{a} t_{l}^{b} v_{jb}^{kl} P(ij) - \frac{r_{ik}^{a} t_{jl}^{bc} v_{bc}^{kl} P(ij)}{2}$$

Computing left sigma amplitudes for IP (skipping summation for dummy variables)

$$(L_1(\overline{H}_{SS} - E_{cc})) = -f_{a}^{i} l^{j} t_{j}^{a} - f_{j}^{i} l^{j} - l^{j} t_{j}^{a} t_{k}^{b} v_{ab}^{ik} + l^{j} t_{k}^{a} v_{aj}^{ik} - \frac{l^{j} t_{jk}^{ab} v_{ab}^{ik}}{2}$$

$$(L_2\overline{H}_{DS}) = f^a_b l^{ij}_a t^b_j + f^a_j l^{ij}_a - f^j_a l^{ik}_b t^a_k t^b_j + f^j_a l^{ik}_b t^{ab}_{jk} - f^k_j l^{ij}_a t^a_k - \frac{f^i_a l^{jk}_b t^{ab}_{jk}}{2} + l^{jk}_a t^a_l t^b_j v^{il}_{bk} - \frac{l^{jk}_a t^a_l t^b_k t^c_j v^{il}_{bc}}{2} + \frac{l^{jk}_a t^a_l t^{bc}_{jk} v^{il}_{bc}}{4} + \frac{l^{jk}_a t^a_l v^{il}_{jk}}{2} - l^{jk}_a t^b_j v^{ia}_{bk} + \frac{l^{jk}_a t^b_k t^c_j v^{ia}_{bc}}{2} - l^{jk}_a t^c_k t^{ab}_{jl} v^{il}_{bc} + \frac{l^{jk}_a t^c_l t^{ab}_{jk} v^{il}_{bc}}{2} - l^{jk}_a t^{ab}_{jl} v^{il}_{bk} - \frac{l^{jk}_a t^{bc}_{jk} v^{ia}_{bc}}{4} - \frac{l^{jk}_a v^{ia}_{jk}}{2} + l^{ij}_a t^a_k t^b_l v^{kl}_{bj} - l^{ij}_a t^a_l t^b_k t^c_j v^{kl}_{bc} + \frac{l^{ij}_a t^a_l t^{bc}_{jk} v^{kl}_{bc}}{2} + l^{ij}_a t^b_j t^c_k v^{ak}_{bc} - l^{ij}_a t^b_k v^{ak}_{bj} + \frac{l^{ij}_a t^c_j t^{ab}_{kl} v^{kl}_{bc}}{2} + l^{ij}_a t^c_l t^{ab}_{jk} v^{kl}_{bc} + \frac{l^{ij}_a t^{ab}_{kl} v^{kl}_{bj}}{2} + \frac{l^{ij}_a t^{bc}_{jk} v^{ak}_{bc}}{2}$$

$$(L_1\overline{H}_{SD}) = -f^i_a l^j + f^j_a l^i + l^k t^b_k v^{ij}_{ab} + l^k v^{ij}_{ak} + l^i t^b_k v^{jk}_{ab} - l^j t^b_k v^{ik}_{ab}$$

$$(L_2(\overline{H}_{DD} - E_{cc})) = f^b_a l^{ij}_b - f^k_a l^{ij}_b t^b_k + f^i_b l^{jk}_a t^b_k P(ij) + f^i_k l^{jk}_a P(ij) - \frac{l^{kl}_b t^{bc}_{kl} v^{ij}_{ac}}{2} + l^{kl}_a t^b_k v^{ij}_{bl} - \frac{l^{kl}_a t^b_l t^c_k v^{ij}_{bc}}{2} + \frac{l^{kl}_a t^{bc}_{kl} v^{ij}_{bc}}{4} + \frac{l^{kl}_a v^{ij}_{kl}}{2} - l^{ik}_b t^b_l t^c_k v^{jl}_{ac} P(ij) - l^{ik}_b t^b_l v^{jl}_{ak} P(ij) + l^{ik}_b t^c_k v^{jb}_{ac} P(ij) + l^{ik}_b t^{bc}_{kl} v^{jl}_{ac} P(ij) + l^{ik}_b v^{jb}_{ak} P(ij) - l^{ik}_a t^b_k t^c_l v^{jl}_{bc} P(ij) + l^{ik}_a t^b_l v^{jl}_{bk} P(ij) - \frac{l^{ik}_a t^{bc}_{kl} v^{jl}_{bc} P(ij)}{2} - l^{ij}_b t^b_k t^c_l v^{kl}_{ac} + l^{ij}_b t^c_k v^{bk}_{ac} - \frac{l^{ij}_b t^{bc}_{kl} v^{kl}_{ac}}{2}$$

1.1.20 Properties

One particle density matrix (OPDM)

$$\gamma^I_{pq} = \langle \Psi_I | \, p^\dagger q \, | \Psi_I \rangle$$

One particle transition density matrix (OPTDM)

$$\gamma^{IJ}_{pq} = \langle \Psi_I | \, p^\dagger q \, | \Psi_J \rangle$$

[13]:
```
import EOM, DM, TDM
flavor1 = "IP"
R0_f1 = EOM.R0(flavor1)
R1_f1 = EOM.R1(flavor1)
R2_f1 = EOM.R2(flavor1)
Rf1  = R0_f1 + R1_f1 + R2_f1

L0_f1 = EOM.L0(flavor1)
L1_f1 = EOM.L1(flavor1)
L2_f1 = EOM.L2(flavor1)
Lf1  = L0_f1 + L1_f1 + L2_f1
DM.OPDM(Lf1,Rf1,flavor1)
```

Computing OPDM for IP (skipping summation for dummy variables)

$$\gamma_{ij} = \delta_{ij} l^k r_k + \frac{\delta_{ij} l^{kl}_a r^a_{kl}}{2} - l^j r_i + l^{jk}_a r_k t^a_i - l^{jk}_a r^a_{ik}$$

$$\gamma_{ia} = l^j r_j t^a_i - l^j r_i t^a_j - l^j r^a_{ij} - l^{jk}_b r_j t^b_i t^a_k + l^{jk}_b r_j t^{ab}_{ik} - \frac{l^{jk}_b r_i t^{ab}_{jk}}{2} + \frac{l^{jk}_b r^b_{jk} t^a_i}{2} + l^{jk}_b r^b_{ij} t^a_k - \frac{l^{jk}_b r^a_{jk} t^b_i}{2}$$

$$\gamma_{ai} = -l^{ij}_a r_j$$

$$\gamma_{ab} = l^{ij}_{a} r_{i} t^{b}_{j} + \frac{l^{ij}_{a} r^{b}_{ij}}{2}$$

[14]:
```
flavor2 = "CCSD"
R0_f2 = EOM.R0(flavor2)
R1_f2 = EOM.R1(flavor2)
R2_f2 = EOM.R2(flavor2)
Rf2  = R0_f2 + R1_f2 + R1_f2

L0_f2 = EOM.L0(flavor2)
L1_f2 = EOM.L1(flavor2)
L2_f2 = EOM.L2(flavor2)
Lf2  = L0_f2 + L1_f2 + L2_f2

TDM.OPTDM(Lf1,Rf1,Lf2,Rf2,flavor1,flavor2)
```

Computing Dyson OPTDM between IP $\rightarrow$ CCSD (skipping summation for dummy variables)

$$\gamma^{R}_{i} = -l^{j}_{b} r_{j} t^{b}_{i} + l^{j}_{b} r^{b}_{ij} - \frac{l^{jk}_{bc} r_{j} t^{bc}_{ik}}{2} + \frac{l^{jk}_{bc} r^{b}_{jk} t^{c}_{i}}{2} + r_{i}$$

$$\gamma^{R}_{a} = l^{j}_{a} r_{j} + \frac{l^{jk}_{ab} r^{b}_{jk}}{2}$$

$$\gamma^{L}_{i} = l^{i}$$

$$\gamma^{L}_{a} = l^{i} t^{a}_{i} + \frac{l^{ij}_{c} t^{ac}_{ij}}{2}$$