Notes on MRPT

B.Mussard
https://mussard.github.io

May 15, 2019

Contents

1 Main equations **1**
1.1 Perturbation theory . . . 2
1.2 Internal contraction . . . 2
1.3 Classes . . . 3
1.4 First order correction to the wavefunction . . . 3
1.5 Third order correction to the energy . . . 4

2 Setup of codes **5**
2.1 Workflow . . . 5
2.2 `PySCF` and `dice` . . . 5
2.3 `sqa`: Second Quantification Algebra (Python) . . . 7
2.4 `icpt`: Internal Contraction PT (C) . . . 9

A Appendices **10**
A.1 Perturber space . . . 10
A.2 Perturbation tensor . . . 11
A.3 Sketch derivation of the overlaps . . . 13
A.4 More on the overlaps . . . 14
A.5 More details about `sqa` . . . 17
A.6 Projects . . . 18

I'll present the equations we're solving and the setup of the codes we have.

1 Main equations

We want to do a perturbative addition of out-of-active-space dynamical correlation on top of a zeroth order multi-configurational wavefunction. The zeroth order wavefunction and energy (and the corresponding orbitals) are obtained

from a CAS-like calculation. We have the usual separation of N_c core, N_a active and N_v virtual orbitals, and:

$$|\Psi_0\rangle = \sum_n c_n |D_n\rangle \tag{1}$$

1.1 Perturbation theory

We are performing Rayleigh-Schrödinger perturbation theory: the partition $\hat{H} = \hat{H}_0 + \hat{V}$ is so that $|\Psi_0\rangle$ and E_0 are eigenvector and eigenvalue of the zeroth order Hamiltonian. This leaves some flexibility : one can choose the Fock operator (yielding CASPT), the Dyall Hamiltonian (yielding NEVPT) or the Fick Hamiltonian (yielding MRLCC). We are looking for second- and third order corrections to the energy:

$$\begin{aligned} E_2 &= \langle\Psi_0|\hat{V}|\Psi_1\rangle \\ E_3 &= \langle\Psi_1|\hat{V} - E_1|\Psi_1\rangle \end{aligned} \tag{2}$$

where the first order correction to the wavefunction obeys:

$$(E_0 - \hat{H}_0)|\Psi_1\rangle = \hat{V}|\Psi_0\rangle \tag{3}$$

i.e. is found minimizing the Hylleraas functional:

$$\langle\Psi_1|E_0 - \hat{H}_0|\Psi_1\rangle - \langle\Psi_1|\hat{V}|\Psi_0\rangle \tag{4}$$

In the more usual single-reference case, the inversion $\frac{1}{E_0 - \hat{H}_0}$ to obtain $|\Psi_1\rangle$ is trivial, but in the multi-reference case the naive inversion in the many-particle space is problematic. The first order wavefunction has to be computed, and this will be the subject of the following sections.

1.2 Internal contraction

We're going to expand Eq.(3) in a basis. A way to expand $|\Psi_1\rangle$ (in the lhs of Eq.(3)) is to use a contracted scheme, which proposes to consider the basis of functions *connected* to $|\Psi_0\rangle$. The expansion of the first order correction to the wavefunction is then:

$$|\Psi_1\rangle = \sum_I d_I \left(\hat{E}_I|\Psi_0\rangle\right) \tag{5}$$

where $\hat{E}_I|\Psi_0\rangle$ are functions connected to $|\Psi_0\rangle$ through $\hat{E}_I$ and where we need to find the coefficients $\mathbf{d}$. The tedious part is that the basis $\{\hat{E}_I|\Psi_0\rangle\}$ is composed of non-orthogonal many-body states, and the equations will involve overlaps.

1.3 Classes

The action of a two-electron operator on an object made of orbitals separated into core, active and virtual orbitals (for example in Eq.(3), $\hat{H}_0|\Psi_1\rangle$ and $\hat{V}|\Psi_0\rangle$) can be separated into the following 8 classes of excitations in terms of the changes of the occupation pattern.

Names			Δc	Δa	Δv	Operators
I	CCVV	$V^{(0)}_{ij,ab}$	-2	0	$+2$	$\hat{E}_I \to \hat{E}^a_i \hat{E}^b_j$
II	ACVV	$V^{(-1)}_{i,ab}$	-1	-1	$+2$	$\hat{E}_I \to \hat{E}^a_i \hat{E}^b_r$
III	CCAV	$V^{(+1)}_{ij,a}$	-2	$+1$	$+1$	$\hat{E}_I \to \hat{E}^a_i \hat{E}^r_j$
IV	AAVV	$V^{(-2)}_{ab}$	0	-2	$+2$	$\hat{E}_I \to \hat{E}^a_r \hat{E}^b_s$
V	CCAA	$V^{(+2)}_{ij}$	-2	$+2$	0	$\hat{E}_I \to \hat{E}^r_i \hat{E}^s_j$
VI	CAAV	$V^{(0)}_{i,a}$	-1	0	$+1$	$\hat{E}_I \to \hat{E}^a_i \hat{E}^r_s, \hat{E}^a_s \hat{E}^r_i$
VIII	AAAV	$V^{(-1)}_{a}$	0	-1	$+1$	$\hat{E}_I \to \hat{E}^r_s \hat{E}^a_r$
VII	AAAC	$V^{(+1)}_{i}$	-1	$+1$	0	$\hat{E}_I \to \hat{E}^r_t \hat{E}^s_j$

Note: the subspaces CCVV and CAAV are apart, but obviously the subspaces ACVV, CCAV, as well as AAVV, CCAA, and AAAV, AAAC are paired two-by-two, and the equations are very similar for each class in these pairs.

From this, the Fock space of $|\Psi_1\rangle$ is divided into the 8 subspaces, and $\hat{H}_0$ is block-diagonal in this representation: this allows one to solve the relevant equations in each subspace subsequently. Everything object is expressed in sums over those classes:

$$|\Psi_1\rangle = \sum_c |\Psi_1^c\rangle = \sum_c \sum_I d_I^c \left(\hat{E}_I^c |\Psi_0\rangle\right) \tag{6}$$

$$E_2 = \sum_c E_2^c \tag{7}$$

...

1.4 First order correction to the wavefunction

The coefficients $\mathbf{d}^c$ of the expansion of the first order wavefunction in a particular class subspace, $|\Psi_1^c\rangle$, are found with (we use sequentially: (a) the expression of $|\Psi_1^c\rangle$ and $\hat{V}^c|\Psi_0\rangle$ in the basis of the perturbed wavefunction, (b) a multiplication

from the left by $\langle\Psi_0|\hat{E}_K^{c\,\dagger}$ and (c) the realization that $(E_0 - \hat{H}_0)|\Psi_0\rangle = 0$):

$$(E_0 - \hat{H}_0)|\Psi_1^c\rangle = \hat{V}^c|\Psi_0\rangle \tag{8}$$

$$\overset{(a)}{\iff} (E_0 - \hat{H}_0)\sum_I d_I^c \hat{E}_I^c|\Psi_0\rangle = \sum_J w_J^c \hat{E}_J^c|\Psi_0\rangle \tag{9}$$

$$\overset{(b)}{\iff} \sum_I \left\langle \hat{E}_K^{c\,\dagger}(E_0 - \hat{H}_0)\hat{E}_I^c \right\rangle d_I^c = \sum_J \left\langle \hat{E}_K^{c\,\dagger}\hat{E}_J^c \right\rangle w_J^c \tag{10}$$

$$\overset{(c)}{\iff} \sum_I \left\langle \hat{E}_K^{c\,\dagger}\left[(E_0 - \hat{H}_0)\hat{E}_I^c\right] \right\rangle d_I^c = \sum_J \left\langle \hat{E}_K^{c\,\dagger}\hat{E}_J^c \right\rangle w_J^c \tag{11}$$

$$\iff (\mathbf{A}^c\mathbf{d}^c)_K = (\mathbf{S}^c\mathbf{w}^c)_K \tag{12}$$

to be solved for the coefficients $\mathbf{d}^c$. The expression of $\hat{V}^c|\Psi_0\rangle$ in the basis of $|\Psi_1^c\rangle$ can be seen as being obtained from the application of projection of the zeroth order Hamiltonian:

$$\hat{V}^c|\Psi_0\rangle = \hat{P}^c\hat{H}_0|\Psi_0\rangle = \sum_J \left(\hat{E}_J^c|\Psi_0\rangle\langle\Psi_0|\hat{E}_J^{c\,\dagger}\right)\hat{H}_0|\Psi_0\rangle = \sum_J w_J^c\hat{E}_J^c|\Psi_0\rangle \tag{13}$$

and is not always only two-electron integrals on the considered subspace (the expression depends on the zeroth order Hamiltonian, of course!).

The terms to manipulate are $\left\langle \hat{E}_I^{c\dagger}[(E_0 - \hat{H}_0)\hat{E}_J^c] \right\rangle$ and $\left\langle \hat{E}_I^{c\dagger}\hat{E}_J^c \right\rangle$, which involve long strings of creation/annihiltion operators, and one can use the Wick's theorem to simplify the expressions. The resulting expressions will be series of tensors involving one- and two-electron integrals (coming from $\hat{H}_0$), and RDMs up to four order (coming from the excitation operators $\hat{E}_I$).

[BM: an example through and through]

The application of the Wick's theorem to the strings of operators is not done by hand but with the Python library `sqa` and neither is the implementation of the resulting equations, done with an interface to the C software used to calculate the energy: `icpt`.

1.5 Third order correction to the energy

The third order correction to the energy in the context of the 8 classes reads:

$$\begin{aligned}
E_3 = \langle\Psi_1|\hat{V}|\Psi_1\rangle &= \sum_{c_1c_2c_{12}} \langle\Psi_1^{c_1}|\hat{V}^{c_{12}}|\Psi_1^{c_2}\rangle \\
&= \sum_{c_1c_2c_{12}}\sum_{IJK} d_I^{c_1} w_J^{c_{12}} d_K^{c_2} \left\langle \hat{E}_I^{c_1\,\dagger}\hat{E}_J^{c_{12}}\hat{E}_K^{c_2} \right\rangle \\
&= \sum_{c_1}\sum_I d_I^{c_1} \sum_{c_2c_{12}}\sum_{JK} w_J^{c_{12}} d_K^{c_2} \left\langle \hat{E}_I^{c_1\,\dagger}\hat{E}_J^{c_{12}}\hat{E}_K^{c_2} \right\rangle \\
&= \sum_{c_1}\sum_I d_I^{c_1} E_I^{c_1}
\end{aligned} \tag{14}$$

where c_{12} runs over the portions of $\hat{V}$ that explicitly connect $\langle\Psi_1^{c_2}|$ and $|\Psi_1^{c_1}\rangle$.

2 Setup of codes

2.1 Workflow

To obtain the second and third order correction to the energy, we need to have at hand:

1. a minimization engine to obtain the Hylleraas functional of Eq.(3)
2. the tensor contractions involved at each step and stemming from applying the Wick's theorem to $\left\langle \hat{E}_I^{c\dagger}[(E_0 - \hat{H}_0)\hat{E}_J^c] \right\rangle$ and $\left\langle \hat{E}_I^{c\dagger}\hat{E}_J^c \right\rangle$
3. those tensors that appear in those tensor contractions, *i.e.* the bi-electronic integrals and the RDMS

Figure 1 is a summary of this workflow.

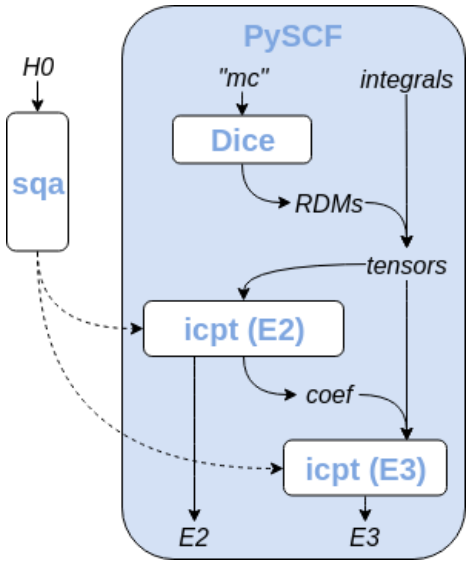

Figure 1: Workflow

2.2 `PySCF` and `dice`

The interface between the different cogs is implemented in `PySCF`. The program is tasked with preparing all ingredients for `icpt` (integral tensors, RDMs tensors, input files) and with gathering the results in one place. Hence `PySCF` will sequentially:

1. Call `Dice` to generate the RDMs, read the generated RDMs and write them back out in a "npy" format. It also checks that the traces are correct. This corresponds to lines 13-20 in the sample below.

2. Produce the integrals in MOs and write out the integral tensors in a "npy" format. This corresponds to lines 21-29 in the sample below.

3. Prepare the input files and run `icpt` for each classes and for the third order calculation. This corresponds to lines 30-43 in the sample below.

The output of an MRPT run in `PySCF` looks like this:

```
1  ------------------------------------------------
2                    MRPT CALCULATION
3  ------------------------------------------------

4  Perturbation type: MRLCC
5  With fully internally contracted scheme
6  With third order correction
7  Recognized a Density Fitting run


8  Number of frozen orbitals:       0
9  Number of core orbitals:        22
10 Number of active orbitals:      16
11 Number of virtual orbitals:     76
12 Number of orbitals:            114

13 Preparing necessary RDMs
14 ......production of RDMs took    2015.70 sec
15 Reading binary 4RDM from DICE
16 ......reading the RDM took       2687.47 sec

17 (GOOD) Trace 4RDM: 491400.0 =   28*   27*   26*   25
18 (GOOD) Trace 3RDM:  19656.0 =   28*   27*   26
19 (GOOD) Trace 2RDM:    756.0 =   28*   27
20 (GOOD) Trace 1RDM:     28.0 =   28

21 Producing the integrals
22 ......production of INT took       0.22 sec

23 Energy_nuc  =  444.71541996
24 Energy_core = -3680.39510562
25 Energy      = -3426.93590197

26 WARNING: Have to use natural orbitals from CAASCF
27          offdiagonal elements:    1.74383453

28 Basic ingredients written to /rc_scratch/bamu3429/31139/int/
29 ......savings of INGREDIENTS took       0.89 sec

30 --ICPT executable:/home/bamu3429/softwares/icpt/icpt.big

31 Second-order:
```

```
perturber CCVV          -0.136103958            464.73
perturber CCAV          -0.017436119            227.57
perturber ACVV          -0.424424406            273.85
perturber CCAA          -0.000847298           1543.85
perturber AAVV          -0.684997895           1051.43
perturber CAAV          -0.095122227           4345.30
perturber AAAV          -0.090682806           2103.23
perturber AAAC          -0.000395028           2217.49
Total:                  -1.450009738

Third-order:             0.137398286            688.40

Total PT                -1.312611452
Total energy         -3428.248513418
```

2.3 `sqa`: Second Quantification Algebra (Python)

The application of Wick's theorem to the expressions involved in the equations presented above would be a cumbersome and error-prone task if done manually. Instead, we use the `sqa`, that stands for "Second Quantification Algebra" program. This Python library automates the manipulation of operators and in particular automates the application of Wick's theorem.

The package lets you define categories of indexes (core, active, virtual for example) and terms that are products of tensors and excitation operators, and is able to enact commutation rules to the creation and destruction operators involved to yield a Wick-ordered result.

I wrote an API to use the library, with for example one-liners that define the zeroth order Hamiltonian, user-friendly commands to simplify the results obtained, and systematic outputs in formats readable by `icpt`. This also includes the possibility to trigger the use of the cumulant approximation for the fourth order RDM, as well as runs adapted to the use of density-fitting for the two-electron integrals. Hence this API basically takes as input:

- the zeroth order Hamiltonian (which will determine that we are doing NEVPT or MRLCC for example)
- the class of excitations, *i.e.* the excitation operators of the basis

The relevant lines of code will be found in home-made scripts in [BM: PATH]. The output of these scripts are the "`.inl`" files containing information on the equations that are to be handled by `icpt`, as shown below, where:

- the tensors that are going to be used are declared, lines 2-28, with a name, a domain in term of core, active, and virtual orbitals and a "usage" (namely: amplitudes that are used in the conjugate gradient algorithm, Hamiltonian tensors, and density-like tensors such as RDMs and overlaps)

- a list of tensor contractions stemming from application of the Wick's theorem, lines 29-41, with list of indexes, a factor, and a list of tensors involved (the tensors are designated by their index in the list of tensors above).
- similar information to construct the overlap, lines 42-47
- general information about the run to be done with `icpt`, lines 48-60

```
1  namespace MRLCC_AAVV {

2    FTensorDecl TensorDecls[25] = {
3      /*  0*/{"R"      , "eeaa"     , "", USAGE_Residual    },
4      /*  1*/{"t"      , "eeaa"     , "", USAGE_Amplitude   },
5      /*  2*/{"T"      , "eeaa"     , "", USAGE_Amplitude   },
6      /*  3*/{"b"      , "eeaa"     , "", USAGE_Amplitude   },
7      /*  4*/{"B"      , "eeaa"     , "", USAGE_Amplitude   },
8      /*  5*/{"p"      , "eeaa"     , "", USAGE_Amplitude   },
9      /*  6*/{"Ap"     , "eeaa"     , "", USAGE_Amplitude   },
10     /*  7*/{"P"      , "eeaa"     , "", USAGE_Amplitude   },
11     /*  8*/{"AP"     , "eeaa"     , "", USAGE_Amplitude   },
12     /*  9*/{"W"      , "eeaa"     , "", USAGE_Hamiltonian },
13     /* 10*/{"k"      , "aa"       , "", USAGE_Hamiltonian },
14     /* 11*/{"W"      , "aaaa"     , "", USAGE_Hamiltonian },
15     /* 12*/{"W"      , "aeae"     , "", USAGE_Hamiltonian },
16     /* 13*/{"W"      , "aeea"     , "", USAGE_Hamiltonian },
17     /* 14*/{"W"      , "caac"     , "", USAGE_Hamiltonian },
18     /* 15*/{"W"      , "caca"     , "", USAGE_Hamiltonian },
19     /* 16*/{"W"      , "cece"     , "", USAGE_Hamiltonian },
20     /* 17*/{"W"      , "ceec"     , "", USAGE_Hamiltonian },
21     /* 18*/{"k"      , "ee"       , "", USAGE_Hamiltonian },
22     /* 19*/{"W"      , "e"        , "", USAGE_Intermediate},
23     /* 20*/{"delta"  , "cc"       , "", USAGE_Density     },
24     /* 21*/{"E2"     , "aaaa"     , "", USAGE_Density     },
25     /* 22*/{"S1"     , "aaaa"     , "", USAGE_Density     },
26     /* 23*/{"S2"     , "aaaa"     , "", USAGE_Density     },
27     /* 24*/{"E3"     , "aaaaaa"   , "", USAGE_Density     },
28   };

29   FEqInfo EqsRes[11] = {
30     {"abcd,ef,abeg,cdfg",       -4.0,   4, { 6,10, 5,21}},
31     {"abcd,efgh,abef,cdgh",     -2.0,   4, { 6,11, 5,21}},
32     {"abcd,efgi,abfh,cdgh,ie",   4.0,   5, { 6,14, 5,21,20}},
33     {"abcd,efig,abfh,cdgh,ie",  -8.0,   5, { 6,15, 5,21,20}},
34     {"abcd,eaif,bfgh,cdhg,ie",   8.0,   5, { 6,16, 5,21,20}},
35     {"abcd,eafi,bfgh,cdhg,ie",  -4.0,   5, { 6,17, 5,21,20}},
36     {"abcd,ae,befg,cdgf",        4.0,   4, { 6,18, 5,21}},
37     {"abcd,abef,efgh,cdgh",      2.0,   4, { 6,19, 5,21}},
38     {"abcd,efgh,abei,cdfgih",   -4.0,   4, { 6,11, 5,24}},
39     {"abcd,eafg,bghi,cdeihf",    4.0,   4, { 6,12, 5,24}},
```

```
    {"abcd,eafg,bfhi,cdeghi",    4.0,   4, { 6,13, 5,24}},
  };

  FEqInfo bVec[4] = {
    {"ABRS,ABPQ,RSPQ",  0.25,   3, { 3, 9,21}},
    {"BARS,ABPQ,RSQP",  0.25,   3, { 3, 9,21}},
    {"ABRS,BAPQ,RSQP",  0.25,   3, { 3, 9,21}},
    {"BARS,BAPQ,RSPQ",  0.25,   3, { 3, 9,21}},
  };

  static void GetMethodInfo(FMethodInfo &Out) {
    Out = FMethodInfo();
    Out.pName = "MRLCC_AAVV";
    Out.perturberClass = "AAVV";
    Out.Whandcoded_if_zero = 1;
    Out.E3handcoded_if_zero = 1;
    Out.pSpinClass = "restricted";
    Out.pTensorDecls = &TensorDecls[0];
    Out.nTensorDecls = 25;
    Out.EqsRes = FEqSet(&EqsRes[0], 11, "MRLCC_AAVV/Res");
    Out.bVec = FEqSet(&bVec[0], 4, "MRLCC_AAVV/bVec");
  };
};
```

2.4 `icpt`: Internal Contraction PT (C)

The main command files are `main.cpp`, `icpt.cpp` and `PerturberDependentCode.cpp`. The file `main.cpp`, commands the input file to be read and the proper job to be run (either all NEVPT or all MRLCC jobs, or one particular job, depending on the input file). The file `icpt.cpp` contains the `ReadInputFile` and `Run` routines. The `Run` routine initializes everything that is needed in the `Init` routine (i.e. the `Domains` and the `Tensors` as seen in the `.inl` file), constructs the amplitude `b` in the `InitAmplitudes` routine, and computes the orthogonal basis in the `MakeOverlapAndOrthogonalBasis` routine. Then the conjugate gradient loop is launched, which is basically the following sequence:

$$P \xrightarrow{\texttt{BackToNonOrthogonal}} p \xrightarrow{\texttt{ExecEquationSet}} Ap \xrightarrow{\texttt{MakeOrthogonal}} AP \xrightarrow{(\text{loop})} P^{\text{new}} \quad (15)$$

i.e. **A** is not stored, only it's action on some tensor: (**Ap**) is needed (this is Eq.(15): $c_y = A_{xy} p_y$).

[BM: Mention Tensor, ReadNpy, where come from, TN versus TND, etc. . .]

A Appendices

A.1 Perturber space

To help in the following, and to more closely follow derivations found in the literature, consider the zeroth order wavefunction to be an antismmetrized product of a core, an active, and a virtual part (where all the determinant of the CAS-CI space share the same inactive part, by definition, and $\sum c_n = 1$):

$$|\Psi_0\rangle = |D^c \Psi^a D^v\rangle = |D^c\rangle \otimes \sum_n c_n |D_n^a\rangle \otimes |D^v\rangle = \sum_n c_n |D^c D_n^a D^v\rangle \tag{16}$$

Any (zeroth order) wavefunction of the Fock space outside the CAS connected to a determinant of the zeroth order wavefunction is a "perturber wavefunctions". The perturber wavefunctions are of the form:

$$|\Psi_\mu^{c'a'v'}\rangle = |D^{c'}\, \Psi_\mu^{a'}\, D^{v'}\rangle = \sum_I c_{\mu,I}^{c'a'v'} |D^{c'}\, D_I^{a'}\, D^{v'}\rangle \tag{17}$$

where the occupation pattern is changed, with respect to the CAS-CI space, by c', v' and a' electrons in the core, virtual and active spaces (all three changes can be zero, negative or positive, see below) and where μ labels the different perturbers. Note that in the literature a' is called k and l is used to collectively designates $(c'v')$.

Now that this is established, we can simplify the notations back to a more condensed form. The determinants seen in the previous equation can be written as excitations to determinants of the zeroth order wavefunction:

$$|D^{c'}\, D_I^{a'}\, D^{v'}\rangle = |\hat{E}^{c'} D^c \;\; \hat{E}_{I,n}^{a'} D_n^a \;\; \hat{E}^{v'} D^v\rangle = \hat{E}_{I,n}^{c'a'v'} |D^c D_n^a D^v\rangle = \hat{E}_{I,n}^{c} |D_n\rangle \tag{18}$$

where (c) collectively designate the "class of excitation" $(c'a'v')$ (see below). The perturber wavefuntions belong to the total Fock space $S = \bigoplus S^c$, where S^c are the Fock spaces spanned by the basis:

$$b = \{\hat{E}_{I,n}^c |D_n\rangle\} \tag{19}$$

and all the objects interesting in this context are decomposed as follow:

$$\hat{E}_{I,n} = \sum \hat{E}_{I,n}^c, \qquad |\Psi_1\rangle = \sum |\Psi_1^c\rangle, \qquad \hat{V} = \sum \hat{V}^c, \qquad E_2 = \sum E_2^c, \text{ etc...} \tag{20}$$

Uncontracted scheme The uncontracted scheme exploits the full dimentionnality of the Fock space of the perturber wavefunctions, that is to say it proposes to expand the perturber wavefunction $|\Psi_\mu^c\rangle$ in the basis:

$$b_{\mathrm{UC}} = b = \{\hat{E}_{I,n}^c |D_n\rangle\} \qquad \text{and} \qquad |\Psi_\mu^c\rangle = \sum_I c_{\mu,I}^c \hat{E}_{I,n}^c |D_n\rangle \tag{21}$$

The perturber wavefunctions could be defined as the eigenfunctions of the Hamiltonian in the spaces S^c, that is to say that the $c^c_{\mu,I}$ coefficients could be obtained by diagonalization of the Hamiltonian in S^c, which is a very expensive calculation. The first order correction to the wavefunction would then be expanded on the perturber wavefunctions:

$$|\Psi_1\rangle = \sum_c |\Psi_1^c\rangle \qquad \text{and} \qquad |\Psi_1^c\rangle = \sum_{c\mu} c^c_\mu |\Psi^c_\mu\rangle \tag{22}$$

but we are actually only interested in the coefficients $d^c_{I,n}$ of the expansion of $|\Psi_1\rangle$ directly on the basis b_{UC}:

$$|\Psi_1\rangle = \sum_c |\Psi_1^c\rangle \qquad \text{and} \qquad |\Psi_1^c\rangle = \sum_{cI} d^c_{I,n} \hat{E}^c_{I,n} |D_n\rangle \tag{23}$$

that can be found using equations like those found later in these notes (see HERE). This is a less expensive calculation, although a large number of linear equation of excited CAS-CI size have to be perfomed...

Internal contraction An alternative way would be to use a contracted scheme, which rather proposes to consider subspaces $\bar{S}^c \supset S^c$ of perturber wavefunctions connected to $|\Psi_0\rangle$ (by opposition to perturber wavefunctions connected to a determinant of $|\Psi_0\rangle$). The basis of such subspaces are:

$$b_{\text{IC}} = \{\hat{E}^c_I |\Psi_0\rangle\} \tag{24}$$

and the expansion of the first order correction to the wavefunction becomes:

$$|\Psi_1\rangle = \sum_c |\Psi_1^c\rangle \qquad \text{and} \qquad |\Psi_1^c\rangle = \sum_I d^c_I \hat{E}^c_I |\Psi_0\rangle = \sum_I d^c_I \hat{E}^c_I \sum_n c_n |D_n\rangle \tag{25}$$

The contraction is a factorization, much like CC is a factorization of the parameters of FCI, and you can get (approximations to) the original coefficients $d^c_{I,n}$ by products of c_n and d^c_I coefficients.

There is different level of contraction depending of the definition chosen for the excitation operators $\hat{E}^c_I$. A strongly contracted scheme would consider only one perturber wavefunction per class of excitation, found by application of the perturbation operator (i.e. by a part of the Hamiltonian):

$$\bar{S}^c = \text{span}(\hat{V}^c |\Psi_0\rangle) \qquad \text{and} \qquad \hat{E}^c_I = \hat{V}^c = \hat{P}^c \hat{H} \tag{26}$$

The coefficients d^c_I are determined by formulas. The tedious part is that the basis $\{\hat{E}_I |\Psi_0\rangle\}$ is composed of non-orthogonal many-body states, and the equations will involve overlaps (see Subsection A.3 Sketch derivation of the overlaps and A.4 More on the overlaps).

A.2 Perturbation tensor

For some classes of excitations (namely CAAV, AAAV and AAAC), we do not have merely $w^c_J = W^c_J$ (W being the tensor of corresponding integrals). In the program (see Section 2.4: `icpt`: Internal Contraction PT (C)), this is handled by adding the desired terms to the W that is to be contracted with the overlap.

CAAV

$$
\begin{aligned}
(ri,\mu) &= \sum_{\substack{ab\\a'b'}} 2W1^{ri}_{ab} \left\langle \hat{E}^{b'}_{a'} \hat{E}^{a}_{b} \right\rangle c^{\mu}_{a'b'} - \sum_{\substack{ab\\a'b'}} W2^{ri}_{ab} \left\langle \hat{E}^{b'}_{a'} \hat{E}^{a}_{b} \right\rangle c^{\mu}_{a'b'} \\
&+ \sum_{ab} 2h^{ri} \left\langle \hat{E}^{b'}_{a'} \right\rangle c^{\mu}_{a'b'} \\
&= \sum_{\substack{ab\\a'b'}} 2W1^{ri}_{ab} \left\langle \hat{E}^{b'}_{a'} \hat{E}^{a}_{b} \right\rangle c^{\mu}_{a'b'} - \sum_{\substack{ab\\a'b'}} W2^{ri}_{ab} \left\langle \hat{E}^{b'}_{a'} \hat{E}^{a}_{b} \right\rangle c^{\mu}_{a'b'} \\
&+ \sum_{\substack{ab\\a'b'}} 2h^{ri} \frac{\delta_{a'b'}}{N_{\mathrm{a}}} \left\langle \hat{E}^{b'}_{a'} \hat{E}^{a}_{b} \right\rangle c^{\mu}_{a'b'} \\
(ri,\mu)' &= - \sum_{\substack{ab\\a'b'}} W1^{ri}_{ab} \left\langle \hat{E}^{b'}_{a'} \hat{E}^{a}_{b} \right\rangle c^{\mu}_{a'b'} + \sum_{\substack{ab\\a'b'}} W2^{ri}_{ab} S''_{a'b',ab} c^{\mu}_{a'b'} \\
&- \sum_{ab} h^{ri} \left\langle \hat{E}^{b'}_{a'} \right\rangle c^{\mu}_{a'b'} \\
&= - \sum_{\substack{ab\\a'b'}} W1^{ri}_{ab} \left\langle \hat{E}^{b'}_{a'} \hat{E}^{a}_{b} \right\rangle c^{\mu}_{a'b'} + \sum_{\substack{ab\\a'b'}} W2^{ri}_{ab} S''_{a'b',ab} c^{\mu}_{a'b'} \\
&- \sum_{\substack{ab\\a'b'}} h^{ri} \frac{\delta_{a'b'}}{N_{\mathrm{a}}} \left\langle \hat{E}^{b'}_{a'} \hat{E}^{a}_{b} \right\rangle c^{\mu}_{a'b'}
\end{aligned} \tag{27}
$$

$$
W1^{ri}_{ab} \leftarrow W1^{ri}_{ab} + h^{ri} \frac{\delta_{a'b'}}{N_{\mathrm{a}}} \tag{28}
$$

AAAV

$$
\begin{aligned}
(r,\mu) &= \sum_{\substack{abc\\a'b'c'}} W^{r}_{abc} \left\langle \hat{E}^{c'}_{a'} \hat{E}^{b'}_{b} \hat{E}^{a}_{c} \right\rangle c^{\mu}_{a'b'c'} + \sum_{\substack{abc\\a'b'c'}} {h'}^{r}_{b} \left\langle \hat{E}^{c'}_{a'} \hat{E}^{b'}_{b} \right\rangle c^{\mu}_{a'b'c'} \\
&= \sum_{\substack{abc\\a'b'c'}} W^{r}_{abc} \left\langle \hat{E}^{c'}_{a'} \hat{E}^{b'}_{b} \hat{E}^{a}_{c} \right\rangle c^{\mu}_{a'b'c'} + \sum_{\substack{abc\\a'b'c'}} {h'}^{r}_{b} \frac{\delta_{ca}}{N_{\mathrm{a}}} \left\langle \hat{E}^{c'}_{a'} \hat{E}^{b'}_{b} \hat{E}^{a}_{c} \right\rangle c^{\mu}_{a'b'c'}
\end{aligned} \tag{29}
$$

$$
W^{r}_{abc} \leftarrow W^{r}_{abc} + {h'}^{r}_{b} \frac{\delta_{ca}}{N_{\mathrm{a}}} \tag{30}
$$

AAAC

$$
\begin{aligned}
(i,\mu) &= \sum_{\substack{abc\\a'b'c'}} W^{i}_{cba} \left\langle \hat{E}^{c'}_{a'} \hat{\bar{E}}^{b'}_{b} \hat{E}^{a}_{c} \right\rangle c^{\mu}_{a'b'c'} + \sum_{\substack{abc\\a'b'c'}} {h'}^{i}_{b} \left\langle \hat{E}^{c'}_{a'} \hat{\bar{E}}^{b'}_{b} \right\rangle c^{\mu}_{a'b'c'} \\
&= \sum_{\substack{abc\\a'b'c'}} W^{i}_{cba} \left\langle \hat{E}^{c'}_{a'} \hat{\bar{E}}^{b'}_{b} \hat{E}^{a}_{c} \right\rangle c^{\mu}_{a'b'c'} + \sum_{\substack{abc\\a'b'c'}} {h'}^{i}_{b} \frac{\delta_{ca}}{N_{\mathrm{a}}} \left\langle \hat{E}^{c'}_{a'} \hat{\bar{E}}^{b'}_{b} \hat{E}^{a}_{c} \right\rangle c^{\mu}_{a'b'c'}
\end{aligned} \tag{31}
$$

$$W^i_{abc} \leftarrow W^i_{abc} + h'^i_{\ b} \frac{\delta_{ca}}{N_{\rm a}} \tag{32}$$

A.3 Sketch derivation of the overlaps

A sketch derivation of the overlaps involved in the different classes might help, notably in realizing that the overlap only contains active space indexes.

ACVV

$$\left\langle \hat{E}^{p'}_{a'} \hat{E}^{i'}_{b'} \hat{E}^{b}_{i} \hat{E}^{a}_{p} \right\rangle \tag{33}$$

where, going from the center, one cannot destroy a b' that has not been created, so this gives rise to a $\delta_{b'b}$ or $\delta_{b'a}$ (and later a $\delta_{a'b}$ or $\delta_{a'a}$). Then one cannot create a i' that has not been destroyed, so this gives rise to a $\delta_{i'i}$. One is left with a $\left\langle \hat{E}^{p'}_{p} \right\rangle = E_1(aa)$.

CCAV

$$\left\langle \hat{E}^{i'}_{p'} \hat{E}^{j'}_{a'} \hat{E}^{a}_{j} \hat{E}^{p}_{i} \right\rangle \tag{34}$$

where, going from the center, one cannot destroy a a' that has not been created, so this gives rise to a $\delta_{a'a}$. Then one cannot create a j' that has not been destroyed, so this gives rise to a $\delta_{j'j}$ or $\delta_{j'i}$ (and later to a $\delta_{i'j}$ or $\delta_{i'i}$). One is left with a $\left\langle \hat{E}^{p}_{p'} \right\rangle = \tilde{E}_1(aa)$.

AAVV

$$\left\langle \hat{E}^{p'}_{a'} \hat{E}^{q'}_{b'} \hat{E}^{b}_{q} \hat{E}^{a}_{p} \right\rangle \tag{35}$$

where, going from the center, one cannot destroy a b' that has not been created, so this gives rise to a $\delta_{b'b}$ or $\delta_{b'a}$ (and later to a $\delta_{a'b}$ or $\delta_{a'a}$). One is left with a $\left\langle \hat{E}^{p'}_{p} \hat{E}^{q'}_{q} \right\rangle = E_2(aaaa)$.

CCAA

$$\left\langle \hat{E}^{i'}_{p'} \hat{E}^{j'}_{q'} \hat{E}^{q}_{j} \hat{E}^{p}_{i} \right\rangle \tag{36}$$

where, going from the center, one cannot create a j' that has not been destroyed, so this gives rise to a $\delta_{j'j}$ or $\delta_{j'i}$ (and later to a $\delta_{i'j}$ or $\delta_{i'i}$). One is left with a $\left\langle \hat{E}^{p}_{p'} \hat{E}^{q}_{q'} \right\rangle = \tilde{E}_2(aaaa)$.

CAAV

$$\left\langle \hat{E}^{i'}_{p'} \hat{E}^{q'}_{a'} \hat{E}^{a}_{q} \hat{E}^{p}_{i} \right\rangle, \quad \left\langle \hat{E}^{q'}_{p'} \hat{E}^{i'}_{a'} \hat{E}^{a}_{i} \hat{E}^{p}_{q} \right\rangle, \quad \left\langle \hat{E}^{q'}_{p'} \hat{E}^{i'}_{a'} \hat{E}^{a}_{q} \hat{E}^{p}_{i} \right\rangle \tag{37}$$

which will give rise to $\delta_{i'i}$ and $\delta_{a'a}$, and an overlap that only depend on $pq, p'q'$ in different order that give rise to E_2 and E_1.)

CCVV

$$\left\langle \hat{E}^{i'}_{a'} \hat{E}^{j'}_{b'} \hat{E}^{b}_{j} \hat{E}^{a}_{i} \right\rangle \tag{38}$$

where, going from the center, one cannot destroy a b' that has not been created, so this gives rise to a $\delta_{b'b}$ or $\delta_{b'a}$. Similarly, one cannot create a j' that has not been destroyed, so this gives rise to a $\delta_{j'j}$ or $\delta_{j'i}$. Then a $\delta_{a'b}$ or $\delta_{a'a}$ and a $\delta_{i'j}$ or $\delta_{i'i}$.

A.4 More on the overlaps

In the code, the overlaps are computed and then diagonalized to solve Eq. 12 in an orthogonal basis. This is how it's thought. The overlap is diagonalized as follow:

$$\mathbf{SU} = \mathbf{UD} \tag{39}$$
$$\mathbf{U}^\dagger\mathbf{SU} = \mathbf{USU}^\dagger = \mathbf{D} \tag{40}$$
$$\mathbf{U}^\dagger\mathbf{U} = \mathbf{UU}^\dagger = \mathbf{1} \tag{41}$$

and the orthogonal basis transformation is defined:

$$\mathbf{V} = \mathbf{UD}^{-1/2} \tag{42}$$
$$\mathbf{VDV}^\dagger = \mathbf{DV}^\dagger\mathbf{V} = \mathbf{V}^\dagger\mathbf{VD} = \mathbf{1} \tag{43}$$
$$\mathbf{V}^\dagger\mathbf{SV} = \mathbf{1} \tag{44}$$

so that the equation to solve becomes:

$$\mathbf{Ad} = \mathbf{Sw} \tag{45}$$
$$\mathbf{V}^\dagger\mathbf{A}(\mathbf{VDV}^\dagger)\mathbf{d} = \mathbf{V}^\dagger\mathbf{S}(\mathbf{VDV}^\dagger)\mathbf{w} \tag{46}$$
$$\tilde{\mathbf{A}}\tilde{\mathbf{d}} = \tilde{\mathbf{w}} \tag{47}$$

where:

$$\tilde{\mathbf{A}} = \mathbf{V}^\dagger\mathbf{AV} \tag{48}$$
$$\tilde{\mathbf{d}} = \mathbf{DV}^\dagger\mathbf{d} = \mathbf{D}^{1/2}\mathbf{U}^\dagger\mathbf{d} \tag{49}$$
$$\tilde{\mathbf{w}} = \mathbf{DV}^\dagger\mathbf{w} = \mathbf{D}^{1/2}\mathbf{U}^\dagger\mathbf{w} \tag{50}$$

ACVV We are considering $(i,p) \to (a,b)$, but it is important to distinguish between t^{ab}_{ip} and t^{ab}_{pi}. By restricting to $a < b$, we obtain:

$$\mathbf{S}_1 = \begin{pmatrix} 2E_1 & -E_1 \\ -E_1 & 2E_1 \end{pmatrix} \tag{51}$$

and when $a = b$, $\mathbf{S}_2 = E_1$. Historically, in the code, this explained the presence of two tensors: `S1:AA` and `S2:aa`, where `A` stands for a domain twice as big as `a`, allowing a block structure. We had:

$$\begin{pmatrix} T^{ab}_{ip} \\ T^{ab}_{pi} \end{pmatrix} = \begin{pmatrix} (V_a)_{pp'} & (V_b)_{pp'} \\ (V_c)_{pp'} & (V_d)_{pp'} \end{pmatrix} \begin{pmatrix} t^{ab}_{ip'} \\ t^{ab}_{p'i} \end{pmatrix} = \begin{pmatrix} V_a t^{ab}_{ip'} + V_b t^{ab}_{p'i} \\ V_c t^{ab}_{ip'} + V_d t^{ab}_{p'i} \end{pmatrix} \tag{52}$$

and in the case where $a = b$, $T = \mathbf{V}_2 t$. The matrices V_a, V_b, V_c and V_d were blocks of the orthogonal basis $\mathbf{V}_1$ formed from the block matrix $\mathbf{S}_1$. Now we use the fact that given the block structure of $\mathbf{S}_1$, its eigenvectors and eigenvalues are:

$$\mathbf{U} = \tag{53}$$

where...

CCAV This case is exactly similar to the previous one, with $(i, j) \to (p, a)$ and the restriction $i < j$. The 1RDM E_1 is replaced with the one-hole RDM $\tilde{E}_1 = 2\mathrm{I} - E_1$, so that we had in the code:

$$\mathbf{S}_1 = \begin{pmatrix} 2\tilde{E}_1 & -\tilde{E}_1 \\ -\tilde{E}_1 & 2\tilde{E}_1 \end{pmatrix} = \begin{pmatrix} -2E_1 & E_1 \\ E_1 & -2E_1 \end{pmatrix} + \begin{pmatrix} 4\mathrm{I} & -2\mathrm{I} \\ -2\mathrm{I} & 4\mathrm{I} \end{pmatrix} \tag{54}$$

and $\mathbf{S}_2 = \tilde{E}_1 = 2\mathrm{I} - E_1$. We know use the same reasonning as for ACVV to handle only `aa`-size matrices.

AAVV We are considering $(p, q) \to (a, b)$, with $a < b$. Restricting ourselves to $p < q$, we have:

$$\mathbf{S}_1 = \begin{pmatrix} E_2 & E_2^\dagger \\ E_2^\dagger & E_2 \end{pmatrix} \tag{55}$$

and for $a = b$, $\mathbf{S}_2 = E_2 + E_2^\dagger$. Historically, this explained the dimension of $\mathbf{S}_1$ being `S1:aAaA` We had:

$$\begin{pmatrix} T^{ab}_{pq} \\ T^{ab}_{qp} \end{pmatrix} = \begin{pmatrix} (V_a)_{pq,p'q'} & (V_b)_{pq,p'q'} \\ (V_c)_{pq,p'q'} & (V_d)_{pq,p'q'} \end{pmatrix} \begin{pmatrix} t^{ab}_{p'q'} \\ t^{ab}_{q'p'} \end{pmatrix} \tag{56}$$

where the matrices V_a, V_b, V_c and V_d were blocks of the orthogonal basis $\mathbf{V}_1$ formed from the block matrix $\mathbf{S}_1$. Now we use the fact that given the block structure of $\mathbf{S}_1$, its eigenvectors and eigenvalues are:

$$\mathbf{U} = \begin{pmatrix} -\mathbf{u} & \mathbf{u} \\ \mathbf{u} & \mathbf{u} \end{pmatrix} \mathbf{D} = \begin{pmatrix} \mathbf{d} - \mathbf{dt} & \mathbf{0} \\ \mathbf{0} & \mathbf{d} - \mathbf{dt} \end{pmatrix} \tag{57}$$

CCAA This case is exactly similar to the previous one, with $(i, j) \to (p, q)$. The 2RDM E_2 is replaced with the twohole RDM $\tilde{E}_2 = E_2 + \delta E_1 - 2\delta E_1 - 2\delta\delta + 4\delta\delta$. That is why the overlaps are a little bit more fastidious to accumulate and historically we resorted to `MakeS1` and `MakeS2`. In the code we do not use a doubled dimension `A=2*a`.

CAAV Similarly, we have to distinguish between t_{ip}^{qa} and t_{pi}^{qa}, which are spanned by two different basis. We have:

$$\mathbf{S} = \begin{pmatrix} A & B \\ B & C \end{pmatrix} \tag{58}$$

where

$$A_{pq,p'q'} = 2E_{2pq,p'q'} + 2\delta_{pp'}E_{1qq'} \tag{59}$$
$$B_{pq,p'q'} = -E_{2pq,p'q'} - \delta_{pp'}E_{1qq'} \tag{60}$$
$$C_{pq,p'q'} = -E_{2pq,q'p'} + 2\delta_{pp'}E_{1qq'} \tag{61}$$

Note: in the program, the $(4N_{\rm a}^4)$ are distributed as $2N_{\rm a}^2.2N_{\rm a}^2$, i.e. as $S_1(pQ, rS)$, i.e. as a two-dimensionnal array.
Note: in the program, for AAVV, we had the $(4N_{\rm a}^4)$ distributed as $N_{\rm a}.2N_{\rm a}.N_{\rm a}.2N_{\rm a}$, i.e. as $S_1(p, Q, r, S)$, i.e. for each p, r, we have:

$$\begin{pmatrix} qs & qs \\ qs & qs \end{pmatrix} \tag{62}$$

CCVV [BM: I though the overlap of this class is made of δ functions, there is $1/\sqrt{2}$ and $1/\sqrt{3}$ that arise : I should check that (norm?).]

The $(ij) \to (ab)$ are separated in four components (with $i < j$ and $a < b$): $t_1 \equiv t_{ij}^{ab}$, $t_2 \equiv t_{ji}^{ab}$, $t_3 \equiv t_{ij}^{ba}$, $t_4 \equiv t_{ji}^{ba}$. We have:

$$\begin{pmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{pmatrix} = \frac{1}{4} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1\sqrt{3} & -1\sqrt{3} & -1\sqrt{3} & 1\sqrt{3} \\ & -- & -- & \\ & -- & -- & \end{pmatrix} \begin{pmatrix} t_1 \\ t_2 \\ t_3 \\ t_4 \end{pmatrix} \tag{63}$$

and in the case where $a = b$, we end up with:

$$\begin{pmatrix} T_1 \\ T_2 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1\sqrt{2} & 1\sqrt{2} \\ -- & -- \end{pmatrix} \begin{pmatrix} t_1 \\ t_2 \end{pmatrix} \tag{64}$$

The reverse transformation reads:

$$\frac{1}{4} \begin{pmatrix} t_1 \\ t_2 \\ t_3 \\ t_4 \end{pmatrix} = \begin{pmatrix} 1 & \sqrt{3} & & \\ 1 & -\sqrt{3} & -- & -- \\ 1 & -\sqrt{3} & -- & -- \\ 1 & \sqrt{3} & & \end{pmatrix} \begin{pmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{pmatrix} \tag{65}$$

and: $$\begin{pmatrix} t_1 \\ t_2 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1\sqrt{2} & -- \\ 1\sqrt{2} & -- \end{pmatrix} \begin{pmatrix} T_1 \\ T_2 \end{pmatrix} \tag{66}$$

About $E.W$ With $E2^{\dagger}_{pq,rs} = E2_{pq,sr}$ and $W^{ab}_{rs} =< ab|rs >= W1^{ab}_{rs}$ and $W^{ba}_{rs} =< ba|rs >= W2^{ab}_{sr}$, we have:

$$\begin{aligned}(E2.W1)^{ab}_{pq} &= \sum_{rs} E2_{pq,rs} W1^{ab}_{rs} \\ &= \sum_{sr} E2_{pq,sr} W1^{ab}_{sr} \\ &= \sum_{rs} E2^{\dagger}_{pq,rs} W2^{ab}_{rs} \\ &= (E2^{\dagger}.W2)^{ab}_{pq} \end{aligned} \tag{67}$$

$$\begin{aligned}(E2^{\dagger}.W1)^{ab}_{pq} &= \sum_{rs} E2^{\dagger}_{pq,rs} W1^{ab}_{rs} \\ &= \sum_{rs} E2_{pq,sr} W1^{ab}_{rs} \\ &= \sum_{sr} E2_{pq,rs} W2^{ab}_{rs} \\ &= (E2.W2)^{ab}_{pq} \end{aligned} \tag{68}$$

A.5 More details about `sqa`

The useful object the library has are:

`sqa.index`	associates a kind to an index (core, active, virtual). These indexes can be make to be dummy indexes or not.
`sqa.sfExOp`	defines a "spin free excitation operator" $\hat{E}^{n\cdots}_{m\cdots}$
`sqa.symmetry`	defines a "symmetry class" applicable to any tensor
`sqa.tensor`	defines a "tensor", with a name, indexes and `sqa.symmetry` classes: $t^{m\cdots}_{n\cdots}$
`sqa.term`	defines a "term" of an equation with `sqa.tensor` and `sqa.sfExOp`
`sqa.commutator`	defines a "commutator" of two `sqa.term`s

The actions on these objects that are interesting for us are:

`sqa.multiplyTerms`	creates an `sqa.term` that is the multiplication of two `sqa.term`s
`sqa.normalOrder`	normal orders a string of `sqa.term`s

We want to handle expressions like (in the example of CCVV, with $\hat{E}^{\dagger}_I = \hat{E}^l_d \hat{E}^k_c$ and $\hat{E}_J = \hat{E}^b_j \hat{E}^a_i$):

$$A_{IJ} = \left\langle \hat{E}^l_d \hat{E}^k_c [\hat{H}_0 \hat{E}^b_j \hat{E}^a_i] \right\rangle \qquad \text{and} \qquad S_{IJ} = \left\langle \hat{E}^l_d \hat{E}^k_c \hat{E}^b_j \hat{E}^a_i \right\rangle \tag{69}$$

Defining the zeroth order Hamiltonian is a simple matter of calling for example:

```
H=h0('MRLCC',df=True)
```

Then, one will want to associate indexes as such:

```
list = gimme('vvcc')
```

Then one actually wants to obtain equations for $(Ap)_I = A_{IJ}.p_J$, so that one will writethe following piece of code:

```
Cin  = sqa.tensor("p",  [a,b,i,j], [Dsym_c])
Cout = sqa.tensor("Ap", [c,d,l,k], [Dsym_c])
Op1 = sqa.term( 1.0, [""],  [Cin,  sqa.sfExOp([a, i]) , sqa.sfExOp([b, j])])
Op2 = sqa.term( 1.0, [""],  [Cout, sqa.sfExOp([k, d]) , sqa.sfExOp([l, c])])
commutator = []
commutator +=  sqa.commutator(HD_A, Op1)
commutator +=  sqa.commutator(T_C,  Op1)
commutator +=  sqa.commutator(T_A,  Op1)
commutator +=  sqa.commutator(T_V,  Op1)
```

where `Cin="p"` is a container for the current trial vector, and `Cout="Ap"` is a container for the updated trial vector.

```
result = []
for t in commutator:
    result += sqa.normalOrder(sqa.multiplyTerms(Op2, t))
result = simplify_all(result,[deltaC, deltaA, deltaV])
```

A.6 Projects

Initial `b` through code and not `inl` files?
Ortho and Back through matrix multiplications routines?
Optimization of tensor contraction
Tensor Hyper-Contraction (for the integrals AND the "amplitudes" $A.p$).
F12 stuff in there