An Efficient Implementation of Second Quantization-Based Many-Body Methods for Electrons and its Application to Coupled-Cluster with Arbitrary Excitation Level

Inaugural-Dissertation

zur

Erlangung des Doktorgrades

der Mathematisch-Naturwissenschaftlichen Fakultät

der Universität zu Köln

vorgelegt von

Anna Engels-Putzka
geb. Engels

aus Siegburg

Köln
2009

Berichterstatter:	Prof. Dr. M. Dolg
	Prof. Dr. U. Deiters
Tag der mündlichen Prüfung:	19.01.2010

Contents

Kurzzusammenfassung

Diese Arbeit beschäftigt sich mit ausgewählten Aspekten einer effizienten Implementierung von Vielelektronen-Methoden, die im Formalismus der zweiten Quantisierung ausgedrückt werden können. Insbesondere wird die Coupled-Cluster-Methode (mit beliebigem Anregungsgrad) betrachtet. Diese Methode wird heute vielfach angewendet, insbesondere für genaue Rechnungen an kleinen und mittelgroßen Molekülen. Sie beruht auf einer nichtlinearen Parametrisierung der Wellenfunktion, was ihre Implementierung verhältnismäßig kompliziert macht.

Da der Rechenaufwand mit der Systemgröße stark ansteigt, ist es wichtig, die Implementierung möglichst effizient zu gestalten. Dies gilt insbesondere, wenn höhere Anregungen berücksichtigt werden sollen als in der Standardmethode CCSD (Coupled-Cluster mit Ein- und Zweifachanregungen). Das in dieser Arbeit beschriebene Programm enthält keine prinzipielle Einschränkung an den Anregungsgrad und soll zukünftig auch auf Multireferenzmethoden erweitert werden.

Nach einem Überblick über die relevante Theorie und die Struktur des Programms werden zwei Programmteile vertieft behandelt. Der erste ist Bestandteil einer automatischen Formelgenerierung und ermöglicht es, äquivalente Terme in Formeln zu erkennen und diese dadurch zu vereinfachen. Der verwendete Algorithmus beruht auf der Interpretation algebraischer Ausdrücke als Graphen. Die automatische Formelgenerierung kann auf beliebige Operatorausdrücke in zweiter Quantisierung angewendet werden.

Im Fall von Coupled-Cluster müssen die so erzeugten Gleichungen iterativ gelöst werden, daher ist die Auswertung der darin auftretenden Terme entscheidend für die Effizienz des Programms. Diese Auswertung kann auf eine Folge von Tensorkontraktionen zurückgeführt werden. Ein zentraler Teil des Programms und dieser Arbeit ist die Implementierung einer generischen Tensorkontraktion. Diese wird dadurch erschwert, dass die auftretenden Tensoren in der Regel eine komplizierte Struktur haben. Der hier verfolgte Ansatz besteht darin, jede Kontraktion auf eine Reihe von Matrixmultiplikationen zurückzuführen, da diese Operationen auf modernen Rechnern besonders schnell ausgeführt werden können. Dies erfordert aber eine vorherige Umspeicherung der Tensoreinträge. Die Optimierung dieses Schrittes wird ausführlich diskutiert.

Erste Tests zeigen, dass das hier beschriebene Programm mindestens so schnell ist wie das effizienteste bisher bekannte allgemeine Coupled-Cluster-Programm, und die Relation verbessert sich für größere Systeme zugunsten unseres Programms, da dann

die Matrixmultiplikation der dominierende Schritt ist.

Am Ende der Arbeit wird ein Ausblick auf mögliche Weiterentwicklungen gegeben. Optimierungspotential bietet insbesondere die Vorbereitung der Gleichungen (Faktorisierung) vor der eigentlichen Auswertung. Dieses wird im Vergleichsprogramm schon teilweise genutzt.

Abstract

This thesis deals with selected aspects of a new implementation of many-body methods which can be formulated in the framework of second quantization, in particular the coupled-cluster (CC) method with arbitrary excitation level. Coupled-cluster is one of the most successful and widely used quantum chemical methods for accurate calculations on small to medium-sized molecules. Since it employs a nonlinear parametrization of the wave function, its implementation is rather difficult, in particular if higher (i.e. more than double) excitations are to be included. The latter is necessary to obtain highly accurate results and also as a prerequisite for the generalization to multi-reference cases.

The implementation described here has a twofold focus. One is on generality and flexibility regarding the method to be implemented, the other is on efficiency. To achieve flexibility, it is useful to have a machinery which automatically derives working equations for a given method. We realize this by applying techniques of second quantization. This work treats in particular the last step of this procedure, namely the simplification of the resulting equations by the identification of equivalent terms. The algorithm used here is based on the interpretation of algebraic terms as graphs.

The derived CC equations then have to be solved iteratively. The efficiency of the program is mainly determined by the evaluation of the occurring expressions, which has to be done in each iteration step. The evaluation is split up in a sequence of tensor contractions. Their generic implementation is complicated by the particular structure of the involved tensors. We reduce each contraction to a sequence of matrix multiplications, which requires a previous data rearranging. But since matrix multiplication is the most efficient operation on modern computers, this additional effort pays off. Preliminary tests show that our program is at least as fast as the most efficient general coupled-cluster implementation so far, and the relation is expected to improve for calculations with larger basis sets where the matrix multiplication becomes the time-determining step.

Finally, we give an outlook to possible further developments. In particular, the preparation of the equations before the actual evaluation (factorization) offers much potential for optimizations which we do not exploit at the moment, in contrast to the program with which we compare, which employs at least a partial optimization at this point.

1. Introduction

1.1. Motivation

The main objective of nonrelativistic quantum chemistry is to calculate observable properties of chemical entities, e.g. molecules or solids, by approximately solving the electronic Schrödinger equation. Although most applications of chemical interest involve rather large systems, it is also desirable to have highly accurate methods applicable to small or medium-sized molecules only. For obtaining accurate results, it is in particular necessary to take into account the electron correlation.

We restrict our attention here to wave function-based correlation methods. The wave function is usually expanded in a set of (many-particle) basis functions. These, in turn, are (linear combinations of) anti-symmetrized products of one-particle (i.e. depending on the coordinates of one electron) basis functions (orbitals). While the one-particle basis set in practical calculations is necessarily finite, and thus incomplete, it is in principle possible to calculate the exact many-particle solution within a given one-particle basis by the so-called *full configuration-interaction* (FCI) method (see 2.1.3.2). But in practice, FCI calculations are infeasible for all but very small molecules, since their cost grows exponentially with the size of the system. Therefore, further approximations are necessary. The quality of an approximation is determined by the subset of the FCI space (complete many-particle basis) in which the wave function is expanded, by the number of independent parameters it contains, and by the way these parameters are optimized.

So far, one of the most successful schemes for applying such approximations in a systematic way is the hierarchy of coupled-cluster (CC) methods, which employs an exponential ansatz for the wave function, in contrast to the linear parametrization in CI. For reviews on coupled-cluster theory and applications see e.g. [1–5]. A brief account will be given in section 2.2.

Coupled-cluster was first developed in the context of nuclear physics [6,7], and later transferred to quantum chemistry by Čížek and Paldus [8–10] who also applied it to simple model systems. First practical implementations of coupled-cluster with only double excitations (CCD) or single and double excitations (CCSD) were reported by Bartlett and Purvis [11,12] and Pople and coworkers [13].

In order to obtain very accurate results, higher excitations have to be included. Therefore, successively implementations treating up to five-fold excitations appeared, e.g. [14–17]. But while CCSD can nowadays be routinely applied to large classes of

molecules, the steep scaling of the computational cost with the system size limits the applicability for higher excitations. Therefore, many schemes have been developed to approximately include higher excitation effects. The most popular one is CCSD(T) [18], which yields satisfactory results in many cases. However, there are certain classes of applications where it fails, including excited states, radicals, or the description of potential energy surfaces for dissociations.

The main problem is that in these cases the reference function (usually a single determinant) which is used as a starting point for coupled-cluster is not a reasonable approximation any more. The qualitatively correct description of the wave function then requires a multi-reference ansatz. In principle, this could be compensated by including sufficiently high excitations, but this would lead to unreasonably high costs. One possible solution for this is to take not all, but only the most important higher excitations, leading to a "multi-reference coupled-cluster method based on the single-reference formalism" [19]. There are many variants of this method, for a recent review see [20]. But none of these is fully satisfactory, so it is still desirable to have a genuine multi-reference CC formalism. Unfortunately, in contrast to the linear configuration interaction ansatz, the generalization of coupled-cluster to the multi-reference case is not straightforward. As a consequence, many different MRCC methods have been developed, and each has its advantages and disadvantages. The basic problems and some of these approaches will be discussed in chapter 2.

There have been other attempts to modify the single-reference CC ansatz in order to model multi-reference situations. For excited states, linear response (LR) methods [21–24] or equation-of-motion (EOM) CC (see [25] for a comprehensive description and [26] for a recent review) can be applied. Another approach is the method of moments for coupled-cluster (MMCC) [27], of which numerous variants exist. All these methods have in common that they can be used to calculate corrections to the SRCC energy or certain properties, e.g. excitation energies, but do not yield wave functions.

The computational costs of real multi-reference methods are rather high and depend in most cases not only on the size of the one-particle basis and the excitation level, but also on the number of references and active electrons. Thus their applicability is usually limited to rather small systems. Nevertheless, it is worthwhile to reduce these costs as much as possible – in the limits of a given method – by an efficient implementation.

The practical implementation of many multi-reference methods brings about the need to deal with (selected) higher excitations, namely if excitations from one reference are interpreted with respect to another reference. So the treatment of high excitations can be seen as a prerequisite for the implementation of these multi-reference methods.

While there are many efficient implementations for coupled-cluster with fixed maximal excitation level, in particular CCSD [28–36], only few attempts have been made so far to efficiently implement coupled-cluster with arbitrary excitation level or MRCC

methods. One example is the "Tensor Contraction Engine" (TCE) developed by Hirata and coworkers [37, 38] and its successor SMITH ("symbolic manipulation interpreter for theoretical chemistry") [39, 40]. Both can be used for various methods. The most efficient coupled-cluster implementation for arbitrary excitations so far is (to our knowledge) that by Kállay and Surján [41]. They also implemented MRCC based on the single-reference formalism [42].

The key for an efficient implementation is the optimal use of the available processor capacity. To reach this is not trivial, since during the last years, the clock frequency of the processors has increased much faster than the memory bandwidth, so the main problem is to access the data to be processed. This can be optimized by a consecutive storage of data and by exploiting the memory hierarchy of registers, processor caches, and main memory. However, in the case of coupled-cluster the data is not given naturally in a way that it can be easily processed. In particular, the same data sets have to be used several times in different order, so that there is no optimal way of storing them.

The time-determining step of any coupled-cluster calculation can be reduced to a sequence of tensor contractions. In view of the last paragraph, it is beneficial to reformulate these as vectorized operations, e.g. matrix multiplications. For these operations highly optimized implementations are available, which are adapted to the respective processor architecture. Due to cache effects, the most efficient operation is matrix–matrix multiplication, in particular for larger matrices.

This reformulation requires a previous rearrangement of the data, but this additional effort is compensated by the gain in efficiency through using matrix multiplications.

1.2. Scope of the Thesis

The goal of this thesis is to describe selected aspects of the coupled-cluster implementation developed in our group. It is intended to be used primarily for calculations with the MRexpT method of M. Hanrath [43]. A pilot implementation which can treat this and other multi-reference methods exists and has been successfully applied to several test systems [44–49]. Our aim is to have not only a pilot or test implementation, but a competitive program in the area of general coupled-cluster implementations. So our main focus – after, of course, correctness of the results – is on efficiency. But we are also interested in a high degree of flexibility, so that large parts of the program can be used for different methods. So far, we have implemented SRCC with (in principle) arbitrary excitation level.

After briefly reviewing theoretical concepts, in particular single- and multi-reference coupled-cluster methods, in chapter 2, we discuss general issues concerning the implementation in chapter 3. In particular, the different parts of our program are described. Roughly, the implementation consists of three steps:

1. Derivation of working equations
2. Preparation of the equations for evaluation
3. Evaluation and solution of the equation system

The second point is not subject of this thesis and will only be briefly touched. The other two are explicated in some more detail. In the following, one aspect of each is discussed at length.

First, in chapter 4 we deal with the term simplification used to minimize the number of terms in the coupled-cluster equations, which is the last step in the first part. After describing the problem, we discuss two different approaches to its solution: a purely algebraic one and a graph-based algorithm. The latter one is used in the actual program, since it is more generally applicable.

The topic of chapter 5 is the tensor contraction used in the evaluation of the equations. As indicated in the previous section, this is a central part of the whole implementation, since it mainly determines the efficiency. We implemented an algorithm based on matrix–matrix multiplication, at the expense of a rather complex data rearranging step. This is necessary because of the particular structure of the tensors occurring in coupled-cluster calculations. This structure is explained, as well as alternative approaches to the contraction problem, before we come to our actual implementation in 5.3. In that section we discuss in particular the addressing structures used to access tensor entries during the rearranging process. Since this part is particularly critical for the efficiency of the program, we put much effort into its optimization.

In the last chapter, we summarize our results and give an outlook to possible future developments.

1.3. Technical Remarks

Our coupled-cluster program is written in C++. Using an object oriented programming language allows for a rather high level of abstraction, and many mathematical objects (e.g. tensors, matrices, groups, permutations) are directly represented as objects in the program. Moreover, by using polymorphism (templates, virtual functions) the program can be made flexible. For example, we make extensive use of data containers from the STL (Standard Template Library, see e.g. [50]), in particular `vector` and `map`. (Here and in the following, if class names or other pieces of code are used in the text, they are written in `typewriter font`.) On the other hand, C++ offers enough freedom for writing an efficient program by using e.g. pointers and bit operations.

For the matrix multiplication, we use the standard routine `dgemm` (double-precision general matrix–matrix multiplication) from the BLAS (Basic Linear Algebra Subpro-

grams) library. This yields the optimal performance, independent of the specific processor architecture (if a different processor is used, only the library has to be exchanged).

To illustrate the (static) class relations of important parts of the program, we use UML diagrams. UML (Unified Modeling Language, see [51]) is a very powerful tool, not only for depicting class structures, but also for supporting the design process. A simple example to illustrate the usage of UML is shown in figure 1.1.

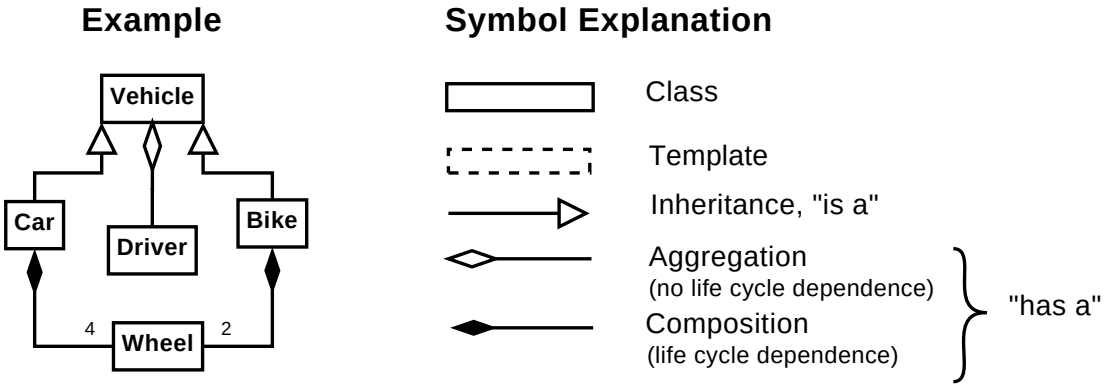


Figure 1.1.: Usage of UML symbols. The "arrows" are to be read from left to right (in the legend). Examples: "A car is a vehicle." "A car has a wheel." (Which is an integral part of the car.) "A car has a driver." (Which exists independent of the car.)

Other graphics which are used to illustrate program structures or algorithms do not follow a uniform convention and are explained where needed.

2. Theory

2.1. Foundations

2.1.1. The Electronic Schrödinger Equation

Our starting point is the (time-independent) Schrödinger equation

$$\hat{H}\Psi = E\Psi \tag{2.1}$$

which describes stationary states of (nonrelativistic) quantum mechanical systems. Here $\hat{H}$ is the Hamiltonian, Ψ the wave function and E the energy. For a molecule, $\hat{H}$ has the following form (in atomic units):

$$\hat{H} = \sum_I \left(-\frac{1}{2m_I}\Delta_I\right) + \sum_i \left(-\frac{1}{2}\Delta_i - \sum_I \frac{Z_I}{r_{iI}}\right) + \sum_{i<j} \frac{1}{r_{ij}} + \sum_{I<J} \frac{Z_I Z_J}{r_{IJ}} \tag{2.2}$$

where the indices I and J run over the nuclei, i and j are electron indices, m_I is the mass and Z_I the charge of nucleus I, r_{ij} is the distance between electrons i and j, r_{IJ} that between nuclei I and J, and r_{iI} the distance between electron i and nucleus I.

In the following we assume the Born–Oppenheimer approximation to hold, that means we treat the nuclei – which are much heavier than the electrons and therefore move much slower – as fixed and consider only the electronic Schrödinger equation

$$\hat{H}_{\text{el}}\Psi_{\text{el}} = E_{\text{el}}\Psi_{\text{el}} \tag{2.3}$$

with

$$\hat{H}_{\text{el}} = \sum_i \left(-\frac{1}{2}\Delta_i - \sum_I \frac{Z_I}{r_{iI}}\right) + \sum_{i<j} \frac{1}{r_{ij}} + \sum_{I<J} \frac{Z_I Z_J}{r_{IJ}} \tag{2.4}$$

$$= \sum_i \hat{h}(i) + \sum_{i<j} \hat{g}(i,j) + \hat{h}_{\text{nuc}} \tag{2.5}$$

The one-electron operator $\hat{h}$ contains the kinetic energy and the electron-nuclear interaction. The last term is treated as a constant, since the electronic Hamiltonian contains the coordinates of the nuclei only as parameters. The values of the electronic energy E_{el} for all possible positions of the nuclei form the potential energy hypersurface (PES)

of the molecule. In the following we drop the subscript el, since we only deal with the electronic problem.

The wave function Ψ depends explicitly on the coordinates of all electrons, i.e. their positions $\mathbf{r}_i$ and their spins. Besides being a solution of (2.3), Ψ has to fulfill another important criterion, namely the Pauli principle. This states that two identical fermions (in particular two electrons) may not occupy the same quantum state (therefore it is also known as the exclusion principle). The mathematical formulation of this is that Ψ has to be antisymmetric with respect to permutations of the electrons (so in particular it vanishes if two of them have the same coordinates).

Other properties of Ψ follow directly from it being a solution of (2.3), i.e. an eigenfunction of $\hat{H}$. Since commuting operators have the same eigenfunctions, Ψ also has to be an eigenfunction of all operators that commute with the Hamiltonian[a]. Since the nonrelativistic Hamiltonian does not depend on the electron spins, it commutes with the total spin operator $\hat{S}^2$, and therefore Ψ is an eigenfunction of $\hat{S}^2$, too (such a function is briefly called a spin eigenfunction). If the molecule has a nontrivial symmetry group (point group), $\hat{H}$ commutes with all operators contained in this group and Ψ is also an eigenfunction of them.

Because of the electron-electron interaction (i.e. the two-particle operators $\hat{g}(i,j)$) contained in the Hamiltonian, the Schrödinger equation (2.3) can not be solved exactly for many-electron systems. When approximate solutions are constructed, it is desirable to conserve as many of the symmetry properties of the exact wave function as possible, but – except for the Pauli principle – this is often difficult to achieve.

2.1.2. The Hartree–Fock Method

If the Hamiltonian of a system can be written as a sum of one-electron operators, the corresponding wave function is a product of one-electron functions (orbitals) ϕ_i:

$$\Psi(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \prod_{i=1}^{n} \phi_i(\mathbf{x}_i), \tag{2.6}$$

where $\mathbf{x}_i$ is the vector containing the coordinates (position and spin) of electron i. This motivates the wave function ansatz for the Hartree–Fock (HF) method. To account for the Pauli principle, the product has to be antisymmetrized, forming a so called *Slater determinant*:

$$\Phi(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \tfrac{1}{\sqrt{n!}} \sum_{\hat{P} \in S_n} \operatorname{sgn}(\hat{P}) \hat{P} \prod_{i=1}^{n} \phi_i(\mathbf{x}_i) \tag{2.7}$$

[a] $\hat{H}$ as defined in (2.5) is obviously invariant under permutations of the electrons, i.e. it commutes with all permutation operators $\hat{P} \in S_n$, where n is the number of electrons. It follows that Ψ is an eigenfunction of all these $\hat{P}$. But a priori each eigenvalue could be 1 or -1, so that the Pauli principle is really an additional requirement.

where the permutation operator $\hat{P}$ permutes the electrons and $\mathrm{sgn}(\hat{P})$ denotes the parity of the permutation. The prefactor $\frac{1}{\sqrt{n!}}$ ensures that Φ is normalized, provided that the orbitals are orthonormal.

The orbitals ϕ_i are determined by applying the variation principle, i.e. by minimizing the energy expectation value $\langle\Phi|\hat{H}|\Phi\rangle$ (with the full Hamiltonian $\hat{H}$) subject to the constraint of orthonormality. This leads to the Hartree–Fock equation

$$\hat{f}\phi_i = \epsilon_i\phi_i \tag{2.8}$$

The Fock operator $\hat{f}$ is an effective one-electron operator which contains the one-electron part of the Hamiltonian, while the two-electron part is replaced by an effective potential (Fock potential). The eigenvalues ϵ_i of $\hat{f}$ are called orbital energies. Since $\hat{f}$ depends on the orbitals ϕ_i, equation (2.8) has to be solved iteratively. There are several possibilities how to do this. For molecules a common approach is to expand the ϕ_i, which are then called molecular orbitals (MOs), in a basis of atomic orbitals (AOs) χ_j:

$$\phi_i = \sum_j C_{ij}\chi_j \tag{2.9}$$

The expansion coefficients C_{ij} are then the variational parameters and the Hartree–Fock equation can be transformed into a matrix equation (Roothaan equation).

2.1.3. Electron Correlation

In statistics, correlation (roughly) means the deviation of two random variables from being independent. The probability distributions of electrons in a molecule are certainly correlated in this sense, but in quantum chemistry, electron correlation is defined a bit differently. Here, the *correlation energy* is the difference between the exact nonrelativistic energy (i.e. FCI in the basis set limit) and the Hartree–Fock energy (also in a complete basis):

$$E_{\text{corr}} = E - E_{HF} \tag{2.10}$$

(Since HF is a variational method, the correlation energy is always negative.) This differs from the mathematical definition, since also in the HF model the electrons are not completely independent. The HF wave function fulfills the Pauli principle, therefore electrons with the same spin can never be at the same place (this is sometimes referred to as Pauli or Fermi correlation). What is neglected in the HF approximation is the Coulomb correlation, i.e. the coupling of the movements of the electrons due to their electrostatic interaction. To include this, we have to go beyond HF, e.g. by approximating the wave function as a linear combination of several Slater determinants.

Here we have to distinguish two cases: For many systems HF is a reasonable approximation, that means the correlation energy makes up only a small fraction of the total energy and the other (possibly many) determinants make small contributions to

the total wave function. This is often referred to as *dynamical correlation*. In contrast, there are situations in which two or more determinants are (almost) equally important. This is called *static correlation*, and systems where static correlation is important are often termed multi-reference systems. To get a good approximate wave function in this case, both statical and dynamical correlation effects have to be taken into account.

2.1.3.1. Multi-configuration Self-consistent Field Method

One way to treat static correlation is provided by the multi-configuration self-consistent field (MCSCF) method. Instead of a single determinant, the wave function is set up as a linear combination of several (important) determinants, and the expansion coefficients are optimized together with the orbitals, which makes this method considerably more complicated than Hartree–Fock. A special case is the complete active space SCF, where the set of determinants is constructed by distributing electrons in all possible ways within a (suitably chosen) set of active orbitals.

2.1.3.2. Configuration Interaction

In contrast to MCSCF, the *configuration interaction* (CI) method is usually used to describe dynamical correlation effects. The wave function here is also a linear combination of determinants, which is usually much longer than in MCSCF calculations. But then only the expansion coefficients are optimized while the orbitals, obtained e.g. from a HF calculation, are kept fixed.

If $|\Phi_0\rangle$ is the HF determinant, the CI wave function is a linear combination of $|\Phi_0\rangle$ and determinants which are obtained from $|\Phi_0\rangle$ by replacing one or more of the occupied orbitals by unoccupied orbitals (this process commonly is referred to as excitation, although it does not necessarily correspond to a physical excitation; a more accurate term would be substitution):

$$\Psi_{\text{CI}} = c_0|\Phi_0\rangle + \sum_{\alpha\in\mathbb{Q}} c_\alpha|\alpha\rangle = c_0|\Phi_0\rangle + \sum_{\alpha\in\mathbb{Q}} c_\alpha\hat{\tau}_\alpha|\Phi_0\rangle \tag{2.11}$$

where $\mathbb{Q}$ is the chosen set of excited determinants and the $\hat{\tau}_\alpha$ are excitation operators. Usually, the excited determinants are classified by their excitation degree (i.e. the number of exchanged orbitals) and all excitations up to a given degree are included in the expansion (2.11). For example, in the CISD method all single and double excitations are taken. Let $\mathbb{O}$ be the set of orbitals occupied in $|\Phi_0\rangle$ and $\mathbb{V}$ the set of unoccupied (virtual) orbitals. If we denote by Φ_i^a the determinant where the orbital $i \in \mathbb{O}$ has been replaced by $a \in \mathbb{V}$, and analogously for higher excitations, the CISD wave function can be written as

$$\Psi_{\text{CISD}} = c_0|\Phi_0\rangle + \sum_{\substack{i\in\mathbb{O}\\ a\in\mathbb{V}}} c_i^a|\Phi_i^a\rangle + \sum_{\substack{i,j\in\mathbb{O}\\ a,b\in\mathbb{V}}} c_{ij}^{ab}|\Phi_{ij}^{ab}\rangle \tag{2.12}$$

If all excitations up to the level which is equal to the number n of electrons are included, the FCI wave function is obtained. The number of determinants in the expansion is then $\binom{m}{n}$, where $m = |\mathbb{O}| + |\mathbb{V}|$ is the number of orbitals. Since this binomial coefficient increases rapidly with growing m and n (i.e. system size), FCI is only feasible for small systems (up to about ten electrons) and limited basis sets.

The coefficients c_α are obtained by minimizing the energy expectation value $\langle\Psi|\hat{H}|\Psi\rangle$ over all wave functions of the form (2.11). So CI is also a variational method. The CI energy is then given as

$$E_{\text{CI}} = \frac{\langle\Psi_{\text{CI}}|\hat{H}|\Psi_{\text{CI}}\rangle}{\langle\Psi_{\text{CI}}|\Psi_{\text{CI}}\rangle} \tag{2.13}$$

It can be shown that the minimization is equivalent to solving the eigenvalue problem

$$\mathbf{Hc} = E_{\text{CI}}\mathbf{c} \tag{2.14}$$

where the vector $\mathbf{c}$ contains the coefficients and $\mathbf{H}$ is the Hamilton matrix with entries

$$H_{\alpha\beta} = \langle\Phi_\alpha|\hat{H}|\Phi_\beta\rangle\,, \quad \{\Phi_\alpha\} \text{ basis.}$$

The CI ansatz can be rather easily generalized to the multi-reference case by including excitations up to a given level not only from one determinant, but from all determinants which are important for the qualitative description of the system (i.e. have a large coefficient in the wave function expansion). We denote the set of reference determinants by $\mathbb{P}$ and define $\mathbb{Q}_i(\mu)$ as the set of all determinants reached by at most i-fold excitations from the determinant μ. If we set

$$\mathbb{Q} = \bigcup_{i=1}^{l} \bigcup_{\mu\in\mathbb{P}} \mathbb{Q}_i(\mu) \setminus \mathbb{P} \tag{2.15}$$

where l is the chosen maximal excitation level (e.g. $l = 2$ for MRCISD), we can write the multi-reference CI wave function as

$$\Psi_{\text{MRCI}} = \sum_{\mu\in\mathbb{P}} c_\mu|\mu\rangle + \sum_{\alpha\in\mathbb{Q}} c_\alpha|\alpha\rangle. \tag{2.16}$$

The distinction between reference and excited determinants is somewhat artificial in this expression, but we want to use the notation later. Note that we do not have a representation analogous to the last part of (2.11) here. Since the union in (2.15) is not disjoint, there is usually no unique reference determinant from which an excited determinant is generated. This is no serious problem here (because the expansion is linear), but will be an important point later in the discussion of multi-reference coupled-cluster.

2.1.4. Size Consistency and Size Extensivity

We briefly discuss these two concepts to establish our terminology, since the terms are used differently in the literature (e.g. what we call size consistency is referred to as size extensivity in [52], while in [53] it is the other way round). For a more detailed discussion, in particular in the context of coupled-cluster , see e.g. [1, 43, 54, 55].

According to the convention we adopt here (following Pople et al. [56]), the notion of size consistency refers to the treatment of non-interacting systems. For a size consistent method, the energy of a system consisting of non-interacting subsystems is equal to the sum of the energies of the subsystems. This means if we have two systems A and B which do not interact (this is equivalent to saying that the Hamiltonian of the combined system AB can be written as the sum of the Hamiltonians of the two systems: $\hat{H}_{AB} = \hat{H}_A + \hat{H}_B$), then for a size consistent method $E_{AB} = E_A + E_B$ holds (additive separability of the energy). This property follows if the wave function for such systems is multiplicatively separable, but this is not a necessary condition. Some authors [57, 58] extend the term consistency to require also the correct description of the dissociation of a molecule into fragments (asymptotic consistency).

Size extensivity implies the correct scaling of the energy (or other extensive properties) with the system size and is related to statistical properties of the parameters of the method (connectivity). For the correlation energy of a system consisting of N identical subsystems it can be expressed as

$$\lim_{N\to\infty} \frac{E_{\text{corr}}(N)}{N} = c > 0.$$

The term size extensivity was introduced by Bartlett [11, 59] while the property itself had been studied earlier in the context of perturbation theory [60, 61].

Truncated configuration interaction methods (single- as well as multi-reference) are neither size consistent nor size extensive. This means in particular that their accuracy deteriorates with increasing system size. One of the major advantages of the (single-reference) coupled-cluster method described in the next section is that it is size consistent (provided the reference has this property) and size extensive.

2.2. Coupled-Cluster Theory

2.2.1. Second Quantization

We briefly recall some facts and definitions, mainly to introduce the notations. More details can be found e.g. in references [1,52,53].

2.2.1.1. Basic Definitions

The formalism of second quantization starts from a set of spin orbitals $\{\phi_p\}$ and uses so called annihilation and creation operators which are defined by their effect on determinants built from these orbitals:

$$\begin{aligned} \hat{a}_p|\phi_p\phi_q\dots\phi_s\rangle &= |\phi_q\dots\phi_s\rangle \quad \text{(annihilation of an electron)} \\ \hat{a}_p^\dagger|\phi_q\dots\phi_s\rangle &= |\phi_p\phi_q\dots\phi_s\rangle \quad \text{(creation of an electron)} \end{aligned}$$

The creation operator is the adjoint of the corresponding annihilation operator and vice versa. The operators fulfill the following anticommutation relations:

$$\hat{a}_p\hat{a}_q + \hat{a}_q\hat{a}_p = 0 \tag{2.17}$$
$$\hat{a}_p^\dagger\hat{a}_q^\dagger + \hat{a}_q^\dagger\hat{a}_p^\dagger = 0 \tag{2.18}$$
$$\hat{a}_p^\dagger\hat{a}_q + \hat{a}_q\hat{a}_p^\dagger = \delta_{pq}. \tag{2.19}$$

A string of annihilation and creation operators is said to be in normal order if all annihilation operators stand to the right of all creation operators. This is useful for the evaluation of matrix elements, since annihilation operators yield zero when applied to the vacuum state.

2.2.1.2. Particle-Hole Formalism

For coupled-cluster theory it is useful to redefine the notion of normal order with respect to a reference determinant $|\Phi_0\rangle$ which is then also called the Fermi vacuum. The orbitals occupied in $|\Phi_0\rangle$ are called hole or occupied orbitals and denoted by the indices $i, j, k, \dots$ For them the meaning of annihilation and creation is reversed, i.e. $\hat{a}_i$ is considered as a creation operator (because when acting on $|\Phi_0\rangle$ it removes an electron and therefore creates a hole) while $\hat{a}_i^\dagger$ is now an annihilation operator (it annihilates a hole). For the orbitals not occupied in $|\Phi_0\rangle$ we use the indices $a, b, c, \dots$ and they are referred to as particle or virtual orbitals. Like in 2.1.3.2, we denote the sets of these orbitals by $\mathbb{O}$ and $\mathbb{V}$, respectively.

We define the substitution operators $\hat{\tau}_{i,a} = \hat{a}_a^\dagger \hat{a}_i$ or, more generally,

$$\hat{\tau}_{i_1 i_2 \ldots, a_1 a_2 \ldots} = \hat{a}_{a_1}^\dagger \hat{a}_{a_2}^\dagger \ldots \hat{a}_{i_2} \hat{a}_{i_1}$$

with $i_\nu \in \mathbb{O}$ and $a_\nu \in \mathbb{V}$. They have the property that they commute with each other as long as $\mathbb{O} \cap \mathbb{V} = \emptyset$ holds.

If a string of second-quantized operators is normal-ordered according with respect to $|\Phi_0\rangle$ we denote this by curly brackets $\{\}$. Within these brackets, all operators exactly anticommute.

2.2.1.3. Wick's Theorem

With the help of Wick's theorem we can write an operator string as a sum of normal-ordered strings more easily than by using only the anticommutator relations. In order to formulate the theorem we first define the contraction of two second-quantized operators:

$$\overbrace{\hat{A}\hat{B}} = \hat{A}\hat{B} - \{\hat{A}\hat{B}\} \tag{2.20}$$

There are only two possible combinations of annihilation and creation operators for which this contraction is not zero, namely

$$\overbrace{\hat{a}_i^\dagger \hat{a}_j} = \delta_{ij} \quad \text{and} \quad \overbrace{\hat{a}_a \hat{a}_b^\dagger} = \delta_{ab}. \tag{2.21}$$

Now Wick's theorem states that a string of annihilation and creation operators is equal to the sum of all contracted normal-ordered products that can be built from these operators. Schematically, this can be written as:

$$\hat{A}\hat{B}\hat{C}\ldots\hat{X}\hat{Y}\hat{Z} = \{\hat{A}\hat{B}\hat{C}\ldots\hat{X}\hat{Y}\hat{Z}\} + \sum_{\substack{\text{single}\\ \text{contractions}}} \{\overbrace{\hat{A}\hat{B}}\hat{C}\ldots\hat{X}\hat{Y}\hat{Z}\} + \sum_{\substack{\text{double}\\ \text{contractions}}} \{\overbrace{\hat{A}\hat{B}}\overbrace{\hat{C}\hat{D}}\ldots\hat{X}\hat{Y}\hat{Z}\} + \ldots + \sum (\textit{fully contracted terms}).$$

The notation here is taken from [1], for the original formulation and proof see [62]. There is also a version of the theorem for a product of two strings which are already normal-ordered:

$$\{\hat{A}\hat{B}\hat{C}\ldots\}\{\hat{X}\hat{Y}\hat{Z}\ldots\} = \{\hat{A}\hat{B}\hat{C}\ldots\hat{X}\hat{Y}\hat{Z}\ldots\} + \sum_{\substack{\text{single}\\ \text{contractions}}} \{\overbrace{\hat{A}\hat{B}\hat{C}\ldots}\hat{X}\hat{Y}\hat{Z}\ldots\} + \sum_{\substack{\text{double}\\ \text{contractions}}} \{\overbrace{\hat{A}\hat{B}\overbrace{\hat{C}\ldots}\hat{X}\hat{Y}}\hat{Z}\} + \ldots + \sum (\textit{fully contracted terms}).$$

The crucial point here is that contractions within a normal-ordered factor do not have to be considered. This can be generalized to products with more than two factors and also to products where only some factors are already normal-ordered.

Helgaker et al. [52] and Harris et al. [53,63] use different strategies for the evaluation of matrix elements in second quantization which do not rely on any kind of normal order.

2.2.1.4. Hamilton Operator

The molecular electronic Hamiltonian can be written in second quantization as

$$\hat{H} = \sum_{pq} \langle p|\hat{h}|q\rangle \hat{a}_p^\dagger \hat{a}_q + \tfrac{1}{4} \sum_{pqrs} v_{\overline{rs}}^{\overline{pq}} \hat{a}_p^\dagger \hat{a}_q^\dagger \hat{a}_s \hat{a}_r. \tag{2.22}$$

Here,

$$v_{\overline{rs}}^{\overline{pq}} = \left\langle pq \,\middle|\, r_{12}^{-1} \,\middle|\, rs \right\rangle - \left\langle pq \,\middle|\, r_{12}^{-1} \,\middle|\, sr \right\rangle \tag{2.23}$$

are anti-antisymmetrized two-electron integrals (the antisymmetry holds within the two pairs of indices, i.e. $v_{\overline{rs}}^{\overline{pq}} = -v_{\overline{rs}}^{\overline{qp}} = -v_{\overline{sr}}^{\overline{pq}} = v_{\overline{sr}}^{\overline{qp}}$). The horizontal bar indicates the symmetry between the index pairs, since we have $v_{\overline{rs}}^{\overline{pq}} = v_{\overline{pq}}^{\overline{rs}}$ (for real orbitals).

The normal-ordered form of the Hamiltonian is given by

$$\hat{H}_N = \hat{H} - \langle\Phi_0|\hat{H}|\Phi_0\rangle = \hat{F}_N + \hat{V}_N = \sum_{pq} f_q^p \left\{\hat{a}_p^\dagger \hat{a}_q\right\} + \tfrac{1}{4} \sum_{pqrs} v_{\overline{rs}}^{\overline{pq}} \left\{\hat{a}_p^\dagger \hat{a}_q^\dagger \hat{a}_s \hat{a}_r\right\}, \tag{2.24}$$

where $f_q^p = \langle p|\hat{h}|q\rangle + \sum_i v_{\overline{qi}}^{\overline{pi}}$ are the matrix elements of the Fock operator. Analogous to the two-electron integrals, they have the symmetry $f_q^p = f_p^q$. $\hat{H}_N$ may be considered as a "pure" correlation operator and is therefore especially suitable for coupled-cluster theory.

2.2.2. Single-Reference Coupled-Cluster

2.2.2.1. The Exponential Ansatz

The coupled-cluster wave function is given by

$$|\Psi\rangle = e^{\hat{T}} |\Phi_0\rangle \tag{2.25}$$

where $|\Phi_0\rangle$ is a reference determinant (usually the Hartree–Fock wave function) and $\hat{T}$ is the so called cluster operator. This exponential ansatz can be justified by the introduction of cluster functions into the HF wave function which account for the electron correlation (see [1]). The cluster operator is an excitation operator which replaces occupied by virtual orbitals. It is usually partitioned according to the level of the excitation:

$$\hat{T} = \hat{T}_1 + \hat{T}_2 + \ldots + \hat{T}_n \quad \text{with}$$

$$\hat{T}_k = \frac{1}{(k!)^2} \sum_{\substack{i_1 \dots i_k \\ a_1 \dots a_k}} t_{i_1 \dots i_k}^{a_1 \dots a_k} \hat{a}_{a_1}^\dagger \dots \hat{a}_{a_k}^\dagger \hat{a}_{i_k} \dots \hat{a}_{i_1} = \frac{1}{(k!)^2} \sum_{\substack{i_1 \dots i_k \\ a_1 \dots a_k}} t_{i_1 \dots i_k}^{a_1 \dots a_k} \hat{\tau}_{i_1 \dots i_k, a_1 \dots a_k}. \tag{2.26}$$

The highest possible excitation level n is the number of (spin) orbitals occupied in $|\Phi_0\rangle$, i.e. the number of electrons in the system under consideration. The parameters $t_{i_1 \dots i_k}^{a_1 \dots a_k}$ are called amplitudes. They have to be determined in a way that the Schrödinger equation

$$\hat{H}_N e^{\hat{T}} |\Phi_0\rangle = E_{\text{corr}} |\Phi_0\rangle \tag{2.27}$$

is satisfied. It practice of course not the full equation can be solved, but only its restriction to a proper subspace.

2.2.2.2. Coupled-Cluster Equations

To get equations from which the energy and the amplitudes may be determined, the Schrödinger equation (2.27) is projected onto the reference determinant $|\Phi_0\rangle$ and all determinants $|\Phi_{i_1 \dots i_k}^{a_1 \dots a_k}\rangle$ which can be generated from $|\Phi_0\rangle$ by the application of the cluster operator. This leads to as many equations as there are unknowns to be determined, since the number of projections onto excited determinants matches the number of amplitudes and the projection onto the reference yields an additional equation for the energy:

$$\left\langle \Phi_0 \middle| \hat{H}_N e^{\hat{T}} \Phi_0 \right\rangle = E_{\text{corr}} \left\langle \Phi_0 \middle| e^{\hat{T}} \Phi_0 \right\rangle = E_{\text{corr}} \tag{2.28}$$

$$\left\langle \Phi_{i_1 \dots i_k}^{a_1 \dots a_k} \middle| \hat{H}_N e^{\hat{T}} \Phi_0 \right\rangle = E_{\text{corr}} \left\langle \Phi_{i_1 \dots i_k}^{a_1 \dots a_k} \middle| e^{\hat{T}} \Phi_0 \right\rangle. \tag{2.29}$$

The last equality in 2.28 holds because of the *intermediate normalization*

$$\left\langle \Phi_0 \middle| e^{\hat{T}} \Phi_0 \right\rangle = 1.$$

If the full cluster operator is used, coupled-cluster is equivalent to full CI, but with nonlinear equations for the amplitudes. Since this is intractable, the cluster operator has to be truncated. Most common is the truncation at a certain excitation level. For example, the inclusion of only single and double excitations ($\hat{T} = \hat{T}_1 + \hat{T}_2$) leads to the CCSD method.

Usually, before the projection the equation is multiplied from the left by $e^{-\hat{T}}$ in order to decouple the energy equation from the amplitude equations. This is the so-called *similarity transformed* or *linked* form of the coupled-cluster equations:

$$\left\langle \Phi_0 \middle| e^{-\hat{T}} \hat{H}_N e^{\hat{T}} \Phi_0 \right\rangle = E_{\text{corr}} \langle \Phi_0 \,|\, \Phi_0 \rangle = E_{\text{corr}} \tag{2.30}$$

$$\left\langle \Phi_{i_1 \dots i_k}^{a_1 \dots a_k} \middle| e^{-\hat{T}} \hat{H}_N e^{\hat{T}} \Phi_0 \right\rangle = E_{\text{corr}} \left\langle \Phi_{i_1 \dots i_k}^{a_1 \dots a_k} \middle| \Phi_0 \right\rangle = 0. \tag{2.31}$$

It can be shown that both sets of equations are equivalent under certain conditions on the excitations included in $\hat{T}$, e.g. if all excitations up to a given level are included. Besides the decoupling, the linked form has another advantage in that the evaluation of the left hand side is simplified by employing the so called Baker–Campbell–Hausdorff (BCH) expansion, which expresses the similarity transformation of an operator by the exponential of another operator in terms of nested commutators:

$$\begin{aligned}\exp\bigl(-\hat{B}\bigr)\,\hat{A}\exp\bigl(\hat{B}\bigr) &= \sum_{n=0}^{\infty}\frac{1}{n!}\bigl[\hat{A},\hat{B}\bigr]^{(n)} \\ &= \hat{A}+\bigl[\hat{A},\hat{B}\bigr]+\tfrac{1}{2}\bigl[\bigl[\hat{A},\hat{B}\bigr],\hat{B}\bigr]+\tfrac{1}{3!}\bigl[\bigl[\bigl[\hat{A},\hat{B}\bigr],\hat{B}\bigr],\hat{B}\bigr]+\ldots \end{aligned} \tag{2.32}$$

A proof of this formula is given in appendix A. In the case of coupled-cluster , i.e. $\hat{A}=\hat{H}$ and $\hat{B}=\hat{T}$, the series can be truncated after the fourth commutator [1, 53]. The reason for this is that the Hamiltonian we are dealing with, being a two-particle operator, has at most four free indices which can be "shared" with a cluster operator. If two operators do not have an index in common they commute, and so at latest the fifth commutator in (2.32) yields zero.

The evaluation of expressions like those on the left hand side of (2.28)–(2.31) will be discussed in the next chapters. Here we give only the result for the simplest case.

2.2.2.3. The CC Energy

The coupled-cluster energy can be expressed explicitly in terms of amplitudes and integrals:

$$\begin{aligned}E_{\text{corr}} &= \langle\Phi_0|\hat{H}_N\hat{T}\Phi_0\rangle+\tfrac{1}{2}\langle\Phi_0|\hat{H}_N\hat{T}^2\Phi_0\rangle \\ &= \sum_{ia} f_a^i t_i^a+\tfrac{1}{4}\sum_{ijab} v_{ab}^{ij}t_{ij}^{ab}+\tfrac{1}{2}\sum_{ijab} v_{ab}^{ij}t_i^a t_j^b. \end{aligned} \tag{2.33}$$

Although it contains only amplitudes of $\hat{T}_1$ and $\hat{T}_2$, this equation is valid for arbitrary excitation levels, since higher than double excitations can not interact with $|\Phi_0\rangle$ through $\hat{H}_N$.

2.2.2.4. Spin and the Parametrization of the Cluster Operator

If the cluster operator is defined as in (2.26), the CC wave function can in general not be expected to be a spin eigenfunction, since the substitution operators $\hat{\tau}$ do not commute with $\hat{S}^2$, i.e. they may change the total spin. To avoid this problem, it is possible to define spin averaged substitution operators $\hat{E}_{p_1p_2\ldots,q_1q_2\ldots}$, e.g. $\hat{E}_{p,q}=\hat{a}_q^\dagger\hat{a}_p+\hat{a}_{\bar{q}}^\dagger\hat{a}_{\bar{p}}$, where p_ν and q_ν are spatial orbitals and the bar denotes β spin. Then the cluster operators can be written in terms of these operators, e.g. $\hat{T}_1=\sum_{ia}t_i^a\hat{E}_{i,a}$, $\hat{T}_2=\frac{1}{2}\sum_{ijab}t_{ij}^{ab}\hat{E}_{i,a}\hat{E}_{j,b}$ (see for example [36, 52, 64–66]). This ansatz, however, leads to other problems when

applied to an open-shell reference function. Either, the substitutions do not span the space of spin eigenfunctions, or they have to be defined with overlapping orbital sets for annihilators and creators. The latter leads to a non-commuting set of operators, which makes their handling much more difficult (in the usual derivation of the CC working equations, the commutativity of the cluster operators is essential).

2.2.3. Multi-Reference Coupled-Cluster

The generalization of the coupled-cluster ansatz to the multi-reference case is not straightforward and there are many different approaches to the problem, of which we discuss a few in the following. Before, we discuss some of the principal difficulties.

2.2.3.1. General Considerations

One of the main problems is the ambiguity in the generation of excited determinants mentioned already in the discussion of MRCI (see 2.1.3.2). The set union in (2.15) makes only sense for "global" objects like determinants. In contrast to CI, which can be formulated in terms of excitation operators as well as in terms of determinants (compare (2.11)), CC relies on the representation in terms of excitation operators. This is due to the exponential ansatz, the argument of the exponential function consists of excitation operators and their coefficients. But these operators are "local" in the sense that they are always defined with respect to a particular reference determinant. Therefore the exponential ansatz can not be readily transferred to the multi-reference case.

Another major issue in setting up an MRCC ansatz is to retain the property of being size extensive (connected). Since this is a rather subtle point we do not discuss it further here.

2.2.3.2. Multi-Reference Ansatz Based on the Single-Reference Formalism (SRMRCC)

The previous problems are avoided if the single-reference formalism is kept and only the cluster operator is modified to model a multi-reference situation. This ansatz was introduced by Oliphant and Adamowicz for the case of two reference determinants differing by a double excitation [19] and later generalized. It requires the choice of a formal reference (Fermi vacuum) $|\mu_0\rangle$ from the reference space, with respect to which all excitations are defined. The cluster operator is then constructed in such a way that its application to $|\mu_0\rangle$ generates all determinants which would appear in a corresponding MRCI wave function, i.e. all excitations (up to a given level) from all references. This corresponds to doing a "normal" coupled-cluster calculation where the higher excitations are incomplete (the highest excitation level is the base excitation level plus the highest excitation between the formal Fermi vacuum and any other reference).

This method inherits many desirable properties from SRCC, in particular size extensivity. But it is not a genuine MRCC method. The main problem with it is that the choice of a formal reference introduces a certain arbitrariness and causes an imbalance in the wave function. It can also lead to a breaking of spin or spatial symmetry, in particular if there are two or more references with (almost) equal weight.

2.2.3.3. State-Universal Ansatz

The first genuine multi-reference coupled-cluster methods were the valence-universal or Fock space ansatz (VUMRCC or FSMRCC) [67–69] and the state-universal or Hilbert space ansatz (SUMRCC) [70]. We describe here only the latter, since it is the starting point for the MRexpT ansatz discussed below (section 2.2.3.4).

The SUMRCC ansatz was developed by Jeziorski and Monkhorst. It starts from an operator $\hat{\Omega}$, called wave operator, which maps a model (i.e. reference) space $\mathbb{P}$ to a set of exact solutions of the Schrödinger equation. This operator is determined by the equation

$$\hat{H}\hat{\Omega} = \hat{\Omega}\hat{H}\hat{\Omega}. \tag{2.34}$$

Making the ansatz

$$\hat{\Omega} = \sum_{\mu\in\mathbb{P}} e^{\hat{T}_\mu}|\mu\rangle\langle\mu|, \tag{2.35}$$

where $\hat{T}_\mu$ is a reference-specific cluster operator, leads to the following expression for the wave function:

$$|\Psi_\lambda\rangle = \sum_{\mu\in\mathbb{P}} c_{\lambda\mu} e^{\hat{T}_\mu}|\mu\rangle. \tag{2.36}$$

The reference coefficients $c_{\lambda\mu}$ can be obtained by diagonalizing the effective Hamiltonian $\hat{P}\hat{H}\hat{\Omega}$ in the reference space (here $\hat{P}$ is the projector onto the reference space). To determine the amplitudes which define the $\hat{T}_\mu$ it is necessary to consider at the same time as many states $|\Psi_\lambda\rangle$ as there are references (hence the term state-universal), since the amplitudes for the different references are independent and taking into account only one state would lead to an underdetermined problem. Explicit equations for the amplitudes can be derived from equation (2.34) by applying both sides to $|\Psi_\lambda\rangle$ and projecting onto the orthogonal complement of $\mathbb{P}$.

The original method of Jeziorski and Monkhorst is connected (and therefore size extensive) only in case of a complete model space. Later the ansatz has been generalized to general (incomplete) model spaces [71,72]. The main problems of SUMRCC are the occurrence of intruder states and the exceedingly high computational effort introduced by state-universality. Since usually one is only interested in one particular state, several attempts have been made to modify the ansatz in a way that it becomes state-specific.

2.2.3.4. Multi-Reference Exponential Wave Function Ansatz (MRexpT)

The key idea of this ansatz [43] is to reduce the number of parameters (compared to SUMRCC) by introducing a determinant based amplitude indexing (in contrast to the excitation based indexing usually employed in coupled-cluster theory). The ansatz for the wave function is

$$|\Psi\rangle = \sum_{\mu\in\mathbb{P}} c_\mu e^{\hat{T}_\mu}|\mu\rangle \tag{2.37}$$

with

$$\hat{T}_\mu = \phi(c_\mu) \sum_{\hat{\tau}_\mu\in\mathbb{T}_\mu} t_{\hat{\tau}_\mu|\mu\rangle}\hat{\tau}_\mu. \tag{2.38}$$

Here $\phi(c_\mu) = e^{-\arg(c_\mu)}$ is a phase compensation factor and $\mathbb{T}_\mu$ denotes the set of all excitations to be applied to $|\mu\rangle$ (e.g. all excitations up to a given level, where excitations to other references are excluded). So the cluster operators are still reference-specific, but the amplitudes are not all independent. Since they are labeled by $\hat{\tau}_\mu|\mu\rangle$, i.e. the result of an excitation applied to the reference $|\mu\rangle$, the same amplitude can occur in several cluster operators, if the corresponding determinant can be reached from different references. Since the determinants are only fixed up to sign, the rule $t_{-|\beta\rangle} = -t_{|\beta\rangle}$ has to be applied.

The wave function (2.37) is inserted into the Schrödinger equation and projections onto all determinants from the reference space $\mathbb{P}$ and from the space $\mathbb{Q} = \bigcup_{\mu\in\mathbb{P}} \mathbb{Q}(\mu)$, where $\mathbb{Q}(\mu) = \bigcup_{\hat{\tau}_\mu\in\mathbb{T}_\mu} \hat{\tau}_\mu|\mu\rangle$ is the set of excited determinants reached from μ, are applied. This yields a system of as many equations as there are amplitudes and reference coefficients. These equations are nonlinear in the amplitudes, but linear in the c_μ. However, since the energy is also unknown, another equation is needed to match the number of variables. This can be obtained by fixing the norm of the reference wave function, e.g.

$$\sum_{\mu\in\mathbb{P}} |c_\mu|^2 = 1. \tag{2.39}$$

MRexpT is size consistent [43], but not rigorously size extensive. However, it has been shown to be core extensive [73], which means that it scales correctly with the number of inactive electrons, which usually grows faster than the number of active electrons.

2.2.3.5. Other State-Specific Variants of SUMRCC

Another possibility to solve the redundancy problem is to define the wave function

$$|\Psi\rangle = \hat{\Omega}\sum_\mu c_\mu|\mu\rangle$$

with $\hat{\Omega}$ as in (2.35), insert it into the Schrödinger equation and then introduce sufficiency conditions by manipulating the projected equations in an appropriate manner.

This can be done in different ways, leading to different methods. One of these methods was developed by Mukherjee and coworkers (MkMRCC) [74, 75], another one is Brillouin–Wigner coupled-cluster (BWCC) [76–79]. In contrast to MkMRCC, BWCC is not size extensive, but a size extensivity correction has been developed [80]. These and related methods have been recently analyzed by Kong [81]. A numerical comparison, also with SUMRCC, was carried out by Evangelista et al. [82].

3. Implementation Overview

The coupled-cluster equations (2.31) have to be solved numerically, and this requires the numerical evaluation of the expressions on the left hand side of these equations. For this task, there are different ways to implement it. Many pilot implementations which aim at dealing with higher excitations or multi-reference approaches [82–85] use a technique which is also commonly used in (full) CI programs [86–88]. The cluster operator is applied recursively to the reference determinant, and then the necessary matrix elements of the Hamiltonian with these excited determinants are calculated.

This method is, however, not very efficient, in particular its complexity does not have the correct scaling with the number of orbitals. The main reason for this is that the evaluation of matrix elements is to a large extent redundant. It can as well be done in an abstract way, i.e. independent of the specific system and the values of the involved quantities, and therefore does not have to be repeated in each calculation (and each iteration within one calculation). So for an efficient implementation the equations are first cast into a more explicit form (like that given in (2.33) for the energy) which contains only the amplitudes appearing in the cluster operator (i.e. the unknowns to be determined), the integrals characterizing the system under consideration, and elementary operations. This transformation requires several steps which are described briefly in section 3.1.

The main elements of this type of implementation are shown in figure 3.1. The crucial step with respect to efficiency is the evaluation of the generated expressions. These expressions contain terms like $\sum_{ia} f^i_a t^a_i$ or $\sum_{iab} v^{Ai}_{ab} t^B_i t^{ab}_{IJ}$. Here I, J, A, B are the indices which label the excited determinant on which the equation is projected (see 2.2.2.2). They are called external indices. In general, all these terms have the form

$$\sum_{i_1 \ldots i_k} \prod_{n=1}^{N} X_{I_1^{(n)} \ldots I_{K_n}^{(n)}, i_1^{(n)} \ldots i_{k_n}^{(n)}} \tag{3.1}$$

with $i_j^{(n)} \in \{i_1, \ldots, i_k\}$ and $I_j^{(n)} \in \{I_1, \ldots, I_K\}$ (a given set of external indices). In the following, summation indices are always lower case letters, while upper case letters denote external indices.

In principle, the evaluation of such terms is a straightforward task – it requires only addition and multiplication –, but doing it efficiently is a not trivial at all, in particular if the involved tensors have a complicated structure. As a first step, the formal scaling

can be reduced by splitting up multiple summations and defining suitable so called intermediates, e.g.

$$\sum_{iab} v_{\overline{ab}}^{Ai} t_i^B t_{IJ}^{ab} = \sum_i t_i^B \underbrace{\sum_{ab} v_{\overline{ab}}^{Ai} t_{IJ}^{ab}}_{X_{IJ,i}^{AB}} = \sum_i t_i^B X_{IJ,i}^{AB} = Z_{IJ}^{AB}.$$

While the first expression here has a formal scaling of $\mathcal{O}(n^7)$, where n is the number of orbitals (since there are three summed and four external indices), the second one consists of two steps where each is $\mathcal{O}(n^5)$. In general, the coupled-cluster equations are rewritten in a form where in each step only two factors are multiplied (contracted). This transformation is commonly referred to as factorization (see e.g. [16, 41], in [37] this step is called "strength reduction"). Finding the optimal factorization is a complicated problem which is beyond the scope of this thesis.

The evaluation of the resulting binary contractions is treated in chapter 5, so we will not explicate it further at this point.

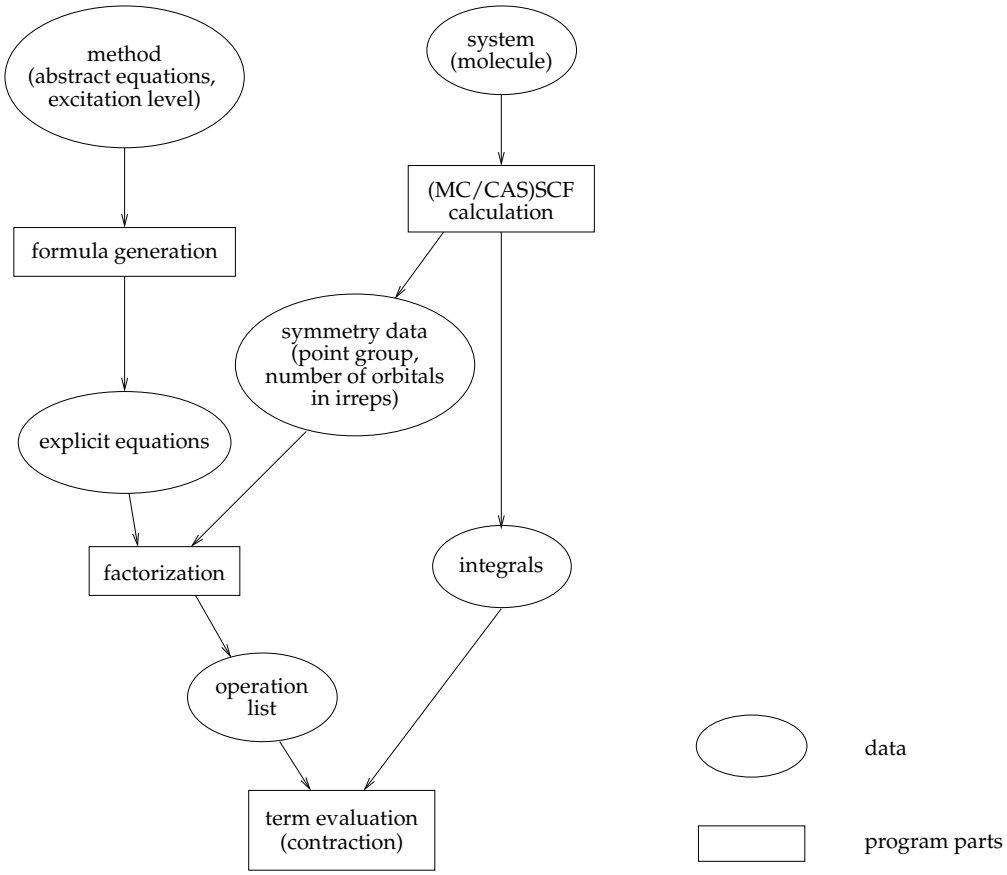


Figure 3.1.: Schematic representation of a coupled-cluster implementation

3.1. Formula Generation

3.1.1. Overview

Since the procedure is not only applicable to coupled-cluster equations, we consider here a more general situation. We start with the Fermi vacuum expectation value of an operator which can be expressed in terms of strings of second quantized operators. The aim is to generate an expression which contains only operator parameters (e.g. amplitudes and integrals) and constants and which is as simple (i.e. short) as possible. To reach this, a number of steps are performed. Figure 3.2 shows an overview together with an example. Listing 3.1 contains a piece of code which generates the CCSD equations [89]. In the following, line numbers refer to this.

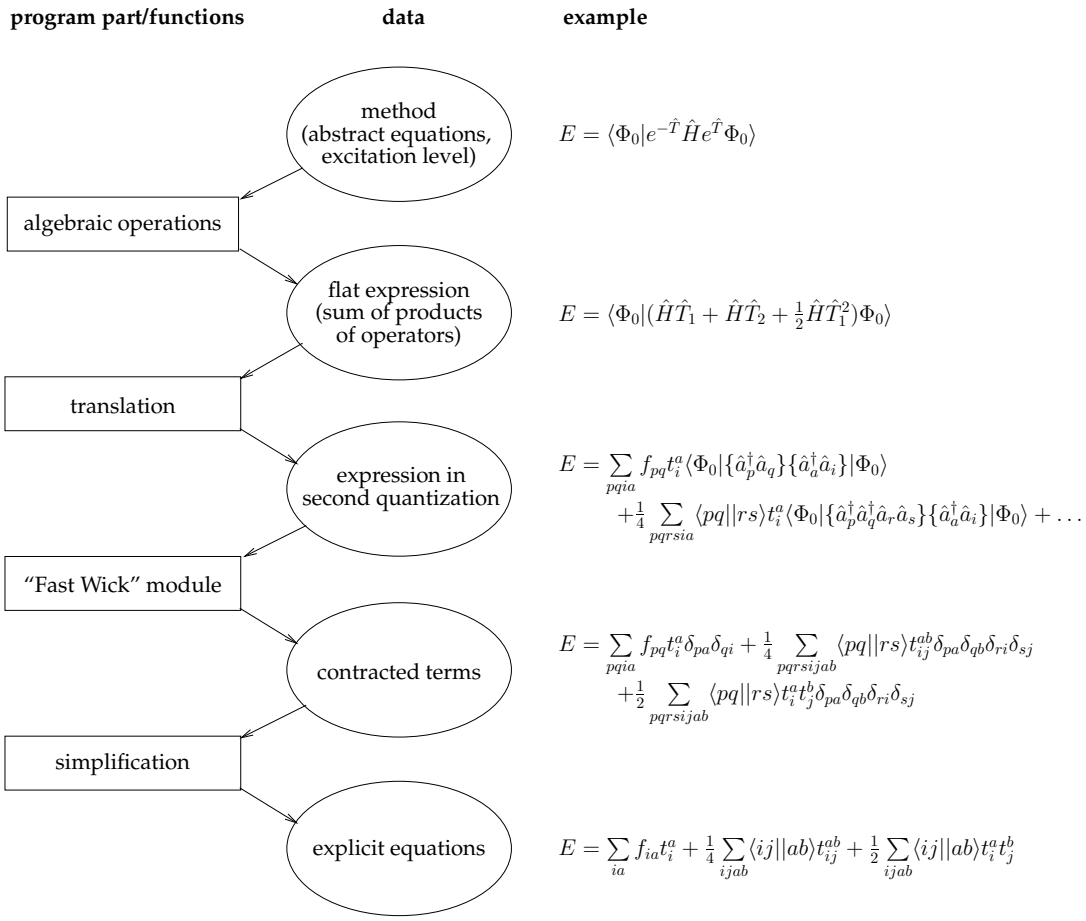


Figure 3.2.: Steps of the formula generation procedure

Listing 3.1: Generation of the CCSD equations

```
1  const int clusterLevel = 2;
2  const int projectionLevel = 2;
3
4  CompoundOperator_Expression TT;   //cluster operator
5    for ( int i=1 ; i<=clusterLevel ; ++i )
6      TT += T(i);
7  CompoundOperator_Expression HN = FN + VN; //Hamiltonian
8  CompoundOperator_Expression I=A(0);   //identity operator
9
10   for ( int i=0 ; i<=projectionLevel ; ++i )
11   {
12 //-------------------------------------------------------
13 //Specification of term/method
14   CompoundOperator_Expression expr(FV(A(-i)*exp(-TT)*HN*exp(TT)));
        // CC
15 //CompoundOperator_Expression expr(FV(A(-i)*HN*(I+TT)));  // CI
16 //-------------------------------------------------------
17   CompoundOperator_Product_Sum  flatExpr(expr);
18   TensorSymbols_ACOperators_Sum<SQIndex,
        ACOperator_Product_NOP_FV<SQIndex> > nopfs(flatExpr);
19   TensorSymbols_Kroneckers_Sum<SQIndex, true>
        expanded(expandFastWick(nopfs));
20   TermGraph_Sum tgs(expanded);
21   TensorSymbols_Sum<SQIndex, true> simplified(tgs);
22   }
```

In detail, the steps are:

1. *High Level Algebra* (line 12)
   If present, exponential series have to be evaluated (up to a certain degree), either directly or by invoking the BCH expansion (2.32). Then products of sums of operators are distributed to get an expression which is a sum of products of operators. Rank considerations can be used to reduce the number of terms at an early stage [89].

2. *Translation* (line 13)
   The "high level" operators have to be transformed into their second quantized representation, e.g.

$$\hat{H} = \sum_{pq} f^{p}_{q} \hat{a}^{\dagger}_{p} \hat{a}_{q} + \tfrac{1}{4} \sum_{pqrs} v^{pq}_{rs} \hat{a}^{\dagger}_{p} \hat{a}^{\dagger}_{q} \hat{a}_{r} \hat{a}_{s}, \quad \hat{T}_1 = \sum_{ia} t^{a}_{i} \hat{a}^{\dagger}_{a} \hat{a}_{i}, \quad \hat{T}_2 = \tfrac{1}{4} \sum_{ijab} t^{ab}_{ij} \hat{a}^{\dagger}_{a} \hat{a}^{\dagger}_{b} \hat{a}_{j} \hat{a}_{i}.$$

3. *Evaluation of matrix elements in second quantization* (line 14)
   For this there are several possibilities, which are discussed below (sections 3.1.2–3.1.5):
   - Transformation of the operator string into a sum of normal-ordered ones by using anticommutator relations or Wick's theorem
   - Diagrams
   - Consideration of equivalence classes of permutations
4. *Simplification* (line 15/16)
   In most cases the result of step 3 contains redundant terms, i.e. terms which can be collected by exploiting index symmetries (compare 3.1.2, 3.1.5 and chapter 4).

3.1.2. Algebraic Methods for the Evaluation of Matrix Elements

In order to evaluate a matrix element of the form $\langle\Phi|\hat{A}\Phi_0\rangle$, where $\hat{A}$ is a string of annihilation and creation operators, $\hat{A}$ may be transformed into a sum of operators each of which is normal-ordered with respect to $|\Phi_0\rangle$. Then all summands which contain at least one annihilator produce zero when applied to $|\Phi_0\rangle$.

In principle this could be achieved by the direct application of the anticommutator rules (2.17)–(2.19), but this is very inefficient. Since every interchange of an annihilator with a creator produces two new terms, the number of summands grows very rapidly, while at the end only few of them yield a non-vanishing result.

One possibility to reduce the number of terms is to use Wick's theorem (see 2.2.1.3). We only have to keep the fully contracted terms, since all other terms have at least one operator on their right which yields zero when applied to $|\Phi_0\rangle$. But even then there are still too many terms. For a string of $2n$ operators there are $n!\prod_{k=1}^{n}(2k-1)$ fully contracted strings which can be produced from it. Of course many of these are zero, but there are still a lot of terms left with much redundancy among them. That means that there are terms which are in fact equal, but look different because of the naming of their indices.

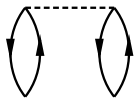
3.1.3. Diagrams

Matrix elements of operators in second quantization can also be evaluated – without explicit manipulations of creation and annihilation operators – by diagrammatic techniques. Several sorts of diagrams are used in physics and theoretical chemistry, most of which are some sort of Feynman diagrams. A formalism based on Goldstone diagrams [61] which is particularly suitable for coupled-cluster and many-body perturbation theory has been introduced by Kucharski and Bartlett [90]. Here we only want

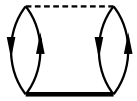
to sketch briefly the ideas of this method and give some examples. A more detailed introduction can be found in [1] or [53].

The starting point is the representation of indices (i.e. orbitals) occurring in operators or determinants by (vertical) lines where the direction of the line indicates whether it stands for a hole or a particle index. These lines can then be joined according to certain rules to form matrix elements. In the last step, the diagrams are translated into explicit formulas, including the correct coefficient and sign. This requires a rather complex set of rules, which we do not want to discuss here. Instead, we show some examples.

The matrix element $\frac{1}{2}\langle\Phi_0|\hat{H}\hat{T}_1^2\Phi_0\rangle$ corresponds to the diagram



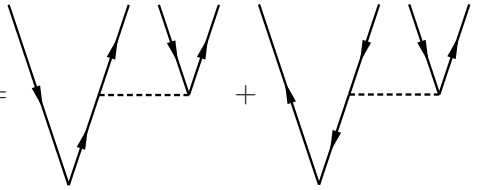
Here the dashed line ("interaction line") denotes the two-electron part $\hat{V}$ of the Hamiltonian (the one-electron part yields a vanishing matrix element in this case), while the lower ends of the "loops" correspond to the two $\hat{T}_1$ operators. The solid lines stand for hole (downward) and particle (upward) indices, respectively. The analogous diagram with $\hat{T}_1^2$ replaced by $\hat{T}_2$ would look like this:



We now get explicit expressions by assigning appropriate index labels to the hole and particle lines and writing down the corresponding quantities:

$$\begin{aligned}\tfrac{1}{2}\langle\Phi_0|\hat{H}\hat{T}_1^2\Phi_0\rangle &= \tfrac{1}{2}\sum_{ijab} v_{ab}^{ij} t_i^a t_j^b \\ \langle\Phi_0|\hat{H}\hat{T}_2\Phi_0\rangle &= \tfrac{1}{4}\sum_{ijab} v_{ab}^{ij} t_{ij}^{ab}\end{aligned}$$

Sometimes more than one diagram is needed to represent a matrix element, e.g.

$\langle\Phi_0|\hat{A}_2\hat{V}\hat{T}_1\Phi_0\rangle =$  $= \left[\sum_i t_i^A v_{Bi}^{IJ}\right]_{\mathcal{A}} + \left[\sum_a t_J^a v_{AB}^{Ia}\right]_{\mathcal{A}}.$

The operator $\hat{A}_2$ generates two-fold excited determinants from $\langle\Phi_0|$. The external indices correspond to the lines with open ends, and the subscript $\mathcal{A}$ means that the term is antisymmetrized with respect to the external indices.

3.1.4. Automatization

While the explicit equations of the standard CCSD method are well known and comparably short, the formulas become long and more complex for increasing excitation level. Therefore it would be very difficult – or even impossible – to derive these by hand. Moreover, it is desirable to have an automatized equation generation procedure if one is interested in variants or generalizations of the standard coupled-cluster methods.

Several approaches to this task exist and have been implemented by various groups. Most of them are based on some sort of diagrams, but there are also purely algebraic ones. We want to give a brief survey here. Some of the mentioned work has also been discussed by Hirata in his review [91].

An algebraic approach based on the unitary group formalism has been pursued by Li and Paldus in the derivation of their spin-adapted open-shell coupled-cluster method [65]. Hirata and Bartlett [85] as well as Olsen [84] use the formalism of spin strings and the anti-commutation rules for elementary operators in second quantization to derive coupled-cluster equations in an automatized way.

The first program using Wick's theorem was SQSYM ("second quantization symbol manipulator") by Janssen and Schaefer [92]. Later this ansatz was adopted by Jankowski and Jeziorski [93], Nooijen and Lotrich [94], Berente et al. [95] and by Hirata in the framework of the TCE [37]. The latter two implementations reduce the number of terms produced when applying Wick's theorem by erasing unnecessary terms as early as possible.

Automatic generation of diagrams has first been used in the context of perturbation theory [96–98]. One of the first diagram generation algorithms for coupled-cluster was implemented by Harris [99] using the computer algebra system Maple. The coupled-cluster implementations with arbitrary excitation level by Kállay and Surján [41] and by Lyakh, Ivanov, and Adamowicz [100] both use the type of diagrams discussed in the previous section. To make them accessible for automatic procession, Kállay and Surján represent diagrams by strings of 13 integer numbers while Lyakh et al. use matrix-like structures. A more general approach is pursued by Bochevarov and Sherrill [101], who developed an algorithm which can handle arbitrary second-quantized expressions. Within their program, diagrams are written as strings of symbols. The symbolic algebra program SMITH by Shiozaki et al. [39] also uses Goldstone diagrams, which are in this case treated as objects directly, i.e. without the introduction of an auxiliary structure. It is also rather general and can treat e.g. CC-R12 methods.

3.1.5. Exploiting Index Symmetry

3.1.5.1. Term Generation

One reason for the inefficiency of the evaluation procedure using Wick's theorem is that it only deals with strings of annihilation and creations operators, ignoring the prefactors (amplitudes and integrals). Starting from this observation M. Hanrath developed an algorithm which directly reduces the number of generated terms by taking into account the index symmetry of the prefactors . This is done by subdividing the possible index permutations into equivalence classes [102]. The contractions are given as in (2.21). That means that the resulting expressions contain Kronecker deltas which have to be resolved in a subsequent step.

3.1.5.2. Term Simplification

If the coupled-cluster equations (2.31) are evaluated using the BCH expansion, i.e. the expressions contain only connected terms, the formulas resulting from the algorithm sketched in the previous section are completely reduced. But if the unlinked form is used, or in other, more complicated cases, some (disconnected) terms remain which are redundant. In these cases a subsequent simplification of the resulting equations is necessary in order to obtain efficient working equations. This problem is discussed at length in chapter 4. Finally, we use a graphical representation of terms which has some similarities with the diagrams in 3.1.3 but is not equivalent. The main difference is that we use the graphs only for the identification of equivalent terms. The coefficients and signs are handled separately, which makes the back-translation into formulas much easier. Another difference shows up if a term contains more than one interaction. The Feynman-type diagrams respect the order of the operators, but the resulting term usually does not depend on this order. Our graphs do not distinguish terms which differ just in the order of the interactions. Finally, our approach is very general, i.e. not restricted to special term structures. The types of operators we can treat are fixed, but could easily be extended. In contrast, while the diagrams themselves are also quite flexible, their computational representations, e.g. those used in [41] and [100], are much more restrictive.

3.1.6. Examples

In line 14 of listing 3.1 we could replace the expression for CC by other expressions. For example (compare line 15), inserting `A(-i)*HN*(I+TT)`, where `I` is the identity operator, into the Fermi vacuum expectation value `FV()` yields equations for the CI method. We can also introduce new operators, like the adjoint of the cluster operators or interaction operators of higher rank (e.g. three-particle operators). In principle we can insert arbitrary sums and products of operators which can be expressed in terms

Table 3.1.: Explicit expressions for one operator product in different projections

i	Explicit expression for `FV(A(-i)*VN*T(1)*T(1))`
0	$\frac{-1}{1} \cdot [t_i^b t_j^a v_{ab}^{ij}]_{\mathcal{A}}$
1	$\frac{-2}{1} \cdot [t_I^b t_i^a v_{ab}^{Ai}]_{\mathcal{A}} + \frac{-2}{1} \cdot [t_i^A t_j^a v_{ij}^{Ia}]_{\mathcal{A}}$
2	$\frac{-1}{1} \cdot [t_I^b t_J^a v_{ab}^{AB}]_{\mathcal{A}} + \frac{-2}{1} \cdot [t_J^A t_i^a v_{Bi}^{Ia}]_{\mathcal{A}} + \frac{2}{1} \cdot [t_J^a t_i^A v_{Bi}^{Ia}]_{\mathcal{A}} + \frac{-1}{1} \cdot [t_i^B t_j^A v_{ij}^{IJ}]_{\mathcal{A}}$
3	$\frac{-2}{1} \cdot [t_J^A t_K^a v_{BC}^{Ia}]_{\mathcal{A}} + \frac{-2}{1} \cdot [t_K^B t_i^A v_{Ci}^{IJ}]_{\mathcal{A}}$
4	$\frac{-1}{1} \cdot [t_K^B t_L^A v_{CD}^{IJ}]_{\mathcal{A}}$

Table 3.2.: Explicit expressions for the Fermi vacuum expectation values of different operators (`WN` is a three-body potential and `T(-1)` is the adjoint of `T(1)`)

Operator `X`	Explicit expression for `FV(X)`
`exp-TT*WN*expTT`	$\frac{1}{36} \cdot [t_{ijk}^{abc} w_{ijk}^{abc}]_{\mathcal{A}} + \frac{1}{4} \cdot [t_i^c t_{jk}^{ab} w_{ijk}^{abc}]_{\mathcal{A}} + \frac{-1}{6} \cdot [t_i^c t_j^b t_k^a w_{ijk}^{abc}]_{\mathcal{A}}$
`HN*HN`	$[f_a^i f_a^i]_{\mathcal{A}} + \frac{1}{4} \cdot [v_{ab}^{ij} v_{ab}^{ij}]_{\mathcal{A}}$
`VN*VN*VN`	$\frac{1}{8} \cdot [v_{kl}^{ij} v_{ab}^{ij} v_{ab}^{kl}]_{\mathcal{A}} + \frac{1}{8} \cdot [v_{ab}^{ij} v_{cd}^{ij} v_{cd}^{ab}]_{\mathcal{A}} + [v_{ab}^{ij} v_{kb}^{ic} v_{ac}^{jk}]_{\mathcal{A}}$
`FN*VN*(T(1)+T(2))`	$\frac{-1}{1} \cdot [f_a^i t_j^b v_{ja}^{ib}]_{\mathcal{A}} + \frac{-1}{2} \cdot [f_a^i t_{ij}^{bc} v_{bc}^{ja}]_{\mathcal{A}} + \frac{-1}{2} \cdot [f_a^i t_{jk}^{ab} v_{jk}^{ib}]_{\mathcal{A}}$
`T(-1)*HN*T(1)`	$\frac{-1}{1} \cdot [f_j^i t_j^a t_a^i]_{\mathcal{A}} + [f_b^a t_i^a t_b^i]_{\mathcal{A}} + \frac{-1}{1} \cdot [t_j^a t_b^i v_{jb}^{ia}]_{\mathcal{A}}$

of elementary annihilation and creation operators. Some examples are given in tables 3.1 and 3.2, for longer expressions see appendix C.

3.2. Equation Solving

The coupled-cluster equations in their explicit form constitute a nonlinear (but polynomial) equation system for the amplitudes which has to be solved numerically. Different possibilities to do this are briefly discussed in 3.2.2. In any case, an iterative procedure is necessary which means that the expressions occurring in the equations have to be evaluated several times. This is the main reason why it is important to do this evaluation efficiently.

3.2.1. Preparation

Before we can start to solve the equations, the input data has to be prepared. First, we have to determine our basis functions (MOs), usually by an SCF calculation (in the multi-reference case, this would be an MCSCF or CASSCF calculation, but we focus here on the single-reference case), and the integrals constructed from them. The SCF calculation also defines the reference determinant, and with it the partitioning of the orbitals into occupied and unoccupied (virtual) ones. The output of this calculation further contains information about the symmetry properties of the system, in particular, which orbitals belong to which irreducible representation. From this data, a table is constructed in which the orbitals are ordered according to the properties occupation status, irreducible representation, and spin (α or β). Our present implementation is based on spin orbitals. We start with a set of spatial orbitals and the corresponding integrals, then we construct two spin orbitals from each spatial orbital and adapt the integrals accordingly. In addition, the two-electron integrals are antisymmetrized according to equation (2.23). Another transformation is necessary because due to the use of the normal-ordered Hamiltonian (see 2.2.1.4) not the plain one-electron integrals $\langle p|\hat{h}|q\rangle$, but the Fock matrix elements f^p_q appear in the explicit equations[a]. Finally, the integrals are arranged in different tensors, each with a fixed pattern of occupied and virtual indices.

Also the amplitudes have to be prepared, at least memory for them has to be reserved. In most single-reference calculations it is possible to set all amplitudes to zero in the beginning, while in more complicated cases a reasonable initial guess (e.g. from a CI calculation) is useful for the calculation to converge to the desired state.

3.2.2. Numerical Methods

There are several ways to solve the coupled-cluster equations numerically, which differ in their applicability (e.g. numerical stability) and efficiency (convergence behavior).

If we use the linked form of the CC equations, it is sufficient to consider the amplitude equations (2.31). If we introduce the amplitude vector $\mathbf{t}$ and set

$$A_\alpha(\mathbf{t}) := \left\langle \alpha \left| e^{-\hat{T}} \hat{H}_N e^{\hat{T}} \right| \Phi_0 \right\rangle, \tag{3.2}$$

where α stands for any excited determinant, the equations can be written in vector form as

$$\mathbf{A}(\mathbf{t}) = 0. \tag{3.3}$$

In principle this can be solved iteratively by applying the standard Newton method for vector-valued functions. But this is not very efficient, since it requires in each iteration

[a] If matrix elements for two determinants are evaluated using the Slater–Condon rules, an analogous summation – restricted to the orbitals occupied in both determinants – has to be carried out each time.

step the inversion of the Jacobian matrix (or at least the solution of a linear equation system) to calculate the correction $\Delta\mathbf{t}$ to the amplitudes (see e.g. [52]).

Another approach is to write the equation (3.3) as a fixed point equation by isolating the amplitude corresponding to the determinant which determines the projection:

$$A_\alpha(\mathbf{t}) = 0 \quad \Leftrightarrow \quad D_\alpha t_\alpha = \tilde{A}_\alpha(\mathbf{t})$$

where for $\alpha = \Phi^{a_1 \dots a_k}_{i_1 \dots i_k}$ the coefficient D_α is given as $\sum_{j=1}^{k} f^{i_j}_{i_j} - \sum_{j=1}^{k} f^{a_j}_{a_j}$. For CCSD, this is carried out explicitly in [1]. This kind of equations can be solved iteratively by choosing a starting vector $\mathbf{t}^{(0)}$ and then setting in each step $t_\alpha^{(n+1)} = \tilde{A}_\alpha(\mathbf{t}^{(n)})/D_\alpha$ until self-consistency is reached. But this would require the evaluation of the modified expressions $\tilde{A}_\alpha(\mathbf{t})$, which is rather inconvenient due to the missing amplitude t_α in some summations. Therefore, in practice one rewrites the equations again, leading to an iteration scheme $\mathbf{t}^{(n+1)} = \mathbf{t}^{(n)} + \Delta\mathbf{t}^{(n)}$ with $\Delta t_\alpha^{(n)} = A_\alpha(\mathbf{t}^{(n)})/D_\alpha$, and stops if the norm of the residual vector $\mathbf{A}$ is sufficiently small. This procedure is equivalent to a quasi-Newton method where the Jacobian matrix is approximated by keeping only the diagonal elements [52]. In our present implementation we use this method. The convergence is usually rather slow, but it can be improved by using the DIIS ("direct inversion in the iterative subspace") scheme [29, 52, 103].

3.2.3. Summary

Assuming that explicit equations have been derived, a (single-reference) CC calculation in our implementation consists of the following steps (the order of steps 3 and 4/5 is interchangeable):

1. Get reference determinant, orbital properties, and MO integrals from an SCF calculation and subsequent MO transformation.
2. Construct orbital table.
3. Factorize equations and determine sequence of contractions (orbital table is used for cost calculation).
4. Construct spin orbital integrals and Fock matrix elements.
5. Choose initial values for amplitudes and construct tensors needed for evaluation (amplitudes, integrals, residuals).
6. Calculate residuals by evaluating the contractions given by 3.
7. Determine the residual norm and check for convergence (norm less than a given threshold).

8. If the calculation is not converged: Update the amplitudes, set the residuals to zero and go back to step 6.

9. Else: Calculate the energy (and the final amplitudes, if desired).

The code of an exemplary program which does such a calculation is shown in appendix B.

4. Term Simplification

4.1. Problem Description

For this section a *term* is an object that can be written in the form (3.1), i.e. a sum over products of certain objects, where all products have the same structure. Each factor in the products can have several indices, and the summation ranges over all or some of these indices. We apply the same convention regarding the indices as in chapter 3. If we do not write the sigma sign explicitly summation over all lower case indices is assumed implicitly. In our cases, each summed index appears twice in a term while an external index is used only once.

Now, as an additional complication, the indices can have different types and each factor has a specific *index structure*. This means that the indices are divided into groups and have a certain pattern of types. In particular, the factors can have different numbers of indices. Moreover, the tensors which the factors represent can have symmetry properties with respect to certain index operations. In the following sections we will specify the kinds of objects we consider and their respective symmetries.

We consider two terms as being equivalent if their numerical evaluation yields – potentially up to sign – the same result for all possible values of the involved objects. But this definition is not very useful, we need criteria for equivalence which refer directly to the structure of the term.

Examples

Anticipating the specialization in the next section, we use terms containing (coupled-cluster) amplitudes and two-electron integrals as examples. In some cases the equivalence is rather obvious, like for the terms

$$v^{ij}_{ab} t^a_i t^b_j \text{ and } v^{ij}_{ab} t^b_j t^a_i \quad \text{or} \quad v^{ij}_{ab} t^A_i t^{ab}_{Ij} \text{ and } v^{ij}_{ab} t^A_j t^{ab}_{Ii},$$

in others it is not. Consider the three terms

$$v^{ik}_{cd} v^{jl}_{ab} t^c_k t^d_l t^{ab}_{ij}, \; v^{ik}_{ab} v^{jl}_{cd} t^c_k t^d_l t^{ab}_{ij}, \text{ and } v^{ik}_{ad} v^{jl}_{cb} t^c_k t^d_l t^{ab}_{ij}.$$

They look very similar at first glance, but it is not easy to decide whether they are really equivalent (by close inspection it can be seen that the first two are equivalent, but the third is not equivalent to the others).

In general, the following operations transform a term into an equivalent one:

T1 Changing the order of the factors

T2 Renaming summation indices

T3 Exploitation of index symmetries within one factor, e.g.

- Symmetry with respect to the exchange of index groups
- (Anti-)symmetry with respect to permutations of indices within one group

At this point it should already be clear that the combination of these different transformations can lead to a huge number of equivalent terms, which makes the identification of all redundant terms in a long equation into a (potentially) very difficult task. Thus it is very desirable to have an automatized procedure for it.

4.2. Algebraic Approach

The main idea of this approach is to define a (arbitrary) total order[a] on the set of terms. Then there is a unique smallest element in every equivalence class, since the number of equivalent terms – as large as it may be – is always finite as long as the set of possible indices is finite. The algorithm checks for each term if it can be transformed into one which is smaller with respect to this order and this process is iterated until no more changes are possible.

4.2.1. Term Representation

The factors in the terms we consider here can have the following types

- operator parameters, like the amplitudes of the cluster operators (t) or of the corresponding deexcitation operators
- one- or two-electron integrals (f^p_q, v^{pq}_{rs})
- strings of second quantized operators, usually in normal order
- permutators containing external (i.e. not summed) indices.

[a] By this we mean a relation that is analogous to the familiar $<$ relation for real numbers, in the sense that it is transitive (i.e. if $a < b$ and $b < c$ then also $a < c$ holds) and for every two elements a, b one of the relations $a < b$, $b < a$ or $a = b$ holds. A total order is sometimes also called a linear order, since the elements of a set with a total order can be arranged in a linear sequence according to relations between them.

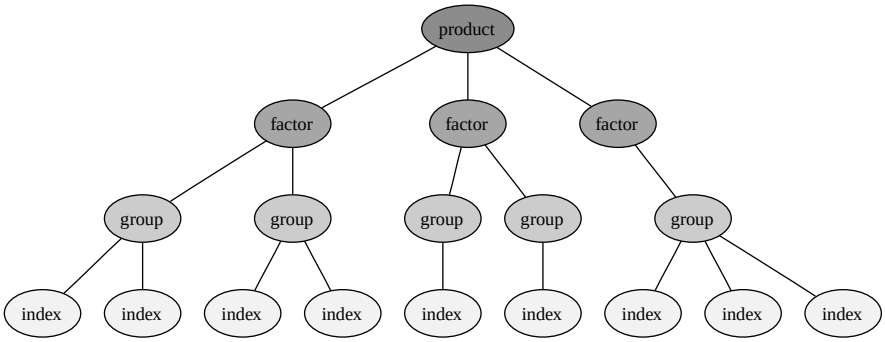


Figure 4.1.: Structure visualization for a general term

Permutators (which could also be called partial antisymmetrizators) are used to collect terms with a similar structure in an equation (the coupled-cluster amplitude equations are antisymmetric with respect to permutations of external indices of the same type). Consider for example the term $t_i^A v_{Bi}^{IJ}$. It is antisymmetric with respect to I and J (because the integral v is), but not with respect to A and B. Its antisymmetrized form is

$$\left[t_i^A v_{Bi}^{IJ}\right]_{\mathcal{A}} = \hat{P}_-(A|B) t_i^A v_{Bi}^{IJ} = t_i^A v_{Bi}^{IJ} - t_i^B v_{Ai}^{IJ}.$$

The permutator $\hat{P}_-(A|B)$ is just a more explicit way of writing the antisymmetrization. A general permutator is defined as follows:

$$\hat{P}_-(I_{1,1} \dots I_{1,n_1} | I_{2,1} \dots I_{2,n_2} | \dots | I_{m,1} \dots I_{m,n_m}) = \sum_{[\sigma] \in Q} \mathrm{sgn}(\sigma) \sigma \tag{4.1}$$

where Q is the quotient of the Symmetric Group S_n ($n = n_1 + \ldots + n_m$) by the direct product of subgroups $S_{n_1} \times \ldots \times S_{n_m}$ and $\mathrm{sgn}(\sigma)$ is the parity of the permutation $\sigma \in S_n$. This means we sum over all permutations of $I_{1,1}, \ldots, I_{m,n_m}$ which do not contain any permutation within one group, multiplied by the appropriate sign.

$\hat{P}_-$ does not depend on the order of the groups or the arguments within one group, i.e. they can be permuted arbitrarily. Two different permutators commute if the sets of indices which they contain are disjoint (otherwise they do not commute in general). In our applications this is always the case since in every term there are at most two permutators, one with particle indices and one with hole indices.

For the naming of indices we employ the same conventions as in chapter 2: p, q, r, s are general indices, while hole indices are denoted by $i, j, \ldots$ and particle indices by $a, b, \ldots$. The same applies for upper case letters.

The general structure of a term can be represented by a tree as shown in Figure 4.1, an example is given in Figure 4.2.

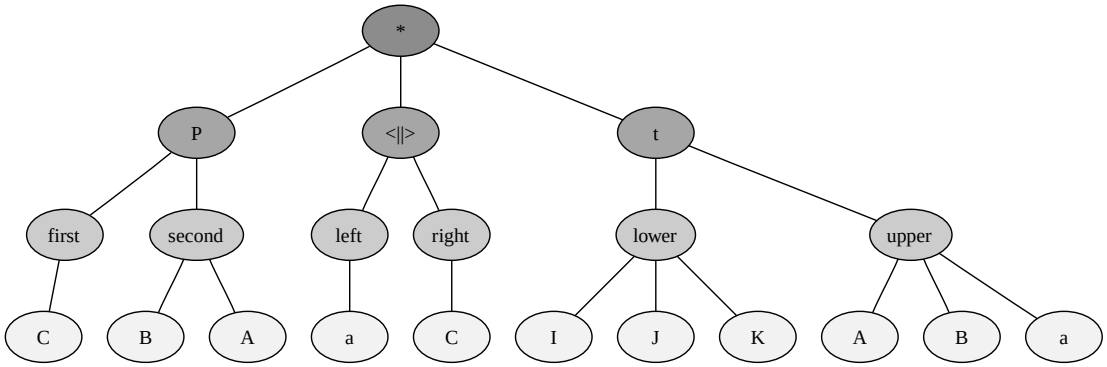


Figure 4.2.: Structure of the term $\hat{P}_{-}(C|BA)\sum_{a} f^{a}_{\overline{C}} t^{ABa}_{IJK}$. In the visualization, the same symbol is used for one- and two-electron integrals.

Now we can specify the transformations summarized under point (T3) in the previous section:

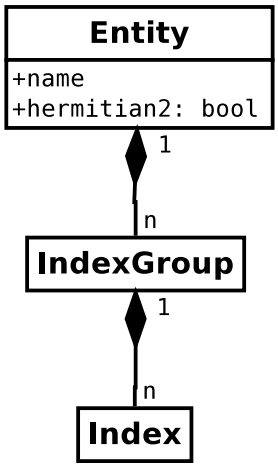
(a) Integrals are symmetric (hermitian) with respect to the interchange of bra and ket (upper and lower) indices.

(b) Amplitudes and two-electron integrals are antisymmetric with respect to permutations within one index group.

(c) Operators within a normal-ordered string anticommute.

(d) Indices within a permutator may be exchanged according to the symmetry properties described above.

The general structure of the factors can be depicted (in UML style, although this does not correspond to actual classes in the program) as follows:



That means, each entity consists of one or more index groups within which we have antisymmetry with respect to index permutations. The parameter `hermitian2` indicates the symmetry in the case that there are two groups (factors with more than two groups do not occur here). For example, integrals consist of two groups and are symmetric with respect to the interchange of them. An amplitude also has two groups, but there is no symmetry between them. A normal-ordered string of operators is just one group (the operators are identified with their indices).

The ambiguities that are due to symmetry properties can be resolved by defining a standard ordering of indices and terms, which will be explained in detail later (see 4.2.2). Therefore terms which only differ in this way are considered equal and the canonical representation is the one which is smallest with respect to the ordering.

But there is one more possibility, namely the renaming of indices. Here we have to distinguish between summed and external indices:

- Summation indices can be renamed arbitrarily as long as they are distinguishable.
- External indices may only be renamed if the term contains a permutator. Only indices already present within the term may be used and the renaming has to be compatible with the structure of the permutator.

Terms differing only by the naming of indices are called equivalent. (It is easy to see that this defines an equivalence relation in the mathematical sense.)

Examples

The terms $\sum_{ijab} v^{ij}_{ab}\, t^a_i\, t^{bC}_{jK}$ and $\sum_{ijab} t^{Ca}_{Ki}\, t^b_j v^{ij}_{ab}$ look quite different at first glance, but they can be transformed into each other through the following steps:

$$\sum_{ijab} v^{ij}_{ab}\, t^a_i\, t^{bC}_{jK} \overset{\text{T1}}{=} \sum_{ijab} t^{bC}_{jK}\, t^a_i v^{ij}_{ab} \overset{\text{T3(b)}}{=} -\sum_{ijab} t^{bC}_{Kj}\, t^a_i v^{ij}_{ab} \overset{\text{T3(b)}}{=} \sum_{ijab} t^{Cb}_{Kj}\, t^a_i v^{ij}_{ab}$$
$$\overset{i\leftrightarrow j}{=} \sum_{ijab} t^{Cb}_{Ki}\, t^a_j v^{ji}_{ab} \overset{a\leftrightarrow b}{=} \sum_{ijab} t^{Ca}_{Ki}\, t^b_j v^{ji}_{ba} \overset{\text{T3(b)}}{=} \sum_{ijab} t^{Ca}_{Ki}\, t^b_j v^{ij}_{ab}$$

Another example illustrates the number of equivalent terms possible, we have:

$$\sum_{ijkabc} t^a_i\, t^b_j\, t^c_k v^{jk}_{ab} \{\hat{a}^\dagger_c \hat{a}_i\} = -\sum_{ijkabc} t^a_i\, t^b_j\, t^c_k v^{ik}_{ab} \{\hat{a}^\dagger_c \hat{a}_j\}$$
$$= \sum_{ijkabc} t^a_i\, t^b_j\, t^c_k v^{ik}_{bc} \{\hat{a}^\dagger_a \hat{a}_j\} = -\sum_{ijkabc} t^a_i\, t^b_j\, t^c_k v^{ij}_{bc} \{\hat{a}^\dagger_a \hat{a}_k\}$$
$$= \sum_{ijkabc} t^a_i\, t^b_j\, t^c_k v^{ij}_{ac} \{\hat{a}^\dagger_b \hat{a}_k\} = -\sum_{ijkabc} t^a_i\, t^b_j\, t^c_k v^{jk}_{ac} \{\hat{a}^\dagger_b \hat{a}_i\}$$

In contrast, for example the term $\sum_{ijkabc} t_i^a\, t_j^b\, t_k^c v_{ab}^{ij} \{\hat{a}_c^\dagger \hat{a}_k\}$, although looking similar, is not equivalent to the others.

4.2.2. Order Relation

4.2.2.1. Definition

To define a total order on the set of terms we first have to order their building blocks, starting with the smallest elements, namely the indices. For them the following rules apply:

1. External indices are smaller than summed indices.
2. Hole indices are smaller than particle indices and these are smaller than general indices.
3. Within one type the indices are ordered alphabetically.

From now on we assume the indices within one group to be ordered according to these rules. For symmetric factors (integrals) the groups are ordered lexicographically.

Now we come to the ordering of the factors for which the following criteria are employed:

1. The different kinds of factors get a priority, e.g. amplitudes are smaller than integrals and operators.
2. Among factors of the same kind, those with more indices come first.
3. If the total number of indices is equal, the number of external indices is taken into account.
4. If this is also equal, the further procedure depends on the nature of the factor:

   Operator Strings In this case no special considerations are necessary, the strings are ordered lexicographically.

   Amplitudes The problem when comparing two amplitudes is that on each amplitude there are two groups of indices (upper and lower) and one has to decide which one is to be considered first. This is resolved in the following way: If for one argument the upper indices are smaller than the lower ones, the upper indices are compared first, only in the case that for both arguments the lower indices are smaller, these are compared first. This procedure is illustrated in figure 4.3.

Integrals In principle the same procedure as for amplitudes is used, with upper/lower indices replaced by left/right ones, respectively. But since in this case the two index groups can be exchanged and we assume them to be ordered, the second case never applies.

Permutators The order of the permutators is not relevant, so we define it arbitrarily. If there are two of them in one term, then one contains only particle indices and the other only hole indices. If their structure is the same, then the one with hole indices is defined to be smaller.

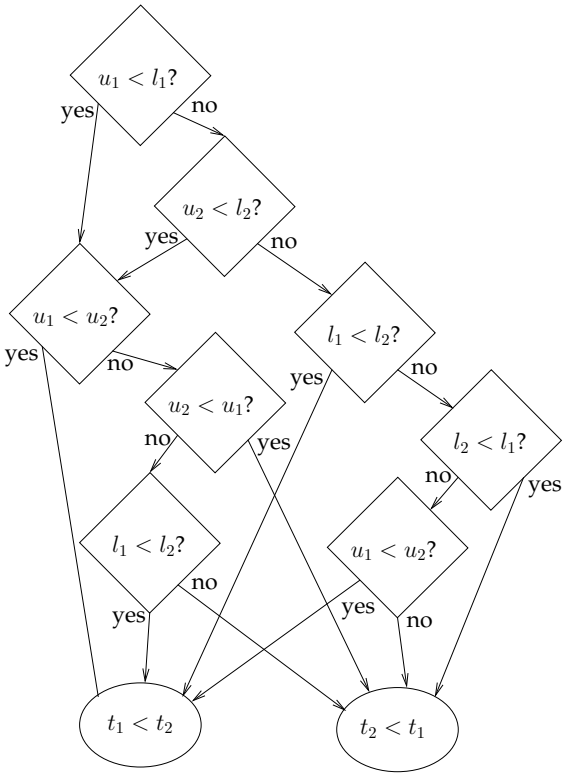


Figure 4.3.: Procedure to determine whether for two different amplitudes $t_1 = (u_1, l_1)$ and $t_2 = (u_2, l_2)$ the relation $t_1 < t_2$ or $t_2 < t_1$ holds.

Terms are now ordered first by their type, where type includes the number and structure of the factors as well as the number and distribution of external indices. This has to be done for the sake of completeness, although terms of different types can never be equivalent. Moreover, the order is used when equal terms are collected in the end.

The interesting point for our simplification is the order among terms of the same type. This is defined by lexicographical comparison of the factors according to the rules specified under point 4 above.

4.2.2.2. Transitivity

For an order relation to be well defined it has to fulfill the transitivity rule, i.e. if a is smaller than b and b is smaller than c, also a has to be smaller than c. For our definitions this is more or less obvious, except for the amplitude order.

Before starting the formal proof we want to comment briefly on the definition and illustrate it by an example. Of course things would be much easier if we decided to take into account primarily one index group (either upper or lower) and look at the other only if the first groups are equal. But this would introduce an unwanted arbitrariness. Consider for example the three amplitudes $t_1 = t_{ij}^{AB}, t_2 = t_{Ik}^{Da}$ and $t_3 = t_{Jl}^{Cb}$. Ordering by upper or lower indices only, we would get $t_1 < t_3 < t_2$ or $t_2 < t_3 < t_1$, respectively, while our order yields $t_1 < t_2 < t_3$. To see this, look at t_1 and t_2 first. We have to compare the upper indices, since there are two external indices for t_1. On the other hand, t_2 has only one external index and therefore we have $t_1 < t_2$. Next we consider t_2 and t_3. There is one external index in every group, so we have to look at the lower indices, and since $I < J$ it follows that $t_2 < t_3$.

The proof that our order is indeed transitive cannot be built on formal arguments alone, it uses special properties of the coupled cluster amplitudes (in particular the fact that the lower indices are always holes and the upper ones are particles).

For this we write the amplitudes as pairs consisting of the groups of upper and lower indices: $t_i = (u_i, l_i)$. We assume now that we have three amplitudes t_1, t_2, t_3 of the same degree and with the same number of external indices (otherwise transitivity is obvious) so that $t_1 < t_2$ and $t_2 < t_3$. We want to show $t_1 < t_3$. As a first step we observe that this is clearly fulfilled if for both comparisons in the assumption the same indices (upper or lower) are used, because then the same indices are also used for the comparison of t_1 and t_3. This is the case if either $l_i < u_i$ for all i (then always the lower indices are used) or if $u_i < l_i$ for at least two amplitudes (then always the upper indices are used). So the only interesting case is that we have $u_i < l_i$ for exactly one amplitude t_i. But this implies that u_i contains more external indices than l_i, since in general hole indices are smaller than particle indices. Since we assume that the total number of external indices is the same for all amplitudes, it follows that u_i also contains more external indices than u_j for $i \neq j$. Therefore we have $t_i < t_j$ for $i \neq j$, which for $i \neq 1$ is a contradiction to our assumptions and for $i = 1$ proves the claim.

4.2.3. Simplification Algorithm

A rough sketch of the procedure is given in algorithm 1, now we want to explain its steps in some detail (see also figure 4.4).

First we have to distinguish two cases: If the term under consideration contains one or more permutators, the following applies to all indices. Otherwise, the external indices are fixed and only summed indices may be changed.

Algorithm 1: Term simplification

```
Input: Term
Output: Term in canonical form
foreach index group do
    mark group as blocked;
    foreach index do
        if index not minimal then
            check if index can be replaced by smaller one;
            if possible then
                rename indices;
            end
        end
    end
end
```

Now we consider a product with several factors, each of which is one of the entities described above. For the first factor, the indices are set to the smallest values possible, and if the same indices occur in other factors as well, these are renamed accordingly. Then the indices of the other factors are optimized successively. More precisely, not the factors themselves are considered but the index groups. For each index group the procedure is as follows: For the first index it is checked whether it has the smallest value within its type. If this is not the case, it is tried out if it can be replaced by this smallest index. In most cases this index has been used before in other groups, then the exchange is only possible if it can be compensated within the groups that were already optimized (by exploiting the antisymmetry properties of amplitudes, integrals and operator strings or by interchanging amplitudes of the same type). If this is not the case the test is repeated with next larger index and this is iterated until a valid index is found or the original index is reached. For the following indices the iteration is started with the successor of the last used index of the same type.

To one case we have to pay special attention, namely to the permutators. Here we have not only the antisymmetry property within the index groups, but we may also interchange different groups. For our purpose now we can restrict to groups of the same size. The interchange is easy if both groups have only one element, otherwise several pairs of indices have to be changed. For each of them the same test as before has to be performed.

Example

We want to demonstrate how the algorithm works with a simple example, namely the term $\sum_{iklcd} t_{kl}^{cd}\, t_i^A v_{kl}^{Id}$, which is equivalent to $\sum_{ijkab} t_{ij}^{ab}\, t_k^A v_{ij}^{Ia}$. The successive index changes are shown in the following table:

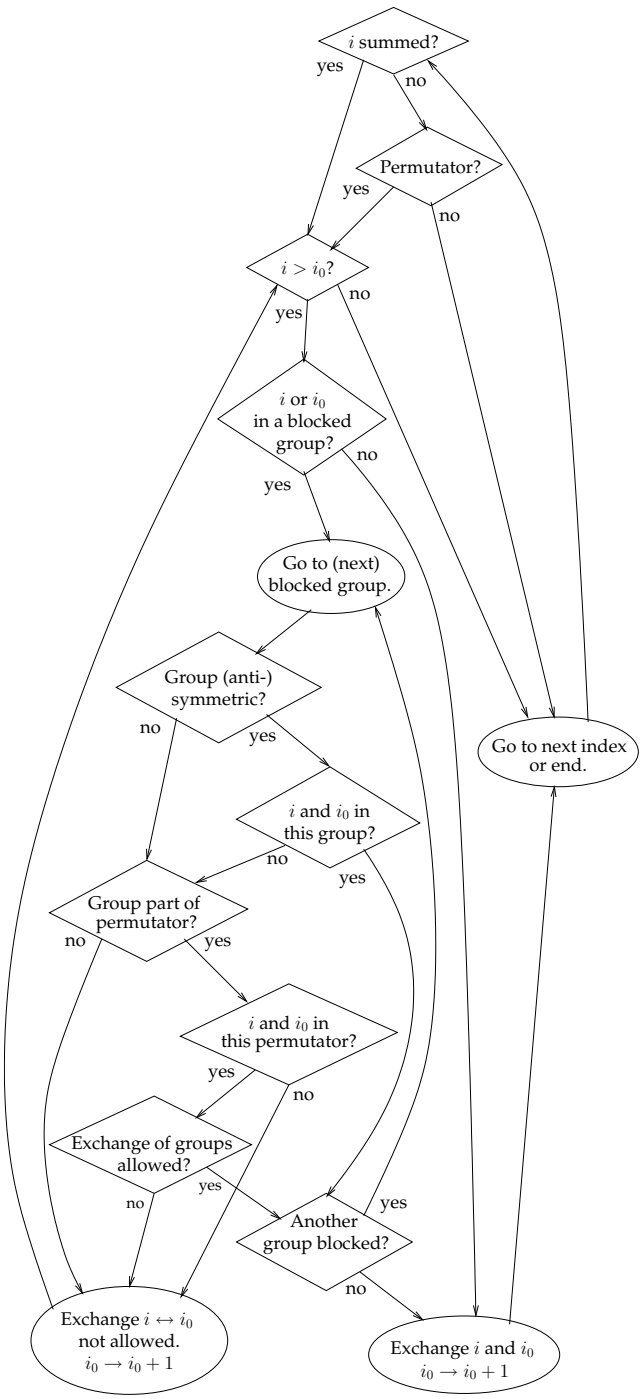


Figure 4.4.: Illustration of the optimization procedure for one index group. Here i denotes an arbitrary index, i_0 the optimal index at the present position. The case that an index exchange can be compensated by interchanging amplitudes is not shown.

step	group	index	minimal?	operation	result
1	kl	k	no	$k \leftrightarrow i$	$\sum_{iklcd} t_{il}^{cd}\, t_k^A v_{\overline{il}}^{Id}$
2		l	no	$l \leftrightarrow j$	$\sum_{ijkcd} t_{ij}^{cd}\, t_k^A v_{\overline{ij}}^{Id}$
3	cd	c	no	$c \leftrightarrow a$	$\sum_{ijkad} t_{ij}^{ad}\, t_k^A v_{\overline{ij}}^{Id}$
4		d	no	$d \leftrightarrow b$	$\sum_{ijkab} t_{ij}^{ab}\, t_k^A v_{\overline{ij}}^{Ib}$
5	k	k	yes		$\sum_{ijkab} t_{ij}^{ab}\, t_k^A v_{\overline{ij}}^{Ib}$
6	Ib	b	no	$b \leftrightarrow a$	$\sum_{ijkab} t_{ij}^{ab}\, t_k^A v_{\overline{ij}}^{Ia}$
7	ij	i	yes		$\sum_{ijkab} t_{ij}^{ab}\, t_k^A v_{\overline{ij}}^{Ia}$
8		i	yes		$\sum_{ijkab} t_{ij}^{ab}\, t_k^A v_{\overline{ij}}^{Ia}$

External indices are not listed since there is no permutator present. In the fifth step, the index k is not the absolute minimum, but it can not be exchanged with i or j because they are blocked.

4.2.4. Implementation

The representation of terms in the program closely resembles the illustration given in figure 4.1. An expression, i.e. a linear combination of terms, corresponds to a tree, and each term is a subtree. There is a general class `Node` and derived from it are special classes representing the different objects within a term, the arithmetic operations (addition and multiplication), and rational numbers (which occur as prefactors of terms). Each node contains a pointer to the node above it ("parent"), which is zero if the node is the root node, and a list of its "children". So it is possible to move within the tree in different directions: upward, downward, and horizontally among the children of one parent node. While some functions, which affect the tree as a whole or are independent of the node type, are only implemented in the base class, for many operations we make use of virtual functions. That means that the same operation can have different effects on different types of nodes. Many functions are called recursively for the children, until they reach the point where they take effect. So we do not need to know in advance which type of node is at which position in the tree.

4.2.5. Discussion

We want to discuss briefly under which conditions this algorithm yields the same canonical form for all equivalent terms. It is rather clear that each step transforms

a given term into an equivalent one which is smaller with respect to the order defined above. So for the algorithm to work, the following statement has to be true: If a term is not the minimal element in its equivalence class, then there is always an allowed index renaming which makes it smaller.

There are, however, cases where this condition is not fulfilled, as the following example shows. Consider the term

$$t_i^a \, t_j^b v_{kb}^{ic} v_{ac}^{jk} \tag{I}$$

which is (internally) ordered since with respect to the lexicographical order described in 4.2.2 the first integral is smaller than the second. This term is not minimal, since it is equivalent to

$$t_i^a \, t_j^b v_{bc}^{ik} v_{ka}^{jc} \tag{II}$$

which is obviously smaller (since k as a hole index is smaller than the particle index c). To transform (I) into (II) we have to interchange i with j and a with b, which yields $t_j^b \, t_i^a v_{ka}^{jc} v_{bc}^{ik}$, and then order the factors. But the algorithm in its present form would not do this, since it does not allow for the interchange of integrals in the index-renaming step. The interchange of b in v_{kb}^{ic} (which is the first non-minimal index in (I)) with a is not allowed, since it requires also the interchange of i and j and this would make the first group in the integral – which has already been optimized is is therefore blocked – larger.

But if we restrict our considerations to standard coupled cluster equations, there are certain constraints which reduce the number of possible index constellations drastically:

C1 There is only one integral, and each summation index has to appear in this integral.

C2 There are at most four summation indices, two of each type.

C3 If there are two summation indices of the same type, they belong to the same group in the integral.

C4 All equivalent terms which are internally ordered have the same index structure.

Under these conditions we can show that the algorithm yields the desired result. Assume that an index p in a given position is not optimal, and denote the optimal index by p_0. Because of C4, p and p_0 have the same type. If p_0 has not appeared in the term before, it is clear that p can be replaced by p_0. The case the p_0 has been used before can not occur, since either the current position is on an amplitude, then p_0 can not be the optimal index in this position if it already present on another amplitude, or the current position is on an integral, and p_0 also has to occur on the integral (C1) in the same group as p (C3). In the second case there are two possibilities: If p stands before

p_0, the term is not internally ordered. If p_0 stands before p, p_0 can not be the optimal index in the position of p. So in any case we get a contradiction.

But we are also interested in more general cases, for example in expressions like the wave function variance[b] where two integrals per term occur. Considering the above example, one could think of modifications solving this problem, e.g. a different ordering relying primarily on the index structure and only after that on a lexicographical ordering. But there is no guarantee that this would help in all cases, e.g. in those where deexcitation operators are present, as for example in gradient calculations. It seems that it is very difficult – or maybe even impossible – to construct an algorithm which transforms such general terms into a canonical representative. Most implementations mentioned in 3.1.4 as well as an algorithm similar to ours, which was developed by Wladyslawski and Nooijen [105], are also restricted to terms with one integral. In the formula generation part of the TCE, canonicalization is used as a first step, while for problematic cases a more elaborate comparison procedure is used [37].

As we wanted to have a program which is as generally applicable as possible, we decided to take a completely different approach.

4.3. Graph-Based Approach

The key observation from which we start here is that the term equivalence can be seen as a topological problem. We have seen in the previous section that the main difficulty in identifying equivalent terms is the arbitrariness in the summation index labels, since it is very hard – sometimes even impossible – to find a canonical form for them. Therefore it is natural to ask what the relevant information carried by these index labels is, and it turns out that it can be reduced to two points:

- Each index has a type, in our case particle or hole.
- Summed indices are distinguished by the fact that they appear twice in a term, in different factors, and so each summed index defines a connection between two factors.

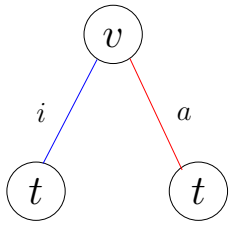
The terms we consider here are slightly different from those in the previous section. We concentrate on the simplification of the final equations (as delivered by our contraction routine described in 3.1.5), therefore the terms do not contain operators any more. We also do not take into account the permutators explicitly. We can assume here that the external indices are always in the same fixed order. Under this condition the permutators are not relevant for the equivalence problem, as will become clear later. So we are left with two types of factors, namely amplitudes and integrals.

[b] This is defined as $\langle\Psi|\hat{H}^2|\Psi\rangle - \langle\Psi|\hat{H}|\Psi\rangle^2$ and can be used as a criterion for the quality of an approximate wave function [104].

4.3.1. Representation of Terms as Graphs

4.3.1.1. General Considerations

In mathematics, a (undirected) graph is defined as a pair $(\mathbb{V}, \mathbb{E})$ where $\mathbb{V}$ is a set of vertices (or nodes) and $\mathbb{E}$ a set of edges (or connections). Each edge is given by a pair of vertices. The second observation above suggests to represent terms as graphs where (summed) indices correspond to edges, while it is less clear what the nodes of this graph should be. At first glance, as noted before, summed indices connect factors. Then the graph corresponding to the term $v^{Ia}_{Bi} t^A_i t^a_J$ would look like this:



But each factor has two distinct groups of indices, coming from annihilation and creation operators, respectively. While for amplitudes (at least those of the usual cluster operator) the group an index belongs to is determined by its type, this is not the case for integrals. On the other hand, indices within one group are exchangeable due to the antisymmetry properties of the corresponding tensors, hence it does not make sense to distinguish between them in the representation of a term. Therefore we decided to take index groups as nodes (figure 4.5(a)). Finally, the external indices can not be ignored completely. They also form two groups, so we need two additional nodes (figure 4.5(b)). Since we assume the terms to be antisymmetric with respect to the exchange of external indices of the same type – here the permutators are implicitly present – these nodes have the same properties as the other ones. To sum up: To represent the kind of terms we are considering here, we need three types of nodes, namely amplitude index groups, integral index groups, and external index groups, and two types of edges, corresponding to particle and hole indices, respectively. Each index is represented by one edge, the distinction between summed and external indices is given by the types of nodes the edge connects. This classification makes our problem somewhat different from those of classical graph theory, where all vertices and edges are in principle equal. Therefore we can not apply one of the known standard algorithms to test the equivalence of graphs.

For the figures in this section showing graphs the following conventions apply: The factors are depicted as rectangles (amplitudes) or ovals (integrals). The actual nodes (index groups) are circles connected to the factors they belong to by black lines. Index groups not connected to a factor are external. The number of the index group is written in the circle. The edges of the graph are drawn as blue or red lines for hole and particle indices, respectively.

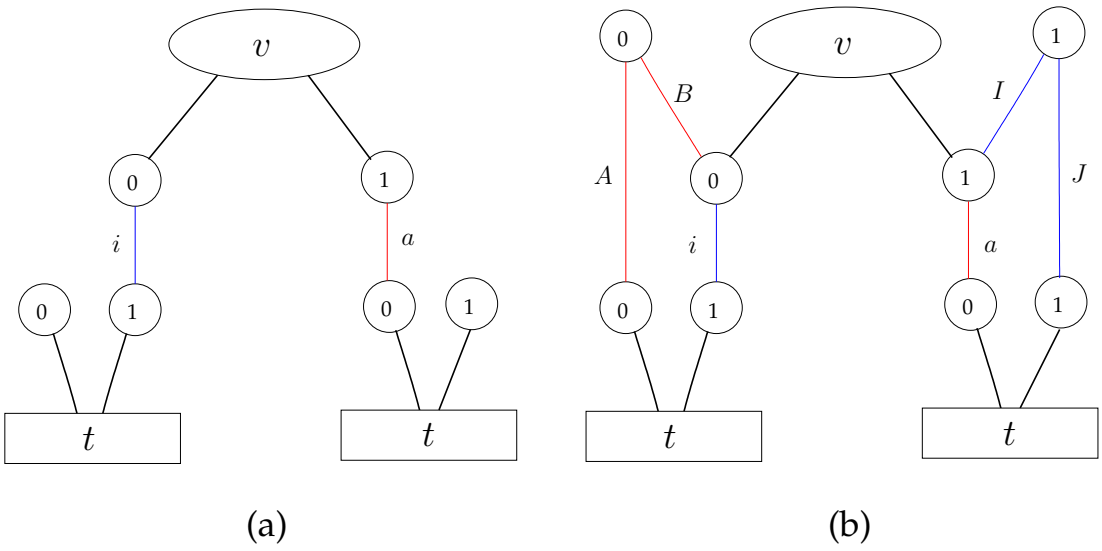


Figure 4.5.: Construction of the graph for the term $v_{Bi}^{Ia} t_i^A t_J^a$.

4.3.1.2. Realization in the Program

There are several different data structures by which a graph can be represented:

1. A list containing the nodes and for each node a list of nodes with which it is connected (or, equivalently, a list of the edges ending in this node)

2. The set of edges (for each edge the two end nodes are given)

3. An ordered list of nodes and a connectivity matrix where the entry with indices i, j is 1 if node i is connected to node j and 0 otherwise

For our purposes the second possibility is the most appropriate. So for us a graph is basically a set of connections (edges), where each connection contains the following data: start node, end node, and type (particle or hole). Although our graphs are not directed, we have to (formally) distinguish start and end node. The roles are determined by ordering the nodes. A connection always runs from the smaller to the larger node. The nodes, called `FactorSockets` here, contain the following information:

- Type of the factor, this can be "hermitian tensor" (e.g. integral), "non-hermitian tensor" (e.g. amplitude), or "external"

- Identifier (name) of the factor, e.g. t for an amplitude

- Structure of the factor, i.e. number and sizes of index groups

- Number of the factor, i.e. position among all factors of the same type and structure
- Number of the index group within the factor
- Position of the index within the group

Since in several places during the graph comparison algorithm described below only part of this data is needed, we did not put it all into one class. Instead, the information is stored in a hierarchy of different classes, which are shown in figure 4.6. The last point is not part of the node specification used for the comparison of graphs, but it is necessary to identify connections uniquely and to determine the sign of the term represented by the graph. Figure 4.7 shows the graph from figure 4.5 with its factor labels. These are to be read as follows: First, the `FactorKey` is given (in the inner parentheses). The first number codes the `FactorType` (0: `HermitianTensor`, 1: `NonHermitianTensor`), then the structure is shown (each number within the square brackets is the size of an index group), and finally the name of the factor. In addition to the `FactorKey`, the number `nth`, which is part of `FactorKey2`, is printed to distinguish between different factors with the same key (in the example, there are two t amplitudes with the same structure).

As another example, we consider the terms $v^{ij}_{ab} t^A_i t^{ab}_{Ij}$ and $v^{ij}_{ab} t^A_j t^{ab}_{Ii}$, which are quite obviously equivalent. The corresponding graphs are shown in Figure 4.8. To illustrate the correspondence between terms and graphs, we have added labels for the factors and connections again. If we disregard these labels, the two graphs look exactly the same. This is because the way the graph is printed does not reflect directly how it was defined. The algorithm which does the graph layout[c] [107] already yields a sort of canonicalization.

4.3.2. Graph Comparison Algorithm

4.3.2.1. Overview

We start from an expression as delivered by the evaluation procedure described in 3.1.5. The simplification of such an expression with our graph-based algorithm in principle consists of the following steps, which will be explained in more detail below:

1. Convert each term of the expression into a graph and store it together with the corresponding coefficient.
2. Determine fingerprints of all graphs.
3. Test graphs with the same fingerprint for equivalence and add coefficients of equivalent graphs.
4. Convert the reduced set of graphs and coefficients into an expression again.

[c] We use the tool dot which is part of the Graphviz package [106].

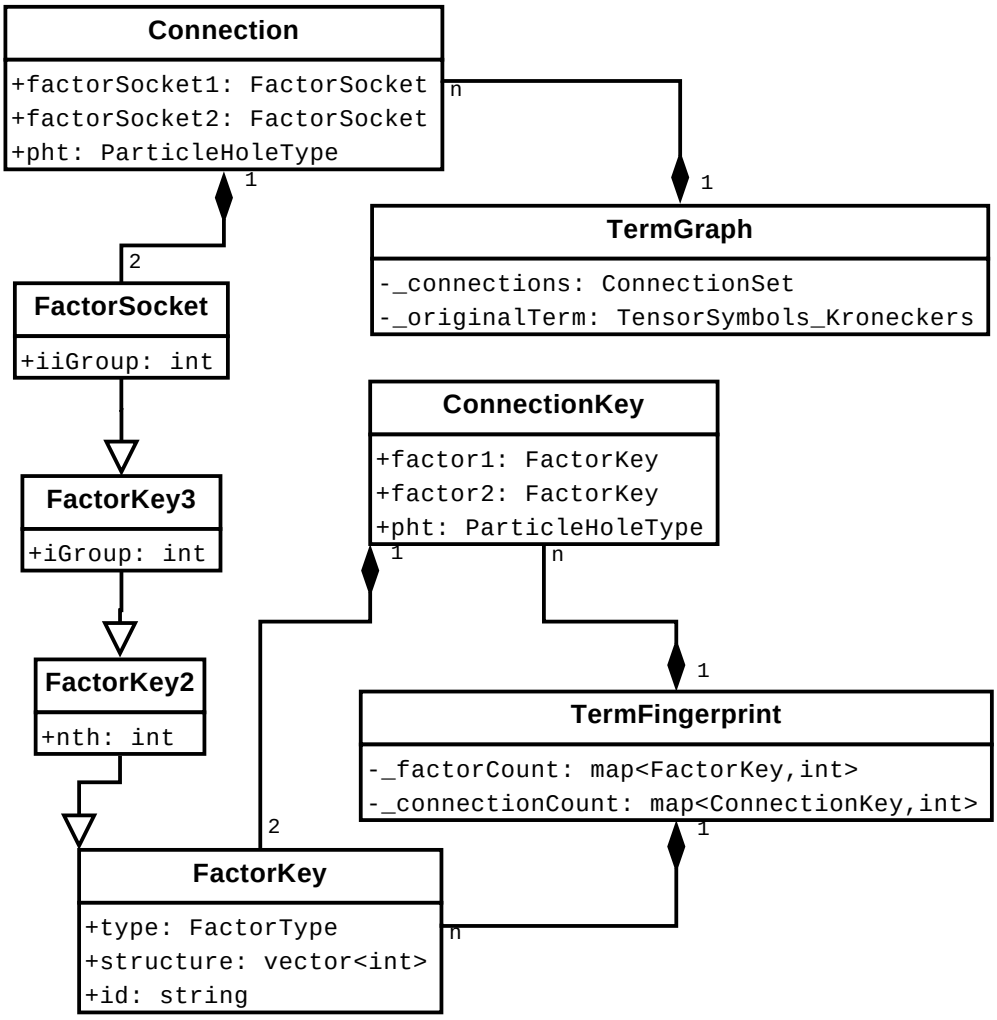


Figure 4.6.: UML diagram for graph components

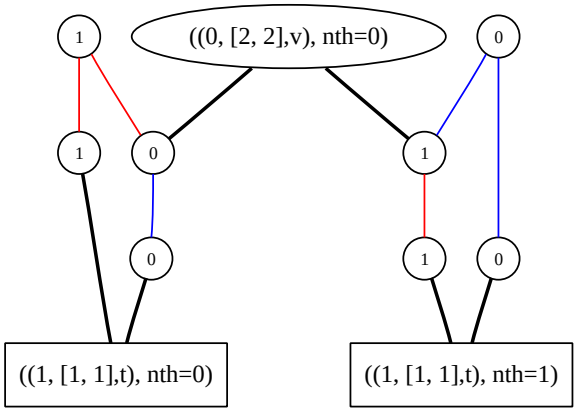


Figure 4.7.: Example graph including factor labels

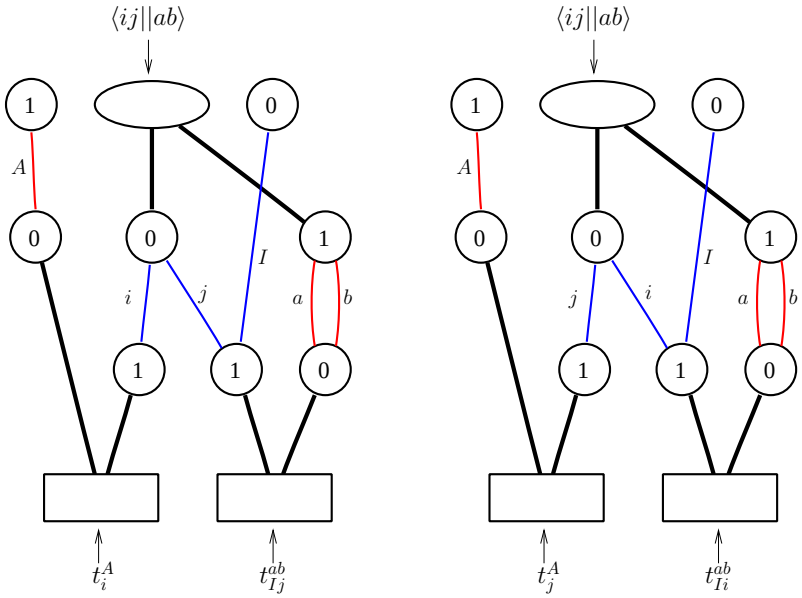


Figure 4.8.: Graphs for two equivalent terms with index labels

The second step is necessary since the explicit comparison of graphs (see 4.3.2.4) is rather time-consuming. The pre-selection by fingerprints avoids explicit comparisons of graphs which are easily seen to be non-equivalent.

In practice, for efficiency reasons all steps except the last are carried out termwise. So, after the fingerprint of a term has been determined, it is first checked if this fingerprint has occurred before. If there has been no term with the same fingerprint so far, the term is just stored, together with this fingerprint and the corresponding coefficient. Otherwise, it is checked for each term with the same fingerprint if it is equivalent to the current one in a recursive procedure (see 4.3.2.4). If an equivalent term is found, the coefficient of the current term – multiplied by -1 where necessary – is added to the stored coefficient. Otherwise, the term is stored as before. In the end we have a collection of inequivalent terms with corresponding coefficients, which can then be converted to a formula again. The whole procedure is summarized in Algorithm 2.

Algorithm 2: Graph-based simplification

```
Input: Expression
foreach Term do
    construct Graph g;
    determine Fingerprint;
    if new Fingerprint then
        store Graph and Coefficient to GraphList
    else
        foreach Graph h in GraphList do
            if recursiveTest(g,h,start_pos) then
                determine sign;
                add coefficient times sign to stored coefficient;
            end
        end
    end
end
```

4.3.2.2. Definition of Equivalence

First we want to discuss briefly how the concept of term equivalence (see 4.1) can be transferred to our graphs. Of course we want the result to be the same in the end, therefore we start with the following definition:

Definition 4.1. Two graphs are called *equivalent* if they represent equivalent terms.

But this is not very useful, we need criteria to check graphs for equivalence, similar to the term transformations listed in section 4.1. The index renaming (T2) does not

appear here. Still, we have two different comparison methods for graphs reflecting different kinds of term transformations, but they are grouped differently: Instead of distinguishing permutations of the factors and operations on one factor, we consider the index groups as primary units (since they are the nodes of our graphs).

First, we need a notion for graphs which are "nearly equal", but not identical.

Definition 4.2. Two Graphs G_1 and G_2 are called *strictly equivalent* if there exists a bijective mapping from the set of connections of G_1 to the set of connections of G_2 which maps each connection to an equivalent one.
Connections are equivalent if they have the same type and their start and end nodes are equal if the index position is disregarded.

That means strictly equivalent graphs are identical up to the order of the indices within the groups and the order of the connections. Since the order of the connections is only a matter of the internal representation of a graph, this property basically reflects the antisymmetry of the corresponding tensors. In particular it implies equivalence. In practice, when checking for strict equivalence it is sufficient to map the connections representing summed indices, if the total number of connections has been compared before, which is sensible anyway for efficiency reasons. Terms which only differ in the naming of their summation indices (operation T2) or the order of indices within the groups lead to strictly equivalent graphs by definition. Since the positions of factors and groups are used to identify nodes, the other operations listed in section 4.1 can also be seen in the graph representation. Both are realized as permutations of nodes.

Definition 4.3. A permutation of the nodes of a graph is *admissible* if it interchanges the index groups of a factor of hermitian tensor type ("Bra-Ket permutation") or permutes the positions of factors with the same `FactorKey` ("factor permutation").

A general admissible permutation contains both kinds of operations, but in practice we apply them separately. Using this definition we can state the following

Criterion for Equivalence:
Two Graphs G_1 and G_2 are equivalent iff there is an admissible permutation of the nodes of G_1 such that the resulting graph is strictly equivalent to G_2.

4.3.2.3. Fingerprints

For the construction of the fingerprint each node is reduced to its `FactorKey` containing the basic data of the factor (type and structure) as shown in Figure 4.6. Similarly, a `ConnectionKey` is defined for each edge. It consists of the `FactorKeys` of start and end node and the type of the connection. The fingerprint now contains a list in which each `FactorKey` appearing in the corresponding graph is stored together with

a number indicating how often it appears (`factorCount`), and a similar list for the `ConnectionKeys` (`connectionCount`). These classes are also shown in figure 4.6. It is obvious that graphs with different fingerprints can not be equivalent.

4.3.2.4. Explicit Comparison

Graphs with the same fingerprint are compared by a recursive algorithm. Since this is a central part of the whole simplification procedure, we show the code for this function explicitly in Listing 4.1. It is a member function of the class `DistinctTermGraphs` which consists essentially of a list of `TermGraphs` together with their corresponding coefficients. The graphs in the list represent different equivalence classes with the same fingerprint.

The recursion parameter (`map<FactorKey, int>::const_iterator i`) is a position in the `factorCount` of the corresponding fingerprint. If this is maximal, i.e. there are no entries left, the recursion base is reached (line 3). Then it is checked for every entry in the list (`iterator j`, line 5) whether the graphs are strictly equivalent. If a matching graph is found, the coefficients of the current graph (multiplied by the appropriate sign, see 4.3.2.5) is added to the stored coefficient (line 9). Otherwise, the function returns `false` (line 12), which means that the tested term belongs to a new equivalence class. Then the graph and its coefficient are added to the list (not shown in the code).

If the recursion is not finished, the current graph is modified by permutations which are admissible in the sense of definition 4.3. First (line 16), a factor permutation is carried out among all factors with the `FactorKey` determined by `i`. The number of these factors is given by `i->second`. If the `FactorKey` represents a hermitian tensor, an additional "Bra-Ket permutation" on the corresponding factors are performed. Since for each factor there are only two possibilities (interchange or not), the possible combinations of these permutations can be represented by bit patterns (line 18). For `j=0`, no Bra-Ket interchange takes place. After the application of the permutation(s), the function is called again for the next `FactorKey` (line 21 or 25). In this way, all possible combinations of permutations are tried out until a matching graph is found or there is no possibility left.

4.3.2.5. Sign Determination

The sign with which the coefficient of a term is multiplied before being added to the "old" coefficient is determined as the signum of the permutation mapping the nodes of the corresponding graph to those of the reference graph with which it was compared. Since the connections are ordered based on the ordering of the nodes described above, differences between the two graphs can only be due to permutations within index groups, so this sign rule reflects the antisymmetry properties of the involved tensors.

Another possible sign change is due to the permutation of external indices. This applies to all terms in an equivalence class and is taken into account when the graph is converted into an algebraic expression again.

Listing 4.1: Recursive graph comparison

```
1  bool DistinctTermGraphs::recursiveFactorPermutations(const
       TermGraph & tg, const map<FactorKey, int> & factorCount,
       map<FactorKey, int>::const_iterator i, const RationalNumber & rn)
2  {
3    if(i==factorCount.end())  //recursion base
4    {
5      for(iterator j=begin(); j!=end(); ++j)
6        if(j->first.strictEqv(tg))  ///matching term found
7        {
8        // determine permutation (lineUp)
9          j->second += RationalNumber(lineUp.parity()) * rn;
10         return true;
11       }
12     return false;
13   }
14   for(Permutator p(i->second); p.valid(); ++p)
15   {
16   TermGraph h(tg.applyFactorPermutation(i->first, p));
17     if(i->first.type==HermitianTensorType)  //bra/ket symmetry
18       for(int j=0; j<(1 << i->second); ++j) //permutations
19       {
20       TermGraph hh(h.applyBraKetPermutation(i->first, j));
21         if(recursiveFactorPermutations(hh, factorCount,
               ++map<FactorKey, int>::const_iterator(i), rn))
22           return true; //matching term found
23       }
24     else
25       if(recursiveFactorPermutations(h, factorCount,
             ++map<FactorKey, int>::const_iterator(i), rn))
26         return true; ///matching term found
27   }
28   return false;
29 }
```

4.3.2.6. Examples

The two graphs in Figure 4.8 look identical, which means essentially that they are strictly equivalent in the sense of Definition 4.2, since they only differ (possibly) in

the numeration of indices within groups, which is not visible in the figure. Figure 4.9 shows an example of two graphs which are equivalent. They are not strictly equivalent since the two amplitudes have to be interchanged to achieve congruence. Here we have included the factor labels to make this visible. In the following figures we do not show them to save space.

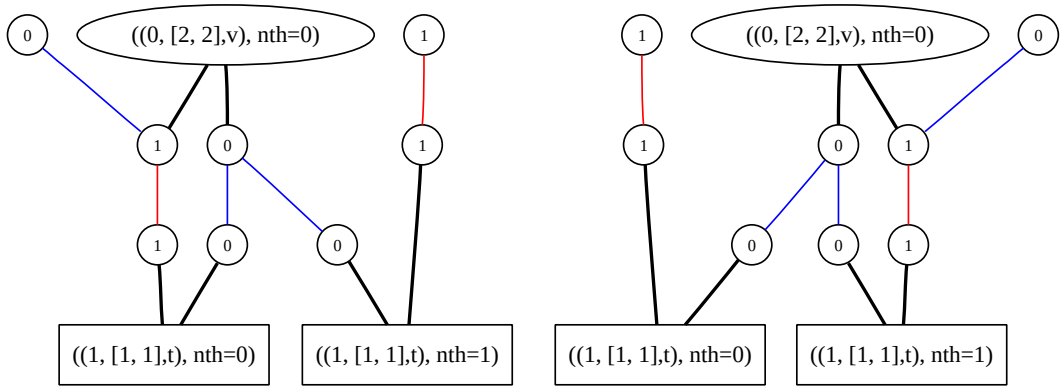


Figure 4.9.: Graphs with the same fingerprint which are equivalent but not strictly equivalent

The two graphs in Figure 4.10 have the same fingerprint, since they both contain two T_1 amplitudes and one two-electron integral and have the same types of connections. But they are not equivalent, because in the left graph the integral is connected to both amplitudes, and in the right graph only to one.

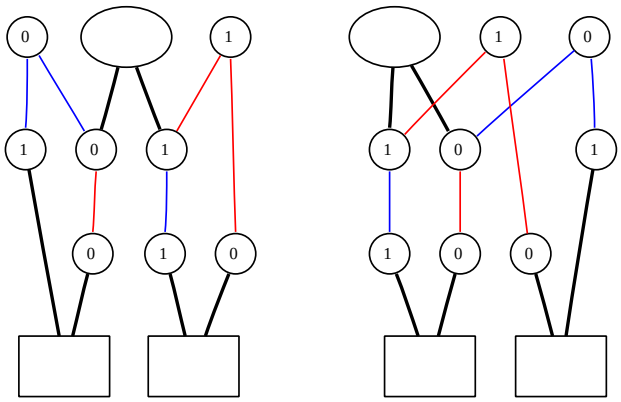


Figure 4.10.: Non-equivalent graphs which have the same fingerprint: $v^{Ia}_{Ai} t^B_i t^a_J$ and $v^{Ia}_{Ai} t^a_i t^B_J$

In this case the non-equivalence could also be seen easily from the algebraic expressions, since the second term is disconnected. But also in the non-obvious case from 4.1 the graphs clearly show the different structure of the terms (figure 4.11).

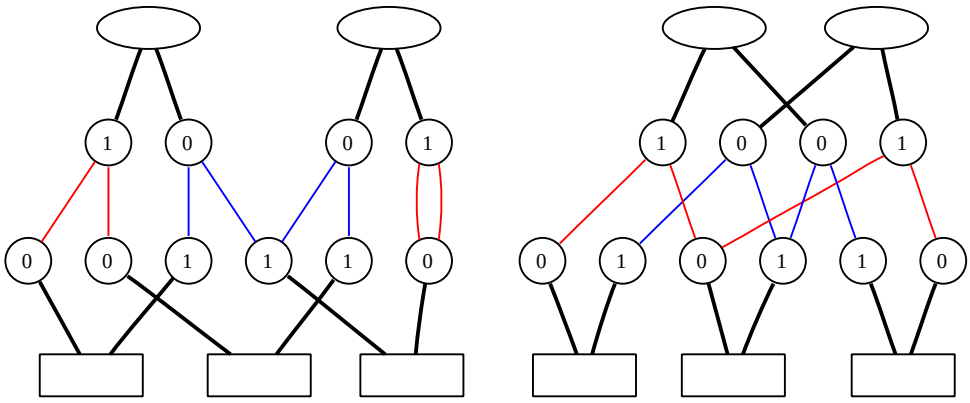


Figure 4.11.: Non-equivalent graphs which have the same fingerprint: $v^{ik}_{ab} v^{jl}_{cd} t^c_k t^d_l t^{ab}_{ij}$ and $v^{ik}_{ad} v^{jl}_{cb} t^c_k t^d_l t^{ab}_{ij}$

4.3.3. Discussion

The simplification procedure described in this section relies on a rather sophisticated data structure used to represent algebraic terms as graphs. The algorithm itself is in turn conceptually simpler than the algebraic approach in section 4.2. Moreover, it is more generally applicable, since it does not need assumptions on the term structure, e.g. the number of integrals. At the moment, the types of factors that can be treated are restricted to one- and two-electron integrals (`f` and `v`, respectively) and amplitudes of the cluster operator (`t`). But this list could easily be extended, since the algorithm does not make use of special properties of these tensors, except that there are two groups of indices on the integrals.

To show the effect of the simplification, we list in table 4.1 the number of terms before and after the simplification for some expressions containing deexcitation operators. These are more complicated than the usual CC equations. Some of the resulting formulas are shown in appendix C.2.

The efficiency of our simplification algorithm could certainly be improved, but since it is not time critical for coupled-cluster calculations, we did not investigate this further.

Table 4.1.: Term numbers before and after simplification for expressions $\langle\Phi_0|\exp(\hat{T})\dot{\hat{X}}\exp(\hat{T})|\Phi_0\rangle$, where $\hat{T} = \hat{T}_1 + \hat{T}_2$ and the exponential series is evaluated up to order n.

		term numbers	
$\hat{X}$	n	before simpl.	after simpl.
$\hat{H}$	1	28	18
	2	1150	319
	3	112394	6132
$\hat{H}^2$	1	650	215
	2	98181	9410

5. Tensor Contraction

The tensor contraction module of our program package is designed to be rather generally applicable, so it can not only be used to evaluate equations of coupled-cluster or similar methods. We only assume that we have objects which are described by a set of indices. These indices can be plain numbers or have additional structure. Our main purpose, however, is the case where the indices represent orbitals and the tensors are amplitudes (of the cluster operator or other operators), integrals (which could also be seen as amplitudes of the Hamilton operator), and objects constructed from them by contraction (intermediates and residuals as defined in chapter 3). For simplicity we will refer to this setup as the "coupled-cluster case" in the following, although it is not limited to that particular method.

The main problem in the (efficient) implementation of a tensor contraction procedure for the evaluation of coupled-cluster equations is the complicated structure of these tensors. This structure – which will be discussed in more detail in section 5.1 – leads to a conflict between memory usage and simple access to the tensor entries. Therefore, when we describe our implementation in section 5.3, we put special emphasis on the structures used to represent tensors, to iterate over them and to address individual tensor entries. Before we do this, we discuss different approaches to the contraction problem in general in section 5.2.

5.1. Tensor Structure

If the coupled-cluster equations are derived in terms of spin orbitals, the amplitudes have the property that they are antisymmetric with respect to the permutation of indices belonging to the same type of operators, i.e. annihilators or creators (in the usual notation this corresponds to lower and upper indices, e.g. we have $t_{ij}^{ab} = -t_{ji}^{ab} = -t_{ij}^{ba} = t_{ji}^{ba}$). To avoid the storage of redundant data, the indices within one such group are usually restricted to be in ascending order. Similar properties hold for the two-electron integrals, which are antisymmetrized as described in 2.2.1.4, and the residuals. The symmetry properties of the intermediates resulting from contractions of the input tensors depend on the factors from which they are constructed and on the contraction pattern.

Besides this "internal" symmetry, additional "external" restrictions can be imposed on the indices:

1. If the system (e.g. the molecule) under consideration has a nontrivial symmetry group (in case of molecules this is a point group), also the expression for its (correlation) energy is invariant under these symmetry operations. To exploit this property, only symmetry-conserving excitations are included in the cluster operator. This means that the product of the irreducible representations of the creation operators must be equal to the product of the irreducible representations of the annihilation operators:

$$\bigotimes_a \Gamma_a = \bigotimes_i \Gamma_i \tag{5.1}$$

2. If spin orbitals are used in coupled-cluster, the resulting wave function can – in the general open shell case – not be expected to be a spin eigenfunction. But at least the desired spin projection – i.e. the expectation value of $\hat{S}_z$ – can be enforced by including only those excitations where the sum of the spin projections of all creation operators minus the sum of the spin projections of all annihilation operators is zero.

The same restrictions apply to the integrals and in principle also to the intermediates. The latter is not completely obvious and will be discussed in section 5.3.

Both types of restrictions together make a "compressed" representation of tensors – i.e. one which contains only non-redundant entries – rather difficult. While the antisymmetry leads to a "triangular" form, the external restrictions cause sparseness in the sense that certain entries are zero. Or, to avoid storing zeros, certain index combinations have to be excluded, but there is no simple rule which ones. And to evaluate a complicated condition (or, in fact, any "if" condition) each time a tensor entry is accessed is prohibitive for efficient operations.

5.2. Possible Approaches

There are several ways to realize a tensor contraction in a computer program, which vary in their implementational complexity, efficiency, and applicability.

5.2.1. Explicit Loops

If a contraction is given in the form

$$\sum_{i_1,\ldots,i_k} X_{I_1\ldots I_m,i_1\ldots i_k} Y_{I_{m+1}\ldots I_n,i_1\ldots i_k}$$

(compare equation (3.1), here the product contains only two factors) it is in principle straightforward to write down the corresponding code. The basic structure is shown in algorithm 3. The `foreach` loops there would in fact be replaced by several loops over the individual indices (summed and non-summed, respectively).

Algorithm 3: Straightforward tensor contraction algorithm

Data: Tensor t_1, Tensor t_2, Tensor t_3 initialized to 0
Result: Tensor t_3
foreach *entry of* t_3 **do**
  foreach *summed index* **do**
    for *j=1,2* **do**
      construct index $\underline{i}^{(j)}$ from $entry$ and $summed\ index$;
    end
    $entry \mathrel{+}= t_1[\underline{i}^{(1)}] * t_2[\underline{i}^{(2)}]$;
  end
end

This approach, however, brings about several problems. First, the loops needed vary from contraction to contraction. To avoid writing separate code for each type of contraction (which would severely limit the applicability), there are basically two possibilities:

1. Use generic loops or an abstract iterator

2. Generate specialized code in an automated way.

The first method is convenient, but inefficient. The main disadvantage is that for each multiplication three tensor entries (first factor, second factor, and result) are needed and the addressing for these - which is expensive if the tensors are stored in a compressed form - takes place within the innermost loop. Moreover, it is difficult to exploit information about the particular structure of the tensors.

While the tensor addressing can be done efficiently in the second approach by using an incremental addressing scheme, the drawback remains that the contraction is performed as a sequence of single multiplications. Such a code can be rather efficient for small tensors, but for larger problems a good performance can only be reached with vectorized operations. Another problem is that for higher excitations the number of possible contraction types grows rapidly, and also the code for one contraction can become very long.

5.2.2. Vectorization

A tensor contraction can be cast into the form of a matrix multiplication by defining super-indices consisting of all indices that are summed or non-summed, respectively, like in the following example: The contraction

$$Z_{IJKL} = \sum_{ij} X_{IJij} Y_{KLij} \tag{5.2}$$

can be written as

$$Z_{\tilde{I}\tilde{K}} = \sum_{\tilde{i}} X_{\tilde{I}\tilde{i}} Y_{\tilde{K}\tilde{i}} = \sum_{\tilde{i}} X_{\tilde{I}\tilde{i}} Y^T_{\tilde{i}\tilde{K}} = XY^T \tag{5.3}$$

where $\tilde{I} = (IJ)$, $\tilde{K} = (KL)$, $\tilde{i} = (ij)$.

However, this leads to several practical problems. First, for the matrix multiplication to be efficient, the matrix entries have to be stored in the right order (row-wise or column-wise, depending on the implementation). In any case, the summed indices have to be those that vary most rapidly. But the same tensor (e.g. the T_2 amplitudes) is used in different contractions and not always the same indices are contracted, so this requirement is difficult to fulfill. Basically there are two possibilities:

1. Store each tensor in different ways corresponding to different contraction patterns.

2. Store each tensor only once and rearrange the entries temporarily for each contraction.

The first approach is clearly more efficient regarding calculation time, but it is only practicable under certain limiting conditions. First, of course all occurring contraction patterns have to be known in advance. Second, the needed storage space has to be affordable, so there should not be too many tensors and storage forms and the individual tensors must not be too large. Since number and size of tensors grow strongly for higher excitations, this is infeasible for our purposes.

Besides the order of the indices, there is an additional complication. If the tensors are stored in a compressed way taking into account the restrictions described above, they can not be interpreted as matrices directly. For in a matrix, row and column (super-) index have to be independent. In particular there can be no order relation between them. Hence, if there is an index group where some indices are contracted and some others are not, the inequality restriction in this group has to be (partly) abandoned. This increases the size of the matrix compared to the tensor and causes some matrix entries to be zero.

If we consider the example above (equations (5.2) and (5.3)) and assume that all tensors are totally antisymmetric (i.e. for instance for X we assume $I < J < i < j$) and all indices run from 1 to some number n, then each tensor has the size $\binom{n}{4}$ (since here

all tensors have four indices). Now for the matrices we are only left with restrictions between two indices each, e.g. $I < J$ and $i < j$, while there is no relation between e.g. I and i. So the size of the matrices corresponding to X, Y and Z is $\binom{n}{2}^2$ which is much larger than $\binom{n}{4}$[a]. While all entries where a row index is equal to a column index (e.g. $I = i$) are zero, each entry where all indices are different appears (up to sign) several times in the matrix (in this example four times). The sign of an entry is the parity of the permutation bringing the indices into the correct (ascending) order. We would like to emphasize that this increase in size affects only the memory requirements. The number of multiplications needed stays the same as for the "direct" contraction. The number of zeros due to antisymmetry is in practice not very large. We found e.g. fractions of $6 - 10\%$ for CCSDT.

For external restrictions, the problem is that the criteria depend on the values of all indices and thus cannot be evaluated for row or column indices alone. So the indices run without restrictions on their symmetry properties, and all matrix entries where the combination of row and column indices does not fulfill the symmetry requirements are zero. But in contrast to the zero entries caused by antisymmetry, these zeros occur in blocks, if the entries are arranged in a smart way. This can be exploited to make the contraction more efficient, as will be discussed later (see 5.3.3).

5.2.2.1. Matrix–Vector Multiplication

In the implementation by Kállay and Surján [41] amplitudes are treated differently from integrals and intermediates. While on amplitudes there are only two sorts of indices, namely summed and external (i.e. those determined by the projection), the other tensors can have a third sort of indices, namely those that will be summed in a subsequent contraction. Only the latter and the summed are used as matrix indices, so each integral or intermediate corresponds to a collection of matrices, labeled by external indices. All loops over external indices are carried out explicitly. The amplitudes having the right external indices are then arranged into a vector indexed by the summed indices. The innermost operation hence is a matrix-vector multiplication.

For us, this procedure has two drawbacks. First, the multiplication of a matrix with a vector (which can of course be seen as a special case of matrix–matrix multiplication) does not have the optimal efficiency (the best case would be the multiplication of two square matrices). This is illustrated by table 5.1. In the limit of large matrices, `dgemm` is about a factor of 14 faster than `dgemv`. Second, the assumptions about the index structure of amplitudes and intermediates are somewhat restrictive. For the sake of generality we prefer to treat all tensors equally.

[a] In the limit for large n the factor is six, but for smaller n it is larger (up to 36 for $n = 4$). Of course other cases are less dramatic, for example for two indices the factor ranges between two and four. But this is still not negligible, and if more than one index group is split in this way, the factors become even larger.

Table 5.1.: Performance (in MFLOPS per second) of matrix–vector (`dgemv`) and matrix–matrix (`dgemm`) multiplication routines from the mkl 10.2 library on a Core 2 Duo 3 GHz processor (single core used). "size" is the dimension of the (square) matrices and the vector.

size	`dgemv`	`dgemm`
4	369	352
7	952	923
10	1701	1931
14	2688	3145
22	3846	4202
32	5814	6667
45	3906	3623
71	2033	5102
100	2358	5500
141	2890	6584
224	3185	11115
316	3597	8557
447	3356	6380
707	3522	6093
1000	909	8929
1414	745	10097
2236	733	10275
3162	706	10218

5.2.2.2. Matrix Multiplication after Rearranging

In order to perform a matrix–matrix multiplication as the innermost step in the contraction process, it is necessary to restore the data contained in the tensors to be contracted in a suitable way, as discussed above. In particular, the tensors, which are stored in the smallest possible form, are partly "unpacked" by releasing some of the restrictions. The increase in size is minimized by contracting tensors blockwise, where the indices in each block have fixed symmetry properties (see 5.3.3 for details).

The rearranging is a rather complicated process, but it scales only as $\mathcal{O}(N^2)$, where N is the matrix dimension (square matrices assumed), compared to $\mathcal{O}(N^3)$ for the matrix multiplication. So for large tensors the matrix multiplication will eventually dominate the total contraction time, but it is hard to get to this point because it has a relatively small prefactor. Moreover, the blockwise processing leads of course to smaller matrices. The goal of our implementation is to get the prefactor for the rear-

ranging step as small as possible.

A similar strategy has been pursued by Hirata in the development of the TCE [37]. This program contains a code generator which produces specialized code for each type of contraction. All tensors are processed in blocks (called tiles there), which can be defined e.g. by symmetry properties. Each tile is rearranged (if necessary) during the contraction, and then a matrix–matrix multiplication is performed.

5.3. Actual Implementation

In this section we start from a rather general setup, but in the course of the discussion several details are only given for the coupled-cluster case. We also use this as an example to illustrate important points.

5.3.1. Tensor Representation

A central idea of the tensor representation in our program is to separate information needed in different steps as far as possible. In particular, the entries of a tensor are separated from the structural information. Following this idea, a tensor is constructed in three steps. The relevant classes and their relations are shown in figure 5.1.

First, there is a so called `SymbolicTensor`, which is a quite immediate adaption of the symbols coming from the formula generation part of the program, but is more general in several respects. The latter consist basically of an identifier – distinguishing between amplitudes, one- and two-electron integrals – and two vectors of indices corresponding to annihilators and creators in the original operators, respectively. The indices carry – besides the operator type – the information whether they are summed or external and whether they are hole or particle indices, i.e. corresponding to occupied or virtual orbitals, and a number differentiating indices of the same type. This unique identification is important for the contraction procedure, since the indices to be contracted are determined as those indices which are common to two tensors. Within the tensor contraction part of the program the indices are just integer numbers, the properties mentioned above are coded in this number by a separate `TensorIndexInterpretation` class. This is done to have a simple structure and to make this module usable for different applications. For example, the information about the operator type is only needed if spin orbitals are used, since it determines the sign with which the spin projection of this orbital is to be multiplied (see 5.3.2.4). Also, the index types are not denoted as hole and particle here, but represented by numbers, so in principle there could be more (or less) than two types.

Another difference to the formula generation module is that now not only amplitudes and integrals have to be represented, but also intermediates, and thus it is not enough to have two groups of indices. To handle such structures we introduced the

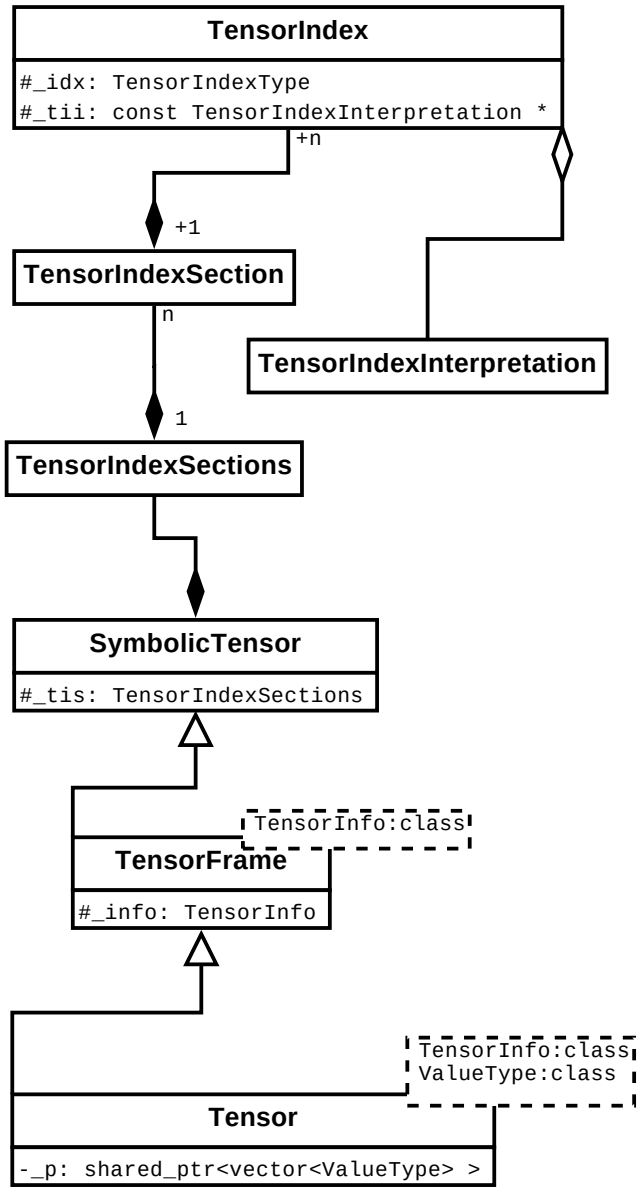


Figure 5.1.: UML diagram for classes related to tensor representation

class `TensorIndexSections`. It derives from `vector<TensorIndexSection>`. Each `TensorIndexSection` represents an index group and can contain one or several indices. Usually, the indices in one section are of the same type, but this is not enforced by the data structure. A `SymbolicTensor` object contains an instance of `TensorIndexSections` and has some additional member functions like addition and multiplication (contraction). It is assumed that the tensor represented by the `SymbolicTensor` is antisymmetric with respect to permutations of indices within each section.

While the `SymbolicTensor` carries only basic structural information about the tensor, e.g. its dimension and symmetry properties, the class `TensorFrame` derived from it, which is constructed in the second step, contains additional data allowing, among other things, to determine the actual size of the tensor. This data, like the point group or the number of orbitals in each irreducible representation, is collected in a `TensorInfo` class.

Finally, the `Tensor` class derived from `TensorFrame` contains the actual entries. For this we use the `vector` class from the STL which allows an efficient access to arbitrary entries by index. For efficiency reasons (to avoid copying), the `Tensor` class does not contain this vector itself, but a shared pointer to it. The structures needed to find a particular entry, defined by a certain index combination, in the tensor are not part of the tensor itself but are built when they are first needed and then cached (see 5.3.4).

5.3.2. Indices and Iterators

5.3.2.1. Index Representation

The indices used to identify particular orbitals and whose combinations identify the tensor entries – not to be confused with the indices described above which describe the structure of a tensor – should be simple integers for efficiency reasons, since they appear in the innermost part of the contraction procedure. On the other hand, each index has to carry a lot of information, e.g. for coupled-cluster whether the orbital is occupied or virtual, which irreducible representation it belongs to and – in the case of spin orbitals – which spin projection it has. Therefore we introduced table classes (e.g. `SpinSpatialMOTable_IrrepSzIdx` for spin orbitals) which can interpret integer numbers as structured indices. Depending on the context (i.e. which information is needed) different tables with different numbers of hierarchy levels can be used. In the case of coupled-cluster there are three or four levels, depending on which type of orbitals is used. The first one is the occupation status (labeled `oav` for occupied/active/virtual). The next levels are the irreducible representation (abbreviated `irrep`) and (only in the case of spin orbitals) the spin projection `sz`. The last one is the position of the index among all with the same properties (`idx`). A small example (ten occupied

and four virtual spin orbitals) is shown in table 5.2.

Table 5.2.: Example for hierarchical index orbital table. Here, `flatIdx` numbers all orbitals, starting with the occupied ones. The value of `oav` is 0 for occupied orbitals and 2 for virtual orbitals (active orbitals, which are not present in the example, would have `oav`=1).

	level			
flatIdx	0	1	2	3
	oav	irrep	sz	idx
0	0	0	0	0
1	0	0	0	1
2	0	0	0	2
3	0	0	1	0
4	0	0	1	1
5	0	0	1	2
6	0	1	0	0
7	0	1	1	0
8	0	2	0	0
9	0	2	1	0
10	2	0	0	0
11	2	0	1	0
12	2	2	0	0
13	2	2	1	0

We want to be able to generate all possible index combinations for a tensor, i.e. iterate over its entries, in a systematic manner. For example it is reasonable to change the symmetry classes the indices belong to as seldom as possible, as we will see later. Therefore we need an iterator which is aware of the particular hierarchical index structure. At the same time, all information which is related to the current state (i.e. which changes in the course of the iteration) should be separated from the actual iterator. To implement this we introduced a class `SuperIteratorIndex` (see listing 5.1 for the declaration). The length of a `SuperIteratorIndex` is the original number of indices times the number of hierarchy levels. So for a tensor with n indices and l levels we have $n \cdot l$ entries, where the first n correspond to the outermost level, the next n to the second, and so on. The last n entries contain the full indices including all levels.

As an example, consider a tensor with four indices i, j, a, b and the orbitals from table 5.2. Then each of the three vectors contained in the `SuperIteratorIndex` (lines 9–11 in listing 5.1) has 16 entries. The `idx` vector for the tensor element with flat indices

$1, 5, 10, 13$ would look like this:

$$[\underbrace{0\,0\,2\,2}_{\texttt{oav}}\ \underbrace{0\,0\,10\,12}_{\texttt{irrep}}\ \underbrace{0\,3\,10\,13}_{\texttt{sz}}\ \underbrace{1\,5\,10\,13}_{\texttt{idx}}]$$

The `oav` level is included here, although at this level no real iteration takes place, since there is only one allowed value for each index (see below). At the outer levels, each index is replaced by the first (flat) index with the same values up to this level. For example, 5 is replaced by 3 in the `sz` group, since 3 is the smallest index of an occupied orbital which belongs to the same representation (0) as orbital 5 and has the same spin projection (1). This way of representing a hierarchical index is more efficient (for iteration) than storing the actual values for each level as given in the table.

Listing 5.1: Class declaration for `SuperIteratorIndex`

```
 1 class SuperIteratorIndex {
 2 public:
 3   SuperIteratorIndex(unsigned int size);
 4
 5   bool operator < (const SuperIteratorIndex & sii) const;
       //compares only idx
 6
 7   void  set(const SuperIteratorIndex & sii);  //copy values
       efficiently
 8
 9 vector<int>     idx;      // actual index (to be iterated)
10 vector<char>    valid;
11 vector<char>    outerSmaller; // flag if outer level satisfies "<"
     restriction
12 bool  valid0;   // this is active iff idx.size()==0 (tensors of
     rank 0) to guarantee a single iteration step
13 };
```

5.3.2.2. Iteration Procedure

The incrementation of a `SuperIteratorIndex`, i.e. finding the next valid index combination, is quite complicated. Of course it would be simple to write a corresponding loop structure for a given tensor. But since there are many different possible tensor structures, we would need many different sets of loops, which is impractical. For a generic incrementation, and a recursive "trial and error" procedure is necessary.

For each index, i.e. each entry in the `SuperIteratorIndex`, there is a separate iterator (`AtomicIterator`, see listing 5.2), and all these iterators are collected in a

`SuperIterator`. The iteration is split up into two functions, which are both members of `AtomicIterator`. The first, `inc`, does the actual incrementation, and `reset` is used to correct the indices following the last incremented one in accordance with all restrictions. Their code is given in listings 5.3 and 5.4. The functions call each other (lines 12 and 33, respectively), and themselves (lines 18 and 31, respectively) recursively.

We do not want to discuss the procedure in all details here, but explain how it works in principle and why it is complicated.

We apply the convention that the last index is the fastest running one. Hence, if a `SuperIterator` has to increment its corresponding index, it first tries to increment the last entry. If this does not work, the index before it is incremented, and so on until the first index is reached. Each time a new index is incremented (after an "overflow" of the previous) all subsequent indices are reset to appropriate start values. The function `inc` returns the position of the outermost incremented index, which is given by the element `_this` of the corresponding `AtomicIterator` (5.2, line 33). If none of the indices can be incremented any more, the `SuperIteratorIndex` index is marked as invalid (5.3, line 22, or 5.4, line 3; if the reset is successful, the index is validated again in line 37). The incrementation itself is done by the index table classes (5.3, line 4, and 5.4, lines 12 and 26), since only they know what the next index at the corresponding level is.

If there are several hierarchy levels, the current index is not incremented to its maximal possible value, but only up to the point where the next higher level would be affected. In the example above, the combination (1,5) in the occupied indices is not followed by (1,6), since this would change `irrep` and `sz` of the second index, but by (2,3). For further examples see table 5.3 and listing 5.5. The different iteration patterns are also illustrated in figure 5.2.

Now we have to take into account the antisymmetry properties of our tensors. To store only non-redundant entries, we require the indices within one index group to be in ascending order. To implement this property for hierarchically structured indices is not completely trivial. The strict inequality relation applies only to the complete indices, i.e. at the innermost level. At outer levels, succeeding indices can also be equal. On the other hand, if the restriction is already strictly fulfilled at some level, all following inner levels are completely free, since the index order is dominated by outer levels. So the possible values for one index entry depend on the values of several other entries, and this has to be taken into account in the incrementation process (5.3, lines 10/11, 5.4, lines 4–11 and 28/29). The basic property whether an index is bound to another index by an inequality restriction is part of the iterator (element `_left`, see line 34 in listing 5.2), while the information about the fulfillment of this restriction at outer levels – which can change during the iteration – is stored in the `SuperIteratorIndex` (`outerSmaller`, line 11 in listing 5.1). Both values together determine whether an index in a specific situation is restricted or not.

Listing 5.2: Class declaration for `AtomicIterator`

```
1  class AtomicIterator :  public SubIterator {
2  public:
3    AtomicIterator();
4
5    // returns true if SubIterator is valid
6    // considers this SubIterator only (==> quick)
7    virtual bool valid(const SuperIteratorIndex &) const;
8
9    // returns true if SubIterator is valid
10   // starts at inner loop, runs outwards, stops at "outermost"
        (inclusive)
11   virtual bool deepValid(const SuperIteratorIndex &, SubIterator *
        outermost) const;
12
13   // returns position of outermost changed index, -1 if no further
        iteration possible
14   // starts at inner loop, runs outwards, stops at "outermost"
        (inclusive) and "innermost" (inclusive) in case of reset
15   virtual int inc(SuperIteratorIndex &, SubIterator * innermost,
        SubIterator * outermost);
16
17   // returns true if SubIterator contains a valid step
18   // starts at outer loop, runs inwards, stops at "innermost"
        (inclusive)
19   virtual bool reset(SuperIteratorIndex &, SubIterator * innermost);
20
21   bool boundToOuter(const SuperIteratorIndex &) const;  //true if
        this loop is restricted by value of next outer loop
22 // ...
23 friend class SuperIterator;
24
25 protected:
26 int _hLevel;  // hierarchy level, (N-1) = innermost
27 bool _applyTriangleRestriction;   // actually ensure "<" restriction
28 const RestrictionPredicate * _restriction;
29
30 // state dependent variables are stored in SuperIteratorIndex
31 // variable locations are stored as array positions within
      SuperIteratorIndex
32 // e.g. _left=3  ==>  SuperIteratorIndex[3].idx is left index
33 int _this;  // position of state dep. variables of this iterator
34 int _left;  // left index if triangle restricted, else -1
35 int _nextInnerLevel, _nextOuterLevel; // corresponding index at
      next inner/outer level, -1 if no inner/outer level exists
36 const FlatIndex2HierarchicalIndex2 *  _hTab;
37 };
```

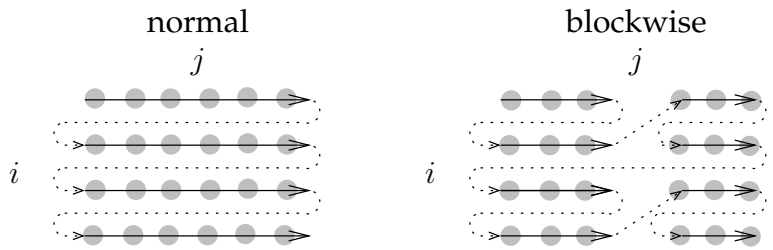


Figure 5.2.: Illustration of "normal" and blockwise (i.e. respecting hierarchy levels) iteration for two indices.

Listing 5.3: Incrementation function of `AtomicIterator`

```
1  int AtomicIterator::inc(SuperIteratorIndex & sii, SubIterator *
       innermost, SubIterator * outermost)
2  {
3  next:
4    if ( _hTab->inc(sii.idx[_this], _hLevel) )  //increment current
         index
5    {
6      if ( _restriction && !(*_restriction)(&sii.idx[_this]) )
7        goto next;
8      if ( !_nextInner )
9        return _this;
10     if ( _left>=0 && _nextInnerLevel>=0 ) //update outerSmaller if
           necessary
11       sii.outerSmaller[_nextInnerLevel]  = sii.outerSmaller[_this]
             || (sii.idx[_left]<sii.idx[_this]);
12     if ( _nextInner->reset(sii, innermost) )  //reset subsequent
           indices
13       return _this;
14     else
15       goto next;
16   }
17 int pos = 0;
18   if ( this!=outermost && _nextOuter && (pos=_nextOuter->inc(sii,
         innermost, outermost))>=0 ) //increment next outer index
19     return pos;
20   else  //incrementation failed, invalidate index
21   {
22     sii.valid[_this] = false;
23     return -1;
24   }
25 }
```

Listing 5.4: Reset function of `AtomicIterator`

```
1  bool AtomicIterator::reset(SuperIteratorIndex & sii, SubIterator *
       innermost)
2  {
3    sii.valid[_this] = false;
4    if ( _left>=0 && _applyTriangleRestriction )  // check if
         triangle restriction applies
5    {
6      if  ( sii.outerSmaller[_this] )
7        sii.idx[_this] = sii.idx[_nextOuterLevel];  //  ==> no
           triangle restriction
8      else
9      { // no further outer level or "=" at outer level ==> triangle
         restriction applies
10       sii.idx[_this] = sii.idx[_left];
11       if ( _nextInnerLevel<0 )  // if innermost level: "<" applies,
           else: "<=" applies
12         if ( !_hTab->inc(sii.idx[_this], _hLevel) ) // check if
             valid
13           return false;
14     }
15   }
16   else
17   {
18     if ( _nextOuterLevel>=0 ) // check if not outermost
19       sii.idx[_this] = sii.idx[_nextOuterLevel];
20     else
21       sii.idx[_this] = _hTab->begin();
22   }
23 Next:
24   if ( _restriction )
25     while ( !(*_restriction)(&sii.idx[_this]) )
26       if ( !_hTab->inc(sii.idx[_this], _hLevel) )
27         return false;
28   if ( _left>=0 && _nextInnerLevel>=0 )
29     sii.outerSmaller[_nextInnerLevel] = sii.outerSmaller[_this] ||
         (sii.idx[_left]<sii.idx[_this]);
30   if ( _nextInner )
31     if ( !_nextInner->reset(sii, innermost) ) // check if inner
         iteration possible
32     {
33       if ( inc(sii, innermost, this)>=0 )
34         goto Next;
35       return false;
36     }
37   sii.valid[_this] = true;
38   return true;
39 }
```

In addition to this "internal" antisymmetry constraint there can be "external" restrictions. In the coupled-cluster case for example, not all combinations of irreducible representations or spin projections are admissible. This will be discussed in detail in sections 5.3.2.3 and 5.3.2.4. The constraint that some indices run over occupied and some over virtual orbitals is also treated as an external restriction.

Since our iterators should be rather generally applicable, we implemented a structure which allows for arbitrary external restrictions: Each iterator can contain a so called `RestrictionPredicate` which imposes a constraint on the corresponding index or on several indices. If an index is changed, the `RestrictionPredicate` at this position (if there is one) is evaluated (5.3, line 6, 5.4, lines 24/25). If this confirms that the current index combination is valid, the incrementation is successful, otherwise the next value is tested.

Table 5.3.: Different iteration patterns for two indices $i, j \in \{0, 1, 2, 3\}$. In the "block iteration" case there are only two hierarchy levels for simplicity, where the outer level is defined by assigning the indices $0, 1$ to one class and $2, 3$ to another. The "symmetry" restriction requires that the classes of i and j are equal, while "triangle" means that the restriction $i < j$ is applied.

	normal iteration				blockwise iteration					
flat index	unrestricted		triangle		unrestricted		symmetry		sym. + triang.	
	i	j	i	j	i	j	i	j	i	j
0	0	0	0	1	0	0	0	0	0	1
1	0	1	0	2	0	1	0	1	2	3
2	0	2	0	3	1	0	1	0		
3	0	3	1	2	1	1	1	1		
4	1	0	1	3	0	2	2	2		
5	1	1	2	3	0	3	2	3		
6	1	2			1	2	3	2		
7	1	3			1	3	3	3		
8	2	0			2	0				
9	2	1			2	1				
10	2	2			3	0				
11	2	3			3	1				
12	3	0			2	2				
13	3	1			2	3				
14	3	2			3	2				
15	3	3			3	3				

The effects of different restrictions for a simple example (two indices) are shown in table 5.3. In the case with four levels described above, each of the indices of the outermost level carries a `RestrictionPredicate` defining the occupation status of the corresponding orbital. These restrictions ensure that the particular index structure of each tensor is respected while we can use the same type of iterator for all of them. The other restrictions (regarding irreducible representations and spin projections, respectively) affect all indices of the corresponding level. The respective `RestrictionPredicate` is located at the last index of that level.

The index table as well as the `RestrictionPredicates` are contained in the `TensorInfo` classes, so that a tensor contains all necessary data to construct an iterator for its entries.

5.3.2.3. Dealing with Point Group Symmetry

In implementations of correlation methods it is common to treat only subgroups of D_{2h}, i.e. the six point groups C_1, C_2, C_s, C_{2v}, C_{2h}, and D_{2h}, since for these, the symmetry condition takes a much simpler form. For all other groups, the additional implementational effort would be very high, compared to a rather small gain in efficiency.

Since D_{2h} contains only elements of order two, all irreducible representations of the groups above are one-dimensional and the characters can only be ± 1. Under these conditions, the requirement (5.1) is equivalent to saying that the product of the irreducible representations of all involved orbitals has to be the totally symmetric irreducible representation.

This condition applies also to intermediates, which can be seen as follows: Consider the contraction of two tensors which both fulfill the condition that the product irreducible representation is totally symmetric. Then the product of all irreducible representations from both factors is also totally symmetric. Since every summed index occurs twice and the product of an irreducible representation with itself is (in the considered case) always totally symmetric, the representations corresponding to summed indices can be crossed out and the remaining product is still totally symmetric.

Within our program, irreducible representations are identified with integer numbers, where 0 corresponds to the totally symmetric one, and the others are numbered arbitrarily. For each group a precalculated multiplication table is stored, so that the direct product of two irreducible representations can be evaluated efficiently. For an index combination to be valid, the total product should yield 0.

5.3.2.4. Enforcing Spin Projections

For an excitation operator to conserve the expectation value of $\hat{S}_z$ the following condition has to be fulfilled:

$$\sum_{\nu=1}^{n} S_z(a_\nu) - \sum_{\nu=1}^{n} S_z(i_\nu) = 0,$$

where a_ν are the creator and i_ν the annihilator indices, or shorter: $\sum_{\nu=1}^{2n} \epsilon_\nu S_z(p_\nu) = 0$ with $\epsilon_\nu = \pm 1$. We now assume this condition to hold for all amplitudes and integrals and consider an intermediate tensor constructed from two such quantities by contraction. We have the two conditions

$$\sum_{\nu=1}^{2n} \epsilon_\nu S_z(p_\nu) \quad = 0 \tag{5.4}$$

$$\sum_{\mu=1}^{2m} \epsilon_\mu S_z(q_\mu) \quad = 0. \tag{5.5}$$

Now we assume that the last s indices are contracted, i.e. we have a bijection

$$\pi : \{2n - s + 1, \ldots, 2n\} \to \{2m - s + 1, \ldots, 2m\}$$

such that $p_\nu = q_{\pi(\nu)}$. If we solve (5.4) and (5.5) for the terms belonging to the contracted indices and combine the two equations, it follows that

$$\sum_{\nu=1}^{2n-s} \epsilon_\nu S_z(p_\nu) = \pm \sum_{\mu=1}^{2m-s} \epsilon_\mu S_z(q_\mu)$$

or equivalently

$$\sum_{\nu=1}^{2n-s} \epsilon_\nu S_z(p_\nu) + \sum_{\mu=1}^{2m-s} \epsilon'_\mu S_z(q_\mu) = 0$$

where

$$\epsilon'_\mu = \begin{cases} -\epsilon_\mu & \text{if } \epsilon_\nu = \epsilon_{\pi(\nu)} \text{ for } \nu = 2n - s + 1, \ldots, 2n \\ \epsilon_\mu & \text{if } \epsilon_\nu = -\epsilon_{\pi(\nu)} \text{ for } \nu = 2n - s + 1, \ldots, 2n \end{cases}$$

So with appropriately defined signs ϵ we can apply the same condition for all tensors. If there is no contracted index, the sign for the indices of the intermediate are chosen such that indices of the same type (particle or hole) get the same sign.

In the actual implementation we replace the S_z values $\pm\frac{1}{2}$ by 1 and 0, i.e. we shift them by $\frac{1}{2}$. Therefore we have to modify the condition slightly, namely

$$\sum_{\nu=1}^{2n} \epsilon_\nu S_z(p_\nu) = \frac{1}{2} \sum_{\nu=1}^{2n} \epsilon_\nu.$$

Listing 5.5: Usage example for tensor classes and iterator

```
int main()
{
//... (initializing MO table etc.)
TensorIndexInterpretation_SE_AC tII;

SymbolicTensor  st(TensorIndexSections(tII, "i-j-,a+b+"));
  cout << "st=" << st << endl;
SpinOrbital_OAV_TensorInfo  info(table, &pg,
   st.tensorIndexSections());
  cout << "Info: " <<  info << endl;

TensorFrame<SpinOrbital_OAV_TensorInfo> tf(st, info);
Tensor<SpinOrbital_OAV_TensorInfo, double> t(tf);

IndexView iView;
boost::shared_ptr<SuperIterator> sIter(tf.getIterator());
  for ( SuperIteratorIndex sIdx(sIter->begin()) ;
      sIter->valid(sIdx) ; sIter->inc(sIdx) )
    cout << iView(sIdx) << endl;
}

//Output:
st={[[S_0_0-, S_0_1-], [S_2_0+, S_2_1+]]}
Info: SpinOrbital_OAV_TensorInfo,
Restrictions:
OAV, value=0, pos=0
OAV, value=0, pos=1
OAV, value=2, pos=2
OAV, value=2, pos=3
Irrep, pos=7
Sz, ac= (-1 -1 1 1 ), pos=11

[0 0 10 10 0 0 10 10 0 0 10 10 0 1 10 11]
[0 0 10 10 0 0 10 10 0 0 10 10 0 1 10 12]
[0 0 10 10 0 0 10 10 0 0 10 10 0 1 10 13]
[0 0 10 10 0 0 10 10 0 0 10 10 0 1 11 12]
[0 0 10 10 0 0 10 10 0 0 10 10 0 1 11 13]
[0 0 10 10 0 0 10 10 0 0 10 10 0 1 12 13]
[0 0 10 10 0 0 10 10 0 0 10 10 0 2 10 11]
[0 0 10 10 0 0 10 10 0 0 10 10 0 2 10 12]
[0 0 10 10 0 0 10 10 0 0 10 10 0 2 10 13]
[0 0 10 10 0 0 10 10 0 0 10 10 0 2 11 12]
[0 0 10 10 0 0 10 10 0 0 10 10 0 2 11 13]
[0 0 10 10 0 0 10 10 0 0 10 10 0 2 12 13]
[0 0 10 10 0 0 10 10 0 0 10 10 1 2 10 11]
//... (630 entries)
[0 0 10 10 8 8 20 20 8 9 20 23 8 9 22 23]
[0 0 10 10 8 8 20 20 8 9 20 23 8 9 22 24]
[0 0 10 10 8 8 20 20 8 9 20 23 8 9 22 25]
```

5.3.2.5. Usage Example

Listing 5.5 illustrates the usage of the structures introduced so far. The initialization of the point group (`pg`) and orbital table (`table`) is not shown. Next, we have to fix a `TensorIndexInterpretation` (line 4). The suffix `_SE_AC` indicates that each index contains – in addition to its number and type – the information whether it is summed or external and whether it belongs to an annihilation or creation operator. The `SymbolicTensor` (`st`, line 6) determines the structure of the considered tensor. Here we have two hole indices (i, j) coming from annihilator (shown by the minus sign) and two particle indices (a, b) coming from creators. The comma groups the indices, so we assume $i < j$ and $a < b$.

Next, the `TensorInfo` object is constructed from this structure together with `pg` and `table` (line 8). From this data the `RestrictionPredicates` for the iterator (line 15) are determined. They are shown in lines 24–29. A sample of the indices resulting from the iteration (line 16) is given in lines 31–47.

5.3.3. Contraction Procedure

As a first step, the contraction of two tensors is carried out at the symbolic level. That means the indices which the tensors have in common are identified and the structure of the resulting tensor is determined. In principle this is given by concatenating the non-contracted indices of the two factors. But if external indices are present, they are collected in two groups, one for hole and one for particle indices. This enforces antisymmetry of the resulting tensor with respect to permutations of external indices, which is what we want to have at the end. Of course the antisymmetrization could also be done afterwards, but it is more efficient to do it in each contraction step, since the intermediates to be stored are smaller. After that, the result tensor is constructed and initialized with zero entries, and all auxiliary structures needed for the contraction are created.

As discussed in 5.2.2, we want to use matrix multiplications to carry out the contraction of two tensors. The straightforward way to do so would be to convert both tensors into matrices, where the contracted indices form a super-index for the columns and the non-contracted are used for labeling the rows (for the second factor it should be the other way round, but we assume this matrix to be transposed[b]), multiply them, and then restore the resulting matrix into a tensor. We have already seen some problems related to this step, in particular the fact that these matrices would contain a lot of zero entries for two reasons:

[b] This does not only spare the distinction between the two factors, but is also advantageous for the matrix multiplication. If the second matrix is not given in transposed form, the `dgemm` routine does the transposition itself in many cases for efficiency reasons.

1. The antisymmetry properties of the original tensors cause matrix entries to be zero when index groups are split, i.e. some indices of the group are contracted and some are not.

2. If there are external restrictions, matrix entries whose indices do not fulfill them are zero. This can be the spin and symmetry restrictions described before, but also other restrictions affecting more than one index.

While the first problem is inevitable in our situation, the second can be circumvented in many cases, as indicated in 5.2.2.2. We make the assumption that each external restriction only takes effect within one index level and that there is no restriction at the innermost level. This is of course fulfilled in the coupled-cluster case. Because of the way our tensors and iterators are constructed, the second sort of zeros then occurs in blocks. So it is a rather obvious idea to also do the contraction blockwise and leave out the zero blocks. So what we actually do is the following (see algorithm 4): First we iterate over the blocks of the first tensor, convert each of them into a matrix and store these matrices. Then we do the block iteration and conversion for the second tensor, and each block is – directly after the conversion – multiplied with all blocks which have the same sequence of contracted indices, and the result is written to the result tensor.

Algorithm 4: Tensor contraction using blockwise matrix multiplication

```
Data: Tensor t_1, Tensor t_2, Tensor t_3 initialized to 0
Result: Tensor t_3
foreach block of t_1 do
    convert block to matrix and store result;
    foreach block of t_2 do
        convert block to matrix;
        foreach matching block from t_1 do
            multiply matrices;
            write result to t_3;
        end
    end
end
```

Now we want to describe the iteration and conversion process in some more detail. For the separate iteration over the blocks and the entries in the block we define so-called `IteratorSections`, that are restrictions of the full iterator. The first section comprises all index levels except the last, the second one is for the innermost index level. For the result, there is no iteration over blocks. Instead, the start index for the block is composed from the non-contracted indices of the factor blocks. The conversion

itself is equal in both directions, expect for the last step, where in the first case the tensor entry is written to the matrix and in the second case the matrix entry is added to the corresponding tensor entry. In both cases the transferred value is first multiplied by the appropriate sign.

If there are split index groups, we do not have a one-to-one correspondence between tensor and matrix entries. Besides the fact that the matrix contains more zeros, each tensor entry appears several times in the matrix, partly with altered sign. Now we have two possibilities. Either we iterate over the tensor and determine all positions in the matrix where this entry has to be put (or which contribute to this entry), together with the corresponding sign. Alternatively, we iterate over the matrix and determine for each entry the corresponding tensor entry and sign. The second approach is easier to implement, since the index for the tensor entry and the sign are given by just sorting the indices within each group. If two or more indices are equal, the entry is zero. How the tensor entry for a given index is found will be discussed in the next subsection.

5.3.4. Tensor Addressing

To access a particular tensor entry, its address, i.e. its position in the vector, has to be determined from the indices defining the entry. For this purpose the block structure of the tensor can also be exploited. The address is calculated in two steps as a block offset (starting address for the block) plus the position of the entry within the block. The whole procedure is shown schematically in figure 5.3.

A block is defined by specifying for each index its value up to the prelast level. That means within the block the indices are only allowed to vary at the innermost level. Because of the restrictions discussed above, the set of allowed index combinations here is "sparse". In particular the next valid combination can not be easily determined from a given one. But since there are usually few blocks compared to the number of entries, all index combinations starting a block can be stored explicitly in a `map`. This data structure allows to find an arbitrary index combination efficiently. The address offset for a block is simply the sum of the sizes of all blocks before it. Together with the offset additional data needed for addressing within the block is stored.

The remaining part of the addressing takes place in the innermost part of the contraction procedure, therefore its efficiency is very important. In particular it is desirable to avoid repeating the complete addressing for each index combination. Instead we would like to have an update function which recalculates only that part of the address which corresponds to the indices last changed during the iteration.

Within a block, each index runs through a connected range of values, i.e. the index space does not contain any "holes", which makes the addressing in principle simple. If there were no further relations between the indices, the address for a given index combination could be easily calculated: Given indices $i_1, \ldots, i_n$ with $i_k \in \{1, \ldots, m_k\}$

the address is

$$\sum_{k=1}^{n} i_k \cdot \prod_{j=k+1}^{n} m_j.$$

Now we have the additional complication of antisymmetry in some indices. But between the different index groups there is no relation and so an analogous formula applies to them. The index values i_k are replaced by "group indices", and the multipliers m are now group sizes instead of index ranges. These multipliers are determined together with the block offsets (the block size is the product of the group sizes).

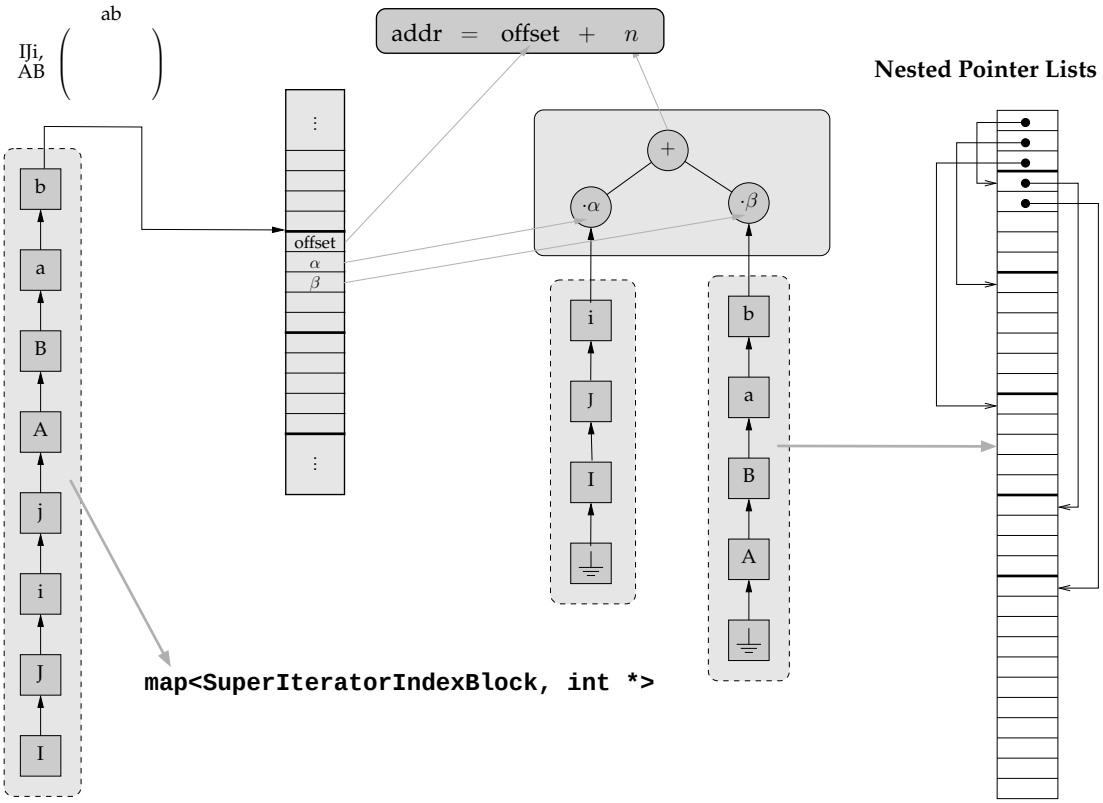


Figure 5.3.: Schematic overview of the tensor address calculation

The addressing within such a group is more complicated. If we assume the indices to be in ascending order, the address could still be calculated explicitly, although the corresponding formulas become increasingly complex when the number of indices grows. But we want to use the addressing also in the context of matrix–tensor conversion, and there we can not make this assumption. If one part of an index group belongs to the contracted indices and the other one not, the inequality relation between these indices is lost. The straightforward solution to this would be to first sort the indices and then do the addressing, but this is very inefficient. On the one hand, the sorting

becomes rather expensive for larger index groups, since its complexity is $\mathcal{O}(n \log n)$ if n is the group length. On the other hand, the whole procedure has to be repeated for each matrix entry, although most of the time during the iteration only a small part of the indices changes.

Another possibility is to precalculate a table where for each possible index combination the corresponding address and the sign the entry has to be multiplied by are stored. Such tables can be constructed in a way that the information which indices have been changed since the last access can be exploited to minimize the access cost. The interesting point is now to find all index combinations which occur during a specific matrix–tensor conversion. Starting from a valid index combination for the tensor, all matrix entries contributing to this tensor entry correspond to index combinations which are related to the given one by a permutation. But not all permutations are allowed. Firstly, the index groups are of course conserved. Restricting our attention now to one index group, we start with a sequence $i_1 < i_2 < \ldots < i_n$ of indices in ascending order. Now we divide this into two parts, namely contracted and non-contracted indices, corresponding to rows and columns of the matrix. Let k be the number of non-contracted indices. Since within the parts we can still assume the indices to be ordered, we do not need to perform permutations among $i_1, \ldots, i_k$ or $i_{k+1}, \ldots, i_n$. So the set of permutations we need is given by the quotient $S_n/(S_k \times S_{n-k})$, i.e. the group of all permutations of n elements modulo the product of the two subgroups corresponding to permutations within the parts. This quotient is not a group in general and its elements are strictly speaking not permutations but equivalence classes of permutations. But by requiring the indices to be in ascending order within each part we can define a canonical set of representatives. The number of elements of the quotient is $\binom{n}{k} = \frac{n!}{k!(n-k)!}$, compared to $n!$ for the full permutation group. But now we have an additional complication. Since we are within one particular block, each index is restricted to a certain subset of all indices. Although this restriction does not only consist of the irreducible representation, we refer to it briefly as the symmetry type of the index. Since the symmetry types are also fixed for the matrix, we have to restrict the permutations to indices with the same symmetry type. If all indices in the considered group have the same type, this does of course not change anything. In the other extreme case, where all types are different, we do not need any permutation at all. In all other cases we have to first determine the permutations for each subset of indices with the same symmetry type. That means we take the corresponding subgroup of S_n and form a quotient set as above. This is only nontrivial if there are indices of this type in both parts, i.e. contracted and non-contracted. The total set of permutations is then the direct product of the permutation sets for all symmetry types occurring in the given index group. The direct product here is to be understood in the sense that every factor acts on a different set of indices, and these sets form a partition of the index group, so the product is a permutation of all indices in the group. This set of permutations

determines the structure of the table in which the addresses have to be stored.

Precalculated address tables are only useful if they can be generated and evaluated efficiently. The initialization of such a table proceeds in two steps. First, a nested structure of pointers is set up. It consists of several "levels", where each level corresponds to an index position and contains as many entries as there are possible index values at this position (given the values at the positions before). For all but the last index, each entry is a pointer to the first entry of the next level. At the innermost level, the addresses have to be stored. For this we have to do two things:

1. Loop over all index combinations (for this group) occurring in the tensor and increment the address accordingly.

2. Loop over the corresponding set of permutations (determined as described above), store the current address and the parity of the permutation in the table entry belonging to the permuted set of indices.

Both could be done by generic iterators, but this would not be efficient. Writing the loops and the permutations out explicitly requires to have a separate initialization function for each type of table. The loop parameters (start and end indices) for the iteration over the tensor can also vary, but they are given as arguments to the function. Although for small index groups there are not many different table types, their number grows rapidly with increasing size of the index groups. And since we want to be able to deal with high excitations, we did not want to restrict the number of indices in one group from the beginning. Hence it would be impractical to write all necessary functions by hand, so we decided to use automatic code generation for these functions. This offers a convenient way of writing a large number of functions with a common interface, which have the same general structure but differ in some details, like the number of loops.

The quite complicated step of finding the correct set of permutations is carried out during the code generation, i.e. before running the actual main program. The problem which remains at runtime of the latter is to choose the correct function and to call it with the right arguments.

5.3.5. Further Optimizations

Most of the time during the iteration only the last indices change. Since our generic iterators are not of optimal efficiency, it is beneficial to code one or several of the innermost loops explicitly. Here we have to differentiate several cases, depending on which indices are coupled by inequality relations. We decided to take at most three explicit loops to keep the number of cases manageable. While the index structure is handled by defining different functions, the index ranges are given to these functions as arguments.

Although the generation of the addressing tables and related structures is already quite efficient, it still takes a non-negligible amount of time. But it is not necessary to generate new tables for every contraction. Each table depends only on the structure of one index group, and the same structures usually appear several times in a sequence of contractions. Therefore it is useful to store all generated tables for later reuse. Although a single table is rather small, the collection of all tables can become large. To limit the amount of memory it takes, we implemented a cache structure which deletes entries if a predefined size is exceeded.

Since a coupled-cluster calculation requires an iterative procedure where the same contractions are performed in each iteration (only with different entries), there are even more possibilities for reuse. Not only addressing tables, but also other administrative structures can be stored during the first iteration and reused later.

5.3.6. Performance Analysis

So far we did only some preliminary tests to check the performance of our program. In table 5.4 we compare the timings for a single CC iteration published by Kállay with the time our program needs to carry out all the contractions needed for this iteration on similar machines. Since we could not use exactly the same ones, we scaled our times appropriately. We observe that the relative performance of our program is better for

Table 5.4.: Calculation times for a single CC iteration

			CPU time/min	
system	method	basis (size)	ours	ref.
H_2O	CCSDTQ	cc-pVDZ (24)	5	6.8[a]
Butadiene	CCSDT	cc-pVTZ (204)	1000	3024[b]

[a] taken from ref. [42], calculation done on Athlon 800 MHz
[b] taken from www.mrcc.hu, calculation done on Pentium IV 3.4 GHz

the larger calculation, as can be expected comparing the performance of matrix–matrix and matrix–vector multiplication (see table 5.1).

In addition, we did a CCSD calculation on the system $4H_2O$ with a cc-pVDZ basis (96 basis functions). For this calculation (11 iterations), the MOLPRO program [108] needs 9.58 s, while the corresponding contractions in our program take 80.46 s. These figures can not be compared directly, since MOLPRO uses spatial orbitals and spin-averaged excitation operators (see section 2.2.2.4) and thus has a much lower number of parameters (220288 amplitudes compared to 920248, which gives a ratio of about 4.18). Since the time for a CCSD calculation should scale with the number of amplitudes to the power of $\frac{3}{2}$, we multiplied the time for the MOLPRO calculation by the

Table 5.5.: Operation statistics for conversions

	tensor → matrix	matrix → tensor
number of operations	2.936e+09	1.781e+09
CPU time/s	14.13	8.07
operations per s	2.079e+08	2.208e+08

amplitude ratio to the power of $\frac{3}{2}$ to estimate the time the calculation with spin orbitals would have taken. The result is 81.56 s, which is of the same order of magnitude as our observed time.

For this example, we also analyzed the performance of the contraction part in some more detail. Table 5.5 shows the CPU time in relation to the number of operations for the rearranging steps (conversion from tensor to matrix and vice versa). The CPU time for the matrix multiplication (`dgemm`) is 42.73 s, which makes up 53.1% of total CPU time. This is a rather good value, taking into account that in a previous version of the implementation the total computation time was completely dominated by the conversions. For larger matrices (i.e. larger molecules or basis sets) the proportion for the matrix multiplication is expected to increase, since this is an $\mathcal{O}(N^3)$ step while the conversion scales only as $\mathcal{O}(N^2)$.

Considering the performance in terms of floating point operations (FLOPS) per second, we have 8515.61 MFLOPS/s for the `dgemm` part and 4522.25 MFLOPS/s in total. This is to be compared to the `dgemm` peak performance of 11 GFLOPS/s. Thus we have reached – for this system – an overall efficiency of 41% of this peak performance. This is a very reasonable result for the implementation of a generic tensor contraction for antisymmetric, externally restricted tensors.

6. Conclusion and Outlook

6.1. Conclusion

We have presented a new implementation of the coupled-cluster method achieving both flexibility and efficiency. Flexibility here means on the one hand that there is no conceptual limitation of the excitation level included in the cluster operator, and on the other hand, that not only standard coupled-cluster expressions can be evaluated by our machinery. This is achieved by an automatized derivation of working equations and by a generic tensor contraction procedure. Our program could be immediately applied to methods like CI or CEPA or variants of CC, and with some modifications also to other methods which can be formulated in terms of second quantization. The program has a modular structure making it relatively easy to exchange method-specific parts.

The first part – the formula generation – is very general. A combination of an algebraic operator evaluation – using an extension of Wick's theorem – with a graph-based simplification algorithm makes it possible to evaluate also complicated second-quantized expressions in a reasonable time.

The latter part – the tensor contraction – is central for an efficient implementation. For a program aiming at generality it is of course not possible to optimize the contraction procedure for every type of contraction separately, so a generic contraction function is necessary. Our approach to reduce the contraction step to a sequence of matrix multiplications has several advantages: First, the (time-critical) multiplication step is independent of the structure of the involved tensors, all are brought to the same basic form. Second, for this step we can use a standard library which is highly optimized and adapted to the actual processor architecture. Finally, by performing conversion and matrix multiplication blockwise, we can effectively exploit symmetry properties of the tensors. The price for this is a rather complex rearrangement step, but by an optimized addressing of tensor entries and several other improvements we also achieved a high efficiency in this part. For some cases (e.g. calculations with high excitations and small basis set) it is still the time-determining step, but for larger basis sets the matrix multiplication dominates, and so the relative performance (compared to programs which do not use matrix multiplications) is better in the latter case.

6.2. Outlook

There are several directions in which the present implementation can be extended or improved. In addition, we want to indicate some interesting possible applications of our program.

6.2.1. Optimization

While we assume that the optimization potential within the tensor contraction is largely exhausted, the step before – which determines the contractions to be carried out – offers several possibilities for further improvements. Besides the factorization briefly mentioned in chapter 3, which determines the order of the factors in one term, it is also possible to factor out common factors from different terms. By such transformations, many expensive multiplications can be saved, while the cost of possible extra additions is negligible. The combination of both types of factorizations leads to a very complicated optimization problem, but even a non-optimal solution could lead to significant savings in the computational time for the subsequent calculation.

In addition, when the set of contractions to be performed has been determined, some further manipulations can be applied. For example, the amount of memory needed during the calculation may depend on the order in which certain contractions are done. Moreover, some of the rearrangement steps could be avoided if the indices which determine the structure of an intermediate (contraction result) are arranged in a way which is suitable for the next contraction using this as a factor.

In the long term, also a parallelization of the implementation, in particular the tensor contraction, would be interesting.

6.2.2. Generalizations

Our next goal is of course to implement multi-reference methods, in particular SR-MRCC and MRexpT, which requires some additional considerations. Being able to treat high excitations is a necessary prerequisite, but now these excitations (more precisely: those which are higher than the base excitation level of the respective calculation) are subject to constraints, which have to be handled appropriately. In the case of SRMRCC the overall procedure is the same as in the single-reference case, and the additional excitations appear in the cluster operator as well as in the projections. That means that constraints have to be taken into account for summed and external indices. For MRexpT this is a bit different. Since the cluster operators are reference specific, their maximal excitation level is always equal to the base excitation level. The projections, however, are global, so here the excitations, when given with respect to one particular reference, can be higher than the base excitation level.

For applications to closed-shell systems it would be advantageous to have an implementation based on spin-averaged excitation operators. In this case the symmetry properties of integrals and amplitudes are different, but the basic machinery of tensor contractions could be used as before.

6.2.3. Applications

For multi-reference methods there are of course many possible applications. But it is also interesting to do SRCC calculations with high excitations, for example as benchmarks for cheaper methods. By combining our program with the implementation of the incremental scheme developed in our group [109] also calculations on larger molecules could be made feasible.

Besides the calculation of correlation energies, we plan to apply our program for the calculation of (approximations to) the variance of single- and multi-reference wave functions. In particular it would be interesting to investigate to what extent approximate expressions for the variance can serve as a measure for the quality of a wave function.

A. Proof of the BCH Formula

The proof below follows [110], but is rewritten in terms of operators (instead of elements on an arbitrary Lie algebra). First we need some notation: Let $\hat{A}$ be an operator, then we define two new operators $\hat{L}_{\hat{A}}$ and $\hat{R}_{\hat{A}}$ as the left and right multiplication with $\hat{A}$, respectively. The commutator of two operators can then be expressed as follows:

$$\big[\hat{A},\hat{B}\big] = \hat{A}\hat{B} - \hat{B}\hat{A} = \hat{R}_{\hat{B}}\hat{A} - \hat{L}_{\hat{B}}\hat{A}.$$

Now we can state the claim:

$$\exp\big(-\hat{B}\big)\hat{A}\exp\big(\hat{B}\big) = \sum_{n=0}^{\infty}\frac{1}{n!}\big(\hat{R}_{\hat{B}} - \hat{L}_{\hat{B}}\big)^n\hat{A} \tag{A.1}$$

Proof: We start from the right hand side and invoke the binomial theorem (this is possible since the operators $\hat{L}_{\hat{B}}$ and $\hat{R}_{\hat{B}}$ commute):

$$\begin{aligned}
\sum_{n=0}^{\infty}\frac{1}{n!}\big(\hat{R}_{\hat{B}} - \hat{L}_{\hat{B}}\big)^n\hat{A} &= \sum_{n=0}^{\infty}\frac{1}{n!}\sum_{k=0}^{n}\frac{n!}{k!(n-k)!}\hat{R}_{\hat{B}}^k\big(-\hat{L}_{\hat{B}}\big)^{n-k}\hat{A} \\
&= \sum_{n=0}^{\infty}\sum_{k+l=n}\frac{1}{k!l!}\big(-\hat{B}\big)^l\hat{A}\hat{B}^k \\
&= \left(\sum_{l=0}^{\infty}\frac{1}{l!}(-\hat{B})^l\right)\hat{A}\left(\sum_{k=0}^{\infty}\frac{1}{k!}\hat{B}^k\right) \\
&= \exp\big(-\hat{B}\big)\hat{A}\exp\big(\hat{B}\big)
\end{aligned}$$

The right hand side of equation (A.1) can be written more explicitly in terms of commutators, yielding the last expression in equation (2.32):

$$\sum_{n=0}^{\infty}\frac{1}{n!}\big(\hat{R}_{\hat{B}} - \hat{L}_{\hat{B}}\big)^n\hat{A} = \hat{A} + \big[\hat{A},\hat{B}\big] + \tfrac{1}{2}\big[\big[\hat{A},\hat{B}\big],\hat{B}\big] + \tfrac{1}{3!}\big[\big[\big[\hat{A},\hat{B}\big],\hat{B}\big],\hat{B}\big] + \ldots$$

B. Example Program

We show here the code of an example program which does a CCSD calculation for H_2O.

```
 1 #include "SQCompoundOperators/CompoundOperator_Product_Sum.H"
 2 #include "SQCompoundOperators/CCOperators.H"
 3 #include "SQStaticTerms/TensorSymbols_Sum.H"
 4 #include "SQStaticTerms/ACOperator_Product_NOP_FV.H"
 5 #include "SQTermSimplification/TermGraph_Sum.H"
 6 #include "SQIndex/SQIndex.H"
 7 #include "SQFastWick/expandFastWick.H"
 8 #include "Factorization3/Graph_Setup.H"
 9 #include "Factorization3/Graph_Factorize.H"
10 #include "ContractionProgram/Program.H"
11
12 #include "InterfaceStoney/StoneyFile.H"
13 #include "InterfaceStoney/
      OneElectronOperatorRepresentation_const_iterator.H"
14 #include "InterfaceStoney/
      TwoElectronOperatorRepresentation_const_iterator.H"
15 #include "PointGroupSymmetry/PointGroup.H"
16 #include "SpinSpatialMolecularOrbital/
      SpinSpatialMOTable_OAVIrrepSzIdx.H"
17 #include "SpinSpatialMolecularOrbital/SpinSpatialMO_IrrepSzIdx.H"
18 #include "OrbitalProduct/SlaterDeterminant.H"
19 #include "Tensor/IntegralInitializer.H"
20
21 #include "Tensor2/TensorIndexInterpretation_SE_AC.H"
22 #include "Tensor2/SymbolicTensor.H"
23 #include "Tensor2/BlockContraction.H"
24 #include "Tensor2/SpinOrbital_OAV_TensorInfo.H"
25 #include "Tensor2/GlobalCaching.H"
26 #include "EvaluationData3.H"
27 #include "ProgramEvaluator.H"
28
29 #include <iostream>
30 #include <sstream>
31 #include <vector>
32
```

```
33 using namespace std;
34 using namespace QOL::SQFastWick;
35 using namespace QOL::SQStaticTerms;
36 using namespace QOL::SQTermSimplification;
37 using namespace QOL::SQCompoundOperators;
38 using namespace QOL::SQCompoundOperators::CCOperators;
39 using namespace QOL::Factorization3;
40 using namespace QOL::Tensor2;
41 using namespace QOL::InterfaceStoney;
42 using namespace QOL::PointGroupSymmetry;
43 using namespace QOL::SpinSpatialMolecularOrbital;
44 using namespace QOL::ContractionProgram;
45 using namespace QOL::SQEquationDrivenFactorizedContraction;
46 using namespace QOL::SQEquationDrivenFactorizedContraction3;
47 using QOL::OrbitalProduct::SlaterDeterminant;
48
49 typedef QOL::Tensor2::GlobalCaching _GlobalCaching;
50
51 int main()
52 {
53 //basic definitions
54 typedef QOL::Tensor::TensorStructure_HI<
      SpinSpatialMOTable_OAVIrrepSzIdx,4> TensorStructure;
55 typedef QOL::Tensor::TensorStructure_HI<
      SpinSpatialMOTable_IrrepSzIdx,3> TensorStructure_Int;
56 typedef TensorStructure_CS<SpatialMOTable_IrrepIdx,2>
      TensorStructure_spatial;
57
58 const int nIter=50;     //max. number of iterations
59 const int clusterLevel = 2;
60 const int projectionLevel = 2;
61 const double t_conv=1e-10;  //convergence threshold for amplitudes
      (2-norm)
62
63 //preparation I (equations)
64
65 //formula generation
66 CompoundOperator_Expression TT;
67   for ( int i=1 ; i<=clusterLevel ; ++i )
68     TT += T(i);
69 CompoundOperator_Expression HN = FN + VN;
70 vector<pair<TermGraph_Sum, int> > targets;
71   for ( int i=0 ; i<=projectionLevel ; ++i )
72   {
73   CompoundOperator_Expression expr(FV(A(-i)*exp(-TT)*HN*exp(TT)));
           // CC
```

```
 74    CompoundOperator_Product_Sum  flatExpr(expr);
 75    TensorSymbols_ACOperators_Sum<SQIndex, ACOperator_Product_NOP_FV<
          SQIndex> > nopfs(flatExpr);
 76    TensorSymbols_Kroneckers_Sum<SQIndex, true> expanded(
          expandFastWick(nopfs));
 77    TermGraph_Sum tgs(expanded);
 78    TensorSymbols_Sum<SQIndex, true> simplified(tgs);
 79      targets.push_back(make_pair(tgs, i));
 80    }
 81
 82  //factorization
 83  TensorIndexInterpretation_SE_AC tII;
 84  Graph_Setup g1(targets, tII);
 85  Graph_Factorize g2(g1);
 86    g2.setSigns();
 87    g2.collectExternal();
 88    g2.optimize();
 89  //generate contraction program
 90  Program p(g2, tII);
 91
 92  //-------------------------------------
 93  //preparation II (integrals etc.)
 94
 95  StoneyFile  stoney("fort.31");  //contains orbital information and
        integrals
 96  PointGroup<abelian> pg(stoney.pointGroup());
 97  //tables
 98  SpatialMOTable_IrrepIdx tab_spatial(stoney.orbitalsPerIrrep());
 99  SpinSpatialMOTable_IrrepSzIdx tab0(stoney.orbitalsPerIrrep());
100
101  string  s("(0a- 0a+ 1a- 1a+ 2a- 2a+ 0b- 0b+ 0c- 0c+)");   //
        reference determinant for H2O
102  istringstream iss(s);
103  MOBasisInfo<SpinSpatialMO_IrrepSzIdx> info0(tab0, pg);
104  SlaterDeterminant<SpinSpatialMO_IrrepSzIdx> ref(iss, info0);
105  SpinSpatialMOTable_OAVIrrepSzIdx  table(tab0, ref.v());
106
107  //IndexRanges
108  TensorStructure_spatial::IndexRange ir_spatial(tab_spatial, pg);
109  TensorStructure_Int::IndexRange ir0(tab0, pg);
110  TensorStructure::IndexRange ir(table, pg);
111  //read integrals
112  IntegralInitializer<OneElectronOperatorRepresentation_const_iterator
        > init1(stoney,&tab_spatial);
113  IntegralInitializer<TwoElectronOperatorRepresentation_const_iterator
        > init2(stoney,&tab_spatial);
```

```
114
115 TensorStructure_spatial::SymbolicTensor sti1("a;b",true);
116 TensorStructure_spatial::SymbolicTensor sti2("ab;cd",true);
117
118 QOL::Tensor::Tensor<double,TensorStructure_spatial>  oneElInt(sti1,&
       ir_spatial,init1);
119 QOL::Tensor::Tensor<double,TensorStructure_spatial>  twoElInt(sti2,&
       ir_spatial,init2);
120 OneElectronIntegrals<double,TensorStructure_Int> oei(&ir0,oneElInt);
121 AntisymmetricTwoElectronIntegrals<double,TensorStructure_Int> atei(&
       ir0,twoElInt);
122
123 //old integral containers
124 OneElectronIntegralContainer<double,TensorStructure> oeic(&ir,oei,
       atei);
125 TwoElectronIntegralContainer<double,TensorStructure> teic(&ir,atei);
126 //new integral container
127 SpinOrbital_OAV_TensorInfo info(table, &pg, TensorIndexSections());
128 IntegralContainer<SpinOrbital_OAV_TensorInfo, double> ic(tII, info,
       oeic, teic);
129 //---------------------------------
130 //Evaluation
131
132 ZeroInitializer<double> zi;
133 EvaluationData3<SpinOrbital_OAV_TensorInfo, double> ed(ic, tII, info
       , clusterLevel, projectionLevel, zi);
134
135 const size_t  maxStorage1 = 300*(1 << 20);
136 const size_t  maxStorage2 = 300*(1 << 20);
137 _GlobalCaching  globalCaching(table.table(), _GlobalCaching::
       DefaultCaching, maxStorage1, maxStorage2);
138 PerformanceStatistics performanceStatistics;
139 BlockContraction<SpinOrbital_OAV_TensorInfo> bc(globalCaching,
       performanceStatistics, true);
140 ProgramEvaluator<SpinOrbital_OAV_TensorInfo,double> pEv(ed, bc);
141
142 double  residualNorm2 = 0;
143   for ( int n=0 ; n<nIter ; ++n ) //iteration
144   {
145     pEv.calcResiduals(p);
146     pEv.updateAmplitudes();
147   //calc residual norm
148     residualNorm2 = 0;
149     for ( unsigned int j=1 ; j<ed.residuals.size() ; ++j )
150     {
151     boost::shared_ptr<SuperIterator> sIter(ed.residuals[j].
```

```
        getIterator());
152     int n=0;
153       for ( SuperIteratorIndex sIdx(sIter->begin()) ; sIter->valid(
              sIdx) ; sIter->inc(sIdx), ++n )
154       {
155       double h = ed.residuals[j].p()[n];
156         residualNorm2 += h*h;
157       }
158     }
159     cout << "E_corr: " << ed.residuals[0]  << " residual norm: " <<
           sqrt(residualNorm2)<< endl;
160     ed.clearResidual();
161     if ( n>0 && sqrt(residualNorm2)<t_conv )
162       break;
163   }
164   return 0;
165 }
```

C. Generated Formulas

C.1. CCSDTQ Amplitude Equations

We give here the explicit expressions for the four types of projections (singles to quadruples) onto excited determinants for the CCSDTQ method.

$$\langle\Phi_0|\hat{A}_1^\dagger \exp(\tfrac{-1}{1}\cdot\hat{T}_1+\tfrac{-1}{1}\cdot\hat{T}_2+\tfrac{-1}{1}\cdot\hat{T}_3+\tfrac{-1}{1}\cdot\hat{T}_4)(\hat{F}+\hat{V})\exp(\hat{T}_1+\hat{T}_2+\hat{T}_3+\hat{T}_4)|\Phi_0\rangle$$
$$= [f_{\overline{A}}^{\overline{I}}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[f_{\overline{i}}^{\overline{I}}t_{\overline{i}}^{\overline{A}}]_{\mathcal{A}} + [f_{\overline{a}}^{\overline{A}}t_{\overline{I}}^{\overline{a}}]_{\mathcal{A}} + [f_{\overline{a}}^{\overline{i}}t_{\overline{I}\overline{i}}^{\overline{A}\overline{a}}]_{\mathcal{A}} + [t_{\overline{i}}^{\overline{a}}v_{\overline{A}\overline{i}}^{\overline{I}\overline{a}}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_{\overline{I}\overline{i}}^{\overline{a}\overline{b}}v_{\overline{a}\overline{b}}^{\overline{A}\overline{i}}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_{\overline{i}\overline{j}}^{\overline{A}\overline{a}}v_{\overline{i}\overline{j}}^{\overline{I}\overline{a}}]_{\mathcal{A}}$$
$$+\tfrac{1}{4}\cdot[t_{\overline{I}\overline{i}\overline{j}}^{\overline{A}\overline{a}\overline{b}}v_{\overline{a}\overline{b}}^{\overline{i}\overline{j}}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[f_{\overline{a}}^{\overline{i}}t_{\overline{I}}^{\overline{a}}t_{\overline{i}}^{\overline{A}}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_{\overline{I}}^{\overline{b}}t_{\overline{i}}^{\overline{a}}v_{\overline{a}\overline{b}}^{\overline{A}\overline{i}}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_{\overline{I}}^{\overline{b}}t_{\overline{i}\overline{j}}^{\overline{A}\overline{a}}v_{\overline{a}\overline{b}}^{\overline{i}\overline{j}}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_{\overline{i}}^{\overline{A}}t_{\overline{j}}^{\overline{a}}v_{\overline{i}\overline{j}}^{\overline{I}\overline{a}}]_{\mathcal{A}}$$
$$+\tfrac{-1}{2}\cdot[t_{\overline{i}}^{\overline{A}}t_{\overline{I}\overline{j}}^{\overline{a}\overline{b}}v_{\overline{a}\overline{b}}^{\overline{i}\overline{j}}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_{\overline{i}}^{\overline{b}}t_{\overline{I}\overline{j}}^{\overline{A}\overline{a}}v_{\overline{a}\overline{b}}^{\overline{i}\overline{j}}]_{\mathcal{A}} + [t_{\overline{I}}^{\overline{b}}t_{\overline{i}}^{\overline{A}}t_{\overline{j}}^{\overline{a}}v_{\overline{a}\overline{b}}^{\overline{i}\overline{j}}]_{\mathcal{A}}$$

$$\langle\Phi_0|\hat{A}_2^\dagger \exp(\tfrac{-1}{1}\cdot\hat{T}_1+\tfrac{-1}{1}\cdot\hat{T}_2+\tfrac{-1}{1}\cdot\hat{T}_3+\tfrac{-1}{1}\cdot\hat{T}_4)(\hat{F}+\hat{V})\exp(\hat{T}_1+\hat{T}_2+\hat{T}_3+\hat{T}_4)|\Phi_0\rangle$$
$$= [v_{\overline{A}\overline{B}}^{\overline{I}\overline{J}}]_{\mathcal{A}} + [f_{\overline{i}}^{\overline{I}}t_{\overline{J}\overline{i}}^{\overline{A}\overline{B}}]_{\mathcal{A}} + [f_{\overline{a}}^{\overline{B}}t_{\overline{I}\overline{J}}^{\overline{A}\overline{a}}]_{\mathcal{A}} + [f_{\overline{a}}^{\overline{i}}t_{\overline{I}\overline{J}\overline{i}}^{\overline{A}\overline{B}\overline{a}}]_{\mathcal{A}} + [t_{\overline{J}}^{\overline{a}}v_{\overline{A}\overline{B}}^{\overline{I}\overline{a}}]_{\mathcal{A}} + [t_{\overline{i}}^{\overline{A}}v_{\overline{B}\overline{i}}^{\overline{I}\overline{J}}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_{\overline{I}\overline{J}}^{\overline{a}\overline{b}}v_{\overline{a}\overline{b}}^{\overline{A}\overline{B}}]_{\mathcal{A}}$$
$$+\tfrac{-1}{1}\cdot[t_{\overline{J}\overline{i}}^{\overline{A}\overline{a}}v_{\overline{B}\overline{i}}^{\overline{I}\overline{a}}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_{\overline{i}\overline{j}}^{\overline{A}\overline{B}}v_{\overline{i}\overline{j}}^{\overline{I}\overline{J}}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_{\overline{I}\overline{J}\overline{i}}^{\overline{A}\overline{a}\overline{b}}v_{\overline{a}\overline{b}}^{\overline{B}\overline{i}}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_{\overline{J}\overline{i}\overline{j}}^{\overline{A}\overline{B}\overline{a}}v_{\overline{i}\overline{j}}^{\overline{I}\overline{a}}]_{\mathcal{A}} + \tfrac{1}{4}\cdot[t_{\overline{I}\overline{J}\overline{i}\overline{j}}^{\overline{A}\overline{B}\overline{a}\overline{b}}v_{\overline{a}\overline{b}}^{\overline{i}\overline{j}}]_{\mathcal{A}}$$
$$+[f_{\overline{a}}^{\overline{i}}t_{\overline{I}}^{\overline{a}}t_{\overline{J}\overline{i}}^{\overline{A}\overline{B}}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[f_{\overline{a}}^{\overline{i}}t_{\overline{i}}^{\overline{B}}t_{\overline{I}\overline{J}}^{\overline{A}\overline{a}}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_{\overline{I}}^{\overline{b}}t_{\overline{J}}^{\overline{a}}v_{\overline{a}\overline{b}}^{\overline{A}\overline{B}}]_{\mathcal{A}} + [t_{\overline{I}}^{\overline{b}}t_{\overline{J}\overline{i}}^{\overline{A}\overline{a}}v_{\overline{a}\overline{b}}^{\overline{B}\overline{i}}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_{\overline{I}}^{\overline{b}}t_{\overline{J}\overline{i}\overline{j}}^{\overline{A}\overline{B}\overline{a}}v_{\overline{a}\overline{b}}^{\overline{i}\overline{j}}]_{\mathcal{A}}$$
$$+[t_{\overline{J}}^{\overline{a}}t_{\overline{i}}^{\overline{A}}v_{\overline{B}\overline{i}}^{\overline{I}\overline{a}}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_{\overline{J}}^{\overline{a}}t_{\overline{i}\overline{j}}^{\overline{A}\overline{B}}v_{\overline{i}\overline{j}}^{\overline{I}\overline{a}}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_{\overline{i}}^{\overline{A}}t_{\overline{I}\overline{J}}^{\overline{a}\overline{b}}v_{\overline{a}\overline{b}}^{\overline{B}\overline{i}}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_{\overline{i}}^{\overline{B}}t_{\overline{j}}^{\overline{A}}v_{\overline{i}\overline{j}}^{\overline{I}\overline{J}}]_{\mathcal{A}} + [t_{\overline{i}}^{\overline{B}}t_{\overline{J}\overline{j}}^{\overline{A}\overline{a}}v_{\overline{i}\overline{j}}^{\overline{I}\overline{a}}]_{\mathcal{A}}$$
$$+\tfrac{-1}{2}\cdot[t_{\overline{i}}^{\overline{B}}t_{\overline{I}\overline{J}\overline{j}}^{\overline{A}\overline{a}\overline{b}}v_{\overline{a}\overline{b}}^{\overline{i}\overline{j}}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_{\overline{i}}^{\overline{a}}t_{\overline{J}\overline{j}}^{\overline{A}\overline{B}}v_{\overline{i}\overline{j}}^{\overline{I}\overline{a}}]_{\mathcal{A}} + [t_{\overline{i}}^{\overline{b}}t_{\overline{I}\overline{J}}^{\overline{A}\overline{a}}v_{\overline{a}\overline{b}}^{\overline{B}\overline{i}}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_{\overline{i}}^{\overline{b}}t_{\overline{I}\overline{J}\overline{j}}^{\overline{A}\overline{B}\overline{a}}v_{\overline{a}\overline{b}}^{\overline{i}\overline{j}}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_{\overline{I}\overline{J}}^{\overline{B}\overline{b}}t_{\overline{i}\overline{j}}^{\overline{A}\overline{a}}v_{\overline{a}\overline{b}}^{\overline{i}\overline{j}}]_{\mathcal{A}}$$
$$+\tfrac{1}{4}\cdot[t_{\overline{I}\overline{J}}^{\overline{a}\overline{b}}t_{\overline{i}\overline{j}}^{\overline{A}\overline{B}}v_{\overline{a}\overline{b}}^{\overline{i}\overline{j}}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_{\overline{I}\overline{i}}^{\overline{A}\overline{B}}t_{\overline{J}\overline{j}}^{\overline{a}\overline{b}}v_{\overline{a}\overline{b}}^{\overline{i}\overline{j}}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_{\overline{I}\overline{i}}^{\overline{B}\overline{b}}t_{\overline{J}\overline{j}}^{\overline{A}\overline{a}}v_{\overline{a}\overline{b}}^{\overline{i}\overline{j}}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_{\overline{I}}^{\overline{b}}t_{\overline{J}}^{\overline{a}}t_{\overline{i}}^{\overline{A}}v_{\overline{B}\overline{i}}^{\overline{I}\overline{a}}]_{\mathcal{A}}$$
$$+\tfrac{-1}{4}\cdot[t_{\overline{I}}^{\overline{b}}t_{\overline{J}}^{\overline{a}}t_{\overline{i}\overline{j}}^{\overline{A}\overline{B}}v_{\overline{a}\overline{b}}^{\overline{i}\overline{j}}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_{\overline{I}}^{\overline{b}}t_{\overline{i}}^{\overline{B}}t_{\overline{J}\overline{j}}^{\overline{A}\overline{a}}v_{\overline{a}\overline{b}}^{\overline{i}\overline{j}}]_{\mathcal{A}} + [t_{\overline{I}}^{\overline{b}}t_{\overline{i}}^{\overline{a}}t_{\overline{J}\overline{j}}^{\overline{A}\overline{B}}v_{\overline{a}\overline{b}}^{\overline{i}\overline{j}}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_{\overline{J}}^{\overline{a}}t_{\overline{i}}^{\overline{B}}t_{\overline{j}}^{\overline{A}}v_{\overline{i}\overline{j}}^{\overline{I}\overline{a}}]_{\mathcal{A}}$$
$$+\tfrac{-1}{4}\cdot[t_{\overline{i}}^{\overline{B}}t_{\overline{j}}^{\overline{A}}t_{\overline{I}\overline{J}}^{\overline{a}\overline{b}}v_{\overline{a}\overline{b}}^{\overline{i}\overline{j}}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_{\overline{i}}^{\overline{B}}t_{\overline{j}}^{\overline{b}}t_{\overline{I}\overline{J}}^{\overline{A}\overline{a}}v_{\overline{a}\overline{b}}^{\overline{i}\overline{j}}]_{\mathcal{A}} + \tfrac{1}{4}\cdot[t_{\overline{I}}^{\overline{b}}t_{\overline{J}}^{\overline{a}}t_{\overline{i}}^{\overline{B}}t_{\overline{j}}^{\overline{A}}v_{\overline{a}\overline{b}}^{\overline{i}\overline{j}}]_{\mathcal{A}}$$

$$\langle\Phi_0|\hat{A}_3^\dagger \exp(\tfrac{-1}{1}\cdot\hat{T}_1+\tfrac{-1}{1}\cdot\hat{T}_2+\tfrac{-1}{1}\cdot\hat{T}_3+\tfrac{-1}{1}\cdot\hat{T}_4)(\hat{F}+\hat{V})\exp(\hat{T}_1+\hat{T}_2+\hat{T}_3+\hat{T}_4)|\Phi_0\rangle$$
$$= \tfrac{-1}{1}\cdot[f_{\overline{i}}^{\overline{I}}t_{\overline{J}\overline{K}\overline{i}}^{\overline{A}\overline{B}\overline{C}}]_{\mathcal{A}} + [f_{\overline{a}}^{\overline{C}}t_{\overline{I}\overline{J}\overline{K}}^{\overline{A}\overline{B}\overline{a}}]_{\mathcal{A}} + [f_{\overline{a}}^{\overline{i}}t_{\overline{I}\overline{J}\overline{K}\overline{i}}^{\overline{A}\overline{B}\overline{C}\overline{a}}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_{\overline{J}\overline{K}}^{\overline{A}\overline{a}}v_{\overline{B}\overline{C}}^{\overline{I}\overline{a}}]_{\mathcal{A}} + [t_{\overline{K}\overline{i}}^{\overline{A}\overline{B}}v_{\overline{C}\overline{i}}^{\overline{I}\overline{J}}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_{\overline{I}\overline{J}\overline{K}}^{\overline{A}\overline{a}\overline{b}}v_{\overline{a}\overline{b}}^{\overline{B}\overline{C}}]_{\mathcal{A}}$$
$$+[t_{\overline{J}\overline{K}\overline{i}}^{\overline{A}\overline{B}\overline{a}}v_{\overline{C}\overline{i}}^{\overline{I}\overline{a}}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_{\overline{K}\overline{i}\overline{j}}^{\overline{A}\overline{B}\overline{C}}v_{\overline{i}\overline{j}}^{\overline{I}\overline{J}}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_{\overline{I}\overline{J}\overline{K}\overline{i}}^{\overline{A}\overline{B}\overline{a}\overline{b}}v_{\overline{a}\overline{b}}^{\overline{C}\overline{i}}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_{\overline{J}\overline{K}\overline{i}\overline{j}}^{\overline{A}\overline{B}\overline{C}\overline{a}}v_{\overline{i}\overline{j}}^{\overline{I}\overline{a}}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[f_{\overline{a}}^{\overline{i}}t_{\overline{I}}^{\overline{a}}t_{\overline{J}\overline{K}\overline{i}}^{\overline{A}\overline{B}\overline{C}}]_{\mathcal{A}}$$
$$+\tfrac{-1}{1}\cdot[f_{\overline{a}}^{\overline{i}}t_{\overline{i}}^{\overline{C}}t_{\overline{I}\overline{J}\overline{K}}^{\overline{A}\overline{B}\overline{a}}]_{\mathcal{A}} + [f_{\overline{a}}^{\overline{i}}t_{\overline{I}\overline{J}}^{\overline{C}\overline{a}}t_{\overline{K}\overline{i}}^{\overline{A}\overline{B}}]_{\mathcal{A}} + [t_{\overline{I}}^{\overline{b}}t_{\overline{J}\overline{K}}^{\overline{A}\overline{a}}v_{\overline{a}\overline{b}}^{\overline{B}\overline{C}}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_{\overline{I}}^{\overline{b}}t_{\overline{J}\overline{K}\overline{i}}^{\overline{A}\overline{B}\overline{a}}v_{\overline{a}\overline{b}}^{\overline{C}\overline{i}}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_{\overline{I}}^{\overline{b}}t_{\overline{J}\overline{K}\overline{i}\overline{j}}^{\overline{A}\overline{B}\overline{C}\overline{a}}v_{\overline{a}\overline{b}}^{\overline{i}\overline{j}}]_{\mathcal{A}}$$
$$+[t_{\overline{J}}^{\overline{a}}t_{\overline{K}\overline{i}}^{\overline{A}\overline{B}}v_{\overline{C}\overline{i}}^{\overline{I}\overline{a}}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_{\overline{J}}^{\overline{a}}t_{\overline{K}\overline{i}\overline{j}}^{\overline{A}\overline{B}\overline{C}}v_{\overline{i}\overline{j}}^{\overline{I}\overline{a}}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_{\overline{i}}^{\overline{B}}t_{\overline{J}\overline{K}}^{\overline{A}\overline{a}}v_{\overline{C}\overline{i}}^{\overline{I}\overline{a}}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_{\overline{i}}^{\overline{B}}t_{\overline{I}\overline{J}\overline{K}}^{\overline{A}\overline{a}\overline{b}}v_{\overline{a}\overline{b}}^{\overline{C}\overline{i}}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_{\overline{i}}^{\overline{C}}t_{\overline{K}\overline{j}}^{\overline{A}\overline{B}}v_{\overline{i}\overline{j}}^{\overline{I}\overline{J}}]_{\mathcal{A}}$$
$$+\tfrac{-1}{1}\cdot[t_{\overline{i}}^{\overline{C}}t_{\overline{J}\overline{K}\overline{j}}^{\overline{A}\overline{B}\overline{a}}v_{\overline{i}\overline{j}}^{\overline{I}\overline{a}}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_{\overline{i}}^{\overline{C}}t_{\overline{I}\overline{J}\overline{K}\overline{j}}^{\overline{A}\overline{B}\overline{a}\overline{b}}v_{\overline{a}\overline{b}}^{\overline{i}\overline{j}}]_{\mathcal{A}} + [t_{\overline{i}}^{\overline{a}}t_{\overline{J}\overline{K}\overline{j}}^{\overline{A}\overline{B}\overline{C}}v_{\overline{i}\overline{j}}^{\overline{I}\overline{a}}]_{\mathcal{A}} + [t_{\overline{i}}^{\overline{b}}t_{\overline{I}\overline{J}\overline{K}}^{\overline{A}\overline{B}\overline{a}}v_{\overline{a}\overline{b}}^{\overline{C}\overline{i}}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_{\overline{i}}^{\overline{b}}t_{\overline{I}\overline{J}\overline{K}\overline{j}}^{\overline{A}\overline{B}\overline{C}\overline{a}}v_{\overline{a}\overline{b}}^{\overline{i}\overline{j}}]_{\mathcal{A}}$$

$$+\tfrac{-1}{1}\cdot[t_{IJ}^{Bb}t_{Ki}^{Aa}v_{\underline{ab}}^{\underline{Ci}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[t_{IJ}^{Cb}t_{Kij}^{ABa}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[t_{IJ}^{ab}t_{Ki}^{AB}v_{\underline{ab}}^{\underline{Ci}}]_{\mathcal{A}}+\tfrac{1}{4}\cdot[t_{IJ}^{ab}t_{Kij}^{ABC}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}$$
$$+\tfrac{1}{2}\cdot[t_{Ii}^{BC}t_{JKj}^{Aab}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[t_{Ii}^{Cb}t_{JKj}^{ABa}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[t_{Ii}^{ab}t_{JKj}^{ABC}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[t_{JK}^{Ca}t_{ij}^{AB}v_{\underline{ij}}^{\underline{Ia}}]_{\mathcal{A}}$$
$$+[t_{Ji}^{BC}t_{Kj}^{Aa}v_{\underline{ij}}^{\underline{Ia}}]_{\mathcal{A}}+\tfrac{1}{4}\cdot[t_{ij}^{BC}t_{IJK}^{Aab}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[t_{ij}^{Cb}t_{IJK}^{ABa}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[t_{I}^{b}t_{J}^{a}t_{Ki}^{AB}v_{\underline{ab}}^{\underline{Ci}}]_{\mathcal{A}}$$
$$+\tfrac{-1}{4}\cdot[t_{I}^{b}t_{J}^{a}t_{Kij}^{ABC}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+[t_{I}^{b}t_{i}^{B}t_{JK}^{Aa}v_{\underline{ab}}^{\underline{Ci}}]_{\mathcal{A}}+[t_{I}^{b}t_{i}^{C}t_{JKj}^{ABa}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[t_{I}^{b}t_{i}^{a}t_{JKj}^{ABC}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}$$
$$+\tfrac{1}{2}\cdot[t_{I}^{b}t_{JK}^{Ca}t_{ij}^{AB}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[t_{I}^{b}t_{Ji}^{BC}t_{Kj}^{Aa}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[t_{J}^{a}t_{i}^{C}t_{Kj}^{AB}v_{\underline{ij}}^{\underline{Ia}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[t_{i}^{C}t_{j}^{B}t_{JK}^{Aa}v_{\underline{ij}}^{\underline{Ia}}]_{\mathcal{A}}$$
$$+\tfrac{-1}{4}\cdot[t_{i}^{C}t_{j}^{B}t_{IJK}^{Aab}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[t_{i}^{C}t_{j}^{b}t_{IJK}^{ABa}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+[t_{i}^{C}t_{IJ}^{Bb}t_{Kj}^{Aa}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[t_{i}^{C}t_{IJ}^{ab}t_{Kj}^{AB}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}$$
$$+\tfrac{-1}{1}\cdot[t_{i}^{b}t_{IJ}^{Ca}t_{Kj}^{AB}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[t_{I}^{b}t_{J}^{a}t_{i}^{C}t_{Kj}^{AB}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[t_{I}^{b}t_{i}^{C}t_{j}^{B}t_{JK}^{Aa}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}$$

$$\langle\Phi_0|\hat{A}_4^{\dagger}\exp(\tfrac{-1}{1}\cdot\hat{T}_1+\tfrac{-1}{1}\cdot\hat{T}_2+\tfrac{-1}{1}\cdot\hat{T}_3+\tfrac{-1}{1}\cdot\hat{T}_4)(\hat{F}+\hat{V})\exp(\hat{T}_1+\hat{T}_2+\hat{T}_3+\hat{T}_4)|\Phi_0\rangle$$
$$=[f_{i}^{I}t_{JKLi}^{ABCD}]_{\mathcal{A}}+[f_{a}^{\underline{D}}t_{IJKL}^{ABCa}]_{\mathcal{A}}+[t_{JKL}^{ABa}v_{\underline{CD}}^{\underline{Ia}}]_{\mathcal{A}}+[t_{KLi}^{ABC}v_{\underline{Di}}^{\underline{IJ}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[t_{IJKL}^{ABab}v_{\underline{ab}}^{\underline{CD}}]_{\mathcal{A}}$$
$$+\tfrac{-1}{1}\cdot[t_{JKLi}^{ABCa}v_{\underline{Di}}^{\underline{Ia}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[t_{KLij}^{ABCD}v_{\underline{ij}}^{\underline{IJ}}]_{\mathcal{A}}+[f_{a}^{i}t_{I}^{a}t_{JKLi}^{ABCD}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f_{a}^{i}t_{i}^{D}t_{IJKL}^{ABCa}]_{\mathcal{A}}+[f_{a}^{i}t_{IJ}^{Da}t_{KLi}^{ABC}]_{\mathcal{A}}$$
$$+\tfrac{-1}{1}\cdot[f_{a}^{i}t_{Ii}^{CD}t_{JKL}^{ABa}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[t_{I}^{b}t_{JKL}^{ABa}v_{\underline{ab}}^{\underline{CD}}]_{\mathcal{A}}+[t_{I}^{b}t_{JKLi}^{ABCa}v_{\underline{ab}}^{\underline{Di}}]_{\mathcal{A}}+[t_{J}^{a}t_{KLi}^{ABC}v_{\underline{Di}}^{\underline{Ia}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[t_{J}^{a}t_{KLij}^{ABCD}v_{\underline{ij}}^{\underline{Ia}}]_{\mathcal{A}}$$
$$+[t_{i}^{C}t_{JKL}^{ABa}v_{\underline{Di}}^{\underline{Ia}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[t_{i}^{C}t_{IJKL}^{ABab}v_{\underline{ab}}^{\underline{Di}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[t_{i}^{D}t_{KLj}^{ABC}v_{\underline{ij}}^{\underline{IJ}}]_{\mathcal{A}}+[t_{i}^{D}t_{JKLj}^{ABCa}v_{\underline{ij}}^{\underline{Ia}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[t_{i}^{a}t_{JKLj}^{ABCD}v_{\underline{ij}}^{\underline{Ia}}]_{\mathcal{A}}$$
$$+[t_{i}^{b}t_{IJKL}^{ABCa}v_{\underline{ab}}^{\underline{Di}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[t_{IJ}^{Bb}t_{KL}^{Aa}v_{\underline{ab}}^{\underline{CD}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[t_{IJ}^{Cb}t_{KLi}^{ABa}v_{\underline{ab}}^{\underline{Di}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[t_{IJ}^{Db}t_{KLij}^{ABCa}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}$$
$$+\tfrac{1}{2}\cdot[t_{IJ}^{ab}t_{KLi}^{ABC}v_{\underline{ab}}^{\underline{Di}}]_{\mathcal{A}}+\tfrac{1}{4}\cdot[t_{IJ}^{ab}t_{KLij}^{ABCD}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[t_{Ii}^{BC}t_{JKL}^{Aab}v_{\underline{ab}}^{\underline{Di}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[t_{Ii}^{CD}t_{JKLj}^{ABab}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}$$
$$+[t_{Ii}^{Cb}t_{JKL}^{ABa}v_{\underline{ab}}^{\underline{Di}}]_{\mathcal{A}}+[t_{Ii}^{Db}t_{JKLj}^{ABCa}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[t_{Ii}^{ab}t_{JKLj}^{ABCD}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[t_{JK}^{Ca}t_{Li}^{AB}v_{\underline{Di}}^{\underline{Ia}}]_{\mathcal{A}}$$
$$+\tfrac{1}{2}\cdot[t_{JK}^{Da}t_{Lij}^{ABC}v_{\underline{ij}}^{\underline{Ia}}]_{\mathcal{A}}+[t_{Ji}^{CD}t_{KLj}^{ABa}v_{\underline{ij}}^{\underline{Ia}}]_{\mathcal{A}}+[t_{Ji}^{Da}t_{KLj}^{ABC}v_{\underline{ij}}^{\underline{Ia}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[t_{Ki}^{CD}t_{Lj}^{AB}v_{\underline{ij}}^{\underline{IJ}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[t_{ij}^{CD}t_{JKL}^{ABa}v_{\underline{ij}}^{\underline{Ia}}]_{\mathcal{A}}$$
$$+\tfrac{1}{4}\cdot[t_{ij}^{CD}t_{IJKL}^{ABab}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[t_{ij}^{Db}t_{IJKL}^{ABCa}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[t_{IJK}^{CDb}t_{Lij}^{ABa}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+\tfrac{-1}{4}\cdot[t_{IJK}^{Dab}t_{Lij}^{ABC}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}$$
$$+\tfrac{-1}{2}\cdot[t_{IJi}^{BCD}t_{KLj}^{Aab}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[t_{IJi}^{CDb}t_{KLj}^{ABa}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[t_{I}^{b}t_{J}^{a}t_{KLi}^{ABC}v_{\underline{ab}}^{\underline{Di}}]_{\mathcal{A}}+\tfrac{-1}{4}\cdot[t_{I}^{b}t_{J}^{a}t_{KLij}^{ABCD}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}$$
$$+\tfrac{-1}{1}\cdot[t_{I}^{b}t_{i}^{C}t_{JKL}^{ABa}v_{\underline{ab}}^{\underline{Di}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[t_{I}^{b}t_{i}^{D}t_{JKLj}^{ABCa}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+[t_{I}^{b}t_{i}^{a}t_{JKLj}^{ABCD}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+[t_{I}^{b}t_{JK}^{Ca}t_{Li}^{AB}v_{\underline{ab}}^{\underline{Di}}]_{\mathcal{A}}$$
$$+\tfrac{-1}{2}\cdot[t_{I}^{b}t_{JK}^{Da}t_{Lij}^{ABC}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[t_{I}^{b}t_{Ji}^{CD}t_{KLj}^{ABa}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[t_{I}^{b}t_{Ji}^{Da}t_{KLj}^{ABC}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[t_{I}^{b}t_{ij}^{CD}t_{JKL}^{ABa}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}$$
$$+\tfrac{-1}{1}\cdot[t_{J}^{a}t_{i}^{D}t_{KLj}^{ABC}v_{\underline{ij}}^{\underline{Ia}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[t_{J}^{a}t_{Ki}^{CD}t_{Lj}^{AB}v_{\underline{ij}}^{\underline{Ia}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[t_{i}^{C}t_{IJ}^{Bb}t_{KL}^{Aa}v_{\underline{ab}}^{\underline{Di}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[t_{i}^{D}t_{j}^{C}t_{JKL}^{ABa}v_{\underline{ij}}^{\underline{Ia}}]_{\mathcal{A}}$$
$$+\tfrac{-1}{4}\cdot[t_{i}^{D}t_{j}^{C}t_{IJKL}^{ABab}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[t_{i}^{D}t_{j}^{b}t_{IJKL}^{ABCa}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+[t_{i}^{D}t_{IJ}^{Cb}t_{KLj}^{ABa}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[t_{i}^{D}t_{IJ}^{ab}t_{KLj}^{ABC}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}$$
$$+\tfrac{1}{2}\cdot[t_{i}^{D}t_{Ij}^{BC}t_{JKL}^{Aab}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[t_{i}^{D}t_{Ij}^{Cb}t_{JKL}^{ABa}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+[t_{i}^{D}t_{JK}^{Ca}t_{Lj}^{AB}v_{\underline{ij}}^{\underline{Ia}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[t_{i}^{b}t_{IJ}^{Da}t_{KLj}^{ABC}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}$$
$$+[t_{i}^{b}t_{Ij}^{CD}t_{JKL}^{ABa}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+\tfrac{-1}{4}\cdot[t_{IJ}^{Db}t_{KL}^{Ca}t_{ij}^{AB}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+[t_{IJ}^{Db}t_{Ki}^{BC}t_{Lj}^{Aa}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+\tfrac{-1}{4}\cdot[t_{IJ}^{ab}t_{Ki}^{CD}t_{Lj}^{AB}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}$$
$$+\tfrac{1}{2}\cdot[t_{I}^{b}t_{J}^{a}t_{i}^{D}t_{KLj}^{ABC}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+\tfrac{1}{4}\cdot[t_{I}^{b}t_{J}^{a}t_{Ki}^{CD}t_{Lj}^{AB}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[t_{I}^{b}t_{i}^{D}t_{j}^{C}t_{JKL}^{ABa}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[t_{I}^{b}t_{i}^{D}t_{JK}^{Ca}t_{Lj}^{AB}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}$$
$$+\tfrac{1}{4}\cdot[t_{i}^{D}t_{j}^{C}t_{IJ}^{Bb}t_{KL}^{Aa}v_{\underline{ab}}^{\underline{ij}}]_{\mathcal{A}}$$

C.2. Expectation Values

In expressions of the type $\langle\Phi_0|\exp(\hat{T})\hat{X}\exp(\hat{T})|\Phi_0\rangle$ there is no algebraic termination like for the BCH expansion. Therefore we have to specify up to which order the exponential series is evaluated. We give the results (expanded operator expressions and explicit formulas) for first and second order with $\hat{X}=\hat{H}$ and for first order with $\hat{X}=\hat{H}^2$.

$$
\begin{aligned}
&\langle\Phi_0|\underset{1\ldots1}{\exp}(\hat{T}_1^\dagger+\hat{T}_2^\dagger)(\hat{V}+\hat{F})\underset{1\ldots1}{\exp}(\hat{T}_1+\hat{T}_2)|\Phi_0\rangle\\
&=\langle\Phi_0|(\hat{F}+\hat{V}+\hat{T}_2^\dagger\hat{V}+\hat{T}_1^\dagger\hat{F}+\hat{T}_1^\dagger\hat{V}+\hat{F}\hat{T}_1+\hat{V}\hat{T}_1+\hat{V}\hat{T}_2+\hat{T}_2^\dagger\hat{F}\hat{T}_1+\hat{T}_2^\dagger\hat{F}\hat{T}_2+\hat{T}_2^\dagger\hat{V}\hat{T}_1\\
&+\hat{T}_2^\dagger\hat{V}\hat{T}_2+\hat{T}_1^\dagger\hat{F}\hat{T}_1+\hat{T}_1^\dagger\hat{F}\hat{T}_2+\hat{T}_1^\dagger\hat{V}\hat{T}_1+\hat{T}_1^\dagger\hat{V}\hat{T}_2)|\Phi_0\rangle\\
&=[f_{\bar{a}}^{i}t_i^a]_{\mathcal{A}}+[f_{\bar{a}}^{i}t_a^i]_{\mathcal{A}}+\tfrac{1}{4}\cdot[t_{ij}^{ab}v_{\overline{ab}}^{\overline{ij}}]_{\mathcal{A}}+\tfrac{1}{4}\cdot[t_{ab}^{ij}v_{\overline{ab}}^{\overline{ij}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f_{\bar{j}}^{i}t_j^at_a^i]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[f_{\bar{j}}^{i}t_{jk}^{ab}t_{ab}^{ik}]_{\mathcal{A}}+[f_{\bar{a}}^{i}t_j^bt_{ab}^{ij}]_{\mathcal{A}}\\
&+[f_{\bar{a}}^{i}t_b^jt_{ij}^{ab}]_{\mathcal{A}}+[f_{\bar{b}}^{a}t_i^at_b^i]_{\mathcal{A}}+\tfrac{1}{2}\cdot[f_{\bar{b}}^{a}t_{ij}^{ac}t_{bc}^{ij}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[t_j^at_b^iv_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[t_j^at_{bc}^{ij}v_{\overline{bc}}^{\overline{ia}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[t_k^bt_{ab}^{ij}v_{\overline{ka}}^{\overline{ij}}]_{\mathcal{A}}\\
&+\tfrac{1}{2}\cdot[t_b^it_{jk}^{ab}v_{\overline{jk}}^{\overline{ia}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[t_c^jt_{ij}^{ab}v_{\overline{ab}}^{\overline{ic}}]_{\mathcal{A}}+\tfrac{1}{8}\cdot[t_{ij}^{ab}t_{cd}^{ij}v_{\overline{cd}}^{\overline{ab}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[t_{jk}^{ac}t_{bc}^{ik}v_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}}+\tfrac{1}{8}\cdot[t_{kl}^{ab}t_{ab}^{ij}v_{\overline{kl}}^{\overline{ij}}]_{\mathcal{A}}
\end{aligned}
$$

$$
\begin{aligned}
&\langle\Phi_0|\underset{1\ldots2}{\exp}(\hat{T}_1^\dagger+\hat{T}_2^\dagger)(\hat{V}+\hat{F})\underset{1\ldots2}{\exp}(\hat{T}_1+\hat{T}_2)|\Phi_0\rangle\\
&=\langle\Phi_0|(\hat{F}+\hat{V}+\hat{T}_2^\dagger\hat{V}+\hat{T}_1^\dagger\hat{F}+\hat{T}_1^\dagger\hat{V}+\tfrac{1}{2}\cdot(\hat{T}_1^\dagger)^{\bar{2}}\hat{V}+\hat{F}\hat{T}_1+\hat{V}\hat{T}_1+\tfrac{1}{2}\cdot\hat{V}(\hat{T}_1)^{\bar{2}}+\hat{V}\hat{T}_2\\
&+\hat{T}_2^\dagger\hat{F}\hat{T}_1+\tfrac{1}{2}\cdot\hat{T}_2^\dagger\hat{F}(\hat{T}_1)^{\bar{2}}+\hat{T}_2^\dagger\hat{F}\hat{T}_2+\hat{T}_2^\dagger\hat{V}\hat{T}_1+\tfrac{1}{2}\cdot\hat{T}_2^\dagger\hat{V}(\hat{T}_1)^{\bar{2}}+\hat{T}_2^\dagger\hat{V}\hat{T}_2+\tfrac{1}{2}\cdot\hat{T}_2^\dagger\hat{V}(\hat{T}_2)^{\bar{2}}\\
&+\tfrac{1}{4}\cdot(\hat{T}_2^\dagger)^{\bar{2}}\hat{F}(\hat{T}_2)^{\bar{2}}+\tfrac{1}{4}\cdot(\hat{T}_2^\dagger)^{\bar{2}}\hat{V}(\hat{T}_1)^{\bar{2}}+\tfrac{1}{2}\cdot(\hat{T}_2^\dagger)^{\bar{2}}\hat{V}\hat{T}_2+\tfrac{1}{4}\cdot(\hat{T}_2^\dagger)^{\bar{2}}\hat{V}(\hat{T}_2)^{\bar{2}}+\hat{T}_1^\dagger\hat{F}\hat{T}_1\\
&+\tfrac{1}{2}\cdot\hat{T}_1^\dagger\hat{F}(\hat{T}_1)^{\bar{2}}+\hat{T}_1^\dagger\hat{F}\hat{T}_2+\hat{T}_1^\dagger\hat{V}\hat{T}_1+\tfrac{1}{2}\cdot\hat{T}_1^\dagger\hat{V}(\hat{T}_1)^{\bar{2}}+\hat{T}_1^\dagger\hat{V}\hat{T}_2+\tfrac{1}{2}\cdot(\hat{T}_1^\dagger)^{\bar{2}}\hat{F}\hat{T}_1+\tfrac{1}{4}\cdot(\hat{T}_1^\dagger)^{\bar{2}}\hat{F}(\hat{T}_1)^{\bar{2}}\\
&+\tfrac{1}{2}\cdot(\hat{T}_1^\dagger)^{\bar{2}}\hat{F}\hat{T}_2+\tfrac{1}{2}\cdot(\hat{T}_1^\dagger)^{\bar{2}}\hat{V}\hat{T}_1+\tfrac{1}{4}\cdot(\hat{T}_1^\dagger)^{\bar{2}}\hat{V}(\hat{T}_1)^{\bar{2}}+\tfrac{1}{2}\cdot(\hat{T}_1^\dagger)^{\bar{2}}\hat{V}\hat{T}_2+\tfrac{1}{4}\cdot(\hat{T}_1^\dagger)^{\bar{2}}\hat{V}(\hat{T}_2)^{\bar{2}}\\
&+\tfrac{1}{2}\cdot\hat{T}_2^\dagger\hat{T}_1^\dagger\hat{F}(\hat{T}_1)^{\bar{2}}+\hat{T}_2^\dagger\hat{T}_1^\dagger\hat{F}\hat{T}_2+\tfrac{1}{2}\cdot\hat{T}_2^\dagger\hat{T}_1^\dagger\hat{F}(\hat{T}_2)^{\bar{2}}+\hat{T}_2^\dagger\hat{T}_1^\dagger\hat{V}\hat{T}_1+\tfrac{1}{2}\cdot\hat{T}_2^\dagger\hat{T}_1^\dagger\hat{V}(\hat{T}_1)^{\bar{2}}+\hat{T}_2^\dagger\hat{T}_1^\dagger\hat{V}\hat{T}_2\\
&+\tfrac{1}{2}\cdot\hat{T}_2^\dagger\hat{T}_1^\dagger\hat{V}(\hat{T}_2)^{\bar{2}}+\hat{T}_2^\dagger\hat{F}\hat{T}_1\hat{T}_2+\hat{T}_2^\dagger\hat{V}\hat{T}_1\hat{T}_2+\tfrac{1}{2}\cdot(\hat{T}_2^\dagger)^{\bar{2}}\hat{F}\hat{T}_1\hat{T}_2+\tfrac{1}{2}\cdot(\hat{T}_2^\dagger)^{\bar{2}}\hat{V}\hat{T}_1\hat{T}_2+\hat{T}_1^\dagger\hat{V}\hat{T}_1\hat{T}_2\\
&+\tfrac{1}{2}\cdot(\hat{T}_1^\dagger)^{\bar{2}}\hat{F}\hat{T}_1\hat{T}_2+\tfrac{1}{2}\cdot(\hat{T}_1^\dagger)^{\bar{2}}\hat{V}\hat{T}_1\hat{T}_2+\hat{T}_2^\dagger\hat{T}_1^\dagger\hat{F}\hat{T}_1\hat{T}_2+\hat{T}_2^\dagger\hat{T}_1^\dagger\hat{V}\hat{T}_1\hat{T}_2)|\Phi_0\rangle\\
&=[f_{\bar{a}}^{i}t_i^a]_{\mathcal{A}}+[f_{\bar{a}}^{i}t_a^i]_{\mathcal{A}}+\tfrac{1}{4}\cdot[t_{ij}^{ab}v_{\overline{ab}}^{\overline{ij}}]_{\mathcal{A}}+\tfrac{1}{4}\cdot[t_{ab}^{ij}v_{\overline{ab}}^{\overline{ij}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f_{\bar{j}}^{i}t_j^at_a^i]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[f_{\bar{j}}^{i}t_{jk}^{ab}t_{ab}^{ik}]_{\mathcal{A}}\\
&+[f_{\bar{a}}^{i}t_b^jt_{ab}^{ij}]_{\mathcal{A}}+[f_{\bar{a}}^{i}t_j^bt_{ij}^{ab}]_{\mathcal{A}}+[f_{\bar{b}}^{a}t_i^at_b^i]_{\mathcal{A}}+\tfrac{1}{2}\cdot[f_{\bar{b}}^{a}t_{ij}^{ac}t_{bc}^{ij}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[t_i^bt_j^av_{\overline{ab}}^{\overline{ij}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[t_j^at_b^iv_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}}\\
&+\tfrac{1}{2}\cdot[t_j^at_{bc}^{ij}v_{\overline{bc}}^{\overline{ia}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[t_k^bt_{ab}^{ij}v_{\overline{ka}}^{\overline{ij}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[t_a^jt_b^iv_{\overline{ab}}^{\overline{ij}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[t_b^it_{jk}^{ab}v_{\overline{jk}}^{\overline{ia}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[t_c^jt_{ij}^{ab}v_{\overline{ab}}^{\overline{ic}}]_{\mathcal{A}}\\
&+\tfrac{1}{8}\cdot[t_{ij}^{ab}t_{cd}^{ij}v_{\overline{cd}}^{\overline{ab}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[t_{jk}^{ac}t_{bc}^{ik}v_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}}+\tfrac{1}{8}\cdot[t_{kl}^{ab}t_{ab}^{ij}v_{\overline{kl}}^{\overline{ij}}]_{\mathcal{A}}+[f_{\bar{j}}^{i}t_j^bt_k^at_{ab}^{ik}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f_{\bar{j}}^{i}t_a^it_b^kt_{jk}^{ab}]_{\mathcal{A}}\\
&+[f_{\bar{a}}^{i}t_i^at_j^bt_b^j]_{\mathcal{A}}+\tfrac{1}{4}\cdot[f_{\bar{a}}^{i}t_i^at_{jk}^{bc}t_{bc}^{jk}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f_{\bar{a}}^{i}t_i^bt_j^at_b^j]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[f_{\bar{a}}^{i}t_i^bt_{jk}^{ac}t_{bc}^{jk}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[f_{\bar{a}}^{i}t_j^at_{ik}^{bc}t_{bc}^{jk}]_{\mathcal{A}}\\
&+[f_{\bar{a}}^{i}t_j^bt_i^at_b^j]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f_{\bar{a}}^{i}t_j^bt_a^jt_b^i]_{\mathcal{A}}+[f_{\bar{a}}^{i}t_j^bt_{ik}^{ac}t_{bc}^{jk}]_{\mathcal{A}}+\tfrac{1}{4}\cdot[f_{\bar{a}}^{i}t_a^it_{jk}^{bc}t_{bc}^{jk}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[f_{\bar{a}}^{i}t_a^jt_{jk}^{bc}t_{bc}^{ik}]_{\mathcal{A}}
\end{aligned}
$$

$$+\tfrac{-1}{2}\cdot[f_{\bar{a}}^{i}t_{b}^{i}t_{jk}^{bc}t_{ac}^{jk}]_{\mathcal{A}}+[f_{\bar{a}}^{i}t_{b}^{j}t_{jk}^{bc}t_{ac}^{ik}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f_{\bar{b}}^{a}t_{i}^{c}t_{j}^{a}t_{bc}^{ij}]_{\mathcal{A}}+[f_{\bar{b}}^{a}t_{b}^{i}t_{c}^{j}t_{ij}^{ac}]_{\mathcal{A}}+[t_{i}^{a}t_{j}^{b}t_{c}^{j}v_{\overline{ab}}^{\overline{ic}}]_{\mathcal{A}}$$
$$+\tfrac{-1}{4}\cdot[t_{i}^{b}t_{j}^{a}t_{cd}^{ij}v_{\overline{cd}}^{\overline{ab}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[t_{i}^{b}t_{c}^{k}t_{jk}^{ac}v_{\overline{ab}}^{\overline{ij}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[t_{i}^{b}t_{jk}^{ad}t_{cd}^{jk}v_{\overline{ab}}^{\overline{ic}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[t_{i}^{c}t_{c}^{k}t_{jk}^{ab}v_{\overline{ab}}^{\overline{ij}}]_{\mathcal{A}}$$
$$+\tfrac{1}{4}\cdot[t_{i}^{d}t_{jk}^{ab}t_{cd}^{jk}v_{\overline{ab}}^{\overline{ic}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[t_{j}^{a}t_{k}^{c}t_{bc}^{ik}v_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}}+[t_{j}^{a}t_{b}^{i}t_{c}^{j}v_{\overline{bc}}^{\overline{ia}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[t_{j}^{a}t_{kl}^{bc}t_{bc}^{il}v_{\overline{jk}}^{\overline{ia}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[t_{j}^{b}t_{k}^{a}t_{b}^{i}v_{\overline{jk}}^{\overline{ia}}]_{\mathcal{A}}$$
$$+[t_{j}^{b}t_{ik}^{ad}t_{cd}^{jk}v_{\overline{ab}}^{\overline{ic}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[t_{j}^{b}t_{kl}^{ac}t_{bc}^{il}v_{\overline{jk}}^{\overline{ia}}]_{\mathcal{A}}+[t_{j}^{c}t_{k}^{a}t_{bc}^{ik}v_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[t_{j}^{d}t_{ik}^{ab}t_{cd}^{jk}v_{\overline{ab}}^{\overline{ic}}]_{\mathcal{A}}+\tfrac{1}{4}\cdot[t_{k}^{a}t_{l}^{b}t_{ab}^{ij}v_{\overline{kl}}^{\overline{ij}}]_{\mathcal{A}}$$
$$+\tfrac{-1}{1}\cdot[t_{k}^{b}t_{a}^{j}t_{b}^{i}v_{\overline{ka}}^{\overline{ij}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[t_{k}^{b}t_{c}^{k}t_{ij}^{ac}v_{\overline{ab}}^{\overline{ij}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[t_{k}^{c}t_{b}^{i}t_{ac}^{jk}v_{\overline{ab}}^{\overline{ij}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[t_{k}^{c}t_{b}^{k}t_{ac}^{ij}v_{\overline{ab}}^{\overline{ij}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[t_{k}^{c}t_{c}^{i}t_{ab}^{jk}v_{\overline{ab}}^{\overline{ij}}]_{\mathcal{A}}$$
$$+\tfrac{1}{4}\cdot[t_{k}^{c}t_{c}^{k}t_{ij}^{ab}v_{\overline{ab}}^{\overline{ij}}]_{\mathcal{A}}+\tfrac{1}{4}\cdot[t_{k}^{c}t_{c}^{k}t_{ab}^{ij}v_{\overline{ab}}^{\overline{ij}}]_{\mathcal{A}}+\tfrac{1}{4}\cdot[t_{l}^{a}t_{jk}^{bc}t_{bc}^{il}v_{\overline{jk}}^{\overline{ia}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[t_{l}^{b}t_{jk}^{ac}t_{bc}^{il}v_{\overline{jk}}^{\overline{ia}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[t_{i}^{a}t_{kl}^{bc}t_{bc}^{jl}v_{\overline{ka}}^{\overline{ij}}]_{\mathcal{A}}$$
$$+\tfrac{-1}{4}\cdot[t_{a}^{j}t_{b}^{i}t_{kl}^{ab}v_{\overline{kl}}^{\overline{ij}}]_{\mathcal{A}}+\tfrac{1}{4}\cdot[t_{a}^{l}t_{kl}^{bc}t_{bc}^{ij}v_{\overline{ka}}^{\overline{ij}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[t_{b}^{i}t_{c}^{k}t_{jk}^{ac}v_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[t_{b}^{i}t_{kl}^{bc}t_{ac}^{jl}v_{\overline{ka}}^{\overline{ij}}]_{\mathcal{A}}+[t_{b}^{k}t_{c}^{i}t_{jk}^{ac}v_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}}$$
$$+\tfrac{-1}{2}\cdot[t_{b}^{l}t_{kl}^{bc}t_{ac}^{ij}v_{\overline{ka}}^{\overline{ij}}]_{\mathcal{A}}+\tfrac{1}{4}\cdot[t_{c}^{i}t_{d}^{j}t_{ij}^{ab}v_{\overline{cd}}^{\overline{ab}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[t_{c}^{i}t_{jk}^{ad}t_{bd}^{jk}v_{\overline{bc}}^{\overline{ia}}]_{\mathcal{A}}+[t_{c}^{j}t_{jk}^{ad}t_{bd}^{ik}v_{\overline{bc}}^{\overline{ia}}]_{\mathcal{A}}+\tfrac{1}{4}\cdot[t_{d}^{i}t_{jk}^{ad}t_{bc}^{jk}v_{\overline{bc}}^{\overline{ia}}]_{\mathcal{A}}$$
$$+\tfrac{-1}{2}\cdot[t_{d}^{j}t_{jk}^{ad}t_{bc}^{ik}v_{\overline{bc}}^{\overline{ia}}]_{\mathcal{A}}+\tfrac{1}{16}\cdot[t_{ij}^{ab}t_{kl}^{cd}t_{cd}^{kl}v_{\overline{ab}}^{\overline{ij}}]_{\mathcal{A}}+\tfrac{-1}{4}\cdot[t_{ij}^{ac}t_{kl}^{bd}t_{cd}^{kl}v_{\overline{ab}}^{\overline{ij}}]_{\mathcal{A}}+\tfrac{1}{16}\cdot[t_{ij}^{cd}t_{kl}^{ab}t_{cd}^{kl}v_{\overline{ab}}^{\overline{ij}}]_{\mathcal{A}}$$
$$+\tfrac{1}{2}\cdot[t_{ik}^{bd}t_{jl}^{ac}t_{cd}^{kl}v_{\overline{ab}}^{\overline{ij}}]_{\mathcal{A}}+\tfrac{-1}{4}\cdot[t_{ik}^{cd}t_{jl}^{ab}t_{cd}^{kl}v_{\overline{ab}}^{\overline{ij}}]_{\mathcal{A}}+\tfrac{1}{16}\cdot[t_{kl}^{cd}t_{ab}^{ij}t_{cd}^{kl}v_{\overline{ab}}^{\overline{ij}}]_{\mathcal{A}}+\tfrac{1}{4}\cdot[t_{kl}^{cd}t_{ab}^{jk}t_{cd}^{il}v_{\overline{ab}}^{\overline{ij}}]_{\mathcal{A}}$$
$$+\tfrac{1}{16}\cdot[t_{kl}^{cd}t_{ab}^{kl}t_{cd}^{ij}v_{\overline{ab}}^{\overline{ij}}]_{\mathcal{A}}+\tfrac{-1}{4}\cdot[t_{kl}^{cd}t_{ac}^{ij}t_{bd}^{kl}v_{\overline{ab}}^{\overline{ij}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[t_{kl}^{cd}t_{ac}^{jk}t_{bd}^{il}v_{\overline{ab}}^{\overline{ij}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f_{\bar{j}}^{i}t_{j}^{a}t_{b}^{k}t_{a}^{i}t_{b}^{k}]_{\mathcal{A}}$$
$$+\tfrac{-1}{4}\cdot[f_{\bar{j}}^{i}t_{j}^{a}t_{a}^{i}t_{kl}^{bc}t_{bc}^{kl}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[f_{\bar{j}}^{i}t_{j}^{a}t_{a}^{k}t_{kl}^{bc}t_{bc}^{il}]_{\mathcal{A}}+[f_{\bar{j}}^{i}t_{j}^{b}t_{k}^{a}t_{a}^{i}t_{b}^{k}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[f_{\bar{j}}^{i}t_{j}^{b}t_{j}^{i}t_{a}^{ac}t_{kl}^{kl}t_{bc}^{kl}]_{\mathcal{A}}$$
$$+\tfrac{-1}{1}\cdot[f_{\bar{j}}^{i}t_{j}^{b}t_{k}^{k}t_{a}^{ac}t_{kl}^{il}t_{bc}^{kl}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[f_{\bar{j}}^{i}t_{k}^{a}t_{a}^{i}t_{jl}^{bc}t_{bc}^{kl}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[f_{\bar{j}}^{i}t_{k}^{a}t_{a}^{k}t_{jl}^{bc}t_{bc}^{il}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f_{\bar{j}}^{i}t_{k}^{b}t_{a}^{i}t_{jl}^{ac}t_{bc}^{kl}]_{\mathcal{A}}$$
$$+[f_{\bar{j}}^{i}t_{k}^{b}t_{a}^{k}t_{jl}^{ac}t_{bc}^{il}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[f_{\bar{j}}^{i}t_{l}^{a}t_{a}^{k}t_{jk}^{bc}t_{bc}^{il}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f_{\bar{j}}^{i}t_{l}^{b}t_{a}^{k}t_{jk}^{ac}t_{bc}^{il}]_{\mathcal{A}}+\tfrac{-1}{8}\cdot[f_{\bar{j}}^{i}t_{jk}^{ab}t_{lm}^{cd}t_{ab}^{ik}t_{cd}^{lm}]_{\mathcal{A}}$$
$$+\tfrac{1}{2}\cdot[f_{\bar{j}}^{i}t_{jk}^{bd}t_{lm}^{ac}t_{ab}^{ik}t_{cd}^{lm}]_{\mathcal{A}}+\tfrac{-1}{8}\cdot[f_{\bar{j}}^{i}t_{jk}^{cd}t_{lm}^{ab}t_{ab}^{ik}t_{cd}^{lm}]_{\mathcal{A}}+\tfrac{-1}{4}\cdot[f_{\bar{j}}^{i}t_{jm}^{ab}t_{kl}^{cd}t_{ab}^{ik}t_{cd}^{lm}]_{\mathcal{A}}+[f_{\bar{j}}^{i}t_{jm}^{ac}t_{kl}^{bd}t_{ab}^{ik}t_{cd}^{lm}]_{\mathcal{A}}$$
$$+\tfrac{-1}{4}\cdot[f_{\bar{j}}^{i}t_{jm}^{cd}t_{kl}^{ab}t_{ab}^{ik}t_{cd}^{lm}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[f_{\bar{a}}^{i}t_{i}^{a}t_{b}^{j}t_{c}^{k}t_{jk}^{bc}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f_{\bar{a}}^{i}t_{i}^{b}t_{b}^{j}t_{c}^{k}t_{jk}^{ac}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f_{\bar{a}}^{i}t_{a}^{a}t_{j}^{j}t_{b}^{k}t_{c}^{bc}t_{ik}^{bc}]_{\mathcal{A}}$$
$$+\tfrac{1}{2}\cdot[f_{\bar{a}}^{i}t_{j}^{b}t_{k}^{c}t_{a}^{i}t_{bc}^{jk}]_{\mathcal{A}}+[f_{\bar{a}}^{i}t_{j}^{b}t_{k}^{c}t_{b}^{j}t_{ac}^{ik}]_{\mathcal{A}}+[f_{\bar{a}}^{i}t_{j}^{b}t_{b}^{j}t_{c}^{k}t_{ik}^{ac}]_{\mathcal{A}}+\tfrac{1}{4}\cdot[f_{\bar{a}}^{i}t_{j}^{b}t_{kl}^{cd}t_{ab}^{ij}t_{cd}^{kl}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[f_{\bar{a}}^{i}t_{j}^{b}t_{kl}^{cd}t_{ab}^{ik}t_{cd}^{jl}]_{\mathcal{A}}$$
$$+\tfrac{-1}{2}\cdot[f_{\bar{a}}^{i}t_{j}^{b}t_{kl}^{cd}t_{ab}^{jl}t_{cd}^{ik}]_{\mathcal{A}}+\tfrac{1}{4}\cdot[f_{\bar{a}}^{i}t_{j}^{b}t_{kl}^{cd}t_{ab}^{kl}t_{cd}^{ij}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[f_{\bar{a}}^{i}t_{j}^{b}t_{kl}^{cd}t_{ac}^{ij}t_{bd}^{kl}]_{\mathcal{A}}+[f_{\bar{a}}^{i}t_{j}^{b}t_{kl}^{cd}t_{ac}^{ik}t_{bd}^{jl}]_{\mathcal{A}}$$
$$+\tfrac{-1}{1}\cdot[f_{\bar{a}}^{i}t_{j}^{b}t_{kl}^{cd}t_{ad}^{jl}t_{bc}^{ik}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[f_{\bar{a}}^{i}t_{j}^{b}t_{kl}^{cd}t_{ad}^{kl}t_{bc}^{ij}]_{\mathcal{A}}+[f_{\bar{a}}^{i}t_{j}^{c}t_{k}^{b}t_{a}^{j}t_{bc}^{ik}]_{\mathcal{A}}+[f_{\bar{a}}^{i}t_{j}^{c}t_{k}^{b}t_{b}^{i}t_{ac}^{jk}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f_{\bar{a}}^{i}t_{j}^{c}t_{k}^{b}t_{b}^{j}t_{ac}^{ik}]_{\mathcal{A}}$$
$$+\tfrac{-1}{1}\cdot[f_{\bar{a}}^{i}t_{j}^{c}t_{b}^{j}t_{c}^{k}t_{ik}^{ab}]_{\mathcal{A}}+\tfrac{1}{4}\cdot[f_{\bar{a}}^{i}t_{b}^{j}t_{ij}^{ab}t_{kl}^{cd}t_{cd}^{kl}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[f_{\bar{a}}^{i}t_{b}^{j}t_{ij}^{ac}t_{kl}^{bd}t_{cd}^{kl}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[f_{\bar{a}}^{i}t_{b}^{j}t_{ij}^{bc}t_{kl}^{ad}t_{cd}^{kl}]_{\mathcal{A}}$$
$$+\tfrac{1}{4}\cdot[f_{\bar{a}}^{i}t_{b}^{j}t_{ij}^{cd}t_{kl}^{ab}t_{cd}^{kl}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[f_{\bar{a}}^{i}t_{b}^{j}t_{il}^{ab}t_{jk}^{cd}t_{cd}^{kl}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f_{\bar{a}}^{i}t_{b}^{j}t_{il}^{ac}t_{jk}^{bd}t_{cd}^{kl}]_{\mathcal{A}}+[f_{\bar{a}}^{i}t_{b}^{j}t_{il}^{bc}t_{jk}^{ad}t_{cd}^{kl}]_{\mathcal{A}}$$
$$+\tfrac{1}{2}\cdot[f_{\bar{a}}^{i}t_{b}^{j}t_{il}^{cd}t_{jk}^{ab}t_{cd}^{kl}]_{\mathcal{A}}+[f_{\bar{b}}^{a}t_{i}^{a}t_{j}^{c}t_{b}^{i}t_{c}^{j}]_{\mathcal{A}}+\tfrac{1}{4}\cdot[f_{\bar{b}}^{a}t_{i}^{a}t_{i}^{i}t_{b}^{i}t_{jk}^{cd}t_{cd}^{jk}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[f_{\bar{b}}^{a}t_{i}^{a}t_{c}^{i}t_{jk}^{cd}t_{bd}^{jk}]_{\mathcal{A}}$$
$$+\tfrac{-1}{1}\cdot[f_{\bar{b}}^{a}t_{i}^{c}t_{j}^{a}t_{b}^{i}t_{c}^{j}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[f_{\bar{b}}^{a}t_{i}^{c}t_{b}^{i}t_{jk}^{ad}t_{cd}^{jk}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[f_{\bar{b}}^{a}t_{i}^{c}t_{c}^{i}t_{jk}^{ad}t_{bd}^{jk}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[f_{\bar{b}}^{a}t_{i}^{d}t_{c}^{i}t_{jk}^{ac}t_{bd}^{jk}]_{\mathcal{A}}$$
$$+\tfrac{-1}{2}\cdot[f_{\bar{b}}^{a}t_{j}^{a}t_{b}^{i}t_{ik}^{cd}t_{cd}^{jk}]_{\mathcal{A}}+[f_{\bar{b}}^{a}t_{j}^{a}t_{c}^{i}t_{ik}^{cd}t_{bd}^{jk}]_{\mathcal{A}}+[f_{\bar{b}}^{a}t_{j}^{c}t_{b}^{i}t_{ik}^{ad}t_{cd}^{jk}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f_{\bar{b}}^{a}t_{j}^{c}t_{c}^{i}t_{ik}^{ad}t_{bd}^{jk}]_{\mathcal{A}}$$
$$+[f_{\bar{b}}^{a}t_{j}^{d}t_{c}^{i}t_{ik}^{ac}t_{bd}^{jk}]_{\mathcal{A}}+\tfrac{1}{8}\cdot[f_{\bar{b}}^{a}t_{ij}^{ac}t_{kl}^{de}t_{bc}^{ij}t_{de}^{kl}]_{\mathcal{A}}+\tfrac{1}{4}\cdot[f_{\bar{b}}^{a}t_{ij}^{ae}t_{kl}^{cd}t_{bc}^{ij}t_{de}^{kl}]_{\mathcal{A}}+\tfrac{-1}{4}\cdot[f_{\bar{b}}^{a}t_{ij}^{ce}t_{kl}^{ad}t_{bc}^{ij}t_{de}^{kl}]_{\mathcal{A}}$$
$$+\tfrac{1}{8}\cdot[f_{\bar{b}}^{a}t_{ij}^{de}t_{kl}^{ac}t_{bc}^{ij}t_{de}^{kl}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[f_{\bar{b}}^{a}t_{ik}^{ac}t_{jl}^{de}t_{bc}^{ij}t_{de}^{kl}]_{\mathcal{A}}+[f_{\bar{b}}^{a}t_{ik}^{ce}t_{jl}^{ad}t_{bc}^{ij}t_{de}^{kl}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[t_{i}^{b}t_{j}^{a}t_{c}^{i}t_{d}^{j}v_{\overline{cd}}^{\overline{ab}}]_{\mathcal{A}}$$

$$+\tfrac{-1}{1}\cdot[t_i^b t_c^j t_d^k t_{jk}^{ad} v_{\overline{ab}}^{\overline{ic}}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_i^b t_d^i t_{jk}^{ae} t_{ce}^{jk} v_{\overline{cd}}^{\overline{ab}}]_{\mathcal{A}} + \tfrac{-1}{4}\cdot[t_i^b t_e^i t_{jk}^{ae} t_{cd}^{jk} v_{\overline{cd}}^{\overline{ab}}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_i^d t_c^j t_d^k t_{jk}^{ab} v_{\overline{ab}}^{\overline{ic}}]_{\mathcal{A}}$$

$$+\tfrac{-1}{4}\cdot[t_i^e t_d^i t_{jk}^{ab} t_{ce}^{jk} v_{\overline{cd}}^{\overline{ab}}]_{\mathcal{A}} + \tfrac{1}{8}\cdot[t_i^e t_e^i t_{jk}^{ab} t_{cd}^{jk} v_{\overline{cd}}^{\overline{ab}}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_j^a t_k^c t_b^i t_c^k v_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}} + [t_j^a t_k^c t_b^k t_c^i v_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}}$$

$$+[t_j^a t_k^d t_c^j t_{bd}^{ik} v_{\overline{bc}}^{\overline{ia}}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_j^a t_k^d t_d^j t_{bc}^{ik} v_{\overline{bc}}^{\overline{ia}}]_{\mathcal{A}} + [t_j^a t_b^i t_c^l t_{kl}^{bc} v_{\overline{jk}}^{\overline{ia}}]_{\mathcal{A}} + \tfrac{-1}{4}\cdot[t_j^a t_b^i t_{kl}^{cd} t_{cd}^{kl} v_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}}$$

$$+\tfrac{1}{2}\cdot[t_j^a t_b^k t_{kl}^{cd} t_{cd}^{il} v_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_j^a t_c^i t_{kl}^{cd} t_{bd}^{kl} v_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_j^a t_c^k t_{kl}^{cd} t_{bd}^{il} v_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}} + \tfrac{1}{8}\cdot[t_j^a t_{kl}^{de} t_{bc}^{ij} t_{de}^{kl} v_{\overline{bc}}^{\overline{ia}}]_{\mathcal{A}}$$

$$+\tfrac{-1}{4}\cdot[t_j^a t_{kl}^{de} t_{bc}^{ik} t_{de}^{jl} v_{\overline{bc}}^{\overline{ia}}]_{\mathcal{A}} + \tfrac{-1}{4}\cdot[t_j^a t_{kl}^{de} t_{bc}^{jl} t_{de}^{ik} v_{\overline{bc}}^{\overline{ia}}]_{\mathcal{A}} + \tfrac{1}{8}\cdot[t_j^a t_{kl}^{de} t_{bc}^{kl} t_{de}^{ij} v_{\overline{bc}}^{\overline{ia}}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_j^a t_{kl}^{de} t_{bd}^{ij} t_{ce}^{kl} v_{\overline{bc}}^{\overline{ia}}]_{\mathcal{A}}$$

$$+[t_j^a t_{kl}^{de} t_{bd}^{ik} t_{ce}^{jl} v_{\overline{bc}}^{\overline{ia}}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_j^b t_b^i t_c^l t_{kl}^{ac} v_{\overline{jk}}^{\overline{ia}}]_{\mathcal{A}} + [t_j^b t_c^j t_d^k t_{ik}^{ad} v_{\overline{ab}}^{\overline{ic}}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_j^b t_c^k t_d^j t_{ik}^{ad} v_{\overline{ab}}^{\overline{ic}}]_{\mathcal{A}}$$

$$+\tfrac{-1}{1}\cdot[t_j^b t_d^i t_{ik}^{ae} t_{ce}^{jk} v_{\overline{cd}}^{\overline{ab}}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_j^b t_e^i t_{ik}^{ae} t_{cd}^{jk} v_{\overline{cd}}^{\overline{ab}}]_{\mathcal{A}} + [t_j^c t_k^a t_b^i t_c^k v_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_j^c t_k^a t_b^k t_c^i v_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}}$$

$$+[t_j^c t_b^i t_c^l t_{kl}^{ab} v_{\overline{jk}}^{\overline{ia}}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_j^c t_b^i t_{kl}^{ad} t_{cd}^{kl} v_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_j^c t_b^k t_{kl}^{ad} t_{cd}^{il} v_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_j^c t_c^i t_{kl}^{ad} t_{bd}^{kl} v_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}}$$

$$+[t_j^c t_c^k t_{kl}^{ad} t_{bd}^{il} v_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}} + [t_j^d t_k^a t_c^i t_{bd}^{jk} v_{\overline{bc}}^{\overline{ia}}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_j^d t_k^a t_c^j t_{bd}^{ik} v_{\overline{bc}}^{\overline{ia}}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_j^d t_k^a t_d^i t_{bc}^{jk} v_{\overline{bc}}^{\overline{ia}}]_{\mathcal{A}}$$

$$+\tfrac{1}{2}\cdot[t_j^d t_k^a t_d^j t_{bc}^{ik} v_{\overline{bc}}^{\overline{ia}}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_j^d t_c^i t_{kl}^{ac} t_{bd}^{kl} v_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_j^d t_c^j t_d^k t_{ik}^{ab} v_{\overline{ab}}^{\overline{ic}}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_j^d t_c^k t_d^j t_{ik}^{ab} v_{\overline{ab}}^{\overline{ic}}]_{\mathcal{A}}$$

$$+\tfrac{-1}{1}\cdot[t_j^d t_c^k t_{kl}^{ac} t_{bd}^{il} v_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}} + \tfrac{-1}{4}\cdot[t_j^d t_{kl}^{ae} t_{bc}^{ij} t_{de}^{kl} v_{\overline{bc}}^{\overline{ia}}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_j^d t_{kl}^{ae} t_{bc}^{ik} t_{de}^{jl} v_{\overline{bc}}^{\overline{ia}}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_j^d t_{kl}^{ae} t_{bc}^{jl} t_{de}^{ik} v_{\overline{bc}}^{\overline{ia}}]_{\mathcal{A}}$$

$$+\tfrac{-1}{4}\cdot[t_j^d t_{kl}^{ae} t_{bc}^{kl} t_{de}^{ij} v_{\overline{bc}}^{\overline{ia}}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_j^d t_{kl}^{ae} t_{bd}^{ij} t_{ce}^{kl} v_{\overline{bc}}^{\overline{ia}}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_j^d t_{kl}^{ae} t_{bd}^{ik} t_{ce}^{jl} v_{\overline{bc}}^{\overline{ia}}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_j^d t_{kl}^{ae} t_{be}^{ij} t_{cd}^{kl} v_{\overline{bc}}^{\overline{ia}}]_{\mathcal{A}}$$

$$+[t_j^d t_{kl}^{ae} t_{be}^{ik} t_{cd}^{jl} v_{\overline{bc}}^{\overline{ia}}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_j^e t_d^i t_{ik}^{ab} t_{ce}^{jk} v_{\overline{cd}}^{\overline{ab}}]_{\mathcal{A}} + \tfrac{-1}{4}\cdot[t_j^e t_e^i t_{ik}^{ab} t_{cd}^{jk} v_{\overline{cd}}^{\overline{ab}}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_k^a t_l^b t_a^j t_b^i v_{\overline{kl}}^{\overline{ij}}]_{\mathcal{A}}$$

$$+\tfrac{1}{2}\cdot[t_k^a t_a^i t_{lm}^{bc} t_{bc}^{jm} v_{\overline{kl}}^{\overline{ij}}]_{\mathcal{A}} + \tfrac{1}{4}\cdot[t_k^a t_a^m t_{lm}^{bc} t_{bc}^{ij} v_{\overline{kl}}^{\overline{ij}}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_k^a t_b^i t_{jl}^{cd} t_{cd}^{kl} v_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_k^a t_b^k t_{jl}^{cd} t_{cd}^{il} v_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}}$$

$$+\tfrac{-1}{1}\cdot[t_k^a t_c^i t_{jl}^{cd} t_{bd}^{kl} v_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}} + [t_k^a t_c^k t_{jl}^{cd} t_{bd}^{il} v_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_k^b t_l^c t_b^i t_{ac}^{jl} v_{\overline{ka}}^{\overline{ij}}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_k^b t_l^c t_b^l t_{ac}^{ij} v_{\overline{ka}}^{\overline{ij}}]_{\mathcal{A}}$$

$$+\tfrac{-1}{1}\cdot[t_k^b t_a^i t_{lm}^{ac} t_{bc}^{jm} v_{\overline{kl}}^{\overline{ij}}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_k^b t_a^m t_{lm}^{ac} t_{bc}^{ij} v_{\overline{kl}}^{\overline{ij}}]_{\mathcal{A}} + \tfrac{1}{8}\cdot[t_k^b t_{lm}^{cd} t_{ab}^{ij} t_{cd}^{lm} v_{\overline{ka}}^{\overline{ij}}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_k^b t_{lm}^{cd} t_{ab}^{jl} t_{cd}^{im} v_{\overline{ka}}^{\overline{ij}}]_{\mathcal{A}}$$

$$+\tfrac{1}{8}\cdot[t_k^b t_{lm}^{cd} t_{ab}^{lm} t_{cd}^{ij} v_{\overline{ka}}^{\overline{ij}}]_{\mathcal{A}} + \tfrac{-1}{4}\cdot[t_k^b t_{lm}^{cd} t_{ac}^{ij} t_{bd}^{lm} v_{\overline{ka}}^{\overline{ij}}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_k^b t_{lm}^{cd} t_{ac}^{jl} t_{bd}^{im} v_{\overline{ka}}^{\overline{ij}}]_{\mathcal{A}} + \tfrac{1}{4}\cdot[t_k^b t_{lm}^{cd} t_{ad}^{lm} t_{bc}^{ij} v_{\overline{ka}}^{\overline{ij}}]_{\mathcal{A}}$$

$$+\tfrac{-1}{1}\cdot[t_k^c t_l^b t_a^i t_{bc}^{jl} v_{\overline{ka}}^{\overline{ij}}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_k^c t_l^b t_a^l t_{bc}^{ij} v_{\overline{ka}}^{\overline{ij}}]_{\mathcal{A}} + [t_k^c t_l^b t_b^i t_{ac}^{jl} v_{\overline{ka}}^{\overline{ij}}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_k^c t_l^b t_b^l t_{ac}^{ij} v_{\overline{ka}}^{\overline{ij}}]_{\mathcal{A}}$$

$$+\tfrac{1}{8}\cdot[t_k^c t_l^d t_{ab}^{ij} t_{cd}^{kl} v_{\overline{ab}}^{\overline{ij}}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_k^c t_l^d t_{ab}^{jk} t_{cd}^{il} v_{\overline{ab}}^{\overline{ij}}]_{\mathcal{A}} + \tfrac{1}{8}\cdot[t_k^c t_l^d t_{ab}^{kl} t_{cd}^{ij} v_{\overline{ab}}^{\overline{ij}}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_k^c t_l^d t_{ac}^{ij} t_{bd}^{kl} v_{\overline{ab}}^{\overline{ij}}]_{\mathcal{A}}$$

$$+\tfrac{-1}{2}\cdot[t_k^c t_l^d t_{ac}^{jk} t_{bd}^{il} v_{\overline{ab}}^{\overline{ij}}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_k^c t_l^d t_{ad}^{jk} t_{bc}^{il} v_{\overline{ab}}^{\overline{ij}}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_k^c t_b^i t_{jl}^{ad} t_{cd}^{kl} v_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}} + [t_k^c t_b^k t_{jl}^{ad} t_{cd}^{il} v_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}}$$

$$+[t_k^c t_c^i t_{jl}^{ad} t_{bd}^{kl} v_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_k^c t_c^k t_{jl}^{ad} t_{bd}^{il} v_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_k^d t_c^i t_{jl}^{ac} t_{bd}^{kl} v_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}} + [t_k^d t_c^k t_{jl}^{ac} t_{bd}^{il} v_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}}$$

$$+\tfrac{1}{2}\cdot[t_l^a t_b^i t_c^l t_{jk}^{bc} v_{\overline{jk}}^{\overline{ia}}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_l^a t_b^k t_{jk}^{cd} t_{cd}^{il} v_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_l^a t_c^k t_{jk}^{cd} t_{bd}^{il} v_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_l^b t_b^i t_c^l t_{jk}^{ac} v_{\overline{jk}}^{\overline{ia}}]_{\mathcal{A}}$$

$$+\tfrac{-1}{4}\cdot[t_l^b t_{km}^{cd} t_{ab}^{ij} t_{cd}^{lm} v_{\overline{ka}}^{\overline{ij}}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_l^b t_{km}^{cd} t_{ab}^{im} t_{cd}^{jl} v_{\overline{ka}}^{\overline{ij}}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_l^b t_{km}^{cd} t_{ab}^{jl} t_{cd}^{im} v_{\overline{ka}}^{\overline{ij}}]_{\mathcal{A}} + \tfrac{-1}{4}\cdot[t_l^b t_{km}^{cd} t_{ab}^{lm} t_{cd}^{ij} v_{\overline{ka}}^{\overline{ij}}]_{\mathcal{A}}$$

$$+\tfrac{1}{2}\cdot[t_l^b t_{km}^{cd} t_{ac}^{ij} t_{bd}^{lm} v_{\overline{ka}}^{\overline{ij}}]_{\mathcal{A}} + [t_l^b t_{km}^{cd} t_{ac}^{jl} t_{bd}^{im} v_{\overline{ka}}^{\overline{ij}}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_l^b t_{km}^{cd} t_{ad}^{im} t_{bc}^{jl} v_{\overline{ka}}^{\overline{ij}}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_l^b t_{km}^{cd} t_{ad}^{lm} t_{bc}^{ij} v_{\overline{ka}}^{\overline{ij}}]_{\mathcal{A}}$$

$$+\tfrac{1}{2}\cdot[t_l^c t_b^i t_c^l t_{jk}^{ab} v_{\overline{jk}}^{\overline{ia}}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_l^c t_b^k t_{jk}^{ad} t_{cd}^{il} v_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}} + [t_l^c t_c^k t_{jk}^{ad} t_{bd}^{il} v_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_l^d t_c^k t_{jk}^{ac} t_{bd}^{il} v_{\overline{jb}}^{\overline{ia}}]_{\mathcal{A}}$$

$$+\tfrac{1}{4}\cdot[t_m^a t_a^i t_{kl}^{bc} t_{bc}^{jm} v_{\overline{kl}}^{\overline{ij}}]_{\mathcal{A}} + \tfrac{1}{8}\cdot[t_m^a t_a^m t_{kl}^{bc} t_{bc}^{ij} v_{\overline{kl}}^{\overline{ij}}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_m^b t_a^i t_{kl}^{ac} t_{bc}^{jm} v_{\overline{kl}}^{\overline{ij}}]_{\mathcal{A}} + \tfrac{-1}{4}\cdot[t_m^b t_a^m t_{kl}^{ac} t_{bc}^{ij} v_{\overline{kl}}^{\overline{ij}}]_{\mathcal{A}}$$

$$+\tfrac{1}{8}\cdot[t_b^i t_{jk}^{ab} t_{lm}^{cd} t_{cd}^{lm} v_{ab}^{ia}]_{\mathcal{A}} + \tfrac{-1}{4}\cdot[t_b^i t_{jk}^{ac} t_{lm}^{bd} t_{cd}^{lm} v_{ab}^{ia}]_{\mathcal{A}} + \tfrac{1}{4}\cdot[t_b^i t_{jk}^{bc} t_{lm}^{ad} t_{cd}^{lm} v_{ab}^{ia}]_{\mathcal{A}} + \tfrac{1}{8}\cdot[t_b^i t_{jk}^{cd} t_{lm}^{ab} t_{cd}^{lm} v_{ab}^{ia}]_{\mathcal{A}}$$
$$+[t_b^i t_{jl}^{bd} t_{km}^{ac} t_{cd}^{lm} v_{jk}^{ia}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_b^i t_{jl}^{cd} t_{km}^{ab} t_{cd}^{lm} v_{jk}^{ia}]_{\mathcal{A}} + \tfrac{-1}{4}\cdot[t_b^l t_{jk}^{ab} t_{lm}^{cd} t_{cd}^{im} v_{jk}^{ia}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_b^l t_{jk}^{ac} t_{lm}^{bd} t_{cd}^{im} v_{jk}^{ia}]_{\mathcal{A}}$$
$$+\tfrac{-1}{2}\cdot[t_b^l t_{jk}^{bc} t_{lm}^{ad} t_{cd}^{im} v_{jk}^{ia}]_{\mathcal{A}} + \tfrac{-1}{4}\cdot[t_b^l t_{jk}^{cd} t_{lm}^{ab} t_{cd}^{im} v_{jk}^{ia}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_b^l t_{jm}^{ab} t_{kl}^{cd} t_{cd}^{im} v_{jk}^{ia}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_b^l t_{jm}^{ad} t_{kl}^{bc} t_{cd}^{im} v_{jk}^{ia}]_{\mathcal{A}}$$
$$+[t_b^l t_{jm}^{bd} t_{kl}^{ac} t_{cd}^{im} v_{jk}^{ia}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_b^l t_{jm}^{cd} t_{kl}^{ab} t_{cd}^{im} v_{jk}^{ia}]_{\mathcal{A}} + \tfrac{1}{8}\cdot[t_c^j t_{ij}^{ab} t_{kl}^{de} t_{de}^{kl} v_{ab}^{ic}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_c^j t_{ij}^{ad} t_{kl}^{be} t_{de}^{kl} v_{ab}^{ic}]_{\mathcal{A}}$$
$$+\tfrac{1}{8}\cdot[t_c^j t_{ij}^{de} t_{kl}^{ab} t_{de}^{kl} v_{ab}^{ic}]_{\mathcal{A}} + \tfrac{1}{4}\cdot[t_c^j t_{il}^{ab} t_{jk}^{de} t_{de}^{kl} v_{ab}^{ic}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_c^j t_{il}^{ad} t_{jk}^{be} t_{de}^{kl} v_{ab}^{ic}]_{\mathcal{A}} + \tfrac{1}{4}\cdot[t_c^j t_{il}^{de} t_{jk}^{ab} t_{de}^{kl} v_{ab}^{ic}]_{\mathcal{A}}$$
$$+\tfrac{1}{8}\cdot[t_c^k t_d^l t_{ij}^{ab} t_{kl}^{cd} v_{ab}^{ij}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_c^k t_d^l t_{ij}^{ad} t_{kl}^{bc} v_{ab}^{ij}]_{\mathcal{A}} + \tfrac{1}{8}\cdot[t_c^k t_d^l t_{ij}^{cd} t_{kl}^{ab} v_{ab}^{ij}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_c^k t_d^l t_{ik}^{bc} t_{jl}^{ad} v_{ab}^{ij}]_{\mathcal{A}}$$
$$+\tfrac{1}{2}\cdot[t_c^k t_d^l t_{ik}^{bd} t_{jl}^{ac} v_{ab}^{ij}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_c^k t_d^l t_{ik}^{cd} t_{jl}^{ab} v_{ab}^{ij}]_{\mathcal{A}} + \tfrac{-1}{4}\cdot[t_d^j t_{ij}^{ab} t_{kl}^{de} t_{ce}^{kl} v_{ab}^{ic}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_d^j t_{ij}^{ad} t_{kl}^{be} t_{ce}^{kl} v_{ab}^{ic}]_{\mathcal{A}}$$
$$+\tfrac{-1}{2}\cdot[t_d^j t_{ij}^{ae} t_{kl}^{bd} t_{ce}^{kl} v_{ab}^{ic}]_{\mathcal{A}} + \tfrac{-1}{4}\cdot[t_d^j t_{ij}^{de} t_{kl}^{ab} t_{ce}^{kl} v_{ab}^{ic}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_d^j t_{il}^{ab} t_{jk}^{de} t_{ce}^{kl} v_{ab}^{ic}]_{\mathcal{A}} + [t_d^j t_{il}^{ad} t_{jk}^{be} t_{ce}^{kl} v_{ab}^{ic}]_{\mathcal{A}}$$
$$+\tfrac{-1}{1}\cdot[t_d^j t_{il}^{ae} t_{jk}^{bd} t_{ce}^{kl} v_{ab}^{ic}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_d^j t_{il}^{de} t_{jk}^{ab} t_{ce}^{kl} v_{ab}^{ic}]_{\mathcal{A}} + \tfrac{1}{32}\cdot[t_{ij}^{ab} t_{kl}^{ef} t_{cd}^{ij} t_{ef}^{kl} v_{cd}^{ab}]_{\mathcal{A}} + \tfrac{-1}{8}\cdot[t_{ij}^{be} t_{kl}^{af} t_{ce}^{ij} t_{df}^{kl} v_{cd}^{ab}]_{\mathcal{A}}$$
$$+\tfrac{-1}{8}\cdot[t_{ij}^{bf} t_{kl}^{ae} t_{cd}^{ij} t_{ef}^{kl} v_{cd}^{ab}]_{\mathcal{A}} + \tfrac{1}{8}\cdot[t_{ij}^{bf} t_{kl}^{ae} t_{ce}^{ij} t_{df}^{kl} v_{cd}^{ab}]_{\mathcal{A}} + \tfrac{1}{32}\cdot[t_{ij}^{ef} t_{kl}^{ab} t_{cd}^{ij} t_{ef}^{kl} v_{cd}^{ab}]_{\mathcal{A}} + \tfrac{-1}{8}\cdot[t_{ij}^{ef} t_{kl}^{ab} t_{ce}^{ij} t_{df}^{kl} v_{cd}^{ab}]_{\mathcal{A}}$$
$$+\tfrac{1}{2}\cdot[t_{ik}^{be} t_{jl}^{af} t_{ce}^{ij} t_{df}^{kl} v_{cd}^{ab}]_{\mathcal{A}} + \tfrac{1}{4}\cdot[t_{ik}^{bf} t_{jl}^{ae} t_{cd}^{ij} t_{ef}^{kl} v_{cd}^{ab}]_{\mathcal{A}} + \tfrac{-1}{8}\cdot[t_{ik}^{ef} t_{jl}^{ab} t_{cd}^{ij} t_{ef}^{kl} v_{cd}^{ab}]_{\mathcal{A}} + \tfrac{1}{4}\cdot[t_{ik}^{ef} t_{jl}^{ab} t_{ce}^{ij} t_{df}^{kl} v_{cd}^{ab}]_{\mathcal{A}}$$
$$+\tfrac{-1}{4}\cdot[t_{jk}^{ac} t_{lm}^{de} t_{bc}^{ik} t_{de}^{lm} v_{jb}^{ia}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_{jk}^{ad} t_{lm}^{ce} t_{be}^{lm} t_{cd}^{ik} v_{jb}^{ia}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_{jk}^{ae} t_{lm}^{cd} t_{bc}^{ik} t_{de}^{lm} v_{jb}^{ia}]_{\mathcal{A}} + \tfrac{-1}{4}\cdot[t_{jk}^{ae} t_{lm}^{cd} t_{be}^{lm} t_{cd}^{ik} v_{jb}^{ia}]_{\mathcal{A}}$$
$$+\tfrac{-1}{4}\cdot[t_{jk}^{cd} t_{lm}^{ae} t_{be}^{lm} t_{cd}^{ik} v_{jb}^{ia}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_{jk}^{ce} t_{lm}^{ad} t_{bc}^{ik} t_{de}^{lm} v_{jb}^{ia}]_{\mathcal{A}} + \tfrac{-1}{4}\cdot[t_{jk}^{de} t_{lm}^{ac} t_{bc}^{ik} t_{de}^{lm} v_{jb}^{ia}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_{jk}^{de} t_{lm}^{ac} t_{be}^{lm} t_{cd}^{ik} v_{jb}^{ia}]_{\mathcal{A}}$$
$$+\tfrac{-1}{2}\cdot[t_{jm}^{ac} t_{kl}^{de} t_{bc}^{ik} t_{de}^{lm} v_{jb}^{ia}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_{jm}^{ac} t_{kl}^{de} t_{be}^{lm} t_{cd}^{ik} v_{jb}^{ia}]_{\mathcal{A}} + [t_{jm}^{ad} t_{kl}^{ce} t_{bc}^{ik} t_{de}^{lm} v_{jb}^{ia}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_{jm}^{ae} t_{kl}^{cd} t_{be}^{lm} t_{cd}^{ik} v_{jb}^{ia}]_{\mathcal{A}}$$
$$+\tfrac{-1}{1}\cdot[t_{jm}^{cd} t_{kl}^{ae} t_{bc}^{ik} t_{de}^{lm} v_{jb}^{ia}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_{jm}^{cd} t_{kl}^{ae} t_{be}^{lm} t_{cd}^{ik} v_{jb}^{ia}]_{\mathcal{A}} + [t_{jm}^{ce} t_{kl}^{ad} t_{be}^{lm} t_{cd}^{ik} v_{jb}^{ia}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_{jm}^{de} t_{kl}^{ac} t_{bc}^{ik} t_{de}^{lm} v_{jb}^{ia}]_{\mathcal{A}}$$
$$+\tfrac{1}{32}\cdot[t_{kl}^{ab} t_{mn}^{cd} t_{ab}^{ij} t_{cd}^{mn} v_{kl}^{ij}]_{\mathcal{A}} + \tfrac{-1}{8}\cdot[t_{kl}^{ac} t_{mn}^{bd} t_{ab}^{ij} t_{cd}^{mn} v_{kl}^{ij}]_{\mathcal{A}} + \tfrac{-1}{4}\cdot[t_{kl}^{ac} t_{mn}^{bd} t_{ab}^{jm} t_{cd}^{in} v_{kl}^{ij}]_{\mathcal{A}}$$
$$+\tfrac{1}{32}\cdot[t_{kl}^{cd} t_{mn}^{ab} t_{ab}^{ij} t_{cd}^{mn} v_{kl}^{ij}]_{\mathcal{A}} + \tfrac{1}{8}\cdot[t_{kl}^{cd} t_{mn}^{ab} t_{ab}^{jm} t_{cd}^{in} v_{kl}^{ij}]_{\mathcal{A}} + \tfrac{-1}{8}\cdot[t_{km}^{ab} t_{ln}^{cd} t_{ab}^{ij} t_{cd}^{mn} v_{kl}^{ij}]_{\mathcal{A}}$$
$$+\tfrac{-1}{8}\cdot[t_{km}^{ab} t_{ln}^{cd} t_{ab}^{jm} t_{cd}^{in} v_{kl}^{ij}]_{\mathcal{A}} + \tfrac{1}{4}\cdot[t_{km}^{bd} t_{ln}^{ac} t_{ab}^{ij} t_{cd}^{mn} v_{kl}^{ij}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_{km}^{bd} t_{ln}^{ac} t_{ab}^{jm} t_{cd}^{in} v_{kl}^{ij}]_{\mathcal{A}} + \tfrac{-1}{8}\cdot[t_{km}^{cd} t_{ln}^{ab} t_{ab}^{jm} t_{cd}^{in} v_{kl}^{ij}]_{\mathcal{A}}$$

$$\langle\Phi_0|\underset{1\ldots1}{\exp}(\hat{T}_1^\dagger+\hat{T}_2^\dagger)(\hat{V}+\hat{F})(\hat{V}+\hat{F})\underset{1\ldots1}{\exp}(\hat{T}_1+\hat{T}_2)|\Phi_0\rangle$$
$$=\langle\Phi_0|((\hat{F})^{\frac{2}{1}}+(\hat{V})^{\frac{2}{1}}+\hat{T}_2^\dagger(\hat{F})^{\frac{2}{1}}+\hat{T}_2^\dagger(\hat{V})^{\frac{2}{1}}+\hat{T}_1^\dagger(\hat{F})^{\frac{2}{1}}+\hat{T}_1^\dagger(\hat{V})^{\frac{2}{1}}+\hat{F}\hat{V}+(\hat{F})^{\frac{2}{1}}\hat{T}_1+(\hat{F})^{\frac{2}{1}}\hat{T}_2$$
$$+\hat{V}\hat{F}+(\hat{V})^{\frac{2}{1}}\hat{T}_1+(\hat{V})^{\frac{2}{1}}\hat{T}_2+\hat{T}_2^\dagger\hat{F}\hat{V}+\hat{T}_2^\dagger(\hat{F})^{\frac{2}{1}}\hat{T}_1+\hat{T}_2^\dagger(\hat{F})^{\frac{2}{1}}\hat{T}_2+\hat{T}_2^\dagger\hat{V}\hat{F}+\hat{T}_2^\dagger(\hat{V})^{\frac{2}{1}}\hat{T}_1$$
$$+\hat{T}_2^\dagger(\hat{V})^{\frac{2}{1}}\hat{T}_2+\hat{T}_1^\dagger\hat{F}\hat{V}+\hat{T}_1^\dagger(\hat{F})^{\frac{2}{1}}\hat{T}_1+\hat{T}_1^\dagger(\hat{F})^{\frac{2}{1}}\hat{T}_2+\hat{T}_1^\dagger\hat{V}\hat{F}+\hat{T}_1^\dagger(\hat{V})^{\frac{2}{1}}\hat{T}_1+\hat{T}_1^\dagger(\hat{V})^{\frac{2}{1}}\hat{T}_2+\hat{F}\hat{V}\hat{T}_1$$
$$+\hat{F}\hat{V}\hat{T}_2+\hat{V}\hat{F}\hat{T}_1+\hat{V}\hat{F}\hat{T}_2+\hat{T}_2^\dagger\hat{F}\hat{V}\hat{T}_1+\hat{T}_2^\dagger\hat{F}\hat{V}\hat{T}_2+\hat{T}_2^\dagger\hat{V}\hat{F}\hat{T}_1+\hat{T}_2^\dagger\hat{V}\hat{F}\hat{T}_2+\hat{T}_1^\dagger\hat{F}\hat{V}\hat{T}_1$$
$$+\hat{T}_1^\dagger\hat{F}\hat{V}\hat{T}_2+\hat{T}_1^\dagger\hat{V}\hat{F}\hat{T}_1+\hat{T}_1^\dagger\hat{V}\hat{F}\hat{T}_2)|\Phi_0\rangle$$
$$=[f_a^i f_a^i]_{\mathcal{A}}+\tfrac{1}{4}\cdot[v_{ab}^{ij}v_{ab}^{ij}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f_j^i f_a^i t_j^a]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f_j^i f_a^j t_a^i]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[f_j^i t_{jk}^{ab} v_{ab}^{ik}]_{\mathcal{A}}$$

$$+\tfrac{-1}{2}\cdot[f^{i}_{\bar{j}}t^{ik}_{ab}v^{\bar{j}\bar{k}}_{\bar{a}\bar{b}}]_{\mathcal{A}}+[f^{i}_{\bar{a}}f^{\bar{j}}_{\bar{b}}t^{ab}_{ij}]_{\mathcal{A}}+[f^{i}_{\bar{a}}f^{\bar{j}}_{\bar{b}}t^{ij}_{ab}]_{\mathcal{A}}+[f^{i}_{\bar{a}}f^{\bar{a}}_{\bar{b}}t^{\bar{b}}_{i}]_{\mathcal{A}}+[f^{i}_{\bar{a}}f^{\bar{a}}_{\bar{b}}t^{i}_{\bar{b}}]_{\mathcal{A}}$$
$$+[f^{i}_{\bar{a}}t^{\bar{b}}_{\bar{j}}v^{\bar{i}\bar{j}}_{\bar{a}\bar{b}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f^{i}_{\bar{a}}t^{\bar{b}}_{\bar{j}}v^{\bar{i}\bar{b}}_{\bar{j}\bar{a}}]_{\mathcal{A}}+[f^{i}_{\bar{a}}t^{\bar{j}}_{\bar{b}}v^{\bar{i}\bar{j}}_{\bar{a}\bar{b}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f^{i}_{\bar{a}}t^{\bar{j}}_{\bar{b}}v^{\bar{i}\bar{b}}_{\bar{j}\bar{a}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[f^{i}_{\bar{a}}t^{bc}_{ij}v^{\bar{j}\bar{a}}_{\bar{b}\bar{c}}]_{\mathcal{A}}$$
$$+\tfrac{-1}{2}\cdot[f^{i}_{\bar{a}}t^{ab}_{jk}v^{\bar{i}\bar{b}}_{\bar{j}\bar{k}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[f^{i}_{\bar{a}}t^{jk}_{ab}v^{\bar{i}\bar{b}}_{\bar{j}\bar{k}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[f^{i}_{\bar{a}}t^{ij}_{bc}v^{\bar{j}\bar{a}}_{\bar{b}\bar{c}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[f^{\bar{a}}_{\bar{b}}t^{ac}_{ij}v^{\bar{i}\bar{j}}_{\bar{b}\bar{c}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[f^{\bar{a}}_{\bar{b}}t^{ij}_{bc}v^{\bar{i}\bar{j}}_{\bar{a}\bar{c}}]_{\mathcal{A}}$$
$$+\tfrac{1}{2}\cdot[t^{c}_{\bar{j}}v^{\bar{i}\bar{j}}_{\bar{a}\bar{b}}v^{\bar{i}\bar{c}}_{\bar{a}\bar{b}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[t^{a}_{\bar{k}}v^{\bar{i}\bar{j}}_{\bar{k}\bar{b}}v^{\bar{i}\bar{j}}_{\bar{a}\bar{b}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[t^{i}_{\bar{b}}v^{\bar{i}\bar{a}}_{\bar{j}\bar{k}}v^{\bar{j}\bar{k}}_{\bar{a}\bar{b}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[t^{\bar{j}}_{\bar{c}}v^{\bar{i}\bar{j}}_{\bar{a}\bar{b}}v^{\bar{i}\bar{c}}_{\bar{a}\bar{b}}]_{\mathcal{A}}+\tfrac{1}{8}\cdot[t^{cd}_{ij}v^{\bar{i}\bar{j}}_{\bar{a}\bar{b}}v^{\bar{a}\bar{b}}_{\bar{c}\bar{d}}]_{\mathcal{A}}$$
$$+[t^{ac}_{jk}v^{\bar{i}\bar{j}}_{\bar{a}\bar{b}}v^{\bar{i}\bar{c}}_{\bar{k}\bar{b}}]_{\mathcal{A}}+\tfrac{1}{8}\cdot[t^{ab}_{kl}v^{\bar{i}\bar{j}}_{\bar{k}\bar{l}}v^{\bar{i}\bar{j}}_{\bar{a}\bar{b}}]_{\mathcal{A}}+\tfrac{1}{8}\cdot[t^{ij}_{ab}v^{\bar{i}\bar{j}}_{\bar{k}\bar{l}}v^{\bar{k}\bar{l}}_{\bar{a}\bar{b}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[t^{ik}_{bc}v^{\bar{i}\bar{a}}_{\bar{j}\bar{b}}v^{\bar{j}\bar{k}}_{\bar{a}\bar{c}}]_{\mathcal{A}}+\tfrac{1}{8}\cdot[t^{ij}_{cd}v^{\bar{i}\bar{j}}_{\bar{a}\bar{b}}v^{\bar{a}\bar{b}}_{\bar{c}\bar{d}}]_{\mathcal{A}}$$
$$+\tfrac{-1}{1}\cdot[f^{i}_{\bar{j}}f^{i}_{\bar{a}}t^{k}_{\bar{b}}t^{ab}_{\bar{j}\bar{k}}]_{\mathcal{A}}+[f^{i}_{\bar{j}}f^{\bar{j}}_{\bar{k}}t^{a}_{\bar{k}}t^{i}_{\bar{a}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[f^{i}_{\bar{j}}f^{\bar{j}}_{\bar{k}}t^{ab}_{kl}t^{il}_{ab}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f^{i}_{\bar{j}}f^{\bar{j}}_{\bar{a}}t^{\bar{b}}_{k}t^{ik}_{ab}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[f^{i}_{\bar{j}}f^{\bar{k}}_{\bar{l}}t^{ab}_{jl}t^{ik}_{ab}]_{\mathcal{A}}$$
$$+\tfrac{2}{1}\cdot[f^{i}_{\bar{j}}f^{\bar{k}}_{\bar{a}}t^{\bar{b}}_{\bar{j}}t^{ik}_{ab}]_{\mathcal{A}}+\tfrac{2}{1}\cdot[f^{i}_{\bar{j}}f^{\bar{k}}_{\bar{a}}t^{i}_{\bar{b}}t^{ab}_{\bar{j}\bar{k}}]_{\mathcal{A}}+\tfrac{-2}{1}\cdot[f^{i}_{\bar{j}}f^{\bar{a}}_{\bar{b}}t^{a}_{\bar{j}}t^{i}_{\bar{b}}]_{\mathcal{A}}+\tfrac{-2}{1}\cdot[f^{i}_{\bar{j}}f^{\bar{a}}_{\bar{b}}t^{ac}_{\bar{j}\bar{k}}t^{ik}_{bc}]_{\mathcal{A}}+[f^{i}_{\bar{j}}t^{a}_{\bar{j}}t^{\bar{k}}_{\bar{b}}v^{\bar{i}\bar{b}}_{\bar{k}\bar{a}}]_{\mathcal{A}}$$
$$+[f^{i}_{\bar{j}}t^{a}_{\bar{j}}t^{ik}_{bc}v^{\bar{k}\bar{a}}_{\bar{b}\bar{c}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[f^{i}_{\bar{j}}t^{\bar{b}}_{\bar{j}}t^{kl}_{ab}v^{\bar{i}\bar{a}}_{\bar{k}\bar{l}}]_{\mathcal{A}}+[f^{i}_{\bar{j}}t^{\bar{k}}_{\bar{k}}t^{i}_{\bar{b}}v^{\bar{j}\bar{a}}_{\bar{k}\bar{b}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[f^{i}_{\bar{j}}t^{a}_{\bar{k}}t^{ik}_{bc}v^{\bar{j}\bar{a}}_{\bar{b}\bar{c}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f^{i}_{\bar{j}}t^{\bar{b}}_{\bar{l}}t^{ik}_{ab}v^{\bar{j}\bar{k}}_{\bar{l}\bar{a}}]_{\mathcal{A}}$$
$$+\tfrac{-1}{2}\cdot[f^{i}_{\bar{j}}t^{i}_{\bar{b}}t^{ab}_{kl}v^{\bar{j}\bar{a}}_{\bar{k}\bar{l}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f^{i}_{\bar{j}}t^{\bar{k}}_{\bar{b}}t^{ab}_{jl}v^{\bar{i}\bar{l}}_{\bar{k}\bar{a}}]_{\mathcal{A}}+[f^{i}_{\bar{j}}t^{i}_{\bar{c}}t^{ab}_{jk}v^{\bar{k}\bar{c}}_{\bar{a}\bar{b}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[f^{i}_{\bar{j}}t^{\bar{k}}_{\bar{c}}t^{ab}_{jk}v^{\bar{i}\bar{c}}_{\bar{a}\bar{b}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[f^{i}_{\bar{j}}t^{ab}_{jk}t^{ik}_{cd}v^{\bar{a}\bar{b}}_{\bar{c}\bar{d}}]_{\mathcal{A}}$$
$$+\tfrac{2}{1}\cdot[f^{i}_{\bar{j}}t^{ac}_{jl}t^{ik}_{bc}v^{\bar{k}\bar{a}}_{\bar{l}\bar{b}}]_{\mathcal{A}}+[f^{i}_{\bar{j}}t^{ac}_{jl}t^{\bar{k}\bar{l}}_{\bar{b}\bar{c}}v^{\bar{i}\bar{b}}_{\bar{k}\bar{a}}]_{\mathcal{A}}+\tfrac{-1}{4}\cdot[f^{i}_{\bar{j}}t^{ab}_{jm}t^{kl}_{ab}v^{\bar{i}\bar{m}}_{\bar{k}\bar{l}}]_{\mathcal{A}}+[f^{i}_{\bar{j}}t^{ac}_{kl}t^{il}_{bc}v^{\bar{j}\bar{a}}_{\bar{k}\bar{b}}]_{\mathcal{A}}+\tfrac{-1}{4}\cdot[f^{i}_{\bar{j}}t^{ab}_{lm}t^{ik}_{ab}v^{\bar{j}\bar{k}}_{\bar{l}\bar{m}}]_{\mathcal{A}}$$
$$+[f^{i}_{\bar{a}}f^{i}_{\bar{a}}t^{\bar{b}}_{\bar{j}}t^{\bar{j}}_{\bar{b}}]_{\mathcal{A}}+\tfrac{1}{4}\cdot[f^{i}_{\bar{a}}f^{i}_{\bar{a}}t^{bc}_{jk}t^{jk}_{bc}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f^{i}_{\bar{a}}f^{i}_{\bar{b}}t^{a}_{\bar{j}}t^{\bar{j}}_{\bar{b}}]_{\mathcal{A}}+\tfrac{-1}{2}\cdot[f^{i}_{\bar{a}}f^{i}_{\bar{b}}t^{ac}_{jk}t^{jk}_{bc}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f^{i}_{\bar{a}}f^{\bar{j}}_{\bar{a}}t^{\bar{b}}_{i}t^{\bar{j}}_{\bar{b}}]_{\mathcal{A}}$$
$$+\tfrac{-1}{2}\cdot[f^{i}_{\bar{a}}f^{\bar{j}}_{\bar{a}}t^{bc}_{ik}t^{jk}_{bc}]_{\mathcal{A}}+\tfrac{2}{1}\cdot[f^{i}_{\bar{a}}f^{\bar{j}}_{\bar{b}}t^{\bar{b}}_{\bar{j}}t^{i}_{\bar{a}}]_{\mathcal{A}}+\tfrac{2}{1}\cdot[f^{i}_{\bar{a}}f^{\bar{j}}_{\bar{b}}t^{bc}_{jk}t^{ik}_{ac}]_{\mathcal{A}}+[f^{i}_{\bar{a}}f^{\bar{a}}_{\bar{b}}t^{\bar{c}}_{\bar{j}}t^{ij}_{bc}]_{\mathcal{A}}+[f^{i}_{\bar{a}}f^{\bar{a}}_{\bar{b}}t^{\bar{j}}_{\bar{c}}t^{\bar{b}\bar{c}}_{ij}]_{\mathcal{A}}$$
$$+\tfrac{2}{1}\cdot[f^{i}_{\bar{a}}f^{\bar{b}}_{\bar{c}}t^{\bar{b}}_{\bar{j}}t^{ij}_{ac}]_{\mathcal{A}}+\tfrac{2}{1}\cdot[f^{i}_{\bar{a}}f^{\bar{b}}_{\bar{c}}t^{\bar{c}}_{\bar{k}}t^{ab}_{ij}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[f^{i}_{\bar{a}}t^{a}_{i}t^{jk}_{bc}v^{\bar{j}\bar{k}}_{\bar{b}\bar{c}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f^{i}_{\bar{a}}t^{\bar{b}}_{i}t^{\bar{j}}_{\bar{c}}v^{\bar{j}\bar{b}}_{\bar{a}\bar{c}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[f^{i}_{\bar{a}}t^{\bar{c}}_{i}t^{jk}_{bc}v^{\bar{j}\bar{k}}_{\bar{a}\bar{b}}]_{\mathcal{A}}$$
$$+\tfrac{-1}{1}\cdot[f^{i}_{\bar{a}}t^{\bar{b}}_{\bar{j}}t^{i}_{\bar{c}}v^{\bar{j}\bar{c}}_{\bar{a}\bar{b}}]_{\mathcal{A}}+[f^{i}_{\bar{a}}t^{\bar{b}}_{\bar{j}}t^{\bar{j}}_{\bar{c}}v^{\bar{i}\bar{b}}_{\bar{a}\bar{c}}]_{\mathcal{A}}+[f^{i}_{\bar{a}}t^{\bar{b}}_{\bar{j}}t^{\bar{j}}_{\bar{c}}v^{\bar{i}\bar{c}}_{\bar{a}\bar{b}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[f^{i}_{\bar{a}}t^{\bar{b}}_{\bar{j}}t^{ij}_{cd}v^{\bar{a}\bar{b}}_{\bar{c}\bar{d}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f^{i}_{\bar{a}}t^{a}_{\bar{k}}t^{\bar{j}}_{\bar{b}}v^{\bar{i}\bar{j}}_{\bar{k}\bar{b}}]_{\mathcal{A}}$$
$$+\tfrac{1}{2}\cdot[f^{i}_{\bar{a}}t^{a}_{\bar{k}}t^{jk}_{bc}v^{\bar{i}\bar{j}}_{\bar{b}\bar{c}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f^{i}_{\bar{a}}t^{\bar{b}}_{\bar{k}}t^{\bar{j}}_{a}v^{\bar{i}\bar{k}}_{\bar{j}\bar{b}}]_{\mathcal{A}}+[f^{i}_{\bar{a}}t^{\bar{b}}_{\bar{k}}t^{\bar{j}}_{\bar{b}}v^{\bar{i}\bar{j}}_{\bar{k}\bar{a}}]_{\mathcal{A}}+[f^{i}_{\bar{a}}t^{\bar{b}}_{\bar{k}}t^{\bar{k}}_{\bar{b}}v^{\bar{i}\bar{k}}_{\bar{j}\bar{a}}]_{\mathcal{A}}+\tfrac{-2}{1}\cdot[f^{i}_{\bar{a}}t^{\bar{b}}_{\bar{k}}t^{ij}_{ac}v^{\bar{j}\bar{b}}_{\bar{k}\bar{c}}]_{\mathcal{A}}$$
$$+\tfrac{-1}{1}\cdot[f^{i}_{\bar{a}}t^{\bar{b}}_{\bar{k}}t^{\bar{j}\bar{k}}_{ac}v^{\bar{i}\bar{c}}_{\bar{j}\bar{b}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f^{i}_{\bar{a}}t^{\bar{c}}_{\bar{k}}t^{ij}_{bc}v^{\bar{j}\bar{a}}_{\bar{k}\bar{b}}]_{\mathcal{A}}+[f^{i}_{\bar{a}}t^{\bar{c}}_{\bar{k}}t^{\bar{j}\bar{k}}_{bc}v^{\bar{i}\bar{j}}_{\bar{a}\bar{b}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f^{i}_{\bar{a}}t^{\bar{c}}_{\bar{k}}t^{\bar{j}\bar{k}}_{bc}v^{\bar{i}\bar{b}}_{\bar{j}\bar{a}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[f^{i}_{\bar{a}}t^{\bar{b}}_{\bar{l}}t^{\bar{j}\bar{k}}_{ab}v^{\bar{i}\bar{l}}_{\bar{j}\bar{k}}]_{\mathcal{A}}$$
$$+\tfrac{1}{2}\cdot[f^{i}_{\bar{a}}t^{a}_{\bar{a}}t^{bc}_{jk}v^{\bar{j}\bar{k}}_{\bar{b}\bar{c}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[f^{i}_{\bar{a}}t^{\bar{k}}_{a}t^{bc}_{jk}v^{\bar{i}\bar{j}}_{\bar{b}\bar{c}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[f^{i}_{\bar{a}}t^{\bar{j}}_{\bar{b}}t^{ab}_{kl}v^{\bar{i}\bar{j}}_{\bar{k}\bar{l}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[f^{i}_{\bar{a}}t^{i}_{\bar{c}}t^{bc}_{jk}v^{\bar{j}\bar{k}}_{\bar{a}\bar{b}}]_{\mathcal{A}}+\tfrac{-2}{1}\cdot[f^{i}_{\bar{a}}t^{\bar{c}}_{\bar{i}}t^{ab}_{ik}v^{\bar{j}\bar{b}}_{\bar{k}\bar{c}}]_{\mathcal{A}}$$
$$+\tfrac{-1}{1}\cdot[f^{i}_{\bar{a}}t^{\bar{j}}_{\bar{c}}t^{bc}_{ik}v^{\bar{j}\bar{b}}_{\bar{k}\bar{a}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f^{i}_{\bar{a}}t^{\bar{k}}_{\bar{c}}t^{ab}_{jk}v^{\bar{i}\bar{b}}_{\bar{j}\bar{c}}]_{\mathcal{A}}+[f^{i}_{\bar{a}}t^{\bar{k}}_{\bar{c}}t^{bc}_{jk}v^{\bar{i}\bar{j}}_{\bar{a}\bar{b}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f^{i}_{\bar{a}}t^{\bar{k}}_{\bar{c}}t^{bc}_{jk}v^{\bar{i}\bar{b}}_{\bar{j}\bar{a}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[f^{i}_{\bar{a}}t^{\bar{j}}_{\bar{d}}t^{bc}_{ij}v^{\bar{a}\bar{d}}_{\bar{b}\bar{c}}]_{\mathcal{A}}$$
$$+[f^{i}_{\bar{a}}t^{ab}_{ik}t^{\bar{j}\bar{k}}_{cd}v^{\bar{j}\bar{b}}_{\bar{c}\bar{d}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f^{i}_{\bar{a}}t^{\bar{b}\bar{d}}_{ik}t^{\bar{j}\bar{k}}_{cd}v^{\bar{j}\bar{b}}_{\bar{a}\bar{c}}]_{\mathcal{A}}+[f^{i}_{\bar{a}}t^{\bar{a}\bar{c}}_{il}t^{\bar{j}\bar{k}}_{bc}v^{\bar{j}\bar{k}}_{\bar{l}\bar{b}}]_{\mathcal{A}}+\tfrac{-1}{4}\cdot[f^{i}_{\bar{a}}t^{\bar{b}\bar{c}}_{il}t^{\bar{j}\bar{k}}_{bc}v^{\bar{j}\bar{k}}_{\bar{l}\bar{a}}]_{\mathcal{A}}+\tfrac{-1}{4}\cdot[f^{i}_{\bar{a}}t^{\bar{a}\bar{b}}_{jk}t^{\bar{j}\bar{k}}_{cd}v^{\bar{i}\bar{b}}_{\bar{c}\bar{d}}]_{\mathcal{A}}$$
$$+[f^{i}_{\bar{a}}t^{\bar{b}\bar{c}}_{jk}t^{\bar{i}\bar{k}}_{\bar{a}\bar{d}}v^{\bar{j}\bar{d}}_{\bar{b}\bar{c}}]_{\mathcal{A}}+\tfrac{-1}{4}\cdot[f^{i}_{\bar{a}}t^{\bar{b}\bar{c}}_{jk}t^{\bar{j}\bar{k}}_{\bar{a}\bar{d}}v^{\bar{i}\bar{d}}_{\bar{b}\bar{c}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f^{i}_{\bar{a}}t^{\bar{b}\bar{d}}_{jk}t^{\bar{i}\bar{k}}_{cd}v^{\bar{j}\bar{c}}_{\bar{a}\bar{b}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[f^{i}_{\bar{a}}t^{\bar{b}\bar{d}}_{jk}t^{\bar{j}\bar{k}}_{cd}v^{\bar{i}\bar{b}}_{\bar{a}\bar{c}}]_{\mathcal{A}}$$
$$+\tfrac{1}{2}\cdot[f^{i}_{\bar{a}}t^{\bar{b}\bar{d}}_{jk}t^{\bar{j}\bar{k}}_{cd}v^{\bar{i}\bar{c}}_{\bar{a}\bar{b}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f^{i}_{\bar{a}}t^{\bar{a}\bar{c}}_{kl}t^{\bar{j}\bar{l}}_{\bar{b}\bar{c}}v^{\bar{i}\bar{j}}_{\bar{k}\bar{b}}]_{\mathcal{A}}+[f^{i}_{\bar{a}}t^{\bar{b}\bar{c}}_{kl}t^{\bar{i}\bar{j}}_{\bar{a}\bar{c}}v^{\bar{j}\bar{b}}_{\bar{k}\bar{l}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f^{i}_{\bar{a}}t^{\bar{b}\bar{c}}_{kl}t^{\bar{j}\bar{l}}_{\bar{a}\bar{c}}v^{\bar{i}\bar{k}}_{\bar{j}\bar{b}}]_{\mathcal{A}}$$
$$+\tfrac{-1}{4}\cdot[f^{i}_{\bar{a}}t^{\bar{b}\bar{c}}_{kl}t^{\bar{i}\bar{j}}_{\bar{b}\bar{c}}v^{\bar{j}\bar{a}}_{\bar{k}\bar{l}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[f^{i}_{\bar{a}}t^{\bar{b}\bar{c}}_{kl}t^{\bar{j}\bar{l}}_{\bar{b}\bar{c}}v^{\bar{i}\bar{j}}_{\bar{k}\bar{a}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[f^{i}_{\bar{a}}t^{\bar{b}\bar{c}}_{kl}t^{\bar{j}\bar{l}}_{\bar{b}\bar{c}}v^{\bar{i}\bar{k}}_{\bar{j}\bar{a}}]_{\mathcal{A}}+[f^{\bar{a}}_{\bar{b}}f^{\bar{a}}_{\bar{c}}t^{\bar{c}}_{i}t^{i}_{\bar{b}}]_{\mathcal{A}}$$
$$+\tfrac{1}{2}\cdot[f^{\bar{a}}_{\bar{b}}f^{\bar{a}}_{\bar{c}}t^{cd}_{ij}t^{ij}_{\bar{b}\bar{d}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[f^{\bar{a}}_{\bar{b}}f^{\bar{c}}_{\bar{d}}t^{ac}_{ij}t^{ij}_{\bar{b}\bar{d}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f^{\bar{a}}_{\bar{b}}t^{a}_{\bar{j}}t^{i}_{\bar{c}}v^{\bar{i}\bar{b}}_{\bar{j}\bar{c}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[f^{\bar{a}}_{\bar{b}}t^{a}_{\bar{j}}t^{ij}_{cd}v^{\bar{i}\bar{b}}_{\bar{c}\bar{d}}]_{\mathcal{A}}$$
$$+\tfrac{-1}{1}\cdot[f^{\bar{a}}_{\bar{b}}t^{c}_{\bar{j}}t^{i}_{b}v^{\bar{i}\bar{c}}_{\bar{j}\bar{a}}]_{\mathcal{A}}+[f^{\bar{a}}_{\bar{b}}t^{c}_{\bar{j}}t^{ij}_{\bar{b}\bar{d}}v^{\bar{i}\bar{c}}_{\bar{a}\bar{d}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f^{\bar{a}}_{\bar{b}}t^{a}_{\bar{k}}t^{ij}_{bc}v^{\bar{i}\bar{j}}_{\bar{k}\bar{c}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[f^{\bar{a}}_{\bar{b}}t^{c}_{\bar{k}}t^{ij}_{bc}v^{\bar{i}\bar{j}}_{\bar{k}\bar{a}}]_{\mathcal{A}}$$
$$+\tfrac{-1}{1}\cdot[f^{\bar{a}}_{\bar{b}}t^{i}_{\bar{b}}t^{ac}_{jk}v^{\bar{i}\bar{c}}_{\bar{j}\bar{k}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[f^{\bar{a}}_{\bar{b}}t^{\bar{j}}_{\bar{b}}t^{cd}_{ij}v^{\bar{i}\bar{a}}_{\bar{c}\bar{d}}]_{\mathcal{A}}+\tfrac{1}{2}\cdot[f^{\bar{a}}_{\bar{b}}t^{i}_{\bar{c}}t^{ac}_{jk}v^{\bar{i}\bar{b}}_{\bar{j}\bar{k}}]_{\mathcal{A}}+[f^{\bar{a}}_{\bar{b}}t^{\bar{j}}_{\bar{d}}t^{ac}_{ij}v^{\bar{i}\bar{d}}_{\bar{b}\bar{c}}]_{\mathcal{A}}+\tfrac{1}{4}\cdot[f^{\bar{a}}_{\bar{b}}t^{ac}_{ij}t^{ij}_{de}v^{\bar{b}\bar{c}}_{\bar{d}\bar{e}}]_{\mathcal{A}}$$
$$+\tfrac{1}{4}\cdot[f^{\bar{a}}_{\bar{b}}t^{cd}_{ij}t^{ij}_{\bar{b}\bar{e}}v^{\bar{a}\bar{e}}_{\bar{c}\bar{d}}]_{\mathcal{A}}+\tfrac{-2}{1}\cdot[f^{\bar{a}}_{\bar{b}}t^{ac}_{jk}t^{ik}_{\bar{b}\bar{d}}v^{\bar{i}\bar{c}}_{\bar{j}\bar{d}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f^{\bar{a}}_{\bar{b}}t^{ad}_{jk}t^{ik}_{cd}v^{\bar{i}\bar{b}}_{\bar{j}\bar{c}}]_{\mathcal{A}}+\tfrac{-1}{1}\cdot[f^{\bar{a}}_{\bar{b}}t^{cd}_{jk}t^{ik}_{\bar{b}\bar{d}}v^{\bar{i}\bar{c}}_{\bar{j}\bar{a}}]_{\mathcal{A}}$$

$$+\tfrac{1}{2}\cdot[f_b^a t_{kl}^{ac} t_{bc}^{ij} v_{kl}^{ij}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_i^a t_{cd}^{jk} v_{ab}^{ic} v_{bd}^{jk}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_i^d t_c^j v_{ab}^{ic} v_{ab}^{jd}]_{\mathcal{A}} + \tfrac{1}{4}\cdot[t_i^d t_{cd}^{jk} v_{ab}^{ic} v_{ab}^{jk}]_{\mathcal{A}}$$
$$+\tfrac{-1}{1}\cdot[t_j^a t_c^k v_{ab}^{ij} v_{bc}^{ik}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_j^c t_c^k v_{ab}^{ij} v_{ab}^{ik}]_{\mathcal{A}} + [t_j^c t_{bd}^{ik} v_{jb}^{ia} v_{ad}^{kc}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_j^d t_c^j v_{ab}^{ic} v_{ab}^{id}]_{\mathcal{A}} + \tfrac{1}{4}\cdot[t_j^e t_{cd}^{ij} v_{ab}^{ie} v_{cd}^{ab}]_{\mathcal{A}}$$
$$+\tfrac{-1}{1}\cdot[t_k^a t_c^j v_{kb}^{ij} v_{ab}^{ic}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_k^a t_c^k v_{ab}^{ij} v_{bc}^{ij}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_k^a t_{bc}^{il} v_{jk}^{ia} v_{bc}^{jl}]_{\mathcal{A}} + \tfrac{-1}{4}\cdot[t_k^a t_{cd}^{ij} v_{kb}^{ij} v_{cd}^{ab}]_{\mathcal{A}} + [t_k^a t_{cd}^{jk} v_{bd}^{ij} v_{ab}^{ic}]_{\mathcal{A}}$$
$$+[t_k^b t_c^i v_{jk}^{ia} v_{ac}^{jb}]_{\mathcal{A}} + [t_k^c t_b^i v_{jb}^{ia} v_{ka}^{jc}]_{\mathcal{A}} + \tfrac{1}{4}\cdot[t_k^c t_c^k v_{ab}^{ij} v_{ab}^{ij}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_k^c t_{bc}^{il} v_{jk}^{ia} v_{ab}^{jl}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_k^c t_{bd}^{ik} v_{jb}^{ia} v_{ad}^{jc}]_{\mathcal{A}}$$
$$+\tfrac{-1}{2}\cdot[t_k^d t_{bc}^{ij} v_{bc}^{ia} v_{ka}^{jd}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_k^d t_{cd}^{jk} v_{ab}^{ij} v_{ab}^{ic}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_l^a t_b^i v_{jk}^{ia} v_{lb}^{jk}]_{\mathcal{A}} + \tfrac{1}{4}\cdot[t_l^a t_{bc}^{ij} v_{kl}^{ij} v_{bc}^{ka}]_{\mathcal{A}} + [t_l^a t_{bc}^{ik} v_{jb}^{ia} v_{lc}^{jk}]_{\mathcal{A}}$$
$$+\tfrac{1}{4}\cdot[t_l^a t_{bc}^{il} v_{jk}^{ia} v_{bc}^{jk}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_l^b t_b^i v_{jk}^{ia} v_{la}^{jk}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_l^b t_{ac}^{ij} v_{ka}^{ij} v_{lc}^{kb}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_l^c t_{bc}^{ik} v_{jb}^{ia} v_{la}^{jk}]_{\mathcal{A}}$$
$$+\tfrac{1}{2}\cdot[t_l^c t_{bc}^{il} v_{jk}^{ia} v_{ab}^{jk}]_{\mathcal{A}} + \tfrac{1}{4}\cdot[t_m^b t_{ab}^{ij} v_{kl}^{ij} v_{ma}^{kl}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_b^i t_{jk}^{cd} v_{jb}^{ia} v_{cd}^{ka}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_b^i t_{kl}^{ac} v_{jb}^{ia} v_{kl}^{jc}]_{\mathcal{A}}$$
$$+\tfrac{1}{4}\cdot[t_b^i t_{lm}^{ab} v_{jk}^{ia} v_{lm}^{jk}]_{\mathcal{A}} + [t_c^i t_{kl}^{ab} v_{jk}^{ia} v_{lc}^{jb}]_{\mathcal{A}} + [t_c^i t_{kl}^{bc} v_{jk}^{ia} v_{la}^{jb}]_{\mathcal{A}} + \tfrac{1}{4}\cdot[t_c^j t_{ij}^{de} v_{ab}^{ic} v_{de}^{ab}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_c^j t_{ik}^{ad} v_{ab}^{ic} v_{kb}^{jd}]_{\mathcal{A}}$$
$$+\tfrac{1}{4}\cdot[t_c^j t_{kl}^{ab} v_{ab}^{ij} v_{ab}^{ic}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_c^k t_{jk}^{ad} v_{ab}^{ic} v_{jb}^{id}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_c^k t_{jl}^{ab} v_{ab}^{ij} v_{lc}^{ik}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_c^k t_{jl}^{ac} v_{ab}^{ij} v_{lb}^{ik}]_{\mathcal{A}}$$
$$+\tfrac{1}{4}\cdot[t_c^l t_{kl}^{ab} v_{kc}^{ij} v_{ab}^{ij}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_c^l t_{kl}^{ac} v_{kb}^{ij} v_{ab}^{ij}]_{\mathcal{A}} + \tfrac{1}{4}\cdot[t_d^i t_{jk}^{bc} v_{jk}^{ia} v_{bc}^{ad}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_d^k t_{ij}^{ac} v_{ab}^{ij} v_{bd}^{kc}]_{\mathcal{A}}$$
$$+\tfrac{1}{4}\cdot[t_d^k t_{ij}^{cd} v_{ab}^{ij} v_{ab}^{kc}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_d^k t_{jk}^{ac} v_{ab}^{ij} v_{bd}^{ic}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_d^k t_{jk}^{cd} v_{ab}^{ij} v_{ab}^{ic}]_{\mathcal{A}} + \tfrac{1}{4}\cdot[t_{ij}^{ad} t_{cd}^{kl} v_{ab}^{ij} v_{bc}^{kl}]_{\mathcal{A}}$$
$$+\tfrac{1}{16}\cdot[t_{ij}^{cd} t_{cd}^{kl} v_{ab}^{ij} v_{ab}^{kl}]_{\mathcal{A}} + \tfrac{1}{16}\cdot[t_{ij}^{ef} t_{cd}^{ij} v_{cd}^{ab} v_{ef}^{ab}]_{\mathcal{A}} + [t_{ik}^{ad} t_{ce}^{jk} v_{ab}^{ic} v_{be}^{jd}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_{ik}^{de} t_{ce}^{jk} v_{ab}^{ic} v_{ab}^{jd}]_{\mathcal{A}}$$
$$+\tfrac{1}{2}\cdot[t_{il}^{ad} t_{cd}^{jk} v_{ab}^{ic} v_{lb}^{jk}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_{jk}^{ab} t_{cd}^{il} v_{jk}^{ia} v_{cd}^{lb}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_{jk}^{ad} t_{ce}^{jk} v_{ab}^{ic} v_{be}^{id}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_{jk}^{ae} t_{cd}^{ik} v_{jb}^{ie} v_{cd}^{ab}]_{\mathcal{A}}$$
$$+\tfrac{1}{2}\cdot[t_{jk}^{bd} t_{cd}^{il} v_{jk}^{ia} v_{ac}^{lb}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_{jk}^{cd} t_{be}^{ik} v_{jb}^{ia} v_{cd}^{ae}]_{\mathcal{A}} + \tfrac{1}{4}\cdot[t_{jk}^{de} t_{bc}^{ik} v_{bc}^{ia} v_{de}^{ja}]_{\mathcal{A}} + \tfrac{1}{4}\cdot[t_{jk}^{de} t_{ce}^{jk} v_{ab}^{ic} v_{ab}^{id}]_{\mathcal{A}}$$
$$+\tfrac{-1}{4}\cdot[t_{jl}^{ab} t_{cd}^{kl} v_{ab}^{ij} v_{cd}^{ik}]_{\mathcal{A}} + [t_{jl}^{ac} t_{bd}^{ik} v_{jb}^{ia} v_{ld}^{kc}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_{jl}^{ad} t_{cd}^{kl} v_{ab}^{ij} v_{bc}^{ik}]_{\mathcal{A}} + [t_{jl}^{cd} t_{bd}^{ik} v_{jb}^{ia} v_{la}^{kc}]_{\mathcal{A}}$$
$$+\tfrac{-1}{4}\cdot[t_{jl}^{cd} t_{cd}^{kl} v_{ab}^{ij} v_{ab}^{ik}]_{\mathcal{A}} + \tfrac{1}{8}\cdot[t_{kl}^{ab} t_{cd}^{ij} v_{kl}^{ij} v_{cd}^{ab}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_{kl}^{ab} t_{cd}^{il} v_{jk}^{ia} v_{cd}^{jb}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_{kl}^{ab} t_{cd}^{jl} v_{kd}^{ij} v_{ab}^{ic}]_{\mathcal{A}}$$
$$+\tfrac{1}{16}\cdot[t_{kl}^{ab} t_{cd}^{kl} v_{ab}^{ij} v_{cd}^{ij}]_{\mathcal{A}} + [t_{kl}^{ac} t_{bd}^{il} v_{jb}^{ia} v_{kd}^{jc}]_{\mathcal{A}} + \tfrac{-1}{1}\cdot[t_{kl}^{ad} t_{cd}^{jl} v_{kb}^{ij} v_{ab}^{ic}]_{\mathcal{A}} + \tfrac{1}{4}\cdot[t_{kl}^{ad} t_{cd}^{kl} v_{ab}^{ij} v_{bc}^{ij}]_{\mathcal{A}}$$
$$+\tfrac{-1}{2}\cdot[t_{kl}^{bc} t_{ad}^{ij} v_{ka}^{ij} v_{bc}^{ld}]_{\mathcal{A}} + [t_{kl}^{bd} t_{cd}^{il} v_{jk}^{ia} v_{ac}^{jb}]_{\mathcal{A}} + \tfrac{1}{8}\cdot[t_{kl}^{cd} t_{ab}^{ij} v_{ab}^{ij} v_{cd}^{kl}]_{\mathcal{A}} + [t_{kl}^{cd} t_{bd}^{il} v_{jb}^{ia} v_{ka}^{jc}]_{\mathcal{A}} + \tfrac{1}{16}\cdot[t_{kl}^{cd} t_{cd}^{kl} v_{ab}^{ij} v_{ab}^{ij}]_{\mathcal{A}}$$
$$+\tfrac{-1}{1}\cdot[t_{km}^{ac} t_{bc}^{il} v_{jk}^{ia} v_{mb}^{jl}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_{km}^{bc} t_{bc}^{il} v_{jk}^{ia} v_{ma}^{jl}]_{\mathcal{A}} + \tfrac{1}{2}\cdot[t_{lm}^{ac} t_{bc}^{ij} v_{kl}^{ij} v_{mb}^{ka}]_{\mathcal{A}} + \tfrac{-1}{2}\cdot[t_{lm}^{ac} t_{bc}^{ik} v_{jb}^{ia} v_{lm}^{jk}]_{\mathcal{A}}$$
$$+\tfrac{-1}{2}\cdot[t_{lm}^{ac} t_{bc}^{im} v_{jk}^{ia} v_{lb}^{jk}]_{\mathcal{A}} + \tfrac{1}{4}\cdot[t_{lm}^{bc} t_{ac}^{ij} v_{ka}^{ij} v_{lm}^{kb}]_{\mathcal{A}} + \tfrac{1}{4}\cdot[t_{lm}^{bc} t_{bc}^{im} v_{jk}^{ia} v_{la}^{jk}]_{\mathcal{A}} + \tfrac{1}{16}\cdot[t_{mn}^{ab} t_{ab}^{ij} v_{kl}^{ij} v_{mn}^{kl}]_{\mathcal{A}}$$

List of Abbreviations

CAS	complete active space
BCH	Baker–Campbell–Hausdorff (expansion)
BWCC	Brillouin–Wigner coupled-cluster (method)
CC	coupled-cluster (method)
CCSD	coupled-cluster method with singles and doubles
CCSD(T)	coupled-cluster method with singles and doubles and a perturbative triples correction
CI	configuration interaction (method)
EOM	equation-of-motion
FCI	full configuration interaction (method)
HF	Hartree–Fock (method)
LR	linear response
MCSCF	multi-configuration self-consistent field (method)
MkMRCC	multi-reference coupled-cluster method of Mukherjee et al.
MO	molecular orbital
MR	multi-reference
MRexpT	multi-reference exponential wave function (ansatz)
PES	potential energy surface
Q	quadruples (quadruple excitations)
SCF	self-consistent field (method)
SR	single-reference
SRMRCC	multi-reference coupled-cluster based on the single-reference formalism
STL	standard template library (for C++)
SUMRCC	state-universal multi-reference coupled-cluster (method)
T	triples (triple excitations)
TCE	Tensor Contraction Engine
UML	unified modeling language
VUMRCC	valence-universal multi-reference coupled-cluster (method)

List of Figures

List of Tables

List of Listings

List of Algorithms

Bibliography

[1] T. D. Crawford, H. F. Schaefer III, Rev. Comput. Chem. **14**, 33 (2000).

[2] R. J. Bartlett, J. Phys. Chem. **93**, 1697 (1989).

[3] O. Christiansen, Theor. Chem. Acc. **116**, 106 (2006).

[4] R. J. Bartlett, M. Musiał, Rev. Mod. Phys. **79**, 291 (2007).

[5] T. Helgaker, W. Klopper, D. P. Tew, Mol. Phys. **106**, 2107 (2008).

[6] F. Coester, Nucl. Phys. **7**, 421 (1958).

[7] F. Coester, H. Kümmel, Nucl. Phys. **17**, 477 (1960).

[8] J. Čížek, J. Chem. Phys. **45**, 4256 (1966).

[9] J. Čížek, Adv. Chem. Phys. **14**, 35 (1969).

[10] J. Čížek, J. Paldus, Int. J. Quantum Chem. **5**, 359 (1971).

[11] R. J. Bartlett, G. D. Purvis III, Int. J. Quantum Chem. **14**, 561 (1978).

[12] G. D. Purvis III, R. J. Bartlett, J. Chem. Phys. **76**, 1910 (1982).

[13] J. A. Pople, R. Krishnan, H. B. Schlegel, J. S. Binkley, Int. J. Quantum Chem. **14**, 545 (1978).

[14] J. Noga, R. J. Bartlett, J. Chem. Phys. **86**, 7041 (1987).

[15] G. E. Scuseria, H. F. Schaefer III, Chem. Phys. Lett. **152**, 382 (1988).

[16] S. A. Kucharski, R. J. Bartlett, Theor. Chim. Acta **80**, 387 (1991).

[17] M. Musiał, S. A. Kucharski, R. J. Bartlett, Chem. Phys. Lett. **320**, 542 (2000).

[18] K. Raghavachari, G. W. Trucks, J. A. Pople, M. Head-Gordon, Chem. Phys. Lett. **157**, 479 (1989).

[19] N. Oliphant, L. Adamowicz, J. Chem. Phys. **94**, 1229 (1991).

[20] V. V. Ivanov, D. I. Lyakh, L. Adamowicz, Phys. Chem. Chem. Phys. **11**, 2355 (2009).

[21] H. J. Monkhorst, Int. J. Quantum Chem. Symp. **11**, 421 (1977).

[22] D. Mukherjee, P. K. Mukherjee, Chem. Phys. **39**, 325 (1979).

[23] H. Koch, P. Jørgensen, J. Chem. Phys. **93**, 3333 (1990).

[24] O. Christiansen, P. Jørgensen, C. Hättig, Int. J. Quantum Chem. **68**, 1 (1998).

[25] J. F. Stanton, R. J. Bartlett, J. Chem. Phys. **98**, 7029 (1993).

[26] A. I. Krylov, Annu. Rev. Phys. Chem. **59**, 433 (2008).

[27] K. Kowalski, P. Piecuch, J. Chem. Phys. **113**, 18 (2000).

[28] G. D. Purvis III, R. J. Bartlett, J. Chem. Phys. **75**, 1284 (1981).

[29] G. E. Scuseria, T. J. Lee, H. F. Schaefer III, Chem. Phys. Lett. **130**, 236 (1986).

[30] G. E. Scuseria, A. C. Scheiner, T. J. Lee, J. E. Rice, H. F. Schaefer III, J. Chem. Phys. **86**, 2881 (1987).

[31] T. J. Lee, J. E. Rice, Chem. Phys. Lett. **150**, 406 (1988).

[32] G. E. Scuseria, C. L. Janssen, H. F. Schaefer III, J. Chem. Phys. **89**, 7382 (1988).

[33] G. E. Scuseria, H. F. Schaefer III, J. Chem. Phys. **90**, 3700 (1989).

[34] J. F. Stanton, J. Gauss, J. D. Watts, R. J. Bartlett, J. Chem. Phys. **94**, 4334 (1991).

[35] C. Hampel, K. A. Peterson, H.-J. Werner, Chem. Phys. Lett. **190**, 1 (1992).

[36] P. J. Knowles, C. Hampel, H.-J. Werner, J. Chem. Phys. **99**, 5219 (1993).

[37] S. Hirata, J. Phys. Chem. A **107**, 9887 (2003).

[38] A. A. Auer, G. Baumgartner, D. E. Bernholdt, A. Bibireata, V. Choppella, D. Cociorva, X. Gao, R. Harrison, S. Krishnamoorthy, S. Krishnan, C.-C. Lam, Q. Lu, M. Nooijen, R. Pitzer, J. Ramanujam, P. Sadayappan, A. Sibiryakov, Mol. Phys. **104**, 211 (2006).

[39] T. Shiozaki, M. Kamiya, S. Hirata, E. F. Valeev, Phys. Chem. Chem. Phys. **10**, 3358 (2008).

[40] T. Shiozaki, M. Kamiya, S. Hirata, E. F. Valeev, J. Chem. Phys. **129**, 071101 (2008).

[41] M. Kállay, P. R. Surján, J. Chem. Phys. **115**, 2945 (2001).

[42] M. Kállay, P. G. Szalay, P. R. Surján, J. Chem. Phys. **117**, 980 (2002).

[43] M. Hanrath, J. Chem. Phys. **123**, 84102 (2005).

[44] M. Hanrath, Chem. Phys. Lett. **420**, 426 (2006).

[45] M. Hanrath, J. Chem. Phys. **128**, 154118 (2008).

[46] M. Hanrath, Mol. Phys. **106**, 1949 (2008).

[47] M. Hanrath, A. Engels-Putzka, Mol. Phys. **107**, 143 (2009).

[48] A. Engels-Putzka, M. Hanrath, Theor. Chem. Acc. **122**, 197 (2009).

[49] A. Engels-Putzka, M. Hanrath, Journal of Molecular Structure: THEOCHEM **902**, 59 (2009).

[50] Silicon Graphics, *Standard Template Library Programmer's Guide*, URL `http://www.sgi.com/tech/stl/`.

[51] Object Management Group, *Unified Modeling Language*, URL `http://www.uml.org/`.

[52] T. Helgaker, P. Jørgensen, J. Olsen, *Modern Electronic Structure Theory*, first. Ed., Wiley (2000).

[53] F. E. Harris, H. J. Monkhorst, D. L. Freeman, *Algebraic and Diagrammatic Methods in Many-Fermion Theory*, Oxford University Press (1992).

[54] M. Nooijen, K. R. Shamasundar, D. Mukherjee, Mol. Phys. **103**, 2277 (2005).

[55] M. Hanrath, Chem. Phys. **356**, 31 (2009).

[56] J. A. Pople, J. S. Binkley, R. Seeger, Int. J. Quantum Chem. Symp. **10**, 1 (1976).

[57] W. Duch, G. H. F. Diercksen, J. Chem. Phys. **101**, 3018 (1994).

[58] P. R. Taylor, Lecture Notes on Quantum Chemistry **64**, 125 (1994).

[59] R. J. Bartlett, Ann. Rev. Phys. Chem. **32**, 359 (1981).

[60] K. A. Brueckner, Phys. Rev. **100**, 36 (1955).

[61] J. Goldstone, Proc. Roy. Soc. A **239**, 267 (1956).

[62] G. C. Wick, Phys. Rev. **80**, 268 (1950).

[63] F. E. Harris, B. Jeziorski, H. J. Monkhorst, Phys. Rev. A **23**, 1632 (1981).

[64] B. Jeziorski, J. Paldus, J. Chem. Phys. **88**, 5673 (1988).

[65] X. Li, J. Paldus, J. Chem. Phys. **101**, 8812 (1994).

[66] M. Nooijen, J. Chem. Phys. **104**, 2638 (1996).

[67] D. Mukherjee, R. K. Moitra, A. Mukhopadhyay, Mol. Phys. **30**, 1861 (1975).

[68] D. Mukherjee, R. K. Moitra, A. Mukhopadhyay, Mol. Phys. **33**, 955 (1977).

[69] I. Lindgren, Int. J. Quantum Chem. Symp. **12**, 33 (1978).

[70] B. Jeziorski, H. J. Monkhorst, Phys. Rev. A **24**, 1668 (1981).

[71] X. Li, J. Paldus, J. Chem. Phys. **119**, 5320 (2003).

[72] X. Li, J. Paldus, J. Chem. Phys. **119**, 5346 (2003).

[73] M. Hanrath, Theor. Chem. Acc. **121**, 187 (2008).

[74] U. S. Mahapatra, B. Datta, D. Mukherjee, Mol. Phys. **94**, 157 (1998).

[75] U. S. Mahapatra, B. Datta, D. Mukherjee, J. Chem. Phys. **110**, 6171 (1999).

[76] J. Mášik, I. Hubač, Adv. Quantum Chem. **31**, 75 (1999).

[77] J. Mášik, I. Hubač, P. Mach, J. Chem. Phys. **108**, 6571 (1998).

[78] J. Pittner, J. Chem. Phys. **118**, 10876 (2003).

[79] J. Pittner, X. Z. Li, J. Paldus, Mol. Phys. **103**, 2239 (2005).

[80] I. Hubač, J. Pittner, P. Carsky, J. Chem. Phys. **112**, 8779 (2000).

[81] L. Kong, Int. J. Quant. Chem. **109**, 441 (2008).

[82] F. A. Evangelista, W. D. Allen, H. F. Schaefer III, J. Chem. Phys. **125**, 154113 (2006).

[83] M. Kállay, P. R. Surján, J. Chem. Phys. **113**, 1359 (2000).

[84] J. Olsen, J. Chem. Phys. **113**, 7140 (2000).

[85] S. Hirata, R. J. Bartlett, Chem. Phys. Lett. **321**, 216 (2000).

[86] P. J. Knowles, N. C. Handy, Chem. Phys. Lett. **111**, 315 (1984).

[87] J. Olsen, B. O. Roos, P. Jørgensen, H. J. A. Jensen, J. Chem. Phys. **89**, 2185 (1988).

[88] P. J. Knowles, Chem. Phys. Lett. **155**, 513 (1989).

[89] M. Hanrath, personal communication.

[90] S. A. Kucharski, R. J. Bartlett, Adv. Quantum Chem. **18**, 281 (1986).

[91] S. Hirata, Theor. Chem. Acc. **116**, 2 (2006).

[92] C. L. Janssen, H. F. Schaefer III, Theor. Chim. Acta **79**, 1 (1991).

[93] P. Jankowski, B. Jeziorski, J. Chem. Phys. **111**, 1857 (1999).

[94] M. Nooijen, V. Lotrich, J. Mol. Struct. THEOCHEM **547**, 253 (2001).

[95] I. Berente, P. G. Szalay, J. Gauss, J. Chem. Phys. **117**, 7872 (2002).

[96] J. Paldus, H. C. Wong, Comput. Phys. Commun. **6**, 1 (1973).

[97] J. Paldus, H. C. Wong, Comput. Phys. Commun. **6**, 9 (1973).

[98] U. Kaldor, J. Comp. Phys. **20**, 432 (1976).

[99] F. E. Harris, Int. J. Quantum Chem. **75**, 593 (1999).

[100] D. I. Lyakh, V. V. Ivanov, L. Adamowicz, J. Chem. Phys. **122**, 24108 (2005).

[101] A. D. Bochevarov, C. D. Sherrill, J. Chem. Phys. **121**, 3374 (2004).

[102] M. Hanrath, personal communication.

[103] P. Pulay, Chem. Phys. Lett. **73**, 393 (1980).

[104] M. Hanrath, Chem. Phys. Lett. **466**, 240 (2008).

[105] M. Wladyslawski, M. Nooijen, Adv. Quantum Chem. **49**, 1 (2005).

[106] E. R. Gansner, S. C. North, Software - Practice and Experience **30**, 1203 (1999), see also www.graphviz.org.

[107] E. R. Gansner, E. Koutsofios, S. C. North, K. Vo, IEEE Transactions on Software Engineering **19**, 214 (1993).

[108] H.-J. Werner, P. J. Knowles, R. Lindh, F. R. Manby, M. Schütz, *et al.*, *MOLPRO, a package of ab initio programs designed by H.-J. Werner and P. J. Knowles version 2006*, Tech. Rport, University of Birmingham (2006).

[109] J. Friedrich, M. Hanrath, M. Dolg, J. Chem. Phys. **126**, 154110 (2007).

[110] N. Bourbaki, *Lie Groups and Lie Algebras*, Springer.

Acknowledgments

The work embodied in this thesis has been carried out at the Institute for Theoretical Chemistry under the supervision of Prof. Dr. Michael Dolg and scientific guidance of Dr. Michael Hanrath. I want to express my gratitude towards them and all the other persons who helped me in some way to finish this thesis.

I am very grateful to Prof. Dolg for giving me the opportunity to do my PhD in his group and for providing an excellent working environment. I have also many reasons to thank Dr. Hanrath, who initiated the project this work is a part of. He introduced me to the interesting subject of (multi-reference) coupled-cluster theory – in particular his MRexpT ansatz – and to programming in C++. I also learned a lot from him about Theoretical Chemistry and computers in general, the way computers work and how to write an efficient program. We had many fruitful discussions and he patiently answered my questions. Finally, he also read several versions of my manuscript and made valuable suggestions. I would also like to thank Prof. U. Deiters for accepting to review this thesis, despite its very theoretical nature.

Furthermore, I thank all my present and former colleagues at the Institute for Theoretical Chemistry for accepting me as a colleague and for the very good atmosphere in our group. In particular I would like to thank my "office-mates" Dr. Joachim Friedrich and Jonas Wiebke for sharing with me the ups and dows of scientific work, for interesting discussions, for encouragement and for helping me with the debugging of some of my programs (special thanks to J.F. for helping me getting started). J.W. also read parts of this thesis and made helpful comments.

I am also grateful to the people who introduced me to the subject of Theoretical Chemistry during my studies at Bonn University.

Part of this work was supported by the Deutsche Forschungsgemeinschaft through a research grant to M. Hanrath (HA 5116/1-1), which I gratefully acknowledge.

I feel deep gratitude towards my parents, who always supported me and encouraged me to follow my interests. Last but not least I thank my husband Jens Putzka for his practical and emotional support, for his interest in my work and willingness to discuss it with me, and for carefully (and critically) reading several versions of this thesis. Without him, the last years certainly would have been much more difficult for me.

Erklärung

Ich versichere, dass ich die von mir vorgelegte Dissertation selbständig angefertigt, die benutzten Quellen und Hilfsmittel vollständig angegeben und die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken im Wortlaut oder dem Sinn nach entnommen sind, in jedem Einzelfall als Entlehnung kenntlich gemacht habe; dass diese Dissertation noch keiner anderen Fakultät oder Universität zur Prüfung vorgelegen hat; dass sie noch nicht veröffentlicht worden ist sowie, dass ich eine solche Veröffentlichung vor Abschluss des Promotionsverfahrens nicht vornehmen werde. Die Bestimmungen dieser Promotionsordnung sind mir bekannt. Die von mir vorgelegte Dissertation ist von Prof. Dr. M. Dolg betreut worden.