Inaugural - Dissertation

zur Erlangung der Doktorwürde der
Naturwissenschaftlich-Mathematischen Gesamtfakultät
der Ruprecht-Karls-Universität Heidelberg

The Algebraic Diagrammatic Construction Method For High Performance Computing Environments

Using an Atomic Orbital Representation

vorgelegt von

Maximilien Alexandre Ambroise
aus Differdingen, Luxemburg

Juli 2021

Gutachter:
Prof. Dr. Andreas Dreuw
...

Licensing

Notation for Orbital Representations

μ, ν, γ, λ, ...	atomic orbitals
i, j, k, l, ...	occupied molecular orbitals
a, b, c, d, ...	virtual molecular orbitals
$\underline{i}$, $\underline{j}$, $\underline{k}$, $\underline{l}$, ...	local occupied molecular orbitals
$\overline{a}$, $\overline{b}$, $\overline{c}$, $\overline{d}$, ...	local virtual molecular orbitals
I, J, K, L, ...	occupied molecular spin orbitals
A, B, C, D, ...	virtual molecular spin orbitals
$\underline{\mu}$, $\underline{\nu}$, $\underline{\lambda}$, $\underline{\sigma}$, ...	occupied projected atomic orbitals
$\overline{\mu}$, $\overline{\nu}$, $\overline{\lambda}$, $\overline{\sigma}$, ...	virtual projected atomic orbitals

Abbreviations

Contents

List of Figures

List of Tables

Introduction

This is the introduction. Talk about sparsity. Show sparse matrix. How does sparsity arise, what can we do with it, where else does it emerge? State of arts: adc now, adc in the future

Exact solution: three body problem Helium: cannot be solved exactly: approximate methods. finite grid, finite basis solutions ground state excited state computational resources

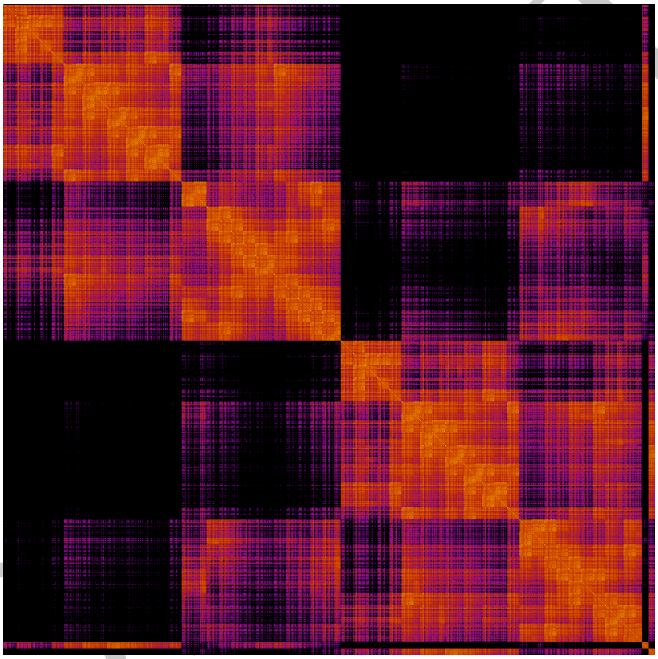


Figure 1: Adenine-guanine fock matrix Hartree FOck cc-pVTZ

Part I

Chapter 1

Theory: The Basics

Over the last decades, quantum chemistry has emerged as a crucial tool for investigating a wide variety of problems in chemistry. This, in combination with the increasing performance and widespread use of computers, has spawned a whole new branch of chemistry known as *computational chemistry*. The field of computational chemistry uses methods developed in *theoretical chemistry* and incorporates them into efficient programs. , quantum chemical methods are routinely applied to assist in solving problems related to chemical structure, reactivity and spectroscopy. However, one of the main problems in computational chemistry is choosing a suitable level of theory for a given problem - the best choice is always a trade-off between speed and accuracy, and requires intricate knowledge of the methods' theoretical and computational limits. This chapter summarizes the core concepts of quantum chemistry and its most important methods. It is by no means complete, and the reader is referred to the original text book resourcesfor more details [8, 9, 1, 10, 2].

1.1 Describing Dynamics in a Molecular System

In order to describe a molecular system, one needs to decide

- what the fundamental particles are
- what forces are governing them
- what the starting conditions are
- and what form the dynamical equations take.

The choice of particles dictates what properties the model is ultimately able to describe. For example, force field methods use atoms as the indivisible unit, which is sufficient to describe the molecular structure and dynamics of large molecules such as proteins, but does not provide any information on electron distribution. Using electrons and nuclei as the fundamental particles allows to get a better picture of the electron density and how it reacts to external perturbation, which is important for studies on reactivity and spectroscopic constants. To describe radioactive decay, it is necessary to further divide the nucleus into protons and neutrons.

Smaller subdivisions lead to a stricter limit on the size of molecules that can be treated. Force field methods may describe the dynamics of molecules with several tens of thousands of atoms, while a finer grained method involving electrons can often only describe molecules one to two orders of magnitudes smaller depending on the approximation used.

The mathematical form of the dynamical equations is determined by the size and mass of the particles. They can be divided into four regimes (Figure). Atoms are sufficiently heavy and slow that their trajectories can be described using classical (Newtonian) mechanics. The time evolution of the positions $\mathbf{r}$ of the atoms in a potential $\mathbf{V}$ can be written as

$$-\frac{\partial \mathbf{V}}{\partial t} = m\frac{\partial^2 \mathbf{r}}{\partial t} \tag{1.1}$$

which is another form of Newton's second law $\mathbf{F} = m\mathbf{a}$. The potential $\mathbf{V}$ is also treated classically as the sum of contributions of particle-particle interactions.

For objects with velocities approaching the speed of light, it is necessary to introduce *relativistic effects*. Mass then becomes a function of velocity

$$m = \frac{m_0}{\sqrt{1 - v^2/c^2}} \tag{1.2}$$

Classical Newtonian mechanics cannot be applied to very light particles such as electrons, and quantum effects need to taken into considerations. For non-relativistic velocities, the dynamics are governed by the time-dependent Schrödinger equation:

$$\hat{H}\Psi(\mathbf{r}, t) = i\frac{\partial \Psi(\mathbf{r}, t)}{\partial t} \tag{1.3}$$

where $\hat{H}$ is the Hamiltonian operator which is a sum of the kinetic and potential energy operators

$$\hat{H} = \hat{T} + \hat{V} \tag{1.4}$$

$$\hat{T} = -\frac{1}{2m}\nabla^2 \tag{1.5}$$

and Ψ is the wave function of the system, which is obtained as the solution to the Schrödinger Equation, and gives the probability of finding a particle at position $\mathbf{r}$ at time t. Here, atomic units are assumed.

For electrons moving at relativistic speed, for example in the core orbitals of super-heavy atoms like Uranium, the Hamiltonian takes a more complex form, and the Schrödinger equation is then known as the *Dirac* equation:

$$\hat{H}_{Dirac} = \left(c\boldsymbol{\alpha}\hat{p} + \boldsymbol{\beta}mc^2\right) + \hat{V} \tag{1.6}$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are 4×4 matrices. The relativistic wave function therefore has four components which are traditionally called the small and large components. Here, only the solutions and approximations to the non-relativistic Schrödinger equation will be discussed.

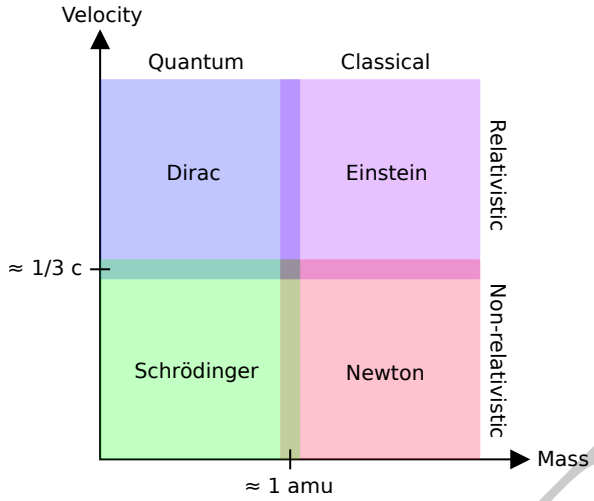


Figure 1.1: Dynamical equations can be divided into four regimes, depending on the size and speed of the individual particles. Adapted form [1]

1.2 The Electronic Schrödinger Equation

> "Where did we get that [Schrödinger's equation] from? It's not possible to derive it from anything you know. It came out of the mind of Schrödinger."
>
> — *Richard Feynman, The Feynman Lectures on Physics*

If one is interested in describing the electron distribution in detail, the Schrödinger Equation (SEQ) is the best starting point. There is no formal, rigorous proof for the Schrödinger equation, similar to how Newton's second law cannot really be "derived", more than simply "motivated" by observation.

As successful as the Schrödinger equation is, finding solutions to it is non-trivial. Different approximations may be applied to the SEQ to solve it more easily, without considerable loss of accuracy.

1.2.1 The Time-Independent Schrödinger Equation

The potential energy operator is the only time-dependent part of the Hamiltonian:

$$\hat{H}(\mathbf{r}, t) = \hat{T}(\mathbf{r}) + \hat{V}(\mathbf{r}, t) \tag{1.7}$$

For systems where the potential is time-independent, e.g. bound systems without external (electromagnetic) perturbation, the Hamiltonian is time-independent as well, which in turn allows to separate space and time variables. It can then be shown that the *time-independent* Schrödinger equation takes the form

$$\hat{H}\Psi(\mathbf{r}) = E\Psi(\mathbf{r}) \tag{1.8}$$

where E is the total energy of the system, and the eigenvalue of the wave function Ψ. The time-dependence is then simply reduced to a phase factor:

$$\Psi(\mathbf{r}, t) = e^{-iEt}\Psi(\mathbf{r}) \tag{1.9}$$

1.2.2 The Born-Oppenheimer Approximation

Atomic nuclei are much heavier than electrons ($m_{proton} \approx 1836 m_{electron}$), and move much slower. To a good approximation, the nuclei can be assumed to be stationary from the point of view of electrons. This is known as the *Born-Oppenheimer approximation*. The total Hamiltonian operator can be written in terms of the kinetic and potential operator of the nuclei (n) and electrons (e) as

$$\hat{H}_{tot} = \hat{T}_n + \hat{T}_e + \hat{V}_{ne} + \hat{V}_{ee} + \hat{V}_{nn} \tag{1.10}$$

In the Born-Oppenheimer approximation, the kinetic energy of the nuclei T_{nn} is neglected, and the nucleus-nucleus potential V_{nn} is taken as a constant, which corresponds to neglecting the coupling between electrons and nuclei. This allows a separation of the electronic and nuclar variables. The remaining terms of Equation 1.10 form the electronic Hamiltonian $\hat{H_{elec}}$. The solutions to the *electronic Schrödinger equation*

$$\hat{H}_{elec}\Psi_{elec}(\mathbf{r}_i, \mathbf{R}_n) = E_{elec}(\mathbf{R}_n)\Psi_{elec}(\mathbf{r}_i, \mathbf{R}_n) \tag{1.11}$$

produce the electronic wave function which depends on the (fixed) *position* $\mathbf{R}_n$ of the nuclei and no longer on the *momentum* of the nuclei. The total energy

$$E_{tot}(\mathbf{R}_n) = E_{elec}(\mathbf{R}_n) + E_{nucl}(\mathbf{R}_n) \tag{1.12}$$

provides a *potential energy surface* (PES) on which the nuclei move. The PES can then be used to solve the nuclear Schrödinger equation to obtain information on vibrational, rotational and translational properties in the molecular system.

From this point onward, the subscript *elec* is dropped, and only the electronic Schrödinger Equation is considered.

1.3 Solutions to the Electronic Schrödinger Equation

1.3.1 Slater Determinants

It is beneficial to first consider the wave function of the single electrons. In single-atom systems, these functions take the form of "atomic orbitals" (AOs). Correspondingly, "molecular orbitals" (MOs) are defined as the single electron wave functions in a molecular system. These spatial orbital functions form the basis of the total electronic wave function.

The Hamiltonian in 1.10 only depends on the spatial coordinates. However, to fully describe an electron, one also needs to consider spin. This is done by introducing two orthonormal spin functions $\alpha(\omega)$ and $\beta(\omega)$ corresponding to spin-up ($\uparrow$) and spin-down ($\downarrow$), with the spin-coordinate ω. For one spatial molecular orbital, this gives two possible *spin-orbitals*

$$\phi(\mathbf{x}) = \begin{cases} \varphi(\mathbf{r})\alpha(\omega) \\ \varphi(\mathbf{r})\beta(\omega) \end{cases} \tag{1.13}$$

where φ are the *spatial orbitals*, and $\mathbf{x}$ are the combined spatial and spin coordinates. The spin orbitals therefore depend on four variables.

To a first approximation, one may consider a molecular system to consist of N *non-interacting*, independent electrons. The Hamiltonian is then written as a sum of one-particle Hamiltonians

$$\mathbf{H} = \sum_{i}^{N} \mathbf{h}_i \tag{1.14}$$

Electron correlation may then be included in some average way by using *effective* one-electron Hamiltonians, which is the basic working idea of the *Hartree* method. The solution to the SEQ can then be expressed as a product of the one electron wave functions

$$\Psi^{HP}(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N) = \phi(\mathbf{x}_1)\phi(\mathbf{x}_2)...\phi(\mathbf{x}_N) \tag{1.15}$$

which is also known as *Hartree product*.

However, the Hartree product does not take into account the *indistinguishability* of electrons. In what is known as the antisymmetry principle, a generilization of the Pauli exclusion principle, the wave function needs to fulfill

$$\Psi(\mathbf{x}_1, \mathbf{x}_2) = -\Psi(\mathbf{x}_2, \mathbf{x}_1) \tag{1.16}$$

upon exchange of any two electrons in the system. This is most easily achieved by using *Slater determinants* (SD). For a molecular system consisting of N electrons distributed over N spin orbitals ϕ_i, the SD takes the form

$$\Psi_{SD}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N) = \frac{1}{\sqrt{N!}} \begin{vmatrix} \phi_I(\mathbf{x}_1) & \phi_J(\mathbf{x}_1) & \ldots & \phi_P(\mathbf{x}_1) \\ \phi_I(\mathbf{x}_2) & \phi_J(\mathbf{x}_2) & \ldots & \phi_P(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_I(\mathbf{x}_N) & \phi_J(\mathbf{x}_N) & \ldots & \phi_P(\mathbf{x}_N) \end{vmatrix} \tag{1.17}$$

Or, using the diagonal of the SD as a short-hand notation

$$\Psi_{SD}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N) = |\phi_I(\mathbf{x}_1), \phi_J(\mathbf{x}_1), \ldots, \phi_K(\mathbf{x}_1)\rangle \tag{1.18}$$

1.3.2 The Fock Space

A more generalized representation of the space of the antisymmetrized electron wave functions can be achieved by introducing the concept of occupation number (ON) vectors in the context of second quantization (Annex). In a system with M possible states (in this case spin molecular orbitals), the ON vectors take the form

$$|\mathbf{k}\rangle = |k_1, k_2, \ldots, k_M\rangle = \begin{cases} 1 & \textit{if } \phi_P \textit{ occupied} \\ 0 & \textit{if } \phi_P \textit{ occupied} \end{cases} \tag{1.19}$$

The occupation number is 1 if ϕ_P is present in the SD, and 0 if not. Together, all possible ON vectors in Equation 1.19 form an orthonormal abstract vector space, known as Fock space. The Fock space formed by N electrons distributed over M orbitals is denoted as $F(M, N)$ with total dimension equal to the binomial coefficient $\binom{M}{N}$. The sum of the occupation numbers in the ON vectors gives the total number of electrons

$$N = \sum_{i}^{M} k_i \tag{1.20}$$

The special fock space $F(0, M)$ contains a single vector known as the vacuum state with

$$|vac\rangle = |0_1, 0_2, \ldots, 0_M\rangle \tag{1.21}$$
$$\langle vac|vac\rangle = 1 \tag{1.22}$$

The ON vectors in $F(M, N)$ can be alternatively expressed in terms of the vacuum state from $F(M, 0)$ using creation operators

$$|\mathbf{k}\rangle = \left[\prod_{P=1}^{M} (a_P^\dagger)^{k_P}\right] |vac\rangle \tag{1.23}$$

In second quantization, the antisymmetry principle of the wave function is guaranteed by the commutator relationship of the annihilation and creation operators a_P and $a_P^\dagger$ which act on the ON vectors.

1.3.3 Exact Solution and Standard Models

The simplest approach to solving the electronic SEQ is by approximating the exact wave function Ψ using a single Slater determinant where the electrons occupy the lowest lying molecular orbitals. The Hartree-Fock method is an example of a *single-determinant method* and finds *the* single best Slater determinant for $|\Psi\rangle$. In Fock space, the best possible determinant is represented by the ON vector where the N lowest lying orbitals are occupied.

As will be discussed in more detail later, a single-determinant treatment of the electronic wave function is insufficient to fully capture *electron correlation*. The electron correlation energy is formerly defined as the difference between the Hartree-Fock energy and the *exact* energy of $|Psi\rangle$

$$E_{correlation} = E_{HF} - E_{exact} \tag{1.24}$$

although the Hartree-Fock wave function does include correlation effects to some degree. In a more general sense, electron correlation is a broad term for any interactions between electrons that make their movement depend on each other, or *correlate* with each other (see Section 1.5).

In order to improve on the HF approximation, it is important to add additional Slater determinants. These SDs can be generated from the HF reference wave function by replacing the occupied MOs ϕ_I in a reference Slater determinant by one or multiple orbitals ϕ_A which were previously unoccupied. This effectively corresponds to exciting one or more electrons from their occupied molecular orbitals $I, J, ..$ to unoccupied, or *virtual* orbitals $A, B, ...$ These excited Slater determinants can be classified by the number of electrons they excite and are often referred to as singles, doubles, triples and so on.

$$|\Phi_0\rangle = |HF\rangle \tag{1.25}$$
$$\left|\Phi_I^A\right\rangle = a_A^\dagger a_I \,|HF\rangle \tag{1.26}$$
$$\left|\Phi_{IJ}^{AB}\right\rangle = a_A^\dagger a_I a_B^\dagger a_J \,|HF\rangle \tag{1.27}$$
$$\left|\Phi_{IJK}^{ABC}\right\rangle = a_A^\dagger a_I a_B^\dagger a_J a_C^\dagger a_K \,|HF\rangle \tag{1.28}$$

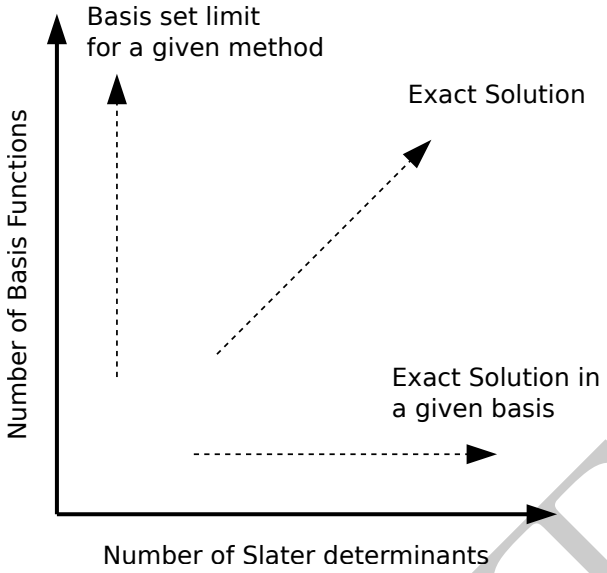


Figure 1.2: The wave function converges to the exact solution in the limit of an infinitely large basis set that spans the molecular orbitals, and an infinitely large correlation space spanned by the Slater Determinants.

$$\ldots \tag{1.29}$$

Alternatively, singles states are known as 1-particle-1-hole (1p-1h or simply p-h) states, doubles as 2-particle-2-hole (2p-2h) and so on, due to the excitation operators effectively creating *holes* in the reference determinant and adding *particles* in higher lying orbitals instead.

The exact solution to the electronic Schrödinger equation is then given by the sum of the Hartree-Fock wave function and all possible ON vectors in $F(M, N)$

$$|\Psi\rangle = c_0 \, |HF\rangle + \sum_{i=1}^{\binom{M}{N}-1} c_i \, |i\rangle \tag{1.30}$$

Equation 1.30 offers a systematic approach to improve on the Hartree Fock method, by increasing (1) the number of Sater determinats and (2) the basis set size M, and gives rise to a hierarchy of methods (Figure 1.2). Correlated electronic structure methods mainly differ by how they determine the expansion coefficients c. For all but the smallest systems, the full $F(M, N)$ Fock space cannot be used in calculations due to the total number of ON vectors which increases binomially with $\binom{M}{N}$, and hence the number of coefficients to be determined. In practice, the Fock space is *truncated* to some degree.

Multi-determinant methods like configuration interaction (CI) or coupled cluster (CC) use the Hartree Fock wave function as the reference from which they generate singles, doubles, triples ... SDs. By truncating at different excitation levels, one gets a hierarchy of CI/CC methods which recover different amounts of correlation energy (e.g. CIS, CISD, CISDT ...). Multi-determinant methods are mainly suited to recover dynamic correlation.

For systems with strong static correlation, additionally, a multi-reference approach is needed. Here, the excited SDs are generated from multiple reference states rather than only from HF. Methods include multi-reference CI (MRCI) and multi-reference CC

(MRCC). The reference states are traditionally obtained from multi-configurational self-consistent-field methods (MCSCF) like the complete active space SCF (CASSCF) or restrcited active space SCF (RASSCF). MCSCF is a combination of HF and CI, which both optimizes the HF molecular coefficients and the CI expansion coefficients. Multi-reference methods methods mainly recover *static correlation.*

1.3.4 The Variational Method

The time-independent Schrödinger equation takes the form of an eigenvalue problem

$$\hat{H}\,|\Psi_i\rangle = E_i\,|\Psi_i\rangle \qquad i = 0, 1, 2, ...\infty \tag{1.31}$$

where the infinite set of exact solutions $|\Psi_i\rangle$ with eigenvalues E_i forms an orthonormal basis

$$\langle\Psi_i|\Psi_j\rangle = \delta_{ij} \tag{1.32}$$

A trial wave function can be expanded in the basis of exact solutions with coefficients c as

$$|\widetilde{\Psi}\rangle = \sum_i a_i\,|\Psi_i\rangle \tag{1.33}$$

The *variation principle* states that the expectation value of the Hamiltonian of an approximate wave function of the form 1.33 is an upper bound to the exact ground state energy. This statement can be expressed as

$$\frac{\langle\widetilde{\Psi}|\,\hat{H}\,|\widetilde{\Psi}\rangle}{\langle\widetilde{\Psi}|\widetilde{\Psi}\rangle} \geq E_0 \tag{1.34}$$

The equality only holds when $|\widetilde{\Psi}\rangle$ is equal to the exact solution $|\Psi_0\rangle$. One can recast the eigenvalue problem 1.31 as a variational optimization problem where the energy is a functional of a trial wave function

$$E[\widetilde{\Psi}] = \frac{\langle\widetilde{\Psi}|\,\hat{H}\,|\widetilde{\Psi}\rangle}{\langle\widetilde{\Psi}|\widetilde{\Psi}\rangle} \tag{1.35}$$

The saddle points of the energy functional then correspond to the exact solutions of the Schrödinger equation. The variational approach to the SEQ provides a powerful tool to solve a wide variety of problems in electronic structure theory.

The trial wave function depends on a set of coefficients c, and hence the energy functional will also depend on these coefficients. In general, determining the coefficients which minimize the functional is very difficult. However, a more simple approach of the variational method can be obtained by using a linear ansatz where the trial function is expanded in a fixed N-dimensional set of orthonormal basis functions $|\phi\rangle$

$$|\widetilde{\Psi}\rangle = \sum_i^N c_i\,|\phi_i\rangle \tag{1.36}$$

By using Lagrange's method of undetermined multipliers

$$\mathcal{L}(\mathbf{c}, E) = \langle\widetilde{\Psi}|\,\hat{H}\,|\widetilde{\Psi}\rangle - E(\langle\widetilde{\Psi}|\widetilde{\Psi}\rangle - 1) \tag{1.37}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{c}} = 0 \tag{1.38}$$

it is possible to show that the variational problem corresponds to solving the eigenvalue problem involving the Hamiltonian matrix **H**:

$$\mathbf{H}\mathbf{c}_n = E_n \mathbf{c_n} \tag{1.39}$$

Or, in a more general form

$$\mathbf{HC} = \mathbf{CE} \tag{1.40}$$

Here, **C** is a N by N coefficient matrix containing N column coefficient vectors $\mathbf{c}_n$ ($n = 0...N$) which describe N possible solutions for the trial wave function $|\widetilde{\Psi}\rangle$. **E** is a diagonal matrix containing the eigenvalues E_n. This approach of finding approximate solutions to the eigenvalue problem 1.31 is known as the *linear variational method*.

1.4 Hartree-Fock

The Hartree-Fock method is central to electronic structure method. It is computationally inexpensive, and is still routinely used in qualitative studies of large molecules, even if it does not accurately account for electron correlation. It also serves as the starting point for correlated methods. Only a few computational methods actually bypass the solution to the Hartree-Fock equations, firmly cementing its place in quantum chemistry.

1.4.1 The Hartree Fock Equations

Recall the structure of the electronic Hamiltonian

$$\hat{H} = \hat{T}_e + \hat{V}_{ne} + \hat{V}_{ee} + \hat{V}_{nn} \tag{1.41}$$

with

$$\hat{T}_e = -\sum_i^N \frac{1}{2}\nabla_i^2 \quad \text{kinetic energy of electrons} \tag{1.42}$$

$$\hat{V}_{ne} = -\sum_a^{N_{nuc}} \sum_i^N \frac{Z_a}{\mid \mathbf{R}_a - \mathbf{r}_i \mid} \quad \text{nuclei-electron repulsion} \tag{1.43}$$

$$\hat{V}_{ee} = \frac{1}{2}\sum_i^N \sum_{j\neq i}^N \frac{1}{\mid \mathbf{r}_i - \mathbf{r}_j \mid} \quad \text{electron-electron repulsion} \tag{1.44}$$

$$\hat{V}_{nn} = \frac{1}{2}\sum_a^{N_{nuc}} \sum_{b\neq a}^{N_{nuc}} \frac{Z_a Z_b}{\mid \mathbf{R}_a - \mathbf{R}_b \mid} \quad \text{nuclei-nuclei repulsion} \tag{1.45}$$

In Hartree-Fock theory, the electrons are treated as independent particles. One can therefore ignore the coupling between electrons in $\hat{V}_{ee}$ and express the Hamiltonian as a sum of an effective one-electron operator **f**, also known as the *Fock* operator, of the form

$$\hat{H} = \sum_i \hat{f}_i = \sum_i \mathbf{h}_i + \frac{1}{2}\sum_i \sum_{j\neq i} \mathbf{g}_i j \tag{1.46}$$

$$\hat{h}_i = -\frac{1}{2}\nabla_i^2 - \sum_a^{N_{nuc}} \frac{Z_a}{\mid \mathbf{R}_a - \mathbf{r}_i \mid} \tag{1.47}$$

$$\hat{g}_{ij} = \frac{1}{\mid \mathbf{r}_i - \mathbf{r}_j \mid} \tag{1.48}$$

The one-electron operator $\hat{h}$, also known as the *core* Hamiltonian describes the movement of the electrons in the field of the nuclei. The two-electron operator $\mathbf{g}_{ij}$ gives the average potential (or "field") experienced by electron i due to the presence of all the other electrons j. For this reason, Hartree-Fock is also known the *mean-field* approximation. In second quantization, the Fock operator takes the form

$$\hat{f} = \sum_{PQ}^{M} h_{PQ} a_P^\dagger a_Q + \frac{1}{2} \sum_{PQRS}^{M} g_{PQRS} a_P^\dagger a_R^\dagger a_S a_Q \tag{1.49}$$

with the matrix elements of the one and two electron operators given by

$$h_{PQ} = \langle \phi_P | \, \mathbf{h} \, | \phi_Q \rangle = \int \phi_P^*(\mathbf{x}) h(\mathbf{x}) \phi_Q(\mathbf{x}) d\mathbf{x} \tag{1.50}$$

$$g_{PQRS} = \langle PQ|RS\rangle = \int\int \phi_P^*(\mathbf{x}_1)\phi_R^*(\mathbf{x}_2) g(\mathbf{x}_1, \mathbf{x}_2) \phi_Q(\mathbf{x}_1)\phi_S(\mathbf{x}_2) d\mathbf{x}_1 d\mathbf{x}_2 \tag{1.51}$$

The elements g_{PQRS} are known as the 2-electron repulsion integrals. Calculating the expectation values for the fock operator in 1.49 using second quantization gives:

$$\begin{aligned} f_{PQ} &= \langle \phi_P | \, \mathbf{f} \, | \phi_Q \rangle \\ &= h_{PQ} + \sum_I^N \frac{1}{2}(g_{PQII} - g_{PIIQ}) \\ &= h_{PQ} + \frac{1}{2}(J_{PQ} - K_{PQ}) \end{aligned} \tag{1.52}$$

The symmetric matrix with entries f_{PQ} is also known as the *Fock matrix*. $\mathbf{J}$ is the *coulomb matrix* and describes electron correlation due to the coulomb potential (coulomb correlation), and $\mathbf{K}$ is the exchange matrix describing the electron correlation which arises due to the Pauli exclusion principle (Fermi correlation). The exchange contributions have no classical counterpart and arise purely from quantum mechanical considerations.

In the special basis where the Fock matrix is diagonal

$$f_{PQ} = \delta_{PQ} \epsilon_P \tag{1.53}$$

the one-electron eigenfunctions of the Fock operator

$$\mathbf{f} \, |\phi_P\rangle = \epsilon_P \, |\phi_P\rangle \tag{1.54}$$

are known as the *canonical molecular spin orbitals*, and the eigenvalues are the *molecular orbital energies*. Solving the *canonical Hartree-Fock equation* 1.54 gives the MOs which form the basis of the Hartree-Fock wave function. It should be stressed that the total

electronic Hartree-Fock energy is not the sum of the individual MO energies, but is given by the expectation value of the Hamiltonian

$$E_{HF} = \langle HF| \mathbf{H} |HF\rangle = \sum_{I}^{N} h_{II} + \frac{1}{2}\sum_{I}^{N} (J_{II} - K_{II}) \tag{1.55}$$

where the sum runs over the *occupied* orbitals I. For N electrons distributed over M MOs, there are N occupied orbitals with $\epsilon_I < 0$ and $M - N$ virtual orbitals with $\epsilon_A > 0$.

1.4.2 The Basis Set Approximation

Up until this point, the electronic wave function was constructed from Slater determinants of molecular spin orbitals. Virtually all applications use a basis set expansion to express the unknown MOs in terms of known functions, conventionally called *atomic orbitals*. Any type of function can be used, e.g. exponentials, Gaussians, polynomials or plane waves. Exponentials and Gaussians are best suited to describe bound systems. The molecular orbitals are then expressed as a *linear combination of atomic orbitals* (LCAO)

$$|\phi_i\rangle = \sum_{i}^{M_{basis}} c_{i\mu}\chi_\mu \tag{1.56}$$

The basis set approximation is *exact* for an infinite number of basis functions. For more details, see 1.10.

1.4.3 Working Equations for Restricted and Unrestricted Hartree-Fock

For reasons of efficient implementation, it is useful to separate out different electron spin components. The Fock matrix has four spin blocks: $F_{\alpha\alpha}$, $F_{\alpha\beta}$, $F_{\beta\alpha}$ and $F_{\beta\beta}$. The Fock matrix in the canonical basis is diagonal, and therefore only the diagonal blocks $F_{\alpha\alpha}$ and $F_{\beta\beta}$ are important. Introducing the notation $\overline{I}$ for MOs with spin σ', and I with opposite spin σ, the matrix elements of a spin block are given by

$$\begin{aligned} f^\sigma_{PQ} &= h^\sigma_{PQ} + \frac{1}{2}\left\{\sum_{PQ}^{N_\sigma}\sum_{I}^{N_\sigma} (PQ \mid II) - (PI \mid QI) - \sum_{PQ}^{N_\sigma}\sum_{I}^{N_{\sigma'}} \left(PQ \mid \overline{JJ}\right) - \left(P\overline{J} \mid Q\overline{J}\right)\right\} \\ &= h^\sigma_{PQ} + \sum_{PQ}^{N_\sigma} J^\sigma_{PQ} - K^\sigma_{PQ} + J^{\sigma'}_{PQ} - K^{\sigma'}_{PQ} \end{aligned} \tag{1.57}$$

The opposite spin block $f^{\sigma'}_{PQ}$ is obtained by substituting indices with a bar by indices without a bar and vice-versa. Spin separation yields two coupled sets of equations for alpha and beta MOs

$$\begin{aligned} \mathbf{f}^\alpha |\phi^\alpha_I\rangle &= \epsilon^\alpha_I |\phi^\alpha_I\rangle \\ \mathbf{f}^\beta \left|\phi^\beta_I\right\rangle &= \epsilon^\beta_I \left|\phi^\beta_I\right\rangle \end{aligned} \tag{1.58}$$

These are known as the unrestricted Hartree-Fock equations (UHF). For closed-shell molecules with equal number of alpha and beta electrons, the spatial part of the MOs is the same for both spins. The expression for the Fock matrix then further simplifies to

$$f_{ij} = h_{ij} + 2J_{ij} - K_{ij} \tag{1.59}$$

$$\mathbf{f}\,|\phi_i\rangle = \epsilon_i\,|\phi_i\rangle \tag{1.60}$$

The equations in 1.60 are known as the restricted Hartree-Fock (RHF) equations.
Using the linear variatonal method explained in the previous section for the MO trial functions expressed as a linear combination of N_{bas} AO basis functions, the eigenvalue problem for RHF can be recast in matrix form as

$$\mathbf{FC} = \mathbf{CE} \tag{1.61}$$

with the MO coefficient matrix $\mathbf{C}$ and the Fock matrix $\mathbf{F}$ in the AO basis given by

$$F_{\mu\nu} = H_{\mu\nu}^{core} + \sum_{\lambda\sigma}^{N_{basis}} \left(2\,(\mu\nu \mid \sigma\lambda)\,P_{\lambda\sigma} - (\mu\sigma \mid \nu\lambda)\,P_{\lambda\sigma}\right) \tag{1.62}$$

$$= H_{\mu\nu}^{core} + 2J_{\mu\nu} - K_{\mu\nu} \tag{1.63}$$

The symmetric matrix $\mathbf{P}$ is the so-called atomic orbital density matrix (DM) of the form

$$P_{\mu\nu} = \sum_{i}^{N_{occ}} C_{\mu i} C_{\nu i} \tag{1.64}$$

A similar expressions is found for UHF

$$\begin{aligned} F_{\mu\nu}^{\sigma} &= H_{\mu\nu}^{core} + \sum_{\lambda\sigma}^{N_{basis}} \left((\mu\nu \mid \sigma\lambda)\,P_{\lambda\sigma}^{T} - (\mu\sigma \mid \nu\lambda)\,P_{\lambda\sigma}^{\sigma}\right) \\ P_{\mu\nu}^{T} &= P_{\mu\nu}^{\sigma} + P_{\mu\nu}^{\sigma'} \end{aligned} \tag{1.65}$$

where the AO spin-density matrices $\mathbf{P}^{\sigma}$ are defined as the product of the corresponding coefficient matrices with spin σ.

1.4.4 The Self-Consistent Field Method

In general, the atomic orbitals are not orthogonal. The overlap matrix $\mathbf{S}$ is defined as

$$S_{\mu\nu} = \int \chi_{\mu}^{*}(\mathbf{r})\chi_{\nu}^{*}(\mathbf{r})d\mathbf{r} \tag{1.66}$$

with diagonal entries $S_{\mu\mu} = 1$, and off-diagonal elements $0 <\mid S_{\mu\nu} \mid < 1$. The eigenvalue problem for RHF then takes the more general form

$$\mathbf{FC} = \mathbf{SCE} \tag{1.67}$$

The equations in 1.67 are known as the *Roothan-Hall equations*. In the unrestricted case, they are called the *Pople-Nesbet equations* which are given by

$$\mathbf{F}^{\alpha}\mathbf{C}^{\alpha} = \mathbf{S}\mathbf{C}^{\alpha}\mathbf{E}^{\alpha} \tag{1.68}$$

$$\mathbf{F}^{\beta}\mathbf{C}^{\beta} = \mathbf{S}\mathbf{C}^{\beta}\mathbf{E}^{\beta} \tag{1.69}$$

The Fock matrix is constructed using the coefficient matrices $\mathbf{C}$ to compute the density matrix $\mathbf{P}$. This means that the RH (and PN) equations depend on their own solution and must be solved iteratively. Popular choices for iterative schemes include *Newton's* method and the *self-consistent field* (SCF) method, with the latter being the most straightforward one to implement.

The SCF procedure is summmarized in Algorithm 1. At every iteration, the Fock matrix is diagonalized to obtain a new guess for the density matrix, which is used for constructing the Fock matrix in the next step. These steps are repeated until *self-consistency* is reached. There are different ways to test for convergence, the simplest being the Hartree-Fock energy difference between subsequent iterations. A more rigorous bound is given by the matrix norm of the error vector $\mathbf{e}$

$$\mathbf{e} = \mathbf{FPS} - \mathbf{SPF} \tag{1.70}$$

At convergence, the density matrix commutes with the Fock matrix though the overlap matrix.

Algorithm 1: Hartree-Fock Self-Consistent Field

Input: Molecule with nuclear coordinates $\{\mathbf{R}_A\}$, atomic numbers $\{Z_A\}$, number of electrons N and basis set $\{\chi_\mu\}$

Output: The matrices $\mathbf{F}$, $\mathbf{P}$, $\mathbf{C}$ and $\mathbf{E}$

1 Calculate all one- and two electron integrals

2 Compute the transformation matrix $\mathbf{X}$ from the overlap matrix $\mathbf{S}$ with

$$\mathbf{X}^{\dagger}\mathbf{S}\mathbf{X} = \mathbf{1} \tag{1.71}$$

3 Generate a set of guess orbitals to compute an initial guess density $\mathbf{P}$

4 **while** *not converged* **do**

5 Construct the Fock matrix $\mathbf{F}$ using the current guess density

6 Orthogonalize the Fock matrix $\mathbf{F}' = \mathbf{X}^{\dagger}\mathbf{F}\mathbf{X}$

7 Diagonalize $\mathbf{F}'$ to obtain the new orthogonalized MO coefficient matrices $\mathbf{C}'$

8 Compute $\mathbf{C} = \mathbf{X}\mathbf{C}$

9 Form the new density $\mathbf{P} = \mathbf{C}\mathbf{C}^{T}$

10 Check convergence using certain criteria

SCF Initial Guesses

The pre-iteration steps compute the two-electron integrals, the transformation matrix $\mathbf{X}$ and a set of guess orbitals. There are different methods for generating a guess. The closer the guess is to the solution, the fewer SCF iterations are needed which saves time. The simplest method consists in using a null matrix for $\mathbf{P}$, which corresponds to setting the Fock matrix to the core Hamiltonian $\mathbf{H}^{core}$. Diagonalization then gives the guess orbitals. The core Hamiltonian gives a sufficiently close starting guess for small molecules, but is unsuitable for larger molecules.

The most popular and efficient method at the time of writing is the superposition of atomic densities (SAD) (ref). For each type of atom in the molecule, an atomic HF calculation is carried out which gives the atomic density matrix for this atom type. The molecular guess density matrix is then constructed by setting its diagonal blocks to to the atomic densities. The SAD method generates densities that are quite close to the solution. For implementation details, see Annex **??**.

An alternative starting guess can be obtained by first carrying out a HF calculation with a smaller basis set using the core or SAD guess, and then projecting the density matrix onto the larger basis set (Annex **??**). This method is especially useful for larger basis sets.

SCF convergence

There is no guarantee that the SCF procedure converges. For small molecules and equilibrium geometries, the unmodified SCF procedure converges quite smoothly. For large, diffuse basis sets or distorted geometries, additional modifications to the algorithms might be necessary:

1. Direct inversion of the iterative subspace (DIIS): the previous Fock matrices are used for extrapolation to generate a better Fock matrix (Annex 11.1)

2. Damping: A damping factor ω is introduced and the density matrix is replaced by a weighted average $\mathbf{P}'_{n+1} = \omega\mathbf{D}_n + (1-\omega)\mathbf{D}_{n+1}$. Damping is especially useful when the SCF energy oscillates around the equilibrium

3. Level shifting: It has been shown that raising the energy of the virtual orbitals guarantees convergence, at the cost of a decreased convergence rate

1.4.5 Brillouin's Theorem and Orbital Rotations

The Fock matrix can be divided into four distinct blocks: occupied-occupied, occupied-virtual, virtual-occupied and virtual-virtual. The terms "particle" (p) and "hole" (h) may be used instead of occupied and virtual. In the special case where the orbitals are the canonical MOs, the Fock matrix is diagonal. This is known as the *canonical condition* for the HF wavefunction. However, diagonality is not necessary for obtaining a valid HF wavefunction. The general Hartree-Fock equations take the form

$$\hat{f}\phi_P = \sum_{PR} \lambda_{PR}\phi_R \tag{1.72}$$

Where λ are the Lagrange multipliers. For non-canonical MOs, the p-p and h-h are non-diagonal, with elemets λ_{IJ} and λ_{AB} respectively. The off-diagonal elements are computed as

$$f_{IA} = \langle HF|\, \hat{f}\, |a_A^\dagger a_I HF\rangle = 0 \tag{1.73}$$

and are zero by virtue of *Brillouin's theorem*. This is also known as the *variational condition* for the HF wavefunction; it is an alternative formulation of the variational principle (see [9], section 10.3.5).

The Hartree-Fock energy is therefore stable under unitary rotations

$$\phi_{I'} = \mathbf{U}\phi_I \qquad U^\dagger U = \mathbf{U} \tag{1.74}$$

The canonical MOs may be rotated to give another orbital representation with smaller spatial extent, also known as localized molecular orbitals (see Section 3.4).

1.5 Electron Correlation

In the Hartree-Fock method, the electron-electron interaction is replaced by an average interaction. For large basis sets, H is actually able to recover approximately 99% of the electron correlation. Unfortunately, the remaining 1% are often important to compute chemical properties with sufficient accuracy.

Electron correlation arises from electrons trying to avoid each other due to coulombic repulsion (coulomb correlation) and the antisymmetry principle (Fermi correlation). This in turn leads to a region of space around each electron where the probability of finding another electron is reduced, typically known as the *coulomb hole* for electrons of opposite spin, and *Fermi hole* for electrons of the same spin.

Another distinction is often drawn between *dynamic correlation* and *static correlation*, although there exists no formal definition. Dynamic correlation generally describes the correlated movement of electrons due to their "instantaneous mutual repulsion" [1]. For example, in the ground-state Helium atom, electron correlation is purely dynamical. Static (or non-dynamic) correlation on the other hand arises in the case of near-degeneracy, where multiple configurations of similar energy contribute to the ground-state wave function. To show the difference between these two two types of correlation, the dissociation of the H_2 molecule is often taken as an illustrative example. At equilibrium distance, correlation is mostly dynamic, and the ground-state can be well described as a singlet state. In the dissociation limit, electrons may be coupled to yield either a singlet *or* a triplet, as the energy difference between those two states vanishes (Figure?). The correlation in the system in then entirely static. There is no clear-cut line between dynamic and static correlation, but it offers a useful classification for correlation effects.

To fully capture both dynamic and static correlation, it is crucial to add additional Slater determinants as mentioned previously. Methods which generate excited Slater determinants from a single reference SD recover mostly dynamical correlation: electrons are disturbed through their instantaneous repulsion and are excited to higher spin orbitals, hence the need of additional excited-type SDs. Methods generating SDs from multiple references recover mostly static correlation. At the theoretical limit (i.e. the full configuration space), both approaches eventually recover all correlation effects. It is therefore the responsibility of the computational chemist to choose the correct method that best fits the problem at hand.

Any electronic structure method that improves on the HF wave function is usually referred to as a "correlated method". Hartree-Fock, by convention, is an "uncorrelated method".

For the rest of this report, only single-reference methods are considered.

1.6 Configuration Interaction

1.6.1 The CI Matrix

Configuration interaction (CI) is the simplest and one of the oldest examples of a correlated electronic structure method. The CI wave function takes the general form

$$|CI\rangle = |\Psi_0\rangle + \sum_{IA} c_I^A |\Phi_{IA}\rangle + \sum_{\substack{I<J \\ A<B}} c_{IJ}^{AB} |\Phi_{IJ}^{AB}\rangle + \sum_{\substack{I<J<K \\ A<B<C}} c_{IJK}^{ABC} \Phi_{IJK}^{ABC} + \dots \tag{1.75}$$

summing all SDs of all possible types (singles, doubles, triples...). Φ_0 is the reference wave function, here taken as the HF ground state. Whereas the HF wave function was a linear combination of molecular orbitals, the CI wave function is a linear expansion of SDs. Similarly, the CI expansion coefficient matrix $\mathbf{C}$ are determined variationally

$$\frac{\partial}{\partial C_i} \frac{\langle CI| \hat{H} |CI\rangle}{\langle CI|C\rangle} \tag{1.76}$$

which, as was shown before, reduces to the eigenvalue problem

$$\mathbf{HC} = \mathbf{CE} \tag{1.77}$$

The matrix $\mathbf{H}$ is also known as the *CI matrix*. Using the notation $|S\rangle$, $|D\rangle$, $|T\rangle$, ... to denote the set of singles, doubles, triples ... SDs, the CI matrix takes the form

$$\begin{bmatrix}
\langle\Phi_0|\,\mathbf{H}\,|\Phi_0\rangle & 0 & \langle\Phi_0|\,\mathbf{H}\,|D\rangle & 0 & 0 & \dots \\
0 & \langle S|\,\mathbf{H}\,|S\rangle & \langle S|\,\mathbf{H}\,|D\rangle & \langle S|\,\mathbf{H}\,|T\rangle & 0 & \dots \\
\langle D|\,\mathbf{H}\,|\Phi_0\rangle & \langle D|\,\mathbf{H}\,|S\rangle & \langle D|\,\mathbf{H}\,|D\rangle & \langle D|\,\mathbf{H}\,|T\rangle & \langle D|\,\mathbf{H}\,|Q\rangle & \dots \\
0 & \langle T|\,\mathbf{H}\,|S\rangle & \langle T|\,\mathbf{H}\,|D\rangle & \langle T|\,\mathbf{H}\,|T\rangle & \langle T|\,\mathbf{H}\,|Q\rangle & \dots \\
0 & 0 & \langle D|\,\mathbf{H}\,|Q\rangle & \langle T|\,\mathbf{H}\,|Q\rangle & \langle Q|\,\mathbf{H}\,|Q\rangle & \dots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{bmatrix} \tag{1.78}$$

In analogy to the Fock matrix, some blocks in the CI matrix are zero. Brillouin's theorem states that there is no coupling between the ground state and the singlet states. That does not imply that they do not contribute to the CI energy at all. Singles mix *indirectly* via doubles. Moreover, matrix blocks of the Hamiltonian between two SDs which differ by more than two spin orbital orbitals are also zero. Triples mix with doubles and singles, but not with the ground state.

A more compact representation of the CI matrix is obtained by taking linear combinations of SDs in the same excitation manifold, known as configuration state functions (CSF) or spin-adapted configurations (SAC). The CSFs form a basis smaller than that composed of all individual SDs which leads to computational savings. However, CSFs were primarly introduced to preserve the spin symmetry of the ground state, or in other words, CSFs are eigenfunctions of the $\mathbf{S}^2$ operator. If the HF ground state is a singlet, a non-spin-symmetric CI basis may lead to the CI wave function being a mixture of singlet and triplet determinants.

1.6.2 Truncated CI

Full CI, i.e. including all excitation manifolds, is only computationally feasible for all but the smallest molecules, due to the binomial increase in the number of SDs as a function of system and basis set size. For this reason, the CI wave function my be truncated at a given excitation level. Including only singles gives configuration interaction with singles (CIS), including singles and doubles yields configuration interaction with singles and doubles (CISD), etc. Higher order methods recover a larger fraction of the correlation energy, but come at a higher computational cost.

It should be noted that the energy of the CIS wave function is equal to the HF energy due to Brillouin's theorem, and hence does not contribute to the correlation energy of the ground state.

1.6.3 Solving the CI Eigenvalue Problem

Even at relatively low truncation levels, the number of matrix elements have a quite steep polynomial scaling with $\mathcal{O}(N^6)$ for CISD and $\mathcal{O}(N^8)$ for CISDT. In most cases however, only the few lowest eigenvalues are needed. Davidson's method of matrix diagonalization (Annex **??**) was specifically developed to tackle this problem. Rather than storing the whole matrix, only matrix-vector products need to be computed

$$\mathbf{r} = \mathbf{M}_{CI}\mathbf{u} \tag{1.79}$$

Closed expressions can be derived and the full matrix is not explicitly needed, but generated on-the-fly.

1.6.4 Size Consistency and Size Extensivity

Over the years, single-reference truncated CI methods have fallen out of favor for more sophisticated methods, due to them not being size-consistent and size-extensive. *Size-consistency* refers to the idea that the energy of two non-interacting systems A and B should be equal to the sum of their individual energies obtained from two different calculations:

$$E(A+B) = E(A) + E(B) \tag{1.80}$$

Size-extensivity is a closely-related criterion that states that the energy should be a linear function of the number of electrons , i.e. the energies of small and large molecules have similar errors, which is important for comparing properties [11]. As the system size increases, truncated CI recovers less and less of the total correlation energy.

1.7 Coupled Cluster

The coupled-cluster (CC) approximation offers a more sophisticated picture of electron correlation than CI, and has become one of the most successful and accurate *ab initio* correlated methods. It is both size-consistent and size-extensive.

1.7.1 Pair Clusters

Consider a system composed of two electrons, occupying the orbitals I and J in the independent particle model. Correlation manifests itself by the electrons' instantaneous repulsion and excitation into higher lying orbitals. Mathematically, this may be expressed as

$$a_I^\dagger a_J^\dagger + \sum_{A>B} t_{IJ}^{AB} a_A^\dagger a_B^\dagger = \left(1 + t_{IJ}^{AB} \hat{\tau}_{IJ}^{AB}\right) a_I^\dagger a_J^\dagger \tag{1.81}$$

where $\mathbf{t}$ are the associated cluster coefficients, also known as *amplitudes*. By virtue of Brillouin's theorem, single excitations are not considered. Equation 1.81 is known as the *electron pair*, *two-electron cluster* or *pair-cluster* approximation.

In a first approximation, electron pairs may be treated independently in a molecular system, in what is known as the independent electron pair approximation (IEPA). The total correlation energy is then simply given as the sum of the individual *pair correlation energies*

$$E_{corr}^{IEPA} = \sum_{I<J} e_{IJ} \tag{1.82}$$

$$|IEPA\rangle = \sum_{I<J,A<B} \left(1 + t_{IJ}^{AB} \tau_{IJ}^{AB}\right) |HF\rangle \tag{1.83}$$

A more complete picture of electron correlation is given by additionally letting electron clusters interact with each other by using the parametrization

$$|CCD\rangle = \left(\prod_{A>B,I>J} 1 + t_{IJ}^{AB} \hat{\tau}_{IJ}^{AB} \right) |HF\rangle \tag{1.84}$$

The resulting wave function corresponds to the coupled cluster approximation including only doubles (CCD). As opposed to CID, CCD additionally includes *products* of doubles cluster operators ($\tau_{IJ}^{AB}\tau_{KL}^{CD}$, or $\tau_{IJ}^{AB}\tau_{KL}^{CD}\tau_{MN}^{EF}$), in other words, doubles excitations are included up to infinite order. It is this property that makes CC size-extensize.

1.7.2 Coupled Cluster Ansatz

The CCD model can be generalized to let clusters of three and more electrons interact with each other, and electrons interact within these clusters. The general CC ansatz reads

$$|CC\rangle = \left(\prod_{\mu} 1 + t_\mu \hat{\tau}_\mu \right) |HF\rangle = exp(t_\mu \hat{\tau}_\mu) |HF\rangle = exp(\hat{T}_\mu) |HF\rangle \tag{1.85}$$

where $\hat{T}_\mu$ is the *cluster operator*, and μ are the excitation manifolds. The cluster operator may be partitioned into classes comprising all singles, doubles, ... excitations:

$$\hat{T} = \hat{T}_1 + \hat{T}_2 + ...\hat{T}_N \tag{1.86}$$

Truncating the cluster operator to include only excitations up to a certain degree yields a hierarchy of CC method named CCS, CCSD, CCSDT etc. Again, including higher orders

implies a higher computational effort. The exponential in Equation 1.85 for different truncation levels is approximated as

$$exp(\hat{T}) = \hat{T}_1 \tag{1.87}$$

$$exp(\hat{T}_1 + \hat{T}_2) = \hat{T}_2 + \frac{1}{2}\hat{T}_1^2 \tag{1.88}$$

$$exp(\hat{T}_1 + \hat{T}_2 + \hat{T}_3) = \hat{T}_3 + \hat{T}_1\hat{T}_2 + \frac{1}{6}\hat{T}_1^3 \tag{1.89}$$

$$\dots$$

Triplet configurations for example are generated by three mechanisms. $\hat{T}_3$ is known as a *connected* term, and the other terms which are products of lower order operators are known as *disconnected* terms.

1.7.3 The Coupled Cluster Equations

The cluster amplitudes are unknown and need to be solved for. Equations for the amplitudes can be obtained by projecting the excitation manifolds onto the CC ground state wave function 1.85. Using the so-called *similarity-transformed* Hamiltonian

$$\hat{\bar{H}} = exp(-\hat{T})\hat{H}exp(\hat{T}) \tag{1.90}$$

gives the set of non-linear equations

$$\langle \mu_1 | \hat{\bar{H}} | HF \rangle = 0 \tag{1.91}$$

$$\langle \mu_2 | \hat{\bar{H}} | HF \rangle = 0 \tag{1.92}$$

$$\langle \mu_3 | \hat{\bar{H}} | HF \rangle = 0 \tag{1.93}$$

$$\vdots \tag{1.94}$$

where μ_n is nth order excitation manifold (singles, doubles ....). The exact expressions of the CC amplitude equations at different truncation levels may be evaluated using the Baker–Campbell–Hausdorff (BCH) formula, but will not be discussed in detail here. As an example of the exact form of the working equations, consider the CCSD model truncated at doubles excitation. Equation 1.94 then reduces to

$$\langle \mu_1 | \hat{\bar{H}} + \left[\hat{\bar{H}}, \hat{T}_2\right] | HF \rangle = 0 \tag{1.95}$$

$$\langle \mu_2 | \hat{\bar{H}} + \left[\hat{\bar{H}}, \hat{T}_2\right] + \frac{1}{2}\left[\left[\hat{\bar{H}}, \hat{T}_2\right], \hat{T}_2\right] | HF \rangle = 0 \tag{1.96}$$

The system of equations 1.94 depends on its own solution, and therefore needs to be solved iteratively. They are most commonly solved using a modified Newton method with DIIS acceleration.

1.8 Perturbation Theory

Coupled cluster and configuration interaction offer a systematic way to move towards the exact solution to the Schrödinger equation by means of adding more Slater determinants.

However, the calculation of the CC and CI wave functions is very expensive, and it may be profitable to look at alternative schemes. *Perturbation theory* (PT) is a different approach to systematically close in on the exact wave function. It is based on the idea that the exact solution differs only slightly from a previously solved problem for a simpler, related system.

1.8.1 Rayleigh-Schrödinger Perturbation Theory

Perturbation theory is used in a wide range of fields and disciplines in natural sciences and mathematics. In the context of molecular electronic structure theory, the most widely used form of PT is Rayleigh Schrödinger perturbation theory (RSPT). In RSPT, the Hamiltonian is partitioned according to

$$\hat{H} = \hat{H}_0 + \hat{U} \tag{1.97}$$

where $\hat{H}_0$ is some reference zero-order Hamiltonian with known eigenfunctions $|\Psi_i^0\rangle$ and eigenvalues E_i^0. $\hat{U}$ is a small perturbation to the system. The exact wave function and energies may be expanded in orders of the perturbation

$$|\Phi_i\rangle = \sum_{k=0}^{\infty} \left|\Psi_i^{(k)}\right\rangle \tag{1.98}$$

$$E_i = \sum_{k=0}^{\infty} E_i^{(k)} \tag{1.99}$$

The task at hand is now to derive closed expression for higher order terms of order n using terms of order $n-1$ and lower. Substituting the expressions 1.98 and 1.99 into the Schrödinger equation gives

$$\left(\hat{H}_0 + \hat{U}\right) \sum_{k=0}^{\infty} \left|\Psi_i^{(k)}\right\rangle = \left(\sum_{k=0}^{\infty} E_i^{(k)}\right) \left(\sum_{k=0}^{\infty} \left|\Psi_i^{(k)}\right\rangle\right) \tag{1.100}$$

Collecting terms of order n, the above expression may be rewritten as a system of equations involving the residual of the Hamiltonian:

$$(\hat{H}_0 - E_i^{(0)})\,|\Psi_i^{(n)}\rangle = -\hat{U}\,|\Psi_i^{(n-1)}\rangle + \sum_{k=1}^{n} E_i^{(k)}\,|\Psi_i^{(n-k)}\rangle \tag{1.101}$$

Or alternatively, when multiplied with the inverse of the Hamiltonian residual:

$$|\Psi_i^{(n)}\rangle = -(\hat{H}_0 - E_i^{(0)})^{-1} \left(\hat{U}\,|\Psi_i^{(n-1)}\rangle + \sum_{k=1}^{n} E_i^{(k)}\,|\Psi_i^{(n-k)}\rangle\right) \tag{1.102}$$

Furthermore, to obtain simpler expressions for $E_i^{(k)}$, the normalization is chosen such that $\langle\Psi_i^{(0)}|\Phi_i\rangle = 1$, also known as intermediate normalization. From this it follows that the approximate wave functions are then orthogonal to the reference states

$$\langle\Psi_i^{(0)}|\Psi_i^{(n)}\rangle = 0 \qquad n = 1, 2, 3, \ldots \tag{1.103}$$

Left-projection of $\langle\Psi_i^{(0)}|$ onto the system of equations in 1.101 and using the orthogonality condition 1.103 yields the master equations for the RSPT energies

$$E_i^{(n)} = \langle\Psi_i^{(0)}|\,\hat{U}\,|\Psi_i^{(n-1)}\rangle \qquad n > 0 \tag{1.104}$$

The approximate energy expressions can be solved for without the need of iterative procedures, and closed expressions may be derived for a given reference. One way of solving 1.104 is to expand the first-order wave function in terms of the the eigenfunctions of $\hat{H}_0$:

$$|\Psi_i^{(1)}\rangle = \sum_n c_n^{(1)}\,|\Psi_i^{(0)}\rangle \tag{1.105}$$

Multiplying from the left by $\langle\Psi_n^{(0)}|$, the expansion coefficients can be obtained with

$$\langle\Psi_n^{(0)}|\Psi_i^{(1)}\rangle = c_n^{(1)} \tag{1.106}$$

From the expression of the first order wave function in 1.98

$$|\Psi_i^{(1)}\rangle = -(\hat{H}_0 - E_i^{(0)})^{-1}(\hat{U} + E_i^{(1)})\,|\Psi_i^{(1)}\rangle \tag{1.107}$$

it follows that the first order expansion coefficients are given by

$$c_n^{(1)} = \frac{\langle\Psi_n^{(0)}|\,\hat{U}\,|\Psi_0^{(0)}\rangle}{E_i^{(0)} - E_n^{(0)}} \tag{1.108}$$

Higher order energy expressions can then be "build up" step by step from lower order approximations following the normalization and orthogonality conditions. The first few closed-form RSPT energy expressions are given by

$$E_0^{(1)} = \langle\Psi_0^{(0)}|\,\hat{U}\,|\Psi_0^{(0)}\rangle \tag{1.109}$$

$$E_0^{(2)} = \sum_n \frac{|\,\langle\Psi_0^{(0)}|\,\hat{U}\,|\Psi_n^{(0)}\rangle\,|^2}{E_0^{(0)} - E_n^{(0)}} \tag{1.110}$$

$$\begin{aligned} E_0^{(3)} = &\sum_{nm} \frac{\langle\Psi_0^{(0)}|\,\hat{U}\,|\Psi_n^{(0)}\rangle\rangle\,\langle\Psi_n^{(0)}|\,\hat{U}\,|\Psi_m^{(0)}\rangle\,\langle\Psi_m^{(0)}|\,\hat{U}\,|\Psi_0^{(0)}\rangle}{(E_0^{(0)} - E_n^{(0)})(E_0^{(0)} - E_m^{(0)})} \\ &- E_0^{(1)} \sum_n \frac{|\,\langle\Psi_0^{(0)}|\,\hat{U}\,|\Psi_n^{(0)}\rangle\,|^2}{(E_0^{(0)} - E_n^{(0)})^2} \end{aligned} \tag{1.111}$$

...

1.8.2 Møller-Plesset Perturbation Theory

The success of RSPT is closely related to the choice of the zero-order Hamiltonian. The most popular variant of RSPT is Møller-Plesset perturbation theory (MPPT), where $\hat{H}_0$ is taken as the Fock operator from HF theory

$$\hat{H}_0 = \hat{f} = \sum_P \epsilon_P a_P^\dagger a_P \tag{1.112}$$

The zero-order wave function $|\Psi_0^{(0)}\rangle$ corresponds to the Hartree Fock wave function $|HF\rangle$. The perturbation operator takes the form

$$\hat{U} = \hat{H} - \hat{f} = \sum_{PQRS} \hat{g}_{PQRS} a_P^\dagger a_R^\dagger a_S a_Q - \hat{V}^{HF} \tag{1.113}$$

where V^{HF} is the Hartree-Fock potential. The zero-order component of the ground state energy is simply given as the sum of the orbital energies

$$E_0^{(0)} = \sum_I \epsilon_I \tag{1.114}$$

The first order energy is

$$\begin{aligned} E_0^{(1)} &= \langle \Psi_0^{(0)} | \hat{U} | \Psi_0^{(0)} \rangle \\ &= \langle HF | \hat{U} | HF \rangle \\ &= -\frac{1}{2} \sum_{IJ} (\langle IJ|IJ\rangle - \langle IJ|JI\rangle) \\ &= -\frac{1}{2} \sum_{IJ} \langle IJ| \, |IJ\rangle \end{aligned} \tag{1.115}$$

where $\langle IJ| \, |IJ\rangle$ are the antisymmetrized two-electron integrals in the MO basis. The energy sum $E_0^{(0)} + E_0^{(1)}$ corresponds to the Hartree-Fock energy. Therefore, the first correction to the Hartree-Fock energy occurs at the second order of MPPT. Using the notation MPn to refer to MPPT including perturbations up to the nth order, the second order energy reads

$$E_0^{(2)} = E_{MP2} = \frac{1}{4} \sum_{IJAB} \frac{| \langle IJ| \, |AB\rangle |^2}{\epsilon_I + \epsilon_J - \epsilon_A - \epsilon_B} \tag{1.116}$$

For a closed-shell molecule, the restricted MP2 energy can be obtained by spin-separation similarly to how it was done in Section 1.4.3 for Hartree-Fock:

$$\begin{aligned} E_{RMP2} &= \sum_{ijab} \frac{(ia \mid jb) \left[2 (ia \mid jb) - (ib \mid ja) \right]}{\epsilon_i + \epsilon_j - \epsilon_a - \epsilon_b} \\ &= \sum_{ijab} t_{iajb} \left[(ia \mid jb) - (ib \mid ja) \right] \end{aligned} \tag{1.117}$$

where $\mathbf{t}$ are the MP2 amplitudes. Furthermore, the MP2 energy may be split into individual electron pair contributions, analogous to CC:

$$E_{MP2} = \sum_{ij} e_{ij} \tag{1.118}$$

The energy expressions for MP3 and beyond will not be discussed here.

1.8.3 Convergence Behavior of the MPn series

MP2 is a computationally cheap correlated method that includes 80% to 90% of electron correlation method, scaling with $\mathcal{O}(N^5)$ as a function of the system size N. Higher order variants like MP3 or MP4 are considerably less popular.

Ideally, the MPn energy should converge monotonically towards the limit with increasing order of perturbation. However, such a behavior is not guaranteed. Contrary to the CI energy, which is determined variationally and therefore has a lower bound, the same is not true for MPPT and the MPn series may become divergent or oscillating for larger basis sets, especially if diffuse functions are used. MP2 improves on the HF wave function, but slightly overestimates correlation energy. MP3 underestimates electron correlation, and properties computed at this level are often inferior to those computed at second order. MP4 again overestimates correlation effects, but is better than MP2.

Due to the erratic convergence behavior of MPPT, higher order variations like MP3 or MP4 have fallen somewhat out of favor in the recent years. Moreover, the requirement of the single-determinant HF wave function being a suitable starting guess makes MPPT ill-suited to describe static correlation effects.

1.8.4 Spin-Component-Scaled Møller Plesset Perturbation Theory

With the rise of density functional theory (DFT), even the computationally inexpensive MP2 method fell out of use in favor of DFT which often shows better performance and accuracy for the same molecular systems.

In the early 2000s, MPPT again gained more popularity with the introduction of *spin-component scaling* (SCS) by Grimme et al. [12, 13] which greatly improves on the accuracy of MP2.

Consider again the MP2 energy in the unrestricted case in Equation 1.117. The energy contributions can be split into same-spin (SS) and opposite spin (OS) components:

$$E_{MP2-SS} = \sum_{IJ} e_{IJ} + e_{\overline{IJ}} \tag{1.119}$$

$$E_{MP2-OS} = \sum_{IJ} e_{I\overline{J}} \tag{1.120}$$

with

$$e_{IJ} = \sum_{AB} t_{IAJB}((IA \mid JB) - (IB \mid JA)) \tag{1.121}$$

$$e_{\overline{IJ}} = \sum_{AB} t_{\overline{IAJB}}(\left(\overline{IA} \mid \overline{JB}\right) - \left(\overline{IB} \mid \overline{JA}\right)) \tag{1.122}$$

$$e_{I\overline{J}} = \sum_{AB} t_{IA\overline{JB}} \left(IA \mid \overline{JB}\right) \tag{1.123}$$

The correlation effects in SS and OS are of different nature as discussed in Section 1.5. Hartree-Fock accounts for Fermi correlation by the antisymmetry principle but does not fully account for Coulomb correlation. MP2 cannot fully rectify this deficiency in the

starting guess. SCS-MP2 accounts for this behavior by scaling down the SS components and scaling up the OS components

$$E_{SCS-MP2} = c_{os}E_{OS-MP2} + c_{ss}E_{SS-MP2} \tag{1.124}$$

where the scaling factors are determined empirically by fitting to a data set, with $c_{ss} = 6/5$ and $c_{os} = 1/3$. In later iterations of SCS-MP2, the two parameters were unified by introducing the relationship

$$c_{ss} = 4 - 3c_{os} \tag{1.125}$$

whith c_{ss} and c_{os} set to 0.4 and 1.3 respectively.

SCS-MP2 gives considerable improvements to reaction energies [12], barrier heights [14, 15], geometries and vibrational frequencies [16], comparable to QCISD(T) with errors on the order of 1.7 kcal/mol. Strictly speaking, SCS-MP2 is no longer an *ab initio* method, but *semi-empirical*.

SCS was initially an *ad-hoc* improvement to the description of the wave function, but it is possible to justify its position in the theoretical framework of MPPT [17, 18].

Over the years, many different variations of SCS-MP2 have been proposed [19, 20, 21, 22]. One variation is the so-called spin-opposite scaled (SOS) MP2 method [19], where the SS components are simply ignored:

$$E_{SOS-MP2} = c_{os}E_{OS-MP2} \tag{1.126}$$

with c_{os} set to 1.3 instead of 1.2 as in SCS-MP2. The method can be justified by observing that the SS components already do not contribute a lot to the SCS-MP2 energy. SOS-MP2 has comparable or slightly worse accuracy than SCS-MP2. The major advantage which makes SOS-MP2 one of the more attractive spin-component scaling variants is the reduced scaling $\mathcal{O}(N^4)$ compared to $\mathcal{O}(N^5)$ for (SCS)-MP2 when the density-fitting approximation is used.

1.8.5 Hybrid Coupled Cluster Methods

Perturbational approaches may also be used to obtain approximate hybrid coupled cluster methods.

Consider again the CCSD equations for the coupled cluster singles 1.95 and doubles amplitudes 1.96. Introducing the same partitioning of the Hamiltonian as in MP2 with

$$\hat{H} = \hat{F} + \hat{U} \tag{1.127}$$

where $\hat{U}$ is also known as the *fluctuation potential*, the coupled cluster singles doubles method by Christiansen et al. [23] approximates the doubles amplitudes to first order only. The doubles equation 1.96 thus becomes

$$\langle \mu_2 | \left[\hat{F}, \hat{T}_2 \right] + \hat{\overline{H}} | HF \rangle = 0 \tag{1.128}$$

Eqautions 1.95 and 1.128 define the so-called *CC2* model. The doubles equations give an MP2-like closed expressions, and only the singles amplitudes need to be determined variationally.

Scaling	Non-correlated	CI methods	CC methods	MP methods
$\mathcal{O}(N^4)$	HF			
$\mathcal{O}(N^5)$		CIS	CC2, SCS-CC2	MP2, SCS-MP2
$\mathcal{O}(N^6)$		CISD	CCSD	MP3
$\mathcal{O}(N^7)$			CC3, CCSD(T)	MP4
$\mathcal{O}(N^8)$		CISDT	MP5	CCSDT

Table 1.1: Formal scaling of popular electronic structure methods

The CC2 energy has a similar accuracy and computational effort to MP2. Spin-component scaling has been generalized to CC2 as well [24, 25], with similar improvements to its accuracy [26].

Higher order CC methods like CCSDT can be approximated in a similar way to obtain the approximate CC3 method [27], where triples contributions are approximated to second order. A related method is the coupled cluster singles doubles with perturbative triples method [28], abbreviated as CCSD(T), where triples contributions are treated perturbatively and *added* to the CCSD model. CCSD(T) and CC3 have comparable accuracy and cost.

1.9 Performance of Correlated Methods

The previous sections give insight into the most popular, single-determinant, correlated electronic structure methods. The methods vary widely in accuracy, computational cost and convergence behavior.

Table **??** gives the formal computational scaling for some of the methods. The notation $\mathcal{O}()$, also known as *Big O notation*, is the standard way of indicating the limiting behavior of algorithms for increasing input size. N is used as a measure of the molecular system size (e.g. number of atoms or basis functions). *Formal* scaling means that factors like sparsity or locality are not considered.

At the time of writing, on current work stations, $\mathcal{O}(N^5)$ methods are limited to system sizes of around 50 to 100 atoms, and $\mathcal{O}(N^6)$ to sizes of several tens of atoms. Models with $\mathcal{O}(N^7)$ scaling and beyond are not used routinely.

In terms accuracy, the current trend is often observed [1]:

$$HF << MP2 \approx CC2 < CCSD < MP4 < CCSD(T) \tag{1.129}$$

1.10 Basis Sets

Chapter 2

Electronic Excited States

> "The XXIst century might be very well the century of light. Understanding and controlling photoexcited systems will be crucial for future research in many branches of optics and photonics."
>
> — *L. González, D. Escudero, L. Serrano-Andrés (2011) [29]*

A quantum system is said to be in an *excited state* if that state is at a higher energy level than the ground state, for example by absorption of one or more light quanta. While computing ground state properties has become routine even for larger molecules, the extension of the standard models to excited state properties is an active field of research. Triggered by the development of complex, high-resolution spectroscopic techniques such as X-ray spectroscopy [30], and advances in photochemistry [29], the demand for accurate and computational methods of excited states has been steadily increasing. Electronic spectra are often very difficult to interpret, and *computational spectroscopy* has emerged as an important tool to explain the underlying mechanisms.

Excited states are notoriously difficult to model, and similarly to their ground state analogs, there is no single method to rule them all. Over the years, many different approaches have been proposed, each with their strengths and weaknesses. This section will go over the most popular, single-reference methods available, with a focus on the algebraic diagrammatic construction method.

2.1 Nature of Excited States

The potential energy landscape of electronic excited states is complex and governed by various absorption and decay mechanisms. Excitations are generally grouped into three categories:

1. *Valence excitations*, where valence electrons are excited into (local) higher lying unoccupied orbitals above the Fermi level
2. *Rydberg excitations*, where electrons are excited into very diffuse orbitals around the molecule
3. *Charge transfer excitations* (CT), where electrons are excited to different parts of the molecule or different molecules entirely.

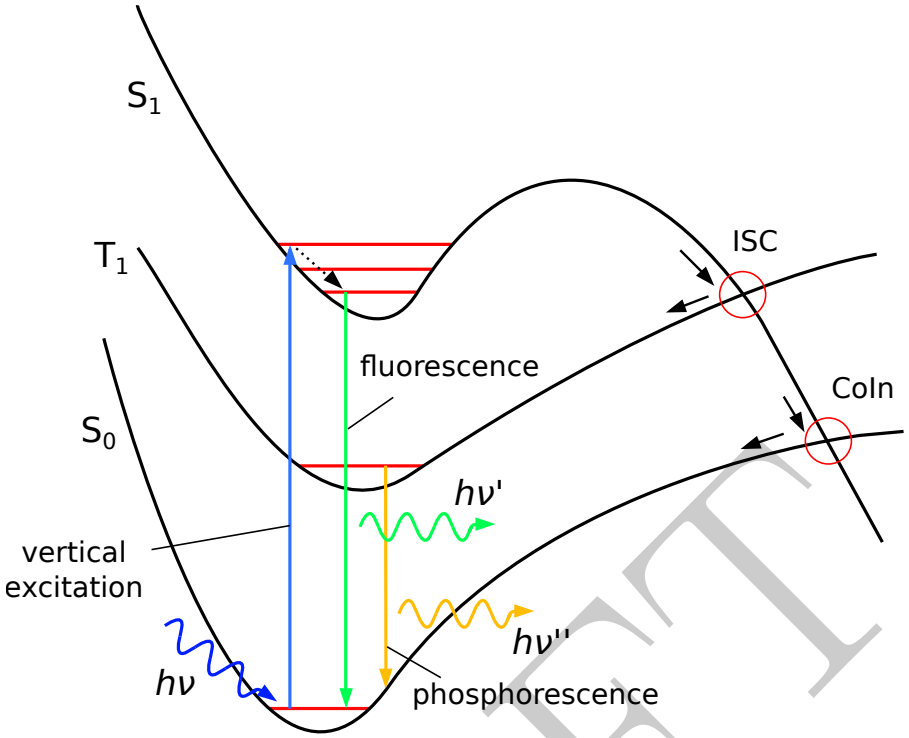


Figure 2.1: Potential energy surface of a chemical system depicting the major pathways encountered in spectroscopy and photochemistry.

Excited states typically have lifetimes and decay back to the ground state via several different mechanisms. Figure 2.1 illustrates the different processes. The notation S, D, T ... is used to denote singlet, doublet, triplet ... states and the subscripts indicate the energy level, where 0 is the ground state, 1 is the first excited state for the given spin symmetry, 2 is the second excited state etc. The transition from the ground state S_0 to the excited state S_1 on the same reaction coordinate is known as a *vertical excitation*. The excited state may be in a higher vibrational state at that reaction coordinate (indicated by the lines within the potential wells), and relax to the lowest level. The difference between these two points is known as the *reorganization energy*, and the difference between the lowest vibrational states of S_0 and the excited state is known as the *adiabatic excitation* energy. The molecule returns to the ground state by emitting a photon in a process known as *fluorescence*.

Surfaces of different states may cross at specific reaction coordinates. The crossing between states with different multiplicity (e.g. S_1 to T_1) is known as an inter-system crossing (ISC). The process between two states where the crossing takes place between molecules of the same spin-symmetry is known as *internal conversion* (IC), and takes place at a *conical intersection* (CoIn). The S_1 excited state can cross over to the T_1 state via an ISC which then decays in a process known as *phosphorescence*, or it can decay radiation-less via the CoIn. At the ISC and CoIn, the Born-Openheimer approximation breaks down due to non-adiabating coupling between electrons and nuclei. ISCs and CoIns are central to describing the dynamics in photochemical events [31, 32, 33, 34].

For the sake of brevity, this chapter will focus on the computational of vertical excitation energies only.

2.2 Explicit Optimization of the Excited State Wave Function

One of the conceptually simplest approaches for obtaining information on excited states is explicit optimization of the excited wave function. The excitation energy is then directly computed by taking the difference between the ground state energy and the energy of excited state i

$$E_{0\to i} = E_i - E_0 \tag{2.1}$$

Any ground state model discussed in the previous chapter can be used. This approach to excited states is known by different names, depending on which approximation is used. Generally, the Greek letter Δ is just prepended to the method name, giving ΔSCF or ΔHF for Hartree-Fock [35, 36, 37], ΔKS (Kohn-Sham) or ΔDFT for DFT [38, 39, 40], and so on. At the moment of writing, excitation energies are also routinely computed using ΔMPn [41], ΔCI, ΔMCSCF [42] and ΔCC [41, 43]. From here on out, ΔX will be used as an umbrella term to group all aforementioned terms.

Despite the simplicity of the ΔX methods, obtaining a solution to the KS or HF equations for higher energy states is non-trivial. By the variational principle, the SCF method finds the lowest energy solution. A HF type excited wave function may therefore collapse to that lowest energy solution during the SCF procedure (variational collapse). For small symmetric molecules, it is possible to converge excited states if they have a different spin multiplicity or spatial point group to the ground state. If the ground and excited state have the same symmetry however, this approach will not work. This technical difficulty was one of reasons why ΔX methods never gained much ground compared to more sophisticated methods.

In 2008, Gilbert et al. [44] proposed a modification the SCF procedure that prevents variational collapse, known as the *maximum overlap method* (MOM). On each iteration, the new guess orbitals are obtained by diagonalization of the Fock matrix which is constructed using the old coefficients

$$\mathbf{F}(\mathbf{C}^{old})\mathbf{C}^{new} = \mathbf{S}\mathbf{C}^{new}\epsilon \tag{2.2}$$

At this step, it is possible to decide which of those new orbitals are actually occupied. Normally, the n_{occ} eigenvectors with the lowest eigenvalues are chosen as the new occupied MOs. Alternatively, the MOM protocol chooses the set of new MOs that overlap most with the span of the old coefficients, by evaluating the overlap matrix

$$\mathbf{O} = (\mathbf{C}^{old})^{\dagger}\mathbf{C} \tag{2.3}$$

The maximum overlap method has led to a renewed interest in the ΔX methods in recent years, especially in the context of core excitations and ionizations.

Adding to the above-mentioned technical difficulties, there are several other known criticisms. First, each excited state requires a separately optimized wave function, which may become a limiting factor. Second, the ΔX methods assume that a transition can be represented by an excitation involving two orbitals. The separate optimization generally leads to the excited states being non-orthogonal, and there is considerable overlap between high and low energy states [45, 46, 44]. ΔX is therefore assumed to be only applicable to

low-lying excited states. Third, the transition moments cannot be computed directly, but need to be evaluated using Fermi's Golden Rule [47]. Futhermore, to allow a comparison with experimenatl XAS spectra, the calculated transition energies must be convoluted, for example by Gaussian functions, to account for the finite experimental resolution and lifetime of the electron hole. Finally, using an unrestricted HF or KS formalism leads to spin contamination. A single excited state is not a pure singlet but a mixture of singlet and triplet state. Spin contamination can be alleviated by applying Ziegler's spin purification formula [48]:

$$E_S = 2E_{mixed} - E_T \tag{2.4}$$

Despite these disadvantages, ΔX is still an attractive and low-cost alternative to response and propagator methods.

2.3 The Algebraic Diagrammatic Construction Method

The algebraic diagrammatic construction (ADC) scheme is an excited state method originating from Green's functions [49, 2]. By diagrammatic perturbation expansion of electron propagators, ADC gives a hierarchy of methods which systematically converge to the exact solution (Full CI).

2.3.1 Many-Body Green's Function

Many-Body Green's Functions (MBGFs) are powerful tools to treat electron correlation in quantum mechanics. They are more commonly encountered in (condensed matter) physics, for the modeling of strongly correlated systems such as metals or semi-conductors. MBGFs get their name from their building blocks: Green's functions (GFs). GFs, or *correlation functions*, are special solutions to differential equations (DEQs).

Consider the inhomogeneous DEQ in one dimension:

$$\hat{D}_x y(x) = f(x) \tag{2.5}$$

where $\hat{D}$ is a linear differential operator. The general solution can be divided into a *homogeneous* and a *special* part

$$y(x) = y_{hom}(x) + y_{spec}(x) \tag{2.6}$$

where y_{hom} is the solution to the homogeneous equation $\hat{D} y_{hom}(x) = 0$. The special solution can be expressed in terms of GFs which are defined as the solution to the DEQ where the inhomogeneity is a Dirac function:

$$\hat{D}_x G(x, x') = \delta(x - x') \tag{2.7}$$

A special solution can then be constructed by

$$y_{spec} = \int G(x, x') f(x') dx \tag{2.8}$$

for any inhomogeneity $f(x)$.

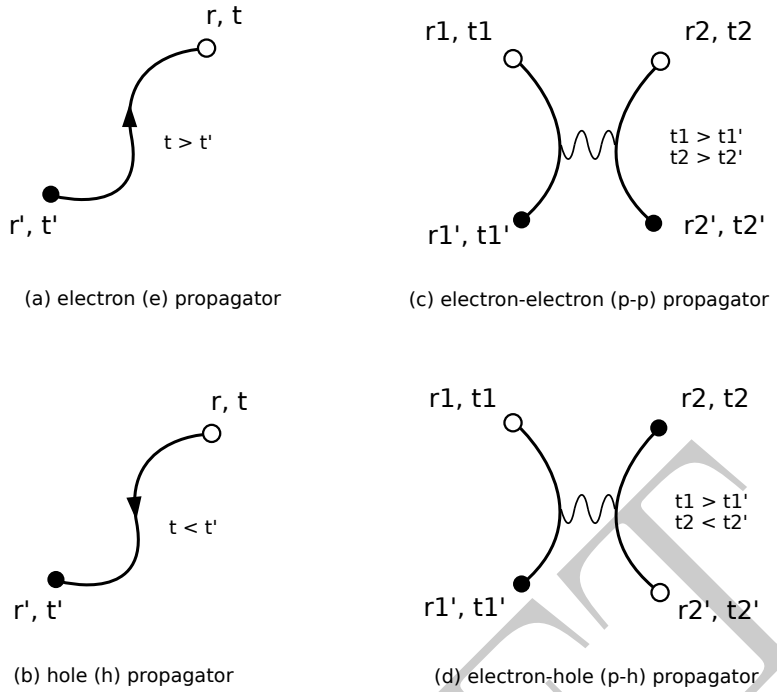


Figure 2.2: Hierarchy of Green's functions

The Schrödinger equation is also a differential equation where the inhomogeneity takes the role of the external perturbation V

$$\left[i\frac{\partial}{\partial t} + \frac{1}{2}\nabla^2\right]\Psi(\mathbf{r},t) = V(\mathbf{r},t)\Psi(\mathbf{r},t) \tag{2.9}$$

The wave function may then be expressed by

$$\Psi(\mathbf{r},t) = \int G(\mathbf{r},t;\mathbf{r}',t')\Psi \tag{2.10}$$

The GF has the effect of *propagating* the wave function from a given time and position to another time and space coordinate. GFs are therefore also known as *propagators*.

The MBGFs form a hierarchy, in which the one-particle GFs are the lowest rank (Figure 2.2). One-particle GFs can be used to extract information on 1-electron processes such as ionization and electron attachment. Two-particle GFs form the next step in the hierarchy, and allow to gain information on two-particle processes such as electron excitation (electron-hole) and two-electron ionization (electron-eletron).

One-electron Propagator

To see how GFs can be used for excited state analysis, consider the 1-electron propagator in the time domain

$$G_{pq}(t,t') = -i\Theta(t-t')\left\langle\Psi_0\right|\hat{T}(a_p[t]a_q^\dagger[t'])\left|\Psi_0\right\rangle \tag{2.11}$$

whith Θ as the Heavyside step function, and the time-ordering operator

$$\hat{T} = \begin{cases} a_p[t]a_q^\dagger[t'] & \text{for } t > t' \\ -a_q^\dagger[t']a_p[t] & \text{for } t < t' \end{cases} \tag{2.12}$$

which plays the role of conserving symmetry with respect to time. It is useful to switch to the energy representation of the GF by Fourier transformation

$$G_{pq}(\omega) = \underbrace{\sum_n \frac{\langle\Psi_0|\, c_p \,|\Psi_n^{N+1}\rangle \langle\Psi_n^{N+1}|\, c_q^\dagger \,|\Psi_0\rangle}{\omega + E_0 - E_n^{N+1} + i\eta}}_{G^+(t,t')} + \underbrace{\sum_n \frac{\langle\Psi_0|\, c_q^\dagger \,|\Psi_n^{N-1}\rangle \langle\Psi_n^{N-1}|\, c_p \,|\Psi_0\rangle}{\omega + E_n^{N-1} - E_0 - i\eta}}_{G^-(t,t')} \tag{2.13}$$

also known as the spectral, energy or Lehmann representation. The superscripts $N+1$ and $N-1$ indicate the addition or removal of an electron form the N-electron wave function. The left-hand sum G^+ describes electron attachment and the right-hand term G^- describes electron detachment (ionization). The singularities or *poles* of the spectral representation give the nth electron affinity and ionization energy

$$A_n = E_0 - E_n^{N+1} \tag{2.14}$$

$$I_n = E_n^{N-1} - E_0 \tag{2.15}$$

Moreover, the transition strengths (or pole strengths) are given by the spectroscopic factors

$$x_p^{(n)} = \langle\Psi_0|\, c_p \left|\Psi_n^{N+1}\right\rangle, \quad n \in \{N+1\} \tag{2.16}$$

$$x_p^{(n)} = \left\langle\Psi_n^{N-1}\right|\, c_p \,|\Psi_0\rangle\,, \quad n \in \{N-1\} \tag{2.17}$$

By analyzing the 1e-GF, it is therefore possible compute the 1-particle excitation spectrum.

Polarization Propagator

A solution to the single-particle SEQ can be given directly by integrating the GFs. For many-electron systems however, one- and two-particle GFs are only building blocks for many-body propagators. The 1p and 2p GFs allow to introduce the particle-hole response function

$$R_{pq,uv}(t_1,t_2;t_1',t_2') = G_{pq,uv}(t_1,t_2;t_1',t_2') - G_{pu}(t_1,t_1')G_{qv}(t_2,t_2') \tag{2.18}$$

also known as the two-particle correlation function. It is the variational derivative of the 1p-GF with respect to an external perturbation $V(t_1,t_2)$, for example in the form of an incoming light quantum [50]. Similarly to the 1p-GF, analyzing the ph response function gives information on the excited state. It can be evaluated directly via the Bethe-Salpeter equations [51, 52], but their dependency on four time variables make them difficult to solve. Fortunately, the same information is already contained in the *polarization propagator* defined by

$$\Pi(t,t') = \lim_{\substack{t_1\to t_1'=t \\ t_2\to t_2'=t'}} iR(t_1,t_2;t_1',t_2') \tag{2.19}$$

The spectral representation of Π takes the form

$$\Pi_{p,q;r,s} = \underbrace{\sum_{n\neq 0} \frac{\langle\Psi_0|\, \hat{c}_q^\dagger \hat{c}_p \,|\Psi_n\rangle \langle\Psi_n|\, \hat{c}_r^\dagger \hat{c}_s \,|\Psi_0\rangle}{\omega - (E_n - E_0) + i\eta}}_{\Pi_+(\omega)} + \underbrace{\sum_{n\neq 0} \frac{\langle\Psi_0|\, \hat{c}_r^\dagger \hat{c}_s \,|\Psi_n\rangle \langle\Psi_n|\, \hat{c}_q^\dagger \hat{c}_p \,|\Psi_0\rangle}{-\omega - (E_n - E_0) + i\eta}}_{\Pi_-(\omega)} \tag{2.20}$$

Here, the poles correspond to the excitation energies $\omega_n = E_n - E_0$ and the spectroscopic factors give the transtion strengths. The polarization propagator is therefore all one needs to evaluate absorption or emission spectra of molecules. The left and right hand terms are related by

$$\Pi(-\omega)^{\dagger}_{+} = \Pi_{-}(\omega) \tag{2.21}$$

Up to this point, the exact wave function was used in the expression for the propagators. To actually be able to compute the propagators, approximations need to be applied. There are a couple of choices. Coupled cluster linear response theory inserts the CC ansatz for the wave fucntion and explicitly evaluates expressions for the polarization propgator truncated to a given level of excitations (LR-CCSD, LR-CCSDT etc.).

Alternatively, the polarization propagator may be approached using perturbation theory.

Diagrammatic Perturbation

Similarly to the wave function in RSPT, the polarization propagator can be expanded as

$$\Pi = \Pi^{(0)} + \Pi^{(}1) + \Pi^{(2)} + \dots \tag{2.22}$$

The same partitioning of the Hamiltonian is used as in RSPT

$$\hat{H} = \hat{H}_0 + \hat{U} \tag{2.23}$$

with the expressions for the wave functions and their energies given in Equations ... and ...

One may then evaluate the series 2.22 using either Rayleigh-Schrödinger perturbation theory or the Gell-Mann Low theorem [2] to obtain master equations for $\Pi^{(n)}$. However, these eqautions are very tedious to solve, even more so than for Møller Plesset, due to the rapidly increasing number of nested terms for higher n. For this reason, diagrams were introduced to better keep track of the contributions at a given level.

Diagrams were originally conceived by Feynman, and are a pictorial representation of mathematical expressions for particle interactions. Over the years, many different types of diagrams were proposed, such as Goldstone, Abrikosov or Hugenholtz diagrams. Each type has its own set of rules on how to construct them and translate them into formulas for a given problem. There is no formal proof: Feynman first worked out the rules by trial and error [53], and later refined the model.

Figure ... shows the Feynman diagrams (in Abrikosov notation) for the polarization propagator up to second order. Each line represents a free particle (electron or hole). Lines with the arrow pointing up are also known as particle lines, while those with the arrow pointing down are known as hole lines. Here, the particle lines represent the time evolution of the electron between t and t'. The perturbation $\hat{V}$ is represented by dots in the diagrams, with the total number of dots indicating the perturbation order of the diagram. Each dot contributes a factor of $V_{rs[r's']} = \langle rs|\,\hat{V}\,|r's'\rangle - \langle rs|\,\hat{V}\,|s'r'\rangle$ to the mathematical expression of the diagram, where r, s and r', s' are incoming and outgoing fermion lines. Each vertex contributes a free one-particle Green's function $G^0_x(t, t')$. Further rules need to be applied to get the correct sign factors from the direction of the

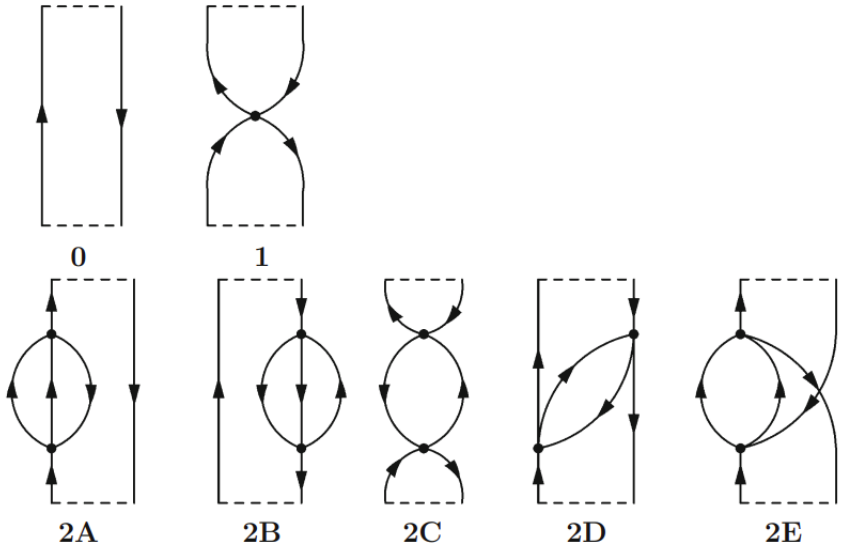


Figure 2.3: Feynman diagrams in Abrikosov notation for the polarization propagator through second order. Taken from [2].

lines. As an example, consider the first order expression of the polarization propagator

$$\Pi^{(1)}_{rs,r's'}(t,t') = \sum_{-\infty}^{\infty} \hat{V}_{rs[r's']} G^0_r(t,t_1) G^0_s(t_1,t) G^0_{r'}(t_1,t') G^0_{s'}(t',t_1) dt_1 \tag{2.24}$$

Equation 2.24 can then be transformed to the energy representation. Alternatively, Goldstone diagrams can be used where the set of rules directly gives the spectral instead of the time representation.

2.3.2 The ADC scheme

The polarization propagator cannot be directly "measured". To establish a bridge between theory and experiments, the *transition function* is introduced as

$$T(\omega) = D^\dagger \mathbf{\Pi}_+ D \tag{2.25}$$

where $\hat{D}$ is an arbitrary operator. The quantity measured during experiments is the *spectral function*, given by

$$f(\omega) = \frac{1}{\pi} Im\{T(\omega)\} \tag{2.26}$$

In the algebraic diagrammatic construction method, the transition function is reformulated as

$$T(\omega) = \mathbf{F}^\dagger \mathbf{\Gamma}(\omega) \mathbf{F} \tag{2.27}$$

where $\mathbf{F}$ are the modified transition moments and the non-diagonal matrix $\mathbf{\Gamma}$ is given by

$$\mathbf{\Gamma}(\omega) = [\omega \mathbf{1} - (\mathbf{K} + \mathbf{C})] = [\omega \mathbf{1} - \mathbf{M}] \tag{2.28}$$

with the ADC matrix $\mathbf{M}$. Writing the transition function, the modified transition moments and $\mathbf{M}$ as a perturbation expansion

$$T(\omega) = \sum_{n=0}^{\infty} T^{(n)}(\omega) = \sum_{n=0}^{\infty} D^{\dagger}\mathbf{\Pi}_{+}^{(n)} D\mathbf{F} \qquad = \sum_{n=0}^{\infty} \mathbf{F}^{(n)} \tag{2.29}$$

$$\mathbf{M} = \mathbf{K} + \sum_{n=1}^{\infty} \mathbf{C}^{(n)} \tag{2.30}$$

the nth order approximations to the transition function read

$$T^{(0)}(\omega) = \mathbf{F}^{(0)\dagger} \left[\omega\mathbf{1} - \mathbf{K}\right]^{-1} \mathbf{F}^{(0)} \tag{2.31}$$

$$\begin{aligned} T^{(1)}(\omega) = {} & \mathbf{F}^{(0)\dagger} \left[\omega\mathbf{1} - \mathbf{K}\right]^{-1} \mathbf{C}^{(1)} \left[\omega\mathbf{1} - \mathbf{K}\right]^{-1} \mathbf{F}^{(0)} + \mathbf{F}^{(1)\dagger} \left[\omega\mathbf{1} - \mathbf{K}\right]^{-1} \mathbf{F}^{(0)} \\ & + \mathbf{F}^{(0)\dagger} \left[\omega\mathbf{1} - \mathbf{K}\right]^{-1} \mathbf{F}^{(1)} \end{aligned} \tag{2.32}$$

$$\begin{aligned} T^{(2)}(\omega) = {} & \mathbf{F}^{(1)\dagger} \left[\omega\mathbf{1} - \mathbf{K}\right]^{-1} \mathbf{F}^{(1)} + \mathbf{F}^{(0)\dagger} \left[\omega\mathbf{1} - \mathbf{K}\right]^{-1} \mathbf{C}^{(2)} \left[\omega\mathbf{1} - \mathbf{K}\right]^{-1} \mathbf{F}^{(0)} \\ & + \mathbf{F}^{(0)\dagger} \left[\omega\mathbf{1} - \mathbf{K}\right]^{-1} \mathbf{C}^{(1)} \left[\omega\mathbf{1} - \mathbf{K}\right]^{-1} \mathbf{C}^{(1)} \left[\omega\mathbf{1} - \mathbf{K}\right]^{-1} \mathbf{F}^{(0)} \\ & + \mathbf{F}^{(1)\dagger} \left[\omega\mathbf{1} - \mathbf{K}\right]^{-1} \mathbf{C}^{(1)} \left[\omega\mathbf{1} - \mathbf{K}\right]^{-1} \mathbf{F}^{(0)} \\ & + \mathbf{F}^{(0)\dagger} \left[\omega\mathbf{1} - \mathbf{K}\right]^{-1} \mathbf{C}^{(1)} \left[\omega\mathbf{1} - \mathbf{K}\right]^{-1} \mathbf{F}^{(1)} \end{aligned} \tag{2.33}$$

By comparing the above nth order expression of the transition operator $T(\omega)^{(n)}$ with the mathematical expression of $\mathbf{D}^{\dagger}\mathbf{\Pi}(\omega)\mathbf{D}$ derived using the diagrammatic perturbation of the polarization propagator, algebraic expressions can be *constructed* for the transition moments $\mathbf{F}$, and the matrices $\mathbf{K}$ and $\mathbf{C}$, hence the name algebraic diagrammatic construction.

2.3.3 Structure of the ADC matrix

At its core, ADC reduces to the eigenvalue problem

$$\mathbf{MX} = \mathbf{X\Omega} \tag{2.34}$$

The solution gives the vertical excitation energies $\mathbf{\Omega}$ and the eigenvectors $\mathbf{X}$. Figure 2.4 shows the structure of the ADC matrix $\mathbf{M}$ up to third order. Each second level n adds an additional higher excitation manifold to the matrix. ADC(0) and ADC(1) include only singles, while ADC(2) and ADC(3) also include doubles.

The ADC(0) matrix contains only the Hartree-Fock orbital energy differences:

$$M^{(0)}_{ia,jb} = K_{ia,jb} = \delta_{ij}\delta_{ab}(\epsilon_i - \epsilon_a) \tag{2.35}$$

The ADC(1) matrix adds the first order expression for $\mathbf{C}$, and is identical to the CIS matrix:

$$M^{(1)}_{ia,jb} = K_{ia,jb} + C^{(1)}_{ia,jb} = \delta_{ij}\delta_{ab}(\epsilon_i - \epsilon_a) - \langle ij| \, |ab\rangle \tag{2.36}$$

The ADC(2) matrix has additional second order contributions to the p-h block, and approximates the 2h-1p, 1h-2p to first order and the 2p-2h to zeroth order.

$$C^{(2)}_{ijab} = C^{(2)A}_{ijab} + C^{(2)B}_{ijab} + C^{(2)C}_{ijab} \tag{2.37}$$

$$C^{(1)}_{ia,jkcl} = \langle kl| \, |id\rangle \, \delta_{ac} - \langle kl| \, |ic\rangle \, \delta_{ad} - \langle al| \, |cd\rangle \, \delta_{ik} + \langle ak| \, |cd\rangle \, \delta_{il} \tag{2.38}$$

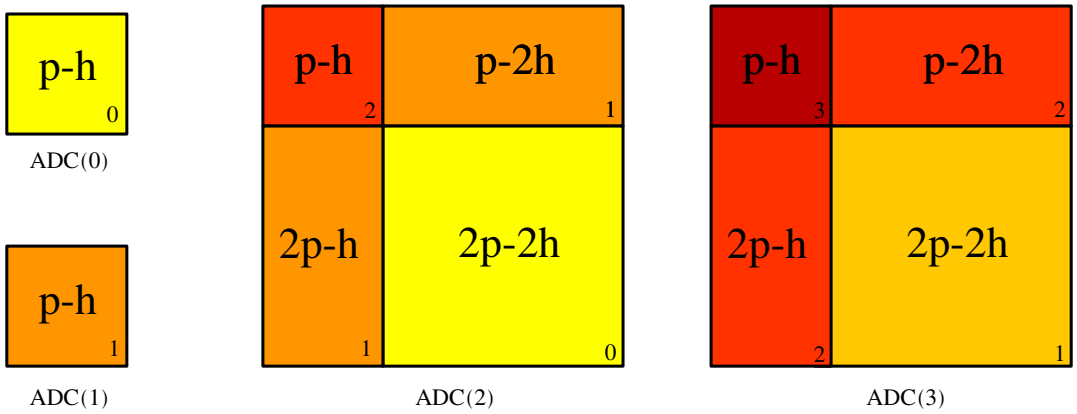


Figure 2.4: Structure of the ADC matrix from zeroth through third order. The number in each block indicates the perturbation order.

$$C^{(1)}_{iajb,kc} = \langle kb|\,|ij\rangle\,\delta_{ac} - \langle ka|\,|ij\rangle\,\delta_{bc} - \langle ab|\,|cj\rangle\,\delta_{ik} + \langle ab|\,|ci\rangle\,\delta_{jk} \tag{2.39}$$

$$K_{iajb,kcld} = (\epsilon_a - \epsilon_i + \epsilon_b - \epsilon_j)\delta_{ac}\delta_{bd}\delta_{ik}\delta_{jl} \tag{2.40}$$

with

$$C^{(2)A}_{ijab} = \frac{1}{4}\delta_{ij}\sum_{ckl}\left[\hat{t}_{ackl}\,\langle kl|\,|bc\rangle + \langle ac|kl\rangle\,\hat{t}_{klbc}\right] \tag{2.41}$$

$$C^{(2)B}_{ijab} = \frac{1}{4}\delta_{ab}\sum_{cdk}\left[\hat{t}_{cdik}\,\langle jk|cd\rangle + \langle cd|ik\rangle\,\hat{t}_{jkcd}\right] \tag{2.42}$$

$$C^{(2)C}_{ijab} = -\frac{1}{2}\sum_{ck}\left[\hat{t}_{acik}\,\langle jk|bc\rangle + \langle ac|ik\rangle\,\hat{t}_{jkbc}\right] \tag{2.43}$$

and the anti-symmetrized MP2 amplitudes

$$\hat{t}_{ijab} = \frac{\langle ij|\,|ab\rangle}{\epsilon_a + \epsilon_b - \epsilon_i - \epsilon_j} \tag{2.44}$$

Approximating the 2p-2h block to first order in the ADC(2) matrix, i.e. swapping the 2p-2h block with the one from the ADC(3) matrix, gives the so-called extended ADC(2) scheme (ADC(2)-x). It is an *ad-hoc* extension without rigoruous theoretical justification [54].

2.3.4 Solving the Eigenvalue Problem

The eigenvalue problem 2.34 is typically solved using the Davidson procedure to extract the first few eigenvalues, analogous to configuration interaction. Rather than constructing the whole ADC matrix, closed expressions are derived for the matrix-vector products of the different blocks with a general trial vector $\mathbf{u}$. For computational considerations, the matrix-vector product is split into its individual components which are then multiplied by the sub-blocks of the ADC matrix. In the case of ADC(2) and ADC(3), the components are limited to singles and doubles contributions:

$$r_{ia} = A_{ia,jb}u_{jb} + A_{ia,jbkc}u_{jbkc} \tag{2.45}$$

$$r_{iajb} = A_{iajb,kc}u_{kc} + A_{iajb,kcld}u_{kcld} \tag{2.46}$$

While the Davidson procedure allows to circumvent storing the whole ADC matrix, the storage of the trial vectors can still be a major memory bottle-neck for ADC(2) and beyond. At second and third order, the doubles part of the vectors scale with $n_{occ}^2 n_{vir}^2$, and take up as much space as the MP2 amplitudes. As the Davidson subspace grows, so does the number of trial vectors. Techniques such as *subspace collapse* (Section 11.2) impose a maximum to the number of trial vectors held in memory, which helps to better estimate the total storage size needed by an ADC calculation, although it increases the total number of iterations to convergence.

An alternative technique to reduce the memory footprint of the Davdison diagonalization is *doubles-folding*. Consider the doubles part of the MVP which is computed as

$$r_{iajb} = A_{iajb,kc}u_{kc} + A_{iajb,kcld}u_{kcld} = \omega u_{iajb} \tag{2.47}$$

By refactoring the above expression, the doubles component of $\mathbf{u}$ can be reformulated in terms of its singles component as

$$u_{iajb} = \frac{A_{iajb,kc}u_{kc}}{\omega - A_{iajb,iajb}} \tag{2.48}$$

This technique is limited to ADC(2) only, where the doubles-doubles block is diagonal. Substituting 2.48 into the singles expression of the MVP, and using the explicit formulas for the doubles-doubles block gives

$$r_{ia} = A_{ia,jb}u_{jb} + A_{ia,jbkc}\frac{A_{jbkc,ld}u_{ld}}{\omega - \epsilon_j - \epsilon_k + \epsilon_b + \epsilon_c} \tag{2.49}$$

The doubles part of the MVP is computed on-the-fly and does not need to be explicitly stored, reducing the overall memory requirements of the Davdison diagonalization to $n_{occ}n_{vir}$. Doubles-folding corresponds to a multiplication of the singles vectors with an *effective* ADC matrix which depends on the eigenvalue ω

$$\mathbf{r}_{\mu_1} = \mathbf{A}(\omega)_{\mu_1\nu_1}\mathbf{u}_{\nu_1} \tag{2.50}$$

One drawback of doubles-folding is that a modified Davidson procedure is necessary to solve this *pseudo* eigenvalue problem (see **??**) due to the dependence on the excitation energy ω.

2.3.5 Intermediate states

An alternative route to deriving the ADC working equations is via the intermediate state representation [55, 56, 57].

The previous derivation showed that the eigenvalues of the ADC matrix $\mathbf{M}$ correspond to the excitation energies, and that it can be expanded in a perturbation series. These features suggest that $\mathbf{M}$ is a representation of the energy-shifted Hamiltonian

$$\mathbf{M} = \mathbf{H} - E_0 \tag{2.51}$$

with the matrix elements

$$M_{IJ} = -\langle \widetilde{\Psi}_I | \, \hat{H} - E_0 \, | \widetilde{\Psi}_J \rangle \tag{2.52}$$

Here, the space of the shifted Hamiltonian is spanned by a set of *intermediate states*. Starting from the set of *correlated excited* (CE) states

$$|\Psi_I^{\#}\rangle = \hat{C}_I \, |\Psi_0\rangle \tag{2.53}$$

with the excitation operators

$$\{\hat{C}_I\} = \{a_a^\dagger a_i; a_b^\dagger a_j c_a^\dagger c_i; \ldots\} \tag{2.54}$$

the intermediate states are obtained by a step-wise Gram-Schmidt orthogonalization of the CE states. The ground state $|\Psi_0\rangle$ is approximated by MPPT. Constructing the intermediate states from the MPn ground state wave function and evaluating the matrix elements according to 2.52 gives the nth order ADC matrix. For this reason, ADC is also known as "excited state method for Møller-Plesset".

2.3.6 Spin-Opposite Scaled ADC

The spin-opposite scaling method previously applied to MP2 and CC2 can be expanded to ADC(2) as well. There are two version of SOS-ADC(2): the version which will be referred to as "standard" SOS-ADC(2) derived from the SOS-CC2 linear response equations [25], and ISR-SOS-ADC(2) derived from SOS-MP2 using the intermediate state representation [6]. Standard SOS-ADC(2) introduces the following modifications to the ADC(2) matrix:

1. The same-spin contributions of antisymmetrized MP2 amplitudes are ignored, and the opposite-spin components are scaled up:

$$\hat{t}_{IAJB}^{SOS} = c_{os} \hat{t}_{IAJB} \left(1 - \delta_{\sigma(I)\sigma(J)}\right) \tag{2.55}$$

   where $\sigma(x)$ gives the spin of x, and with the amplitudes given by

$$\hat{t}_{IAJB} = \frac{\langle IJ| \, |AB\rangle}{\epsilon_A + \epsilon_B - \epsilon_I - \epsilon_J} \tag{2.56}$$

2. All same-spin entries of the 2p-1h and 1p-2h blocks of the ADC(2) matrix are deleted ($\alpha\alpha\alpha\alpha$ and $\beta\beta\beta\beta$), and the remaining blocks are scaled up:

$$M_{ia,kcld} = c_{osc} \left(\langle kl| \, |id\rangle \, \delta_{ac} - \langle kl| \, |ic\rangle \, \delta_{ac} - \langle al| \, |cd\rangle \, \delta_{ik} + \langle ak| \, |cd\rangle \, \delta_{il}\right) \left(1 - \delta_{\sigma(k)\sigma(l)}\right) \tag{2.57}$$

$$M_{iajb,kc} = c_{osc} \left(\langle kb|ij\rangle \, \delta_{ac} - \langle ka|ij\rangle \, \delta_{bc} - \langle ab|cj\rangle \, \delta_{ik} + \langle ab|ci\rangle \, \delta_{jk}\right) \left(1 - \delta_{\sigma(i)\sigma(j)}\right) \tag{2.58}$$

   where c_{osc} is the opposite-spin coupling constant, typcially set to 1.15 [6] or 1.17 (diss).

For open-shell molecules, this drastically reduces the size of the matrix, reducing the prefactor of the method. By applying density fitting, the total scaling can be further reduced by an order of magnitude [25].

ISR-SOS-ADC(2) does not modifiy the off-diagonal blocks of the ADC(2) and only replaces the amplitudes as in Eqaution 2.55, and therefore offers no substantial improvement.

2.3.7 Performance and Accuracy

Table 2.1 lists the formal scaling, mean errors and standard deviation of excitation energies for the ADC(n) methods. ADC(2), similarly to MP2, offers an economical way of computing excited state properties compared to other excited state methods with similar accuracy. ADC(2)-x and ADC(3) have the same scaling factor, but ADC(2)-x has a lower prefactor.

The ADC methods offer high accuracy and high precision on the order of a few tenths of eV . The SOS method can significantly reduce the errors, but it should be kept in mind that the spin coefficients were fitted to the benchmark set, and similar accuracy is not guaranteed for other systems.

Method	Scaling	Singlets	Triplets
ADC(2)	$\mathcal{O}(N^5)$	0.22 ± 0.38 (62)	0.12 ± 0.16 (62)
SOS-ADC(2)	$\mathcal{O}(N^5)$	0.00 ± 0.15 (87)	0.06 ± 0.10 (87)
ADC(2)-x	$\mathcal{O}(N^6)$	-0.70 ± 0.37 (62)	-0.55 ± 0.20 (62)
SOS-ADC(2)-x	$\mathcal{O}(N^6)$	-0.11 ± 0.18 (87)	-0.04 ± 0.12 (87)
ADC(3)	$\mathcal{O}(N^6)$	0.12 ± 0.28 (64)	-0.18 ± 0.16 (64)

Table 2.1: Mean absolute errors (MAE) and deviations (in eV) for closed-shell molecules at various levels of theory. [a] [5], [b] [6], [c] [7]

2.4 Response Theory

Response theory is a popular tool similar to propagators that provides methods for computing the response of a molecule to an external, time-dependent perturbation, such as an electromagnetic field. It can be applied to different levels of theory, such as Hartree-Fock, DFT or Coupled Cluster, to gain information on various excited state properties.

2.4.1 Exact Response Theory

Consider a molecular system described by the time-independent Hamiltonian $\hat{H}_0$ with eigenfunctions $|\Psi_0\rangle$ exposed to an external perturbation $\hat{V}$ given by [58]

$$\hat{V}(t) = \int_{-\infty}^{\infty} \hat{V}^{\omega} e^{i\omega t} d\omega \tag{2.59}$$

where $\hat{V}^{\omega}$ is the representation of the external perturbation in the frequency domain, and ϵ is a real positive infinitesimal. It has the role of slowly "switching on" the perturbation as time progresses. For $t \to -\infty$, the perturbation is zero, and at $t \to \infty$, the perturbation is fully applied. This slow gradual switching makes sure that the process is *adiabatic*.

The time-dependent wave function may be expanded in orders of the perturbation $\hat{V}(t)$ as

$$|\Psi(t)\rangle = |\Psi_0\rangle + |\Psi^{(1)}(t)\rangle + |\Psi^{(2)}(t)\rangle + \dots \tag{2.60}$$

which can be determined using Eherenfest's theorem. Using this wave function expansion, the expectation value of a time-independent operator $\hat{A}$ reads

$$\begin{aligned}\langle\Psi(t)|\,\hat{A}\,|\Psi(t)\rangle = \langle\Psi_0|\,\hat{A}\,|\Psi_0\rangle + \int_{-\infty}^{\infty} \underbrace{\langle\langle\hat{A};\hat{V}^{\omega_1}\rangle\rangle}_{\text{linear response}} e^{(-i\omega_1+\epsilon)t} d\omega_1 \\ + \frac{1}{2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \underbrace{\langle\langle\hat{A};\hat{V}^{\omega_1},\hat{V}^{\omega_2}\rangle\rangle}_{\text{quadratic reponse}} e^{-i(\omega_1+\omega_2)+2\epsilon)t} d\omega_1 d\omega_2 + \ldots\end{aligned} \tag{2.61}$$

The expansion coefficients $\langle\langle\hat{A};\cdot\rangle\rangle$ are known as *response functions*. Different orders (linear, quadratic...) describe different processes. The linear response function may be used to describe single-photon absorption and polarizability, while the quadratic response function is needed to describe two-photon absorption and hyperpolarizability.

The spectral representation of the linear response function takes the form

$$\langle\langle\hat{A};\hat{B}\rangle\rangle = \sum_k \frac{\langle\Psi_0|\,\hat{A}\,|\Psi_k\rangle\langle\Psi_k|\,\hat{B}\,|\Psi_0\rangle}{\omega - E_n + E_0} - \frac{\langle\Psi_0|\,\hat{B}\,|\Psi_k\rangle\langle\Psi_k|\,\hat{A}\,|\Psi_0\rangle}{\omega + E_n - E_0} \tag{2.62}$$

and can be analyzed similarly to the polarization propagator: the poles of the function give the excitation energy

$$\omega_i = E_i - E_0 \tag{2.63}$$

and the residues give information about the transition moments

$$\lim_{\omega\to\omega_i}(\omega-\omega_i)\langle\langle\hat{A};\hat{B}\rangle\rangle = \langle\Psi_0|\,\hat{A}\,|\Psi_i\rangle\langle\Psi_i|\,\hat{B}\,|\Psi_0\rangle \tag{2.64}$$

for the ith excited state. The linear response function and the polarization propagator are related by [2]

$$\langle\langle\hat{A};\hat{B}\rangle\rangle = \sum_{rsr's'} A_{rs}\mathbf{\Pi}B_{r's'} \tag{2.65}$$

The expressions for response functions are exact, and need to be evaluated by introducing approximations. Similar to the ADC scheme, finding the poles and residues of the response function ultimately reduces to an eigenvalue problem of the form

$$\mathbf{Av} = \mathbf{v\Omega} \tag{2.66}$$

where $\mathbf{A}$ can be symmetric (HF,DFT) or non-symmetric (CC).

2.4.2 Time-Dependent Hartree-Fock

There are many different routes for deriving the expressions for the matrix elements of $\mathbf{A}$ for linear response time-dependent Hartree-Fock (TDHF) (cf. **??**), which all lead to the same eigenvalue problem given by

$$\begin{bmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{B}^* & \mathbf{A}^*\end{bmatrix}\begin{bmatrix}\mathbf{X}\\ \mathbf{Y}\end{bmatrix} = \omega\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}\begin{bmatrix}\mathbf{X}\\ \mathbf{Y}\end{bmatrix} \tag{2.67}$$

where **A** is the matrix of single excitations, and **B** couples the excitations with the de-excitations. The matrix elements are given by

$$A_{IA,JB} = \delta_{IA,JB}(\epsilon_A - \epsilon_I) + |IJ\rangle\, AB \tag{2.68}$$
$$B_{IA,JB} = |IJ\rangle\, AB \tag{2.69}$$

Setting the coupling block **B** to zero, the TDHF equations reduce to the CIS eqautions. Even if TDHF can therefore be seen as an extension to CIS, it does not give a considerable improvement. Over the years, it has fallen into disuse.

Linear response TDHF is equivalent to the *random phase approximation*.

2.4.3 Time-Dependent DFT

The foundations of time-dependent DFT will not be discussed here. The reader is referred to [59] and references therein for more details.

The TDDFT linear response equations are similar in structure to TDHF, reducing to the same eigenvalue problem 2.67, with two different blocks **A** and **B** given by

$$A_{IA,JB} = \delta_{IA,JB}(\epsilon_A - \epsilon_I) + \langle IJ|AB\rangle + \langle IJ|\, \hat{f}_{xc}\, |AB\rangle \tag{2.70}$$
$$B_{IA,JB} = \langle IJ|AB\rangle + \langle IJ|\, \hat{f}_{xc}\, |AB\rangle \tag{2.71}$$

Here, the exchange contributions are replaced by the so-called *xc kernel*. In the adiabatic local density approximation (ALDA), the time dependent xc kernel is substituted by a time-independent kernel

$$\langle IJ|\, \hat{f}_{xc}\, |AB\rangle = \int \phi_i^*(\mathbf{r})\phi_j(\mathbf{r}') \frac{\partial^2 E_{xc}}{\partial\rho(\mathbf{r})\rho(\mathbf{r}')} \phi_a(\mathbf{r})\phi_b^*(\mathbf{r}') \tag{2.72}$$

which allows the use of standard xc functionals for the ground state.

Since its introduction, TDDFT has evolved to become the most prominent method for computing excited state energies and transition moments. It has a computational cost on the same order as CIS, with an error of ≈ 0.3 eV [60] for low-lying valence states. However TDDFT is not a panacea: excitation energies for Rydberg states, valence states of molecules with extended π-systems, doubly excited states and charge-transfer states exhibit errors on the order of several eV.

2.4.4 Coupled Cluster

The derivation of the coupled cluster response equations is again a very lengthy and complex process [58, 61]. The most important steps will be summarized in this section.

For a molecular system in the presence of a static external perturbation, such as a constant electric or magnetic field with strength parameter λ, the Hellmann-Feynman theorem relates the expectation value of $\hat{X}_\lambda$ to the energy derivative by

$$\frac{\partial E}{\partial \lambda_x} = \langle\Psi|\, \frac{\partial \hat{H}}{\partial \lambda_x}\, |\Psi\rangle = \langle\Psi|\, \hat{X}\, |\Psi\rangle \tag{2.73}$$

By perturbation expansion of $\hat{X}$, the nth order property can then be related to the nth order derivative of the wave function energy, which is in most cases is more readily

derived than the expectation value. For time-dependent perturbations, the theorem can be reformulated as

$$\frac{\partial\{Q\}_T}{\partial\lambda_x} = \langle\Psi(t)|\,\hat{X}(t)\,|\Psi(t)\rangle \tag{2.74}$$

where $\{Q\}_T$ is the time-averaged *quasi-energy* given by

$$\{Q\}_T = \frac{1}{T}\int_0^T \langle\Psi(t)|\,(\hat{H} - i\frac{\partial}{\partial t})\,|\Psi(t)\rangle \tag{2.75}$$

The time-averaged quasi-energy is the analog of the ground state energy for static perturbations in Equation 2.73. Using the perturbation expansion 2.61 for the operator, the first, second, ... derivative of the quasi-energy can be related to the linear, quadratic ... response function. Similarly to the energy derivative, expressions for quasi-energy derivatives are more easily evaluated. The CC response eqautions are then obtained by constructing a Lagrangian of the CC quasi-energy.

Analysis of the CC response equations leads to the non-symmetric eigenvalue problem

$$\mathbf{AR} = \mathbf{\Omega R} \tag{2.76}$$

$$\mathbf{LA} = \mathbf{R\Omega} \tag{2.77}$$

where $\mathbf{R}$ and $\mathbf{L}$ are the right and left eigenvectors respectively.

Analogous to ground state calculations, a hierarchy of CC response methods is obtained by truncating the excitation operator $\hat{T}$ to singles, doubles, triples, ... to yield CCS-LR, CCSD-LR, CCSDT-LR. Approximate CC methods may also be used, such as CC2, CC3 or CCSD(T). The formal scaling of the different methods is equal to their respective scaling for ground states, although with a higher prefactor. CC2 excitation energies are similar in accuracy to ADC(2). CCSD is better than ADC(2), and CC3 is slightly better than ADC(3) [62].

2.4.5 Connection between ADC(2) and CC2-LR

An interesting relationship can be established between ADC(2) and CC2-LR [63]. Consider the expression for the CC2 Jacobian:

$$\mathbf{A}^{CC2-LR} = \begin{pmatrix} \langle\mu_1|\left[(\hat{\bar{H}} + [\hat{\bar{H}}, \hat{T}_2]), \tau_{\nu_1}\right]|HF\rangle & \langle\mu_1|\left[\hat{\bar{H}}, \tau_{\nu_2}\right]|HF\rangle \\ \langle\mu_2|\left[\hat{\bar{H}}, \tau_{\nu_1}\right]|HF\rangle & \langle\mu_2|\left[(\hat{\bar{H}} + [\hat{F}, \hat{T}_2]), \tau_{\nu_2}\right]|HF\rangle \end{pmatrix} \tag{2.78}$$

where μ_1, μ_2 are the single and double excitation manifolds. By replacing the similarity-transformed Hamiltoninan $\hat{\bar{H}}$ by the Hamiltonian itself, the CIS(D) matrix reads

$$\mathbf{A}^{CIS(D)} = \begin{pmatrix} \langle\mu_1|\left[(\hat{\bar{H}} + [\hat{H}, \hat{T}_2]), \tau_{\nu_1}\right]|HF\rangle & \langle\mu_1|\left[\hat{H}, \tau_{\nu_2}\right]|HF\rangle \\ \langle\mu_2|\left[\hat{H}, \tau_{\nu_1}\right]|HF\rangle & \langle\mu_2|\left[(\hat{H} + [\hat{F}, \hat{T}_2]), \tau_{\nu_2}\right]|HF\rangle \end{pmatrix} \tag{2.79}$$

CIS(D) is a second-order perturbative correction of CIS which includes doubles contributions [64] scaling with $\mathcal{O}(N^5)$. The ADC(2) matrix is then obtained by symmetrization

$$\mathbf{A}^{ADC(2)} = \frac{1}{2}\left(\mathbf{A}^{CIS(D)} + (\mathbf{A}^{CIS(D)})^\dagger\right) \tag{2.80}$$

This relationship is especially useful for easily deriving approximate methods for ADC(2) from CC2, such as SOS-ADC(2), or local ADC(2).

2.5 Equation-of-Motion Coupled Cluster

An alternative way for describing the molecular response to an external field using coupled cluster is via the equation-of-motion (EOM) ansatz [65, 66, 67, 68, 69]. In the EOM approach, the target excited (R) and de-excited (L) wave functions are parameterized as

$$|\Psi_R\rangle = e^{\hat{T}}\hat{R}\,|\Psi_0\rangle \tag{2.81}$$

$$\langle\Psi_L| = \langle\Phi_0|\,\hat{L}e^{-\hat{T}} \tag{2.82}$$

with the excitation and de-excitation operators $\hat{R}$ and $\hat{L}$

$$\hat{R} = \sum r_\mu \hat{\tau}_\mu \tag{2.83}$$

$$\hat{L} = \sum l_\mu \hat{\tau}_\mu \tag{2.84}$$

where r and l are the excitation and de-excitation cluster amplitudes. The exact form of $\hat{R}$ and $\hat{L}$ depend on the nature of the reference and final states. The most common uses of EOM are for the calculation of excitation energies and ionization potentials. In the case of EOM-EE, with Ψ_0 taken as the Hartree-Fock wave function, $\hat{R}$ and $\hat{L}$ conserve the number of electrons and are given as

$$\hat{R}_{EE} = r_0 + \sum_{IA} r_I^A a_A^\dagger a_I + \frac{1}{4}\sum_{IAJB} r_{IJ}^{AB} a_A^\dagger a_I a_B^\dagger a_J + \ldots \tag{2.85}$$

$$\hat{L}_{EE} = l_0 + \sum_{IA} l_I^A a_I^\dagger a_A + \frac{1}{4}\sum_{IAJB} l_{IJ}^{AB} a_I^\dagger a_A a_J^\dagger a_B + \ldots \tag{2.86}$$

The (de-)excitation operators can be truncated similar to $\hat{T}$ to yield different approximations (EOM-CCS, EOM-CCSD...). The cluster amplitudes and the excitation energies are obtained by solving the non-Hermitian eigenvalue problem

$$\langle\mu|\,\hat{\bar{H}}\hat{R}\,|HF\rangle = 0 \tag{2.87}$$

$$\langle HF\hat{L}|\,\hat{\bar{H}} - E\,|HF\rangle = 0 \tag{2.88}$$

The excitation energies obtained by solving the LR-CC and EOM-CC eigenvalue problems are identical for pure CC models (CCSD, CCSDT...), but they give different results for transition moments and excited-state properties.

The EOM-CC methods, alongside ADC and CC-LR, are among the most accurate methods available for excited states.

Chapter 3

Local Correlation Methods (I): Tools and Concepts

While computational chemistry has emerged as a reliable experimental tool, the inherent steep scaling of its most accurate methods like coupled cluster or perturbation theory often imposes strict limits on the maximum molecular system size that can be treated. Even the Hartree-Fock method formerly scales with $O(N^4)$, and becomes prohibitively expensive for larger molecules if no further approximations are introduced. For post-Hartree-Fock methods, the major bottle necks are the transformation of the 2-electron repulsion integrals from the atomic orbital into the molecular orbital basis, and evaluation of the working equations using these integrals. The $OVOV$-type MO integrals, as they appear in coupled cluster and Møller-Plesset perturbation theory, are given by

$$(ia \mid jb) = \sum_{b}^{vir} C_{\sigma b} \sum_{a}^{vir} C_{\nu a} \sum_{j}^{occ} C_{\lambda j} \sum_{i}^{occ} C_{\mu i} \, (\mu\nu \mid \lambda\sigma) \tag{3.1}$$

The AO-MO transformation step scales quartically with system size. Over the years, several different strategies have been proposed to speed up this step. Rank-reduction approaches like density fitting or the Cholesky decomposition, split the 4-index integral tensor into a product of two 3-index tensors, which reduces the memory footprint and the *prefactor* of the transformation. Methods that exploit the nearsightedness of the electrons use a different molecular orbital representations, such as local molecular orbitals or natural orbitals, to obtain a more compact representation of the virtual MO space, and consequently reduce the *scaling*. The AO-MO transformation may also be completely skipped by reformulating the working equations in an atomic orbital basis and using sparsity to speed up the calculations.

This chapter introduces the most important tools used in local correlation methods which will be discussed in the next chapter.

3.1 Sparsity in Electronic Structure Theory

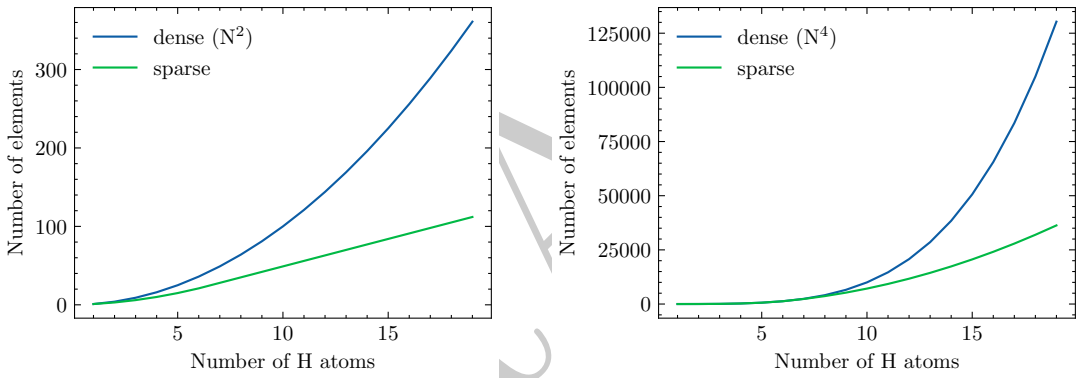
Sparsity is a core concept in electronic structure theory. Many of the most commonly encountered matrices and tensors exhibit some form of sparsity, for example, the 2-

electron repulsion integrals in the AO basis. This section analyses in detail the different possible types of sparsity.

3.1.1 Element-Wise Sparsity of Electron Integrals

Molecular electron integral evaluation can become prohibitively expensive for large systems, especially the four-dimensional electron-repulsion integral (ERI) tensor which formerly scales as $\mathcal{O}(N^4)$. It is therefore imperative to exploit the exponential decay of the GTO basis.

Consider a model system consisting of n hydrogen atoms arranged in a line, with a distance of 1 a_0 between one another, and a primitive 1s Gaussian function attached to each atom. Figure 3.1 shows the scaling behavior of the overlap and electron repulsion integrals for this toy system. A blue line is used to show the number of total elements, while the green line represents the number of significant integrals with an absolute value below 1e-10. From observing both graphs, it becomes apparent that for increasing number of atoms, many of the electron integrals can be ignored. Therefore, one only needs to store integrals above a certain threshold. This is also known as *element-wise sparsity*.



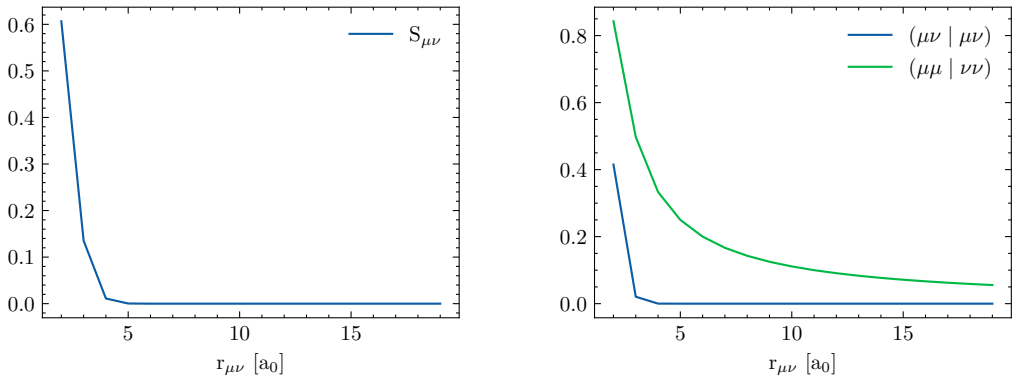
(a) (b)

Figure 3.1: (a) Number of significant entries (green line) in the overlap matrix for a hydrogen atom chain, with a threshold of 1e-10. The blue line shows the total number of elements for the dense matrix, which scale as N^2. (b) Number of significant entries (green line) in the electron repulsion integral tensor for a hydrogen atom chain, with a threshold of 1e-10. The blue line shows the total number of elements for the dense tensor, which scale as N^4.

Linear Scaling Overlap Integrals

While the overlap integrals formerly scale with $\mathcal{O}(N^2)$, it can be shown that the number of significant elements scales *linearly*. First, consider the product of two 1s GTOSs χ_A and χ_B, centered at $\mathbf{A}$ and $\mathbf{B}$, with exponents α and β. The Gaussian product theorem (GPT) states that the result is itself also a (scaled) Gaussian function

$$\chi(A,\alpha)\chi(B,\beta) = e^{-\alpha|\mathbf{r}-\mathbf{A}|^2} e^{-\beta|\mathbf{r}-\mathbf{B}|^2} = \kappa\chi(P,\alpha+\beta) \tag{3.2}$$



(a) (b)

Figure 3.2: (a) Magnitude of the overlap integral between two Gaussian 1s orbitals as a function of distance r (exponential decay). (b) Magnitude of the electron repulsion integral between two Gaussian 1s orbitals as a function of r. The short range interaction $(\mu\nu \mid \mu\nu)$ decays at a much faster rate with e^{-r^2}, compared to the long range interaction $(\mu\mu \mid \nu\nu)$ with $1/R$.

with the scaling factor κ

$$\kappa = e^{-\frac{\alpha\beta}{\alpha+\beta}|\mathbf{A}-\mathbf{B}|^2} \tag{3.3}$$

and the center-of-charge coordinate P

$$\mathbf{P} = \frac{\alpha\mathbf{A} + \beta\mathbf{B}}{\alpha + \beta} \tag{3.4}$$

Spatial integration yields the expression for the overlap between χ_A and χ_B

$$S_{AB} = \int \kappa\chi_P dr = \kappa \left(\frac{\pi}{\alpha+\beta}\right)^{3/2} \tag{3.5}$$

The magnitude of the overlap integral is proportional to the scaling factor κ which decays exponentially with the distance between GTO centers. In the case of the model system given above, where $\alpha = \beta$, the distance at which the integral falls below a certain threshold ϵ is given by

$$d_s = \sqrt{\alpha^{-1} ln\left[\left(\frac{\pi}{2\alpha}\right)^3 \epsilon^{-1/2}\right]} \tag{3.6}$$

Which in our case is equal to 6.9 a_0. Each hydrogen atom therefore only has significant overlap with a finite number n_{max} of other centers. For atom chains with $n > n_{max}$, the number of non-zero elements in the overlap matrix will no longer scale as n^2, but *linearly* with n_{max}. For more realistic, three-dimnensional molecular systems, the crossover is less clearly defined due to the non-uniform distribution of atoms and different GTO exponents. Nonetheless, if a system grows sufficiently large, the overlap integrals still scale linearly. Similar arguments can be brought forth for the kinetic-energy integrals as well.

Quadratic Scaling Electron Repulsion Integrals

Using the Gaussian product theorem established above, we can express the two-electron repulsion integrals of four primitive 1s Gaussian functions $s(A,\alpha)$, $s(B,\beta)$, $s(C,\gamma)$ and $s(D,\delta)$ as

$$\begin{aligned} g_{ABCD} &= \int s(A,\alpha)s(B,\beta)\frac{1}{|\mathbf{r_1}-\mathbf{r_2}|}s(C,\gamma)s(D,\delta)dr \\ &= \int \kappa s(P,\alpha+\beta)\frac{1}{|\mathbf{r_1}-\mathbf{r_2}|}\lambda s(Q,\gamma+\delta) \end{aligned} \tag{3.7}$$

where $s(P,p)$ and $s(Q,q)$ are Gaussian distributions with

$$\mathbf{P} = \frac{\alpha\mathbf{A}+\beta\mathbf{B}}{\alpha+\beta}; \quad \mathbf{Q} = \frac{\gamma\mathbf{C}+\delta\mathbf{D}}{\gamma+\delta} \tag{3.8}$$

$$\kappa = e^{-p|\mathbf{A}-\mathbf{B}|^2}; \quad \lambda = e^{-q|\mathbf{C}-\mathbf{D}|^2} \tag{3.9}$$

$$p = \frac{\alpha\beta}{\alpha+\beta}; \quad q = \frac{\gamma\delta}{\gamma+\delta} \tag{3.10}$$

The coulomb integrals can then be evaluated as

$$g_{ABCD} = \sqrt{\frac{4\eta}{\pi}}S_{AB}S_{CD}F_0\left(\eta\left|\mathbf{P}-\mathbf{Q}\right|^2\right) \tag{3.11}$$

with the Boys function F_0 and the reduced exponent η given by

$$\eta = \frac{pq}{p+q} \tag{3.12}$$

The Boys function is an important function appearing in many expressions for molecular integral evaluation. There are two expressions that bound the Boys function

$$\begin{aligned} F_n(x) &\leq \frac{1}{2n+1} \quad \text{for small } x \\ F_n(x) &\leq \frac{(2n-1)!!}{2^{n+1}}\sqrt{\frac{\pi}{x^{2n+1}}} \quad \text{for large } x \end{aligned} \tag{3.13}$$

Using the Boys function's upper bounds, we can derive an upper bound for the electron repulsion integrals of our model system

$$g_{ABCD} \leq min\left\{\sqrt{\frac{4\eta}{\pi}}S_{AB}S_{CD}, \frac{S_{AB}S_{CD}}{|\mathbf{P}-\mathbf{Q}|}\right\} \tag{3.14}$$

The left-hand upper bound represents the short-range limit of the Boys function, and the right-hand one the long-range limit. In the short-range limit, i.e. for increasing distance R_{AB} or R_{CD}, the magnitude of g decreases *exponentially*. As shown in the previous section, the non-zero elements of the overlap integrals S_{AB} and S_{CD} scale linearly with system size, and therefore the number of significant electron repulsion integrals scales with N^2 in total. It should be noted, that in the long-range limit with increasing

distance R_{PQ} between product densities, the number of elements in g will eventually scale linearly. However, the *algebraic* $1/R$ decay of the long-range interactions is so slow that it practically useless for the size of molecules that can be tackled with current technologies. In the case of the hydrogen atom chain, the integrals $(\mu\mu \mid \nu\nu)$ only fall below 1e-10 for R_{PQ} greater than 10^{10} a_0. While the long-range decay is impractical for use in the case of the electron repulsion integrals, there are instances such as in tomic orbital MP2 (see Chapter 4) where *bra* and *ket* decay as $1/R^4$. Knowing that the electron repulsion integrals are sparse is only the first step. One also has to develop a *screening* method to avoid computing small integrals, by finding a general upper bound. It has been shown [70] that g is positive-definite, and fulfills the relationship

$$\sum_{abcd} c_{ab} g_{abcd} c_{cd} > 0 \tag{3.15}$$

where c are one-electron orbital distributions. One can then apply the Schwarz inequality [71] to obtain an upper bound expression for g

$$(\mu\nu \mid \sigma\lambda) \leq (\mu\nu \mid \mu\nu)\,(\sigma\lambda \mid \sigma\lambda) = Q_{\mu\nu} Q_{\sigma\lambda} \tag{3.16}$$

The matrix $\mathbb{Q}$ contains the square root of the short-range diagonal entries of g, and is also known as the Schwarz matrix. $\mathbf{Q}$ can be evaluated quickly with $\mathcal{O}(N^2)$ effort and individual integrals can be efficiently screened. It should be noted that Schwarz screening does not take into account the $1/R$ decay between product densities, which makes the method less useful in methods like AO-MP2.

3.1.2 Element-Wise Sparsity of the Density Matrix

The decaying behavior of the density matrix has been extensively studied in solids for atom-centred Bloch and Wannier functions [72, 73, 74, 75, 76]. It was shown that for insulators, i.e. systems with large band-gaps, the contributions $P_{\mu\nu}$ decay exponentially with increasing distance $R_{\mu\nu}$, while for systems with small or no band gaps, such as metals, the elements decay algebraically. This is also known as Kohn's conjecture [72]. The same observations have been made for non-periodic systems using atomic orbitals as basis. For molecules with a large HOMO-LUMO gap, e.g. alkanes, the number of non-zero elements in the atomic orbital density matrix scales linearly with increasing system size. On the other hand, molecules with strong electron delocalization, such as conjugated polyenes, have have a small HOMO-LUMO gap, and the density matrix elements decay much slower. Consider again a chain of hydrogen atoms, equally spaced by a_0, each with one 1s Gaussian function, this time with N_{atom} atoms. Figure 3.2a shows the MO diagram for an increasing chain length. In the limit where $N_{atom} \to \infty$, the system takes on a band structure, similar to how they are encountered in a metal, with a smooth transition between occupied (valence) and virtual (conductance) band. In other words, the HOMO-LUMO gap becomes increasingly small. For a hydrogen chain, where each individual atom contributes one electron, the band is half filled, and the system is a conductor. If the Hydrogen atoms are replaced by Helium atoms, with two electrons per site, the band is fully filled and the system becomes an insulator. The magnitude of the density matrix elements $P_{\mu\nu}$ is plotted in Figure 3.3b as a function of increasing distance between 1s

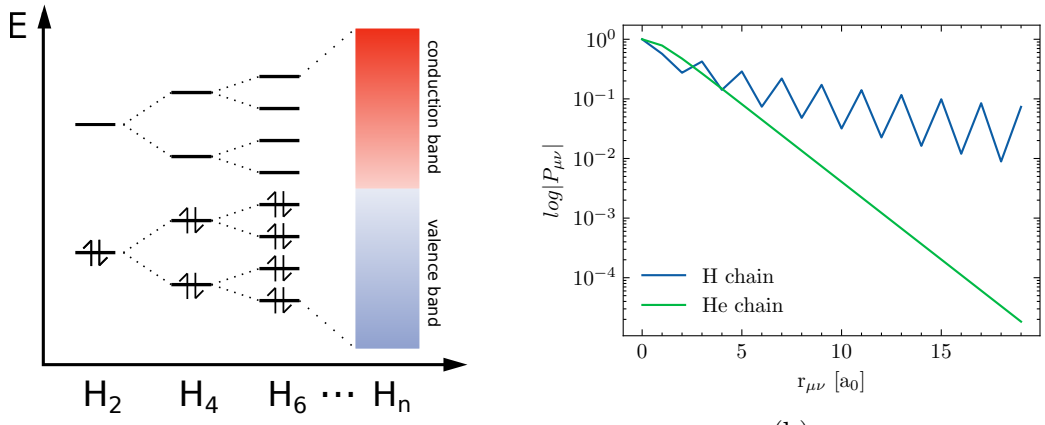


Figure 3.3: (a) Molecular orbital diagram for a hydrogen with increasing chain length. (b) Logarithm of the absolute value of the density matrix element $P_{\mu\nu}$ as a function of increasing distance $R_{\mu\nu}$ for a Hydrogen and a Helium atom chain.

functions. The elements decay much slower for the conducting hydrogen chain (algebraic decay), while a rapid exponential decay can be observed in the case of the insulating helium chain. An interesting thing to note is the oscillating values of the density matrix for the hydrogen chain. This phenomenon arises due to the hydrogen atoms pairing up into loosely bound H_2 molecules.

3.1.3 Diagrammatic Notation

Hollmann et al. [77] have introduced a simple graphical representation to show contributing factors to the sparsity of a given matrix, tensor or tensor contraction. Each tensor index is represented as a vertex. Non-connected vertices each contribute $\mathcal{O}(N)$ elements to the overall expression. A sparsity relationship between two indices is represented as an *edge* connecting two vertices. In that case, the number of *pairs* scales as $\mathcal{O}(N)$. Consider the two electron integral tensor $(\mu\nu \mid \sigma\lambda)$. From the previous section, we know that the index pairs μ, ν and λ, σ are related by overlap. The diagrammatic representation takes the form:

$$\mu \overset{\mathrm{S}}{\longleftrightarrow} \nu \qquad \sigma \overset{\mathrm{S}}{\longleftrightarrow} \lambda$$

There are two pairs of connected vertices, which indicates that the integrals can be evaluated with $\mathcal{O}(N^2)$ effort, which is in agreement with the findings above. The S denotes the overlap relationship between vertices. For another example, consider the Hartree-Fock expression for the exchange matrix

$$K_{\mu\nu} = (\mu\sigma \mid \nu\lambda)\, P_{\lambda\sigma} \tag{3.17}$$

Diagrammatically, the expression for $\mathbf{K}$ can be represented as

$$\mu \overset{\mathrm{S}}{\longleftrightarrow} \sigma \overset{\mathrm{P}}{\longleftrightarrow} \lambda \overset{\mathrm{S}}{\longleftrightarrow} \nu$$

The connection between σ and λ is also known as a "P-junction" [78], which represents the sparsity relationship arising due to the exponential decay of density matrix elements. The sparsity graph is fully connected, which suggests that $\mathbf{K}$ can be evaluated with $\mathcal{O}(N)$ effort. This is indeed the case, as shown by the ONX or LinK method. For linear scaling to emerge, indices of an expression therefore need to be fully linked. This simple but important fact is also known the linked index rule (LIR). Diagrams also show which factors can influence the performance of the scaling, such as diffuseness of the atomic orbitals (slower S decay) or size of of the HOMO-LUMO gap (slower P decay). One can therefore conclude that the expression for $\mathbf{K}$ as given above, is less suitable for large basis sets and conducting systems.

3.1.4 Rank Sparsity

A positive semi-definite matrix $\mathbf{A}$ has the property that it can be decomposed as a product

$$\mathbf{A} = \mathbf{B}\mathbf{B}^T \tag{3.18}$$

where $\mathbf{A}$ has dimensions N by N, and $\mathbf{B}$ has dimensions N by $rank(A)$. The rank represents the number of linearly independent column vectors in matrix, and for $rank(A) < N$, the matrix is said to be rank-deficient. The decomposition matrix $\mathbf{B}$ therefore is more compact and needs less storage space than $\mathbf{A}$. There are different ways to compute $\mathbf{B}$, such as Cholesky decomposition or QR decomposition. The tensor $(\mu\nu \mid \sigma\lambda)$ can be represented as a N_{AO}^2 by $N_{AO}^2 0$ matrix with combined row indices $I = \mu + N_{AO} * \nu$ and column indices $J = \sigma + N_{AO} * \lambda$. Because the tensor has been shown to be positive semi-definite, there also exists a decomposition, such that

$$(\mu\nu \mid \sigma\lambda) = A_{(\mu\nu)(\sigma\lambda)} = B_{(\mu\nu)X} B_{(\sigma\lambda)X} \tag{3.19}$$

The rank of $\mathbf{A}$ is in general much smaller than the combined index range N_{AO}^2, and scales linearly rather than quadratically with the number of basis sets. The decomposition tensor $\mathbf{B}$ is therefore 3-dimensional, rather than 4-dimensional, which reduces the storage needed by an order of magnitude from $\mathcal{O}(N^4)$ to $\mathcal{O}(N^3)$, ignoring sparsity. In the limit of large molecules, the NZEs of $\mathbf{B}$ also scale with $\mathcal{O}(N^2)$. Rather than for the molecular integrals in the AO basis, decomposition techniques are more useful for reducing the storage size of molecular integrals in the canonical MO basis

$$(ia \mid jb) = C_{\mu i} C_{\sigma a} B_{\mu\sigma X} B_{X\nu\lambda} C_{\nu j} C_{\lambda b} = B_{iaX} B_{Xjb} \tag{3.20}$$

The AO-MO transformation step is also drastically sped up, but remains a $\mathcal{O}(N^4)$ effort. Rank sparsity has therefore little impact on the overall scaling, but rather reduces the scaling *prefactor*. Over the years, different methods have been proposed to compute $\mathbf{C}$, such as density fitting, Cholesky decompostion, pseudo-spectral methods, or tensor hypercontraction. Density matrices at different levels of theory (Hartree Fock, MP2, CC ...) also exhibit rank sparsity. Decomposition of such matrices play an important role in local molecular orbital schemes and low scaling electronic structure methods, as will be shown in later sections.

3.2 Density Fitting

The method of choice in this thesis for the decomposition of two-electron molecular integrals is *density fitting* (DF), also known as *resolution of the identity* (RI). For a brief exploration of other popular methods, the reader is referred to Annex **??**.

3.2.1 Basics of Density Fitting

The two-electron integrals can be expressed in terms of the charge product densities $\rho_{\mu\nu} = \chi_\mu \chi_\nu$ as

$$(\mu\nu \mid \sigma\lambda) = \int\int \frac{\rho_{\mu\nu}(\mathbf{r_1})\rho_{\sigma\lambda}(\mathbf{r_2})}{\mathbf{r_1} - \mathbf{r_2}} d\mathbf{r_1} d\mathbf{r_2} \tag{3.21}$$

The charge densities ρ can be approximated by fitting them to a set of atom-centered auxiliary functions χ_P

$$\rho_{\mu\nu}(\mathbf{r}) = C_{P\mu\nu}\chi_P(\mathbf{r}) + \Delta\rho_{\mu\nu} \tag{3.22}$$

Or in the chemist's notation:

$$|\mu\nu) = C_{P\mu\nu}\,|P) + |\epsilon_{\mu\nu}) = |\widetilde{\mu\nu}) + |\epsilon_{\mu\nu}) \tag{3.23}$$

where $C_{P\mu\nu}$ are the fitting coefficients, and $\Delta\rho_{\mu\nu}$ or $|\epsilon_{\mu\nu})$ is the error introduced by the fitting procedure. Equation 3.23 is known as the density fitting approximation [79, 80, 81, 82]. The two-electron integrals then take the form

$$\begin{aligned}(\mu\nu \mid \sigma\lambda) &= (\widetilde{\mu\nu}|\widetilde{\sigma\lambda}) + \underbrace{(\widetilde{\mu\nu}|\epsilon_{\sigma\lambda}) + (\epsilon_{\mu\nu}|\widetilde{\sigma\lambda})}_{\text{first order}} + \underbrace{(\epsilon_{\mu\nu} \mid \epsilon_{\sigma\lambda})}_{\text{second order}} \\ &= (\widetilde{\mu\nu}|\widetilde{\sigma\lambda}) + \epsilon_J^{(1)} + \epsilon_J^{(2)}\end{aligned} \tag{3.24}$$

Here, $\epsilon_J^{(1)}$ and $\epsilon_J^{(2)}$ represent the first order (linear) and second order (quadratic) error. The fitting coefficients are then generally found by minimizing $\epsilon_J^{(2)}$. Substituting $(\epsilon_{\mu\nu}| = (\mu\nu - \widetilde{\mu\nu}|$ gives

$$\frac{\partial}{\partial C_{\mu\nu}^P}(\mu\nu - \widetilde{\mu\nu} \mid \sigma\lambda - \widetilde{\sigma\lambda}) = 0 \tag{3.25}$$

which then yields a set of linear equations

$$(\mu\nu \mid P) - \sum_Q C_{\mu\nu}^Q (Q \mid P) = 0 \tag{3.26}$$

Finding the fitting coefficients by minimizing $\epsilon_J^{(2)}$ has the important feature that $eps_J^{(1)} = 0$, which can be shown by substituting Equation 3.26 back into Equation 3.24. The total electron integral error is therefore *quadratic* in the fitting error. Fitting procedures where the coefficients $C_{\mu\nu}^P$ satisfy Equation 3.26 are termed *robust* [83]. Any restrictions posed on $C_{\mu\nu}^P$ makes ϵ_1 different from zero and the error scales linearly. Equation 3.26 requires the evaluation of the three-centre-two-electron (3c2e) and two-centre-two-electron (2c2e) integrals in the auxiliary basis set $\{P\}$

$$(\mu\nu \mid P) = \int\int \chi_\mu(\mathbf{r_1})\chi_\mu(\mathbf{r_1})\frac{1}{\mathbf{r_1} - \mathbf{r_2}}\chi_P(\mathbf{r_2}) d\mathbf{r_1} d\mathbf{r_2} \tag{3.27}$$

$$(P \mid Q) = \int\int \chi_P(\mathbf{r_1}) \frac{1}{\mathbf{r_1} - \mathbf{r_2}} \chi_Q(\mathbf{r_2}) d\mathbf{r_1} d\mathbf{r_2} \tag{3.28}$$

The fitting coefficients are generally computed by inverting $(P \mid Q)$, which leads to the following approximation for the four-center-two-electron integrals (4c2e)

$$(\mu\nu \mid \sigma\lambda) \approx (\mu\nu \mid P) (P \mid Q)^{-1} (Q \mid \sigma\lambda) \tag{3.29}$$

Matrix inversion is a $\mathcal{O}(N^3)$ computational effort. For more details on precision and best practices involving matrix inversion, see Annex **??**.

3.2.2 Scaling of the 3c2e Integrals

Using the diagrammatic representation introduced earlier, the 3c2e integral tensor reduces to

$$\mu \overset{\mathrm{S}}{\longleftrightarrow} \nu \qquad P$$

The number of non-zero elements therefore scales as $\mathcal{O}(N^2)$, just like for the 4c2e integrals. Similarly, the Schwarz inequality can be used to screen out small integrals

$$|(\mu\nu \mid P)| \leq |(\mu\nu \mid \mu\nu)|^{1/2} |(P \mid P)|^{1/2} \tag{3.30}$$

As mentioned above, Schwarz screening does not take into account increasing bra-ket distance. The long-range decay is too slow to be of any advantage in the case of the 4c2e integrals. However, it was found [84] that for an auxiliary density $\chi_P(\mathbf{r}$ with angular momentum l_P, the 3c2e integrals actually decay as $1/R^{-1-l_P}$ with increasing bra-ket distance, establishing a weak but not insignificant sparsity relationship between $(\mu\nu|$ and $|P)$

$$\mu \overset{\mathrm{S}}{\longleftrightarrow} \nu \qquad \overset{P}{\underset{1/R^{-1-l_P}}{\cdots\cdots}}$$

In principle, the 3c2e integrals can be evaluated with linear effort. Hollmann et al. [84] have introduced a tight upper bound, known as the SVQl estimator, to exploit this faster decay. Due to the dependence on l_P, the screening is most effective with larger basis sets with high angular momentum functions.

The fitting coefficients evaluated as $C^P_{\mu\nu} = (\mu\nu \mid Q) (Q \mid P)^{-1}$ formerly scale with $\mathcal{O}(N^3)$

$$\mu \overset{\mathrm{S}}{\longleftrightarrow} \nu \qquad Q \qquad P$$

due to the inverse of $(P \mid Q)$ not being sparse.

3.2.3 Local Density Fitting: Principles

The long-range behaviour introduced by Equation 3.26 is often deemed "unphysical" [85]. *Local density fitting* (LDF) methods circumvent this problem by forcing a more rapid decay of long-range contributions, either (a) by using a different metric in the fitting procedure or (b) by constructing domains $[\mu\nu]$ that exclude distant fitting functions P a priori. In both cases, Equation 3.26 no longer holds and the error in the electron integrals increases linearly with the fitting error. Therefore, the density fitting procedure is no longer robust. Fortunately, LDF methods can use a different expression for the electron integrals which includes the first order terms to remove the linear error

$$(\mu\nu \mid \sigma\lambda) \approx (\widetilde{\mu\nu}|\sigma\lambda) + (\mu\nu|\widetilde{\sigma\lambda}) - (\widetilde{\mu\nu}|\widetilde{\sigma\lambda}) \tag{3.31}$$

which is known as Dunlap's robust density fitting formula **??**. It greatly increases accuracy for LDF.

3.2.4 LDF (I): Short-Range Metrics

The first type of LDF methods replaces the fitting procedure in the Coulomb metric in Equation 3.26 by a more general expression

$$B^P_{\mu\nu} - C^Q_{\mu\nu} M_{QP} = 0 \tag{3.32}$$

where $B^P_{\mu\nu}$ and M_{PQ} are the 3c2e and 2c2e integrals given by

$$B^P_{\mu\nu} = \int\int \chi_\mu(\mathbf{r_1})\chi_\nu(\mathbf{r_1}) g(\mathbf{r_1},\mathbf{r_2}) \chi_P(\mathbf{r_2}) d\mathbf{r_1} d\mathbf{r_2} \tag{3.33}$$

$$M_{PQ} = \int\int \chi_P(\mathbf{r_1}) g(\mathbf{r_1},\mathbf{r_2}) \chi_Q(\mathbf{r_2}) d\mathbf{r_1} d\mathbf{r_2} \tag{3.34}$$

with g being the operator for the local metric. A list of known local metrics is given in Table 3.1. Earliest forms of density fitting actually first used an overlap metric to directly minimize the norm of the residual $R_{\mu\nu} = (\mu\nu| - (\widetilde{\mu\nu}|$ by the linear least squares methods [80], and the fitting coefficients are computed as

$$C^P_{\mu\nu} = S^{-1}_{PQ}(\mu\nu Q) \tag{3.35}$$

where S_{PQ} is the overlap matrix of the auxiliary basis, and $(\mu\nu Q)$ are the 3-center-**1-electron** overlap integrals. While the overlap metric has the most rapid decay and the quantities in Equation 3.35 can be evaluated in $\mathcal{O}(N)$ time, it has the worst accuracy of all metrics. One solution to this problem is to introduce a metric which is intermediate between overlap and coulomb fitting. Examples include the Yukawa, Coulomb- and Gaussian-attenuated metrics. These intermediate metrics introduce a damping factor ω to control the sparsity and accuracy of the density fit. In the limit where $\omega \to 0$, and $\omega \to \infty$, one recovers the coulomb and overlap metric, respectively. Figure 3.4 shows the decay behaviour of the Coulomb-attenuated metric, for $\omega = 0.01$, 0.1 and 1.0, compared to the overlap and the Coulomb metric.

Metric	$g(r_{12})$
Overlap [80]	1
Coulomb-Attenuated [86]	$\frac{erfc(\omega r_{12})}{r_{12}}$
Yukawa [87]	$\frac{e^{-\omega r_{12}}}{r_{12}}$
Gaussian-Damped [88]	$\frac{e^{-\omega r_{12}^2}}{r_{12}}$

Table 3.1: Expressions for the operator g in different local metrics.

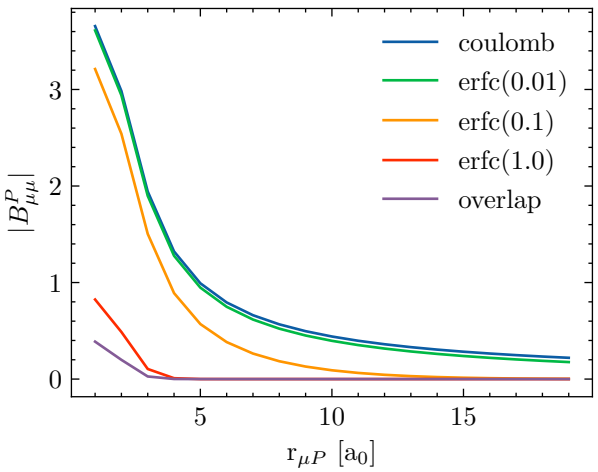


Figure 3.4: Absolute value of the 3c2e integral $B_{\mu\mu P}$ between two 1s GTOs μ and P with $\alpha = 1.0$ using different metrics.

3.2.5 LDF (II): Local Domains

The second method to force locality in density fitting consists in constructing local domains for each atom, pair of atoms or local molecular orbital, and excluding any auxiliary functions that lie outside, which can drastically reduce the dimension of the fitting procedure.

Atomic Resolution of the Identity

The simplest example of a domain is one that includes a single atom. The atomic resolution of the identity (ARI) [89] uses a fitting procedure where the sum over auxiliary function Q only includes those which are centered on the same atom A as the atomic orbital μ

$$|\widetilde{\mu\nu}) = \sum_{Q \cup A_\mu} (P \mid Q)^{-1}_{A_\mu} (\mu\nu \mid Q) \tag{3.36}$$

Each atom X has its own metric matrix inverse $(P \mid Q)^{-1}_X$ which takes the form

$$(P \mid Q)^{-1}_X = B_X \left((P \mid Q)_D + B_X (P \mid Q)_{OD} B_X \right)^{-1} B_X \tag{3.37}$$

where $(P \mid Q)_D$ and $(P \mid Q)_{OD}$ are the diagonal and off-diagonal part of $(P \mid Q)$ respectively. B_X is a so-called *bump matrix* which imposes a fast, but smooth decay between functions P and Q in order to avoid using all functions P for the fitting. For further details, the reader is referred to the original publication. The bump matrix uses multiple distance-dependent criteria which make the ARI less of a black-box method.

Pair-Atomic Resolution of the Identity

A more popular and simple variant of atomic density fitting is the pair-atomic resolution of the identity (PARI) method [90]. As the name implies, the domains include atom *pairs* rather than a single atom. Again expressing it in terms of the fitting procedure

$$(P \mid \mu\nu) = \sum_{Q \in A \cup B} (P \mid Q) C^Q_{\mu\nu} \quad \forall P \in A \cup B \tag{3.38}$$

The number of linear equations is equal to the number of non-zero pairs $\mu\nu$, which scales linearly. However, the PARI approach enforces heavy constraints on the fitting coefficients, which leads to large integral errors. Merlot et al. proposed to increase the atomic pair domain with any atoms which lie between A and B. Alternatively, larger and more diffuse basis sets can be used. In both cases, performance is sacrificed for increased accuracy. The absence of any distance dependent parameters or thresholds still make it an attractive method both for Hartree Fock and Post-Hartree Fock methods [91, 92].

LDF using Local Molecular Orbitals

Finally, domains can also be formed using local molecular orbitals instead of AOs. LMOs are larger than AOs, but are still generally centered on only a few atoms. The exact atomic sites can be determined for example by using a Mulliken population analysis.

Consider the density fitting procedure as proposed by Polly et al. for their LDF-Hartree Fock method **??**

$$(\mu i \mid P) = \sum_{Q \in [i]_{fit}} (P \mid Q)\, C^Q_{\mu i} \tag{3.39}$$

The fitting coefficients are determined individually for each AO-LMO pair $|\mu i)$, and include only those auxiliary functions centred on atoms in the fitting domain $[i]_{fit}$ for which the Mulliken charges are above a given threshold. Although the fitting coefficients need to be recomputed for each update of the MO coefficients, the number of $|\mu i)$ pairs scales linearly with system size. This type of local density fitting and variations thereof are predominantly used in pair-orbital specific local correlation methods, and will be explained in more detail further below.

3.2.6 LDF (III): Quasi-Robust Density Fitting

Local density fitting imposes constraints on the fitting procedure, and the integral error consequently scales linearly with the fitting error. Using Dunlap's robust formula is deemed necessary in most cases to achieve acceptable accuracy, but reintroduces the slowly decaying 3c2e integrals. Furthermore, replacing the 4c2e integrals by Equation 3.31 greatly increases the complexity of expressions in electronic structure theory, which is still manageable for ground state methods, but quickly becomes cumbersome for excited states.

Quasi-robust density fitting (QRDF) [85] aims to combine the exponential decay behavior of LDF with an accuracy comparable to standard density fitting, without the use of Dunlap's formula. Again, consider the fitting procedure

$$\sum_Q (P \mid Q)\, C^Q_{\mu\nu} = (P \mid \mu\nu) \tag{3.40}$$

The sets of auxiliary functions $\{P\}$ and $\{Q\}$ have different roles. The functions Q fit the charge density $|\mu\nu)$, while the P functions act as *test functions* where the electron integrals should be accurate, i.e. where $(X \mid \widetilde{\mu\nu}) \approx (X \mid \mu\nu)$. For two functions μ and ν not located on the same atom, their charge density $|\mu\nu)$ lies in the vacuum between them, and the atom-centered auxiliary functions may be ill-suited to fit $|\mu\nu)$. For this reason, the fitting procedure draws from all fitting functions $\{P\}$ spanning the whole molecule to cancel out the linear error, which in consequence introduces long-range contributions in $C^P_{\mu\nu}$ in the coulomb metric, even if $|P)$ is not close to $|\mu\nu)$. The basic idea of QRDF is to only chose fitting functions $\{P\}$ close to $|\mu\nu)$ via overlap criteria, but still perform the fitting procedure in the coulomb metric.

The QRDF Fitting Procedure

For a set of given μ, ν, select a set of *fitting function* $\{P_{\mu\nu}\} \in \{P\}$ close to $|\mu\nu)$ according to the criteria

$$\left| \sum_R S^{-1}_{PR} (R\mu\nu) \right| > T \tag{3.41}$$

where S is the auxiliary overlap matrix, and $(R\mu\nu)$ are the 3-centre-1-electron overlap integrals. Next, choose a set of test functions $\{Q_{\mu\nu}\} \in \{P\}$ using

$$f(Q_{\mu\nu}, P_{\mu\nu}) < R \tag{3.42}$$

with

$$f(A, B) = \frac{\alpha\beta}{\alpha + \beta} |\mathbf{A} - \mathbf{B}|^2 \tag{3.43}$$

where for two auxiliary functions A and B, the values α, β are their smallest primitive exponents and $\mathbf{A}$, $\mathbf{B}$ are their respective positions. The fitting coefficients are then determined via

$$\sum_P (Q_{\mu\nu} \mid P_{\mu\nu}) C^P_{\mu\nu} = (Q_{\mu\nu} \mid \mu\nu) \tag{3.44}$$

where the fitting coefficients are accurate within the set of test functions $\{Q_{\mu\nu}\}$. The linear equations in Equation 3.44 can be solved via QR decomposition or singular value decomposition (SVD) of the rectangular matrix $(Q_{\mu\nu} \mid P_{\mu\nu})$. The QRDF scheme depends on two parameters, T and R. In the limit where $T \to 0$ and $R \to \infty$, the standard fitting procedure in the coulomb metric is recovered.

The fitting functions $\{P_{\mu\nu}\}$ are selected via overlap criteria and therefore scale linearly with the number of pairs $|\mu\nu)$, and consequently the same holds true for the number of test functions $\{Q_{\mu\nu}\}$ close to $\{P_{\mu\nu}\}$ chosen by Equation 3.41. In the limit of large molecules, the size of the rectangular matrix in Equation 3.42 becomes constant and the fitting procedure can be evaluated with $\mathcal{O}(N)$ effort. However, a QR decomposition needs to be computed for each set of $|\mu\nu)$, leading to relatively high prefactor which makes the method not competitive for dense 3D structures like water clusters, as will be discussed in the results section. The QRDF method has been shown to deliver accuracies comparable to standard density fitting, without the use of Dunlap's formula, making it a very attractive alternative to other LDF schemes, especially if one wishes to reduce the complexity of expressions involving LDF.

3.2.7 Auxiliary Basis Sets

The density fitting approximation does not make any assumptions about the size or shape of the auxiliary basis set used. In principle, the fit is exact for the basis set containing all N^2_{AO} Gaussian products $\chi_P = \chi_\mu \chi_\nu$. In practice, the product space is over-complete and can be represented by much smaller basis sets. Accurate results can be obtained for auxiliary basis sets which are about 4 times larger than the principal basis set they are used with.

Auxiliary basis sets generally need more higher angular momentum functions than standard basis sets. Consider an isolated, unperturbed atom, with electrons occupying atomic orbitals with highest angular momentum l_{occ}. A minimal basis set for this atom contains functions of angular momenta 0 to l_{occ}. However, a minimal auxiliary basis set for fitting the product space $\chi_\mu^{0\ldots l_{occ}} \chi_\nu^{0\ldots l_{occ}}$ needs functions with maximum angular momentum $2l_{occ}$. For example, 2nd row elements ($l_{occ} = 1$) need an auxiliary basis set containing d-functions, and first row transition metals ($l_{occ} = 2$) even need g-functions. Similarly to standard basis sets, to describe atoms in molecules where the orbitals are subject to polarization effects, even higher angular momentum functions are needed

to fit polarization functions. In practice, a principal basis set with maximum angular momentum l_{bas} is paired with an auxiliary basis set with highest angular momentum $l_{bas} + l_{occ}$.

Auxiliary basis sets have the drawback of being method-specific. There are two categories: auxiliary basis sets for density fitted Hartree-Fock (DF-HF) and for density fitted correlated methods (e.g. DF-MP2, DF-CCSD, DF-ADC(2)). Auxiliary basis sets for DF-HF not only need to reproduce Hartree Fock energies, but also need to minimize negative impact on post-Hartree methods. An ill-suited auxiliary basis set leads to a deterioration of the virtual orbital space, and hence an increased error for correlated methods.

Optimization procedures often try to minimize the energy differences between the standard method and its density fitting approximation in a series of atomic calculations. For example, the jkfit family of basis sets (cc-pVXZ-JKFIT [93], def-XVP-JKFIT [94]) minimize the error

$$\Delta E_{HF} = E_{HF} - E_{DF-HF} \tag{3.45}$$

The RI basis set family (cc-pVXZ-RIFIT [95], def2-XVP-RIFIT [96]) minimizes the same energy difference but for MP2 or Coupled Cluster.

Another disadvantage of auxiliary basis sets is that the accuracy of the fitting procedure cannot be easily controlled as a function of its composition (number of functions, angular momenta...), but rather extensive benchmarks are needed for each basis set that is introduced. An alternative approach was proposed by Aquilante et al. [97] where the fitting basis sets are generated automatically by Cholesky decomposition of the atomic 2-electron integrals

$$(\mu\nu \mid \sigma\lambda) = L^X_{\mu\nu} L^X_{\sigma\lambda} \tag{3.46}$$

The Choleksy vectors $\mathbf{L}_{\mu\nu}$ indicate which product densities should be taken to construct the auxiliary basis. This type of atomic Cholesky decomposition (aCD) basis sets has the advantage that the accuracy can be rigorously controlled by the decomposition threshold θ. To remove linear dependencies in the aCD basis set, another Cholesky decomposition can be performed to yield the atomic compact Cholesky decomposition (acCD) auxiliary basis set [98].

3.3 Multipole Expansion of the Electron Integrals

The slow $1/R$ between the product densities $\Omega_{\mu\nu}$ and $\Omega_{\lambda\sigma}$ in the coulomb integrals is a major obstacle for achieving linear scaling in cases where no other sparsity relationship can be established between indices belonging to separate charge densities, e.g. in the evaluation of the coulomb matrix $\mathbf{J}$ versus the exchange matrix $\mathbf{K}$. Luckily, there are approximate methods for integral evaluation that can be computed with $\mathcal{O}(N)$ effort, known as *multipole methods*.

3.3.1 Classical and Non-Classical Electron Integrals

First, consider the concept of classical and non-classical interactions. Two electron integrals are said to be *non-classical* if the two charge densities $\Omega_{\mu\nu}$ and $\Omega_{\sigma\lambda}$ overlap, and *classical* if the charge densities are well separated. In the latter case, the electron integrals

represent classical interactions between disjoint point charges, and can be approximated using multipole methods, whereas the non-classical contributions must be evaluated using the more expensive standard integral codes such as McMurchie-Davidson or Obara-Saika.

Two Gaussian distributions Ω_P and Ω_Q are considered *well-separated* up to a target accuracy 10^{-k}, if their center-to-center distance R_{PQ} is larger than the sum of their extents ext_P and ext_Q:

$$R_{PQ} > ext_P + ext_Q \tag{3.47}$$

with the extent of a Gaussian product P defined as

$$r_P = \frac{1}{\sqrt{p}} erfc^{-1}(10^{-k}) \tag{3.48}$$

where p is the reduced exponent. Another important thing to note is that the number of significant non-classical and classical integrals scale as $\mathcal{O}(N)$ and $\mathcal{O}(N^2)$ respectively [9], which has important consequences as will be shown further below.

3.3.2 Multipole Expansion

For two well-separated charge distributions P and Q, the inverse inter-electronic distance can be expanded in terms of Legendre polynomials $\mathcal{P}$ as

$$\frac{1}{r_{12}} = \sum_{l=0}^{\infty} \frac{\Delta r_{12}^l}{R_{PQ}^l} \mathcal{P}_l cos\theta \tag{3.49}$$

with

$$cos\theta = \frac{\Delta \mathbf{r}_{12}^l \mathcal{R}_{QP}}{\Delta r_{12} R_{QP}} \tag{3.50}$$

$$\Delta \mathbf{r}_{12} = \mathbf{r}_{1P} \mathbf{r}_{2Q} \tag{3.51}$$

where $\mathbf{r}_{1P}$ and $\mathbf{r}_{2Q}$ are the distance between electron 1,2 and the centers P,Q. Equation 3.49 is also known as the partial-wave expansion of the coulomb operator [99]. Plugging Equation 3.49 into the expression for the two-electron repulsion integrals gives the bipolar multipole expansion of the two-electron integrals

$$g_{abcd} = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \sum_{j=0}^{\infty} \sum_{k=-j}^{j} M_{ab}^{lm}(P) T_{lm,jk} M_{cd}^{jk} \tag{3.52}$$

where $\mathbf{M}_{ab}^{lm}(P)$ is the multipole moment of the charge distribution P with total moment $l + k$, and $\mathbf{T}$ is the so-called interaction matrix. As such, the complicated 6-dimensional evaluation of g can be simply substituted by two 3-dimensional integrations of the multipole moments $\mathbf{M}$ at a much lower cost. For the lowest order expansion, where $l = m = 0$, and $j = k = 0$, the multipole moments and the interaction matrix become

$$M_{ab}^{00} = S_{ab} \tag{3.53}$$

$$M_{cd}^{00} = S_{cd} \tag{3.54}$$

$$T_{00,00} = 1/R_{PQ} \tag{3.55}$$

The zero order term of the multipole expansion therefore takes the form

$$g_{abcd} \approx \frac{S_{ab} S_{cd}}{R_{PQ}} \tag{3.56}$$

3.3.3 Fast Multipole Method

While the number of individual non-zero integrals still scales with $\mathcal{O}(N^2)$, the total contribution of all pair-wise interactions to the total energy (Hartree Fock, MP2 ...), can actually be evaluated in $\mathcal{O}(N)$.

For the sake of simplicity, consider a system with point-charge particles with charge Z, in a 2-dimensional plane. The total interaction energy is given by

$$U = \sum_{i>j} \frac{Z_i Z_j}{r_{ij}} \tag{3.57}$$

Evaluating Equation 3.57 as is takes a quadratic effort. In a first approximation, one can divide the plane into a grid of blocks of equal size, where each block contains a certain number of particles (Figure 3.5). Consider the interaction of a single particle i in its source block C with the other particles in the system. The interaction has two contributions: near-field (NF) contributions U_{NF} from the other particles in the source block, and the blocks immediately surrounding it, and far-field (FF) contributions U_{FF} from boxes that are well-separated from C. The NF interactions are evaluated directly by summing over all particles j in the near-field

$$U_i^{NF} = \sum_{j \in NF} \frac{Z_i Z_j}{R_{ij}} \tag{3.58}$$

while FF interactions are computed using multipole expansions $\mathbf{q}_{iC}$ and $\mathbf{q}_A$ of the FF boxes and the particle i

$$U_i^{FF} = \sum_{A \in FF} \mathbf{q}_{iC} \mathbf{T}_{CA} \mathbf{q}_A \tag{3.59}$$

While evaluating the interaction energy at block-level rather than particle-level can considerably reduce the prefactor, the cost of this *single-level multipole method* is still quadratic, since for each particle i, there is a system-dependent number of FF boxes. The granularity of the blocks is the same, independent of how far away the blocks are. To achieve linear scaling, the crucial point to realize is that, the further one gets from the source block C, the smaller the single-particle interaction U_i becomes, and the less accurately it actually needs to be evaluated. This means that the farther one moves away from C, the larger the FF boxes can be. For this reason, *multi-level multipole methods* introduce a hierarchy of boxes (Figure **??**), where at level 0, the whole system is in a single box, and for each subsequent level, the field is divided into fourths. FF boxes that are closest to C are evaluated at the highest level/granularity S. The region of FF boxes sourrounding the closest FF boxes are then treated at a lower level $S-1$, and so on, until all interactions have been computed. Because the multipole expansion is evaluated for increasing box size, it can be shown that the total number of boxes is constant for a single particle i. This is the basic idea on which the *Fast Multipole Method* (FMM) operates [100, 101, 102], and it has quickly become one of the most important algorithms in scientific computing, as the problem of particle-particle interaction is not limited to the field of quantum chemistry. FMM can evaluate the total interaction energy U with linear computational complexity.

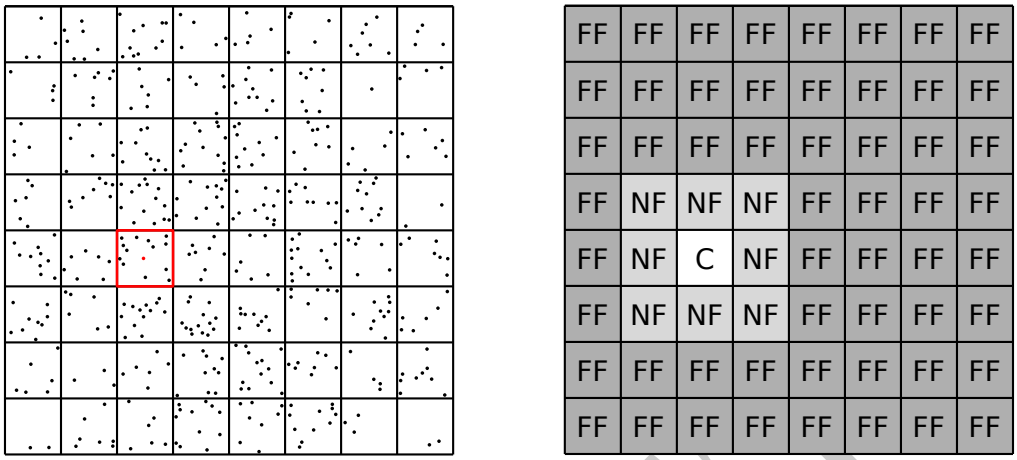


Figure 3.5: In multipole methods, the system is subdivided into blocks of equal size containing one or more particles. For a reference block C, its surrounding blocks are categorized into near-field and far-field contributions which are treated using separate methods.

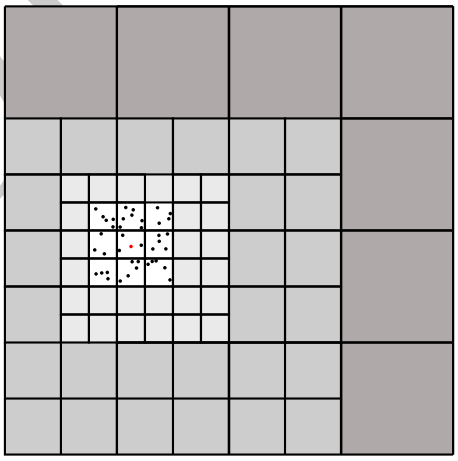


Figure 3.6: In the fast multiple method, the granularity of the boxes becomes coarser the further away they are from the source blocks.

3.3.4 Continuous Fast Multipole Method

The fast multipole method does not work for continuous charge distrbutions like Gaussian functions, as their extents can be quite different from one another, making the separation into NF and FF contributions more difficult. Nonetheless, FMM has been generalized to the continous case, known as the continuous fast multipole method (CFMM) [103]. The principle is the same as in multi-level multipole methods, only special care needs to be taken to only include classical contributions into the FMM treatment. For further details, the reader is referred to the original publication.

3.4 The ABCs of LMOs and NOs: Orbital Representations

To solve the Hartree-Fock equations in a computationally efficient way, it is favorable to choose the set of molecular orbitals such that they diagonalize the Fock matrix $\mathbf{F}$. In other words, they are eigenfunctions of the Fock operator

$$\hat{f}\,|\phi_i\rangle = \epsilon_i\,|\phi_i\rangle \tag{3.60}$$

Here, $\{\phi_i\}$ are also known as the *canonical molecular orbitals*. CMOs are not unique in the sense that there are infinitely many alternative molecular representations which yield the same electron density $\mathbf{P}$. Quantities like the total wave function energy or the electronic density are said to be *orbitally invariant*. Non-observables like the MO energies are not preserved under orbital rotation. Let $\mathbf{C}$ be the CMO coeffcient matrix. A new solution to the HF equations can then be generated by applying a unitary transformation such that

$$C^{new}_{\mu\tilde{j}} = U_{ji}C_{\mu i} \qquad \mathbf{U}\mathbf{U}^{\dagger} = \mathbf{1} \tag{3.61}$$

There are different reasons why one would want to use another MO basis: they can (a) offer a more intuitive picture for the interpretation of chemical phenomena and (b) help to achieve are more localized and compact representation of the wave function which is helpful for local correlation methods.

There are two significant types of molecular representations besides CMOs: local molecular orbitals (LMOs) and natural orbitals (NOs). The following sections will introduce both types in more detail.

3.4.1 Local Molecular Orbitals

In contrast to canonical molecular orbitals, which are generally delocalized over the whole molecule, local molecular orbitals are confined to a relatively small volume and span only a few atoms (except in large conjugated systems). LMOs are well suited for a qualitative description of chemical reaction in terms of molecular bonds, lone pairs and π systems [104]. Moreover, they are often used in local correlation methods due to their reduced orbital span. There are several different ways for generating LMOs.

LMOs by Reducing a Functional

One of the most popular methods for finding LMOs consists in maximizing a localization function $\eta(\phi)$ by successive rotation of the orbital space. The most prominent examples are Foster-Boys (FB) [105], Edmiston-Ruedenberg (ER) *Edm1963* and Pipek-Mezey (PM) [106]. Their functionals can be written as

$$\zeta_{FB}(\chi) = \sum_i \langle \chi_i | \, \mathbf{r} \, | \chi i \rangle^2 \tag{3.62}$$

$$\zeta_{ER}(\chi) = \sum_i (\chi_i \chi_i \mid \chi_i \chi_i) \tag{3.63}$$

$$\zeta_{FB}(\chi) = \sum_i \sum_A \langle \chi_i | \, \mathbf{P}_A \, | \chi i \rangle^2 \tag{3.64}$$

The problem is generally solved using an iterative procedure consisting in consecutive pair-wise rotations, known as Jacobi sweeps (Section **??**). These sweeps are repeated until convergence is reached, which may be slow. The methods differ within the procedure by how the rotational angle is computed, and scale differently with system size, with $\mathcal{O}(N^3)$ for FB, $\mathcal{O}(N^5)$ for ER and $\mathcal{O}(N^4)$ for PM. A faster alternative to Jacobi sweeps does also exist [107].

Over the years, PM has been the more popular choice of the three: like ER and unlike FB, it conserves σ-π separation [108], but scales more favorably than ER.

Functional localization methods are most often used for rotating occupied MOs. Virtual MOs are often plagued by convergence issues and have a steep computational cost simply due to being much more numerous than occupied MOs [109]. It is crucial that molecular localization should not take longer than the methods they are used for, and hence VMOs are often localized using separate methods (e.g. PAO).

Projected Atomic Orbitals

A set of highly localized molecular orbitals can be obtained by projecting the CMOs onto the atomic orbital basis, known as projected atomic orbitals (PAO) [110]. For a set of orthonormal occupied and virtual molecular orbitals $\{\Psi_i\}$ and $\{\Psi_a\}$, the projection operators $\hat{P}$ and $\hat{Q}$ are defined as [111]

$$\hat{P} \;\; = |\Psi_i\rangle \langle \Psi_i| \;\; = |\chi_\mu\rangle \, C_{\mu i} C_{\nu i} \, \langle \chi_\nu| \tag{3.65}$$

$$\hat{Q} \;\; = |\Psi_a\rangle \langle \Psi_a| \;\; = |\chi_\mu\rangle \, C_{\mu a} C_{\nu a} \, \langle \chi_\nu| \tag{3.66}$$

which are then applied to the atomic orbitals χ

$$\hat{P} \, |\chi_{\mu'}\rangle \;\; = \textstyle\sum_\mu P_{\mu\nu} S_{\nu\mu'} \, |\chi_{\mu'}\rangle \;\; = \displaystyle\sum_\mu \overline{P}_{\mu\mu'} \, |\chi_{\mu'}\rangle = \left|\chi_{\underline{\mu}}\right\rangle \tag{3.67}$$

$$\hat{Q} \, |\chi_{\mu'}\rangle \;\; = \textstyle\sum_\mu Q_{\mu\nu} S_{\nu\mu'} \, |\chi_{\mu'}\rangle \;\; = \overline{Q}_{\mu\mu'} \, |\chi_{\mu'}\rangle = |\chi_{\overline{\mu}}\rangle \tag{3.68}$$

The projection operators $\hat{P}$, $\hat{Q}$ and the non-symmetric PAO coefficient matrices $\overline{\mathbf{P}}$, $\overline{\mathbf{Q}}$ are *idempotent*

$$\overline{\mathbf{PP}} = \overline{\mathbf{P}} \qquad \overline{\mathbf{QQ}} = \overline{\mathbf{Q}} \tag{3.69}$$

and *mutually orthogonal*

$$\overline{\mathbf{PQ}} = \mathbf{0} \qquad \overline{\mathbf{P}} + \overline{\mathbf{Q}} = \mathbf{1} \tag{3.70}$$

But not orthogonal within themselves

$$\langle \chi_{\underline{\mu}} | \chi_{\underline{\nu}} \rangle = S^{PAO}_{\underline{\mu\nu}} \qquad \langle \chi_{\overline{\mu}} | \, | \chi_{\overline{\nu}} \rangle = S^{PAO}_{\overline{\mu\nu}} \tag{3.71}$$

Here, the indices $\underline{\mu}, ulgn, ...$ and $\overline{\mu}, \overline{\nu}, ...$ are used for occupied and virtual projected atomic orbitals, respectively. The number of PAOs (occupied or virtual) is equal to the number of AOs, and are therefore linearly dependent (redundant). CMOs are transformed to PAOs by using

$$|\chi_{\underline{\mu}}\rangle \quad = (\mathbf{SC})_{\mu i} \, |\Psi_i\rangle \quad = \overline{C}_{\mu i} \, |Psi_i\rangle \tag{3.72}$$

$$|\chi_{\overline{\mu}}\rangle \quad = (\mathbf{SC})_{\mu a} \, |\Psi_a\rangle \quad = \overline{C}_{\mu a} \, |\Psi_a\rangle \tag{3.73}$$

The back-transformation is defined as

$$|\Psi_i\rangle \quad = C_{\mu i} \, |\underline{\mu}\rangle \tag{3.74}$$

$$|\Psi_a\rangle \quad = C_{\mu a} \, |\overline{\nu}\rangle \tag{3.75}$$

PAOs are centred on the atom on which their corresponding AO is localized, but can still be delocalized over multiple atoms, depending on the sparsity of the density matrix. Methods which are entirely formulated in PAOs are rare but possible [111]. The projection method is most often used on the virtual orbital space, where standrad localization procedures fail. In literature, the following alternative formular is often used for expression the virtual PAO coefficient matrix in terms of the occupied LMOs:

$$\overline{\mathbf{Q}} = \left(\mathbf{1} - \mathbf{L}\mathbf{L}^{\dagger}\mathbf{S}\right) \mathbf{C} \tag{3.76}$$

where $\mathbf{L}$ is the coefficient matrix of the occupied LMOs.

Cholesky Molecular Orbitals

Sparsity of the atomic density matrix is crucial for achieving low-scaling electronic structure methods. Aquilante et al. proposed [108] to define a set of occupied molecular orbitals by Cholesky decomposition of the density matrix. Analysis of the resulting Cholesky molecular orbitals (CholMOs) showed their localized character inherited from the sparsity of the density matrix.

$$\mathbf{P} = \mathbf{L}\mathbf{L}^{\mathbf{T}} \tag{3.77}$$

Figure 3.7 shows the sparsity of the occupied density matrix and the occupied Cholesky molecular coefficient matrix of the linear alkane $H_{322}C_{160}$. The number of CholMOs is equal to the rank of the density matrix, which is equal to the number of occupied orbitals. The CholMOs are computed by an incomplete Cholesky decomposition with full row and column pivoting (Section **??**. The unitary transformation matrix is given by

$$U_{i\underline{i}} = C_{\mu i} S_{\mu\nu} L_{\nu\underline{i}} \tag{3.78}$$

The decomposition algorithm scales with $\mathcal{O}(N^3)$ but can be made linearly scaling by using sparse matrix algebra. CholMOs have several advantages: the Cholesky decomposition is fast and non-iterative, and an initial guess for molecular orbitals is not needed.

The scheme can be extended to virtual orbitals as well, by CD of the virtual atomic density matrix $\mathbf{Q}$. The rank of $\mathbf{Q}$ is equal to the number of virtual orbitals $n_v ir$, therefore the prefactor of the incomplete CD increases with basis set size. Especially in the presence of diffuse functions, the rank reduction might not offer much of an advantage compared to simpler localization methods such as PAOs.

Moreover, orbitals obtained by CD are less localized than FB or ER LMOs, especially for small molecules. Low scaling is still possible using CholMOs in the context of LMO correlation methods, albeit with a larger prefactor.

CD is also used in the context of AO-MP2 to reduce the prefactor of integral transformation by using the rank sparsity of the pseudo-density matrices, as will be shown further below. CholMOs can also used as an initial guess for iterative localization schemes to achive faster convergence.

3.4.2 Natural Orbitals

While the schemes described above try to generate a set of occupied and/or virtual molecular orbitals localized in space, natural orbital (NOs) methods try to generate a set of "compact" orbitals, i.e. a minimal set of orbitals that can describe the problem at hand. The concept of natural orbitals was first introduced by Löwdin [112]. The natural orbitals Θ_i of a wave function Ψ are defined as the eigenfunctions of the one-particle density operator $\hat{n}$

$$\hat{n} \left|\Theta_i\right\rangle = n_i \left|\Theta_i\right\rangle \tag{3.79}$$

where n_i are the occupation numbers of the associated orbital Θ_i. One can then choose a reduced orbital space $\{\tilde{\Psi}_i\}$ by only taking into account those orbitals with an occupation number above a certain threshold τ. The orbitals are "natural" in the sense that they are determined purely using Ψ, and are intrinsic to the system. NOs are computed by diagonalizing the one-particle density matrix at the desired level of theory (Hartree-Fock, MP, CIS, CC).

Natural Orbitals in Hartree Fock Theory

In Hartree Fock theory, natural orbitals are mostly reserved for qualitative population and bond order analysis.

Natural atomic orbitals (NAOs) are computed by diagonalizing the blocks $P_{\mu_A \nu_A}$ of the atomic density matrix, where μ_A, ν_A are basis functions centred on atom A. NAOs are optimal for describing the electron density around individual atom centres. NAOs are also useful for obtaining a set of guess orbitals from densitiy matrices formed from the superposition of atomic densities (SAD) guess (Annex **??**).

Furthermore, NAOs serve as the starting point for obtaining natural hybrid orbitals (NHOs), which in turn are used for constructing natural bond orbitals (NBOs). NBOs

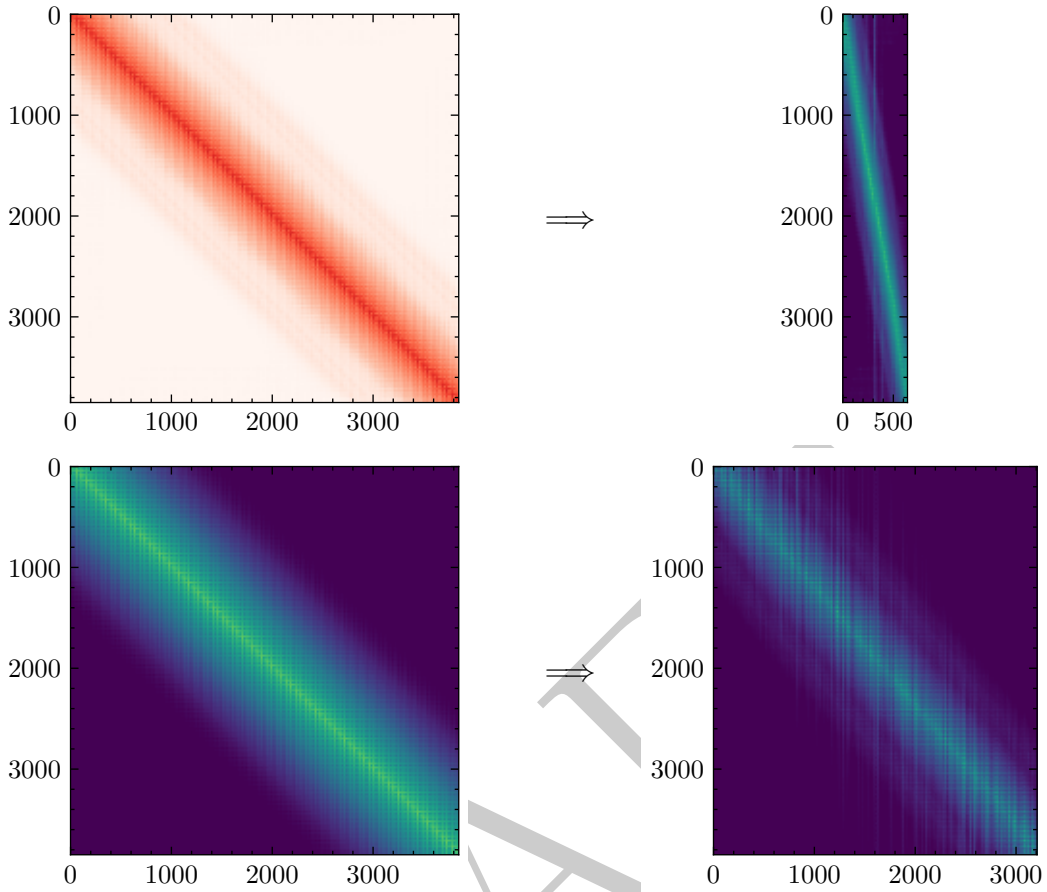


Figure 3.7: Cholesky decomposition, the occupied and virtual density matrices (left) yields a set of coefficients that describe a localized MO basis in the orbital and virtual space, respectively (right). The matrices are represented in terms of a heat map which inidicates the magnitude of $log_{10}(abs(a_{ij}))$.

are a useful orbital representation for analyzing molecular bonds (e.g. bond order, bond polarity). They are conceptually close to the traditional Lewis structure of a molecule [113, 114, 115].

Frozen Natural Orbitals

For large basis sets, the number of occupied canonical MOs is several times lower than the number of virtual canonical orbitals. Furthermore, NOs do not considerably reduce the number of occupied orbitals. It is therefore sufficient to only compute the eigenfunctions of the virtual-virtual block of the one-particle density matrix, in what is known as the frozen natural orbitals (FNOs) approach [116]. FNOs need information of the correlated wave function, and are therefore typically computed at a lower level of theory. For example, the easiest way to obtain a set of FNOs for CCSD or CCSD(T) computation is to diagonalize the virtual-virtual block of the MP2 density matrix [117, 118, 119]

$$D_{ab} = \frac{1}{2} sum_{cij} \frac{K_{ij}^{cb} K_{ij}^{ca}}{\epsilon_{ij}^{ab} \epsilon_{ij}^{ca}} \tag{3.80}$$

with

$$K_{ij}^{ab} = 2\,(ia \mid jb) - (ib \mid ja) \tag{3.81}$$

$$\epsilon_{ij}^{ab} = \epsilon_i + \epsilon_j - \epsilon_a - \epsilon_b \tag{3.82}$$

The FNOs are then canonicalized (**??**). The combined set of occupied CMOs and virtual FNOs forms a very compact representation suitable for CC ground state and excited state calculations.

Natural Transition Orbitals

Consider the CIS eigenvalue problem for finding the excitation energies ω_n and their associated transition density matrices R_n

$$\mathbf{A_{CIS}} R_n = \omega_n R_n \tag{3.83}$$

The matrices $\mathbf{R}_n$ contain $n_{occ}n_{vir}$ expansion coefficients c_{ia} which show how much an orbital-virtual MO pair ia contributes to the excitation n. The number of non-negligible coefficients can be far from zero, making interpretations of the computed results difficult for some systems.

Natural transition orbitals (NTOs) were introduced to facilitate the qualitative description of an excited state and finding connections to experimental spectra [120, 121]. NTOs are typically obtained by computing the singular value decomposition (SVD) of the state densities $\mathbf{R}_n$

$$\mathbf{R} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\dagger \tag{3.84}$$

where $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices with dimension $n_{occ}n_{NTO}$ and $n_{vir}n_{NTO}$, and Σ is a n_{NTO} by n_{NTO} matrix containing the singular values s on its diagonal. The CMOs $\{\Psi_i^{occ}, \Psi_a^{vir}\}$ are transformed to the NTO basis $\{\overline{\Psi}_k^{occ}, \overline{\Psi}_k^{vir}\}$ using

$$\left|\overline{\Psi}_k^{occ}\right\rangle = U_{ki}\left|\Psi_i^{occ}\right\rangle \tag{3.85}$$

$$\left|\overline{\Psi}_k^{vir}\right\rangle = V_{ka}\left|\Psi_a^{vir}\right\rangle \tag{3.86}$$

The singular value s_k show the contribution of an NTO pair k to the excited state. In most cases, the number of significant NTO pairs is significantly lower than $n_{occ}n_{vir}$ and at most equal to n_{occ}. NTOs are not limited to CIS, but can also be obtained by SVD decomposition of the singles-singles block of excited state densities from higher order methods such as ADC or CCLR.

Natural transition orbitals have also found use in local excited state correlation methods [122, 123], where CIS NTOs are combined with MP2 NOs to obtain a compact orbital representation for ground and excited state coupled cluster calculations.

EXAMPLE!!! phenylalanine

3.4.3 Specific Virtual Orbitals

In most cases, using LMOs instead of CMOs does not offer any a priori advantage in terms of the computational complexity associated with correlated methods, and additional approximations are necessary. In local correlation methods, this is often done by truncating the VMO space. Truncation of the VMOs has been an active field of research for several decades. A naive approach to truncate the virtual space would be to eliminate VMOs with orbital energies above a certain threshold; however, this proved to be unusable in most contexts [124]. More successful methods for VMO truncation use the concept of what will referred to as *specific virtual orbitals* (SVOs). SVOs are specific in the sense that each individual occupied MO i or each pair of MOs ij has their own set of SVOs a_i (orbital specific virtual orbitals) or a_{ij} (pair specific virtual orbitals) associated to it. The concept of SVOs naturally arises in the context of correlated methods such as the coupled electron pair approximation (CEPA) where the total energy is computed is computed as the sum of electron pair energies e

$$E_{CEPA} = \sum_{ij} e_{ij} \tag{3.87}$$

The electron pair energy decays rapidly as a function of the distance r between MO centers in an LMO basis. Distant virtual orbitals contribute less to the electron pair energy as virtual orbitals close to ij. It has been shown early on that instead of using the whole virtual orbital span, one can correlate only a subset or reduced set of virtual orbitals with each electron pair [125, 126, 127, 128] and still recover most of the correlation energy. In the limit of large molecules, the number of significant virtual orbitals for an electron pair becomes independent of system size [129] (see also Chapter 4). There are different ways to choose how to define the VMO subsets, either by using local molecular orbitals or natural orbitals.

Domain Specific Virtual Orbitals

The term *domain specific virtual orbitals* (DSVOs) will be used to denote any type of virtual orbitals where the subsets are formed *a priori* by distance or partial charge criteria. Examples include the local MP2 and local CCSD implementations by Schütz et al. [3, 130, 131].

First, occupied CMOs are localized by one of the methods described above, e.g. FB or ER. Virtual CMOs are transformed into the PAO basis. Each individual occupied LMO $|\Psi_i\rangle$ is then assigned a subset $[i]$ of PAOs, chosen by a Boughton-Pulay (BP) criterion [132] or by population analysis [133].

For a given electron pair ij, the pair domain is then formed by taking the union $[ij] = [i] \cup [j]$. The set of all virtual pair domains $[ij]$ forms the DSVOs.

Alongside AOs, DSVOs were among the first orbital representations in which linear scaling correlated methods were formulated. Their dependency on distance criteria for selecting the pair domains makes them less rigorous than other methods, and they have fallen out of favor over the years.

Pair Natural Orbitals

First introduced under the guise of "pseudo-natural orbitals" [126], then rediscovered by Neese [134, 135, 136], projected natural orbitals (PNOs) have risen in popularity in the recent years. Similarly to DSVMOs, each electron pair has a set of PNOs associated to it. PNOs are formed by diagonalizing the MP2 pair density matrix for each MO pair ij (hence "pair-natural")

$$\mathbf{D}^{ij} = \frac{1}{1+\delta_{ij}} \left(\tilde{\mathbf{t}}^{ij} \mathbf{t}^{ij} + tilde\mathbf{t}^{ij} \mathbf{t}^{ij\dagger} \right) \tag{3.88}$$

with

$$\tilde{\mathbf{t}}^{ij}_{ab} = 2\mathbf{t}^{ij}_{ab} - \mathbf{t}^{ji}_{ab} \tag{3.89}$$

The eigenvalue decomposition of $\mathbf{D}$ then gives

$$\mathbf{D}^{ij}\mathbf{Q}^{ij} = n^{ij}\mathbf{Q}^{ij} \tag{3.90}$$

where $\mathbf{Q^{ij}}$ are the pair specific transformation matrices, and n^{ij} their occupation numbers. The Fock matrix in the LMO representation is not diagonal, and the MP2 amplitudes are approximated by

$$t^{ab}_{ij} = \frac{(ia \mid jb)}{\epsilon_a + \epsilon_b - f_{ii} - f_{jj}} \tag{3.91}$$

where f_{ii} are the diagonal entries of the Fock matrix in the LMO basis. The pair domains $[ij]$ are chosen by keeping the PNOs with an occupation number larger than a threshold τ_{PAO}. Therefore, accuracy is controlled by a single, distance-independent parameter, which is an advantage over other methods like DSVOs.

However, computing the PNOs requires a full MP2 calculation, and with density fitting scales with $\mathcal{O}(N^5)$. Moreover, even if the PNO basis is compact, the fact that each LMO pair has its own virtual orbital basis may lead to a prohibitively large number of PNOs for large molecules.

The concept of PNOs can be combined with the domain-based approach using LMOs, in what is known as local pair-natural orbitals (LPNOs) [137].

Orbital Specific Virtuals

Closely related to PNOs are the orbital specific virtual orbitals (OSVs) [138]. The OSVs for an LMO $|\Psi_i\rangle$ are obtained by taking the diagonal PNOs for the domain $[ii]$. The MP2 density matrix reduces to

$$\mathbf{D}^{ii} = 4\mathbf{t}^{ii}\mathbf{t_{ii}} \tag{3.92}$$

Instead of reducing the density matrix, one can just diagonalize $\mathbf{t}^{ij}$ instead.

$$\mathbf{t}^{ii}\mathbf{Q}^{ii} = t^{ii}\mathbf{Q}^{ii} \tag{3.93}$$

where t^{ii} are the eigenvalues, which are used to compute the compute the occupation numbers $n^{ii} = (t^{ii})^2$. OSVs for which $n^{ii} > \tau_{OSV}$ are included into the orbital specific domain $[i]$. Pair domains $[ij]$ are then formed as the union of $[i]$ and $[j]$ similar to DSVOs.

OSVs have the advantage that the can be constructed with $\mathcal{O}(N^3)$ scaling provided that density fitting is used. However, OSVs are less compact than PNOs.

OSVs can be used to lower the computational complexity to construct PNOs. Several hybrid OSV-PNO schemes have been proposed whith a computaional complexity of $\mathcal{O}(N^4)$ [129], $\mathcal{O}(N^3)$ [139] and finally $\mathcal{O}(N)$ [137].

Chapter 4

Local Correlation Methods (II): Ground State

Since the seminal work of Sæbø and Pulay **??**, demonstrating how the computational effort of electronic structure methods may be reduced via a local treatment of electron correlation using LMOs and PAOs, the concept of locality has been applied to Hartree-Fock, Møller-Plesset and Coupled-Cluster methods. Over the years, pure LMO methods have somewhat fallen out of use, in favor for an atomic orbital or (local) natural orbital treatment. This chapter summarizes how the tools and concepts of sparsity and local electron correlation presented in the previous chapter can be used to reduce the scaling of electronic structure methods. The focus will mainly be on Hartree-Fock and Møller-Plesset, with some words on Coupled-Cluster as well.

4.1 Low-Scaling Self-Consistent Field Methods

Hartree-Fock and density field theory, also grouped under the umbrella term of self-consistent field (SCF) methods, are the working horses in quantum chemistry, and the underlying equations, the Roothan and Kohn-Sham equations, are well known and studied. Conventional formulations of HF and DFT, without inclusion of sparsity, scale with $\mathcal{O}(N^3)$ to $\mathcal{O}(N^4)$ which hampers extension to very large systems. There are three major bottle-necks: (1) computation of the coulomb matrix, (2) computation of the exchange matrix and (3) diagonalization of the Fock matrix. Over the last couple of decades, multiple different approaches have been proposed on how to lower the scaling of constructing the Fock matrix and circumvent matrix diagonalization. To this day, the field of low scaling SCF methods remains an active area of research in theoretical chemistry. The next sections will address the time-determining steps in detail.

4.1.1 The Coulomb Matrix

Consider again the expression for the coulomb matrix **J**

$$J_{\mu\nu} = (\mu\nu \mid \lambda\sigma)\, P_{\sigma\lambda} \tag{4.1}$$

which gives the following sparsity diagram:

$$\mu \overset{\mathrm{S}}{\longleftrightarrow} \nu \qquad \sigma \overset{\mathrm{P/S}}{\longleftrightarrow} \lambda$$

The construction of **J** has an inherent computational complexity of $\mathcal{O}(N^2)$, even though the number of non-zero elements scales linearly due to the overlap relationship between μ and ν. Quadratic scaling algorithms are straight-forward to implement. The first method to construct **J** with $\mathcal{O}(N)$ effort was the continuous fast multipole method (CFMM) [103]. For each element $\mu\nu$ the contributions $\sigma\lambda$ are split into near-field and far-field contributions. NF interactions are computed using standard integration techniques, while FF interactions are computed using multi-level multipole expansion. The linear scaling consists even if the density matrix is not sparse. Other tree-like algorithms have also been proposed [140, 141]. In all cases, computing the NF interactions are by far the most time-consuming step.

One way to speed up the evaluation of the non-classical contributions is by moving the contraction step with the density matrix into the underlying integral evaluation. By modifying the formulas for the Gaussian integrals, the explicit storage of the 2-electron repulsion integrals can be skipped which greatly increases computational efficiency [103, 142, 143]. To this day, the *J engine method*, in combination with CFMM, remains the most efficient way to evaluate the exact coulomb matrix.

Alternatively, one may introduce the density fitting approximation. The coulomb matrix is then evaluated in several steps via an intermediate **d** as

$$J_{\mu\nu} = (\mu\nu \mid X)\, \tilde{d}_X \tag{4.2}$$

$$d_X = (Y \mid \mu\nu)\, P_{\nu\mu}; \quad \tilde{d}_X = (X \mid Y)^{-1}\, d_X \tag{4.3}$$

The computational effort remains unchanged, with $\mathcal{O}(N^2)$, but with a much lower prefactor, especially for larger, more diffuse basis sets [93]. Furthermore, the integrals can be recomputed on the fly, and do not need to be explicitly stored. The inversion of the metric matrix $(X \mid Y)$ scales cubically, and dominates the cost of the DF approximation for large molecules.

For large molecules, one could also consider using local density fitting, such as the atomic resolution of the identity or pair-atomic resolution of the identity. Unfortunately, LDF also only reduces the prefactor, rather than scaling. Furthermore, it is necessary to use the robust density fitting of the electron integrals (Equation 3.31) to recover quadratic scaling in the fitting error, due to the constraints imposed on the density fitting procedure. The coulomb matrix is then expressed as

$$J_{\mu\nu} = (\mu\nu \mid X)\, b_X + c^Y_{\mu\nu}\, [g_Y - \tilde{g}_Y] \tag{4.4}$$

$$g_X = C^X_{\mu\nu} P_{\mu\nu} \tag{4.5}$$

$$\tilde{g}_X = (X \mid Y)\, b_Y \tag{4.6}$$

$$b_X = C^X_{\mu\nu} P_{\mu\nu} \tag{4.7}$$

The robust LDF-J approximation is evaluated at an effort similar to standard DF-J. The only advantages to LDF in this case pertain to the fitting procedure itself. It is no longer necessary to invert the 2c2e integral matrix $(X \mid Y)$, and the fitting coefficients $C^X_{\mu\nu}$ can be evaluated in linear scaling fashion. However, it has been demonstrated that using

Dunlap's robust formula in combination with local metrics leads to "attractive electron" states where the SCF energy may converge to very high positive values [90, 144]. The reason is that the the two-electron integral tensor in the robust LDF approximation is no longer postive semidefinite, but *indefinite*, which can lead to severe convergence problems. One way to circumvent this problem is to loosen the constraints on the density fitting procedure, or use a larger auxiliary basis set. In both cases, performance is compromised.

As shall be shown in chapter **??**, quasi-robust density offers a better alternative to robust LDF-J methods, with similar sparsity and higher accuracy without the use of Dunlap's formula.

4.1.2 The Exchange Matrix

Exact Exchange

The expression for the exchange matrix is given by

$$K_{\mu\nu} = (\mu\sigma \mid \nu\lambda)\, P_{\lambda\sigma} \tag{4.8}$$

In the section where sparsity diagrams were introduced, it was demonstrated that the indices of the exchange expression can be fully linked:

$$\mu \overset{\mathrm{S}}{\longleftrightarrow} \sigma \overset{\mathrm{P}}{\longleftrightarrow} \lambda \overset{\mathrm{S}}{\longleftrightarrow} \nu$$

The non-zero elements in $\mathbf{K}$ scale linearly and can also be evaluated with $\mathcal{O}(N)$. This property has been realized quite early on [145]. However, a straight-forward implementation where the 4c2e integrals are directly contracted with the density matrix $\mathbf{P}$ using sparse matrix algebra does not give the desired results, when applying the standard $\mathcal{O}(N^2)$ Schwarz-screening to $(\mu\sigma \mid \nu\lambda)$. To lower the scaling of electron integral evaluation, it is important to design a screening algorithm which imposes the P junction between σ and λ which in turn leads to only a linear increase in the number of bra-ket pairs.

The ONX method by Schwegler [145] was the first $\mathcal{O}(N)$ scheme for constructing the exchange matrix, but did not exploit permutational symmetry, which lead to a four fold increase in the prefactor. More competitive methods were proposed later, such as Linear Exchange (LinK) [146], or symmetrized ONX (SONX) [131] that could also be applied to small systems without major overhead.

For all approaches, an important step is the screening of the bra-ket pairs using a density-weighted integral estimate

$$|P_{\lambda\sigma}|\,|\,(\mu\sigma \mid \mu\sigma)\,|^{1/2}|\,(\nu\lambda \mid \nu\lambda)\,|^{1/2} \leq \tau \tag{4.9}$$

which scales linearly in the limit of large systems with large HOMO-LUMO gaps. Furthermore, shell ordering is very important to avoid the $\mathcal{O}(N^2)$ complexity of screening and enable early exit out of the shell loops during construction of the exchange matrix. Similarly to the J kernel, the electron integrals need not to be held in memory, but can be recomputed on the fly.

Density Fitting

The downside of exact linear exchange algorithms is the steep $\mathcal{O}(N^4)$ scaling with increasing basis set size which delays the onset of the low-scaling regime. For this reason, considerable effort has been invested in recent years to also exploit rank sparsity by density fitting. The DF expression for the exchange matrix in the AO basis reads

$$C^X_{\mu\nu} = (X \mid Y)^{-1} (Y \mid \mu\nu) \tag{4.10}$$

$$K_{\mu\nu} = C^X_{\nu\lambda} (X \mid \mu\sigma) P_{\sigma\lambda} \tag{4.11}$$

In a straight-forward implementation using sparse matrix algebra, Equation 4.10 and Equation 4.11 are evaluated with $\mathcal{O}(N^3)$ and $\mathcal{O}(N^2)$ effort respectively. The non-zero elements of both the fitting coefficients C and the 3c2e integrals increase quadratically. Their storage can quickly become problematic for large basis sets if both are held in-core. In principle, both tensors can be recomputed batchwise on-the-fly to reduce memory-footprint, but in contrast to the DF-J kernel where only the 3c2e integrals need to be generated at each iteration, recomputing the fitting coefficients each time introduces a prefactor that is too large for an out-of-core DF-K kernel to be of any practical use (see also Chapter **??**).

For an efficient, direct evaluation of the exchange matrix using density fitting, an MO based approach is much more favorable [93]. The MO-DF-K kernel is evaluated as

$$B^X_{\mu i} = C_{\nu i} (X \mid \mu\nu) \tag{4.12}$$

$$D^X_{\mu i} = (X \mid Y)^{-1/2} B^X_{\mu i} \tag{4.13}$$

$$K_{\mu\nu} = B^X_{\mu i} B^X_{\nu i} \tag{4.14}$$

with cubic computational complexity. The matrix elements of the exchange matrix are evaluated batch-wise over occupied blocks I. By contracting the 3c2e integrals with the coefficient matrix $C_{\mu i}$ to form the half-transformed integrals $B^X_{\mu i}$, storage can be reduced from $N_{aux} N^2_{AO}$ to $N_{aux} N_{AO} N_{occ} / N_I$. The 3c2e integrals need to be recomputed for each block I, but in practice the number of blocks can be held quite small. The DF-MO-K method is especially well suited for small to medium sized molecules with large diffuse basis sets for post-HF calculations.

Local Density Fitting

Standard density fitting introduces long-range interactions which inhibit the linear scaling construction of the exchange matrix. This problem can be solved by using LDF. Again, Dunlap's robust density fitting needs to applied to get accurate results. The robust DF-K kernel can take the form

$$E^X_{\mu\nu} = C^X_{\mu\sigma} P_{\lambda\nu} \tag{4.15}$$

$$L_{\mu\nu} = E^X_{\mu\sigma} (X \mid \nu\sigma) - \tfrac{1}{2} E^X_{\mu\sigma} (X \mid Y) C^Y_{\nu\sigma} \tag{4.16}$$

$$K_{\mu\nu} = L_{\mu\nu} + L_{\nu\mu} \tag{4.17}$$

All steps can be evaluated in $\mathcal{O}(N)$ time, under the assumption that the fitting coefficients scale linearly:

$$E: \quad X \overset{\text{LDF}}{\leftrightarrow} \mu \overset{\text{S}}{\leftrightarrow} \sigma \overset{\text{P}}{\leftrightarrow} \nu$$

$$L: \quad X \overset{\text{LDF}}{\leftrightarrow} \mu \overset{\text{P}}{\leftrightarrow} \sigma \overset{\text{S}}{\leftrightarrow} \nu \quad + \quad X \overset{\text{LDF}}{\leftrightarrow} \mu \overset{\text{P}}{\leftrightarrow} \sigma \overset{\text{S}}{\leftrightarrow} \nu \overset{\text{LDF}}{\leftrightarrow} Y$$

$$K: \quad \mu \leftrightarrow \nu \quad + \quad \mu \leftrightarrow \nu$$

Alternatively, an LDF-K scheme based on LMOs is also possible [147, 148]. Over the years, many different LDF-K kernels have been proposed that approximate $C^X_{\mu\nu}$ based on LMO domains [147, 148], the atomic resolution of the identity (ARI) [89], the pair-atomic resolution of the identity (PARI) [90] or the concentric atomic density fitting (CADF) [77]. Although the electron integrals are no longer positive semidefinite, LDF-K is not plagued by the same convergence problems as LDF-J, and it has been shown that LDF-K can be combined with standard DF-J to circumvent convergence problems [91].

Other Methods

Other methods for linear-scaling evaluation of Hartree-Fock exchange (also often abbreviated as HFX in literature) are the semi-numerical chain-of-spheres methods (COS) [149, 150] and the auxiliary density matrix methods (ADMM) [151]. For further details, the reader is referred to the original publications. (Maybe elaborate?)

4.1.3 The SCF Procedure

In the standard SCF procedure, the construction of the Fock matrix is followed by a cubic scaling diagonalization to obtain the MO coefficient matrix and the MO energies. As was discussed in detail in the previous chapter, the eigenvectors of the Fock matrix, i.e. the canonical MOs, are delocalized, and therefore using a sparse eigenvalue solver is unfortunately not an option. The solution is to entirely avoid any MO quantities and replace the Fock diagonalization step. For a functional, fully AO-based method, the same constraints on the density matrix need to be fulfilled as in the standard SCF procedure:

$$\mathbf{P} = \mathbf{P}^\dagger \qquad \text{Hermiticity} \tag{4.18}$$

$$Tr(\mathbf{PS}) = N_{ele} \qquad \text{N-representability} \tag{4.19}$$

$$\mathbf{PSP} = \mathbf{P} \qquad \text{Idempotency} \tag{4.20}$$

$$\mathbf{FPS} - \mathbf{SPF} = \mathbf{0} \qquad \text{Commutator} \tag{4.21}$$

There are two main approaches for replacing Fock matrix diagonalization: Purification (or spectral projection) and density matrix minimization [152].

In spectral projection methods, an initial guess for the density matrix $\mathbf{P}$ of its Fock matrix $\mathbf{F}$ is obtained as

$$\mathbf{P} = \Theta\left(\mu\mathbf{I} - \mathbf{F}\right) \tag{4.22}$$

where μ is the chemical potential, and Θ is the Heavyside step-function. The guess density then has orbital occupation numbers spread between 0 and 1, and has the same eigenvectors as the Fock matrix. The idempotent density is then determined by

density purification. Density purification is a iterative procedure where a purification transformation is repeatedly applied to the density matrix which converges the orbital occupation numbers either towards 0 or 1. One of the earliest purification transformation by McWeeny [153] takes the form

$$\mathbf{P}_{n+1} = 3\mathbf{P}_n\mathbf{S}\mathbf{P}_n - 2\mathbf{P}_n\mathbf{S}\mathbf{P}_n\mathbf{S}\mathbf{P}_n \tag{4.23}$$

Equation 4.23 is also known as the grand-canonical purification scheme. Other transformations have been proposed over the years, such as canonical purification [154] or trace-resetting purification [155], which do not need the chemical potential μ. The only operations in purification schemes are matrix multiplications, which can be made linearly scaling using sparse matrix algebra. Compared to Fock diagonalization, density purification has a larger overhead, but can be easily integrated without needing large modifications of existing SCF code.

The second method, density matrix minimization, starts from an existing idempotent density matrix guess from a previous SCF cycle and minimizes the energy functional [156, 157, 158]

$$E = Tr\left[\left(3\mathbf{PSP} - 2\mathbf{PSPSP}\right)\left(\mathbf{K} - \mu\mathbf{I}\right)\right] \tag{4.24}$$

where $\mathbf{K}$ is the effective one-electron Hamiltonian matrix. The energy minimum is found either by gradient descent or the curvy-step approach [9, 159].

Routine application of density matrix purification or minimization has been mainly limited by uncontrolled error accumulation and convergence problems [160].

4.2 Møller-Plesset

Second-Order Møller Plesset is one of the simplest post-Hartree Fock methods available, but still scales as $\mathcal{O}(N^5)$. Attempts to reduce computational complexity can generally be grouped into two categories: AO-MP2, LMO-MP2. Independent on which method is used, they share two problems.

First, the energy denominator in the MP2-amplitudes t make it difficult to transform the MP2 energy expressions into a different basis. AO-MP2 and LMO-MP2 take different approaches to an *orbital-invariant* formulation of the MP2 energy expressions: AO-MP2 solves the problem using the Laplace quadrature, while LMO-MP2 methods generally use the Hylleraas functional.

Second, steps involving the transformation of the AO 2-electron integrals to the Pseudo-AO or LMO basis still remain a major bottle-neck, even with sparsity involved. Both AO- and LMO-MP2 use screening criteria, additional domain restrictions, density fitting or similar methods to lower the cost of integral transformation. These additional procedures are crucial if one wishes to achieve a truly linear scaling MP2 method with a reduced overhead.

4.2.1 Atomic Orbital MP2

MP2 was first formulated in the AO basis in 1993 by Häser [161], and a linear scaling algorithm was presented by Scuseria and Ayala in 1999 [162].

The Laplace Transform

In 1991, Almlöf showed [163] that the energy denominator in the MP2 amplitudes can be removed using an integral transform called the *Laplace Transform*

$$\frac{1}{\epsilon_a + \epsilon_b - \epsilon_i - \epsilon_j} = \int_0^\infty e^{-(\epsilon_a+\epsilon_b-\epsilon i-\epsilon_j)t} dt \tag{4.25}$$

The t-integration can be replaced [161] by a finite summation using a functional approximation:

$$\frac{1}{\epsilon_a + \epsilon_b - \epsilon_i - \epsilon_j} \approx \sum_\alpha^n w^{(\alpha)} e^{-(\epsilon_a+\epsilon_b-\epsilon i-\epsilon_j)t^{(\alpha)}} \tag{4.26}$$

where $w^{(\alpha)}$ and $t^{(\alpha)}$ are the Laplace weights and exponents at the Laplace points α. Accuracy can be controlled by the number of Laplace points n. An efficient AO-MP2 implementation heavily relies on an accurate quadrature scheme to achieve the desired accuracy using as few Laplace points as possible to reduce overhead caused by the repeated AO transformation at each step. In general, 5-8 Laplace points are needed to achieve milli-Hartree accuracy, and 10 to 15 points for μHartree accuracy (see also 11.4).

AO-MP2 Equations

Using the Laplace transform, the energy expression for restricted canonical MP2 can be expressed as

$$\begin{aligned} E_{MP2} &= -\sum_{iajb} \frac{(ia \mid jb)\left[2\,(ia \mid ib) - (ib \mid ja)\right]}{\epsilon_a + \epsilon_b - \epsilon_i - \epsilon_j} \\ &\approx -\sum_\alpha^n \sum_{iajb} (ia \mid jb)\left[2\,(ia \mid ib) - (ib \mid ja)\right] w^{(\alpha)} e^{-(\epsilon_a+\epsilon_b-\epsilon i-\epsilon_j)t^{(\alpha)}} \end{aligned} \tag{4.27}$$

The coefficient matrices can then be factored out as

$$\begin{aligned} &-\sum_\alpha^n \sum_{iajb} (ia \mid jb)\left[2\,(ia \mid ib) - (ib \mid ja)\right] w^{(\alpha)} e^{-(\epsilon_a+\epsilon_b-\epsilon i-\epsilon_j)t^{(\alpha)}} \\ &= -\sum_\alpha^n \sum_{iajb} \sum_{\substack{\mu\nu\lambda\sigma \\ \mu'\nu'\lambda'\sigma'}} w^{(\alpha)} e^{-(\epsilon_a+\epsilon_b-\epsilon i-\epsilon_j)t^{(\alpha)}} C_{\mu' i} C_{\sigma' a} \left(\mu'\sigma' \mid \nu'\lambda'\right) C_{\nu' j} C_{\lambda' b} \\ &\qquad \times \left\{C_{\mu i} C_{\sigma a}\left[2\,(\mu\sigma \mid \nu\lambda) - (\mu\lambda \mid \nu\sigma)\right] C_{\nu j} C_{\lambda b}\right\} \\ &= -\sum_\alpha^n \sum_{\substack{\mu\nu\lambda\sigma \\ \mu'\nu'\lambda'\sigma'}} P^{(\alpha)}_{\mu\mu'} Q^{(\alpha)}_{\sigma\sigma'} \left(\mu'\sigma' \mid \nu'\lambda'\right) P^{(\alpha)}_{\nu\nu'} Q^{(\alpha)}_{\lambda\lambda'} \left[2\,(\mu\sigma \mid \nu\lambda) - (\mu\lambda \mid \nu\sigma)\right] \end{aligned} \tag{4.28}$$

with the occupied and virtual *pseudo* or *Laplace* density matrices

$$\begin{aligned} P^{(\alpha)}_{\mu\mu'} &= \sum_i C_{\mu i} e^{0.25 ln(w^{(\alpha)}) + \epsilon_i t^{(\alpha)}} C_{\mu' i} \\ Q^{(\alpha)}_{\mu\mu'} &= \sum_i C_{\sigma a} e^{0.25 ln(w^{(\alpha)}) - \epsilon_a t^{(\alpha)}} C_{\sigma' i} \end{aligned} \tag{4.29}$$

Introducing the *pseudo-AO* transformed electron integrals

$$\left(\underline{\mu}\overline{\sigma} \mid \underline{\nu}\overline{\lambda}\right)^{(\alpha)} = P^{(\alpha)}_{\mu\mu'} Q^{(\alpha)}_{\sigma\sigma'} \left(\mu'\sigma' \mid \nu'\lambda'\right) P^{(\alpha)}_{\nu\nu'} Q^{(\alpha)}_{\lambda\lambda'} \tag{4.30}$$

the energy expression for AO-MP2 then reads

$$E_{AO-MP2} = -\sum_{\alpha}^{n} \sum_{\mu\nu\lambda\sigma} \left(\underline{\mu}\overline{\sigma} \mid \underline{\nu}\overline{\lambda}\right)^{(\alpha)} \left[2\left(\mu\sigma \mid \nu\lambda\right) - \left(\mu\lambda \mid \nu\sigma\right)\right] \tag{4.31}$$

For $t = 0$, $\mathbf{P}^{(\alpha)}$ and $\mathbf{Q}^{(\alpha)}$ are equal to the Hartree Fock density matrices. The Laplace matrices also fulfill similar relationships:

$$\mathbf{P}^{(\alpha)}\mathbf{S}\mathbf{P}^{(\alpha)} = \mathbf{0} \tag{4.32}$$

$$\mathbf{P}^{(\alpha)}\mathbf{S} + \mathbf{Q}^{(\alpha)}\mathbf{S} = \mathbf{I}_{exp} \tag{4.33}$$

where $\mathbf{I}_{exp}$ is a diagonal matrix with trace

$$Tr[\mathbf{I}_{exp}] = \sum_i e^{0.25 ln(w^{(\alpha)}) + \epsilon_i t^{(\alpha)}} + \sum_a e^{0.25 ln(w^{(\alpha)}) - \epsilon_a t^{(\alpha)}} \tag{4.34}$$

The entries of the pseudo-density also decay exponentially as function of the distance between pseudo-AO centres. Strictly speaking, AO-MP2 is not only formulated in a pure AO-basis ($\mu\nu$...), but also a PAO-like basis ($\underline{\mu}, olgn, ...$).

Quadratic Scaling AO-MP2

Using the linked index rule, the complexity of the AO-MP2 method can easily be determined. The energy expression in Equation 4.31 involves the dot product between two different tensors, the AO-ERIs $(\mu\sigma \mid \nu\lambda)$ and the pseudo-AO-ERIs $\left(\underline{\mu}\overline{\sigma} \mid \underline{\nu}\overline{\lambda}\right)^{(\alpha)}$. The scaling is thus determined by the sparsity of those two tensors. From the discussion in Chapter 3, it followed that the ERIs can be computed with $\mathcal{O}(N^2)$ effort. The pseudo-AO ERIs are computed by transforming the ERIs with the pseudo-density matrices, whose indices μ, ν are connected by a P junction. The diagrammatic expression for the pseudo-AO ERIs 4.30 is given by

$$\begin{aligned} \mu &\overset{P}{\longleftrightarrow} \mu' \overset{S}{\longleftrightarrow} \sigma' \overset{P}{\longleftrightarrow} \sigma \\ \nu &\overset{P}{\longleftrightarrow} \nu' \overset{S}{\longleftrightarrow} \lambda' \overset{P}{\longleftrightarrow} \lambda \end{aligned} \tag{4.35}$$

Two vertices indicate an $\mathcal{O}(N^2)$ effort for evaluating Equation 4.30. Therefore, the inherent asymptotic scaling of AO-MP2, without any other further approximations, is

$\mathcal{O}(N^2)$ as well. Similarly to the AO ERIs, a quadratic scaling evaluation of the pseudo-AO ERIs can be achieved using a Schwarz-like screening, as first advocated by Almlöf. Defining the screening matrices

$$\begin{aligned} Q_{\mu\nu} &= \mid (\mu\nu \mid \mu\nu) \mid^{1/2} \\ X_{\mu\nu} &= \mid \left(\underline{\mu}\nu \mid \underline{\mu}\nu\right) \mid^{1/2} \\ Y_{\mu\nu} &= \mid (\mu\overline{\nu} \mid \mu\overline{\nu}) \mid^{1/2} \\ Z_{\mu\nu} &= min\left(\sum_{\sigma} A_{\mu\sigma} \mid \overline{P}_{\sigma\nu} \mid; \sum_{\sigma} B_{\mu\sigma} \mid \underline{P}_{\sigma\nu} \mid\right) \end{aligned} \tag{4.36}$$

gives an upper-bound for each transformation step in Equation 4.30, for example

$$(\mu'\sigma' \mid \nu'\lambda') \leq Q_{\mu'\sigma'} Q_{\nu'\lambda'} \tag{4.37}$$

$$\left(\underline{\mu}\sigma' \mid \nu'\lambda'\right) \leq X_{\mu'\sigma'} Q_{\nu'\lambda'} \tag{4.38}$$

$$\left(\underline{\mu}\overline{\sigma} \mid \underline{\nu}\overline{\lambda}\right) \leq Z_{\mu\sigma} Z_{\nu\lambda} \tag{4.39}$$

An efficient screening protocol can be devised [161] to get quadratic scaling AO-MP2.

Linear Scaling AO-MP2

For the two-electron repulsion integrals, the $1/R$ decay between the charge densities $(\mu\sigma|$ and $|\nu\lambda)$ is too slow to be of any use even for large systems. However, it has been shown [164] that *bra* and *ket* in the Laplace integral tensor $e^{(\alpha)}$ decays much faster with $1/R^3$. Here, we follow the discussion in Ref [165].

For two non-overlapping charge densities $(\mu\sigma|$ and $|nu\lambda)$ the following inequality holds

$$(\mu\sigma \mid \nu\lambda) = (\mu\sigma| \frac{1}{\mathbf{r}_{12}} |\nu\lambda) \leq \frac{1}{R} \left| \sum_{n=0}^{\infty} \frac{(\mu\sigma| \left(\mathbf{r}_1 - \mathbf{r}_2\right)^n |\nu\lambda)}{R^n} \right| \tag{4.40}$$

Introducing the following abbreviation for the nth order 1-center multipole integrals

$$M_{\mu\sigma}^{(n)} = \int \chi_{\mu}(r_1) \mathbf{r}_1^n \chi_{\sigma}(r_1) dr \tag{4.41}$$

where M^0 are the overlap integrals, M^1 are the dipole integrals etc. Rewriting equation 4.40 as a multipole expansion gives

$$\begin{aligned} (\mu\sigma \mid \nu\lambda) \leq\ & R^{-1} \left| M_{\mu\sigma}^{(0)} M_{\nu\lambda}^{(0)} \right| + R^{-2} \left| M_{\mu\sigma}^{(1)} M_{\nu\lambda}^{(0)} - M_{\mu\sigma}^{(0)} M_{\nu\lambda}^{(1)} \right| \\ & + R^{-3} \left| M_{\mu\sigma}^{(2)} M_{\nu\lambda}^{(0)} - 2 M_{\mu\sigma}^{(1)} M_{\nu\lambda}^{(1)} + M_{\mu\sigma}^{(0)} M_{\nu\lambda}^{(2)} \right| \\ & + R^{-4} \left| M_{\mu\sigma}^{(3)} M_{\nu\lambda}^{(0)} - 3 M_{\mu\sigma}^{(2)} M_{\nu\lambda}^{(1)} + 3 M_{\mu\sigma}^{(1)} M_{\nu\lambda}^{(2)} - M_{\mu\sigma}^{(0)} M_{\nu\lambda}^{(3)} \right| \\ & + \mathcal{O}(R^{-5}) \end{aligned} \tag{4.42}$$

From equation 4.32, it follows that $M^{(0)}_{\underline{\mu}\overline{\sigma}} = S_{\underline{\mu}\overline{\sigma}} = 0$. The multipole expansion for the pseudo-AO ERIs $e^{(\alpha)}$ therefore reduces to

$$\begin{aligned}\left(\underline{\mu}\overline{\sigma} \mid \underline{\nu}\overline{\lambda}\right) \leq\ & R^{-3}\left|-2M^{(1)}_{\underline{\mu}\overline{\sigma}}M^{(1)}_{\underline{\nu}\overline{\lambda}}\right| \\ & + R^{-4}\left|-3M^{(2)}_{\underline{\mu}\overline{\sigma}}M^{(1)}_{\underline{\nu}\overline{\lambda}} + 3M^{(1)}_{\underline{\mu}\overline{\sigma}}M^{(2)}_{\underline{\nu}\overline{\lambda}}\right| + \mathcal{O} \\ & + \mathcal{O}(R^{-5})\end{aligned} \tag{4.43}$$

which shows the $1/R^3$ dependence of the tensor $\left(\underline{\mu}\overline{\sigma} \mid \underline{\nu}\overline{\lambda}\right)$. Combined with the $1/R$ decay of the AO ERIs, this leads to an overall $1/R^4$ behavior for the AO-MP2 energy. This long-range decay can be exploited to introduce a sparsity relationship between the bra and ket quantities, and reduce the scaling of AO-MP2 from $\mathcal{O}(N^2)$ to $\mathcal{O}(N^1)$. In the original paper by Ayala and Scuseria [164], this decay was accounted for by introducing an interaction domain centered on each atomic orbital μ in the form of a sphere. For the integrals $\left(\underline{\mu}\overline{\mu} \mid \underline{\nu}\overline{\nu}\right)$, the domain $\mathcal{D}(\mu)$, comprises all charge distributions $\sigma\lambda$ for which

$$\left(\underline{P}_{\mu\sigma}S_{\sigma\lambda}\overline{P}_{\lambda\mu}\right) \geq \epsilon \tag{4.44}$$

The radius R_μ of the interaction sphere is defined by the maximum distance between μ and the charge density $\sigma\lambda$ in its domain. One can then screen long-range behavior for the interaction sphere μ and ν by the distance criterion

$$r_{\mu\nu} - R_\mu - R_\nu \geq r_0 \tag{4.45}$$

The biggest drawback of the scheme above is that the threshold parameters r_0 and ϵ are system-dependent. A more rigorous screening method has been proposed by Lambrecht et al. known as multipole based integral estimates (MBIE) [166, 165, 167]. MBIEs offer a tight upper bound for the AO and pseudo-AO electron integrals by using the multipole expansion and replacing the higher order terms $\mathcal{O}(R^{-5})$ by lower-order ones.

Cholesky Decomposition of Pseudo-Densities

As with any method formulated entirely in an AO basis, AO-MP2 suffers from $\mathcal{O}(N^4)$ scaling with increasing basis set N. The cost associated with larger basis sets can be mitigated by Cholesky decomposition of the pseudo-density matrices (CDD) [168]. Similar to the orbital localization technique described in Chapter 3, where the (incomplete) CD of the occupied and virtual Hartree Fock density matrices yields a set of occupied and virtual Cholesky molecular orbitals, the CD of the pseudo-density matrices $\underline{P}^{(\alpha)}$ and $\overline{P}^{(\alpha)}$ yields a set of Cholesky pseudo-molecular orbitals:

$$\mathbf{P}^{(\alpha)} = \underline{\mathbf{L}}^{(\alpha)}\underline{\mathbf{L}}^{(\alpha)T} \tag{4.46}$$

$$\mathbf{Q}^{(\alpha)} = \overline{\mathbf{L}}^{(\alpha)}\overline{\mathbf{L}}^{(\alpha)T} \tag{4.47}$$

The pseudo-molecular orbitals show a local behavior inherited from the sparsity of the pseudo-density matrices. It has been observed however [169], that the pseudo-MOs L are

not always very well localized. A more localized set of MOs can be obtained by using the orthogonalized pseudo-density matrices, for example in the case of $\underline{\mathbf{P}}^{(\alpha)}$:

$$\mathbf{P}^{(\alpha)}_{orth} = \mathbf{S}^{1/2}\mathbf{P}^{(\alpha)}\mathbf{S}^{1/2} \tag{4.48}$$

The pseudo-MO coefficients are then computed as

$$\underline{L}^{(\alpha)} = \mathbf{S}^{-1/2}\underline{\mathbf{L}}^{(\alpha)}_{orth} \tag{4.49}$$

The square root and inverse square root of the overlap matrix $\mathbf{S}$ are most reliably found by (full) Cholesky decomposition. The number of occupied and virtual pseudo-MOs is given by the rank of the occupied/virtual pseudo-density matrices, which is equal or a little less than the number of occupied/virtual CMOs.

One can then formulate the CD-AO-MP2 energy expression as

$$E_{CD-AO-MP2} = -\sum_{\alpha}^{n}\sum_{\underline{i}\overline{a}\underline{j}\overline{b}} \left(\underline{i}\overline{a} \mid \underline{j}\overline{b}\right)^{(\alpha)} \left[2\left(\underline{i}\overline{a} \mid \underline{j}\overline{b}\right)^{(\alpha)} - \left(\underline{i}\overline{b} \mid \underline{j}\overline{a}\right)^{(\alpha)}\right] \tag{4.50}$$

whith the pseudo-MO integrals

$$\left(\underline{i}\overline{a} \mid \underline{j}\overline{b}\right)^{(\alpha)} = \underline{L}^{(\alpha)}_{\mu\underline{i}}\overline{L}^{(\alpha)}_{\sigma\overline{a}} \left(\mu\sigma \mid \nu\lambda\right) \underline{L}^{(\alpha)}_{\nu\underline{j}}\overline{L}^{(\alpha)}_{\lambda\overline{b}} \tag{4.51}$$

CD-AO-MP2 therefore reduces the sizes of the tensors from N^4_{AO} to $N^2_{occ}N^2_{vir}$, while still being sparse. Similar to AO-MP2, Schwarz screening and interaction domains can be introduced to obtain quadratic and linear scaling CDD-AO-MP2.

Density Fitting in AO-MP2

To reduce the prefactor associated with integral transformation, either from AOs to pseudo-AOs, or from AOs to pseudo-MOs, on can furthermore introduce density fitting [168, 170]. The transformed 3c2e integrals are given at each Laplace point α by

$$\left(X \mid \underline{\mu}\overline{\nu}\right)^{(\alpha)} = \left(X \mid \mu'\nu'\right) \underline{P}^{(\alpha)}_{\mu\mu'}\overline{P}^{(\alpha)}_{\nu\nu'} \tag{4.52}$$

which are evaluated with $\mathcal{O}(N^2)$ cost. Using local density fitting approximations, this step can be reduced to approximately $\mathcal{O}(N)$ [171].

SOS-AO-DF-MP2

SOS-MP2 is a cost-efficient variant of MP2 with improved accuracy (Section 1.8.4. One of the advantages of SOS-MP2 is that by omitting the same-spin contributions, the energy expressions can be effciently factored when using the density fitting approximation [170, 171]:

$$\begin{aligned} E_{AO-DF-SOS-MP2} = -c_{os}\sum_{\alpha=1}^{nlap}\sum_{\mu\nu\sigma\lambda} &\left(\underline{\nu}\overline{\sigma} \mid X\right)^{(\alpha)} \left(X \mid Y\right)^{-1} \left(Y \mid \underline{\nu}\overline{\lambda}\right)^{(\alpha)} \\ &\left(\mu\sigma \mid X'\right)\left(X' \mid Y'\right)\left(Y' \mid \nu\lambda\right) \end{aligned} \tag{4.53}$$

Introducing the intermediates

$$Z_{XY}^{(\alpha)} = \left(X \mid \underline{\mu}\overline{\sigma}\right)^{(\alpha)} (\mu\sigma \mid Y) \tag{4.54}$$

$$\tilde{Z}_{XY}^{(\alpha)} = (X \mid R)^{-1} Z_{RX}^{(\alpha)} \tag{4.55}$$

a compact energy expression can be obtained which reads

$$E_{AO-DF-SOS-MP2} = -c_{os} \sum_{\alpha=1}^{nlap} \sum_{XY} \tilde{Z}_{XY}^{(\alpha)} \tilde{Z}_{YX}^{(\alpha)} \tag{4.56}$$

The computation of $\mathbf{Z}^{(\alpha)}$ is the time-determining step. The sparse map of the intermediate is given by

$$\begin{array}{ccc} X & \mu' \overset{\mathrm{S}}{\leftrightarrow} \nu' & \\ & \mathrm{P}\updownarrow \qquad \updownarrow\mathrm{P} & \\ & \mu \underset{\mathrm{S}}{\leftrightarrow} \nu & Y \end{array}$$

which suggests that AO-DF-SOS-MP2 has an overall asymptotic scaling of $\mathcal{O}(N^3)$. With local density fitting, the graph can become fully connected

$$\begin{array}{ccc} & \mathrm{LDF} & \\ X & \mu' \overset{\mathrm{S}}{\leftrightarrow} \nu' & \\ & \mathrm{P}\updownarrow \qquad \updownarrow\mathrm{P} & \\ & \mu \underset{\mathrm{S}}{\leftrightarrow} \nu & Y \\ & & \mathrm{LDF} \end{array}$$

where "LDF" is the sparsity relationship introduced between the auxiliary density X and the product density $(\mu\nu|$, which is metric-specific. In the case of quasi-robust density fitting, LDF = S, and the intermediates $\mathbf{Z}^{(\alpha)}$ can be constructed with linear effort. For weaker decay behaviour, such as the error function coulomb-attenuated metric, the scaling is intermediate between linear and quadratic [171].

4.2.2 Local Molecular Orbital MP2

While linear scaling MP2 was first achieved using an atomic orbital formulation, the first low-scaling MP2 implementations were actually formulated in a local molecular orbital basis with domain-specific virtual orbitals [172, 125, 173, 174, 175]. While pure LMO methods are not used very often nowadays, the concepts and tools they introduced are still found in the context of (pair) natural orbitals [176].

Laplace LMP2

In the local molecular orbital basis, the Fock matrix is no longer diagonal, and the amplitudes t_{ia}^{jb} cannot be easily expressed in a local basis, due to the energy denominator. AO-MP2 tackle this problem by virtue of the Laplace transform. Similarly, one can

obtain an energy expression in the LMO basis. The Laplace decomposed MP2 energy is given by

$$E_{MP2} = \sum_{\alpha}^{n} \sum_{iajb} |w^{(\alpha)}| e^{(\epsilon_i + \epsilon_j - \epsilon_a - \epsilon_j) t^{(\alpha)}} \left[2\,(ia \mid jb) - (ib \mid ja) \right] (ia \mid jb) \tag{4.57}$$

Introducing the unitary occupied and virtual LMO-MO transformation matrix $\mathbf{U}$

$$|i\rangle = U_{i\underline{i}} \, |\underline{i}\rangle \tag{4.58}$$

$$|a\rangle = U_{i\overline{a}} \, |\overline{a}\rangle \tag{4.59}$$

which is factorized out, Equation 4.57 becomes

$$\begin{aligned} E_{MP2} &= \sum_{\alpha}^{n} \sum_{iajb} \sum_{\underline{i}\overline{a}\overline{b}} \sum_{\underline{k}\overline{c}\underline{l}\overline{d}} |w^{(\alpha)}| e^{(\epsilon_i + \epsilon_j - \epsilon_a - \epsilon_j) t^{(\alpha)}} U_{i\underline{i}} U_{a\overline{a}} \left[2\left(\underline{i}\overline{a} \mid \underline{j}\overline{b}\right) - \left(\underline{j}\overline{b} \mid \underline{j}\overline{b}\right) \right] U_{j\underline{j}} U_{b\overline{b}} \\ &\qquad U_{i\underline{k}} U_{a\overline{c}} \left(\underline{k}\overline{c} \mid \underline{l}\overline{d}\right) U_{j\underline{l}} U_{b\overline{d}} \\ &= \sum_{\alpha}^{n} \sum_{\underline{i}\overline{a}\overline{b}} \left[2\left(\underline{i}\overline{a} \mid \underline{j}\overline{b}\right) - \left(\underline{j}\overline{b} \mid \underline{j}\overline{b}\right) \right] \sum_{\underline{k}\overline{c}\underline{l}\overline{d}} X^{(\alpha)}_{\underline{i}\underline{k}} Y^{(\alpha)}_{\overline{a}\overline{c}} \left(\underline{k}\overline{c} \mid \underline{l}\overline{d}\right) X^{(\alpha)}_{\underline{j}\underline{l}} Y^{(\alpha)}_{\overline{b}\overline{d}} \\ &= \sum_{\underline{i}\overline{a}\overline{b}} \left[2\left(\underline{i}\overline{a} \mid \underline{j}\overline{b}\right) - \left(\underline{j}\overline{b} \mid \underline{j}\overline{b}\right) \right] \mathcal{T}_{\underline{i}\overline{a}\underline{j}\overline{b}} \end{aligned} \tag{4.60}$$

with the Laplace amplitudes $\mathcal{T}$ and the Laplace matrices

$$X^{(\alpha)}_{\underline{i}\underline{k}} = \sum_{i} U_{i\underline{i}} |w^{(\alpha)}|^{1/4} e^{\epsilon_i t^{(\alpha)}} U_{k\underline{k}} \tag{4.61}$$

$$Y^{(\alpha)}_{\overline{a}\overline{c}} = \sum_{a} U_{a\overline{a}} |w^{(\alpha)}|^{1/4} e^{-\epsilon_a t^{(\alpha)}} U_{c\overline{c}} \tag{4.62}$$

Equation 4.60 is the general expression for the MP2 energy in a local molecular orbital basis, where both the occupied and virtual orbitals are *orthogonal*. The situation changes slightly when using non-orthogonal, domain-specific PAOs $\overline{\mu}$, which are related to CMOs via

$$|\overline{\mu}\rangle = \mathbf{S}\mathbf{C}_{\mu i} \, |i\rangle = \bar{C}_{\mu i} \, |i\rangle \, |i\rangle \qquad\qquad = \bar{C}_{\nu i} S^{-1}_{\overline{\nu}\overline{\mu}} \, |\overline{\mu}\rangle \tag{4.63}$$

with $\mathbf{S}$ being the overlap matrix in the PAO basis. Its inverse is computed by canonical orthogonalization (Annex). This approximate inverse will be given by the matrix $\mathbf{V}$, which is also specific to a given pair $[ij]$. In general, only the virtual CMOs are transformed to the PAO base. The virtual Laplace matrix then takes the following form [177]:

$$\mathbf{Y}^{(\alpha)} = \mathbf{V}\mathbf{B}^{(\alpha)}\mathbf{V}^{\dagger} \tag{4.64}$$

$$B^{(\alpha)}_{\overline{\mu}\overline{\nu}} = \sum_{a} \bar{C}_{\mu a} |w^{(\alpha)}|^{1/4} e^{-\epsilon_a t^{(\alpha)}} \bar{C}_{\nu a} \tag{4.65}$$

with the other expressions of the LMO-MP2 energy remaining the same as before.

Hylleraas Functional

Alternatively, the local MP2 amplitudes can be determined iteratively via an orbital-invariant formulation of the MP2 energy expression based on the *Hylleraas functional* [178, 173]. The Hylleraas functional form of the energy is given by minimizing

$$E^{(2)} = min\left[2\left\langle\Psi^{(1)}\right|\mathbf{H} - E_0\left|Psi^{(0)}\right\rangle - \left\langle\Psi^{(1)}\right|\mathbf{H}_0 - E_0\left|\Psi^{(1)}\right\rangle\right] \tag{4.66}$$

In the case of MP2, the quantities in Equation 4.66 take the form

$$\left\langle\Psi^{(1)}\right|\mathbf{H} - E_0\left|\Psi^{(0)}\right\rangle = \frac{1}{4}\sum_{ijab} t_{ijab}\left\langle ij|ab\right\rangle \tag{4.67}$$

$$\left\langle\Psi^{(1)}\right|\mathbf{H}_0 - E_0\left|\Psi^{(1)}\right\rangle = \frac{1}{8}\sum_{ijabc} t_{iajb}f_{cb}t_{iajc} - \frac{1}{8}\sum_{ijkab} t_{iajb}f_{jk}t_{iakb} \tag{4.68}$$

Minimization of the MP2 Hylleraas functional, with respect to the amplitudes $\mathbf{t}$ yields a set of linear equations given by

$$R_{iajb} = \left\langle ij\right|\left|ab\right\rangle + \sum_c\left(t_{ijab}f_{cb} + f_{ac}t_{iacb}\right) - \sum_k\left(t_{iakb}f_{kj} + f_{ik}t_{kajb}\right) = 0 \tag{4.69}$$

where $\mathbf{R}$ is the residual. The amplitudes $\mathbf{t}$ are then no longer computed directly by a closed expression, but iteratively by solving the system of equations, in a similar vein to coupled cluster. For a set of orthogonal MOs $\underline{i}$ and $\overline{a}$, the quantities in Equation 4.69 are simply replaced by their local equivalent. Analogous to LT-LMP2, if PAOs are to be used for the virtual orbital space, the non-orthogonality needs to be taken into consideration. For a mixed LMO-PAO basis, the MP2 residual reads

$$\begin{aligned} R_{\underline{i}\overline{\mu}\underline{j}\overline{\nu}} = \left\langle\underline{ij}\right|\left|\overline{\mu\nu}\right\rangle + \sum_{\overline{\sigma}\overline{\lambda}}\left(f_{\overline{\mu}\overline{\sigma}}t_{\underline{i}\overline{\sigma}\underline{j}\overline{\lambda}}S_{\overline{\lambda}\overline{\nu}} + S_{\overline{\mu}\overline{\sigma}}t_{\underline{i}\overline{\sigma}\underline{j}\overline{\lambda}}f_{\overline{\lambda}\overline{\nu}}\right) \\ - \sum_{\underline{k}}\left(f_{\underline{i}\underline{k}}S_{\overline{\mu}\overline{\sigma}}t_{\underline{k}\overline{\sigma}\underline{j}\overline{\lambda}}S_{\overline{\lambda}\overline{\nu}} + f_{\underline{k}\underline{j}}S_{\underline{\mu}\overline{\sigma}}t_{\underline{i}\overline{\sigma}\underline{j}\overline{\lambda}}S_{\overline{\lambda}\overline{\nu}}\right) = 0 \end{aligned} \tag{4.70}$$

For specific virtual orbitals, the equations are solved individually for each electron pair ij to obtain their amplitude $\mathbf{t}_{ij}$ and to compute the pair correlation energy.

Quadratic Scaling LMP2

Similar to CEPA, the MP2 energy can be computed as a sum of electron pair energies

$$E_{MP2} = \sum_{ij} e_{ij} \tag{4.71}$$

$$e_{ij} = \left(2t_{ij}^{ab} - t_{ij}^{ba}\right)(ia \mid jb) \tag{4.72}$$

For well localized orbitals, the electron pair correlation e_{ij} decays with $1/r_{ij}^6$ with the distance between orbital centers. Electron pairs are generally divided into four groups: strong pairs ($r_{ij} < 1a_0$), weak pairs ($1 < r_{ij} \leq 8a_0$), distant pairs ($8 < r_{ij} \leq 15a_0$), and very distant pairs ($15 < r_{ij}$) [3]. Other than by distance criteria, electron pairs can also

be grouped by their pair energy [78]. Figure **??** shows the number of significant electron pairs in each category for glycine chains. The number of strong, weak and distant pairs scale as $\mathcal{O}(N)$, while the number of very distant pairs scales quadratically.

The occupied molecular orbitals ij are localized using e.g. Foster-Boys, Pipiek-Mezey or a Cholesky decomposition of the density matrix. Virtual orbitals are generally localized by projection onto the atomic orbital space (PAOs) and subsequently assigning them to pair domains $[ij]$ (domain specific virtuals), or by diagonalizing the MP2 density matrix for each electron pair (pair natural orbitals). In all cases, the number of virtual orbitals within a domain scales as $\mathbf{O}(1)$ for each electron pair ij, in the limit of large molecules, such that the scaling of LMP2 is determined by the scaling of significant electron pairs only.

The major bottle-neck in LMP2 is, as usual, the transformation of the 2 electron integrals from the AO basis into the local basis

$$\left(\underline{i}\overline{a} \mid \underline{j}\overline{b}\right) = L_{\mu\underline{i}} L_{\sigma\overline{a}} \left(\mu\sigma \mid \nu\lambda\right) L_{\nu\underline{j}} L_{\lambda\overline{b}} \tag{4.73}$$

The expression above translates into the sparsity diagram

$$\begin{array}{ccc} \mu & \overset{\mathrm{S}}{\longleftrightarrow} & \sigma \\ \updownarrow & & \updownarrow \\ \underline{i} & \longleftrightarrow & \overline{a} \end{array} \qquad \begin{array}{ccc} \nu & \overset{\mathrm{S}}{\longleftrightarrow} & \lambda \\ \updownarrow & & \updownarrow \\ \underline{j} & \longleftrightarrow & \overline{b} \end{array}$$

which indicates that the MO integrals can be evaluated with $\mathcal{O}(N^2)$ effort without further approximations. One thing to note is that the quadratic scaling is also obtained, even if the sparsity relationships $i \leftrightarrow a$ and $j \leftrightarrow b$ did not exist, i.e. where virtual orbitals are localized, but not grouped into (pair) domains. The major disadvantage of such non-pair specific methods is that the virtual orbital space is less compact, which leads to a high overhead for integral transformation involving virtual orbitals, which could be the reason that there are no examples in literature using such a scheme. Establishing an a priori sparsity relationship between occupied and virtual space allows to more easily reach the low-scaling regime.

Linear Scaling LMP2

It has been found [174] early on that the quadratic scaling very distant electron pairs can safely ignored without major impact on the total correlation energy. Distant pairs may also be approximated either by a multipole expansion [179] or empirically [180]. As a consequence, this establishes a sparsity relationship between i and j, and the sparsity diagram for the MO integrals becomes fully connected

$$\begin{array}{ccc} \mu & \overset{\mathrm{S}}{\longleftrightarrow} & \sigma \\ \updownarrow & & \updownarrow \\ \underline{i} & \longleftrightarrow & \overline{a} \end{array} \qquad \begin{array}{ccc} \nu & \overset{\mathrm{S}}{\longleftrightarrow} & \lambda \\ \updownarrow & & \updownarrow \\ \underline{j} & \longleftrightarrow & \overline{b} \end{array}$$

$$\underbrace{\qquad\qquad}_{1/R^6}$$

Linear scaling MP2 can therefore be achieved suing an LMO formulation [3].

Instead of using distance criteria, can use screening:

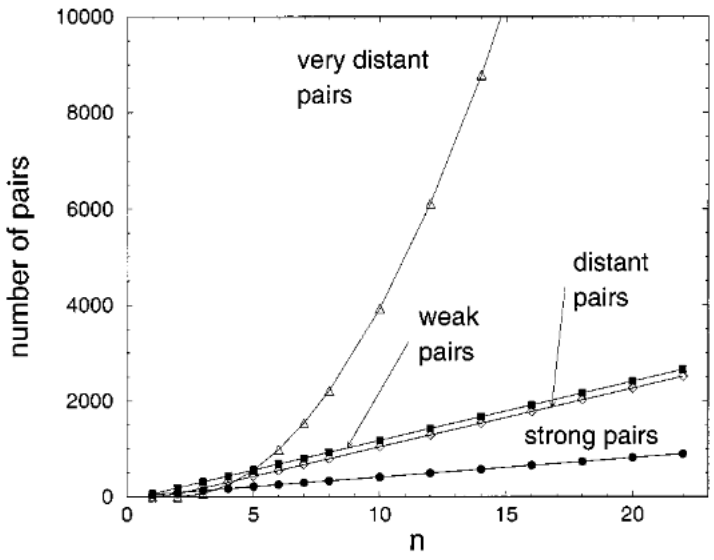


Figure 4.1: Number of significant electron pairs in glycine chains. Taken from [3]

Density Fitting for LMP2

While specific virtual orbitals form a very compact representation of the virtual space, the fact that each electron pair has their own orthogonal virtual orbital basis means that the total number of virtuals can become exceedingly large, and consequently increases the cost associated with the AO-MO transformation step. The most expensive step then becomes

$$(\underline{i}\overline{a} \mid P) = L_{\mu\underline{i}} \, (\mu\nu \mid P) \, L_{\nu\overline{a}} \tag{4.74}$$

Transformation of the 3c2e integrals scales with $\mathcal{O}(N^2)$. Linear scaling can be achieved by introducing an orbital-specific fitting domain $[i]_{fit}$, e.g. by assigning all auxiliary functions P on atoms with a Mulliken charge above a given threshold for the local orbital i [181], or by using a Boughton-Pulay like scheme [182]. This yields the sparsity diagram

$$\begin{array}{ccc} P & \overset{\text{S}}{\mu \longleftrightarrow} & \sigma \\ \uparrow & \updownarrow & \updownarrow \\ \text{LDF} \longleftarrow & \underline{i} & \longleftrightarrow \overline{a} \end{array}$$

As opposed to SOS-AO-MP2, where density fitting can give a more favorable factorization of the energy expression, the MO integrals need to be fully assembled for LMP2 in order to solve the linear equations 4.69. The assembly is done in two steps

$$B^X_{\underline{i}\overline{a}} = \sum_{Y \in [i]_{fit} \cup [j]_{fit}} (X \mid Y)^{-1/2} \, (Y \mid \underline{i}\overline{a}) \tag{4.75}$$

$$\left(\underline{i}\overline{a} \mid \underline{j}\overline{b}\right) = \sum_{X \in [i]_{fit} \cup [j]_{fit}} B^X_{\underline{i}\overline{a}} B^X_{\underline{j}\overline{b}} \tag{4.76}$$

The two steps are repeated for each electron pair ij, and the sum runs over all auxiliary functions P in the unified fitting domain $[i]_{fit} \cup [j]_{fit}$, which enforces linear scaling for these steps as well.

4.2.3 Natural Orbitals

Pure natural orbital techniques are generally not used in the context of MP2, given that MP2 is also used as a guess density to obtain the natural orbitals. Rather, NOs are more popular in the context of coupled cluster methods like CCSD and CCSD(T), though there have been some application in the context of PNOs [183, 139] and domain-based local PNOs (DLPNO) [182, 176]. The latter combines the compact virtual representation of PNOs with the local electron-pair treatment of LMOs described in the previous sections.

As was previously explained, the main idea of NOs is to truncate the virtual space by omitting all orbitals with an occupation number below a certain threshold. The MO integrals in the truncated NO basis can then be plugged into one of the orbital-invariant formulations of the MP2 energy expressions, or they can be canonicalized to be used in the standard MP2 expressions.

4.3 Coupled Cluster

Virtually all of the concepts introduced in the previous section can also be applied in the context of coupled cluster. Local MP2 and local CC evaluate the MO integrals in the same exact manner, either using LMOs [130, 184, 185, 186, 130], natural orbitals [119, 187, 188] or pair-natural orbitals [134, 189, 190, 137]. The orbital-invariant CC amplitude equations can then be directly solved at a reduced cost by plugging in the truncated MO integrals.

An atomic orbital formulation of the coupled cluster equations is possible [162], but has never been pursued further. Again, the AO-CCSD method as presented by Scuseria et al. is not really a pure AO-method, but rather a PAO-like approach similar to PAO-CCSD later proposed by Christiansen and Koch [111]. The reason why AO/PAO coupled cluster methods have not been popular is likely due to the very high prefactor for larger basis sets, which are often crucial for obtaining accurate correlation energies. Moreover, in contrast to MPn amplitudes, the coupled cluster amplitudes do not have closed expressions, which in turn inhibits any further factorizations of coefficient matrices to reduce the overhead like in Cholesky decomposed MP2, with the exception of hybrid methods like CC2 [191].

4.4 Summary

LMO vs PNO vs NO ?

Chapter 5

Local Correlation Methods (III): Excited State

While local correlation methods for the ground state have been around since the 1980s, the extension of the local treatment of electron correlation to excited states is relatively new. With the earliest attempts dating back to the 2000s, many new approaches and approximations have emerged over the last decade, building on the concepts of LMOs, NOs and PNOs. One of the major obstacles that makes a straight-forward extension to excited states difficult is the long-range character of certain excitations, such as charge transfer states. In contrast to the description of electron correlation in the ground state, the occupied and virtual orbitals involved in electron transitions can be very far apart. This means that the optimal molecular orbital space for the excited state can be very different from that of the ground state.

This chapter presents the state of the art for local correlated excited state methods (ADC, CCLR, EOM-CC) for LMOs, NOs, PNOs, and combinations thereof. Atomic orbital approaches are discussed as well.

5.1 Low-Scaling Correlated Excited State Methods

All of the existing low-scaling implementations of ADC, CCLR and EOM use some form of local or compact molecular orbital representation, to varying degrees of success. As mentioned above, the major problem that these methods face is the non-locality of certain excited states such as charge transfer states, which can involve occupied and virtually orbitals which are localized on entirely different parts in the system. Clearly, truncating virtual orbitals spatially is no longer a valid option, and makes a straight-forward extension of LMO-methods difficult, because they cut out far-away contributions. Similar problems are encountered in NO formulations, as the excited state is often not properly described by the electronic ground state (pair-)densities and their associated (pair) natural representation. Over the years, various strategies have been proposed to adapt existing LMO and NO schemes to excited states as well.

5.1.1 Orbital Invariance of the Matrix Expressions

Correlated excited state methods involve some form of symmetric or non-symmetric eigenvalue problem, which is generally solved using the Davidson procedure. The time-determining step is given by the computation of the matrix-vector product of the ADC, response or EOM matrix with the Davdison trial vectors **u**

$$\mathbf{r} = \mathbf{A}\mathbf{u} \tag{5.1}$$

Closed expressions can be derived for the MVPs, with the matrix elements computed on the fly. The MVPs and trial vectors are divided into blocks of singles, doubles, triples, ... (u_i^a, u_{ij}^{ab}, u_{ijk}^{abc}) depending on the level of approximation of the underlying methods. In the case of ADC(2), the MVP is split according to

$$r_{ia} = A_{ia,jb}u_{jb} + A_{ia,jbkc}u_{jbkc} \tag{5.2}$$

$$r_{iajb} = A_{iajb,kc}u_{kc} + A_{iajb,kcld}u_{kcld} \tag{5.3}$$

The eigenvalue problem is generally solved in the canonical molecular orbital basis, but other orbital representations can also be used, by swapping all the CMO quantities $(ia \mid jb)$, t_{iajb}, ... by their local counterparts $\left(\underline{i}\overline{a} \mid \underline{j}\overline{b}\right)$, $t_{\underline{i}\overline{a}\underline{j}\overline{b}}$, ... The eigenvalue problem can then be solved e.g. in an LMO basis, or the MVPs can be computed in the LMO basis and transformed to the CMO basis using the transformation matrix **U**:

$$r_{ia} = U_{i\underline{i}}r_{\underline{i}\overline{a}}U_{\overline{a}a} \tag{5.4}$$

The eigenvalues of the LMO matrix appear to not differ from the ones obtained via a CMO formalism [**?**].

For local EOM-CC and CCLR, the singles and doubles amplitudes t_{μ_1} and t_{μ_2} are determined iteratively from a local ground state calculations using the techniques in the previous chapter for reduced scaling. For the approximate EOM-CC2 and CC2-LR methods, as well as ADC(2), the MP2 amplitudes may also be computed iteratively in the local basis using the Hylleraas functional, or using a closed form expression via the Laplace transform techniques.

EOM-CC2, CC2-LR and ADC(2) also allow an on-the-fly computation of the doubles part (see section 2.3.4). Here, the orbital invariant formulation becomes less straightforward because the doubles-doubles block of the non-canonical ADC and CC2 Jacobian matrix is not diagonal. Fortunately, the Laplace transform can be applied to circumvent this problem. Using the ADC(2) eigenvalue problem as an example, the doubles-folded MVP expression is given by

$$\begin{aligned} r_{ia}(\omega) &= A_{ia,jb}u_{jb} + A_{ia,jbkc}\frac{A_{iajb,kc}u_{kc}}{\omega - \epsilon_a - \epsilon_b + \epsilon_i + \epsilon_j} \\ &= A_{ia,jb}u_{jb} - \sum_{\alpha}^{n} |w^{(\alpha)}| e^{(\omega - \epsilon_a - \epsilon_b + \epsilon_i + \epsilon_j)tpa} A_{ia,jbkc}A_{,kc}u_{kc} \end{aligned} \tag{5.5}$$

After transforming to the LMO basis, the local ADC(2) equations read

$$r_{\underline{i}\overline{a}}(\omega) = A_{\underline{i}\overline{a},\underline{j}\overline{b}}u_{\underline{j}\overline{b}} - A_{\underline{i}\overline{a},\underline{j}\overline{b}\underline{k}\overline{c}}\sum_{\alpha}^{n} e^{\omega t^{(\alpha)}} X^{(\alpha)}_{\underline{j}\underline{j}'}Y^{(\alpha)}_{\overline{b}\overline{b}'}A_{\underline{j}'\overline{b}'\underline{k}'\overline{c}',\underline{l}\overline{d}}u_{\underline{l}\overline{d}}X^{(\alpha)}_{\underline{k}\underline{k}'}Y^{(\alpha)}_{\overline{c}\overline{c}'} \tag{5.6}$$

whith the Laplace matrices $\mathbf{X}$ and $\mathbf{Y}$

$$X_{\underline{ik}}^{(\alpha)} = \sum_{i} U_{i\underline{i}} |w'^{(\alpha)}|^{1/4} e^{\epsilon_i t'^{(\alpha)}} U_{k\underline{k}} \tag{5.7}$$

$$Y_{\overline{ac}}^{(\alpha)} = \sum_{a} U_{a\overline{a}} |w'^{(\alpha)}|^{1/4} e^{-\epsilon_a t'^{(\alpha)}} U_{c\overline{c}} \tag{5.8}$$

Note that the optimal Laplace parameters $w'^{(\alpha)}$ and $t'^{(\alpha)}$ are different from the ones used in the MP2 amplitudes, due to the presence of the eigenvalue *omega* in the denominator. Each time the eigenvalue changes, the Laplace parameters need to be recomputed to obtain an accurate approximation.

An orbital invariant reformulation is not needed by every method. NOs, PNOs and NTOs can be *canonicalized* (Annex **??**) by diagonalizing the occupied-occupied and virtual-virtual block of the Fock matrix in the truncated NO/PNO/NTO basis to get a smaller set of canonical molecular orbitals and orbital energies. Because these types of representations generally do not depend on distance criteria, they are unaffected by the delocalized nature of the canonical basis, as they seek compactness rather than locality.

5.1.2 Local Molecular Orbitals and Domains

The most challenging part in extending domain-specific virtual orbital methods to excited states lies in determining a suitable excitation domain in which to expand the virtual space. The first implementations of local excited state EOM-CCSD [192, 193] and CC2-LR [**?**] constructed the domains using a Mulliken-charge like analysis of the CIS coeffcients $r_i a$. The CIS coefficients are first transformed to the LMO-PAO basis

$$r_{\underline{i}\overline{\mu}} = U_{ii} r_{ia} \bar{C}_{\mu a} \tag{5.9}$$

To determine the importance w of each LMO/PAO, the squares of the norms of the coefficients are summed up row- and column-wise

$$\begin{aligned} w_{\underline{i}} &= \sum_{\mu} |r_{\underline{i}\overline{\mu}}|^2 \\ w_{\overline{\mu}} &= \sum_{\underline{i}} |r_{\underline{i}\overline{\mu}}|^2 \end{aligned} \tag{5.10}$$

The LMOs/PAOs are then ordered by decreasing weight. Their weights are then summed up until a certain threshold T_{LMO}/T_{PAO} is reached (typically around 0.995 to 0.9999). The excited state orbital domains $[\underline{i}]_{ES}$ containing the relevant virtual orbitals are then constructed by applying the Boughton-Pulay algorithm to a set of "excited natural orbitals" [192]

$$\phi_{\underline{i}}^{*} = \sum_{\overline{a}} r_{\underline{i}\overline{\mu}} \phi_{\overline{\mu}} \tag{5.11}$$

The full orbital domain of i is then given as the union of its ground-state and excited state domain $[\underline{i}] = [i]_{GS} \cup [i]_{ES}$. The virtual orbital weights $w_{\overline{\mu}}$ can be used to impose further restrictions on the virtual orbital space. Finally, the pair domains ij are formed as the union $[\underline{i}] \cup [\underline{j}]$. In general, only the computation of the doubles part, which is

time-determining, is subject to domain-restrictions, while the singles part is computed without domain lists.

The method however has the major flaw that the orbital domains are highly sensitive to the CIS transition density, which does not describe the excited state very accurately. Some orbitals can be dropped in the domain construction which might become important for doubles contributions. LMO methods face an interesting chicken-or-egg problem where they need information from the excited state wave function, to accurately compute properties of said function. There are several ways to address this problem. In their local CC2-LR implementation, Kats and Schütz [194] use Laplace transformed doubles folding to recompute the doubles amplitudes on the fly, which allows to adapt the excited state domains dynamically during the optimization procedure. Starting from the CIS transition density, the domains are recomputed at each step by analyzing the state vector $r_{i\bar{\mu}}(\omega)$ as described above. This greatly increased accuracy compared to canonical calculations with energy differences well below 0.1 eV.

Mester et al. [195] proposed a more pragmatic approach, where they first analyze the CIS state vector to extract the important LMOs and PAOs. They then augment the domains $[i]_{EX}$ by adding all remaining molecular orbitals that have a significant Mulliken charge on an atom that is also significant for i. This is based on the assumption that, although CIS might not be a good approximation, the important orbitals should still be close by.

Nonetheless, the LMO method is again plagued by spurious distance dependent thresholds and Mulliken charge thresholds. Nowadays, low scaling excited state methods are mostly dominated by PNOs, NOs, or NTOs.

5.1.3 Natural Orbitals

NO methods achieve performance by dropping virtual natural orbitals with low occupation numbers. The first implementations of EOM-CC and CCLR in the NO representation used natural orbitals obtained from the diagonalization of the ground state MP2 density matrix [196]. A reasonable speed-up could be observed, although the excited state character was not taken into account. However, it was shown [197] that properties like the polarizability are much more sensitive to the truncation of the virtual orbitals than the ground state correlation energy, with the error increasing linearly as a function of the number of dropped virtual natural orbitals. While VNOs with low occupation numbers, i.e. diffuse character, can be safely ignored for the ground state correlation energy, diffuse VNOs play a much more important role for response properties, and hence fewer VNOs can be omitted. Better results could be obtained by simply truncating the virtual CMOs instead, which invalidates the use of VNOs.

In their NO-CC2 and NO-ADC(2) implementations, Mester et al. [195, 198, 4] proposed to compute a set of occupied and virtual NOs by diagonalizing the occupied and virtual state-averaged densities

$$\mathbf{D}_{ij} = \frac{1}{2}\left(\mathbf{D}_{ij}^{MP2} + \mathbf{D}_{ij}^{CIS(D)}\right) \tag{5.12}$$

$$\mathbf{D}_{ab} = \frac{1}{2}\left(\mathbf{D}_{ab}^{MP2} + \mathbf{D}_{ab}^{CIS(D)}\right) \tag{5.13}$$

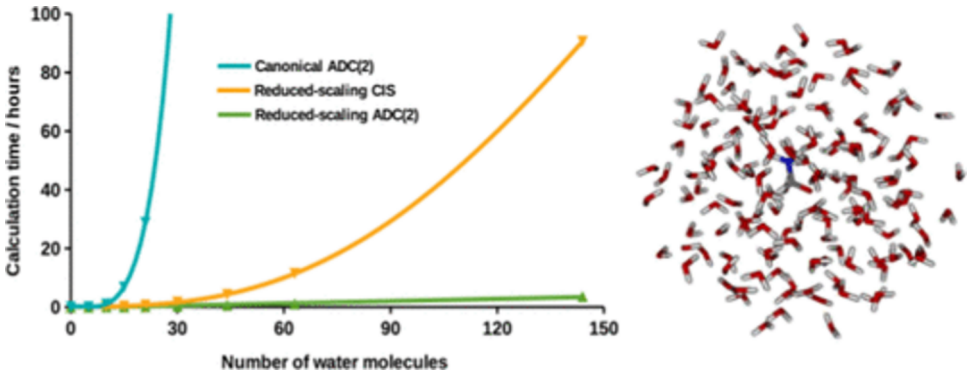


Figure 5.1: Wall times of CIS compared to NO-ADC(2) as a function of system size of hydrated formamide. Taken from [4].

where $\mathbf{D}^{MP2}$ is the MP2 ground state density and $\mathbf{D}^{CIS}$ is the state-specific CIS(D) excited state density. Their restricted expressions read

$$D_{ij}^{MP2} = \sum_{kab} \left(2t_{ik}^{ab} t_{jk}^{ab} - t_{ik}^{ab} t_{jk}^{ab}\right) \tag{5.14}$$

$$D_{ab}^{MP2} = \sum_{ijc} \left(2t_{ij}^{ca} t_{ij}^{cb} - t_{ij}^{ca} t_{ij}^{bc}\right) \tag{5.15}$$

$$D_{ij}^{CIS(D)} = \sum_{a} c_i^a c_j^a + \sum_{kab} \left(2t_{ik}^{ab} t_{jk}^{ab} - t_{ik}^{ab} t_{jk}^{ab}\right) \tag{5.16}$$

$$D_{ab}^{CIS(D)} = \sum_{i} c_i^a c_i^b + \sum_{ijc} \left(2c_{ij}^{ca} c_{ij}^{cb} - c_{ij}^{ca} c_{ij}^{bc}\right) \tag{5.17}$$

where c_i^a are the CIS coefficients and the c_{ij}^{ab} are the CIS(D) doubles coefficients

$$c_{ij}^{ab} = \frac{\sum_c \left[(ac \mid bj)\, c_i^c + (ac \mid bi)\, c_j^c\right] - \sum_k \left[(kj \mid ai)\, c_k^b + (kj \mid bj)\, c_k^a\right]}{D_{ij}^{ab} + \omega_{CIS}} \tag{5.18}$$

The state-averaged density needs to be recomputed and diagonalized for each state because $\mathbf{D}^{CIS(D)}$ depends on the excitation energy ω. While the CIS(D) density is much easier to compute than the ADC(2) or CC2-LR state density, it still scales with $\mathcal{O}(N^5)$. To reduce the computational complexity, the density is constructed in a truncated orbital space: first, a set of occupied and virtual LMOs are chosen according to the CIS weighting criteria w described in the previous section. The basis is augmented by spatially close orbitals, and then canonicalized to yield a highly compact orbital molecular space which lowers the cost of constructing the CIS(D) densities.

In combination with natural auxiliary functions, this hybrid NO-LMO scheme can reduce the timings for CC2 and ADC(2) to such a drastic extent that the CIS pre-iterations become the time-determining step (Figure 5.1), with an additional error of only 2-4 meV. The reduced scaling however comes at a high prefactor when computing multiple different excitation energies.

5.1.4 Pair Natural Orbitals

Pair natural orbital methods face the same problems as NOs, where PNOs with low occupation numbers are considerably more important for response properties than for ground state properties [**?**]. In a similar vein, excited state PNOs can be generated by considering lower level excited state electron pair densities [199]. PNO methods have been successfully extended to ADC(2), CC2-LR [200], ADC(2)-x [201] and CCSD-LR [202] by using CIS(D) or CIS(D)-like densities

$$D_{ij}^{ab} = \sum_{c} \left(2b_{ij}^{ab} - b_{ij}^{ba}\right) b_{ij}^{ab} + \left(2b_{ij}^{ab} - b_{ij}^{ba}\right) b_{ij}^{ba} \tag{5.19}$$

where $\mathbf{b}_{ij}$ are state-specific modified pair amplitudes which are not uniquely defined. Again, these methods come at the cost of a higher prefactor due to the relatively high cost of constructing PNOs.

Efforts have also been made to develop PNO response methods which are more economical for computing larger excitation manifolds by removing the state-specificity. Instead of taking individual excited state densities, Peng et al. [**?**] proposed to generate a set of *state-averaged* PNOs obtained by diagonalization of the average excited state density over an N-state manifold

$$\mathbf{D}_{ij} = \frac{1}{N} \sum_{k}^{N} \mathbf{D}_{ij}^{(k)} \tag{5.20}$$

A production-quality implementation has not yet been shown which uses this approach.

In their perturbed pair-natural orbital (PNO++) approach for CCLR, Cunha and Crawford [**?**] incorporate the external perturbation into the electron pair density

$$D_{ij}^{ab} = \sum_{c} \left(2x_{ij}^{ab} - x_{ij}^{ba}\right) x_{ij}^{ab} + \left(2x_{ij}^{ab} - x_{ij}^{ba}\right) x_{ij}^{ba} \tag{5.21}$$

where $\mathbf{x}$ are perturbed amplitudes given by

$$x_{ij}^{ab} = \frac{\overline{B}}{\overline{H}_{aa} + \overline{H}_{bb} - \overline{H}_{ii} - \overline{H}_{jj} + \omega} \tag{5.22}$$

with an external perturbation $\overline{B}$ and the similarity transformed Hamiltonian $\overline{H}$. This gives a set of "perturbation-aware" PNOs customized for a given external perturbation [203].

Finally, there are also the *back-transformed* PNOs, or bt-PNOs, where the ground state PNO-quantities like the amplitudes are transformed back to the canonical basis and used in the canonical working equations [204].

In the end, most local excited state methods using natural orbitals differ by how they redefine the amplitudes $\mathbf{b}$ for the individual excited states or the whole perturbed molecular system. It is still an active field of research.

5.1.5 Natural Transition Orbitals

The last method to obtain a compact representation of excited states is via natural transition orbitals. NTOs are the equivalent of NOs for excited states, and represent

a compact representations of their dominant contribution (Figure 5.2). Baudin and Kristensen have developped two different CC2LR schemes based on NTOs called LoFEX (local framework for calculating excitation energies) [205] and CornFLEX (correlated natural transition orbital framework for calculating excitation energies) [122]

Again, one needs information about the excited state to efficiently compute its properties. The LoFEX method starts with a time-dependent Hartree Fock calculation and generates a set of NTOs by decomposition of the TDHF transition vectors $\mathbf{r}$ by diagonalization

$$\mathbf{r}\mathbf{r}^\dagger\mathbf{U} = \lambda_o\mathbf{U} \tag{5.23}$$

$$\mathbf{r}^\dagger\mathbf{r}\mathbf{V} = \lambda_v\mathbf{V} \tag{5.24}$$

which is just an alternative way to compute the occupied and virtual NTO transformation matrices $\mathbf{U}$ and $\mathbf{V}$, rather than by singular value decomposition. A set of dominant NTO pairs is then chosen for which their occupation numbers are above a given threshold τ_{LoFEX}. The non-dominant NTOs are not discarded, but rather localized. The idea is to construct a surrounding excitation orbital space (XOS) containing LMOs that are important for correlation effects of the NTOs. A first guess to the XOS is chosen based on distance criteria and Löwdin charges. The CC2-LR eigenvalue problem is then solved in that basis, and new NTOs are computed from the CC2 transition vector and added to the XOS. This procedure is repeated until the excitation energy ω for that state has converged. While the guess XOS is first formed using distance criteria, the subsequent optimization procedure makes the method much more robust and black-box. Even for relatively small molecules, LoFEX can obtain considerable speed-ups. The main disadvantage is that LoFEX does not give any leverage for very delocalized excitations.

The improved CornFLEX method constructs a set of CIS(D)-like NTOs (CIS(D')-NTOs) which is obtained from diagonalizing a CIS(D)-like density matrix in the CIS-NTO basis. As opposed to CIS-NTOs, the CIS(D') NTOs also include correlation effects and are a more robust representation that the simple ad-hoc extension of CIS-NTOs using LMOs. Speed-ups can be observed in CornFLEX even for delocalized excitations.

5.2 Atomic Orbital Configuration Interaction Singles

The methods presented in the previous section all work similarly. They first start by approximating the targeted excited states with a lower level of theory using CIS or CIS(D). They then solve higher order equations in the basis obtained from that approximation and may also dynamically augment the correlation domain while optimizing the excitation energies. The methods work on the principle of orbital *compactness* rather than sparsity

At the moment of writing, CIS is the only excited state method which is routinely evaluated using an AO approach. While CIS does not give qualitatively good results, it is still a very important stepping stone for higher order methods, as was demonstrated in the previous section. Omitting the zero-order contributions, the CIS working equations are given by

$$u_{ia} = \left[2\,(ia \mid jb) - (ib \mid ja)\right] u_{jb} \tag{5.25}$$

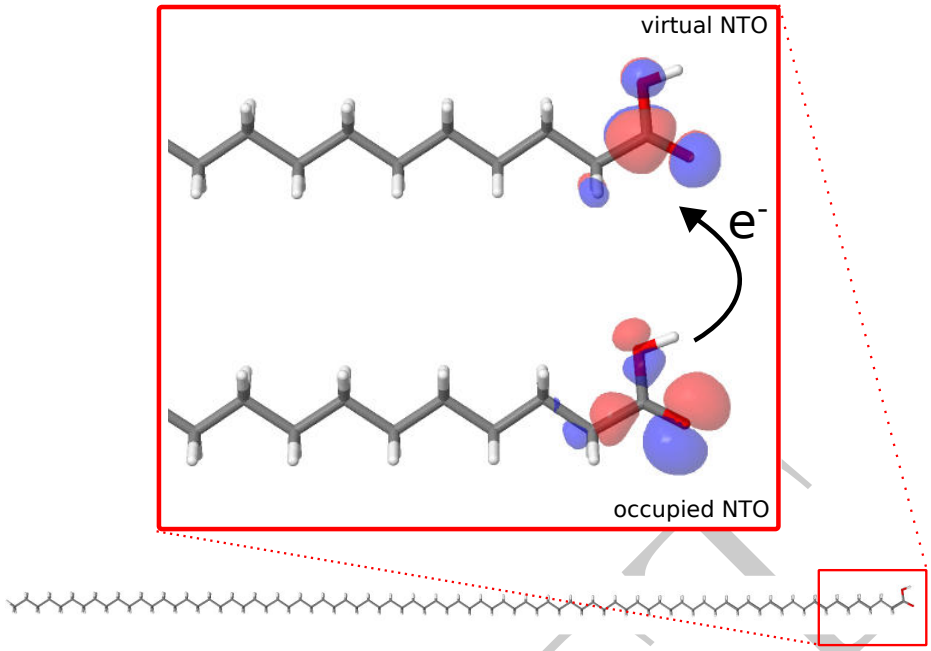


Figure 5.2: Dominant natural transition orbital pair for the lowest excitation of the carboxylic acid $C_{79}H_{159}COOH$ ($\pi \rightarrow \pi^*$ transition). The span of the NTOs is very small compared to the rest of the molecule, and the compactness can be used to drastically speed up excited state caculations.

Factoring out the MO coefficient matrices:

$$\begin{aligned} u_{ia} &= C_{\mu i} C_{\sigma a} \left[2\left(\mu\sigma \mid \nu\lambda\right) - \left(\mu\lambda \mid \nu\sigma\right) \right] C_{\nu j} C_{\lambda b} u_{jb} \\ &= C_{\mu i} C_{\sigma a} \left[2\left(\mu\sigma \mid \nu\lambda\right) - \left(\mu\lambda \mid \nu\sigma\right) \right] P_{\nu\lambda} \end{aligned} \tag{5.26}$$

where $\mathbf{P}$ is the non-symmetric transition density in the AO basis. The CIS working equations can be reduced to the construction of a "pseudo"-Fock matrix which has a coulomb and an exchange part. The Fock matrix is then transformed to the MO basis:

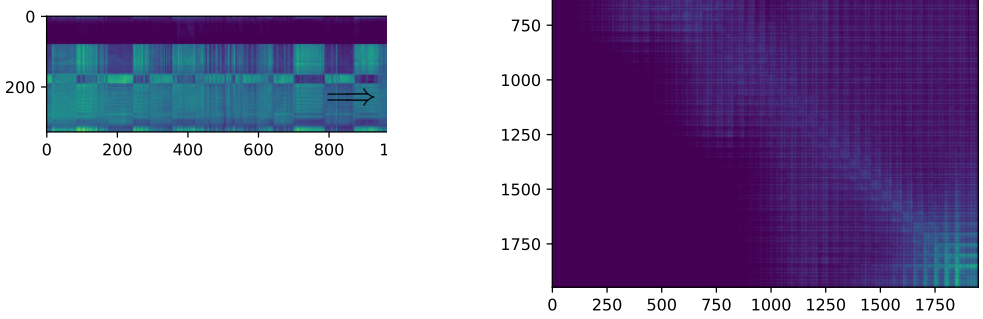
$$F_{\mu\nu} = J_{\mu\nu} + K_{\mu\nu} \tag{5.27}$$

$$u_{ia} = C_{\mu i} F_{\mu\nu} C_{\mu a} \tag{5.28}$$

For localized excitations, the AO transition density is sparse (Figure 5.3), and similar approximation can be used as in Hartree Fock, e.g. LinK, CFMM, or LDF. CIS can therefore be evaluated with $\mathcal{O}(N)$ computational effort. Strictly speaking, it is not a "pure" AO formulation, because the AO intermediates still need to be transformed to the MO basis.

5.3 Molecular Orbital-free Approaches

Maybe if I have time



(a) (b)

Figure 5.3: Logarithm of the absolute values of the matrix elements in the transition densities in the MO (left) and AO basis (right) for the lowest excited state for the carboxylic acid $C_{79}H_{159}COOH$. The excitation domain is entirely localized on the carboxylic group. Using sparse matrix algebra, significant speed-ups can be obtained for CIS in the AO basis

Chapter 6

The Spin-Opposite Scaled Algebraic Diagrammatic Construction Method in the Atomic Orbital Basis

The Algebraic Diagrammatic Construction method can be considered as Møller Plesset for excited states. It is therefore not surprising that local correlation method for MP can also be applied to MP. In chapter 4, it has been shown that local approximations for the ground state can be grouped into 3 categories: atomic orbitals, local orbitals and natural orbitals. Only the latter two have been used in the context of ADC as discussed in chapter 6. An atomic orbital representation of ADC has not yet been considered in literature, and will be the subject of this chapter. First, the restricted doubles-folded ADC working equations are derived. Then the SOS approximation is applied. Finally, the restricted SOS-ADC working equations are derived in the AO basis, with and without density fitting.

6.1 Working Equations for Restricted ADC with Doubles-Folding

The eigenvalue problem in the algebraic diagrammatic construction method truncated at doubles excitations takes the form

$$\begin{bmatrix} \mathbf{A}_{SS} & \mathbf{A}_{SD} \\ \mathbf{A}_{DS} & \mathbf{A}_{DD} \end{bmatrix} = \begin{bmatrix} \mathbf{v}_S \\ \mathbf{v}_D \end{bmatrix} \boldsymbol{\Omega} \tag{6.1}$$

where $\mathbf{A}$ is the symmetric ADC matrix with the singles-singles (SS), doubles-singles (DS), singles-doubles (SD) and doubles-doubles (DD) sub-blocks, with the eigenvectors $\mathbf{v}$ and the diagonal eigenvalue matrix Ω. The eigenvalue problem is generally solved using the Davidson diagonalization procedure to extract the first few lowest eigenvalues. Rather than constructing the entire Jacobian matrix which scales as $\mathcal{O}(N^8)$, the Davidson method computes the matrix-vector products $\mathbf{r} = \mathbf{A}\mathbf{u}$ with the current trial vectors $\mathbf{u}$ using a closed-form expression, which reduces the computational complexity to $\mathcal{O}(N^5)$.

The MVPs can be expressed in block-form as

$$\begin{aligned} \mathbf{r}_S &= \mathbf{A}_{SS}\mathbf{u}_S + \mathbf{A}_{SD}\mathbf{u}_D \\ \mathbf{r}_D &= \mathbf{A}_{DS}\mathbf{u}_S + \mathbf{A}_{DD}\mathbf{u}_D \end{aligned} \tag{6.2}$$

The trial vector space in the Davidson diagonalization scales with fourth order as o^2v^2, and can quickly become a memory bottle-neck for large molecules. As shown in chapter 2, one can recompute the doubles-part of the MVP on-the-fly using *doubles-folding*

$$\mathbf{r}_S(\omega) = \mathbf{A}_{SS}\mathbf{u}_S + \frac{\mathbf{A}_{SD}}{\omega - \mathbf{DD}}\mathbf{u}_S \tag{6.3}$$

This trick is only possible for ADC(2)-s where the doubles-doubles block of the ADC matrix is diagonal. While the memory footprint for the diagonalization is reduced from o^2v^2 to ov, the MVP becomes dependent on the eigenvalue ω and a modified Davidson procedure needs to be used to solve this *pseudo*-eigenvalue problem. The working equations for the folded ADC(2) matrix-vector product is given by [206]

$$\begin{aligned} r_{ia} = &\sum_b u_{ib} f_{ab} - \sum_j f_{ij} u_{jb} + \sum_{jb} \langle ja| \, |ib\rangle \, u_{jb} + \left[\sum_b I^{(1)}_{ab} u_{ib} + \sum_j I^{(2)}_{ij} u_{ja} \right] \\ &- \frac{1}{2} \left[\sum_{jb} t_{ijab} I^{(3)}_{jb} - \sum_{jb} \langle ij| \, |ab\rangle \, I^{(4)}_{jb} \right] \\ &+ \left[\sum_{jkb} \langle jk| \, |ib\rangle \, u_{jkab}(\omega) + \sum_{jbc} u_{ijbc}(\omega) \, |ja\rangle \, \langle bc| \right] \end{aligned} \tag{6.4}$$

where the doubles part is computed as

$$u_{iajb}(\omega) = \frac{1}{2} \frac{[sum_k \langle ij| \, |kb\rangle \, u_{ka} - \langle ij| \, kau_{kb}] + \sum_c u_{ic} \langle jc| \, |ab\rangle - u_{jc} \langle ic| \, |ab\rangle}{\omega + \epsilon_i + \epsilon_j - \epsilon_a - \epsilon_b} \tag{6.5}$$

and with the intermediates $I^{(1)}$, $I^{(2)}$, $I^{(3)}$ and $I^{(4)}$ as given in [206]. Until now, only spin molecular orbitals were assumed. Implementations such as adcman in Q-Chem [207] can use these formulae directly by delegating any considerations of spin-symmetry to a special tensor library called libtensor [208], which programmatically keeps track of the non-vanishing spin block components and reduces the expressions to the restricted ADC(2) equations for closed-shell molecules.

If no special block tensor library is used, it is numerically advantageous to split the ADC(2) matrix-vector products into their spin-components, and compute each block individually. Using a double-bar notation to indicate MOs with opposite spin $\sigma(i) \neq \sigma(\bar{i})$,

the matrix-vector product can be written as

$$\begin{aligned} r_{ia}(\omega) =&(\epsilon_a - \epsilon_i)u_{ia} - \sum_{jb} \left[(ij \mid ab) - (ia \mid jb)\right] u_{jb} + \sum_{\overline{jb}} \left(ia \mid \overline{jb}\right) u_{\overline{jb}} \\ &+ \sum_b I_{ab} u_{ib} + \sum_j I_{ij} u_{ja} - \frac{1}{2}\left[t_{ia\overline{jb}} I^{(1)}_{\overline{jb}} + \left(ia \mid \overline{jb}\right) I^{(2)}_{\overline{jb}}\right] \\ &- \frac{1}{2}\left[\left(t_{iajb} - t_{jaib}\right) I^{(}1)_{jb} + \left((ia \mid jb) - (ja \mid ib)\right) I^{(2)}_{jb}\right] \\ &+ \sum_{kcl} u_{kalc}(\omega)\,(ik \mid cl) + \sum_{k\overline{cl}} u_{ka\overline{lc}}(\omega)\left(ik \mid \overline{cl}\right) \\ &- \sum_{ckd} (ac \mid kd)\, u_{ikcd}(\omega) - \sum_{\overline{ckd}} \left(ac \mid \overline{kd}\right) u_{ic\overline{kd}}(\omega) \end{aligned} \tag{6.6}$$

with the pre-iteration intermediates (computed only once)

$$\begin{aligned} I_{ab} = &\frac{1}{2}\sum_{kcl} \left[t_{kalc}\,(kb \mid lc) - t_{kalc}\,(kc \mid lb) + (ak \mid cl)\, t_{kblc} - (al \mid ck)\, t_{kb}lc\right] \\ &+ \frac{1}{2}\sum_{k\overline{cl}} \left[t_{ka\overline{lc}}\left(kb \mid \overline{lc}\right) + \left(ak \mid \overline{cl}\right) t_{kb}\overline{lc}\right] \end{aligned} \tag{6.7}$$

$$\begin{aligned} I_{ij} = &\frac{1}{2}\sum_{ckd} \left[t_{ickd}\,(jc \mid kd) - t_{ickd}\,(jd \mid kc) + (ci \mid dk)\, t_{jckd} - (ck \mid di)\, t_{jckd}\right] \\ &+ \frac{1}{2}\sum_{c\overline{kd}} \left[t_{ic\overline{kd}}\left(jc \mid \overline{kd}\right) + \left(ci \mid \overline{dk}\right) t_{jc\overline{kd}}\right] \end{aligned} \tag{6.8}$$

and the iteration intermediates which depend on the trial vectors $\mathbf{u}$ (computed at each Davidson iteration)

$$I^{(1)}_{ia} = \sum_{jb} \left[(jb \mid ia) - (ja \mid ib)\right] u_{jb} + \sum_{\overline{jb}} \left(\overline{jb} \mid ia\right) u_{jb} \tag{6.9}$$

$$I^{(2)}_{ia} = \sum_{jb} \left[t_{iajb} - t_{jaib}\right] u_{jb} + \sum_{\overline{jb}} t_{ia\overline{jb}} u_{\overline{jb}} \tag{6.10}$$

The doubles spin components are computed on-the-fly and read

$$\begin{aligned} u_{iajb}(\omega) = \frac{1}{\omega - \epsilon_a - \epsilon_b + \epsilon_i + \epsilon_j} \Bigg\{ &\sum_k \left[u_{ka}\,(ki \mid bj) - u_{ka}\,(kj \mid bi) - u_{kb}\,(ki \mid aj)\right. \\ &\left. +\; u_{kb}\,(kj \mid ai)\right] - \sum_c \left[u_{ic}\,(ac \mid bj) - u_{ic}\,(aj \mid bc) - u_{jc}\,(ac \mid bi) + u_{jc}\,(bc \mid ai)\right] \Bigg\} \end{aligned} \tag{6.11}$$

$$\begin{aligned} u_{ia\overline{jb}}(\omega) = \frac{1}{\omega - \epsilon_a - \epsilon_b + \epsilon_i + \epsilon_j} \Bigg\{ &\sum_k u_{ka}\left(ki \mid \overline{bj}\right) + \sum_{\overline{k}} u_{\overline{kb}}\left(\overline{kj} \mid ai\right) \\ &- \sum_c u_i c\left(ac \mid \overline{bj}\right) - \sum_{\overline{c}} u_{\overline{jc}}\left(\overline{bc} \mid ai\right) \Bigg\} \end{aligned} \tag{6.12}$$

The expression for the MVP for beta electrons ($r_{\overline{ia}}$) is obtained by replacing alpha orbitals (i) by beta orbitals ($\overline{i}$) and vice-versa in the expressions above. The off-diagonal blocks $r_{i\overline{a}}$, i.e. the spins-flip states will not be considered here and are set to zero. For closed-shell molecules, the complexity of the formulas can be drastically reduced by introducing the following spin-symmetry relationships:

$$t_{iajb} = t_{ia\overline{jb}} = t_{\overline{ia}jb} = t_{\overline{iajb}} \tag{6.13}$$

$$(ia \mid jb) = \left(ia \mid \overline{jb}\right) = \left(\overline{ia} \mid jb\right) = \left(\overline{ia} \mid \overline{jb}\right) \tag{6.14}$$

$$u_{ia} = u_{\overline{ia}} \qquad \text{if singlet} \tag{6.15}$$

$$u_{ia} = -u_{\overline{ia}} \qquad \text{if triplet} \tag{6.16}$$

One then obtains two separate expressions for restricted ADC(2), depending on whether singlet or triplet states are addressed

$$\begin{aligned} r^S_{ia}(\omega) = {} & (\epsilon_a - \epsilon_i)u_{ia} - \sum_{jb} \left[2\,(ia \mid jb) - (ij \mid ab)\right] u^S_{jb} + \sum_b I_{ab}u_{ib} + \sum_j u^S_{ja}I_{ij} \\ & - \frac{1}{2}\sum_{jb} \left[2t_{iajb} - t_{ibja}\right] I^{(1)S}_{jb} - \frac{1}{2} sum_{jb} \left[2\,(ia \mid jb) - (ib \mid ja)\right] I^{(2)S}_{jb} \\ & + \sum_{kcl} (ik \mid lc) \left(2u^S_{kalc}(\omega) - u^S_{lakc}(\omega)\right) + \sum_{ckd} \left(2u^S_{ickd}(\omega) - u^S_{kcid}(\omega)\right) (kd \mid ac) \end{aligned} \tag{6.17}$$

$$\begin{aligned} r^T_{ia}(\omega) = {} & (\epsilon_a - \epsilon_i)u^T_{ia} - \sum_{jb} (ij \mid ab)\, u^T_{jb} + \sum_b I_{ab}u^T_{ib} + \sum_j I_{ij}u^T_{ja} \\ & + \frac{1}{2}\sum_{jb} t_{ibja} I^{(1)T}_{jb} + \frac{1}{2}\sum_{jb} (ib \mid ja)\, I^{(2)T}_{jb} \\ & + \sum_{kcl} (ik \mid lc)\, u^T_{kalc}(\omega) + \sum_{ckd} \left[2u^T_{ickd}(\omega) - u^T_{idkc}(\omega) - u^T_{kcid}(\omega)\right] \end{aligned} \tag{6.18}$$

with the singlet and triplet doubles intermediates

$$\begin{aligned} u^{S,SOS}_{iajb}(\omega) = {} & \frac{c_{osc}}{\omega - \epsilon_a + \epsilon_i - \epsilon_b + \epsilon_j} \left\{ \sum_k \left[u^S_{ka}\,(ki \mid jb) + u^S_{kb}\,(kj \mid ai)\right] \right. \\ & \left. - \sum_c \left[u^S_{ic}\,(jb \mid ac) - u^S_{jc}\,(ib \mid ac)\right] \right\} \end{aligned} \tag{6.19}$$

$$u^T_{iajb}(\omega) = \frac{1}{\omega - \epsilon_a + \epsilon_i - \epsilon_b + \epsilon_j} \left\{ \sum_k u^T_{ka}\,(ki \mid bj) - \sum_c u^T_{ic}\,(ac \mid jb) \right\} \tag{6.20}$$

The pre-iteration intermediates are given by

$$\begin{aligned} I_{ab} &= \frac{1}{2}\sum_{kcl} \left[(2t_{kalc} - t_{kcla})\,(kb \mid lc) + (2t_{kblc} - t_{kclb})\,(ka \mid lc)\right] \\ &= \frac{1}{2}\left[\sum_{kcl} (2t_{kalc} - t_{kcla})\,(kb \mid lc)\right]_{a \leftrightarrow b} \end{aligned} \tag{6.21}$$

$$\begin{aligned} I_{ij} &= \frac{1}{2} \sum_{ckd} \left[\left(2t_{ickd} - t_{idkc}\right) (jc \mid kd) + \left(2t_{jckd} - t_{jdkc}\right) (ic \mid kd) \right] \\ &= \frac{1}{2} \left[\sum_{ckd} \left(2t_{ickd} - t_{idkc}\right) (jc \mid kd) \right]_{i \leftrightarrow j} \end{aligned} \tag{6.22}$$

and are the same for both singlet and triplet expressions. The iterative intermediates however are split:

$$I_{ia}^{(1)S} = \sum_{jb} \left(2 \left(ia \mid jb\right) - \left(ib \mid ja\right)\right) u_{jb}^{S} \tag{6.23}$$

$$I_{ia}^{(2)S} = \sum_{jb} \left(2t_{iajb} - t_{ibja}\right) u_{jb}^{S} \tag{6.24}$$

$$I_{ia}^{(1)T} = -\sum_{jb} \left(ib \mid ja\right) u_{jb}^{T} \tag{6.25}$$

$$I_{ia}^{(2)T} = -\sum_{jb} t_{ibja} u_{jb}^{T} \tag{6.26}$$

6.2 Working Equations for Restricted SOS-ADC(2) with Doubles-Folding

In the restricted ADC expressions for the matrix-vector product, the 4-index intermediates $u_{iajb}(\omega)$ need to be evaluated and temporarily stored, even if the density fitting approximation is used. This memory-intensive step can be avoided by using spin-opposite scaling (Section 2.3.6). Consider again the approximations introduced by the SOS method for the unrestricted ADC(2) matrix equations

- In the spin-amplitudes $\hat{t}_{iajb}$ the same-spin contributions are nulled and the opposite-spin contributions are scales by $c_o s$

$$\hat{t}_{SOS} = c_{os} \hat{t}_{iajb} \left(1 - \delta_{\sigma(i)\sigma(j)}\right) \tag{6.27}$$

- In the doubles-singles and singles-doubles block of the ADC matrix, some same-spin contributions are also removed, and the whole block is scaled by a different constant c_{osc}

$$C_{ia,kcld} = c_{osc} \left[\langle kl| \, |id\rangle \, \delta_{ac} - \langle kl| \, |ic\rangle \, \delta_{ac} - \langle al| \, |cd\rangle \, \delta_{ik} + \langle ak| \, |cd\rangle \, \delta_{il} \right] \left(1 - \delta_{\sigma(k)\sigma(l)}\right) \tag{6.28}$$

$$C_{iajb,kc} = c_{osc} \left[\langle kb| \, |ij\rangle \, \delta_{ac} - \langle ka| \, |ij\rangle \, \delta_{bc} - \langle ab| \, |cj\rangle \, \delta_{ik} + \langle ab| \, |ci\rangle \, \delta_{jk} \right] \left(1 - \delta_{\sigma(i)\sigma(j)}\right) \tag{6.29}$$

Here, the function $\sigma(x)$ returns the spin of orbital x.

Applying the above constraints to the MVP expressions 6.17 (shown for singles only) gives the spin components of the SOS-ADC(2) matrix-vector product:

$$\begin{aligned} r_{ia}^{SOS}(\omega) =&(\epsilon_a - \epsilon_i)u_{ia} - \sum_{jb}\left[(ij \mid ab) - (ia \mid jb)\right] u_{jb} + \sum_{\overline{jb}} \left(ia \mid \overline{jb}\right) u_{\overline{jb}} \\ &+ \sum_{b} I_{ab}^{SOS} u_{ib} + \sum_{j} I_{ij}^{SOS} u_{ja} - \frac{1}{2}\left[t_{ia\overline{jb}} I_{\overline{jb}}^{(1)SOS} + \left(ia \mid \overline{jb}\right) I_{\overline{jb}}^{(2)SOS}\right] \\ &- \frac{1}{2}\left[(ia \mid jb) - (ja \mid ib)\right] I_{jb}^{(2)SOS} \\ &+ c_{osc}\left\{\sum_{k\overline{cl}} u_{ka\overline{lc}}(\omega)\left(ik \mid \overline{cl}\right) - \sum_{c\overline{kd}}\left(ac \mid \overline{kd}\right) u_{ic\overline{kd}}(\omega)\right\} \end{aligned} \tag{6.30}$$

with the SOS pre-iteration intermediates

$$I_{ab}^{SOS} = \frac{1}{2}\sum_{k\overline{cl}}\left[t_{ka\overline{lc}}\left(kb \mid \overline{lc}\right) + \left(ak \mid \overline{cl}\right) t_{kb\overline{lc}}\right] \tag{6.31}$$

$$I_{ij}^{SOS} = \frac{1}{2}\sum_{c\overline{kd}}\left[t_{ic\overline{kd}}\left(jc \mid \overline{kd}\right) + \left(ci \mid \overline{dk}\right) t_{jc\overline{kd}}\right] \tag{6.32}$$

and the singlet SOS iteration intermediates

$$I_{ia}^{(1)SOS} = \sum_{jb}\left[(jb \mid ia) - (ja \mid ib)\right] u_{jb} + \sum_{\overline{jb}}\left(\overline{jb} \mid ia\right) u_{\overline{jb}} \tag{6.33}$$

$$I_{ia}^{(2)SOS} = \sum_{\overline{jb}} t_{ia\overline{jb}} u_{\overline{jb}} \tag{6.34}$$

Only the opposite-spin components of the doubles components are needed:

$$\begin{aligned} u_{ia\overline{jb}}^{SOS}(\omega) = \frac{c_{osc}}{\omega - \epsilon_a - \epsilon_b + \epsilon_i + \epsilon_j}&\left\{\sum_{k} u_{ka}\left(ki \mid \overline{bj}\right) + \sum_{\overline{k}} u_{\overline{kb}}\left(\overline{kj} \mid ai\right)\right. \\ &\left. - \sum_{c} u_{i}c\left(ac \mid \overline{bj}\right) - \sum_{\overline{c}} u_{\overline{jc}}\left(\overline{bc} \mid ai\right)\right\} \end{aligned} \tag{6.35}$$

Finally, for closed-shell molecules, the matrix vector products for singlet and triplet excitations for restricted SOS-ADC(2) can be obtained by inserting the spin-symmetry relationships 6.13 to 6.16 into Equation 6.30, and performing a similar manipulation for the triplet expression:

$$\begin{aligned} r_{ia}^{S,SOS}(\omega) = (\epsilon_a - \epsilon_i)u_{ia}^{S} - \sum_{jb}\left[2\,(ia \mid jb) - (ij \mid ab)\right] u_{jb}^{S} + \sum_{b} I_{ab}^{SOS} u_{ib} + \sum_{j} u_{ja}^{S} I_{ij}^{SOS} \\ - \frac{1}{2}\sum_{jb} t_{iajb} I_{jb}^{(1)S,SOS} - \frac{1}{2}\sum_{jb}\left[2\,(ia \mid jb) - (ib \mid ja)\right] I_{jb}^{(2)S,SOS} \\ + c_{osc}\left\{\sum_{kcl}(ik \mid lc)\, u_{kalc}^{S,SOS}(\omega) - \sum_{ckd} u_{ickd}^{S,SOS}(\omega)\,(kd \mid ac)\right\} \end{aligned} \tag{6.36}$$

$$\begin{aligned} r_{ia}^{T,SOS}(\omega) = (\epsilon_a - \epsilon_i) u_{ia}^S - \sum_{jb} (ij \mid ab)\, u_{jb}^T + \sum_b I_{ab}^{SOS} u_{ib} + \sum_j u_{ja}^S I_{ij}^{SOS} \\ - \frac{1}{2} \sum_{jb} t_{iajb} I_{jb}^{(1)T,SOS} + \frac{1}{2} \sum_{jb} (ib \mid ja)\, I_{jb}^{(2)T,SOS} \\ + c_{osc} \left\{ \sum_{kcl} (ik \mid lc)\, u_{kalc}^{T,SOS}(\omega) - \sum_{ckd} u_{ickd}^{T,SOS}(\omega)\, (kd \mid ac) \right\} \end{aligned} \tag{6.37}$$

with the on-the-fly doubles

$$\begin{aligned} u_{iajb}^{S,SOS}(\omega) = \frac{c_{osc}}{\omega - \epsilon_a - \epsilon_b + \epsilon_i + \epsilon_j} \left\{ \sum_k u_{ka}\, (ki \mid bj) + \sum_k u_{kb}\, (kj \mid ai) \right. \\ \left. - \sum_c u_{ic}\, (ac \mid bj) - \sum_c u_{jc}\, (bc \mid ai) \right\} \end{aligned} \tag{6.38}$$

$$\begin{aligned} u_{iajb}^{T,SOS}(\omega) = \frac{c_{osc}}{\omega - \epsilon_a - \epsilon_b + \epsilon_i + \epsilon_j} \left\{ \sum_k u_{ka}\, (ki \mid bj) - \sum_k u_{kb}\, (kj \mid ai) \right. \\ \left. - \sum_c u_{ic}\, (ac \mid bj) + \sum_c u_{jc}\, (bc \mid ai) \right\} \end{aligned} \tag{6.39}$$

and the intermediates

$$I_{ab}^{SOS} = \frac{c_o s}{2} \left[\sum_{kcl} t_{kalc}\, (kb \mid lc) \right]_{a \leftrightarrow b} \tag{6.40}$$

$$I_{ij}^{SOS} = \frac{c_o s}{2} \left[\sum_{ckd} t_{ickd}\, (jc \mid kd) \right]_{i \leftrightarrow j} \tag{6.41}$$

$$I_{ia}^{(1)S,SOS} = \sum_{jb} \left(2\, (ia \mid jb) - (ib \mid ja) \right) u_{jb}^S \tag{6.42}$$

$$I_{ia}^{(2)S,SOS} = c_{os} \sum_{jb} t_{iajb} u_{jb}^S \tag{6.43}$$

$$I_{ia}^{(1)T,SOS} = - \sum_{jb} (ib \mid ja)\, u_{jb}^T \tag{6.44}$$

$$I_{ia}^{(2)T,SOS} = -c_{os} \sum_{jb} t_{iajb} u_{jb}^T \tag{6.45}$$

6.3 Working Equations For Restricted SOS-ADC(2) with Doubles-Folding in an Atomic Orbital Basis

The goal of an atomic orbital based formulation of ADC(2) is to compute the matrix-vector product in an intermediate AO basis and transform it back to the MO basis (or alternatively an LMO basis) for the Davidson procedure, similarly to how it is done for CIS:

$$r_{ia} = C_{\mu i} r_{\underline{\mu}\overline{\nu}} C_{\nu a} \tag{6.46}$$

Furthermore, it is convenient to split the MVP into six components which are evaluated individually

$$r_{ia}(\omega) = r_{ia}^{CIS} + r_{ia}^{2A} + r_{ia}^{2B} + r_{ia}^{2C} + r_{ia}^{2D} + r_{ia}^{2E}(\omega) \tag{6.47}$$

In the next sections, using Equations 6.36 and 6.37 as starting points, the working equations for restricted AO-SOS-ADC(2) will be derived and discussed in detail.

6.3.1 First Order

The first order part of the MVP is identical in both ADC(2) and SOS-ADC(2)

$$r_{ia}^{S,CIS} = (\epsilon_a - \epsilon_i)u_{ia}^S + \sum_{jb} \left[2\,(ia \mid jb) - (ij \mid ab)\right] u_{jb}^S \tag{6.48}$$

$$r_{ia}^{T,CIS} = (\epsilon_a - \epsilon_i)u_{ia}^T - \sum_{jb} (ij \mid ab)\, u_{jb}^T \tag{6.49}$$

An AO formulation is obtained in an identical manner to AO-CIS by factoring out the coefficient matrices to obtain Hartree-Fock-like expressions:

$$\begin{aligned} r_{ia}^{S,CIS,AO} &= (\epsilon_a - \epsilon_i)u_{ia}^S + \sum_{ia} C_{\mu i}C_{\sigma a}\left[\left(2\,(\mu\sigma \mid \nu\lambda) - (\mu\nu \mid \sigma\lambda)\right) u_{\underline{\nu\lambda}}^S\right] \\ &= (\epsilon_a - \epsilon_i)u_{ia}^S + \sum_{ia} C_{\mu i}C_{\sigma a}\left[2\tilde{J}_{\mu\sigma}^S - \tilde{K}_{\mu\sigma}^S\right] \end{aligned} \tag{6.50}$$

$$\begin{aligned} r_{ia}^{T,CIS,AO} &= (\epsilon_a - \epsilon_i)u_{ia}^T - \sum_{ia} C_{\mu i}C_{\sigma a}\left[(\mu\nu \mid \sigma\lambda)\, u_{\underline{\nu\lambda}}^T\right] \\ &= (\epsilon_a - \epsilon_i)u_{ia}^T - \sum_{ia} C_{\mu i}C_{\sigma a}\tilde{K}_{\mu\sigma}^T \end{aligned} \tag{6.51}$$

where $\tilde{\mathbf{J}}$ and $\tilde{\mathbf{K}}$ are the coulomb and exchange kernels, and $u_{\underline{\mu\bar{\sigma}}}$ is the transition density in the AO basis

$$u_{\underline{\mu\bar{\sigma}}} = C_{\mu i}u_{ia}C_{\bar{\sigma}a} \tag{6.52}$$

The zero order terms (i.e. the molecular orbital energy differences) do not need to be formulated in an AO basis, because the computation time is negligible. Similarly, transforming $\tilde{\mathbf{J}}$ and $\tilde{\mathbf{K}}$ to the MO basis formerly scales as $\mathcal{O}(N^3)$ but has very low overhead and does not influence the overall scaling of AO-SOS-ADC(2). The time-determining steps are the computation of the J-kernel, which scales as $\mathcal{O}(N^2)$ and the K-kernel, which scales as $\mathcal{O}(N)$ in limit of large systems. For triplet excitations, the scaling is reduced to linear due to the absence of coulomb contributions.

6.3.2 Second Order: Part 2A and 2B

The expressions for component 2A and 2B read

$$r_{ia}^{S,SOS,2A} = \sum_b I_{ab}^{SOS}u_{ib}^S + \sum_j I_{ij}^{SOS}u_{ja}^S \tag{6.53}$$

$$r_{ia}^{T,SOS,2A} = \sum_b I_{ab}^{SOS}u_{ia}^T + \sum_j I_{ij}^{SOS}u_{ja}^T \tag{6.54}$$

with the intermediates as defined in the previous section. Rather than casting the whole expression into the AO basis, it is more convenient to evaluate only the non-symmetrized intermediates $I_a^{SOS,ns}b$ and $I_{ij}^{SOS,ns}$ in the AO basis. The expressions for the intermediates involve the t-amplitudes, and to obtain an orbital-invariant formulation, it is necessary to use the Laplace transform

$$\frac{1}{\epsilon_a - \epsilon_i + \epsilon_b - \epsilon_j} = \sum_{\alpha}^{nlap} |w^{(\alpha)}| e^{-\epsilon_a t^{(\alpha)}} e^{\epsilon_i t^{(\alpha)}} e^{-\epsilon_b t^{(\alpha)}} e^{\epsilon_j t^{(\alpha)}} \tag{6.55}$$

Using a similar strategy to AO-MP2 to factor out the coefficient matrices, the intermediates can be formulated as

$$I_{ab}^{AO-SOS,ns} = \frac{c_o s}{2} \sum_{kcl} \sum_{\alpha} |w^{(\alpha)}| e^{\Delta_{kalc} t^{(\alpha)}} (ka \mid lc) (kb \mid lc) \tag{6.56}$$

$$= \frac{c_o s}{2} \sum_{b} C_{\lambda b} \sum_{\alpha} \mid w^{(\alpha)} \mid^{1/4} e^{-\epsilon_a t^{(\alpha)}} C_{\sigma a} \sum_{\kappa\gamma\tau} (\underline{\kappa}\sigma \mid \underline{\tau}\overline{\gamma})^{(\alpha)} (\kappa\lambda \mid \tau\gamma) \tag{6.57}$$

$$= \frac{c_o s}{2} \sum_{b} C_{\lambda b} \sum_{\alpha} \mid w^{(\alpha)} \mid^{1/4} e^{-\epsilon_a t^{(\alpha)}} C_{\sigma a} A_{\sigma\lambda}^{(\alpha)} \tag{6.58}$$

with the pseudo-AO electron integrals and the occupied/virtual pseudo density matrices

$$(\underline{\kappa}\sigma \mid \underline{\tau}\overline{\gamma})^{(\alpha)} = P_{\kappa\kappa'}^{(\alpha)} (\kappa'\sigma \mid \tau'\gamma') P_{\tau\tau'}^{(\alpha)} Q_{\gamma\gamma'}^{(\alpha)} \tag{6.59}$$

$$P_{\mu\mu'}^{(\alpha)} = \sum_{i} C_{\mu i} e^{0.25 ln \left|w^{(\alpha)}\right| + \epsilon_i t^{(\alpha)}} C_{\mu' i} \tag{6.60}$$

$$Q_{\nu\nu'}^{(\alpha)} = \sum_{a} C_{\nu a} e^{0.25 ln \left|w^{(\alpha)}\right| - \epsilon_a t^{(\alpha)}} C_{\nu' a} \tag{6.61}$$

Similarly

$$I_{ij}^{AO-SOS,ns} = \frac{c_{os}}{2} \sum_{\alpha} \sum_{ckd} \left|w^{(\alpha)}\right| e^{\Delta_{ickd} t^{(\alpha)}} (ic \mid kd) (jc \mid kd) \tag{6.62}$$

$$= \frac{c_{os}}{2} \sum_{j} C_{\nu j} \sum_{i} \left|w^{(\alpha)}\right|^{1/4} C_{\mu i} e^{\epsilon_i t^{(\alpha)}} \sum_{\gamma\kappa\delta} \left(\mu\overline{\gamma} \mid \underline{\kappa}\overline{\delta}\right) (\nu\gamma \mid \kappa\delta) \tag{6.63}$$

$$= \frac{c_{os}}{2} \sum_{j} C_{\nu j} \sum_{i} \left|w^{(\alpha)}\right|^{1/4} C_{\mu i} e^{\epsilon_i t^{(\alpha)}} B_{\mu\nu}^{(\alpha)} \tag{6.64}$$

Finally, the intermediates are symmetrized

$$I_{ab}^{AO-SOS} = I_{ab}^{AO-SOS,ns} + I_{ba}^{AO-SOS,ns} \tag{6.65}$$

$$I_{ij}^{AO-SOS} = I_{ij}^{AO-SOS,ns} + I_{ji}^{AO-SOS,ns} \tag{6.66}$$

The time-determining step for both intermediates is the computation of the Laplace intermediates $\mathbf{A}^{(\alpha)}$ and $\mathbf{B}^{(\alpha)}$. The subsequent multiplication with the coefficient matrices is again negligible. Consider now the sparsity diagram for the Laplace intermediate $\mathbf{A}^{(\alpha)}$:

$$\lambda \overset{\mathrm{S}}{\leftrightarrow} \kappa \overset{\mathrm{P}}{\leftrightarrow} \kappa' \overset{\mathrm{S}}{\leftrightarrow} \sigma \qquad \tau \overset{\mathrm{P}}{\leftrightarrow} \tau' \overset{\mathrm{S}}{\leftrightarrow} \gamma' \overset{\mathrm{P}}{\leftrightarrow} \gamma$$

The diagram has two edges, and hence $\mathbf{A}$ is evaluated with $\mathcal{O}(N^2)$ computational complexity. The same can be shown for $\mathbf{B}$. The total memory footprint is also quadratic in N.

6.3.3 Second Order: Part 2C

Component 2C is computed as

$$r_{ia}^{S,SOS,2C} = -\frac{c_{os}}{2} \sum_{jb} t_{iajb} I_{jb}^{(1)S,SOS} \tag{6.67}$$

$$r_{ia}^{T,SOS,2C} = -\frac{c_{os}}{2} \sum_{jb} t_{iajb} I_{jb}^{(1)T,SOS} \tag{6.68}$$

Applying the Laplace transform, this then gives

$$\begin{aligned}
r_{ia}^{S,AO-SOS,2C} &= -\frac{c_{os}}{2} \sum_{jb} \sum_{alpha} \left| w^{(\alpha)} \right| e^{\Delta_{iajb} t^{(\alpha)}} (ia \mid jb) \left[\sum_{kc} \left(2 (jb \mid kc) - (jc \mid kb) \right) u_{kc}^{S} \right] \\
&= -\frac{c_{os}}{2} \sum_{alpha} \sum_{ia} \left| w^{(\alpha)} \right|^{1/2} C_{\mu i} e^{\epsilon_i t^{(\alpha)}} C_{\sigma a} e^{-\epsilon_a t^{(\alpha)}} \\
&\quad \left\{ \sum_{\nu\lambda} \left(\mu\alpha \mid \underline{\nu\lambda} \right)^{(\alpha)} \left[\sum_{\kappa\gamma} \left(2 (\nu\lambda \mid \kappa\gamma) - (\nu\gamma \mid \kappa\lambda) \right) u_{\underline{\kappa\gamma}}^{S} \right] \right\} \\
&= -\frac{c_{os}}{2} \sum_{alpha} \sum_{ia} \left| w^{(\alpha)} \right|^{1/2} C_{\mu i} e^{\epsilon_i t^{(\alpha)}} C_{\sigma a} e^{-\epsilon_a t^{(\alpha)}} \left\{ \sum_{\nu\lambda} \left(\mu\alpha \mid \underline{\nu\lambda} \right)^{(\alpha)} \left[2\tilde{J}_{\lambda\nu} - \tilde{K}_{\lambda\nu} \right] \right\} \\
&= -\frac{c_{os}}{2} \sum_{alpha} \sum_{ia} \left| w^{(\alpha)} \right|^{1/2} C_{\mu i} e^{\epsilon_i t^{(\alpha)}} C_{\sigma a} e^{-\epsilon_a t^{(\alpha)}} I_{\mu\sigma}^{(1)(\alpha)S,AO-SOS}
\end{aligned} \tag{6.69}$$

Similarly, triplet contributions are given by

$$\begin{aligned}
r_{ia}^{T,AO-SOS,2C} &= \frac{c_{os}}{2} \sum_{alpha} \sum_{ia} \left| w^{(\alpha)} \right|^{1/2} C_{\mu i} e^{\epsilon_i t^{(\alpha)}} C_{\sigma a} e^{-\epsilon_a t^{(\alpha)}} \left\{ \sum_{\nu\lambda} \left(\mu\alpha \mid \underline{\nu\lambda} \right)^{(\alpha)} \tilde{K}_{\lambda\nu} \right\} \\
&= \frac{c_{os}}{2} \sum_{alpha} \sum_{ia} \left| w^{(\alpha)} \right|^{1/2} C_{\mu i} e^{\epsilon_i t^{(\alpha)}} C_{\sigma a} e^{-\epsilon_a t^{(\alpha)}} I_{\mu\sigma}^{(1)(\alpha)T,AO-SOS}
\end{aligned} \tag{6.70}$$

where $\tilde{\mathbf{J}}$ and $\tilde{\mathbf{K}}$ are the same matrices needed for the CIS contributions. Note that the matrices are *transposed*, i.e. the index order is $\lambda\nu$, and not $\nu\lambda$. The time-determining step is the formation of the Laplace AO intermediates $I_{\mu\nu}^{\alpha(1)}$. Their sparsity diagrams read

$$\mu \overset{\mathrm{S}}{\leftrightarrow} \sigma \qquad \underbrace{\nu \overset{\mathrm{P}}{\leftrightarrow} \nu' \overset{\mathrm{S}}{\leftrightarrow} \lambda' \overset{\mathrm{P}}{\leftrightarrow} \lambda}_{\mathrm{J/K}}$$

The singlet and triplet AO intermediates are therefore evaluated in $\mathcal{O}(N^2)$ time.

6.3.4 Second Order: Part 2D

Now consider part 2D

$$\begin{aligned} r_{ia}^{S,SOS,2D} &= -\frac{1}{2}\sum_{jb}\left[2\left(ia \mid jb\right) - \left(ib \mid ja\right)\right] I_{jb}^{(2)S,SOS} \\ &= -\frac{1}{2}\sum_{jb} K_{iajb} I_{jb}^{(2)S,SOS} \end{aligned} \tag{6.71}$$

$$r_{ia}^{T,SOS,2D} = \frac{1}{2}\sum_{jb}\left(ib \mid ja\right) I_{jb}^{(2)T,SOS} \tag{6.72}$$

Applying the Laplace transform gives the singlet expression

$$\begin{aligned} r_{ia}^{S,AO-SOS,2D} &= -\frac{c_{os}}{2}\sum_{jb} K_{iajb}\sum_{kc}\sum_{\alpha}\left|w^{(\alpha)}\right| e^{\Delta_{iajb}tpa}\left(jb \mid kc\right) u_{kc}^{S} \\ &= -\frac{c_{os}}{2}\sum_{ia} C_{\mu i}C_{\sigma a}\left[K_{\mu\sigma\nu\lambda}\left(\sum_{\alpha}\left(\underline{\nu}\overline{\lambda} \mid \kappa\gamma\right)^{(\alpha)} u_{\underline{\kappa}\overline{\gamma}}^{(\alpha)S}\right)\right] \\ &= -\frac{c_{os}}{2}\sum_{ia} C_{\mu i}C_{\sigma a} I_{\mu\sigma}^{(2)S,AO-SOS} \end{aligned} \tag{6.73}$$

Similarly, the triplet expressions

$$\begin{aligned} r_{ia}^{T,AO-SOS,2D} &= \frac{c_{os}}{2}\sum_{jb}\left(ia \mid jb\right)\sum_{kc}\sum_{\alpha}\left|w^{(\alpha)}\right| e^{\Delta_{iajb}t^{(\alpha)}}\left(jb \mid kc\right) u_{kc}^{T} \\ &= \frac{c_{os}}{2}\sum_{ia} C_{\mu i}C_{\sigma a}\left[\left(\mu\sigma \mid \nu\lambda\right)\left(\sum_{\alpha}\left(\underline{\nu}\overline{\lambda} \mid \kappa\gamma\right)^{(\alpha)} u_{\underline{\kappa}\overline{\gamma}}^{(\alpha)T}\right)\right] \\ &= \frac{c_{os}}{2}\sum_{ia} C_{\mu i}C_{\sigma a} I_{\mu\sigma}^{(2)T,AO-SOS} \end{aligned} \tag{6.74}$$

With the transition density in the pseudo atomic orbital basis

$$u_{\underline{\mu}\overline{\sigma}}^{(\alpha)} = \left|w^{(\alpha)}\right|^{1/2} C_{\mu i} e^{\epsilon_i t^{(\alpha)}} u_{ia} C_{\sigma a} e^{-\epsilon_a t^{(\alpha)}} \tag{6.75}$$

The computation of the AO intermediates $I^{(2)SOS-AO}$ is the time-determining step, and is best evaluated as

$$\tilde{J}_{\mu\sigma}^{(\alpha)} = \left(\mu\sigma \mid \nu\lambda\right) u_{\underline{\nu}\overline{\lambda}}^{(\alpha)} \tag{6.76}$$

$$\tilde{J}_{\underline{\mu}\overline{\sigma}}^{(\alpha)} = P_{\mu\mu'}^{(\alpha)} \tilde{J}_{\mu\nu}^{(\alpha)} Q_{\nu\nu'}^{(\alpha)} \tag{6.77}$$

$$I_{\mu\sigma}^{(2)SOS-AO} = \sum_{\alpha}\left[2\left(\mu\sigma \mid \nu\lambda\right) - \left(\mu\lambda \mid \nu\sigma\right)\right] \tilde{J}_{\underline{\nu}\overline{\lambda}}^{(\alpha)} \tag{6.78}$$

Every individual step can be computed with $\mathcal{O}(N^2)$ complexity, meaning the intermediate is also evaluated with overall quadratic effort.

6.3.5 Second Order: Part 2E

The final part is given by

$$r_{ia}^{S,SOS,2E}(\omega) = c_{osc}\left\{\sum_{kcl}(ik \mid lc)\, u_{kalc}^{S,SOS}(\omega) - \sum_{ckd} u_{ickd}^{S,SOS}(\omega)\,(kd \mid ac)\right\} \tag{6.79}$$

$$r_{ia}^{T,SOS,2E}(\omega) = c_{osc}\left\{\sum_{kcl}(ik \mid lc)\, u_{kalc}^{T,SOS}(\omega) - \sum_{ckd} u_{ickd}^{T,SOS}(\omega)\,(kd \mid ac)\right\} \tag{6.80}$$

With the doubles intermediates as given in Equation 6.38 and 6.39. The Laplace transform needs to be applied to the energy denominator present in these intermediates. The optimal Laplace parameters are however different from the ones used for the t-amplitudes, due to the additional factor of the excitation energy ω. For each different excitation energy ω, a new Laplace quadrature needs to be computed, alongside a new set of pseudo-density matrices $\mathbf{P}$ and $\mathbf{Q}$. The additional time is however negligible for the standard number of quadrature points ($n_{lap} < 10$). The symbol θ is used to designate the Laplace quadrature for the doubles denominator to differentiate them from the ones for the t-amplitudes.

First, an AO formulation of the doubles amplitudes will be derived such that

$$u_{iajb}(\omega) = C_{\mu i} C_{\sigma a} u_{\mu\sigma\nu\lambda} C_{\nu j} C_{\lambda b} \tag{6.81}$$

For quantities like the MO integrals $(ia \mid jb)$, this is straight-forwardly done by factoring out the coefficient matrices. However, the situation is more complex in the doubles intermediates, due to the presence of terms like $u_{ka}\,(ki \mid bj)$. For the MO transition densities, the non-orthogonality of the AO basis needs to be taken into consideration. The MO coefficient matrices are factored out by a PAO backtransform:

$$u_{ia} = C_{\mu i} S_{\mu\mu'} u_{\underline{\mu}'\overline{\sigma}'} S_{\sigma'\sigma} C_{\sigma a} \tag{6.82}$$

The doubles intermediates can then be expressed as

$$\begin{aligned} u_{iajb}^{S/T}(\omega) = -c_{osc} \sum_{\theta} \sum_{\mu\sigma\nu\lambda} |w^{(\theta)}| e^{(\omega-\epsilon_a-\epsilon_b+\epsilon_i+\epsilon_j)t^{(\theta)}} C_{\mu i} C_{\sigma a} C_{\nu j} C_{\lambda b} \Bigg\{ \\ \sum_{\kappa} \left[u_{\underline{\kappa}\overline{\sigma}'}^{S/T} S_{\sigma'\sigma}\,(\kappa\mu \mid \nu\lambda) \pm u_{\underline{\kappa}\overline{\lambda}'}^{S/T} S_{\lambda'\lambda}\,(\nu\kappa \mid \mu\sigma) \right] \\ - \sum_{\gamma} \left[S_{\mu\mu'} u_{\underline{\mu}'\overline{\sigma}}^{S/T}\,(\nu\lambda \mid \sigma\gamma) \pm S_{\nu\nu'} u_{\underline{\nu}'\overline{\gamma}}^{S/T}\,(\mu\sigma \mid \gamma\lambda) \right] \Bigg\} \\ = -\sum_{\theta} \sum_{\mu\sigma\nu\lambda} |w^{(\theta)}| e^{(\omega-\epsilon_a-\epsilon_b+\epsilon_i+\epsilon_j)t^{(\theta)}} C_{\mu i} C_{\sigma a} u_{\mu\sigma\nu\lambda}^{S/T} C_{\nu j} C_{\lambda b} \end{aligned} \tag{6.83}$$

Note the additional minus sign in front of the Laplace summation. After the Laplace transform, the sign of the denominator is swapped, i.e. $\frac{1}{\pm x} \rightarrow exp(\mp x t^{(\theta)})$. For large negative occupied molecular orbital energies ϵ_i or large positive virtual molecular orbital energies ϵ_a, this would lead to very large values and numerical instabilities. For this reason, the minus sign is factored out to reverse the sign in the exponent.

Inserting 6.83 into Equations 6.79 and 6.80 gives the expression for part 2E constructed via AO intermediates:

$$\begin{aligned} r_{ia}^{S/T,AO-SOS,2E}(\omega) &= -c_{osc}^2 \sum_{\theta} e^{\omega t^{(\theta)}} \left\{ C_{\mu i} |w^{(\theta)}|^{1/4} C_{\sigma a} e^{-\epsilon_a t^{(\theta)}} \left[\sum_{\kappa\gamma\tau} \left(\mu\underline{\kappa} \mid \underline{\tau}\overline{\gamma}\right)^{(\theta)} u_{\kappa\sigma\tau\gamma}^{S/T} \right] \right. \\ &\quad \left. - |w^{(\theta)}|^{1/4} C_{\mu i} e^{\epsilon_i t^{(\theta)}} C_{\sigma a} \left[\sum_{\gamma\kappa\delta} u_{\mu\gamma\kappa\delta}^{S/T} \left(\underline{\kappa}\overline{\delta} \mid \sigma\overline{\gamma}\right)^{(\theta)} \right] \right\} \\ &= -c_{osc}^2 \sum_{\theta} e^{\omega t^{(\theta)}} \left\{ C_{\mu i} |w^{(\theta)}|^{1/4} C_{\sigma a} e^{-\epsilon_a t^{(\theta)}} R_{\mu\overline{\sigma}}^{(\theta)(1)S/T} \right. \\ &\quad \left. - |w^{(\theta)}|^{1/4} C_{\mu i} e^{\epsilon_i t^{(\theta)}} C_{\sigma a} R_{\underline{\mu}\sigma}^{(\theta)(2)S/T} \right\} \end{aligned} \tag{6.84}$$

Similarly to previous expressions, the AO electron repulsion integrals are not completely transformed into the pseudo-AO basis, but only three-quarter transformed integrals are obtained. To obtain fully-transformed integrals, it is beneficial to perform the following transformation:

$$r_{ia} = \bar{C}_{\mu i'} r_{\mu\sigma} \bar{C}_{\sigma a'} = \bar{C}_{\mu i'} C_{\mu i} r_{ia} C_{\sigma a} \bar{C}_{\sigma a'} \tag{6.85}$$

Inserting this expression into Equation 6.84 yields

$$\begin{aligned} r_{ia}^{S/T,AO-SOS,2E}(\omega) &= -c_{osc}^2 \sum_{\theta} e^{\omega t^{(\theta)}} \left\{ \bar{C}_{\mu i} P_{\mu\nu} \bar{C}_{\sigma a} \left[\sum_{\kappa\gamma\tau} \left(\underline{\nu\kappa} \mid \underline{\tau}\overline{\gamma}\right)^{(\theta)} u_{\kappa\sigma\tau\gamma}^{S/T} \right] \right. \\ &\quad \left. - \bar{C}_{\mu i} \bar{C}_{\sigma a} Q_{\sigma\nu} \left[\sum_{\gamma\kappa\delta} u_{\mu\gamma\kappa\delta}^{S/T} \left(\underline{\kappa}\overline{\delta} \mid \overline{\nu\gamma}\right)^{(\theta)} \right] \right\} \\ &= -c_{osc}^2 \sum_{\theta} e^{\omega t^{(\theta)}} \left\{ \bar{C}_{\mu i} \bar{C}_{\sigma a} R_{\underline{\mu}\overline{\sigma}}^{(\theta)(1)S/T} - \bar{C}_{\mu i} \bar{C}_{\sigma a} R_{\underline{\mu}\overline{\sigma}}^{(\theta)(2)S/T} \right\} \end{aligned} \tag{6.86}$$

which gives fully transformed integrals. This step is necessary to obtain a better factorization for part 2E in the density fitting approximation. Note that there are other cases of non-fully transformed integrals in the previous parts - however a full transformation does not give any significant advantage for a DF formulation, so they are left unchanged.

The time-determining step is the formation of the **R** intermediates, which in turn depend on the AO doubles intermediates. Consider the sparsity diagram for the following term encountered in Equation 6.83:

$$u_{\underline{\kappa}\overline{\sigma}'} S_{\sigma'\sigma} \left(\kappa\mu \mid \nu\lambda\right) \tag{6.87}$$

$$\nu \overset{\mathrm{S}}{\longleftrightarrow} \lambda \qquad \mu \overset{\mathrm{S}}{\longleftrightarrow} \kappa \overset{\mathrm{P}}{\longleftrightarrow} \sigma' \overset{\mathrm{S}}{\longleftrightarrow} \sigma$$

Similar diagrams can be derived for the other three contractions in Equation 6.83. This shows an overall quadratic scaling in computational effort and number of non-zero elements for the AO doubles intermediates $u_{\mu\sigma\nu\lambda}$. The indices μ/σ and ν/λ are connected by either

an S/P junction, but no sparsity relationship can be established between those pairs, similar to the AO electron integrals.

This information can be used to find the scaling of the $\mathbf{R}$ intermediates. For example, the sparsity diagram for $\mathbf{R}^{(1)}_{\underline{\mu}\overline{\sigma}} = \left(\underline{\mu}\underline{\kappa} \mid \underline{\tau}\overline{\gamma}\right)^{(\theta)} u_{\kappa\sigma\tau\gamma}$ reads

$$\mu \overset{\mathrm{P}}{\leftrightarrow} \mu' \overset{\mathrm{S}}{\leftrightarrow} \kappa' \overset{\mathrm{P}}{\leftrightarrow} \kappa \overset{\mathrm{S/P}}{\leftrightarrow} \sigma \qquad \tau \overset{\mathrm{P}}{\leftrightarrow} \tau' \overset{\mathrm{S}}{\leftrightarrow} \gamma' \overset{\mathrm{P}}{\leftrightarrow} \gamma$$

and a similar diagram can be drawn for $\mathbf{R}^{(2)}$, which again shows quadratic scaling.

6.3.6 Summary

The SOS-ADC(2) matrix-vector product is finally computed as

$$\begin{aligned}
r_{ia}^{S,AO-DF-SOS}(\omega) &= (\epsilon_a - \epsilon_i) u_{ia}^S + \sum_{\mu\nu} C_{\mu i} C_{\nu a} \left(2\tilde{J}^S_{\mu\nu} - \tilde{K}^S_{\mu\nu}\right) \\
&\quad + \sum_b I^{AO-DF-SOS}_{ab} u^S_{ib} + \sum_j I^{AO-DF-SOS}_{ij} u^S_{ja} \\
&\quad - \frac{c_{os}}{2} \sum_\alpha \sum_{ia} \left|w^{(\alpha)}\right|^{1/2} C^{(\alpha)}_{\mu i} C^{(\alpha)}_{\sigma a} I^{(1)(\alpha)S,AO-DF-SOS}_{\mu\sigma} \\
&\quad - \frac{c_{os}}{2} \sum_{ia} C_{\mu i} C_{\sigma a} I^{(2)S,AO-DF-SOS}_{\mu\sigma} \\
&\quad - c^2_{osc} \sum_\theta e^{\omega t^{(\theta)}} \left\{ \bar{C}_{\mu i} \bar{C}_{\sigma a} R^{(\theta)(1)S}_{\underline{\mu}\overline{\sigma}} - \bar{C}_{\mu i} \bar{C}_{\sigma a} R^{(\theta)(2)S}_{\underline{\mu}\overline{\sigma}} \right\}
\end{aligned} \tag{6.88}$$

$$\begin{aligned}
r_{ia}^{T,AO-DF-SOS}(\omega) &= (\epsilon_a - \epsilon_i) u_{ia}^T + \sum_{\mu\nu} C_{\mu i} C_{\nu a} \tilde{K}^T_{\mu\nu} \\
&\quad + \sum_b I^{AO-DF-SOS}_{ab} u^T_{ib} + \sum_j I^{AO-DF-SOS}_{ij} u^T_{ja} \\
&\quad - \frac{c_{os}}{2} \sum_\alpha \sum_{ia} \left|w^{(\alpha)}\right|^{1/2} C^{(\alpha)}_{\mu i} C^{(\alpha)}_{\sigma a} I^{(1)(\alpha)T,AO-DF-SOS}_{\mu\sigma} \\
&\quad - \frac{c_{os}}{2} \sum_{ia} C_{\mu i} C_{\sigma a} I^{(2)T,AO-DF-SOS}_{\mu\sigma} \\
&\quad - c^2_{osc} \sum_\theta e^{\omega t^{(\theta)}} \left\{ \bar{C}_{\mu i} \bar{C}_{\sigma a} R^{(\theta)(1)T}_{\underline{\mu}\overline{\sigma}} - \bar{C}_{\mu i} \bar{C}_{\sigma a} R^{(\theta)(2)T}_{\underline{\mu}\overline{\sigma}} \right\}
\end{aligned} \tag{6.89}$$

where the intermediates are evaluated as presented in the previous sections. The 2-index intermediates still need to be transformed back to the canonical MO basis for the Davidson procedure, but the computational effort required is negligible. The overall cost of the AO-SOS-ADC(2) scales quadratically both in time and memory requirements.

It is expected that AO-SOS-ADC(2) has the same drawback as LinK or AO-MP2, namely a late onset of the low-scaling regime for larger basis sets. This additional overhead is even worse for AO-ADC due to the complexity of the formulas which involve many more tensor contractions than the ground state, and more S and P junctions which makes the method much more dependent on basis set size. This is further aggravated by the fact that diffuse basis functions are essential to obtain accurate excitation energies as opposed to ground state correlation energies.

6.4 Working Equations for Restricted AO-DF-SOS-ADC(2) with Doubles-Folding

To lower the steep scaling associated with increasing basis set size, the density fitting approximation is introduced. The two-electron repulsion integrals are approximated using the generalized form

$$(\mu\sigma \mid \nu\lambda) = B_{\mu\sigma X} M_{XY} B_{Y\nu\lambda} \tag{6.90}$$

where the quantities **B** and **M** depend on the density fitting method. Furthermore, the J-,K- and Z-kernels are introduced:

$$\mathcal{J}\left\{M,P\right\}_{\mu\nu} = B_{\mu\nu X} M_{XY} B_{Y\sigma\lambda} P_{\lambda\sigma} \tag{6.91}$$

$$\mathcal{K}\left\{M,P\right\}_{\mu\nu} = B_{\mu\sigma X} M_{XY} B_{Y\nu\lambda} P_{\lambda\sigma} \tag{6.92}$$

$$\mathcal{Z}\left\{P,Q\right\}_{XY} = B_{X\mu\nu} P_{\mu\mu'} B_{\mu'\nu'Y} Q_{\nu'\nu} \tag{6.93}$$

This notation allows to reduce the complexity of the formulas to some degree.

The working equations for AO-DF-SOS-ADC(2) are given in Tables 2, 3 and 4. Table 2 shows the pre-iteration steps, that is, the computation of the Laplace parameters and the intermediates I_{ij} and I_{ab}. The construction of the intermediates can be formulated in terms of the Z- and K-kernels. In the K-kernel, the metric matrix **M** is replaced by the Laplace matrix **G** formed by the Z-kernel. The contraction of **G** with $B_{X\mu\nu}$ in the K-kernel is the most expensive step in the pre-iteration procedure. The resulting tensor can be computed on-the-fly as it is not needed for any other contraction.

Table 3 shows the steps to form part 1 and 2A-2D of the matrix-vector product. The steps are listed for both singlet and triplet. First the CIS Fock-like matrix $\mathbf{F}^{CIS}$ is formed, as it represents an important intermediate for the subsequent steps. This is easily done using the J/K-kernels. Zero and first order contributions, as well as part 2A and 2B of the second order contributions are formed trivially afterwards using the intermediates. Part 2C is most conveniently computed using the J-kernel where the density matrix is replaced by a pseudo-density matrix **H** formed by contracting with a pseudo-AO CIS matrix. Part 2D is a bit more involved. First, the matrix **T** is formed by looping over the Laplace points and using the J-kernel. Then, this matrix is used in a fock-like construction scheme to get the final result.

Finally, Table 4 shows the construction of part 2E of the MVP. This is the most expensive step of the whole procedure. First, the Laplace parameters are (re-)computed for the current excitation energy ω. The algorithm then enters the Laplace loop in step 2. First, the intermediate tensors $B_{X\underline{\mu}\bar{\nu}}$ and $R_{X\underline{\mu}\underline{\nu}}$ are formed and stored. They are then used in step 2f and 2g to form two intermediate matrices in the auxiliary basis. The final intermediate $D_{X\underline{\mu}\bar{\nu}}$ is formed in step. After two major contraction steps, the contributions are added to part 2E.

Using the density fitting approximation, the four-index intermediates $u_{\mu\sigma\nu\lambda}$ from AO-SOS-ADC(2) can be avoided, and the procedure only depends on 3-index intermediates.

Similarly to AO-MP2, the prefactor can be significantly lowered by virtue of the incomplete Cholesky decomposition which yields the CD-DF-SOS-ADC(2) method. Table 5 and 6 show how the intermediates in part 2E can be formed in a mixed pseudo-AO/MO basis ("OB" algorithm) or in a complete pseudo-MO basis ("OV" algorithm). While the

Algorithm 2: Pre-iterative steps for computing the AO-ADC(2) intermediates

Compute Laplace quadrature parameters $\{w^{(\alpha)}, t^{(\alpha)}\}$ for $(\epsilon_i + \epsilon_j - \epsilon_a - \epsilon_b)^{-1}$
If needed by the J,K or Z kernels, compute the Cholesky decompositions of the occupied and virtual pseudo-density matrix Compute the intermediate matrices I_{ij} and I_{ab}
for $\alpha = 0$ **to** n_{lap} **do**
    $G^{(\alpha)}_{XY} \leftarrow M_{XR} \mathcal{Z}\left\{P^{(\alpha)}, Q^{(\alpha)}\right\}_{RS} M_{SR}$
    $I_{ab} \leftarrow -\frac{c_{os}}{2} C_{\nu b} \sum_\alpha C_{\mu a} \mid w^{(\alpha)} \mid^{1/4} e^{-\epsilon_a t^{(\alpha)}} \mathcal{K}\left\{P^{(\alpha)}, G^{(\alpha)}\right\}_{\mu\nu}$
    $I_{ij} \leftarrow -\frac{c_{os}}{2} C_{\nu j} \sum_\alpha C_{\mu i} \mid w^{(\alpha)} \mid^{1/4} e^{\epsilon_i t^{(\alpha)}} \mathcal{K}\left\{Q^{(\alpha)}, G^{(\alpha)}\right\}_{\mu\nu}$
Symmetrize matrices
    $I_{ij} \leftarrow I_{ji}$
    $I_{ab} \leftarrow I_{ba}$

Algorithm 3: Steps for computing the singles part of the MVP of AO-ADC(2)

Compute the CIS Fock matrices
    if singlet: $U_{\mu\nu} \leftarrow C_{\mu i} u^S_{ia} C_{\nu a}$
    if triplet: $U_{\mu\nu} \leftarrow C_{\mu i} u^T_{ia} C_{\nu a}$
    if singlet: $F^{CIS}_{\mu\nu} \leftarrow 2 * \mathcal{J}\left\{U, M\right\}_{\nu\mu} - \mathcal{K}\left\{U, M\right\}_{\nu\mu}$
    if triplet: $F^{CIS}_{\mu\nu} \leftarrow -\mathcal{K}\left\{U, M\right\}_{\nu\mu}$
Add zero- and first-order terms
    $r_{ia} \leftarrow (\epsilon_a - \epsilon_i) u_{ia}$
    $r_{ia} \leftarrow C_{\mu i} F^{CIS}_{\mu\nu} C_{\nu a}$
Compute part (A) and (B) of second-order term
    $r_{ia} \leftarrow u_{ib} I_{ab}$
    $r_{ia} \leftarrow u_{ja} I_{ij}$
Compute part (C) of second-order term
    for $\alpha = 0$ **to** n_{lap} **do**
        if singlet: $H^{(\alpha)}_{\mu\nu} \leftarrow P^{(\alpha)}_{\mu'\mu} Q^{(\alpha)}_{\nu'\nu} F^{CIS}_{\nu'\mu'}$
        if triplet: $H^{(\alpha)}_{\mu\nu} \leftarrow -P^{(\alpha)}_{\mu'\mu} Q^{(\alpha)}_{\nu'\nu} F^{CIS}_{\nu'\mu'}$
        $r_{ia} \leftarrow \frac{-c_{os}}{4} \mid w^{(\alpha)} \mid^{1/2} C_{\mu i} e^{\epsilon_i t^{(\alpha)}} C_{\nu a} e^{-\epsilon_a t^{(\alpha)}} \mathcal{J}\left\{H^{(\alpha)}, M\right\}_{\mu\nu}$
Compute part (D) of second-order term
    for $\alpha = 0$ **to** n_{lap} **do**
        if singlet: $U^{(\alpha)}_{\mu\nu} \leftarrow \mid w^{(\alpha)} \mid^{1/2} C_{\mu i} e^{\epsilon_i t^{(\alpha)}} C_{\nu a} e^{-\epsilon_a t^{(\alpha)}} u^S_{ia}$
        if triplet: $U^{(\alpha)}_{\mu\nu} \leftarrow - \mid w^{(\alpha)} \mid^{1/2} C_{\mu i} e^{\epsilon_i t^{(\alpha)}} C_{\nu a} e^{-\epsilon_a t^{(\alpha)}} u^T_{ia}$
        $T_{\mu\nu} \leftarrow \frac{1}{2} \sum_\alpha P^{(\alpha)}_{\mu\mu'} Q^{(\alpha)}_{\nu\nu'} \mathcal{J}\left\{U^{(\alpha)}, M\right\}_{\nu\mu}$
    $r_{ia} \leftarrow -\frac{c_{os}}{2} C_{\mu i} C_{\nu a} \left[2\mathcal{J}\left\{T, M\right\}_{\mu\nu} - \mathcal{K}\left\{T, M\right\}_{\mu\nu}\right]$

Algorithm 4: Steps for computing the doubles part of the MVP of AO-ADC(2)

Compute Laplace quadrature parameters $\{w^{(\theta)}, t^{(\theta)}\}$ for $(-\omega + \epsilon_i + \epsilon_j - \epsilon_a - \epsilon_b)^{-1}$
for $\alpha = 0$ **to** n_{lap} **do**
  Compute doubles pseudo-matrices $P^{(\theta)}$ and $Q^{(\theta)}$
  $B^{(\theta)}_{X\underline{\mu}\overline{\sigma}} \leftarrow P^{(\theta)}_{\mu\mu'} B_{X\mu'\nu'} Q^{(\theta)}_{\nu'\nu}$
  $v^{(1)(\theta)}_{\underline{\mu}\overline{\sigma}} \leftarrow P^{(\theta)}_{\mu\lambda} S_{\lambda\nu} u_{\underline{\nu}\overline{\sigma}}$
  $v^{(2)(\theta)}_{\underline{\mu}\overline{\sigma}} \leftarrow u_{\underline{\mu}\overline{\gamma}} S_{\gamma\lambda} Q^{(\theta)}_{\lambda\sigma}$
  $R^{(\theta)}_{X\underline{\mu}\overline{\sigma}} \leftarrow P^{(\theta)}_{\mu\lambda} B_{X\lambda\nu} v^{(1)(\theta)}_{\underline{\nu}\overline{\sigma}} - v^{(2)(\theta)}_{\underline{\mu}\overline{\gamma}} B_{X\gamma\nu} Q^{(\theta)}_{\nu\sigma}$
  $H^{(\theta)}_{XY} \leftarrow B^{(\theta)}_{X\underline{\mu}\overline{\sigma}} B_{Y\mu\sigma}$
  $G^{(\theta)}_{XY} \leftarrow R^{(\theta)}_{X\underline{\mu}\overline{\sigma}} B_{Y\mu\sigma}$
  $D^{(\theta)}_{X\underline{\mu}\overline{\sigma}} \leftarrow H^{(\theta)}_{XY} R^{(\theta)}_{Y\underline{\mu}\overline{\sigma}} + G^{(\theta)}_{XY} B_{Y\underline{\mu}\overline{\sigma}}$
  $r^{(A)}_{ia}(\omega) \leftarrow \overline{C}_{\mu i} P_{\mu\mu'} \left[D^{(\theta)}_{X\underline{\nu}\overline{\sigma}} B_{X\nu\mu'} \right] \overline{C}_{\sigma a}$
  $r^{(B)}_{ia}(\omega) \leftarrow \overline{C}_{\mu i} \left[D^{(\theta)}_{X\underline{\mu}\overline{\gamma}} B_{X\gamma\sigma'} \right] Q_{\sigma'\sigma} \overline{C}_{\sigma a}$
  $r_{ia} += c^2_{os-coupling} e^{\omega t^{(\alpha)}} \left[-r^{(A)}_{ia}(\alpha, \omega) + r^{(B)}_{ia}(\alpha, \omega) \right]$

fully transformed OV version has a lower memory footprint by reducing the dimension from $N_X N_{AO}^2$ to $N_X OV$, the overhead due to the additional Cholesky decomposition of the virtual pseudo-density might outweigh the benefit: first, the virtual space may be quite large, especially if large basis sets are used, increasing the rank of the matrix and hence the computational effort of the Cholesky decomposition. Second, the decompositions need to be recomputed for each ω, and for each Laplace point. The OB version might offer a compromise between memory savings and additional overhead.

It should be noted that the Cholesky decomposition can also be used in the Z-kernels in Table 2 to reduce its prefactor as well.

By using local density fitting, the inherent quadratic scaling of the AO-SOS-ADC(2) method also applies for AO-DF-SOS-ADC(2) and CD-DF-SOS-ADC(2).

Algorithm 5: Steps for computing the doubles part of the MVP of AO-ADC(2), with Cholesky decomposition of occupied densities

Compute Laplace quadrature parameters $\{w^{(\theta)}, t^{(\theta)}\}$ for $(-\omega + \epsilon_i + \epsilon_j - \epsilon_a - \epsilon_b)^{-1}$
for $\alpha = 0$ **to** n_{lap} **do**
  Compute doubles pseudo-matrices $P^{(\theta)}$ and $Q^{(\theta)}$ and the cholesky decomposition $L_{\mu\underline{i}}$
  $B^{(\theta)}_{X\underline{i}\overline{\sigma}} \leftarrow L^{(\theta)}_{\mu\underline{i}} B_{X\mu\nu} Q^{(\theta)}_{\nu\sigma}$
  $v^{(1)(\theta)}_{\underline{i}\overline{\sigma}} \leftarrow L^{(\theta)}_{\mu\underline{i}} S_{\mu\nu} u_{\underline{\nu}\overline{\sigma}}$
  $v^{(2)(\theta)}_{\underline{\mu}\overline{\sigma}} \leftarrow u_{\underline{\mu}\overline{\gamma}} S_{\gamma\lambda} Q^{(\theta)}_{\lambda\sigma}$
  $R^{(\theta)}_{X\underline{i}\overline{\sigma}} \leftarrow L^{(\theta)}_{\mu\underline{i}} B_{X\mu\nu} v^{(1)(\theta)}_{\underline{\nu}\overline{\sigma}} - v^{(2)(\theta)}_{\underline{i}\overline{\gamma}} B_{X\gamma\nu} Q^{(\theta)}_{\nu\sigma}$
  $H^{(\theta)}_{XY} \leftarrow L^{(\theta)}_{\mu\underline{i}} B^{(\theta)}_{X\underline{i}\overline{\sigma}} B_{Y\mu\sigma}$
  $G^{(\theta)}_{XY} \leftarrow L_{\mu\underline{i}} R^{(\theta)}_{X\underline{i}\overline{\sigma}} B_{Y\mu\sigma}$
  $D^{(\theta)}_{X\underline{i}\overline{\sigma}} \leftarrow H^{(\theta)}_{XY} R^{(\theta)}_{Y\underline{i}\overline{\sigma}} + G^{(\theta)}_{XY} B_{Y\underline{i}\overline{\sigma}}$
  $r^{(A)}_{ia}(\omega) \leftarrow \overline{C}_{\lambda i} P_{\lambda\mu} \left[D^{(\theta)}_{X\underline{k}\overline{\sigma}} B_{X\nu\mu} L^{(\theta)}_{\nu\underline{k}} \right] \overline{C}_{\sigma a}$
  $r^{(B)}_{ia}(\omega) \leftarrow \overline{C}_{\mu i} L_{\mu\underline{i}} \left[D^{(\theta)}_{X\underline{i}\overline{\gamma}} B_{X\sigma\gamma} \right] Q_{\gamma\lambda} \overline{C}_{\lambda a}$
  $r_{ia} += c^2_{os-coupling} e^{\omega t^{(\alpha)}} \left[-r^{(A)}_{ia}(\alpha, \omega) + r^{(B)}_{ia}(\alpha, \omega) \right]$

Algorithm 6: Steps for computing the doubles part of the MVP of AO-ADC(2), with Cholesky decomposition of occupied and virtual densities

Compute Laplace quadrature parameters $\{w^{(\theta)}, t^{(\theta)}\}$ for $(-\omega + \epsilon_i + \epsilon_j - \epsilon_a - \epsilon_b)^{-1}$
for $\alpha = 0$ **to** n_{lap} **do**
  Compute doubles pseudo-matrices $P^{(\theta)}$ and $Q^{(\theta)}$ and their cholesky decompositions $L^{(\theta)}_{\mu\underline{i}}$ and $L^{(\theta)}_{\sigma\overline{a}}$
  $B^{(\theta)}_{X\underline{i}\overline{a}} \leftarrow L^{(\theta)}_{\mu\underline{i}} B_{X\mu\nu} L^{(\theta)}_{\nu\overline{a}}$
  $v^{(1)(\theta)}_{\underline{i}\overline{\sigma}} \leftarrow L^{(\theta)}_{\mu\underline{i}} S_{\mu\nu} u_{\underline{\nu}\overline{\sigma}}$
  $v^{(2)(\theta)}_{\underline{\mu}\overline{a}} \leftarrow u_{\underline{\mu}\overline{\gamma}} S_{\gamma\lambda} L^{(\theta)}_{\lambda\overline{a}}$
  $R^{(\theta)}_{X\underline{i}\overline{a}} \leftarrow L^{(\theta)}_{\mu\underline{i}} B_{X\mu\nu} v^{(1)(\theta)}_{\underline{\nu}\overline{\alpha}} - v^{(2)(\theta)}_{\underline{i}\overline{\gamma}} B_{X\gamma\nu} L^{(\theta)}_{\nu\overline{a}}$
  $H^{(\theta)}_{XY} \leftarrow B^{(\theta)}_{X\underline{i}\overline{a}} B_{Y\underline{i}\overline{a}}$
  $G^{(\theta)}_{XY} \leftarrow R^{(\theta)}_{X\underline{i}\overline{a}} B_{Y\underline{i}\overline{a}}$
  $D^{(\theta)}_{X\underline{i}\overline{a}} \leftarrow H^{(\theta)}_{XY} R^{(\theta)}_{Y\underline{i}\overline{a}} + G^{(\theta)}_{XY} B_{Y\underline{i}\overline{a}}$
  $r^{(A)}_{ia}(\omega) \leftarrow \overline{C}_{\lambda i} P_{\lambda\mu} \left[D^{(\theta)}_{X\underline{k}\overline{a}} B_{X\nu\mu} L_{\nu\underline{k}} \right] L_{\sigma\overline{a}} \overline{C}_{\sigma a}$
  $r^{(B)}_{ia}(\omega) \leftarrow \overline{C}_{\mu i} L_{\mu\underline{i}} \left[D^{(\theta)}_{X\underline{i}\overline{b}} B_{X\gamma\sigma} L_{\sigma\overline{b}} \right] Q_{\gamma\lambda} \overline{C}_{\lambda a}$
  $r_{ia} += c^2_{os-coupling} e^{\omega t^{(\alpha)}} \left[-r^{(A)}_{ia}(\alpha, \omega) + r^{(B)}_{ia}(\alpha, \omega) \right]$

Chapter 7

Scaling and Accuracy of atomic-orbital based SOS-ADC(2)

The theoretical groundwork for CD-DF-SOS-ADC(2) was laid out in detail in the previous chapters. In this chapter, the performance of the method is explored in terms of scaling, memory footprint and accuracy. First, the J, K and Z kernels are analyzed in the context of ground state Hartree-Fock and SOS-MP2 calculations. They form the heart of the AO-SOS-ADC(2) machinery and a separate analysis is therefore justified. Moreover, bugs and performance issues are easier to spot. In the second part of this chapter, the performance of the AO-SOS-ADC(2) method itself is investigated. Results are then summarized at the end of this chapter.

7.1 Computational Details

If not stated otherwise, results presented in this chapter were obtained using MEGALOchem, an open-source quantum chemistry package which provides an optimized environment for algorithms exploiting the sparsity of the AO basis. For more details, the reader is referred to the subsequent chapters.

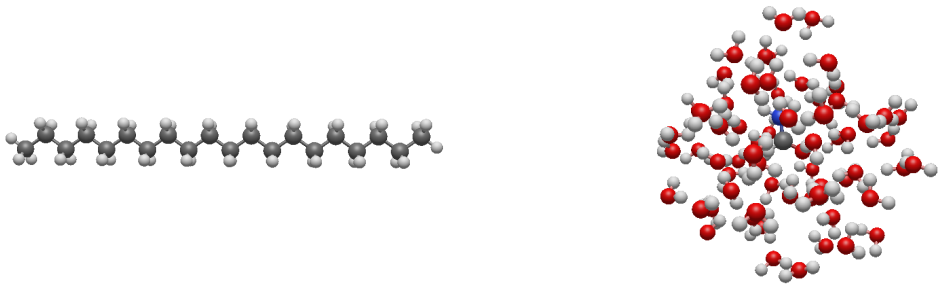
Reference values for Hartree-Fock and SOS-MP2 energies, as well as excitation energies for canonical SOS-ADC(2) were obtained with version 5.1 of the Q-Chem quantum chemistry package.

All calculations that use MEGALOchem have been performed on two nodes with two Intel E5-2690v3 Haswell CPUs and 512 GB RAM each (48 cores in total), located on the Tegner cluster at the PDC in Stockholm. Absolute wall times for scalings should not be considered, as developmental builds of MEGALOchem are used, and MPI processes are not optimally distributed.

7.2 Ground-state Prerequisites

7.2.1 Molecular Test Systems

Virtually all works that present some form of low-scaling electronic structure methods use linear alkanes (LA) as their test systems. Linear molecular system represent a



(a) (b)

Figure 7.1: Molecular systems used for the analysis of the J, K and Z kernels: (a) linear alkanes (LA); (b) hydrated formamide (FW)

best case scenario where the overlap between basis functions μ, ν decays very rapidly as function of distance $r_{\mu\nu}$. The low scaling regime can generally be reached quite quickly, which is great for getting a first impression on the performance on the methods. Unfortunately, linear systems like alkanes are chemically uninteresting. For this reason, the present benchmarking also looks at the worst case scenario of spherically shaped and thus electron-dense molecular system, in this case hydrated formamide (FW) with differently sized solvation shells. For systems such as these, the strength of electron correlation effects decreases much more slowly as a function of increasing system size N. The systems are illustrated in Figure 7.1. LA structure are taken from **??** and FW structures from **??**.

Basis sets also play an important role. For correlated methods such as MP2, Coupled Cluster and their excited state analogs, triple-zeta quality basis sets are mandatory for obtaining accurate results for small to medium-sized molecules. A larger basis set implicates a larger virtual space to capture correlation effects. Fortunately, for larger molecules, basis set superposition makes larger basis sets less crucial. Here, the small but still routinely used cc-pVDZ basis set is considered for the computation of ground state properties. For Hartree-Fock and MP2, the auxiliary basis sets cc-pVDZ-jkfit and cc-pVDZ-ri are used, respectively.

Table **??** shows the total number of basis functions of the systems considered in this section. The length of the alkane chains and the size of the solvation shell of FW are chosen such that they are on the same order of magnitude. For each system type, four sizes are chosen.

7.2.2 Illustrating the Scaling

There are generally two ways how scaling is illustrated in literature: (1) a graph that shows the total wall time as a function of increasing system size and/or (2) a table with

LA			FW		
Molecule	Abbrev.	N_{AO}	Molecule	Abbrev.	N_{AO}
$H_{42}C_{20}$	LA20	490	$H_{33}CNO_{16}$	FW15	417
$H_{82}C_{40}$	LA40	970	$H_{63}CNO_{31}$	FW30	777
$H_{162}C_{80}$	LA80	1930	$H_{129}CNO_{64}$	FW63	1569
$H_{322}C_{160}$	LA160	3850	$H_{291}CNO_{145}$	FW144	3513

Table 7.1: Molecular formula for the considered systems and the number of basis functions for cc-pVDZ

scaling coefficients computed relative to a previous system size as

$$x = \frac{log(N_i/N_{i-1})}{log(T_i/T_{i-1})} \tag{7.1}$$

Graphs are most useful to show the prefactor of the methods, while tables are much better suited for showing the polynomial scaling, as the differences between e.g. $\mathcal{O}(N^2)$ and $\mathcal{O}(N^{2.5})$ are difficult to pick up with the naked eye. Here, both methods are used.

7.2.3 Integral Evaluation

The J and K kernels have the tensors $B_{X\mu\nu}$ and M_{XY} as common input. They are defined by

$$(\mu\nu \mid \lambda\sigma) = B_{X\mu\nu} M_{XY} B_{X\lambda\sigma} \tag{7.2}$$

Three different density fitting approximations are considered for benchmarking: the standard density fitting in the coulomb metric (DFCM), local density fitting in the coulomb attenuated metric with the complimentary error function (DFCAM), and quasi-robust density fitting (QRDF). The exact form of **B** and **M** depends on the DF approximation (Table **??**). If not indicated otherwise, DFCAM uses an attenuation factor of 0.1, and QRDF uses $T =$ 1e-5 and $R = 40$.

Figure 7.2 shows the time needed to evaluate **B** for the LA systems. There is no difference between DFCM and DFCAM, hence only DFCM is shown. QRDF has a much higher prefactor due to the huge number of QR decompositions that are necessary for the calculation of the fitting coefficients. The consequence is that QRDF is an order of magnitude more expensive than DFCM or DFCAM. Comparing the wall times here to the the total time needed for the Hartree-Fock procedure, QRDF is actually more expensive than the hole HF computation for LA20, LA40 and LA80. However, QRDF

DF method	$B^X_{\mu\nu}$	M_{XY}
DFCM	$(X \mid \mu\nu)$	$(X \mid Y)^{-1}$
DFCAM	$(X \mid \mu\nu)_\omega$	$(X \mid Y)^{-1}_\omega (Y \mid R) (R \mid S)^{-1}_\omega$
QRDF	$C^{QRDF}_{X\mu\nu}$	$(X \mid Y)$

Table 7.2: Expressions for **B** and **M** for kernels presented in this work. The subscript ω indicates that the coulomb attenuated metric is used.

has the advantage of becoming linearly scaling going from LA80 to LA160, compared to the quadratic scaling of DFCM.

In QRDF, the test functions $\{Q\}$ for the fitting procedure are chosen based on overlap criteria with the fitting functions $\{P\}$ (Equation 3.42). Here lies the reason for the massive overhead of QRDF: in the original implementation by Tew, the smallest exponent of a GTO is chosen for determining the overlap between μ_Q and ν_P. In the QRDF procedure as implemented in MEGALOchem, the scheme is generalized to *blocks* of basis shells centered on the same atom, i.e. the smallest exponent of the whole block is used for the overlap screening. This had the unwanted side-effect that the screening for the test functions is much harsher than necessary. Even if the overlap values for most exponents of a block with another block are zero, only one exponent decides whether the block is included in the fitting procedure. A more lenient criteria based on an average overlap or matrix norm can reduce the overhead with little impact on accuracy. For the remainder of this chapter, the original, strict version of the QRDF algorithm is used. A newer version of the QRDF procedure is currently being developed. The strictness of the method has no impact on the sparsity of the fitting coefficients, and hence all findings on the scaling of the kernels will still be valid for the new implementation.

Figure 7.3 shows the number of non-zero elements of $\mathbf{B}$ for LA. Quadratic scaling can be observed for DFCM, and linear scaling for DFCAM and QRDF. An interesting observation is that $\mathbf{B}$ is evaluated with $\mathcal{O}(N^2)$ effort in DFCAM, but it scales linearly in its elements. This is due to the screening procedure: for DFCAM, the normal DFCM Schwarz screening is used, which does not take into account the exponential decay between $(X|$ and $|\mu\nu)$. More strict screening procedures could be introduced to speed up integral evaluation of DFCAM, which will be especially useful for direct algorithms. Similarly, the QRDF scheme has mostly cubic scaling effort for evaluation, but ultimately the elements scale linearly with system size.

Finally, Figure 7.4 shows the memory footprint of $\mathbf{B}$ for the hydrated formamide systems. The tensors are naturally much more dense due to the molecular structure. Nonetheless, the QRDF fitting coefficients still scale quite favorably. Unfortunately, the QRDF procedure itself takes ten times longer than the HF calculation itself. Whether this is due to the strictness of QRDF, or just the nature of QRDF itself, is a point for further investigation. For now, QRDF is not considered useful for dense electronic systems.

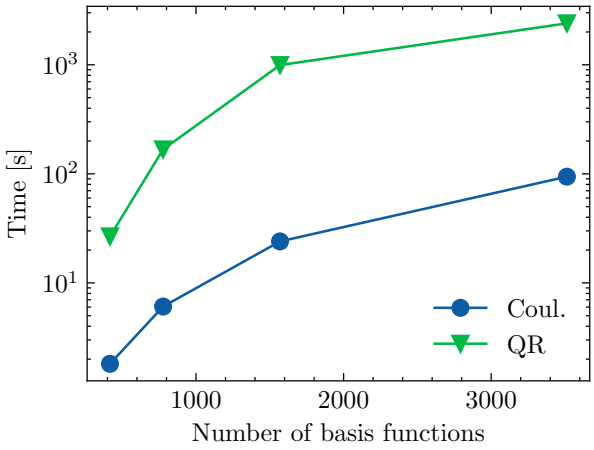
To compute $\mathbf{M}$ in DFCM and DFCAM, a matrix inversion needs to computed, which scales with $\mathcal{O}(N^3)$. For the system sizes considered in this report, the relative wall times for matrix inversions are small compared to all other steps. The QR decompositions in QRDF also scale cubically, although the size of the linear lest squares problem will eventually become constant, i.e. the number of test functions will be independent of system sizes. This is a major advantage of QRDF, although as was already pointed out, it does matter for the current molecular sizes.

7.2.4 Hartree-Fock

Scaling: Coulomb Matrix

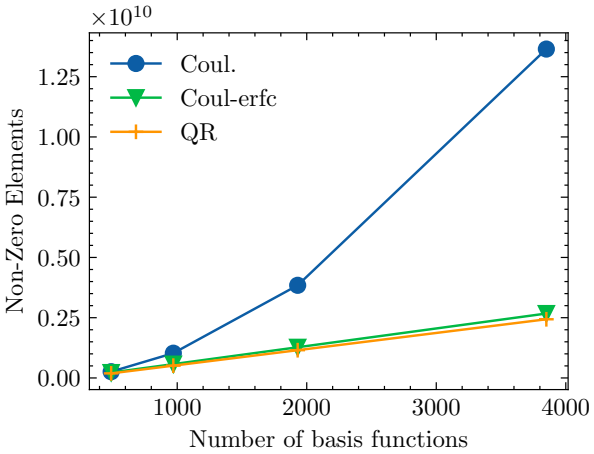
The coulomb matrix is evaluated as

$$d_X = M_{XY} B_{Y\mu\nu} P_{\nu\mu} \tag{7.3}$$



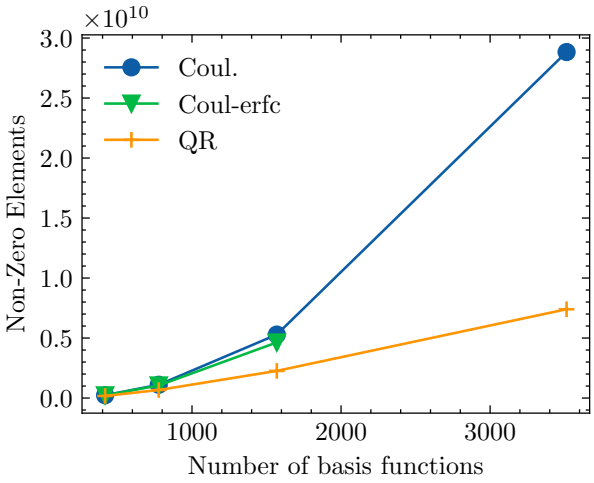
N_{AO}	DFCM	QRDF
490	—	—
970	1.94	2.95
1930	1.96	2.53
3850	1.70	1.09

Figure 7.2: Total walltime needed to construct the tensor $B_{X\mu\nu}$ (3c2e integrals, QR fitting coefficients) for LA



N_{AO}	DFCM	DFCAM	QRDF
490	—	—	—
970	2.01	1.37	1.45
1930	1.92	1.16	1.18
3850	1.83	1.07	1.08

Figure 7.3: Number of non-zero elements in the tensor $B_{X\mu\nu}$ as a function of N_{AO} for LA, with the corresponding scaling coefficients



N_{AO}	DFCM	DFCAM	QRDF
417	—	—	—
777	2.50	2.45	2.15
1569	2.23	2.11	1.73
3513	2.10	—	1.47

Figure 7.4: Number of non-zero elements in the tensor $B_{X\mu\nu}$ as a function of N_{AO} for FW, with the corresponding scaling coefficients

$$J_{\mu\nu} = B_{X\mu\nu} d_X \tag{7.4}$$

The scaling of the individual steps is not considered, but only the total time needed to construct the coulomb matrix. Figure 7.5 shows the scaling behavior of the J kernel for the LA systems. DFCM does not lower the scaling compared to the exact evaluation of the coulomb matrix which scales quadratically if sparse matrix algebra is used. Local metrics can however lower the scaling by an order of magnitude. QRDF even exhibits sublinear scaling from LA80 to LA160. The sublinear scaling originates from the increased sparsity of $\mathbf{M}$. While DFCM and DFCAM need to invert matrices, which also destroys sparsity, QRDF only needs the 2c2e integrals which have an $1/R^{l+1}$ decay between centers. Although this decay is slow, it is faster compared to the 4c2e integrals and can affect the scaling positively.

Table 7.3 shows the scaling coefficients for the FW systems. Here, DFCAM and QRDF do not give any considerable advantage over DFCM.

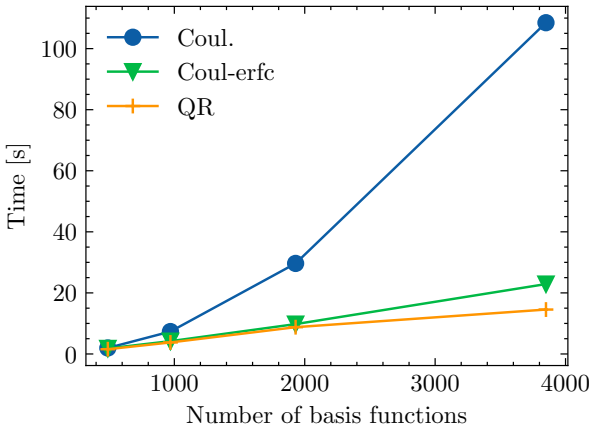
Scaling: Exchange Matrix

The exchange matrix is computed as

$$C_{X\mu\nu} = M_{XY} B_{Y\mu\nu} \tag{7.5}$$

$$K_{\mu\nu} = B_{X\nu\sigma} C_{X\mu\lambda} P_{\lambda\sigma} \tag{7.6}$$

Both steps are analyzed separately. Step 1 actually only needs to be evaluated once, while step 2 is repeated at each iteration. Figure 7.6 illustrates the scaling for step 1 for linear alkanes. It is the most expensive step of the Hartree-Fock procedure in terms of scaling (ignoring diagonalization and matrix inversion), and profits most from the local density fitting approximation. The computational effort can be reduced to $\mathcal{O}(N^2)$ and almost $\mathcal{O}(N)$ for QRDF. Step 1 forms a 3-index tensor which needs to be stored in memory. The sparsity of this tensor is illustrated in Figure 7.9. The number of elements scales approximately quadratically for all metrics, although QRDF has a slight edge on



N_{AO}	DFCM	DFCAM	QRDF
490	—	—	—
970	1.93	1.23	1.29
1930	2.03	1.23	1.23
3850	1.88	1.23	0.73

Figure 7.5: Scaling behaviour for the construction of the coulomb matrix using different metrics (LA)

	J			K (STEP 1)			K (STEP 2)			
N_{AO}	DFCM	DFCAM	QRDF	DFCM	DFCAM	QRDF	DFCM	DFCAM	QRDF	DFMO
417	—	—	—	—	—	—	—	—	—	—
777	2.18	2.20	1.88	2.77	2.77	2.51	2.55	2.55	2.33	2.33
1569	2.32	1.71	1.28	2.77	2.18	1.85	2.82	2.13	1.67	2.86
3513	2.08	—	1.94	2.79	—	2.74	2.59	—	2.65	3.13

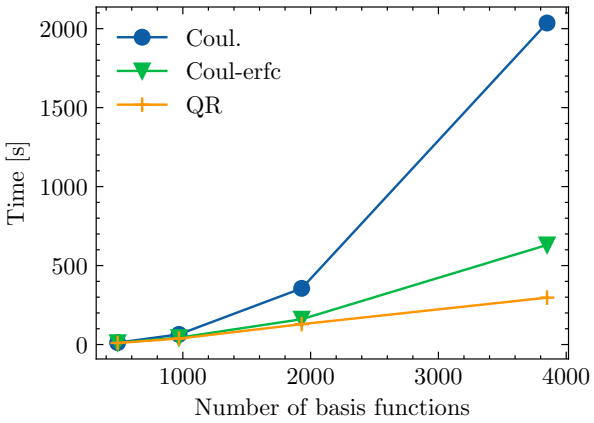
Table 7.3: Scaling coefficients for the J kernel and the two steps of the K kernel (FW)

the other methods. It appears that local density fitting approximations do not have a lot of impact on the memory requirements of step 1, although they can speed up the computation considerably.

Step 2 is less expensive, but still more demanding than the computation of the coulomb matrix. Figure 7.7 shows the scaling for linear alkanes. Moreover, the MO algorithm is also included for comparison. Even standard density fitting can positively impact the scaling, with a reduction from cubic to $\mathcal{O}(N^{1.5})$. DFCAM does not improve on the scaling, although the additional sparsity lowers the prefactor compared to DFCM. QRDF can again achieve linear scaling, for similar reasons as discussed above.

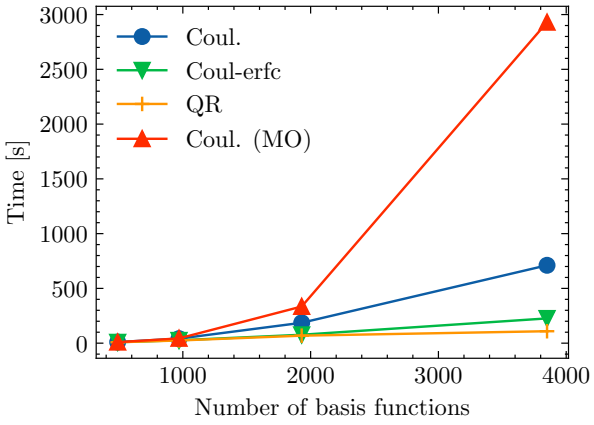
Finally, the scaling coefficients for the FW systems are collected in Table 7.3. As expected, local density fitting does not considerably reduce the scaling for evaluating the exchange matrix and shows the limit of what AO methods can do. Of course, at some point, the computational effort will eventually decrease with increasing system size N, although the crossover point is much later than for LA. The major advantage is that, compared to a MO implementation, local density fitting decreases the overhead compared to DFCM, making it more competitive to the MO methods (Figure **??**). The AO method is therefore still applicable in a dense context.

For FW, numerical issues were encountered for FW144, hence this point is missing in the tables and figures.



N_{AO}	DFCM	DFCAM	QRDF
490	—	—	—
970	2.43	1.97	1.96
1930	2.49	1.89	1.78
3850	2.53	1.98	1.21

Figure 7.6: Scaling behaviour for the construction of the exchange matrix (step1) using different metrics (LA)



N_{AO}	DFCM	DFCAM	QRDF	DFMO
490	—	—	—	—
970	1.58	1.58	1.60	2.34
1930	1.57	1.57	1.50	2.96
3850	1.56	1.56	0.67	3.14

Figure 7.7: Scaling behaviour for the construction of the exchange matrix (step 2) using different metrics (LA)

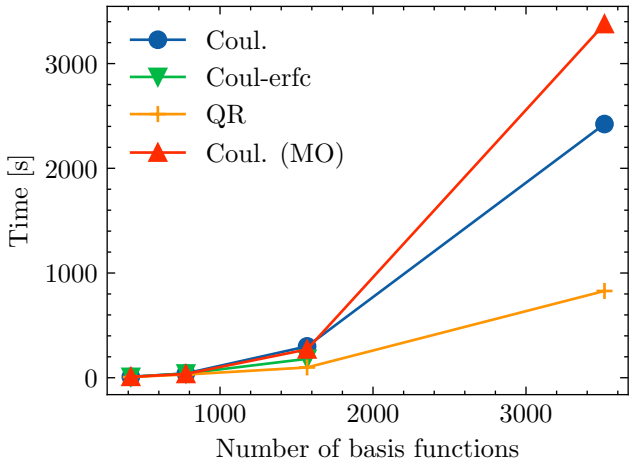
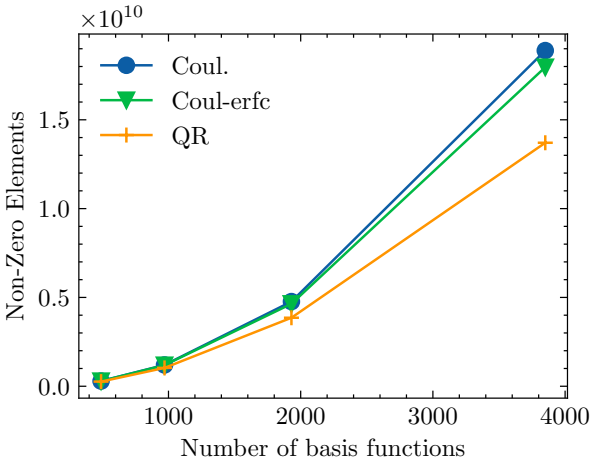
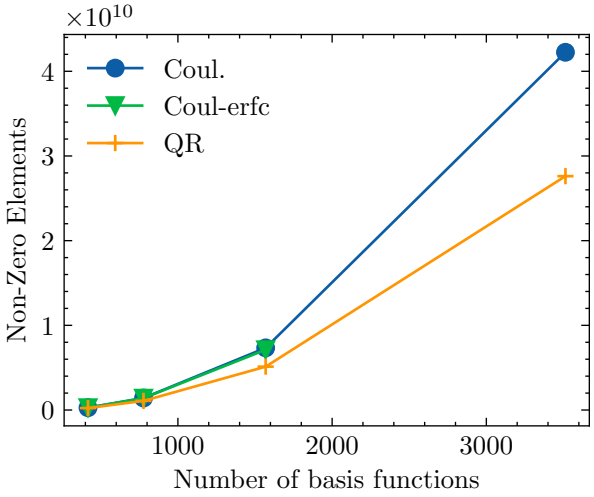


Figure 7.8: Scaling for the construction of the exchange matrix (step 2) for hydrated formamide. Although local density approximations do not lower the scaling, a reduction of the prefactor can be observed.



N_{AO}	DFCM	DFCAM	QRDF
490	—	—	—
970	2.07	2.06	2.01
1930	2.01	1.98	1.92
3850	1.99	1.96	1.84

Figure 7.9: Number of non-zero elements of the tensor $M_{XY}B_{Y\mu\nu}$ as a function of N_{AO}



N_{AO}	DFCM	DFCAM	QRDF
417	—	—	—
777	2.27	2.26	2.20
1569	1.96	1.94	1.88
3513	1.71	1.68	1.57

Table 7.4: Scaling coefficients of $M_{XY}B_{Y\mu\nu}$ (FW)

Figure 7.10: Number of non-zero elements of $M_{XY}B_{Y\mu\nu}$ as a function of N_{AO} (FW)

	DFCM	DFCAM	QRDF
LA20	3.76	235.21	3.80
LA30	3.85	274.59	3.85
LA40	3.85	368.60	3.84
FW15	8.23	713.43	8.21
FW21	7.95	524.92	7.95
FW30	1.59	111.87	1.59

Table 7.5: Absolute Hartree-Fock energy difference in μHartrees per occupied orbital compared to exact Hartree-Fock.

Accuracy

The accuracy of the J and K kernels are compared collectively by considering the total Hartree Fock energy. Table **??** lists the total energy differences per occupied orbital in μHartrees for a small set of molecules. Here, LA30 and FW21 are introduced, with the respective molecular formulas $H_6 2C_3 0$ and $CH_4 5O_2 2$. For standard density fitting, one finds errors on the order of several μHartrees, which is in accordance to results in literature. QRDF has virtually the same errors compared to DFCM, although this might also be partly due the strict screening of test functions. DFCAM unfortunately shows very large errors that are two orders of magnitude larger than QRDF and DFCM. Here lies yet another advantage of the QRDF scheme: while Dunlap's robust density fitting is crucial for obtaining accurate results for local density fitting approximations like DFCAM, QRDF has no such restrictions.

7.2.5 MP2

Scaling

The Z kernel is defined here as

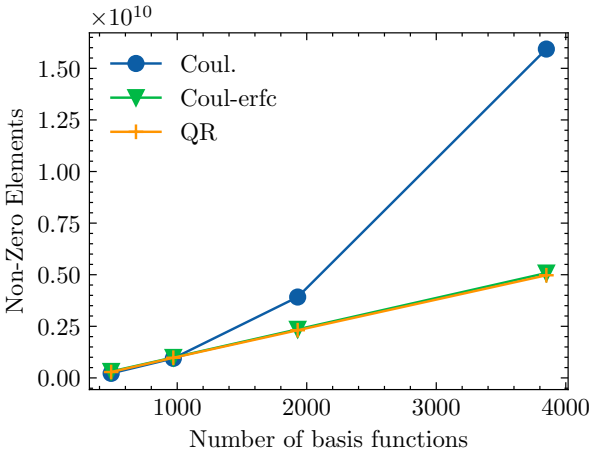
$$D^{(\alpha)}_{X\underline{\mu\bar{\nu}}} = L^{(\alpha)}_{\mu\bar{i}} \left(L^{(\alpha)}_{\mu'\bar{i}} B_{X\mu'\nu'} Q^{(\alpha)}_{\nu'\nu} \right) \tag{7.7}$$

$$Z^{(\alpha)}_{XY} = D^{(\alpha)}_{X\underline{\mu\bar{\nu}}} B_{Y\mu\nu} \tag{7.8}$$

and is evaluated for each Laplace point α. The 3-index intermediate $\mathbf{D}$ can either be stored in main memory, stored on disk, or computed on the fly. Here, only the first variant of the algorithm is considered, although the other methods are implemented as well. First, the sparsity of the intermediate tensor will be considered (Figure 7.11) for the LA systems. Quadratic scaling is observed for DFCM, while linear scaling is achievable for DFCAM and QRDF. This is in accordance to the sparsity analysis in chapter 4. Again, local density fitting is crucial to reduce the memory footprint.

The scaling behaviour of the Z kernel is collected in Figure 7.12 for LA. Even with DFCM, the computational effort is drastically reduced from quartic to quadratic. It is even further reduced with LDF, and linear scaling of the Z kernel is obtained for QRDF from LA80 to LA160.

For the hydrated formamide systems, scaling is again less favorable (Figure 7.13). Nonetheless, it still scales at $\mathcal{O}(N^3)$, i.e. one order of mangnitude less than the canonical



N_{AO}	DFCM	DFCAM	QRDF
490	—	—	—
970	2.14	1.69	1.80
1930	2.06	1.26	1.25
3850	2.03	1.11	1.11

Figure 7.11: Sparsity behaviour of the intermediate tensor **D** in the Z kernel for the first Laplace point.

	Coul.	Coul-erfc	QR
alkan20	28.81	28.76	28.85
alkan30	29.04	29.02	29.14
alkan40	28.38	28.38	28.50
FW15	23.31	23.29	23.27
FW21	23.27	23.26	23.26
FW30	24.30	24.20	24.10

Table 7.6: SOS-MP2 energy differences in μHartrees per occupied orbital compared to the canonical SOS-MP2 reference.

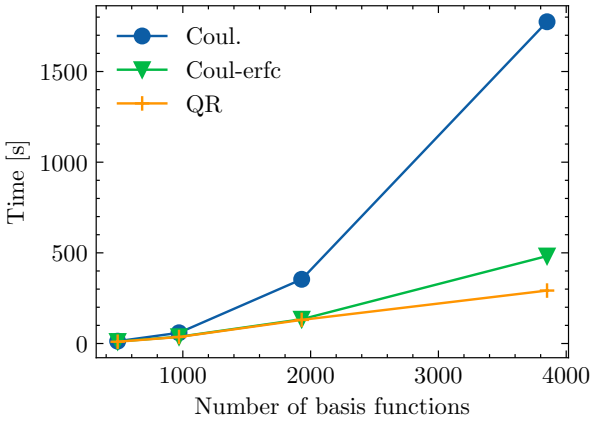
algorithm. As opposed to HF, LDF has no major impact on the prefactor. The last point for FW163 is missing for QRDF due to numerical issues.

Accuracy

The accuracy for the different density fitting approximations is given in Table 7.6, compared to the canonical SOS-MP2, in μHartrees per occupied orbital. For all methods, the Hartree-Fock reference is calculated with DFCM.

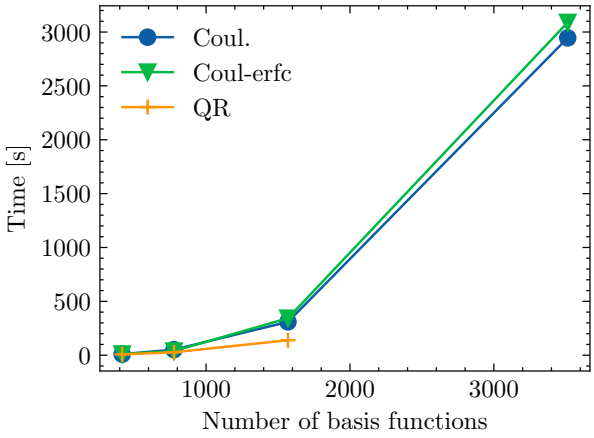
Immediately, it is apparent that the energy errors are much higher than for Hartree-Fock. This is due to the quality of the HF wave function, which was computed using density fitting. Results may be improved by using a larger auxiliary basis set to improve the description of the virtual space. The impact of the HF wave function on the SOS-MP2 energy is not explored here.

DFCM, DFCAM and QRDF all have similar errors. The SOS-MP2 energy is therefore much less sensitive to the density fitting approximation, because it is a non-iterative method and DFCAM can be used without problems in this case.



N_{AO}	DFCM	DFCAM	QRDF
490	—	—	—
970	2.45	1.96	2.09
1930	2.55	1.82	1.83
3850	2.00	1.59	1.00

Figure 7.12: Average wall times for the construction of the Z kernel and scaling coefficients (LA)



N_{AO}	DFCM	DFCAM	QRDF
490	—	—	—
970	2.51	1.88	1.88
1930	2.59	3.18	2.36
3850	3.26	3.18	—

Figure 7.13: Average wall times for the construction of the Z kernel and scaling coefficients (FW)

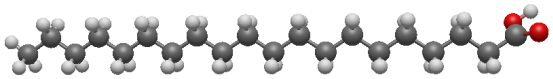


Figure 7.14: Structure of the linear carboxylic acids (LCA).

7.3 Atomic Orbital ADC

7.3.1 Molecular Test Systems

For the performance analysis of AO-ADC(2), again two molecular systems are chosen: linear carboxylic acids (LCA, Figure 7.14) and the hydrated formamide systems (FW) from the previous section. The LCAs were optimized using DFT/B3LYP and the 6-31G* basis set.

Two major factors influence the performance of local excited state methods: locality of electron correlation, and locality of the excitation space. The LCA systems form the best case scenario for both types of locality. The atomic ground state density becomes sparse very rapidly with increasing chain length, and so does the atomic transition density for the lowest lying singlet excitation. The excitation space for that transition is localized on the COOH group, and the number of elements in the transition density matrix scales with with $\mathcal{O}(1)$ in the limit of large systems. On the other hand, FW has a non-sparse AO ground state density, but a sparse AO transition density. Here, the first singlet excited state is localized on the formamide molecule. With increasing size of the solvation shell, the number of significant elements in the transition matrix grows slowly, and the excitation energy will eventually converge to a constant value in the limit of an infinitely large solvation shell [122], demonstrating the intensive property of excitations. As such, while FW is a worst case scenario for local ground state calculations, excited state methods are able to exploit the locality of the excitation in order to lower the computational cost.

7.3.2 Scaling

For analyzing the scaling of the AO-ADC(2) method, the cc-pVDZ basis set is used, with cc-pVDZ-ri for density fitting. Table **??** shows the number of basis functions for the test systems. Furthermore, only the performance of the first matrix-vector product is analyzed, using a (singlet) CIS-optimized transition matrix as the input. The CIS vectors are sparse, and the density of the ADC(2) trial vectors will be of similar sparsity.

It should be noted that it is very important to use CIS as the the initial guess, because a guess based on molecular orbital energies can be very dense.

For evaluating the doubles part of the MVP, the OV version of the algorithm will be used.

Figure 7.15 shows the performance of the matrix-vector product for the LCA systems. Again, three metrics are used: DFCM, DFCAM ($\omega = 0.1$) and QRDF ($T =$ 1e-5, $R = 40$). The atomic orbital formulation of ADC(2) drastically reduces the scaling from $\mathcal{O}(N^4)$ to $\mathcal{O}(N^2)$ for standard density fitting. Using local density fitting, the computational effort can be further pushed down to $\mathcal{O}(N^{1.5})$ and $\mathcal{O}(N^1)$ for DFCAM and QRDF respectively. The cross-over is very early, and sub-quadratic scaling is already reached at LCA40. The performance of AO-ADC(2) is therefore even better than the predicted $\mathcal{O}(N^2)$ scaling. This discrepancy is due to the sparsity of the AO transition matrix: during the sparsity analysis, $\mathcal{O}(N^1)$ scaling was assumed, analogous to the ground state density. However, the transition density scales with $\mathcal{O}(1)$ in the asymptotic limit, which leads to sub-quadratic performance.

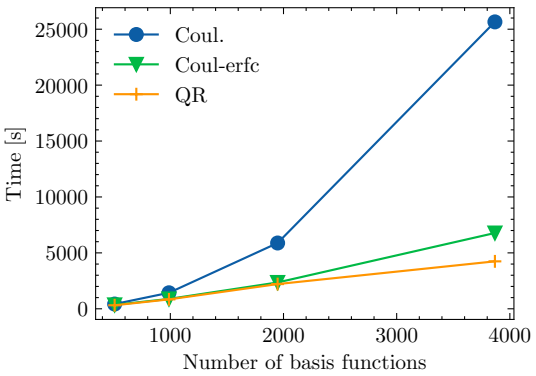
Figure 7.17 shows the total wall time for each individual component of the MVP calculation. The evaluation of the intermediate matrices (intermeds) and the doubles-part (2E) are the most expensive steps. The computational timings for computing part 2C, 2D and the CIS Fock matrices (jk) are one order of magnitude lower. Part 2A and 2B only involve a single matrix multiplication of the MO transition matrix with the intermediate matrices, and are therefore evaluated very quickly. The Cholesky decompositions also do not considerably influence the total scaling.

Concerning the memory footprint of AO-ADC(2), the same 3-index tensors that appeared in the evaluation of the Hartree-Fock and SOS-MP2 ground state energy also appear here: $B_{X\mu\nu}$ (the 3c2e integrals or the QRDF fitting coefficients), $C_{X\mu\nu}$ ($M_{XY}B_{Y\mu\nu}$) and $B^{(\alpha)}_{X\underline{i}\overline{a}}$ ($P^{(\alpha)}_{\mu\mu'}B_{X\mu'\nu'}Q^{(\alpha)}_{\nu'\nu}$. Their sparsity was discussed in detail in the previous section. Additionally, the following tensors need to be stored in memory: $C'_{X\mu\nu}$ ($G_{XY}B_{Y\mu\nu}$ in the K kernel during evaluation of the intermediates, see Algorithm 2), and the intermediate Laplace tensors in the Cholesky MO basis ($B^{(\theta)}_{X\underline{i}\overline{a}}$, $R^{(\theta)}_{X\underline{i}\overline{a}}$, $D^{(\theta)}_{X\underline{i}\overline{a}}$), as defined in the previous chapter in Algorithm 6. The tensors will be abbreviated as C', B_{MO}, R_{MO} and D_{MO} for this discussion, and B_{AO} is used for $B_{X\mu\nu}$.

7.18 shows the block sparsity of those tensors for QRDF, with B_{AO} as reference points. Block sparsity is defined as the number of significant blocks divided by the total number of blocks in the dense tensor. C' and B_{MO} are very dense, with a block sparsity slightly below 10%. The intermediate Laplace tensors D_{MO} and R_{MO} decay much faster than B_{AO} which was shown to scale with $\mathcal{O}(N^N)$. For LCA160, I_{MO} and R_{MO} are an order of magnitude sparser ($\approx 0.1\%$) than B_{AO}. The difference to the other Laplace tensor B_{MO} is that I_{MO} and R_{MO} are formed by the contraction of B_{AO} with both the AO ground state density $\mathbf{P}$ and the AO transition matrix $\mathbf{U}$, which explains their much higher degree of sparsity compared to B_{MO} which is formed using the AO ground state densities only. One peculiar thing to notice here is that the intermediate tensor D_{MO}, which is formed from B_{MO} and R_{MO}, is sparser than both of its input tensors. This indicates a potentially faster route to evaluating D_{MO} by imposing sparsity criteria on B_{MO} and R_{MO}. How these criteria exactly look like is subject of future investigation. The most plausible route goes via a sparsity analysis of the trial vector u, similar to how LMO or NO excited state methods generate a compact virtual orbital space.

LCA			FW		
Molecule	Abbrev.	N_{AO}	Molecule	Abbrev.	N_{AO}
$H_{41}C_{20}O$	LCA20	508	$H_{33}CNO_{16}$	FW15	417
$H_{81}C_{40}O$	LCA40	988	$H_{63}CNO_{31}$	FW30	777
$H_{161}C_{80}O$	LCA80	1948	$H_{129}CNO_{64}$	FW63	1569
$H_{321}C_{160}O$	LCA160	3868	$H_{291}CNO_{145}$	FW144	3513

Table 7.7: Total number of basis functions for linear carboxlyic acids (LCA) and solvated formamide (FW) with the aug-cc-pVDZ basis set



N_{AO}	DFCM	DFCAM	QRDF
508	—	—	—
988	1.8	1.4	1.5
1948	2.1	1.5	1.4
3868	2.2	1.5	1.0

Figure 7.15: Performance of a single matrix-vector-product with a CIS optimized (singlet) trial vector, for linear carboxylic acids.

Similarly, the evaluation of the intermediate tensors can be sped up by only taking into the account the AOs which are near or within the excitation space. This would however lead to a loss of state specificity of the AO-ADC(2) MVP.

Figure 7.19 shows the performance of AO-ADC(2) for solvated formamide. Although the density of the systems negatively impacts performance compared to LCA, it is still possible to achieve sub-cubic scaling thanks to the sparsity of the AO transition matrix. The point for FW144 with DFCAM is missing due to numerical issues that were encountered during the calculation. Nonetheless, the graph speaks for the success of the AO-ADC(2), even for non-ideal systems, provided that the excitation space is small. Figure 7.20 shows the block sparsity of the tensors discussed in the previous paragraphs for LCA. Although most tensors are very dense, the intermediate D_{MO} almost falls below the 0.1% mark. Similarly to LCA, this sparsity might be used for imposing conditions on the other tensors, and speeding up the calculations for all components of the AO-ADC(2) vector.

7.3.3 Accuracy

To get an impression on the accuracy that can be achieved with AO-ADC(2), the lowest lying singlet excitation energies for a small set of molecules are compared to the exact results obtained with canonical SOS-ADC(2), as implemented in Q-Chem. The Hartree-Fock wave function is optimized using standard density fitting without any local approximations as the reference for AO-ADC, while the exact HF ground state is used for the canonical calculation. The aug-cc-pVDZ basis set, and the auxiliary basis sets cc-pVTZ-jkfit

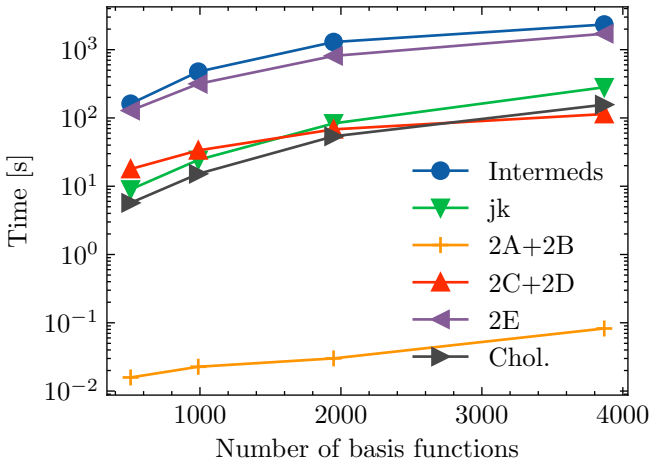


Figure 7.16: Figure

Figure 7.17: Total time needed to evaluate each separate component of the MVP using quasi-robust density fitting (linear carboxylic acid)

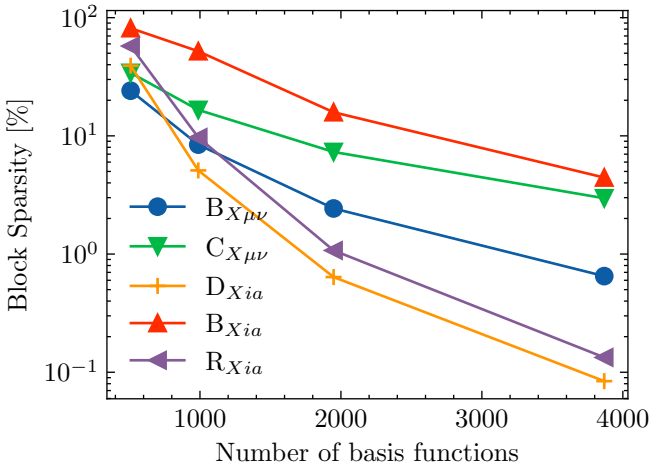
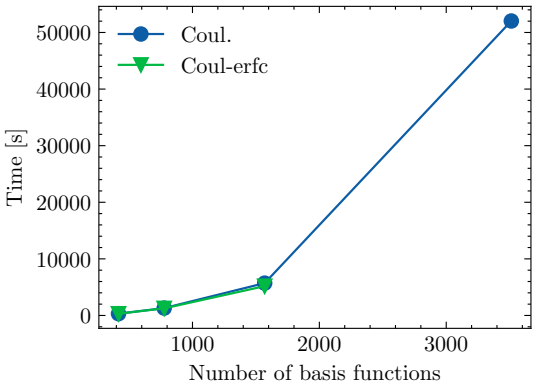


Figure 7.18: Block sparsity for the major 3-index tensors appearing in the evaluation of the MVP (linear carboxylic acid)



N_{AO}	DFCM	DFCAM
508	—	—
988	2.3	2.3
1948	2.1	2.0
3868	2.74	—

Figure 7.19: Performance of a single matrix-vector-product with a CIS optimized (singlet) trial vector, for hydrated formamide.

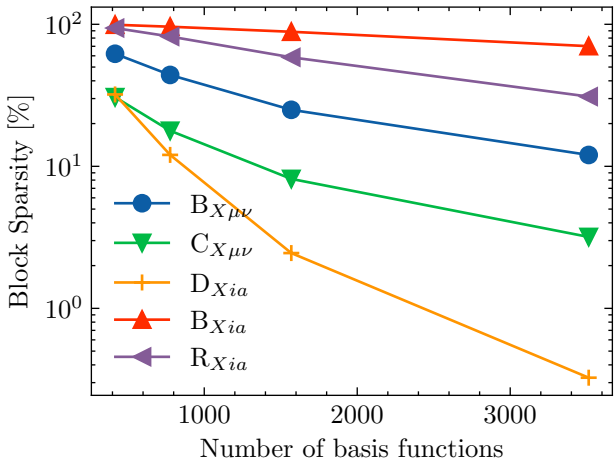


Figure 7.20: Block sparsity for the major 3-index tensors appearing in the evaluation of the MVP (hydrated formamide)

and aug-cc-pVDZ-ri are used for Hartree-Fock and AO-ADC(2) respectively. The test systems include linear alkanes and carboxylic acids, solvated formamide, as well as the borondipyrromethene-flavin dyad (FLVA) [209] and the phenothiazine-isoalloxazine dyad (DYAP) [193]. The structures of FLVA and DYAP are given in Figure **??**.

Before considering the results, it is important to note that convergence issues were commonly encountered in the Davdison procedure when using local density approximations. This is due to linear dependencies in the auxiliary basis set space, which then leads to numerical issues in the local density fitting procedure. For DFCAM, the matrix of the 2c2e integrals in the coulomb-attenuated metric, i.e. $(X \mid Y)_\omega$ needs to be inverted. This can be done by solving the eigenvalue problem, then scaling the eigenvectors by the inverse of the eigenvalues. However, for a linearly dependent basis set, the eigenvalues are very small, which leads to large entries in $(X \mid Y)_\omega$, and a loss of accuracy. The error propagates through the Davidson iterations and causes convergence issues in the form of negative excitation energies. Fortunately, the problem can often be solved by filtering out all eigenvectors with an associated eigenvalue below a certain threshold (typically around 1e-6 to 1e-4). In the quantum chemistry community, this procedure is known as *canonical orthogonalization*, and gives an approximation to the exact matrix inverse. Alternatively, the problem can be solved by removing the linear dependencies from the basis set itself (see Annex **??**). Finally, a less diffuse basis set can be used as well. For example, instead of using aug-cc-pVDZ-ri, one may use cc-pVTZ-ri.

However, none of these methods worked for the "strict" QRDF. Due to the harsh screening procedure for the test functions mentioned in the previous sections, many matrix entries of the rectangular matrix $(Q_{test} \mid P_{fit})$ will be very small. This leads to numerical issues when solving the linear-least squares problem, even with linear dependencies removed. Here, an alternative QRDF algorithm is used (see **??**), where the test functions are chosen by a weighted average criteria using the auxiliary overlap matrix, with the parameters T = 1e-4 and S = 1e-4. Numerical issues can be completely avoided, as a QR decomposition is more robust than matrix inversion. (TO DO: SHOW SCALING)

Table **??** shows the SOS-ADC(2) excitation energy differences in μHartrees per occupied orbital. Values are given for DFCM, DFCAM with attenuation factors 0.1 and 1.0,

System	Coul.	Coul-erfc (0.1)	Coul-erfc (1.0)	QRDF (new)
LCA12	3.48	0.11	2.02	2.70
LCA20	1.52	4.01	22.66	1.32
LA20	0.90	6.29	1.24	0.50
FW10	0.09	6.37	0.70	0.20
FW15	0.49	0.62	0.86	0.10
FLVA	0.26	$0.04^{(a)}$	$5.25^{(a)}$	0.26
DYAP	1.77	27.25	94.89	1.76

Table 7.8: Difference in excitation energy between canonical SOS-ADC(2) and CD-DF-SOS-ADC(2), in μHartrees per occupied orbital, for different density fitting approximations. LA = linear alkane, LCA = linear carboxylic acid, FW = solvated formamide, flva(a) = borondipyrromethene-flavin dyad, iso = phenothiazine-isoalloxazine dyad. $^{(a)}$ The cc-pVTZ-ri auxiliary basis set was used instead of aug-cc-pVDZ-ri due to convergence problems.

and the improved QRDF algorithm. DFCM shows acceptable accuracy, on the order of several μHartrees for the linear systems LCA12, LCA20 and LA20. The errors are even lower for the other test systems. As expected, DFCAM introduces much larger errors, similar to Hartree-Fock. The iterative nature of the Davidson procedure therefore has a significant impact on accuracy for DFCAM. The largest errors are again observed for linear molecules, especially for DFCAM(1.0) where errors for LA20 are almost two orders of magnitude larger. Errors are much smaller for QRDF, and even problematic systems like LA20 have similar accuracy to DFCM, again showing the superiority of the quasi-robust density fitting scheme. Only a single set of parameters for QRDF was used for testing accuracy. A more extensive benchmark, as well as the impact of diffuse basis sets on scaling, are subjects for further investigation.

7.3.4 Large Molecules: Challenges and Limitations

An attempt was made to apply CD-DF-SOS-ADC(2) to chemically interesting molecules with 10,000 basis functions or more, with limited success: even if the atomic orbital formulation can exploit sparsity to reduce the memory footprint and scaling, the need for diffuse basis functions practically nullifies any advantages it provides. The block occupation with aug-cc-pVDZ or def2-SVPD is still above 50% for non-linear molecules with $\approx$ 300 atoms, and the very large auxiliary basis sets often lead to OOM (out-of-memory) errors due to data duplication in the QRDF algorithm and other routines. For diffuse auxiliary functions, processes often need to hold the whole $(X \mid Y)$ matrix in memory while solving the linear least squares problem. For multiple ranks on a single node, this can quickly lead to memory problems. While memory duplication can be reduced by simply using fewer ranks per node, it should be noted that currently the DBCSR tensor library is only optimized for one MPI rank per CPU. Furthermore, each rank can only allocate approximately 2GB of memory at once, because only 4-byte integers (with maximum value 2,147,483,648) are used to specify the size of the memory window that is allocated. Moreover, the need to reorder the tensors before each contraction effectively doubles the space needed by each tensor (see also the discussion

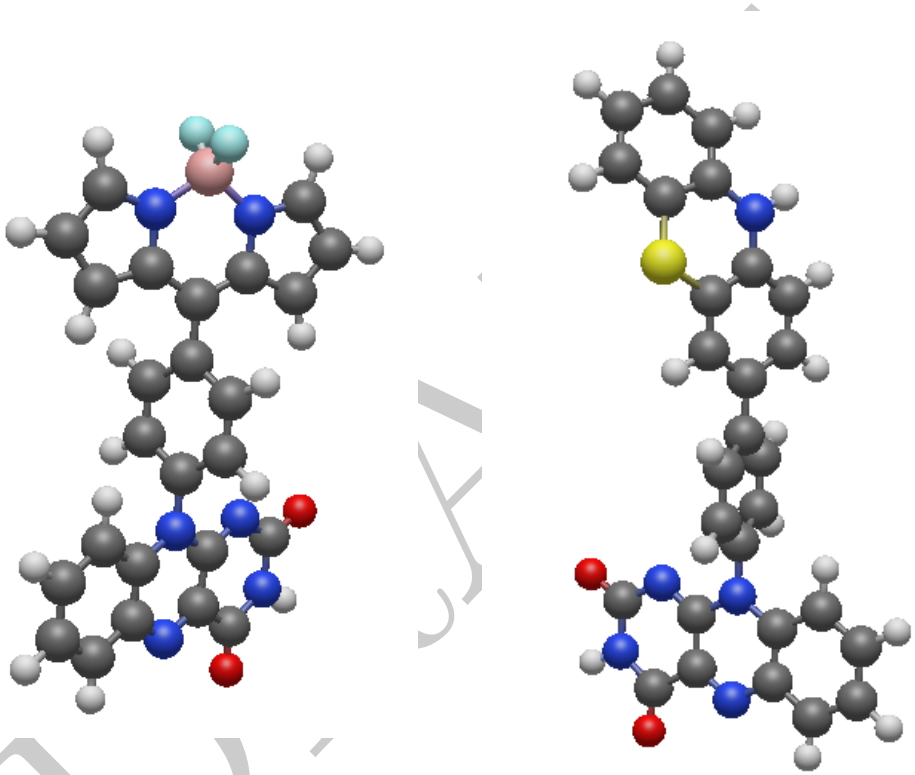


Figure 7.21: Molecular structure of borondipyrromethene-flavin (a) and phenothiazine-isoalloxazine (b)

in section 9.3.2).

In ground state methods, OOM errors can often be solved by using *direct* methods where tensors are recomputed on-the-fly. By increasing the number of batches, the memory footprint can be lowered quite considerably. A direct version of CD-DF-SOS-ADC(2) could also be considered, although the need to recompute tensors like $C_{X\mu\nu}$, $R_{X\bar{i}\bar{\nu}}$ or $B_{X\bar{i}\bar{\nu}}$ multiple times would lead to a much higher prefactor which further pushes the low-scaling regime to larger molecular sizes.

7.4 Summary and Outlook

An atomic orbital formulation of the spin-opposite-scaled algebraic diagrammatic construction method can drastically reduce the formal quartic scaling of canonical, density-fitted SOS-ADC(2). This method, named CD-DF-SOS-ADC(2), can successfully exploit the sparsity of the AO ground state density and the AO transition density. Furthermore, local density fitting significantly reduces the overhead, and the low scaling regime is more easily reached. For linear carboxylic acids, the method scales linearly, and even electron-dense systems, like hydrated formamide, were shown to have sub-cubic scaling if the transition density is sparse. The 3-index tensors necessary for an AO-ADC(2) calculation show a high degree of sparsity, which, when using block-sparse matrix storage, significantly reduces the memory footprint. Furthermore, the high prefactor of AO-ADC(2) can be mitigated using local density fitting, making it somewhat competitive with canonical density-fitted SOS-ADC(2) for less sparse systems, although with a larger memory footprint.

Accuracy was shown to be within acceptable range for standard density fitting and quasi-robust density fitting. Convergence issues were encountered due to linear dependencies with the auxiliary basis sets, which can be addressed by removing certain basis functions, by canonical orthogonalization of the metric inverse, or by just using another basis set.

Unfortunately, the need for augmented basis sets negatively impacts the scaling of CD-DF-SOS-ADC(2) and delays the onset of the linear-scaling regime. Compared to NO methods, CD-DF-SOS-ADC(2) includes all atomic orbitals and molecular orbitals, as opposed to only the ones close to the excitation region, making it more computationally expensive, although it has the advantage of not being state-specific.

Pushing the current implementation of CD-DF-SOS-ADC(2) to system sizes with a large number of basis functions (> 5000 basis functions) quickly leads to memory problems, even on distributed systems. This, in combination with diffuse basis functions and molecular systems that are not strictly linear in nature, as well as the increased algebraic complexity of the working equations, means that CD-DF-SOS-ADC(2) cannot really reach the low-scaling regime except in very specific cases. Introducing similar pre-screening techniques to NO methods based on some CIS or CIS(D) density might be beneficial. Moreover, a less memory-intensive tensor library should also be considered. However, at the moment, there are no other tensor libraries besides DBCSR that are well suited. Further development of sparse block tensor libraries are crucial for the future of atomic orbital based methods.

The equations for computing triplet excitations have not yet been implemented, but similar scaling and accuracy to singlet excitations are expected.

Part II

Chapter 8

Parallel Computing

The popularity of computational chemistry can be attributed in no small part to the advances and devlopment of highly efficient algorithms in theoretical chemistry. Equally important however is the ever increasing accessibility and performance of computing resources: commercially available work stations can handle chemical systems which could only be modelled on supercomputers a couple decades ago, and firmly cemented the position of computational chemsitry as an important "experimental" tool in the toolbox of a chemist.

As the speed of computers increased over the years, so did the complexity of their components. Nowadays, programmers can choose between several types of architectures, such as shared or distributed memory systems, or accelerators like GPUs. Knowing the strenths and weaknesses of each type is paramount to developing efficient algorithms and tackling larger molecular systems.

This chapter gives an overview on computer architecture, and the different types of parallelism encountered on modern hardware.

8.1 Moore's Law

Moore's Law states that the transistor density in integrated chips doubles every 12 to 24 months. First formulated in 1965 by Gordon More, his prediction has held up fairly well over the years. However, the technology enabling this trend has changed over the years.

Figure ... shows the trends in clock speed, single-thread performance, power consumption and number of logical cores and transistors for microprocessors from 1970 to 2000. Since the early 2000s, clock-speed and single-thread performance have begun to plateau, and have stagnated from 2010 onwards. Increasing the clock speed to values beyond 4 to 5 GHz generates too much stress on the microchip in form of heat, and decreases its performance. This flaw was compensated by using the growing transistor density to instead increase the number of logical cores on a single chip.

Shifting towards increasing core count however entails that the most ideal perfpormance of a CPU can only be achieved though parallel programming. With the rise of parallel computing, the number of different parallel hardware features drastically increased, and it can be difficult for programmers to fully exploit the available computing resources. Moreover, different programming language and compiler extensions have emerged alongside, with numerous competing standards, especially for GPUs.

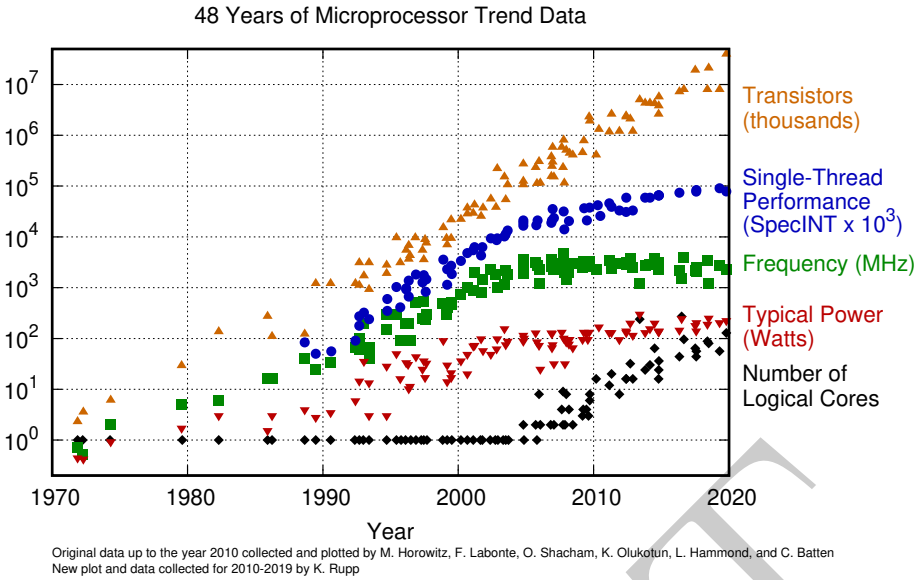


Figure 8.1: Taken from `https://github.com/karlrupp/microprocessor-trend-data`

8.2 Benefits and Limits of Parallel Computing

While the different available programming models can seem daunting at first, one of the major advantages of parallel computing is improved *scalabilty*. An application that exposes parallelism can be sped up by several orders of magnitude, simply by adding more compute power, with several different architectures to choose from. The limit of what problem sizes can be tackled is mostly dictated by the *amount* of available computing resources and storage, rather than individual processor charcterstics.

As important as parallel computing has become in recent years, there is a reason why increasing clock speed was seen as the foremost strategy in keeping Moore's law alive. First, modifying a serial program to exploit parallelism can be a time-consuming endeavour, and second, not all tasks can be effectively parallelized. This means that the potential amount of seed-up is limited by the amount of parallel code. This is known as *Amdahl's law*. The speed-up for a number of cores N_c is given by

$$Speed - Up(N_c) = \frac{1}{S + \frac{P}{N_c}} \tag{8.1}$$

where S is the fraction of serial code and P is the fraction of parallel code. The speedup for a fixed-size problem as a number of cores is known as *strong scaling*, and the time-to-solution on each indivual core *decreases* when more cores are added.

An alternate way to compute potential speed-up is given by Gustafson-Barsis's Law

$$Speed - Up(N_c) = N_c - S(N_c - 1) \tag{8.2}$$

where the problem size also increases proportionally to the number of cores. The scaling for this trend is known as *weak scaling*. In this scenario, the time-to-solution spend on each core remains constant, as the system size and number of cores increases. Even if this type is called "weak", both forms of scaling are equally important, as they adress different scenarios.

8.3 Types of Parallelism and Memory Hierarchy

Nowadays, a programmer has access to four categories of parallelism:

1. vectorization
2. thread-based parallelism
3. process-based parallelism
4. streaming

Leveraging the power of each type requires some understanding of the underlying hardware.

Figure 8.2 shows the major components and memory pathways in a modern computing cluster. A *cluster* is a collection of individual computers that work together and form a single unit. Individual computers are also called *nodes* and occupy a single rack (or "shelf") each in a large server cabinet. The nodes are connected via a low-latency, high through-put network, e.g. ethernet cables to enable inter-node communication. Each node contains one or more central proecessing units (CPU) and optionally one or more graphical processing units (GPU). Systems where different types of hardware architecture are mixed are also known as *heterogenous* systems. The individual components are fixed on a *motherboard*: CPUs are plugged into *sockets* and GPUs into *PCIe slots*. A CPU is composed of one or more cores, where the actual processing of data is carried out. A GPU is also composed of multiple cores, which are grouped into independent *streaming multiprocessors* (SM -¿ nvidia CU (compute units OpenCL, subslices).

Memory is also a crucial component of computer architecture and is a limited resource. The speed at which data is read from memory can also become a major bottle-neck: No matter how fast a processor is, if the feed rate is too low, it cannot reach its peak performance becuase it wastes cycles while waiting for data to arrive. To optimize data through-put, the memory model in modern computer architecture requires a complex hierarchy, with different sizes and speeds. Computer memory at the top of the hierarchy has a high response rate, but low complexity. It is also very expensive to produce and therefore much smaller. At the bottom of the hierarchy is memory with large storage space and capable of complex tasks. It is cheap but has low reponse rate. Over the years, the number of levels in the memory hierarchy has increased. Most modern computers have six levels: CPU registers, L1 chache, L2 chache, L3 chache, DRAM and disk.

CPU registers sit at the top of the hierarchy and are closest to the cores. It is the region of memory where data is directly manipulated by arithmetic operations and machine code. Its size is typically on the order of several tens to hundreds of bytes. Data is loaded into the registers from *cache*, a region of memory which is further subdivided into different levels named L1, L2 and L3. Each core has its own L1 cache, but might share L2 and L3 cache with other cores. Caches have different sizes and speeds, with L1 being the fastest and smallest at several tens of kB, and L3 being the slowest and largest at several tens to hundreds of kB. Memory is transferred from L3, to L2, to L1 and finally to the CPU registry. The reason why there are multiple levels of chache is to reduce *cache misses*. If data requested by the core is not found in L1, then L2 is searched, then L3. A cache miss is an event where the data is not found anywhere in chache. In that case, a request has to be put out to the dynamical random access memory (DRAM) to retrive

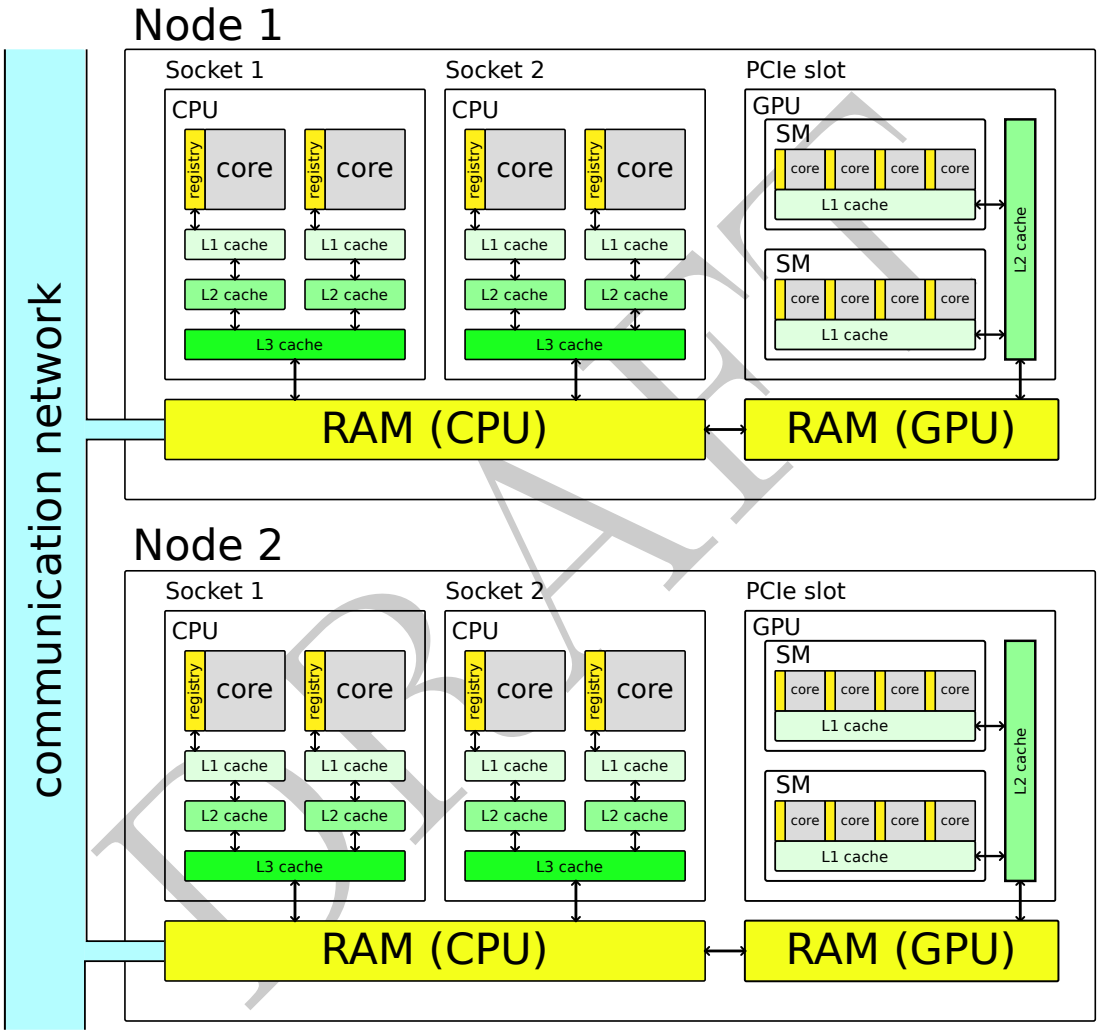


Figure 8.2: Memory hierarchy

Release	Vector Length (bit)	No. Registers
SSE (Streaming SIMD Extension)	128	8
SSE2/SSE3/SSE4	128	16
AVX/AVX2 (advanced vector instructions)	256	16
AVX512	512	32

Table 8.1: Adpated from ...

data. Modern techniques such as *chache prefetching* can minimize the amount of cache misses by loading the data into higher chache levels before it is actually needed by the lower levels.

The speed of DRAM is 10 to 100 time slower than cache, but much larger in size. It is the main memory pool and shared by all cores. CPUs and GPUs have separate DRAM regions which communicate via a PCIe bus. DRAM sizes vary drastically, and are on the order of 10^0 to 10^1 for GPUs and 10^0 to 10^3 for CPUs. Data can be transferred from one node to another via CPU DRAM through the communication network, with transfer rates on the order of several GB/s. For programs which are not reading from disk, this is weakest link in the memory hierarchy ???

8.4 Vectorization

Vectorization is the pocess of operating on multiple variables at the same type. This type of parallelism is encountered at the highest level of the memory hierarchy introduced in the previous section, i.e. CPU registers. Each core has multiple registers, also called *vector registers* with a certain size or *vector length*. Instead of loading each inidividual element from cache and operating on it, one vector operation on a range of elements can replace multiple single operations. For a 512-bit register, two vectors with 8 floats (32 bit) can be summed within one cycle instead of eight (Figure ...). This type of parallelism is also known as single instruction multiple data (SIMD).

The length of vector registers, number of registers as well as the number of supported vector operations have greatly expanded over the years (Table 8.1).

8.4.1 Parallel SAXPY using vectorization

To show how vectorization can be introduced into a program, consider the following vector operation:

$$y \leftarrow \alpha x + y \tag{8.3}$$

where y, x are vectors of equal length, and α is a scaling factor. The vector operation 8.3 is also known as "saxpy" for single precision, and "daxpy" for double precision. A naive implementation of the saxpy-kernel is given in Listing 8.4.1 for vector size N

Listing 8.4.1: Parallelization-unaware implementation of saxpy (`listings/saxpy_nopara.c`)

```
1  #define ARRAY_SIZE 1024
2
3  static float x[] = {[0 ... ARRAY_SIZE] = 1.0};
4  static float y[] = {[0 ... ARRAY_SIZE] = 2.0};
5  static float a = 3.0;
6
7  int main() {
8
9    for (int i = 0; i < ARRAY_SIZE; ++i) {
10     y[i] += a * x[i];
11   }
12
13 }
```

Two arrays are allocated, x and y, and all their entries set to 1 and 2 respectively. Each element is updated individually in the for loop. There are several possibilities to introduce vectorization:

1. auto-vectorization
2. compiler directives
3. intrinsic functions
4. optimized libraries

Auto-vectorization is by far the easiest approach: the compiler automatically recognizes that the loop can be vectorized and generates optimized machine code that uses vector instructions. This does not require any input from the user. Compiling the code on a machine with AVX support, using the GNU C compiler, and passing the compiler flags "-O2 -march=native -ftree-vectorize -fopt-info-vec-optimized" generates the following report:

```
saxpy_nopara.c:9:3: optimized: loop vectorized using 32 byte
   vectors
```

which indicates that the vectors are loaded and operated on in 32 byte chunks, or 8 floats at once. The flag "-ftree-vectorize" (or alternatively "-O3") activates auto-vectorization and "-fopt-info-vec" generates the report. To make sure that the C compiler uses the right vectorization relase, the flag "-march=native" is needed, or else the comiler might fall back to SSE.

In some cases, auto-vectorization cannot take place because the compiler did not recognize that the loop can be vectorized. It can then be beneficial to use *intrinsic functions*. Intrinsic functions are compiler-dependent functions that map to processor

operations. When targeting an AVX architecture with 256-bit registers, the saxpy kernel can be rewritten as

Listing 8.4.2: SAXPY using intrinsics (`listings/saxpy_intrinsic.c`)

```
1  #include <immintrin.h>
2  #define ARRAY_SIZE 1024
3
4  // attribute needed for alignment, misalignment leads to
       errors
5  static float x[] __attribute__ ((aligned(8*ARRAY_SIZE))) = {[0
       ... ARRAY_SIZE] = 1.0};
6  static float y[] __attribute__ ((aligned(8*ARRAY_SIZE))) = {[0
       ... ARRAY_SIZE] = 2.0};
7  static float a = 3.0;
8
9
10 int main() {
11
12   __m256 a_vec, x_vec, y_vec, r_vec;
13
14   // set each entry of a_vec to a
15   a_vec = _mm256_set1_ps(a);
16
17   int stride = 8;
18   for (int i = 0; i < ARRAY_SIZE; i += stride) {
19     // load values into registers with appropriate offset i
20     x_vec = _mm256_load_ps(&x[i]);
21     y_vec = _mm256_load_ps(&y[i]);
22
23     // perform saxpy: (1) multiply, (2) add
24     r_vec = _mm256_add_ps(_mm256_mul_ps(a_vec, x_vec), y_vec);
25
26     // copy results back to y
27     _mm256_store_ps(&y[i], r_vec);
28   }
29
30 }
```

It is apparent that using intrinsics makes the program much more complex. The arrays cannot be fed directly to the functions, but need to be loaded into vectors of type `__mm256`, using "set" or "load" functions. Furthermore, the data needs to be aligned correctly using `__attribute__ ((aligned(...)))`. The arrays are then loaded in chunks into the registers and given to the vector functions for multiplying and adding.

The major problem with using intrinsic functions, besides increased complexity, is *portability*. The code in 8.4.1 does not compile on machines that do not support AVX, and is limited to 256-bit registers even on AVX-512 machines. Portable alternatives include using compiler directives or optimized libraries.

Directives (or "pragmas") are hints that can be given to the compiler that suggest that the loop might be vectorizeable. By far the most popular set of compiler directives that provide vectorization capabilities is undoubtedly included in the OpenMP application programming interface (API). The OpenMP API is standardized accross all compilers, making it highly portable. The OpenMP directives greatly simplify the SAXPY program:

Listing 8.4.3: SAXPY using compiler directives (`listings/saxpy_simd.c`)

```
1  #define ARRAY_SIZE 1024
2
3  static float x[] = {[0 ... ARRAY_SIZE] = 1.0};
4  static float y[] = {[0 ... ARRAY_SIZE] = 2.0};
5  static float a = 3.0;
6
7  int main() {
8
9    #pragma omp simd
10   for (int i = 0; i < ARRAY_SIZE; ++i) {
11     y[i] += a * x[i];
12   }
13
14 }
```

Simply plopping the directive in front of the for loop takes care of generating the appropriate machine code for the targeted architecture.

The last way to introduce vectorization is via external programs, such as the basic linear algebra subprograms (BLAS) library. It provides a set of specific functions for performing basic vector and matrix operations. Similar to OpenMP, it only provides spcifications, and the direct implementation is compiler-dependent. In the BLAS routines, there is a saxpy functions available that can be called directly. It has the advantage of completely removing the loop and clearly states what operation is performed.

Listing 8.4.4: SAXPY using BLAS (`listings/saxpy_blas.c`)

```
1  #include <cblas.h>
2  #define ARRAY_SIZE 1024
3
4  static float x[] = {[0 ... ARRAY_SIZE] = 1.0};
5  static float y[] = {[0 ... ARRAY_SIZE] = 2.0};
6  static float a = 3.0;
7
8  int main() {
9          cblas_saxpy(ARRAY_SIZE, a, x, 1, y, 1);
10 }
```

External libraries can however be associated with a steeper learning curve depending on the complexity of function signatures.

8.5 Thread-based Parallelism

Vectorization is limited to single cores only. For multicore processing, it is important to understand the concept of processes and threads. Processes are executing instances of programs and group related operating system resources together. These resources are exclusive (private) to the process which allocated them, such as system memory, file handles, I/O status information, scheduling information and accounting information.

A process spawns one or more *threads* (Figure 8.3). A thread is the smallest subset of a process that can be scheduled independently by the OS scheduler. Unlike processes, threads of the same process share resources. They can be seen as "light-weight" processes: start-up of individual threads and communication between threads is much faster. Each core executes only one thread at a time, but can quickly switch between different threads (or *contexts*). Threads with a higher *priority* are granted more CPU time than threads with lower priority. One a single-core processor, threads are only executed *concurrently*, i.e. they are paused and resumed at regular intervals depending on their priority. One multi-core processors, threads can be executed at the same time (multithreading), but concurrency is still needed if the number of threads exceeds the number of logical cores.

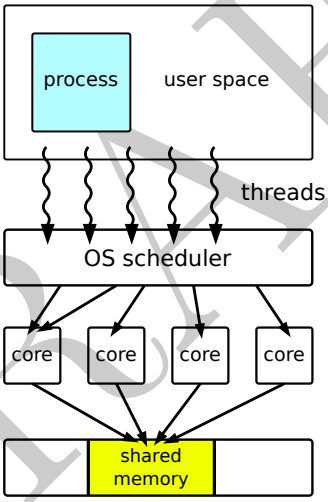


Figure 8.3: Shared memory parallelism

8.5.1 SAXPY using OpenMP

The most popular standard for thread-based parallelism (or *shared-memory parallelism*) is OpenMP, which was briefly discussed in the previous section. OpenMP was originally introduced to parallelize highly regular loops, but the standard has since then been greatly expanded and includes vectorization directives as well. Using pragmas, multiple threads can be spawned that execute the code within the parallel region. Lising ... shows an OpenMP parallel version of SAXPY. Each thread in the parallel region has a unique number associated with it which is used to divide up the arrays. Caution needs to be taken that threads do not operate on the same region at once, as this can lead to undefined behaviour ("race conditions").

Listing 8.5.1: SAXPY using OpenMP (listings/saxpy_omp.c)

```
#include <omp.h>
#define ARRAY_SIZE 1024

static float x[] = {[0 ... ARRAY_SIZE] = 1.0};
static float y[] = {[0 ... ARRAY_SIZE] = 2.0};
static float a = 3.0;

int main() {

  // launch threads
  #pragma omp parallel
  {
    // The code in this region is executed by ALL threads
    int threadnum = omp_get_thread_num();
    int numthreads = omp_get_num_threads();

    // lower and upper bound for this thread are determined
       using the thread number and number of threads
    int lb = ARRAY_SIZE*threadnum/numthreads;
    int ub = ARRAY_SIZE*(threadnum+1)/numthreads;

    for (int i = lb; i < ub; ++i) {
      y[i] += a * x[i];
    }

  } // end parallel region, synchronize

  // tasks share memory, and the result is visible here

}
```

Alternatively, the code may be more easily expressed by using a single compiler directive:

Listing 8.5.2: SAXPY using OpenMP (listings/saxpy_omp2.c)

```
#define ARRAY_SIZE 1024

static float x[] = {[0 ... ARRAY_SIZE] = 1.0};
static float y[] = {[0 ... ARRAY_SIZE] = 2.0};
static float a = 3.0;

int main() {

  #pragma omp parallel for
  for (int i = 0; i < ARRAY_SIZE; ++i) {
```

```
11      y[i] += a * x[i];
12    }
13
14  }
```

The `omp parallel for` directive also offers different scheduling tatctics which may be difficult to programm by hand. Listing 8.5.1 is an example of *static scheduling* the work is divided equally among threads *a priori*. However, in the case where each step requires different computational cost, *dynamic scheduling* offers a more balanced approach, although with a somewhat higher prefactor. Threads request tasks, or a chunk of multiple tasks, from the main thread, which distributes work on a first-come first-serve basis. There are many more directives available in the OpenMP standard with a lot of options for fine-tuning. For further details, the reader is referred to the official specifications (ref)

OpenMP is not the only option available for introducing shared-memory parallelism into a program. Alternatives include the `pthread` (POSIX threads) library for C, the `std::thread` library, or the Intel TBB (thread building blocks) library for C++. While these libraries can offer a more fine-grained control of threads, the learning curve can be very steep, and parallelizing an application can be very time-consuming. In contrast to OpenMP, they are not based on high-level compiler directives, but rather expose low-level functions and structures that create and control threads.

Shared-memory parallelism, as the name implies, is limited to cores sharing the same memory space. The primary disadvantage of this model is scalability. It becomes increasingly expensive to produce shared memory systems with more and more cores. Moreover, a larger number of cores implies heavier CPU traffic and time penalties due to *cache coherence*. Cache coherence is the mechanism by which uniformity of data is guaranteed and propagated among processes. There can be multiple copies of the same memory region in chache, which is operated on by different cores. Checking how to combine data from different caches can become costly for a large number of cores.

8.6 Process-based Parallelism

One of the most scalable aproaches in parallel computing is process-based parallelism, also known as distributed memory parallelism. Multiple copies of the same program run on separate processes, each with thier own local memory and instruction set. In contrast to threads, processes cannot exchange data via shared memory, but need to communicate via a network (Figure ...) using *message passing*. The increased cost of communication is often outweighed by the increased potential for scalability. Larger problems can be tackled simply ba adding more nodes.

Pure message-passing assigns or *binds* one rank to a single CPU core. In hybrid parallel approaches, a single process can be bound to a whole node, using shared-memory parallelism for intra-node communication and message passing for inter-node communication. It is also possible to spawn more processes than there are available cores: in that case, the processes are scheduled and excuted concurrently similar to threads.

The most popular standard for message passing is the message passing interface

(MPI). It defines a set of functions that allow processes to send and receive messages. There are different implementations of MPI, such as the open-source implementations OpenMPI, MPICH, or the vendor implementations Intel MPI or Cray MPI. MPI programs are run using a special start-up command:

```
mpirun -n <nprocs> ./myprogram.exe
```

The exact command and options depends on the MPI implementation. The start-up program is reponsible for duplicating the program on the different processes and establishing the communication network. Most implementations use the flag `-n` to pass the number of processes to be spawned.

All processes are characterized by their *rank*, a unique, portable identifier, normally an integer between $[0 : nprocs - 1]$. Only processes using the same *communicator* can exchange messages. A communicator is a special handle of type `MPI_Comm` and describes a group of processes. Communicators may also have different *topologies*, such as cartesian grids, or graphs, which restrict the flow communication between processes to nearest neighbours. Cartesian grids for example are best suited for matrix operations (cf. section ...). By passing topology information to MPI, it can optimize the runtime environemnt by renumbering tasks such that processes are physically closer to lower communivation overhead. By default, the topology is undefined, and any processor can exchange messages with all other processors.

8.6.1 SAXPY using MPI

Modifiying programs to exploit distrubuted parallelism requires significant effort, the most crucial point being how data is distributed over the processes. Consider again the SAXPY kernel. There are two memory models: either every process has a copy of the whole array, or the arrays are distrubuted in chunks over all processes. If data is duplicated on every rank, this can quickly exhaust memory resources for large datasets. On the other hand, if data is distributed in a non-ideal way, communation may incur major overhead. A programmer has to outweigh the benefits of data duplication and network communication.

Listing 8.6.1 shows a distributed memory approach to the SAXPY kernel

Listing 8.6.1: SAXPY using MPI (`listings/saxpy_mpi.c`)

```
1  #include <mpi.h>
2  #include <stdlib.h>
3  #include <stdio.h>
4  #define ARRAY_SIZE 1024
5
6  static float a = 3.0;
7
8  int main(int argc, char** argv) {
9
10   MPI_Init(&argc,&argv);
11   MPI_Comm comm = MPI_COMM_WORLD;
12
```

```
13   int rank, size;
14   // get information form communicator
15   MPI_Comm_rank(comm, &rank);
16   MPI_Comm_size(comm, &size);
17
18   // size needs to be a divisor of ARRAY_SIZE
19   if (ARRAY_SIZE % size != 0) exit(-1);
20   int num_local = ARRAY_SIZE / size;
21
22   // allocate local arrays on each process
23   float* x_loc = (float*)malloc(num_local*sizeof(float));
24   float* y_loc = (float*)malloc(num_local*sizeof(float));
25
26   // global arrays
27   float *x, *y;
28
29   if (rank == 0) {
30         // init data on rank 0
31         x = (float*)malloc(ARRAY_SIZE*sizeof(float));
32         y = (float*)malloc(ARRAY_SIZE*sizeof(float));
33         for (int i = 0; i < ARRAY_SIZE; ++i) {
34                 x[i] = 1.0; y[i] = 2.0;
35         }
36   }
37
38   // send to other processes
39   MPI_Scatter(x,num_local,MPI_FLOAT,x_loc,num_local,MPI_FLOAT
        ,0,comm);
40   MPI_Scatter(y,num_local,MPI_FLOAT,y_loc,num_local,MPI_FLOAT
        ,0,comm);
41
42   // perform operation on local arrays
43   for (int  i = 0; i < num_local; ++i) {
44           y_loc[i] += a * x_loc[i];
45   }
46
47   // gather data on rank 0
48   MPI_Gather(y_loc,num_local,MPI_FLOAT,y,num_local,MPI_FLOAT
        ,0,comm);
49   // print it out, store it ...
50
51   // clean up
52   free(x_loc); free(y_loc);
53   if (rank == 0) {
54           free(x); free(y);
55   }
56
57   MPI_Finalize();
```

```
58
59 }
```

The program starts by initializing the executing environment using `MPI_Init`, and defining the communicator handle. The communicator that groups all processes is called `MPI_COMM_WORLD`, and is a macro defined in the header `<mpi.h>`. If the programm is stand-alone, it is safe to use as the primary communicator. When writing a library that is used in combination with other MPI libraries, it is crucial to use a communicator that is passed from the main program. There are two reasons for this: (1) the user may want to use a subcommunicator, rather than the whole process group, and (2) MPI messages have a unique identifier (integer) and if another library uses the same identifier, erroneous communication can take place. In this example, the global communicator is suffcient.

Each process then allocates a local array of size `num_local` $<$ `ARRAY_SIZE`. In this example, the main data is intialized on rank 0, which is then split into equal chunks and send to the other processes' local array using `MPI_Scatter`. Each process then only needs to perform the SAXPY operation on its local chunk. Afterwards, the results are collected on rank 0 using the routine `MPI_Gather`. Data is then available on rank 0 to be further manipulated, written to console, etc.

It should be noted that the above program only works for a number of processes that is a divisor of `ARRAY_SIZE`, because `MPI_Scatter` and `MPI_Gather` can only handle chunks that are of equal size on each process. If that is not the case, one can use the more general routines `MPI_Scatterv` and `MPI_Gatherv` which also take the different chunk sizes as an input.

Furthermore, the local SAXPY for loop can be parallelized using the techniques described in the previous sections (vectorization, threads) in case where a process is bound to a whole node and has access to further compute resources.

This programm shows only a few of the many functions available for message passing. The MPI standard defines many more functions for sending, receiving, broadcasting, reduction, one-sided communication etc. The official document for the newest MPI standard (ref) is over a thousand pages long, and is updated regularly. Message passing has a very steep learning curve, and can be quite difficult to debug.

8.6.2 MPI and Shared Memory

One may come under the impression that MPI is not efficient on shared-memory systems, as data is exchanged via network-calls. However, MPI is optimized to recognize shared memory and data is then copied within DRAM rather than send through a comunication layer, making MPI competitive even on single nodes. Of course, there is still a slight overhead associated with it compared to thread-based parallelism.

MPI standards of version 3.0 and above furthermore introduce functions to allocate shared memory, also known as *windows*, for processes on the same node. Processes can then read/write to the same memory region without using network calls, much in the same manner as OpenMP. This approach can be seen as a MPI+MPI hybrid parallel approach rather than the more often used MPI+OpenMP approach. Hybrid parallelism is much more efficient than "MPI-everywhere" approaches due to reduced communication

overhead and easier load-balancing.

MPI+MPI has the advantage of only needing one programming model for both distributed and shared memory access, with relatively easy syntax. However, MPI+MPI is much lesser known, and there are is no automatic support for splitting up loops or atomic loads/stores to shared memory. An external library that offers auxiliary functions for MPI+MPI would be most beneficial.

8.7 Stream Processing

The last type of parallelism tackled in this chapter is *stream processing*. A stream is defined as a sequence of data that requires similar (low-level) computation. Streams are processed by *accelerators*, special hardware components with high data throughput, that complement the general purpose CPU by speeding up certain operations.

8.7.1 GPU Architecture

By far the most popular type of accelerators are graphical processing units (GPUs). As the name implies, GPUs were originally conceived to speed up graphics-related computation, but their use was later extended to include non-graphics workloads as well, in what is known as *general-purpose* graphic processing unit (GPGPU) programming. Nowaydays, computing clusters increasingly come with one or more GPGPUs included on each node.

The hardware archituecture design of modern GPUs is much more varied than that of CPUs. GPU vendors, like NVIDIA or AMD, have different hardware variations and use different terminology for similar components, and it can be difficult to abstract hardware as some vendor-specific features need to be omitted. Figure 8.4 shows a simplified block diagram of the most common GPU components. A GPU is composed of multiple compute units (CUs), also known as streaming multi-processors (SMs) in NVIDIA GPUs. Work is assigned to different CUs by a workload distributor. Each CU is further subdivided into an array of processing elements (PEs), or CUDA cores as referred to by NVIDIA. A single PE can operate on multiple data elements at once using SIMD or a variation known as single instruction multiple threads (SIMT) used in CUDA cores.

The overall performance of a GPU is given by the number of CUs and PEs, as well as the bandwidth of the PCIe land connecting CPU and GPU. Data transfer between the two devices is often the time-determinig step.

8.7.2 GPU Programming Model

The rise of GPGPU programming has given rise to a plethora of different programming languages and models. Low-level languages like CUDA, OpenCL or HIP directly reflect features of the targeted GPU hardware as part of their syntax, and the user is reponsible for managing each aspect of parallelism and data transfer. They require a very thorough understanding of how GPUs operate. Due to the complexity of "native" languages, higher-level languages have gained popularity over the years. Pragma based languages like OpenACC or OpenMP offer a much easier approach to GPU programming that ressembles shared memory parallelism on CPUs, and offer a higher degree of portability, at the cost of lower performance ceiling.

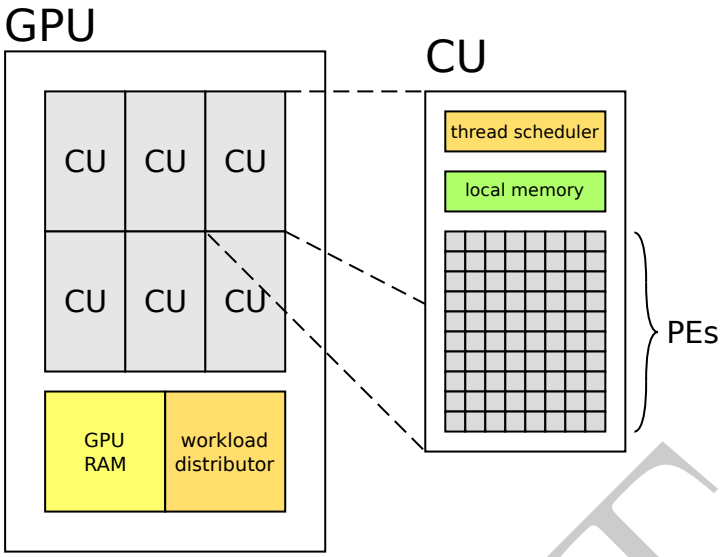


Figure 8.4: GPU architecture

Low level languages all operate on a similar programming model which allow the programmer to define special functions called *kernels*, which, when called on a GPU, are executed N times in parallel by N GPU threads. Rather than using for loops, kernels oparate on a whole range of data at once, known as a *grid* or NDRange. A grid can be one, two or three-dimensional to allow a more intuitive view of the data at hand. Each grid is subdivided into blocks or work groups which can be executed independently. The maximum size of a block is given by the number of threads in a CU. "Good" values are generally 128, 256 or 512, but the optimal size depends on the hardware and the problem addressed. Threads in a block share memory which is useful in case data from neighbouring subblocks is needed. Information on grid and block dimensions are needed by the kernels for distributing work over the GPU.

As an example, consider the SAXPY function using the CUDA programming language:

Listing 8.7.1: SAXPY using CUDA (`listings/saxpy_cuda.cu`)

```
1  #define ARRAY_SIZE 1024
2  #define BLOCK_SIZE 256
3
4  static float x[ARRAY_SIZE];
5  static float y[ARRAY_SIZE];
6  static float a = 3.0;
7
8  // the identifier __global__ indicates that it should be
       launched on the GPU
9  __global__
10 void saxpy(int N, float d_a, float* d_x, float* d_y) {
11         int i = blockIdx.x*blockDim.x+threadIdx.x;
12         if (i < N) d_y[i] = d_a*d_x[i] + d_y[i];
13 }
```

```
14
15 int main() {
16
17         // initialize memory on CPU
18         for (int i = 0; i < ARRAY_SIZE; ++i) {
19                 x[i] = 1.0; y[i] = 2.0;
20         }
21
22         // allocate memory on GPU
23         float *d_x, *d_y;
24         cudaMalloc(&d_x, ARRAY_SIZE*sizeof(float));
25         cudaMalloc(&d_y, ARRAY_SIZE*sizeof(float));
26
27         // copy data to GPU
28         cudaMemcpy(d_x, x, ARRAY_SIZE*sizeof(float),
              cudaMemcpyHostToDevice);
29         cudaMemcpy(d_y, y, ARRAY_SIZE*sizeof(float),
              cudaMemcpyHostToDevice);
30
31         // launch kernel
32         int nblocks = (ARRAY_SIZE + BLOCK_SIZE) / BLOCK_SIZE;
33         saxpy<<<nblocks,BLOCK_SIZE>>>(ARRAY_SIZE, a, d_x, d_y)
              ;
34
35         // copy back
36         cudaMemcpy(y, d_y, ARRAY_SIZE*sizeof(float),
              cudaMemcpyDeviceToHost);
37
38         // free up memory on GPU
39         cudaFree(d_x);
40         cudaFree(d_y);
41
42 }
```

CPU and GPU do not share memory. The first steps therefore involve allocating memory using `cudaMalloc` and copying data from CPU ("Host") to GPU ("Device") with `cudaMemcpy`. The arrays can then be passed to the kernel. The kernel function is indicated by the identifier `__global__`, and is executed at the same time by all threads in a block. Each thread has a unique block and thread id which indicate its position in the work group. This information is used to make sure that the thread is operating on the correct portions of the array. The GPU threads do not have any information of the loop extent, and it is the user's responsibility to make sure that threads do not write past the allocated memory window by introduing boundary checks (`if (i < N)`).

In this example for a 1D array, the kernel takes two launch parameters: the number of blocks, and the block size which are put between `<<< .... >>>` after the function call. Afterwards, the data can be copied back to the CPU for further processing.

The concept of operating relative to thread coordinates rather than using a for

loop with a single index can be quite confusing. Nowadays, there are directive-based compiler extensions, that allow to easily offload work to accelerators, such as OpenMP and OpenACC (Open ACCelerators). Listing 8.7.2 shows how a for loop can be easily modified using OpenACC

Listing 8.7.2: SAXPY using OpenACC (`listings/saxpy_acc.c`)

```
1  #define ARRAY_SIZE 1024
2
3  static float x[] = {[0 ... ARRAY_SIZE] = 1.0};
4  static float y[] = {[0 ... ARRAY_SIZE] = 2.0};
5  static float a = 3.0;
6
7  int main() {
8
9    #pragma acc data copy(x,y,a)
10   {
11     #pragma acc parallel loop
12     for (int i = 0; i < ARRAY_SIZE; ++i) {
13       y[i] += a * x[i];
14     }
15   }
16
17 }
```

OpenACC offers several pragmas for allocating and copying arrays to GPUs and back. By only adding two directives, the SAXPY code is easily ported to GPUs.

8.7.3 When to Use GPUs

Despite the success of GPU computing, GPUs only complement, rather than replace CPUs. GPU cores are very light-weight and have a limited instruction set compared to CPU cores, making them ill-suited for *task-based parallelism*, e.g. running an operating system with many different processes executing in the background. The strength of GPUs rather lies in *data-based parallelism*. Generally, any code that can benefit from vectorization can be sped up using GPUs.

Chapter 9

Into The Matrix

At its core, computational chemistry is basically matrix algebra applied to molecular systems. The performance of an algorithm, in other words its scaling, memory footprint and even accuracy, is intimately linked to how matrix operations and storage are handled internally. While the earliest implementations of molecular electronic structure methods often used hand-coded loops running over matrix indices, over the years, a whole ecosystem of matrix and tensor libraries has emerged in an attempt to lessen the burden of programmers, to various degrees of success, each with their strong and weak points. Choosing the best library for an algorithm is a non-trivial task, and depends on various factors: if memory is the bottle-neck, a sparse matrix library might be able to sufficiently compress data into a smaller space. If speed is the greatest concern, using targeting different archituectures like distributed memory systems, accelerators, or both, might be the solution. -.... The modern theoretical chemist needs to be aware of the possibilities ....

This chapter introduces basic concepts of matrix and tensor algebra, with a focus on storage orders. After introducing the core concepts, an (incomplete) list of matrix libraries is provided with small examples on how to use them.

9.1 Linear Algebra

Linear algebra is a crucial tool in many areas of computational science, such as image processing, machine learning, computational physics, computational biology and of course computational chemistry. The most important concepts are discussed in this section, and generalized to tensors. Until mentioned otherwise, zero-based indexing is assumed.

9.1.1 Matrices

A matrix is a rectangular, 2-dimensional array of dimension M by N, containing a set of complex or real numbers $\{a_{ij}\}$ with ordered subscripts $i = 0, 1, 2, ...M$ and $j = 0, 1, 2, ..., N$, of the form

$$\mathbf{A} = \begin{pmatrix} a_{00} & a_{01} & \dots & a_{0N} \\ a_{10} & a_{11} & \dots & a_{1N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M0} & a_{M1} & \dots & a_{MN} \end{pmatrix} \tag{9.1}$$

with M *rows* and N *columns*. *Row vectors* and *column vectors* are a type of matrix that contain only one row or one column, respectively.

Matrix multiplication is only defined between matrices of dimension M by K and K by N. The product yields an M by N matrix

$$\mathbf{C}_{M\times N} = \mathbf{A}_{M\times K}\mathbf{B}_{K\times N} \tag{9.2}$$

$$c_{ij} = \sum_k a_{ik}b_{kj} \tag{9.3}$$

A matrix with an equal number of rows and columns is called a *square matrix*. There a different types of square matrices:

1. *Diagonal matrices* only have non-zero entries on the diagonal

$$a_{ij} = a_{ii}\delta_{ij} \tag{9.4}$$

2. If all entries below the diagonal are zero, the matrix is *lower triangular*. Similarly, if all elements above the diagonal are zero, the matrix is *upper triangular*. Furthermore, if the diagional entries are all equal to 1, the matrix is *unit triangular*.

3. The *identity* or *unit* matrix $\mathbf{1}$ is defined as

$$\mathbf{A1} = \mathbf{1A} \tag{9.5}$$

$$\mathbf{1}_{ij} = \delta_{ij} \tag{9.6}$$

4. The *inverse* of a matrix $\mathbf{A}$ is defined as

$$\mathbf{A}^{-1}\mathbf{A} = \mathbf{1} \tag{9.7}$$

   It only exists for matrices with a non-zero determinant. If a matrix is non-invertible, it is also called *singular*.

5. A matrix is *unitary*, if its inverse is equal to its conjugate transpose

$$\mathbf{A}^{-1} = \mathbf{A}^\dagger \tag{9.8}$$

   A real unitary matrix is called *orthogonal*

6. A *hermititan* matrix is its own conugate transpose

$$\mathbf{A} = \mathbf{A}^\dagger \tag{9.9}$$

$$a_{ij} = (a_{ji})^* \tag{9.10}$$

   A real hermitian matrix is called *symmetric*

7. A hermitian matrix is said to be *positive definite* if all of its eigenvalues are positive, and *negative definite* if all eigenvalues are negative. *Semi-definite* matrices additionally have some eigenvalues which are equal to zero.

Decompositions?

9.2 Matrix Storage Formats

The main memory in a computer, i.e. DRAM, is a linear address space and can be seen as a single, contiguous, one-dimensional array. Two-dimensional arrays like matrices need to be *mapped* to this linear space by the compiler. This mapping depends on whether a dense or sparse view of the matrix is desired.

9.2.1 Dense Storage

There are two ways to map a dense matrix to memory: row-major ordering and column-major ordering. The two storage orders differ by the order inwhich the individual elements are stored in memory (Figure ...). For an M by N matrix stored in a linear array a in row-major format, the address offset of an element a_{ij} compared to element a_{00} is given by

$$A(i,j) \rightarrow \text{a[i * N + j]} \tag{9.11}$$

and in column-major format by

$$A(i,j) \rightarrow \text{a[i + j * M]} \tag{9.12}$$

Neither storage order has any advantages or disadvantes over the other one. It is only a matter of convention, and depend on the library and programming language. Fortran arrays are column-major, while C/C++ arrays are row-major.

However, algorithms optimized for column-major access might not give the same performance for row-major formats and vice-versa. Consider a simple loop over matrix elements in the Fortran programming language

Listing 9.2.1: Matrix Loop

```
1 DO i = 1, N
2  DO j = 1, N
3   A(i,j) = A(i,j) + 2
4  ENDDO
5 ENDDO
```

By first looping over the rows, and then over the columns, the elements in the column-major matrix are accessed in a *non-contiguous* manner. This prevents efficient optimizations like vectorization due to cache misses. By swapping the loops, the inner loop can be vectorized. Therefore, if algorithms assume a certain ordering, it is important to pass matrices of the same ordering.

Column-major ordering is by far the most common in scientifc computing. Matrix algebra routines like the QR, Eigenvalue or Cholesky decompositions are most intuitively written using column operations rather than row operations. Examples include the LAPACK and Eigen matrix libraries.

9.2.2 Sparse Storage

A matrix is called *sparse* if "most" of its elements are zero. The threshold between sparse and dense is ill-defined, and range anywhere between 1% and 10%. Sparse matrices

are encountered in many fields of computational science in systems with few pair-wise interactions. By storing and looping over the siginifcant elements only, the memory footprint and the scaling of matrix multilications or decompositions can be significantly reduced. There are several different sparse matrix formats available. The following sections will demonstrate how the matrix

$$\begin{pmatrix} 3 & 5 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 6 & 7 \end{pmatrix}$$

may be represented using some of these formats.

Coordinate Format

The coordinate format (COO) is one of the simplest storage schemes for sparse matrices. The matrix is stored using three arrays of lenghth NNZ (number of non-zero elements) containing the row indices, column indices and matrix entries. The above matrix reduces to

$$COO : \begin{cases} \text{idx\_i} &= [0,0,1,2,3,2,3] \\ \text{idx\_j} &= [0,1,1,2,2,3,3] \\ \text{val} &= [3,5,4,1,6,2,7] \end{cases}$$

The values do not need to be necessarily ordered. The COO format has the advantage that it is very easy to insert elements and change the sparsity pattern with low overhead. COO is generally not used for algebra due to slow random access of elements if elments are not in order. Rather, COO is an intermediate format for incremental construction of sparse matrices in the CSR or CSC format.

Compressed Sparse Row

The compressed sparse row format (CSR) is similar to COO, but compresses the rows. A CSR matrix is represented by three arrays: the extent of rows, the coulmn indices and non-zero matrix entries, with dimension M, NNZ and NNZ respectively. For the example matrix above, the CSR format gives

$$CSR : \begin{cases} \text{row\_ext} &= [0,2,3,5] \\ \text{col\_idx} &= [0,1,1,2,2,3,3] \\ \text{val} &= [3,5,4,1,2,6,7] \end{cases}$$

The ith entry of the row extents contains the offset to the first non-zero element in the array val in the ith row. The entries in col_idx are the associated column indices. Compared to COO, accessing elements is much faster, and the memory prefactor is reduced from $3NNZ$ to $2NNZE + M$. Matrix-vector products are very fast to compute, but CSR in not well suited for column slicing.

Compressed Sparse Column

The compressed sparse column format is the "column-major" analog to the CSR format. Instead of compressing the rows, it compresses the columns. The matrix in the CSC format is represented as

$$CSC : \begin{cases} \text{col\_ext} &= [0,1,3,5] \\ \text{row\_idx} &= [0,0,1,2,2,3,3] \\ \text{val} &= [3,5,4,1,6,2,7] \end{cases}$$

CSC has similar performance to CSR, but is better suited for column slicing and therefore column-oriented matrix decomposition.

9.2.3 Block Compressed Sparse Row

The block-compressed sparse row (BCSR or BSR) format is a generalization of CSR. BCSR is ideal for matrices which are *block-sparse*, i.e. matrices with few dense submatrices. For a constant block size n by m, let the number of row and column blocks be M_{blk} and N_{blk}, and NNZ_{blk} the number of significant blocks. A matrix is then represented by three arrays containing the row block offsets, column block indices, and non-zero indices with size M_{blk}, NNZ_{blk} and $NNZ_{blk}mn$. For blocks of size 2 by 2, the matrix above in the BCSR format yields

$$BCSR : \begin{cases} \text{rowblk\_ext} &= [0,1] \\ \text{colblk\_idx} &= [0,1] \\ \text{val} &= [3,0,5,4,1,6,2,7] \end{cases}$$

The matrix is therefore split into two non-zero blocks of size 2-by-2 each, with elements $\{3,0,5,4\}$ and $\{1,6,2,7\}$ using column-major ordering. The BCSR format can further reduce the storage prefactor for block-sparse matrices. Smaller block-sizes can account for a higher degree of sparsity, but lead to a higher prefactor. Large block sizes have lower prefactor, but capture less of the sparsity. Figure 9.3 shows the BCSR format for different split factors. The block sizes do not need to be the same, although a constant block size is best suited for load balancing. Whether a block is significant or not depends on the matrix norm that is chosen as a criterion for filtering. Popular choices include the Frobenius norm or the max norm.

9.2.4 Distributed Storage

In distributed memory parallelism, processes do no longer have access to the whole matrix. The question then arises how to distribute data over multiple nodes or processes. Distribution can be *blocked* or *cyclic* (Figure 9.1). In the block representation, the matrix is distributed by dividing rows, columns (or both) by the number of processes, and distributing them sequentially over the processes. On the other hand, the cyclic representation distributes each individual row, column or matrix element over the set of processes and wrapping over them until all elements are assigned to a processor. The cyclic data layout achieves great load balancing by evenly distributing the matrix elements, but is not suitable for row or column manipulations due to a high communication overhead. A blocked representation can reduce this overhead, but load-balancing is more difficult to achieve if the number of elements is not evenly divisible by the number of processes.

The most commonly used format for distributed matrix storage is the *block-cyclic* representation (Figure 9.3). It combines the best of both worlds, with good load-balancing

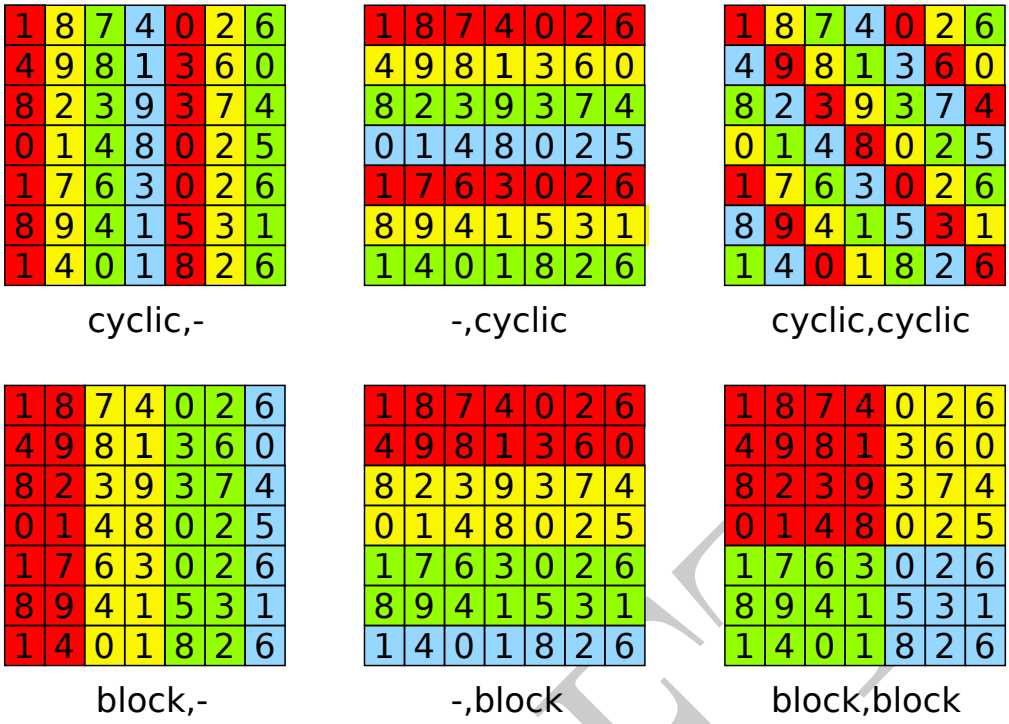


Figure 9.1: There are two distinct ways to distribute rows, column or elements along processes, namely cyclic and block. Each color represents a processor (4 in total).

and low communication overhead for matrix algebra. Communication is most efficient by arranging processes into a 2-dimensional rectangular grid with $N_{proc} = N_{row} \times N_{col}$, such that each process is uniquely identifies by its coordinates (i, j). A 4-by-2 grid therefore has 4 *process rows* and 2 *process columns*. With MPI, this is most easily achieved by using *Cartesian grids*. In Cartesian grids, processes can only communicate with their nearest neighbor.

Each process has a local array which can be packed in different ways. The ScaLAPACK library keeps the original ordering of the matrix elements, while other matrix libraries like DBCSR store the individual blocks sequentially in memory. Blocks do not necessarily need to be equal in size. Matrices like DBCSR also support heterogeneous block sizes.

All the sparse storage schemes discussed in the previous sections can also be applied to distributed matrices. DBCSR uses a block-sparse row format which distributes the significant blocks over the processors in a row-major format (Figure **??**).

9.3 Tensors

9.3.1 Definitions

Tensors are multi-dimensional arrays and can be seen as a generalization of matrix objects to higher dimensions, containing a set of complex or real values $\{a_{ijk...}\}$ with the ordered indices $i = 0, 1, 2, ...N_0$, $j = 0, 1, 2, ...N_1$, $k = 0, 1, 2, ...N_2$ etc. The total number of indices required to identify an element in a tensor corresponds to its *dimension*, also called *rank*, *order* or *degree*. It should be stressed that tensor rank is different from the concept of a

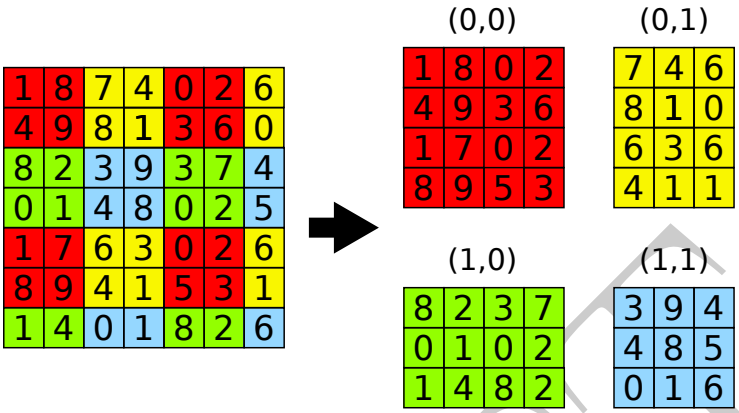


Figure 9.2: In the block-cyclic distribution, the matrix is divided into blocks which are distributed over a processor grid, in this case 2×2.

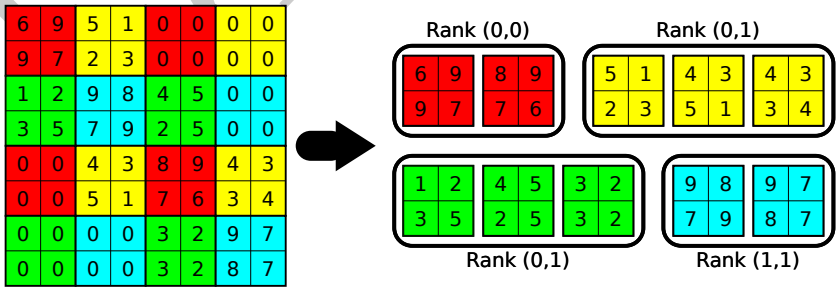


Figure 9.3: The DBCSR library distributes the significant blocks over the processor grid in row-major format.

matrix rank.

Tensors are important quantities in various scientific fields, especially in physics and quantum chemistry. Examples of tensor are the 4-dimensional MP2-amplitudes t_{iajb} or the 3-dimensional 3c2e electron integrals. Matrices can be seen as 2-dimensional tensors.

The analog of matrix multiplication is *tensor contraction*, nd involves the summation over one or multiple indices. Consider for example the tensor contraction

$$C_{ijm} = \sum_{kl} A_{ikjl} B_{lkm} \tag{9.13}$$

Here, the summation runs over the indices kl. Alternatively, the expression above can be abbreviated as

$$C_{ijm} = A_{ikjl} B_{lkm} \tag{9.14}$$

which is known as *Einstein summation*, a notational convention used to simplify tensor expressions. Indices appearing on the right but not on the left are implicitly summed over.

9.3.2 Tensor Storage and Mapping

Mapping multi-dimensional to the linear main memory is non-trivial. Tensors may follow a generalized column or row major storage such that

$$A(i_0, i_1, ..., i_n) \rightarrow \sum_{k=0} \left(\prod_{l=k+1}^{N_{dim}} n_l \right) i_k \qquad \text{row-major} \tag{9.15}$$

$$A(i_0, i_1, ..., i_n) \rightarrow \sum_{k=0} \left(\prod_{l=0}^{k-1} n_l \right) i_k \qquad \text{column-major} \tag{9.16}$$

where N_{dim} is the dimension of the tensor, and n_k is the size of dimension k.

Consider again the tensor contraction 9.14. loops running over the indices kl:

Listing 9.3.1: Tensor Loop

```
DO i = 1, ndim_i
  DO j = 1, ndim_j
    DO m = 1, ndim_m
      DO k = 1, ndim_k
        DO l = 1, ndim_l
          C(i,j,m) = A(i,k,j,l) * B(l,k,m)
        ENDDO
      ENDDO
    ENDDO
  ENDDO
ENDDO
```

Similarly to Example 9.2.1 which loops over matrix elements, efficient parallelization by means of vectorization is a major concern when coding tensor contractions by hand.

Ideally, tensor contractions should be offloaded to a specialized library. However, writing a general tensor library is a complex task, as the optimal kernel is dependent on the nature of the tensor contraction, i.e. number of indices involved, tensor dimensions and memory mapping. Whereas the dgemm BLAS routine only needs to know whether the matrices are transposed or not, the rapidly increasing number of parameters and loops makes it very difficult to write optimized code for arbitrary tensor contractions. For the example above, there are already $5! = 120$ different ways to arrange the loops.

Many tensor libraries solve this problem by mapping tensors to matrices so that existing code for optimized matrix-matrix multiplication can be used instead. Consider the tensor A_{ikjl}. By introducing the *super-indices* P and Q, the tensor can be mapped to a matrix with $N_P = n_i n_k$ rows and $N_Q = n_j n_l$ columns with

$$P(i,k) = i + k * n_i \tag{9.17}$$

$$Q(j,l) = j + l * n_j \tag{9.18}$$

The individual tensor elements are then accessed by

$$A(i,k,j,l) \to P(i,k) + Q(j,l)N_P \tag{9.19}$$

using column-major storage for both the indices and super-indices. This corresponds to mapping ik to the rows and jl to the columns of a large matrix, which may also be written in a condensed notation as $(01 \mid 23)$, where $0, 1, ...$ indicate the mapping of the first, second, ... index of the tensor. As another example, consider the mapping $(0 \mid 213)$: this indicates that the index i is mapped to rows, and jkl are mapped to the columns. Alternatively, the mapping can be indicated by using upper or lower indices. $(01 \mid 23)$ corresponds to A^{jl}_{ik} and $(0 \mid 213)$ to A^{jkl}_{i}. The lower indices are also known as *covariant* indices, and the upper indices as *contravariant* indices. From here on out, only the notation $(\cdot \mid \cdot)$ will be used.

Coming back to the tensor contraction in 9.14, mapping the tensors according to $A^{(02|13)}_{ikjl}$, $B^{(10|2)}_{lkm}$ and $C^{(01|2)}_{ijm}$, the contraction can be reformulated as a simple matrix multiplication

$$\mathbf{C} = \mathbf{AB} \tag{9.20}$$

and optimized matrix libraries can be used to perform the tensor contraction. It is also possible to choose the mapping $B^{(2|10)}_{lkm}$, and the tensor contraction may be expressed as

$$\mathbf{C} = \mathbf{AB}^T \tag{9.21}$$

by taking the transpose of $\mathbf{B}$. In general, any tensor contraction can be reformulated as a matrix multiplication by mapping the indices which are summed over to either the row or column of a matrix, and the indices which are not involved to the remaining dimension.

There are two drawbacks to this approach: first, tensors which are participating in different types of tensor contractions may need to be reordered. Reordering can take up a significant amount of time and add overhead to the tensor contraction, and it effectively doubles the amount of space needed by the tensor. Second, standard matrix libraries may not be optimized to handle tall-and-skinny (TAS) matrices. TAS matrices are matrices where one dimension is much larger then the other one, which is often the case for tensors with uneven mappings. Consider the tensor $B^{(10|2)}_{lkm}$. It maps to a matrix

with $n_k n_l$ rows and n_m columns. If this matrix is distributed over an MPI grid, the process columns have much more work to perform than the process rows, which may lead to load imbalances. This problem can be addressed by dividing the TAS matrices into multiple square submatrices to improve load balancing, but at the cost of adding further complexity to the code.

9.4 Matrix Libraries

This section briefly describes the different matrix libraries that are used in MEGALOchem.

9.4.1 BLAS

Basic Linear Algebra Subroutines (BLAS) is a *specification* that describes a set of fundamental operations on vectors and matrices which serve as building blocks for higher level algebraic functionality **??**. The subroutines are grouped into different *levels*. BLAS 1 provides vector-vector operations, BLAS 2 performs matrix-vector operations, and BLAS 3 provides matrix-matrix operations. Implementations of the BLAS library are often optimized for a certain type of processor, such as Intel's math kernel library (MKL) or the BLAS-like instantiation software (BLIS) which are optimized for Intel and AMD processors respectively. By providing a general interface, BLAS-based code is highly portable and should be preferred over processor-dependent, intrinsic functions.

9.4.2 LAPACK

The linear algebra package (LAPACK) provides routines for matrix factorizations such as the QR decomposition, eigenvalue decomposition or singular value decomposition [210]. It also includes functions for solving systems of linear equations and linear least squares problems. The official Netlib LAPACK implementation is based on BLAS and therefore highly portable, and simply linking the corresponding BLAS library that is optimized for the targeted architecture is sufficient. However, processor-specific re-implementations of LAPACK such as Intel MKL also exist.

LAPACK can handle dense matrices only.

9.4.3 Eigen

Eigen is a C++ template library that provides basic and advanced matrix algebra routines, including numerical solvers and matrix decompositions [211]. Compared to LAPACK, Eigen offers a more user-friendly interface with an API that is closer in form to to standard algebraic notation. A matrix multiplication with addtion is as simple as writing

Listing 9.4.1: Eigen Matrix Multiplication

```
1 Eigen::MatrixXd A, B, C;
2 // initialize matrices
3 A = A + B * C;
```

Matrix manipulations, such as row, column or block operations are also much more intuitive, which is its primary use in MEGALOchem.

9.4.4 PBLAS

PBLAS (Parallel BLAS) is a generalization of the BLAS specification to distributed memory environments [212]. It uses BLACS (basic linear algebra subprograms) as the message passing interface, typically built on MPI, and optimized for 2D grid communication. Matrices and vectors use a block-cyclic distribution with homogeneous block sizes across processor rows and columns. Similarly to BLAS, it provides a general interface and several different implementations exist (Netlib, Intel MKL, IBM, Cray). It closely mimics the function signature of BLAS, but matrices have an additional *array descriptor* which holds all parameters of the matrix distribution.

Matrix multiplication is performed using Cannon's algorithm [213].

9.4.5 ScaLAPACK

Scalable LAPACK (ScaLAPACK) is LAPACK's analog for distributed memory systems [214]. It provides highly optimized, advanced linear algebra routines based on the PBLAS interface. It solves linear least squares, eigenvalue and singular value decomposition problems, among others. Again, while the specification is general, processor optimized implementations are often encountered, which are even more crucial in the context of high-performance computing.

Compared to LAPACK, ScaLAPACK does not support many matrix decompositions with pivoting. Pivoting incurs a very high overhead due to the increased communication between nodes.

9.4.6 DBCSR

DBCSR (distributed block compressed sparse row) is a Fortran library designed for efficient sparse matrix multiplication in distributed memory environments **??**. It is OpenMP and MPI parallel and can explot Nvidia and AMD GPUs via Cuda and HIP. It uses a sparse block-cyclic matrix storage format (Figure 9.3) and was originally developed for linear-scaling self-consistent field calculations within CP2K, a quantum chemistry and solid state physics software package [215, 216].

The DBCSR matrix multiplication architecture is divided into several layers managing data transfer, data access and offloading. Figure 9.4 shows a schematic representation of the layers. The top-most layer manages the parallelization of the matrix multiplication over multiple nodes, and subdivides the matrix into large sparse sub-matrices, or *panels*, which contain approximately the same number of blocks. A block-generalized version of Cannon's algorithm **??** is used to communicate the panels to different processes in a regular and ordered fashion. At each "tick" of Canon's algorithm, each process sends and receives panels which it then multiplies and adds to the local blocks.

The multiplication of the panels is handled by the multrec layer, which subdivides panels *recursively* along the longest dimension until the submatrices are small enough to fit into cache size.

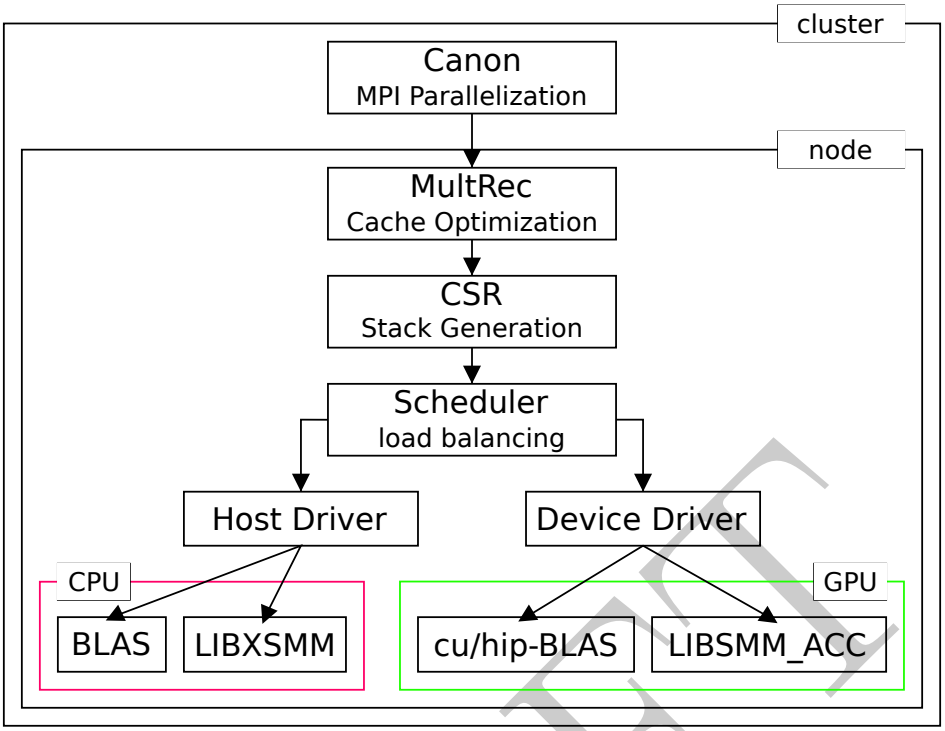


Figure 9.4: Code architecture of the DBCSR library.

The CSR layer then determines which blocks need to be multiplied from the sparsity pattern of the matrices. It generates a list of needed block-multiplications, called *stacks* and passes this information to the scheduler. The scheduler receives the stacks and arranges them for processing by handing them off to different drivers. Stacks can be processed either by the host (CPU) or the device (GPU). Both drivers support standard matrix multiplication by using (cu/hip)-BLAS, as well as small matrix multiplication (SMM) which are handled by specialized SMM libraries optimized for CPU (libxsmm) or GPU (libsmm_acc).

For a multiplication of matrices with dimension N by K and K by M, a problem size suitable for SMM libraries approximately falls within $(MNK)^{1/3} < 64$. Block-sparse matrices in quantum chemistry often have very small block sizes. However, standard libraries like BLAS are most often optimized for very large matrices, and SMM libraries are often crucial to reduce the overhead when using small block sizes.

In 2019, the DBCSR library was extended to handle tensor contractions using the TAS matrix mapping discussed above [217]. It is an *ad-hoc* extension of the DBCSR matrix machinery, and still somewhat experimental. Memory issues are commonly encountered (see chapter 7).

A C-interface to the DBCSR library was written as part of this PhD project [218].

OTHER POSSIBILTY: https://arxiv.org/abs/1607.00291

Chapter 10

The MEGALOchem Program Package

MEGALOchem is a quantum chemistry program that specializes in and provides tools for low scaling electronic structure methods for ground and excited states in the atomic and local molecular orbital basis. It is MPI parallel and can use GPUs for accelerating sparse matrix multiplication and tensor contractions via the DBCSR library. It is open-source and available on GitHub (`github.com/ambmax00/megalochem`). This chapter will give an overview on the software architecture and features of MEGALOchem. As the program is still in development at the moment of writing, details are subject to change, and the reader is referred to the online documentation for the up-to-date user and developer guides.

10.1 Motivation

Nowadays, quantum chemists have access to a healthy ecosystem of open-source quantum chemistry software that are under active development, including, but not limited to, CP2K [], Dalton [], GAMESS [], NWChem [], Psi4 [], pyscf [] or VeloxChem []. They offer efficient algorithms for computing correlated ground and excited state properties, either with thread- or task-based parallelism. However, none of them support local correlation methods based on local molecular orbitals or atomic orbitals. The goal of MEGALOchem is two-fold: (1) provide a framework for the development of local correlation methods and (2) offer a set of kernels for accelerating Hartree-Fock, MP2 and ADC(2) calculations, using MPI parallelism. While MEGALOchem is stand-alone, it should be primarily seen as an external library or module. Quantum chemistry software packages are often composed of many different, often quite sophisticated libraries that handle e.g. evaluation of electron integrals, computation of the coulomb and exchange matrix or solving the eigenvalue problem of excited state methods. Modules allow teams of developers to concentrate on the optimization of specific libraries, rather than a whole program suite.

10.2 Software Architecture

MEGALOchem is written in C++17 and Fortran. Figure 10.1 shows a rough outline of its structure and dependencies. Its work flow is identical to other quantum chemistry software. An input file containing job information is parsed by the program and passed to a driver or scheduler which sets up a series of calculations, most often in the order Hartree-Fock → Correlated Methods → Properties. The results are collected and saved to an output file.

MEGALOchem takes two inputs: a `.json` file which contains the requested jobs, and an optional `.hdf5` file which contains information of the wave function from previous calculations. HDF5 (hierarchical data format) is a high-performance library and file format for storing and managing data. Data sets are organized into a filesystem-like format, called *groups*, and can be accessed with a similar syntax `/path/to/dataset`. The HDF5 library is written in C, but has bindings for C++, Fortran, Python and Java, among others. HDF5 files can also be opened and edited using external programs like HDFview or Vitables. The support for multiple languages and the simple user interface allows to easily extract information from the output file without the need to write complex bash scripts, especially for matrices like the HF coefficients, or the ADC transition matrices.

The JSON format was chosen as the data format for the job control input, because it is a well known file format that is easy to read and edit, with multiple open-source parser libraries already available. MEGALOchem has an object-oriented approach for setting up jobs, and examples of input files will be shown in section 10.4.

After parsing the JSON input file and optionally loading data from previous calculations, the main driver sets up the jobs in a queue. The queue can contain multiple jobs of the same type, e.g. Hartree-Fock, with different systems and options. After initializing the parallel runtime environment, the queue is worked off and the job information is passed to the different subdrivers. MEGALOchem currently has four different job types it can handle: Hartree Fock and MP2 ground state calculations, excited state calculations (CIS and ADC(2)), as well as orbital localization, analysis and plotting. Subdrivers like MP2 or ADC can read information from previous calculations, meaning that the HF wave function does not need to be recomputed each time. Output is written to `stdout` and a new HDF5 file.

The subdrivers depend on other modules for handling evaluation of electron integrals, linear algebra and sparse matrix multiplication.

Integrals are computed using the `libcint` library [**?**], which is also used in PySCF. Alternatively another branch of libcint called `qcint` may be used, that is optimized against AVX, AVX2 and AVX512. It provides the same API as libcint and no modifications to MEGALOchem are necessary. However, other integral libraries can be easily swapped in. MEGALOchem has its own data structures for molecules, basis sets and a general interface for requesting and handling electron integrals. Only two files need to be modified to accommodate other libraries.

Linear algebra, such as eigenvalue decomposition, singular value decomposition and other dense matrix operations are mainly handled by the ScaLAPACK library. ScaLAPACK is written in Fortran, but MEGALOchem provides C and C++ bindings for it. For node-local matrix operations, the Eigen library is used as well.

The most crucial part of MEGALOchem are the *kernels*, which construct the coulomb,

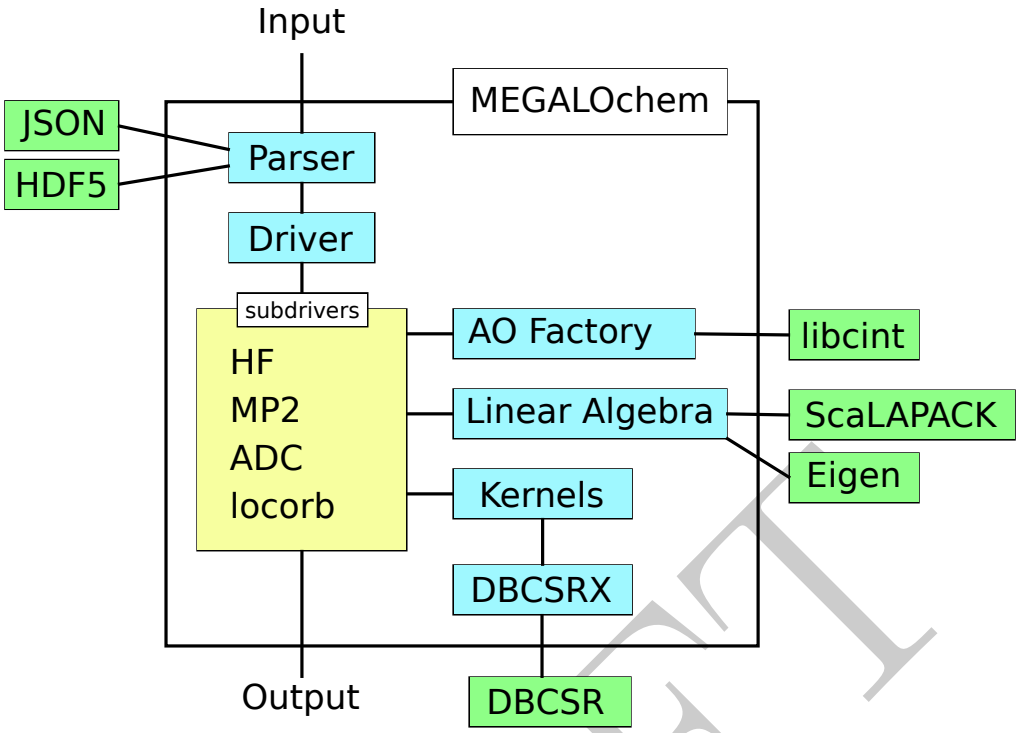


Figure 10.1: Software architecture of the `MEGALOchem` program package with external dependencies.

exchange and Z matrices (encountered in AO-SOS-MP2), and compute the CIS and ADC(2) matrix vector products. Sparse matrix multiplication in the kernels is managed by the DBCSR library. `MEGALOchem` provides an extended C++ interface (DBCSRX) with additional capabilities for ease of use.

10.3 Parallel Runtime Environment

As mentioned in the introduction, `MEGALOchem` is MPI parallel. Calculations are launched using the command `mpirun`, `mpiexec` or similar. For example, the command

```
mpirun -n 64 bin/chem h2o /scratch
```

launches 64 MPI processes with the input file `h2o.json` located in the directory `/scratch`. Process are arranged in a Cartesian grid. It is therefore crucial to choose a square number of processes for optimal performance, even if some cores on the nodes might be idle. For example, for three nodes with 24 cores each, it is better to run 64 processes instead of 72, as the performance penalty for idling cores is not significant.

Most of the considerations for performance are dictated by the DBCSR library. DBCSR also supports OpenMP parallelism, meaning a hybrid MPI+OpenMP approach is also possible for `MEGALOchem`. However, the tensor module of DBCSR has optimal performance for 1 OpenMP thread per MPI process. The reasons are two-fold: first, each process can only allocate approximately 2.1 GB at once, which means that a larger number of MPI processes per node reduces the possibility of crashes for memory-intensive calculations. Second, not every task in the tensor library is parallelized with OpenMP

threads. For these reasons, MEGALOchem was also not optimized for a lower number of MPI processes, and the 1-OpenMP-thread rule is also valid for the rest of the program modules. This preference for pure MPI might conflict with other quantum chemistry software with a hybrid MPI+OpenMP approach. For more information on MPI binding and mapping, the reader is referred to the CP2K documentation, which also uses DBCSR (https://xconfigure.readthedocs.io/en/latest/cp2k/#running-cp2k).

Besides the DBCSR runtime environment, ScaLAPACK also works best with a square process grid. MEGALOchem uses two separate grid instances for DBCSR and ScaLAPACK to avoid conflicts during message passing.

10.4 JSON interface

Job control is specified by an input file in the JSON format. It is inspired by the BAGLE qunatum chemistry software [**?**] and follows a similar scheme to the object-oriented python interfaces like Psi4 or PySCF. Consider the following example for running a Hartree-Fock calculation on a water molecule with the cc-pVDZ basis set

```
{
  "megalochem": [
  {
    "type": "atoms",
    "tag": "xyz",
    "unit": "angstrom",
    "geometry": [
      0.00000,         0.00000,         0.11779,
      0.00000,         0.75545,        -0.47116,
      0.00000,        -0.75545,        -0.47116
    ],
    "symbols": ["O", "H", "H"]
  },
  {
    "type": "basis",
    "atoms": "xyz",
    "tag": "basis1",
    "name": "cc-pvdz"
  },
  {
    "type": "molecule",
    "tag": "mol",
    "atoms": "xyz",
    "basis": "basis1",
    "mult": 1,
    "charge": 0
  },
  {
    "type": "hfwfn",
    "tag": "hfwfn",
    "molecule": "mol",
```

```
    "guess": "SAD",
    "build_J": "exact",
    "build_K": "exact"
   }]
}
```

The JSON file contains a main structure `megalochem` which specifies an ordered array of objects. Each object must contain the fields `type` and `tag`. The entry `type` specifies what kind of object it is, and `tag` is a user-defined name to uniquely define that object. This is similar to declaring a variable in a programming language. The available types are

- `global`: Specifies global variables like block thresholds or damping factors for electron integrals
- `atoms`: Specifies the atomic structure of the system. It can either conatin an array with coordinates or reference an XYZ file
- `basis`: Specifies the basis set and what kind of clustering is used. Clustering indicates how atomic orbitals are grouped into blocks. A cutoff parameter can also be specified for removing linear dependencies
- `molecule`: Specifies molecular information like multiplicity and charge
- `hfwfn`: Job control for computing the HF wave function
- `mpwfn`: Job control for computing the MP2 wave function
- `adcwfn`: Job control for computing CIS and ADC(2) excitation energies
- `moprint`: Job control for localizing and plotting orbitals (CMOs, LMOs, NTOs) from different calculations

Some objects may reference other preceding objects. For example, the `adcwfn` object needs to reference a `hfwfn` object which is then used as the reference.

This object-oriented approach will also facilitate a potential implementation of a python interface to `MEGALOchem`.

10.5 Design Patterns

One of the major challenges in designing a modular quantum chemistry software is deciding how arguments are passed from one module to another. Implementations of electronic structure methods often accept dozens of different keywords, leading to complex function and constructor signatures, and grouping keywords into logical structures is non-trivial. A popular solution to this problem is the use of a `options` class which group parameters into an overarching *context*, such as `psi4::options` in Psi4 or `libctx::context` in Q-Chem. The advantage is that keywords are easily accessed by requesting it by name. Existence of the keys can also be checked by using member functions. However, this is at the same time the greatest downside of context classes. For a complex hierarchy of interacting modules and functions, which can all insert or delete keywords, it is often

impossible to know what objects are or are not present within the context at a given level.

Contexts are therefore not used in `MEGALOchem`. This leads to more verbose code, as each class needs to explicitly define all its parameters, but it is less error-prone. Optional parameters are handled using a modified `std::optional` class.

Large classes are constructed using the *factory method pattern* and parameters are passed to the factory class by calling a chain of *setter* member functions. Consider the following example for defining a Hartree-Fock object:

```
auto hfobj = hf::hfmod::create()
 .set_world(my_world)
 .set_molecule(my_molecule)
 .df_basis(dfbas)
 .build_J("dfao") // optional
 .build_K("dfao") // optional
 .guess("SAD")    // optional
 .build();
```

It sets up a density-fitting calculation of a Hartree-Fock wave function. Objects like `world` and `molecule` are required, and will give a runtime error if they are not set before calling `build()`. Other parameters are optional, and can either be set or just omitted. The factory function pattern needs a lot of boiler-plate code, but imposing the pattern is facilitated by the use of preprocessor directives. Each module header file has a preprocessor list of required and optional parameters before the class definition. For the `hfmod` class, it reads

```
#define HFMOD_LIST \
  (((world), set_world),
   ((desc::shared_molecule), set_molecule), \
   // ... other parameters
  )
#define HFMOD_LIST_OPT \
  (((util::optional<std::string>), guess, "SAD"), \
   ((util::optional<double>), scf_threshold, 1e-6), \
   ((util::optional<int>), max_iter, 100), \
   ((util::optional<bool>), do_diis, true), \
   ((util::optional<int>), diis_max_vecs, 10), \
   ((util::optional<int>), diis_min_vecs, 2), \
   ((util::optional<int>), diis_start, 1), \
   ((util::optional<std::string>), build_J, "exact"), \
   ((util::optional<std::string>), build_K, "exact"), \
   ((util::optional<std::string>), df_metric, "coulomb"), \
   // ... other parameters
   )
```

New keywords can be easily added to the list. For optional parameters, there is also the possibility to add default values. The lists are then passed to preprocessor functions which automatically generate the necessary code for the factory functions. They construct a `struct` containing all of the parameters in the list which can be used in the constructor of the module class. While this approach is more verbose, it provides a clearer overview

of possible input parameters.

10.6 Libraries sec:LIBS

This section provides an overview of the different modules present in `MEGALOchem`, and presents some examples on how to use them.

All information on the parallel MPI runtime environment, that is, communicators and Cartesian grids for the DBCSR and ScaLAPACK libraries, are stored in the `megalochem::world` object. It contains information on the rank, size and topology of the different grids.

10.6.1 dbcsrx

The `dbcsrx` module is an extended C++ interface to the DBCSR library. Most objects are constructed using the factory function pattern explained in the previous section. Classes may have different constructors with a different set of keywords. Matrices are easily constructed from scratch by using the `create` function

```
std::vector<int> blksizes = {2,5,6,3};
dbcsr::shared_matrix<double>
my_matrix = dbcsr::matrix<>::create()
        .set_cart(world.dbcsr_grid())
        .name("My Matrix")
        .row_blk_sizes(blksizes)
        .col_blk_sizes(blksizes)
        .matrix_type(dbcsr::type::symmetric)
        .build();
```

which returns a `std::shared_ptr` to a dbcsr matrix object, also known as `dbcsr::shared_matrix`. In general, all the factory functions return pointers rather than objects. The matrix class needs the Cartesian grids, row and column block sizes, as well as the matrix type (symmetric, hermitian, nonsymmetric, ...). Factory functions can also be used to construct copies of the matrix, e.g.

```
auto my_copy = dbcsr::matrix<>::copy(*my_matrix)
        .name("My Copy")
        .build();
```

There are further factory functions for constructing templates, transposes or reading data from disk.

Tensors are constructed in a similar way, but additionally need a `pgrid<N>` (process grid) object which is a N-dimensional general Cartesian grid. The underlying matrix representation also needs mapping information (see previous chapter). A tensor can then be allocated by

```
std::array<std::vector<int>,3>
arrvec = {blksizes1, blksizes2, blksizes3};

dbcsr::shared_pgrid<3>
my_pgrid = dbcsr::pgrid<3>::create(world.comm()).build();
```

```
dbcsr::shared_tensor<3>
my_tensor = dbcsr::tensor<3>::create()
        .set_pgrid(my_pgrid)
        .name("My Tensor")
        .blk_sizes(arrvec)
        .map1({0}).map2({1,2})
        .build();
```

Each class has member functions which have a Fortran analog in the DBCSR library. The object-oriented C++ interface greatly reduces the complexity of the code, as the C interface of the DBCSR library can become quite unwieldy at times.

The matrix multiplication and tensor contraction functions are also used in a factory function-like manner and their function signatures resemble those of the LAPACK `gemm` routines

```
dbcsr::shared_matrix<> A, B, C;
// allocate and fill matrices ...

// perform C = 3 * A * B + C
dbcsr::multiply('N', 'N', 3.0, *A, *B, 1.0, *C)
        .filter_eps(1e-8) // pass optional parameters
        .perform();
```

The tensor contraction routines have been extended to work with Einstein summation:

```
dbcsr::shared_tensor<3> A, C;
dbcsr::shared_tensor<4> B;
// allocate and fill ...

// perform C("ikl") = 3 * A("ljm") * B("ikjm")
dbcsr::contract(3.0, *A, *B, 0.0, *C)
        .perform("ljm, ikjm -> ikl");
```

which greatly simplifies the implementation.

One of the more prominent extensions in dbcsrx is the `btensor` (batched tensor) class. Tensor contractions often need to be performed in batches for memory efficiency. This however means that the tensor needs to be spit appropriately along each dimension. The `btensor` manages the necessary metadata of batched tensor contractions and provides helper functions to the developer. Moreover, it is possible to pass a `btype` variable to the constructor which specifies whether the tensor is held in core memory, read from disk, or generated on-the-fly. The interface remains the same, irregardless of how the tensor is stored, meaning that in-core, disk and direct algorithms do not need separate implementations. Disk batched tensors store data in block-wise format on disk using MPI I/O, meaning each process can read and write independently. Direct batch tensors furthermore need a generator function that indicates how the data is generated, for example in the case of atomic orbital electron integrals.

`dbcsrx` also provides several functions for converting to ScaLAPACK and Eigen matrix formats.

10.6.2 `math`

The `math` library contains the C++ interface to the ScaLAPACK functions for matrix decompositions, as well as DIIS and Davdison solver routines, a routine for incomplete pivoted Cholesky decompostion, and an interface to the Fortran library by Helmich-Paris and Visscher [**?**] for computing the Laplace quadrature parameters.

The ScaLAPACK interface accepts DBCSR matrices which are then converted to the fixed block-cyclic format used by ScaLAPACK. Using the interface is easy. The following example shows the eigenvalue decomposition of a hermitian matrix:

```
hermitian_eigen_solver hsolver(world, *my_matrix, 'V', true);
hsolver.compute();
auto eigenvalues = hsolver.eigvals();
auto eigenvectors = hsolver.eigvecs();
```

10.6.3 `ints`

Evaluating of electron integrals, as well as the computation of density fitting coefficients is handled by the `ints` module. The interface for requesting integrals is general and does not depend on what integral library is used.

Integrals are computed by constructing an `aofactory` object

```
auto factory = ints::factory(my_world, my_molecule);

// get the overlap integrals
dbcsr::shared_matrix<> s_bb = factory.ao_overlap();
dbcsr::shared_matrix<> k_bb = factory.ao_kinetic();
```

Computing the 3c2e and 4c2e integrals is more complex, as they need to be evaluated in batches and more information needs to be passed to the functions:

```
dbcsr::shared_tensor<3> eri_3c2e;
// allocate tensor

factory.ao_3c2e_setup(metric::coulomb);
factory.ao_3c2e_fill(eri_3c2e);
```

For this reason, the user should prefer the auxiliary `aoloder` class, where integrals are first requested, and then computed.

```
auto loader = ints::aoloader::create()
        .set_world(my_world)
        .set_molecule(my_molecule)
        .build();

loader.request(ints::key::s_bb)
        .request(ints::key::coul_xbb)
        .request(ints::key::dfit_coul_xbb);

loader.compute();
```

Integrals are requested by a predefined set of keys. Fitting coefficients can also be computed by the aoloader. Furthermore, it also resolves any dependencies and computes all the necessary integrals, even if they were not specifically requested.

The integral library can compute the following density fitting coefficients:

- Standard coulomb metric
- Coulomb-attenuated metric
- Overlap metric
- Quasi-robust density fitting
- Pair-atomic resolution of the identity

10.6.4 fock

The `fock` library provides kernels for computing the coulomb and exchange matrix. The kernels follow the factory pattern, which returns a general `fock::J` or `fock::K` which works the same irrespective of which method is chosen to evaluate the coulomb and exchange matrices.

```
std::shared_ptr<fock::J>
jbuilder = fock::DF_J::create()
        .set_world(my_world)
        .molecule(my_molecule)
        .eri3c2e_batches(my_tensor)
        .metric_inv(my_matrix)
        .build();

jbuilder->init();

// compute density matrix
dbcsr::shared_matrix P_bb = ...
jbuilder->set_density_alpha(*P_bb);

// compute coulomb matrix
auto J_bb = jbuilder->get_J();
```

After calling `build()`, the object needs to be initialized. For each SCF cycle, the kernel is updated with the current density matrix. The matrix is then constructed by calling the `get` functions.

The coulomb matrix can be computed with the kernels

- `EXACT`: exact coulomb matrix without density fitting
- `DFAO`: coulomb matrix with density fitting in the AO basis

and the following kernels are available for the exchange matrix

- `EXACT`: exact exchange matrix wihout density fitting

- `DFAO`: density fitting in the AO basis
- `DFMO`: density fitting in the MO basis
- `DFAOMEM`: density fitting in the AO basis, with the fitting coefficients computed on-the-fly
- `DFROBUST`: density fitting using Dunlap's robust density fitting formula

The `EXACT` kernels are not optimized and should only be used for reference calculations.

10.6.5 `hf`

The `hf` module computes the Hartree-Fock wave function using the self-consistent field methods. The Fock matrix with the J and K kernels, and different methods can be easily combined. The guess is computed either from the core Hamiltonian, the superposition of atomic densities (SAD) or projection methods (see Annex **??**).

10.6.6 `mp`

The `mp` library contains the kernels for computing the Z matrix encountered in AO-DF-SOS-MP2. They kernels are initialized similarly to the J and K kernels. Two different types have been implemented

- `LLMP_FULL`: computes the Z kernel by constructing the fully transformed $B_{X\underline{\mu\bar{\nu}}}$ tensor and keeping it in core memory
- `LLMP_MEM`: the tensor $B_{X\underline{\mu\bar{\nu}}}$ is recomputed on-the-fly when contracting it with $B_{X\mu\nu}$

10.6.7 `adc`

The final set of kernels is contained in the `adc` library, which compute the AO-CIS and CDD-DF-SOS-ADC(2) matrix-vector product and reuse the J, K and Z kernels from the other modules to evaluate some of the expressions. The `adc` library solves the ADC eigenvalue problem by using the Davidson diagonalization.

10.6.8 `locorb`

The `locorb` module provides routines for localizing molecular orbitals. The available localization schemes are

- Foster-Boys LMOs
- Cholesky LMOs
- Projected atomic orbitals

Furthermore, the library can compute the natural transition orbitals from the CIS or ADC(2) density matrices. The orbital information can be written to a file in `molden` format for plotting in external software.

Chapter 11

Algorithms

The final chapter collects a few of the more important numerical algorithms and solvers used in the MEGALOchem software, with some comments on the implementation details.

11.1 Direct Inversion of The Iterative Subspace

The direct inversion of the iterative subspace (DIIS) method is an acceleration technique for solvers of nonlinear equations, originally introduced by Pulay [219] in the context of the self-consistent field method. It has also found use in the solution of the coupled cluster amplitude equations via Newton-Raphson methods, as well as in the Davidson diagonalization procedure.

At a given iteration i, the DIIS method tries to find a set of coefficients c_i such that the sum of the m previous error vectors

$$\mathbf{e}_{i+1} = \sum_{i=0}^{m} c_i \mathbf{e}_i \tag{11.1}$$

approximates the null vector in a least-squares sense. The new coefficients are then used to extrapolate the solution $\mathbf{p}$ for the next iteration using the current set of solution vectors:

$$\mathbf{p}_{i+1} = \sum_{i=0}^{m} c_i \mathbf{p} \tag{11.2}$$

In the classical DIIS scheme, the coefficients are furthermore required to sum to one

$$\sum_{i}^{m} c_i = 1 \tag{11.3}$$

Finding the coefficients thus corresponds to minimizing the Lagrangian subject to the constraint 11.3:

$$\mathcal{L} = \mathbf{c}^{\dagger}\mathbf{B}\mathbf{c} - \lambda \left(1 - \sum_{i}^{m} c_i \right) \tag{11.4}$$

where $\mathbf{B}$ is the matrix (or *subspace*) containing the overlaps

$$B_{ij} = \langle e_i | \, | e_j \rangle \tag{11.5}$$

Minimizing $\mathcal{L}$ with respect to $\mathbf{c}$ gives

$$\frac{\partial \mathcal{L}}{\partial c_k} = 2\sum_i^m c_i B_{ki} - \lambda = 0 \tag{11.6}$$

which then reduces to the matrix equation

$$\begin{pmatrix} B_{00} & B_{01} & \dots & B_{0m} & -1 \\ B_{10} & B_{11} & \dots & B_{1m} & -1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ B_{m0} & B_{m1} & \dots & B_{mm} & -1 \\ -1 & -1 & \dots & -1 & 0 \end{pmatrix} \begin{pmatrix} c_0 \\ c_2 \\ \vdots \\ c_m \\ \lambda \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ -1 \end{pmatrix} \tag{11.7}$$

The system can be solved by inverting the subspace matrix (hence the name DIIS). The DIIS method is straight-forward to implement. Algorithm 7 gives a summary for the method as it is used in the SCF procedure. The exact form of the error vector (or *residual*) depends on the nature of the problem. In the general case, one can use the quantity

$$\mathbf{e}_i = \mathbf{e}_i - \mathbf{e}_{i-1} \tag{11.8}$$

In the case of the SCF method, the commutator relationship may be used instead:

$$\mathbf{e}_i = \mathbf{F}_i\mathbf{P}_i\mathbf{S} - \mathbf{S}\mathbf{P}_i\mathbf{F}_i \tag{11.9}$$

In some cases, it may be beneficial to remove some error vectors from the subspace to avoid linear dependencies. One can either the remove the first error vector, or the vector with the largest norm. In most quantum chemistry software packages, the maximum DIIS space is set between 8 and 12.

Algorithm 7: DIIS method for SCF

1. Compute the new Fock matrix $\mathbf{F}_i$ for the current iteration i using the density matrix $\mathbf{P}_i$
2. Compute the error vector $\mathbf{e}_i = \mathbf{F}_i\mathbf{P}_i\mathbf{S} - \mathbf{S}\mathbf{P}_i\mathbf{F}_i$
3. Add the error vector $\mathbf{e}_i$ and the $\mathbf{F}_i$ to the trial vectors. If the number of trail vectors is larger than a given threshold DIIS_MAX_SUBSPACE, remove the first vector in the sets $\{\mathbf{F}\}$ and $\{\mathbf{e}\}$, or erase entry k where $\mathbf{e}_k$ is the error with the largest norm.
4. Compute the overlap matrix $\mathbf{B}$ and minimize the Lagrangian
5. Compute a new Fock matrix using the new coefficients $\mathbf{F}_{i+1} = \sum_i^m c_i\mathbf{F}_i$, and diagonalize it to get the new density matrix $\mathbf{P}_{i+1}$.
6. Increment i, begin new cycle

11.2 Davidson Diagonalization

Eigenvalue problems involving large, symmetric matrices are ubiquitous in electronic structure theory. Due to the steep scaling in the number of elements, a full diagonalization

is often not possible. Fortunately, in most cases the interest lies only in a few select eigenvalues, rather than the whole spectrum, and iterative methods may be used that avoid storing the whole matrix. Davidson's diagonalization procedure [220] was originally introduced to extract the first few eigenvalues CI matrix, and is mainly used for eigenvalue problems of large, sparse, diagonally dominant matrices. Davidson's method is part of a larger category called *Krylov subspace* methods. Krylov subspaces help to find approximate solutions to a higher-dimensional problem by projecting the matrix onto a smaller subspace that fits in memory. Other methods in this family include the Lanczos and Arnoldi algorithm, which also find uses in quantum chemistry [221].

11.2.1 Davidson-Liu Method

The Davidson method builds up an iterative subspace representation of the full matrix which corresponds to the overlap $\langle r_i|u_j\rangle$ between the current set of trial vectors $\mathbf{u}$ and the set of matrix-vector products $\mathbf{r} = \mathbf{Au}$. Diagonalizing this subspace matrix gives a set of approximate eigenvalues and eigenvectors. New trial vectors are constructed using the preconditioned residual. A *preconditioner* helps to "steer" the problem into the right direction by better approximating the matrix $\mathbf{A}$ and thus speeding up convergence. In the original paper, Davidson uses $\mathbf{M} = \mathbf{D} - \lambda\mathbf{I}$ as the preconditioner, where $\mathbf{D}$ is the exact diagonal of the matrix. If constructing the diagonal is too expensive, the *Olsen preconditioner* may be used as an alternative, which approximates the diagonal to zeroth order [222]. For example, the diagonal of the singles-singles block is then simply given by the MO energy differences $D_{ia} = \epsilon_i - \epsilon_a$. Other more sophisticated preconditioners have also been investigated [223].

The quality of the starting guess vectors also influences convergence rate. In most quantum chemistry programs, they are constructed by considering the exact or approximate matrix diagonal $\mathbf{D}$. The entries of $\mathbf{D}$ are ordered from highest to lowest norm. The first guess vector is then generated by putting a 1 on the position of the matrix element with the highest norm, with the rest of the elements set to 0. The second, third, ... eigenvectors are constructed in a similar way. Eigenvectors from a lower order calculation can also be used as a starting guess for higher order methods (e.g. using CIS eigenvectors for ADC(2)).

The Davidson method was originally a single-root method, i.e. each root needed a separate optimization. A *blocked* version was proposed by Liu [224], which allows to optimize multiple roots at once. The blocked Davidson-Liu method as implemented in MEGALOchem is given in Algorithm 8. If only a single root is needed, the loops over the states are restricted to a single index instead, where i_{root} corresponds to the desired root index.

Furthermore, the Davidson procedure can be modified such that it follows eigenvectors that have a desired structure. The so-called *root-homing* procedure [225] reorders the eigenvalues and eigenvectors at each iteration according to overlap criteria. If one wishes to preserve the structure of the initial guesses, root-homing is essential. This modification to the Davidson algorithm is not implemented in MEGALOchem.

While the Davidson algorithm avoids storing the whole matrix, it still needs to save the trial vectors and matrix-vector products. As the subspace grows, so does the number of vectors, which might become a memory bottleneck. To limit the number of vectors,

the Davidson subspace can be collapsed and a new set of vectors can be formed according to

$$\mathbf{u}'_i = \sum_j^{n_{dav}} V_{ji}\mathbf{u}_j \tag{11.10}$$

where $\mathbf{V}$ are the eigenvectors of the subspace matrix. This is followed by a normalization step

$$\mathbf{u}_i^{new} = \frac{\mathbf{u}'_i}{\|\mathbf{u}'_i\|} \tag{11.11}$$

Subspace collapse allows to formulate a better estimate of the total memory requirements of the procedure.

11.2.2 Modified Davidson Method for the Pseudo-Eigenvalue Problem

When using doubles-folding in the context of the ADC(2) or CC2 method (section 2.3.4), the effective matrix $\mathbf{A}_{eff}$ becomes dependent on its own eigenvalues:

$$\mathbf{A}_{eff}(\omega)\mathbf{v} = \omega\mathbf{v} \tag{11.12}$$

In this case, the standard Davidson method does not work, and needs to be modified to be able to solve this pseudo-eigenvalue problem. Different approaches have been proposed over the years, but are each based on similar principles [194, 25]. The modified Davidson algorithm is given in Algorithm **??**, as implemented in MEGALOchem.

The algorithm is split into *macro* and *micro* iterations, which are state-specific. The micro-iterations corresponds to the standard single-root Davidson iterations for diagonalizing an effective matrix $\mathbf{A}_{eff}(\omega_i)$ with *fixed* eigenvalue ω_i. When the procedure has converged, ω_i is set to ω'_i, and a new Davidson procedure, or macro-iteration commences with the new effective matrix $\mathbf{A}_{eff}(\omega_i)$. The macro-iterations are repeated until the eigenvectors have converged to a certain threshold. The individual roots are then further converged using DIIS.

11.3 Incomplete Cholesky Decomposition

The Cholesky factorization decomposes a symmetric, positive definite (PD) matrix into a lower and upper triangular matrix:

$$\mathbf{A} = \mathbf{L}\mathbf{L}^T \tag{11.13}$$

where $\mathbf{L}$ has the same dimension as $\mathbf{A}$. However, the Cholesky factorization, as defined in 11.13, does not exist for positive semi-definite (PSD) matrices. To see the reason why, consider Algorithm 10. The loop runs over all columns i of $\mathbf{A}$ and involves divisions by the diagonal elements L_{ii}. For PSD matrices, some of these elements will be zero due to linear dependencies of the column vectors, and the operation is therefore not defined.

A PSD matrix is also called *rank-deficient*. The rank of a symmetric matrix is equal to the number of linearly independent column vectors. The occupied atomic density

Algorithm 8: Davdison-Liu Algorithm. Indices $i, j, k...$ implicitly loop over the number of roots n_{roots}, and indices $\bar{i}, \bar{j}, \bar{k}, ...$ implicitly loop over the whole Davidson subspace n_{dav}.

Input: Number of desired roots n_{root}, guess vectors $\mathbf{U} = \{\mathbf{u}_i\}$, convergence threshold d_{conv}, (approximate) diagonal $\mathbf{D}$, matrix-vector product function mvp_func()

Output: Converged eigenvectors $\{\mathbf{v}_i\}$ and eigenvalues $\{\omega_i\}$

while *not converged* **do**

Compute all the matrix-vector products which have not yet been computed

$$\mathbf{r}_{\bar{i}} = \text{mvp\_func}(\mathbf{u}_{\bar{i}})$$

Form the subspace matrix

$$A_{\bar{i}\bar{j}} = \mathbf{r}_{\bar{i}} \cdot \mathbf{u}_{\bar{j}}$$

Diagonalize $\mathbf{A}$ to get the eigenvectors $\mathbf{V} = \{\mathbf{v}_{\bar{i}}\}$ and eigenvalues $\{\omega_{\bar{i}}\}$

Compute the n_{root} residuals

$$\boldsymbol{\rho}_i = \mathbf{r}_{\bar{j}} V_{\bar{j}i} - \mathbf{u}_{\bar{j}} V_{\bar{j}i} \omega_i$$

Root i has converged if $\|\boldsymbol{\rho}_i\| < d_{conv}$ or $|V(n_{dav} - 1, i)| < d_{conv}$

If all roots have converged, then break, else continue.

Compute the correction vectors $\{\mathbf{e}_i\}$

$$\mathbf{e}_i = \frac{\boldsymbol{\rho}_i}{\mathbf{D} - \omega_i \mathbf{I}}$$

Compute a new set of vectors $\{\mathbf{b}_i\}$ by Gram-Schmidt orthogonalization of $\{\mathbf{d}_i\}$ against the current set of $\{\mathbf{u}_{\bar{i}}\}$

Normalize the new vectors $\{\mathbf{b}_i\}$ and add all vectors to $\{\mathbf{u}_{\bar{i}}\}$ for which $\|\mathbf{b}_i\| \,/\, \|\mathbf{d}_i\|$ are above 1e-3 (to remove linear dependencies)

If the size of the Davidson subspace n_{dav} is above a given threshold, collapse the subspace and form a new set according to Equations 11.10 and 11.11

The final eigenvalues are equal to the eigenvalues of the last subspace matrix. The final eigenvectors $\{\mathbf{v}_i\}$ are formed according to Equations 11.10 and 11.11 using the trial vectors $\{\mathbf{u}_i\}$ and the eigenvectors of the subspace matrix

Algorithm 9: Modified Davdison algorithm with DIIS acceleration for pseudo-eigenvalue problems.

Input: Number of desired roots n_{root}, guess vectors $\{\mathbf{u}_i\}$ and eigenvalues $\{\omega_i^{(0)}\}$, convergence threshold d_{conv}, (approximate) diagonal $\mathbf{D}$, matrix-vector product function `mvp_func()`

Output: Converged eigenvectors $\{\mathbf{v}_i\}$ and eigenvalues $\{\omega_i\}$

Set ω_i^{macro} to $\omega_i^{(0)}$ and start macro-iterations for each individual state i

while *not converged* **do**

  Perform Davidson diagonalization on the effective matrix $\mathbf{A}(\omega_i^{macro})$. Convergence is reached when the difference between eigenvalues ω_i^{micro} from subsequent micro-iterations is smaller than the total change since the start of the procedure, i.e. at iteration n:

$$\mathsf{abs}\left(\omega_i^{macro} - \omega_i^{micro}|_{\text{iter}=0}\right) > \mathsf{abs}\left(\omega_i^{micro}|_{\text{iter}=n} - \omega_i^{micro}|_{\text{iter}=n-1}\right)$$

  If the residual $\boldsymbol{\rho}_i$ is smaller than 1e-3, break, else continue

  Set ω_i^{macro} to ω_i^{micro}, and $\mathbf{u}_i$ to $\mathbf{v}_i^{micro}$

Set ω_i to ω_i^{macro}, start DIIS

while *not converged* **do**

  Compute the matrix-vector product $\mathbf{r}_i = \mathsf{mvp\_prod}(\mathbf{u}_i, \omega_i)$

  Compute the residual

$$\boldsymbol{\rho}_i = \frac{\mathbf{r}_i - \omega_i \mathbf{u}_i}{\|\mathbf{u}_i\|}$$

  Compute the new eigenvalue ω_i

$$\omega_i = \frac{\mathbf{u}_i \cdot \mathbf{r}_i}{\|\mathbf{u}_i\|}$$

  If $\|\boldsymbol{\rho}_i\| < d_{conv}$, root i has converged, break

  Compute the corrected vector $\mathbf{b}_i$ according to

$$\mathbf{b}_i = \mathbf{u}_i + \frac{\boldsymbol{\rho}_i}{\mathbf{D}}$$

  Add $\boldsymbol{\rho}_i$ to the DIIS error vector space, and perform extrapolation on $\boldsymbol{b}_i$ to get a new guess vector $\mathbf{u}_i$

matrix **P** is an example of a rank-deficient matrix. The atomic orbital basis, with N_{bas} functions, is often redundant and has linear dependencies, which translates to a lower rank r ¡ N_{bas} of the occupied density matrix. The rank of **P** is equal to the number of occupied molecular orbitals. The atomic orbital overlap matrix **S** is an example of a "numerically" positive semi-definite matrix. While the column vectors are not strictly linear dependent from a mathematical point of view, they are linearly dependent in the sense of floating-point precision. The diagonals L_{ii} are close to zero (¡ 1e-5), which will lead to large values and reduced accuracy when dividing.

Algorithm 10: Cholesky decomposition without pivoting.

Input: Symmetric matrix **A** with dimension N by N

Output: Cholesky factors **L**

1 $L_{00} \leftarrow \sqrt{A_0 0}$

2 $L_{j0} \leftarrow \frac{a_{j0}}{L_{00}} \quad j \in [1:N]$

3 $L_{ii} \leftarrow \sqrt{A_{ii} - \sum_{k=0}^{i} L_{ik}^2} \quad i \in [1:N]$

4 $L_{ji} \leftarrow \left(A_{ji} - \sum_{k=0}^{i} L_{ik} L_{jk}\right) / L_{ii} \quad i \in [1:N] \quad j \in [i+1:N]$

The Cholesky factorization can be generalized to PSD matrices by introducing permutation matrices which pivot the columns and rows of **A**:

$$\mathbf{PAP}^T = \mathbf{LL}^T \tag{11.14}$$

or

$$\mathbf{A} = \mathbf{P}^T\mathbf{LL}^T\mathbf{P} \tag{11.15}$$

where **L** is a N by k lower triangular matrix with $k = rank(\mathbf{A})$. The permutation matrices swap the diagonal entries L_{ii}, also known as *pivots*, in a way that division by zero is avoided during the procedure. If the pivot is below a certain threshold, the procedure halts and the number of total iterations is equal to the rank of **A**. The *incomplete* pivoted Cholesky factorization is therefore *rank-revaling*. Other examples of rank-revealing decompositions include the pivoted QR decomposition, and the singular value decomposition.

The Cholesky decomposition with full pivoting is given in Algorithm 11. It can also be used in cases where the matrix is nearly positive semi-definite ($N \approx rank(A)$) for extra numerical stability, e.g. the AO overlap matrix. In MEGALOchem, the pivoted Cholesky factorization is used to obtain a set of localized Cholesky MO coefficients which help to reduce the prefactor of atomic orbital methods.

Comments

It should be noted that pivoting destroys the banded structure of the matrix **L** (Figure 3.7). In other words, the pivoted Cholesky factorization of a diagonally dominant, sparse matrix may not give sparse matrices **L** in the block-diagonal form. Blocking significant elements together reduces the number of non-zero blocks, which is crucial for the performance of atomic orbital based methods. It is therefore necessary to reorder the

Algorithm 11: Incomplete Cholesky decomposition with full pivoting.

Input: Symmetric matrix $\mathbf{A}$ with dimension N by N, threshold d_{lindep}
Output: Cholesky factors $\mathbf{L}$ and the rank r of $\mathbf{A}$

Initialize index vector $perm = \{0, 1, 2, ..., N\}$
for $i = 0$ ***to*** N **do**
  Find maximum diagonal element $A_{max} = A_{jj}$
  Swap rows i and j of $\mathbf{A}$
  Swap columns i and j of $\mathbf{A}$
  Swap $perm(i)$ and $perm(j)$
  if $A_{max} < 0$ **then**
    Negative pivot element, the Cholesky decomposition is not possible
  if $abs(A_{max}) < d_{lindep}$ **then**
    $r = i + 1$, break
  $L_{ii} = \sqrt{A_{max}}$
  $L_{ki} = A_{ki}/\sqrt{A_{max}} \quad k \in [i + 1 : N]$
  $A_{kl} = A_{kl} - L_{ki}L_{kl} \quad k, l \in [i + 1 : N]$
Impose original order of the rows, set row $perm(i)$ of the new Cholesky marix $\mathbf{L}'$ to row i of $\mathbf{L} \quad i \in [0 : r]$
Set $\mathbf{L}$ to $\mathbf{L}'$

columns of $\mathbf{P^T L}$ at the end of the procedure. In MEGALOchem, this is done by sorting the columns by their weigth $n = (p_f - p_i)/2$, where p_i is the first significant element in the column, and p_f is the last significant element. The block-diagonal form can then restored for $\mathbf{L}$.

The pivoted Cholesky decomposition in MEGALOchem does not exploit sparsity and scales with $\mathcal{N}^{\in}\nabla\dashv\backslash\|(\mathcal{A})$. While a sparse implementation could be considered, row and column pivoting often destroys sparsity patterns (also known as *fill-in*) and leads to a lot of reordering within the data structures. Pivoting needs to be applied in such a way that fill-in is reduced to keep scaling low. It is not currently considered.

Finally, pivoting also negatively impacts parallelization when using MPI. When columns are swapped, this incurs additional communication overhead between processes. Moreover, global communication is necessary to communicate the position and value of the maximum diagonal element to each process, which further slows down the procedure. Efficient parallelization of matrix decompositions with pivoting is subject of current research [226, 227].

11.4 Laplace Transformation

The Laplace transformation is a crucial step for formulating an orbital invariant MP2 energy expression, where the orbital energy denominator is converted to an exponential form

$$\frac{1}{\epsilon_a + \epsilon_b - \epsilon_i - \epsilon_j} = \frac{1}{x} = \int_0^\infty e^{-xt} dt \tag{11.16}$$

The integral is then approximated by the Laplace quadrature

$$\frac{1}{x} = \sum_{\alpha=0}^{k} \omega^{(\alpha)} e^{-x t^{(\alpha)}} \tag{11.17}$$

where k is the number of quadrature points, $\omega^{(\alpha)}$ are the Laplace weights and $t^{(\alpha)}$ are the Laplace exponents. In their original paper, Häser and Almlöf [**?**] computed the Laplace parameters by least-squares minimization of the error distribution function

$$\eta_k(x, \omega^{(\alpha)}, t^{(\alpha)}) = \sum_{\alpha=0}^{k} \omega^{(\alpha)} e^{-x t^{(\alpha)}} - \frac{1}{x} \tag{11.18}$$

in the interval $[x_{min}, x_{max}]$. Later, it was shown that the quadrature parameters can be computed at a much lower cost using a minimax approximation (MA) [**?**], which *minimizes* the *maximum* Chebychev norm

$$\delta_{k[1,R]}(\bar{\omega}^{(\alpha)}, \bar{t}^{(\alpha)}) = \max_{x \in [1,R]} \left| \eta_k(x, \bar{\omega}^{(\alpha)}, \bar{t}^{(\alpha)}) \right| \tag{11.19}$$

with the scaled Laplace parameters $\bar{\omega} = \omega x_{min}$, $\bar{t} = t x_{min}$ in the new interval $[1, R]$ where $R = x_{max}/x_{min}$. The scaled Laplace coefficients in the minimax approximation are obtained by repeating the following two steps until self-consistency is reached:

- Determine the $2k-1$ extremum points x_i of the error distribution function $\eta_k(x, \bar{\omega}^{(\alpha)}, \bar{t}^{(\alpha)})$ with the current set of $\{\bar{\omega}\}$ and $\{\bar{t}^{(\alpha)}\}$
- Optimize the $2k+1$ parameters $\{\bar{\omega}\}$ and $\{\bar{t}^{(\alpha)}\}$ by solving the $2k+1$ non-linear equations

$$\eta_k(x_i, \bar{\omega}^{(\alpha)}, \bar{t}^{(\alpha)}) = (-1)^i \delta_{k[1,R]}(\bar{\omega}^{(\alpha)}, \bar{t}^{(\alpha)}) \tag{11.20}$$

This procedure is also known as the Remez algorithm (RA). Each of the two steps are non-trivial. The set of non-linear equations in step 2 can be solved by performing a Newton-Raphson minimization using pre-tabulated values of $\{\bar{\omega}\}$ and $\{\bar{t}^{(\alpha)}\}$ as guesses for the first RA iteration. Step 1 is a bit more complicated, and is either solved (1) by first finding the nodal points x_0 of the error distribution function to compute the extremum points using Newton-Raphson [**?**], or (2) by finding the extremum points directly using the Newton-Maehly algorithm [**?**]. For further details, the reader is referred to the original publications.

All calculations in this report use the robust minimax approximation as proposed by Helmich-Paris and Visscher [**?**]. They have published their source code on Github (github.com/bhelmichparis/laplace-minimax), which was in turn incorporated into the MEGALOchem software.

11.5 Cuthill-McKee

The Cuthill-McKee (CM) algorithm finds a permutation P of a sparse, symmetric matrix that minimizes its *bandwidth* [**?**]. A banded matrix is a matrix that has all significant

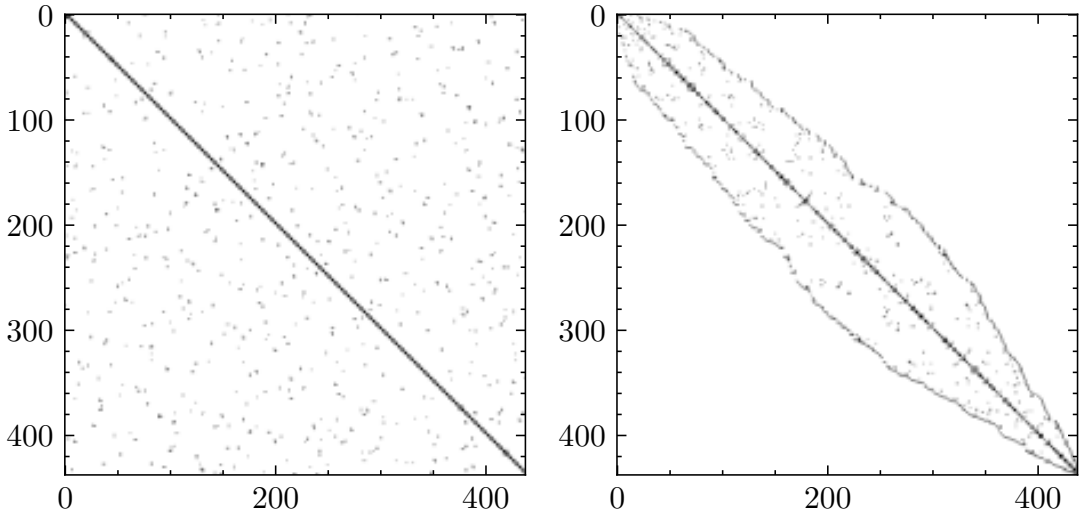


Figure 11.1: Connectivity matrix of FW144 before reordering (left), and connectivity matrix after applying the reverse Cuthill-McKee algorithm (right). Reducing the bandwidth allows to compress matrices and tensors in the AO basis into a much smaller space when using the block-sparse format.

elements clustered around the main diagonal. The bandwidth k is defined as the smallest positive index for which

$$\min_k |A(k,k)| = 0 \qquad k \in [0, N] \tag{11.21}$$

The CM algorithm reduces the bandwidth of a matrix by reordering the nodes of the corresponding adjacency or connectivity matrix $\mathbf{C}$ (Algorithm 12). For a N-by-N matrix, there are N nodes. Node i and j are connected if the entry (i, j) of the connectivity matrix is 1, and not connected if the entry is 0. Here, the *degree* of a node is defined by the total number of connections, i.e. the row or column sum

$$degree(i) = \sum_j^N C(i,j) \tag{11.22}$$

In the context of quantum chemistry, the nodes correspond to atoms in a molecule, and the connections to bonds. Reordering the indexing of the atoms based on the connectivity matrix of a molecule allows to significantly reduce the number of blocks needed to store quantities like the overlap matrix, density matrix or 2-electron repulsion integrals when using block-sparse matrix algebra by grouping close atoms together. Figure 11.1 shows how the connectivity matrix is reordered for the FW144 system using the reverse CM algorithm. Two atoms are "connected", if their are within 5 a_0 of each other.

Algorithm 12: (Reverse) Cuthill-McKee algorithm.

Input: Sparse symmetric matrix **A** with dimension N by N
Output: Reordered matrix **A** with minimized bandwidth.

```
Form the binary connectivity matrix C of the input matrix
Instantiate empty queue Q and result array R
while true do
    Find node p with minimum degree that is not yet in R, and put it into R
    Add all nodes to the queue that are connected to p
    while length of Q ≠ 0 do
        Get fist node q in queue, and pop it from queue
        if q in R then
            skip to next loop
        else
            add q to R
        Append all nodes connected to q not yet in R both to R and Q
    if size of R = N then
        break
Reorder the rows and columns of A according to R (standard Cuthill-McKee) or
  the reverse order of R (reverse Cuthill-McKee)
```

Chapter 12

Conclusion and Outlook

Appendix A

Second Quantization

Creation operator
- One electron
- two electron

Appendix B

Hartree-Fock Starting Guesses

MEGALOchem can use three different starting guesses for the Hartree-Fock procedure: the core Hamiltonian, SAD and projection. This section gives some additional implementation details on the latter two methods

B.1 Superposition of Atomic Densities

The superposition of atomic densities (SAD) is a simple, yet powerful method to generate a HF starting guess that is already very close to the solution [**?**, **?**]. To a very good approximation, a molecule can be seen as a collection of atoms. The electronic guess density $\mathbf{P}$ is then simply the direct sum of the individual atomic densities

$$\mathbf{P} = \bigoplus_{i=0}^{N_{atoms}} \mathbf{P}_i^{atomic} \tag{B.1}$$

The resulting density matrix $\mathbf{P}$ is block-diagonal. The atomic densities are obtained by performing a Hartree-Fock calculation on the individual atoms using partial occupation.

The SAD guess only gives an initial density, but no molecular orbitals, which might be necessary in some cases to construct the Fock matrix, depending on which algorithm is chosen. There are two choices in MEGALOchem to generate starting orbitals:

- Natural orbitals by diagonalization of the guess density
- Local molecular orbitals by performing an incomplete cholesky decomposition with full pivoting on the guess density

The Cholesky decomposition has the advantage of revealing the rank of the matrix, and are therefore much more compact and the default option in MEGALOchem

B.1.1 Partial Occupation Hartree-Fock

There are several different ways to perform the atomic Hartree-Fock calculations. Individual atoms often have unpaired electrons, and may be computed using unrestricted Hartree-Fock. Alternatively, it is possible to perform fractional occupation Hartree-Fock (FOHF)

calculations [**?**]. After diagonalization of the Fock matrix, FOHF scales the coefficient matrices by an occupation vector $\mathbf{v}$

$$\mathbf{C}^{\sigma}_{frac} = \mathbf{C}^{\sigma}\mathbf{v} \tag{B.2}$$

where σ is either α or β spin. In standard HF, all entries in $\mathbf{v}$ are set to 1 for occupied, and 0 for virtual orbitals. FOHF allows fractional values between 0 and 1 for occupied orbitals. The exact form of $\mathbf{v}$ depends on the atom. Consider for example the oxygen atom with configuration $1s^{\uparrow\downarrow}2s^{\uparrow\downarrow}2p_x^{\uparrow\downarrow}2p_y^{\uparrow}2_z^{\uparrow}$, using the aufbau principle. In standard UHF, the single $\downarrow$ (or β) electron in the p orbitals will occupy *either* x, y or z. However, in absence of any external perturbation, the electron should have no preference for any of them. FOHF allows the electron to occupy all orbitals, and the occupation vectors for the occupied AOs are given by

$$\mathbf{v}^{\alpha} = \{1, 1, 1, 1, 1\} \tag{B.3}$$

$$\mathbf{v}^{\beta} = \{1, 1, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}\} \tag{B.4}$$

Through electron-delocalization accross all three p-orbitals, the energy of the FOHF wave function is actually lowered compared to the UHF wave function. FOHF can substantially improve the description of single atoms and open-shell molecules. Restricted FOHF is also possible by simply spin-averaging the α and β occupation vectors:

$$v^{restricted} = \frac{1}{2}\left(\mathbf{v}^{\alpha} + \mathbf{v}^{\beta}\right) \tag{B.5}$$

MEGALOchem supports both unrestricted and restricted FOHF, though only for a single atoms. The occupation numbers for each atom-type are hard-coded and taken from Psi4 [**?**].

B.2 Projection Methods

For Hartree-Fock calculations that encounter convergence difficulties when using large basis sets with many diffuse basis functions, it may be beneficial to first compute the HF wave function in a minimal basis, and then project it onto the larger basis set [**?**]. Let $|\chi^{min}\rangle$ be the atomic orbitals for the minimal basis set and $|\chi^{full}\rangle$ the AOs of the larger basis set. By defining the projection operator for a non-orthogonal AO basis

$$\hat{P} = |\chi_{\mu}\rangle\, S^{-1}_{\mu\nu}\, \langle\chi_{\nu}| \tag{B.6}$$

the MOs computed in the minimal basis set can then be projected onto the larger AO space as

$$\begin{aligned} \hat{P}^{full}\,|\phi_i\rangle &= \hat{P}^{full} \sum_i C^{min}_{\mu i}\, |\chi^{min}_{\mu}\rangle \\ &= (S^{full}_{\mu\nu})^{-1}\, \langle\chi^{full}_{\nu}|\chi^{min}_{\sigma}\rangle\, |\chi^{full}_{\sigma}\rangle \end{aligned} \tag{B.7}$$

Using the above expression, and introducing the cross-overlap matrix $S^{full,min} = \langle\chi^{full}_{\nu}|\chi^{min}_{\sigma}\rangle$ between the two basis sets, the projected coefficient matrix is then computed as

$$\mathbf{C}^{full} = (\mathbf{S}^{full})^{-1}\mathbf{S}^{full,min}\mathbf{C}^{min} \tag{B.8}$$

The projection method is also useful in cases where the SAD guess cannot be used (see C).

Appendix C

Removing Linear Dependencies in Basis Sets

Excited state methods often need larger basis sets with diffuse functions to accurately describe electron transitions into higher lying orbitals. However, large basis sets often introduce linear dependencies which may lead to decreased accuracy, numerical instabilities or even crashes. MEGALOchem can use two different methods to remove linear dependencies from basisn sets.

C.1 Canonical Orthogonalization

Most quantum chemistry programs use canonical orthogonalization [112, **?**] to remove linear dependencies in basis sets. Diagonalization of the AO overlap matrix

$$\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{\dagger} \tag{C.1}$$

gives the eigenvectors $\mathbf{V}$ and the diagonal matrix $\mathbf{\Lambda}$ containing the eigenvalues. For near-linearly dependent basis sets, some the eigenvalues become very small and introduce large numerical errors for subsequent operations such as matrix inversion. Canonical orthogonalization removes all eigenvectors with eigenvalues below a certain threshold (1e-4 to 1e-6).

C.2 Cholesky Decomposition

Linear dependencies may alternatively be removed by an incomplete pivoted Cholesky decomposition of the overlap matrix [**?**]

$$\mathbf{PSP}^{T} = \mathbf{LL}^{T} \tag{C.2}$$

where $\mathbf{P}$ is a permutation matrix, and $\mathbf{L}$ are the Cholesky factors of dimension $N_{AO} \times r$, where r is the rank of $\mathbf{S}$. The Cholesky procedure as given in Algorithm 11) outputs a vector *perm* which contains the pivoting indices. A new basis set is constructed such that each atomic orbital with indices $perm[0:r]$ are included. The remaining functions $perm[r:N]$ are discarded as the pivots lie below the threshold.

Because the removal of linear dependencies from basis sets can significantly alter the basis sets on individual atoms, the SAD method cannot be used for an initial guess, and projective approaches should be used.

- The evil matrix inversion: considerations

Bibliography

[1] Frank Jensen. *Introduction to Computational Chemistry, 3rd Edition.* Wiley, Hoboken, NJ, USA, Feb 2017.

[2] Jochen Schirmer. *Many-Body Methods for Atoms, Molecules and Clusters.* Springer International Publishing, Cham, Switzerland, 2018.

[3] Martin Schütz, Georg Hetzer, and Hans-Joachim Werner. Low-order scaling local electron correlation methods. I. Linear scaling local MP2. *J. Chem. Phys.*, 111(13):5691–5705, Oct 1999.

[4] Dávid Mester, Péter R. Nagy, and Mihály Kállay. Reduced-Scaling Correlation Methods for the Excited States of Large Molecules: Implementation and Benchmarks for the Second-Order Algebraic-Diagrammatic Construction Approach. *J. Chem. Theory Comput.*, 15(11):6111–6126, Nov 2019.

[5] Philipp H. P. Harbach, Michael Wormit, and Andreas Dreuw. The third-order algebraic diagrammatic construction method (ADC(3)) for the polarization propagator for closed-shell molecules: Efficient implementation and benchmarking. *J. Chem. Phys.*, 141(6):064113, Aug 2014.

[6] Caroline M. Krauter, Markus Pernpointner, and Andreas Dreuw. Application of the scaled-opposite-spin approximation to algebraic diagrammatic construction schemes of second order. *J. Chem. Phys.*, 138(4):044107, Jan 2013.

[7] A. B. Trofimov, I. L. Krivdina, J. Weller, and J. Schirmer. Algebraic-diagrammatic construction propagator approach to molecular response properties. *Chem. Phys.*, 1-3(329):1–10, 2006.

[8] Attila Szabo and Neil S. Ostlund. *Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory.* Courier Corporation, Jul 1996.

[9] Trygve Helgaker, Poul Jørgensen, and Jeppe Olsen. *Molecular Electronic-Structure Theory.* Wiley, Aug 2000.

[10] Patrick Norman, Kenneth Ruud, and Trond Saue. *Principles and Practices of Molecular Properties: Theory, Modeling, and Simulations.* Wiley, Hoboken, NJ, USA, Mar 2018.

[11] D. Young. *Computational Chemistry.* Wiley, Chichester, England, UK, 2004.

[12] Stefan Grimme. Improved second-order Møller–Plesset perturbation theory by separate scaling of parallel- and antiparallel-spin pair correlation energies. *J. Chem. Phys.*, 118(20):9095–9102, May 2003.

[13] Stefan Grimme, Lars Goerigk, and Reinhold F. Fink. Spin-component-scaled electron correlation methods. *WIREs Comput. Mol. Sci.*, 2(6):886–906, Nov 2012.

[14] T. P. M. Goumans, Andreas W. Ehlers, Koop Lammertsma, Ernst-Ulrich Würthwein, and Stefan Grimme. Improved Reaction and Activation Energies of [4+2] Cycloadditions, [3,3] Sigmatropic Rearrangements and Electrocyclizations with the Spin-Component-Scaled MP2 Method. *Chemistry – A European Journal*, 10(24):6468–6475, Dec 2004.

[15] Rosa E. Bulo, Helen Jansen, Andreas W. Ehlers, Fransiscus J. J. de Kanter, Marius Schakel, Martin Lutz, Anthony L. Spek, and Koop Lammertsma. The Circumambulation of a Phosphirane: Taking 9-Phenyl-9-phosphabicyclo[6.1.0]nona-2,4,6-triene for a "Walk". *Angew. Chem.*, 116(6):732–735, Jan 2004.

[16] Mareike Gerenkamp and Stefan Grimme. Spin-component scaled second-order Møller–Plesset perturbation theory for the calculation of molecular geometries and harmonic vibrational frequencies. *Chem. Phys. Lett.*, 392(1):229–235, Jul 2004.

[17] Ágnes Szabados. Theoretical interpretation of Grimme's spin-component-scaled second order Møller-Plesset theory. *J. Chem. Phys.*, 125(21):214105, Dec 2006.

[18] Reinhold F. Fink. Spin-component-scaled Møller–Plesset (SCS-MP) perturbation theory: A generalization of the MP approach with improved properties. *J. Chem. Phys.*, 133(17):174113, Nov 2010.

[19] Yousung Jung, Rohini C. Lochan, Anthony D. Dutoi, and Martin Head-Gordon. Scaled opposite-spin second order Møller-Plesset correlation energy: an economical electronic structure method. *J. Chem. Phys.*, 121(20):9793–9802, Nov 2004.

[20] Rohini C. Lochan, Yousung Jung, and Martin Head-Gordon. Scaled opposite spin second order Møller-Plesset theory with improved physical description of long-range dispersion interactions. *J. Phys. Chem. A*, 109(33):7598–7605, Aug 2005.

[21] Robert A. Distasio J. R.. and Martin Head-Gordon. Optimized spin-component scaled second-order Møller-Plesset perturbation theory for intermolecular interaction energies. *Mol. Phys.*, 105(8):1073–1083, Apr 2007.

[22] J. Grant Hill and James A. Platts. Spin-Component Scaling Methods for Weak and Stacking Interactions. *J. Chem. Theory Comput.*, 3(1):80–85, Jan 2007.

[23] Ove Christiansen, Henrik Koch, and Poul Jørgensen. The second-order approximate coupled cluster singles and doubles model CC2. *Chem. Phys. Lett.*, 243(5):409–418, Sep 1995.

[24] Arnim Hellweg, Sarah A. Grün, and Christof Hättig. Benchmarking the performance of spin-component scaled CC2 in ground and electronically excited states. *Phys. Chem. Chem. Phys.*, 10(28):4119–4127, Jul 2008.

[25] Nina O. C. Winter and Christof Hättig. Scaled opposite-spin CC2 for ground and excited states with fourth order scaling computational costs. *J. Chem. Phys.*, 134(18):184101, May 2011.

[26] Attila Tajti and Péter G. Szalay. Accuracy of Spin-Component-Scaled CC2 Excitation Energies and Potential Energy Surfaces. *J. Chem. Theory Comput.*, 15(10):5523–5531, Oct 2019.

[27] Henrik Koch, Ove Christiansen, Poul Jo/rgensen, Alfredo M. Sanchez de Merás, and Trygve Helgaker. The CC3 model: An iterative coupled cluster approach including connected triples. *J. Chem. Phys.*, 106(5):1808–1818, Feb 1997.

[28] Krishnan Raghavachari, Gary W. Trucks, John A. Pople, and Martin Head-Gordon. A fifth-order perturbation comparison of electron correlation theories. *Chem. Phys. Lett.*, 157(6):479–483, May 1989.

[29] Leticia González, Daniel Escudero, and Luis Serrano-Andrés. Progress and Challenges in the Calculation of Electronic Excited States. *ChemPhysChem*, 13(1):28–51, Jan 2012.

[30] Patrick Norman and Andreas Dreuw. Simulating X-ray Spectroscopies and Calculating Core-Excited States of Molecules. *Chem. Rev.*, 118(15):7208–7248, Aug 2018.

[31] Annapaola Migani, Lluís Blancafort, Michael A. Robb, and Anthony D. DeBellis. An Extended Conical Intersection Seam Associated with a Manifold of Decay Paths: Excited-State Intramolecular Proton Transfer in O-Hydroxybenzaldehyde. *J. Am. Chem. Soc.*, 130(22):6932–6933, Jun 2008.

[32] Spiridoula Matsika and Pascal Krause. Nonadiabatic Events and Conical Intersections. *Annu. Rev. Phys. Chem.*, 62(1):621–643, Mar 2011.

[33] Jaehee Kim, Hongli Tao, Todd J. Martinez, and Phil Bucksbaum. Ab initio multiple spawning on laser-dressed states: a study of 1,3-cyclohexadiene photoisomerization via light-induced conical intersections. *J. Phys. B: At. Mol. Opt. Phys.*, 48(16):164003, Jul 2015.

[34] Xiaolei Zhu and David R. Yarkony. Non-adiabaticity: the importance of conical intersections. *Mol. Phys.*, 114(13):1983–2013, Jul 2016.

[35] P. S. Bagus. Self-Consistent-Field Wave Functions for Hole States of Some Ne-Like and Ar-Like Ions. *Phys. Rev.*, 139(3A):A619–A634, Aug 1965.

[36] P. W. Deutsch and L. A. Curtiss. Ab initio calculation of the K-shell excitation and ionization energies of CH4, NH3, H2O, and HF. *Chem. Phys. Lett.*, 39(3):588–592, May 1976.

[37] Maximilien A. Ambroise and Frank Jensen. Probing Basis Set Requirements for Calculating Core Ionization and Core Excitation Spectroscopy by the Δ Self-Consistent-Field Approach. *J. Chem. Theory Comput.*, 15(1):325–337, Jan 2019.

[38] L. Triguero, O. Plashkevych, L. G. M. Pettersson, and H. Ågren. Separate state vs. transition state Kohn-Sham calculations of X-ray photoelectron binding energies and chemical shifts. *J. Electron Spectrosc. Relat. Phenom.*, 1-3(104):195–207, 1999.

[39] Yuji Takahata and Delano P. Chong. DFT calculation of core-electron binding energies. *J. Electron Spectrosc. Relat. Phenom.*, 133(1):69–76, Nov 2003.

[40] Nicholas A. Besley, Andrew T. B. Gilbert, and Peter M. W. Gill. Self-consistent-field calculations of core excited states. *J. Chem. Phys.*, 130(12):124308, Mar 2009.

[41] Alf Holme, Knut J. Børve, Leif J. Sæthre, and T. Darrah Thomas. Accuracy of Calculated Chemical Shifts in Carbon 1s Ionization Energies from Single-Reference ab Initio Methods and Density Functional Theory. *J. Chem. Theory Comput.*, 7(12):4104–4114, Dec 2011.

[42] Luca Schio, Cui Li, Susanna Monti, Peter Salén, Vasyl Yatsyna, Raimund Feifel, Michele Alagia, Robert Richter, Stefano Falcinelli, Stefano Stranges, Vitali Zhaunerchyk, and Vincenzo Carravetta. NEXAFS and XPS studies of nitrosyl chloride. *Phys. Chem. Chem. Phys.*, 17(14):9040–9048, Mar 2015.

[43] Xuechen Zheng and Lan Cheng. Performance of Delta-Coupled-Cluster Methods for Calculations of Core-Ionization Energies of First-Row Elements. *J. Chem. Theory Comput.*, 15(9):4945–4955, Sep 2019.

[44] Andrew T. B. Gilbert, Nicholas A. Besley, and Peter M. W. Gill. Self-consistent field calculations of excited states using the maximum overlap method (MOM). *J. Phys. Chem. A*, 112(50):13164–13171, Dec 2008.

[45] Ernest R. Davidson. Single-Configuration Calculations on Excited States of Helium. *J. Chem. Phys.*, 41(3):656–658, Aug 1964.

[46] Ernest R. Davidson. Single-Configuration Calculations on Excited States of Helium. II. *J. Chem. Phys.*, 42(12):4199–4200, Jun 1965.

[47] Akio Kotani Frank de Groot. *Core Level Spectroscopy of Solids.* Taylor & Francis, Andover, England, UK, Mar 2008.

[48] Tom Ziegler, Arvi Rauk, and Evert J. Baerends. On the calculation of multiplet energies by the hartree-fock-slater method. *Theor. Chim. Acta*, 43(3):261–271, Sep 1977.

[49] Jochen Schirmer. Beyond the random-phase approximation: A new approximation scheme for the polarization propagator. *Phys. Rev. A*, 26(5):2395–2416, Nov 1982.

[50] Gordon Baym. Self-Consistent Approximations in Many-Body Systems. *Phys. Rev.*, 127(4):1391–1401, Aug 1962.

[51] Yoichiro Nambu. Force Potentials in Quantum Field Theory. *Prog. Theor. Phys.*, 5(4):614–633, Jul 1950.

[52] E. E. Salpeter and H. A. Bethe. A Relativistic Equation for Bound-State Problems. *Phys. Rev.*, 84(6):1232–1242, Dec 1951.

[53] Richard P. Feynman. The Development of the Space-Time View of Quantum Electrodynamics. *Science*, 153(3737):699–708, Aug 1966.

[54] A. B. Trofimov and J. Schirmer. An efficient polarization propagator approach to valence electron excitation spectra. *J. Phys. B: At. Mol. Opt. Phys.*, 28(12):2299–2324, Jun 1995.

[55] Jochen Schirmer. Closed-form intermediate representations of many-body propagators and resolvent matrices. *Phys. Rev. A*, 43(9):4647–4659, May 1991.

[56] J. Schirmer and A. B. Trofimov. Intermediate state representation approach to physical properties of electronically excited molecules. *J. Chem. Phys.*, 120(24):11449–11464, Jun 2004.

[57] S. Knippenberg, D. R. Rehn, M. Wormit, J. H. Starcke, I. L. Rusakova, A. B. Trofimov, and A. Dreuw. Calculations of nonlinear response properties using the intermediate state representation and the algebraic-diagrammatic construction polarization propagator approach: two-photon absorption spectra. *J. Chem. Phys.*, 136(6):064107., Feb 2012.

[58] Henrik Koch and Poul Jo/rgensen. Coupled cluster response functions. *J. Chem. Phys.*, 93(5):3333–3344, Sep 1990.

[59] Andreas Dreuw and Martin Head-Gordon. Single-Reference ab Initio Methods for the Calculation of Excited States of Large Molecules. *Chem. Rev.*, 105(11):4009–4037, Nov 2005.

[60] Adèle D. Laurent and Denis Jacquemin. TD-DFT benchmarks: A review. *Int. J. Quantum Chem.*, 113(17):2019–2039, Sep 2013.

[61] Ove Christiansen, Poul Jørgensen, and Christof Hättig. Response functions from Fourier component variational perturbation theory applied to a time-averaged quasienergy. *Int. J. Quantum Chem.*, 68(1):1–52, Jan 1998.

[62] Pierre-François Loos and Denis Jacquemin. Is ADC(3) as Accurate as CC3 for Valence and Rydberg Transition Energies? *J. Phys. Chem. Lett.*, 11(3):974–980, Feb 2020.

[63] Christof Hättig. Structure Optimizations for Excited States with Correlated Second-Order Methods: CC2 and ADC(2). In *Advances in Quantum Chemistry*, volume 50, pages 37–60. Academic Press, Cambridge, MA, USA, Jan 2005.

[64] Martin Head-Gordon, Rudolph J. Rico, Manabu Oumi, and Timothy J. Lee. A doubles correction to electronic excited states from configuration interaction in the space of single substitutions. *Chem. Phys. Lett.*, 219(1):21–29, Mar 1994.

[65] Jan Geertsen, Magnus Rittby, and Rodney J. Bartlett. The equation-of-motion coupled-cluster method: Excitation energies of Be and CO. *Chem. Phys. Lett.*, 164(1):57–62, Dec 1989.

[66] K. Emrich. An extension of the coupled cluster formalism to excited states (I). *Nucl. Phys. A*, 351(3):379–396, Jan 1981.

[67] John F. Stanton and Rodney J. Bartlett. The equation of motion coupled-cluster method. A systematic biorthogonal approach to molecular excitation energies, transition probabilities, and excited state properties. *J. Chem. Phys.*, 98(9):7029–7039, May 1993.

[68] John F. Stanton and Jürgen Gauss. Analytic energy derivatives for ionized states described by the equation-of-motion coupled cluster method. *J. Chem. Phys.*, 101(10):8938–8944, Nov 1994.

[69] Anna I. Krylov. Equation-of-Motion Coupled-Cluster Methods for Open-Shell and Electronically Excited Species: The Hitchhiker's Guide to Fock Space. *Annu. Rev. Phys. Chem.*, 59(1):433–462, Apr 2008.

[70] C. C. J. Roothaan. New Developments in Molecular Orbital Theory. *Rev. Mod. Phys.*, 23(2):69–89, Apr 1951.

[71] Marco Häser and Reinhart Ahlrichs. Improvements on the direct SCF method. *J. Comput. Chem.*, 10(1):104–111, Jan 1989.

[72] W. Kohn. Analytic Properties of Bloch Waves and Wannier Functions. *Phys. Rev.*, 115(4):809–821, Aug 1959.

[73] Sohrab Ismail-Beigi and T. A. Arias. Locality of the Density Matrix in Metals, Semiconductors, and Insulators. *Phys. Rev. Lett.*, 82(10):2127–2130, Mar 1999.

[74] S. Goedecker and L. Colombo. Efficient Linear Scaling Algorithm for Tight-Binding Molecular Dynamics. *Phys. Rev. Lett.*, 73(1):122–125, Jul 1994.

[75] S. Goedecker. Decay properties of the finite-temperature density matrix in metals. *Phys. Rev. B*, 58(7):3501–3502, Aug 1998.

[76] S. N. Taraskin, P. A. Fry, Xiaodong Zhang, D. A. Drabold, and S. R. Elliott. Spatial decay of the single-particle density matrix in tight-binding metals: Analytic results in two dimensions. *Phys. Rev. B*, 66(23):233101, Dec 2002.

[77] David S. Hollman, Henry F. Schaefer, and Edward F. Valeev. Fast construction of the exchange operator in an atom-centred basis with concentric atomic density fitting. *Mol. Phys.*, 115(17-18):2065–2076, Sep 2017.

[78] Frank Neese, Frank Wennmohs, Andreas Hansen, and Ute Becker. Efficient, approximate and parallel Hartree–Fock and hybrid DFT calculations. A 'chain-of-spheres' algorithm for the Hartree–Fock exchange. *Chem. Phys.*, 356(1):98–109, Feb 2009.

[79] J. L. Whitten. Coulombic potential energy integrals and approximations. *J. Chem. Phys.*, 58(10):4496–4501, May 1973.

[80] E. J. Baerends, D. E. Ellis, and P. Ros. Self-consistent molecular Hartree—Fock—Slater calculations I. The computational procedure. *Chem. Phys.*, 2(1):41–51, Sep 1973.

[81] O. Vahtras, J. Almlöf, and M. W. Feyereisen. Integral approximations for LCAO-SCF calculations. *Chem. Phys. Lett.*, 213(5):514–518, Oct 1993.

[82] C.-K. Skylaris, L. Gagliardi, N. C. Handy, A. G. Ioannou, S. Spencer, and A. Willetts. On the resolution of identity Coulomb energy approximation in density functional theory. *J. Mol. Struct. THEOCHEM*, 501-502:229–239, Apr 2000.

[83] B. I. Dunlap. Robust variational fitting: Gáspár's variational exchange can accurately be treated analytically. *J. Mol. Struct. THEOCHEM*, 501-502:221–228, Apr 2000.

[84] David S. Hollman, Henry F. Schaefer, and Edward F. Valeev. A tight distance-dependent estimator for screening three-center Coulomb integrals over Gaussian basis functions. *J. Chem. Phys.*, 142(15):154106, Apr 2015.

[85] David P. Tew. Communication: Quasi-robust local density fitting. *J. Chem. Phys.*, 148(1):011102, Jan 2018.

[86] Yousung Jung, Alex Sodt, Peter M. W. Gill, and Martin Head-Gordon. Auxiliary basis expansions for large-scale electronic structure calculations. *Proc. Natl. Acad. Sci. U.S.A.*, 102(19):6692–6697, May 2005.

[87] P. M. W. Gill, A. T. B. Gilbert, S. W. Taylor, G. Friesecke, and M. Head-Gordon. Decay behavior of least-squares coefficients in auxiliary basis expansions. *J. Chem. Phys.*, 123(6):061101, Aug 2005.

[88] Simen Reine, Erik Tellgren, Andreas Krapp, Thomas Kjærgaard, Trygve Helgaker, Branislav Jansik, Stinne Høst, and Paweł Salek. Variational and robust density fitting of four-center two-electron integrals in local metrics. *J. Chem. Phys.*, 129(10):104101, Sep 2008.

[89] Alex Sodt and Martin Head-Gordon. Hartree-Fock exchange computed using the atomic resolution of the identity approximation. *J. Chem. Phys.*, 128(10):104106, Mar 2008.

[90] Patrick Merlot, Thomas Kjærgaard, Trygve Helgaker, Roland Lindh, Francesco Aquilante, Simen Reine, and Thomas Bondo Pedersen. Attractive electron–electron interactions within robust local fitting approximations. *J. Comput. Chem.*, 34(17):1486–1496, Jun 2013.

[91] Samuel F. Manzer, Evgeny Epifanovsky, and Martin Head-Gordon. Efficient Implementation of the Pair Atomic Resolution of the Identity Approximation for Exact Exchange for Hybrid and Range-Separated Density Functionals. *J. Chem. Theory Comput.*, 11(2):518–527, Feb 2015.

[92] Arno Förster, Mirko Franchini, Erik van Lenthe, and Lucas Visscher. A Quadratic Pair Atomic Resolution of the Identity Based SOS-AO-MP2 Algorithm Using Slater Type Orbitals. *J. Chem. Theory Comput.*, 16(2):875–891, Feb 2020.

[93] Florian Weigend. A fully direct RI-HF algorithm: Implementation, optimised auxiliary basis sets, demonstration of accuracy and efficiency. *Phys. Chem. Chem. Phys.*, 4(18):4285–4291, Sep 2002.

[94] Florian Weigend. Hartree–Fock exchange fitting basis sets for H to Rn †. *J. Comput. Chem.*, 29(2):167–175, Jan 2008.

[95] Florian Weigend, Marco Häser, Holger Patzelt, and Reinhart Ahlrichs. RI-MP2: optimized auxiliary basis sets and demonstration of efficiency. *Chem. Phys. Lett.*, 294(1):143–152, Sep 1998.

[96] David E. Bernholdt and Robert J. Harrison. Fitting basis sets for the RI-MP2 approximate second-order many-body perturbation theory method. *J. Chem. Phys.*, 109(5):1593–1600, Aug 1998.

[97] Francesco Aquilante, Roland Lindh, and Thomas Bondo Pedersen. Unbiased auxiliary basis sets for accurate two-electron integral approximations. *J. Chem. Phys.*, 127(11):114107, Sep 2007.

[98] Francesco Aquilante, Laura Gagliardi, Thomas Bondo Pedersen, and Roland Lindh. Atomic Cholesky decompositions: A route to unbiased auxiliary basis sets for density fitting approximation with tunable accuracy and efficiency. *J. Chem. Phys.*, 130(15):154107, Apr 2009.

[99] George B. Arfken, Hans J. Weber, and Frank E. Harris. *Mathematical Methods for Physicists*. Elsevier, Academic Press, 2012.

[100] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *J. Comput. Phys.*, 73(2):325–348, Dec 1987.

[101] Leslie Greengard. Fast Algorithms for Classical Physics. *Science*, 265(5174):909–914, Aug 1994.

[102] Hong-Qiang Ding, Naoki Karasawa, and William A. Goddard. Atomic level simulations on a million particles: The cell multipole method for Coulomb and London nonbond interactions. *J. Chem. Phys.*, 97(6):4309–4315, Sep 1992.

[103] Christopher A. White, Benny G. Johnson, Peter M. W. Gill, and Martin Head-Gordon. Linear scaling density functional calculations via the continuous fast multipole method. *Chem. Phys. Lett.*, 253(3):268–278, May 1996.

[104] James J. P. Stewart. An examination of the nature of localized molecular orbitals and their value in understanding various phenomena that occur in organic chemistry. *J. Mol. Model.*, 25(1):1–17, Jan 2019.

[105] S. F. Boys. Construction of Some Molecular Orbitals to Be Approximately Invariant for Changes from One Molecule to Another. *Rev. Mod. Phys.*, 32(2):296–299, Apr 1960.

[106] János Pipek and Paul G. Mezey. A fast intrinsic localization procedure applicable for ab initio and semiempirical linear combination of atomic orbital wave functions. *J. Chem. Phys.*, 90(9):4916–4926, May 1989.

[107] Joseph E. Subotnik, Yihan Shao, WanZhen Liang, and Martin Head-Gordon. An efficient method for calculating maxima of homogeneous functions of orthogonal matrices: Applications to localized occupied orbitals. *J. Chem. Phys.*, 121(19):9220–9229, Nov 2004.

[108] Francesco Aquilante, Thomas Bondo Pedersen, Alfredo Sánchez de Merás, and Henrik Koch. Fast noniterative orbital localization for large molecules. *J. Chem. Phys.*, 125(17):174101, Nov 2006.

[109] Joseph E. Subotnik, Anthony D. Dutoi, and Martin Head-Gordon. Fast localized orthonormal virtual orbitals which depend smoothly on nuclear coordinates. *J. Chem. Phys.*, 123(11):114108, Sep 2005.

[110] S. Saebo and P. Pulay. Local Treatment of Electron Correlation. *Annu. Rev. Phys. Chem.*, 44(1):213–236, Oct 1993.

[111] Ove Christiansen, Pekka Manninen, Poul Jørgensen, and Jeppe Olsen. Coupled-cluster theory in a projected atomic orbital basis. *J. Chem. Phys.*, 124(8):084103, Feb 2006.

[112] Per-Olov Löwdin and Harrison Shull. Natural Orbitals in the Quantum Theory of Two-Electron Systems. *Phys. Rev.*, 101(6):1730–1739, Mar 1956.

[113] Alan E. Reed and Frank Weinhold. Natural bond orbital analysis of near-Hartree–Fock water dimer. *J. Chem. Phys.*, 78(6):4066–4073, Mar 1983.

[114] Frank Weinhold and Clark R. Landis. NATURAL BOND ORBITALS AND EXTENSIONS OF LOCALIZED BONDING CONCEPTS. *Chem. Educ. Res. Pract.*, 2(2):91–104, May 2001.

[115] Eric D. Glendening, Clark R. Landis, and Frank Weinhold. Natural bond orbital methods. *WIREs Comput. Mol. Sci.*, 2(1):1–42, Jan 2012.

[116] Tery L. Barr and Ernest R. Davidson. Nature of the Configuration-Interaction Method in Ab Initio Calculations. I. Ne Ground State. *Phys. Rev. A*, 1(3):644–658, Mar 1970.

[117] Carlos Sosa, Jan Geertsen, Gary W. Trucks, Rodney J. Bartlett, and James A. Franz. Selection of the reduced virtual space for correlated calculations. An application to the energy and dipole moment of H2O. *Chem. Phys. Lett.*, 159(2):148–154, Jul 1989.

[118] Andrew G. Taube and Rodney J. Bartlett. Frozen Natural Orbitals: Systematic Basis Set Truncation for Coupled-Cluster Theory. *Collect. Czech. Chem. Commun.*, 70(6):837–850, 2005.

[119] Andrew G. Taube and Rodney J. Bartlett. Frozen natural orbital coupled-cluster theory: Forces and application to decomposition of nitroethane. *J. Chem. Phys.*, 128(16):164101, Apr 2008.

[120] A. V. Luzanov, A. A. Sukhorukov, and V. É. Umanskii. Application of transition density matrix for analysis of excited states. *Theor. Exp. Chem.*, 10(4):354–361, Jul 1976.

[121] Richard L. Martin. Natural transition orbitals. *J. Chem. Phys.*, 118(11):4775–4777, Mar 2003.

[122] Pablo Baudin and Kasper Kristensen. Correlated natural transition orbital framework for low-scaling excitation energy calculations (CorNFLEx). *J. Chem. Phys.*, 146(21):214114, Jun 2017.

[123] Sebastian Höfener and Wim Klopper. Natural transition orbitals for the calculation of correlation and excitation energies. *Chem. Phys. Lett.*, 679:52–59, Jul 2017.

[124] Robert Send, Ville R. I. Kaila, and Dage Sundholm. Reduction of the virtual space for coupled-cluster excitation energies of large molecules and embedded systems. *J. Chem. Phys.*, 134(21):214114, Jun 2011.

[125] Svein Sæbø and Peter Pulay. Local configuration interaction: An efficient approach for larger molecules. *Chem. Phys. Lett.*, 113(1):13–18, Jan 1985.

[126] C. Edmiston and M. Krauss. Configuration-Interaction Calculation of H3 and H2. *J. Chem. Phys.*, 42(3):1119–1120, Feb 1965.

[127] Wilfried Meyer. Ionization energies of water from PNO-CI calculations. *Int. J. Quantum Chem.*, 5(S5):341–348, Jan 1971.

[128] Wilfried Meyer. PNO–CI Studies of electron correlation effects. I. Configuration expansion by means of nonorthogonal orbitals, and application to the ground state and ionized states of methane. *J. Chem. Phys.*, 58(3):1017–1035, Feb 1973.

[129] Christine Krause and Hans-Joachim Werner. Comparison of explicitly correlated local coupled-cluster methods with various choices of virtual orbitals. *Phys. Chem. Chem. Phys.*, 14(21):7591–7604, May 2012.

[130] Martin Schütz and Hans-Joachim Werner. Low-order scaling local electron correlation methods. IV. Linear scaling local coupled-cluster (LCCSD). *J. Chem. Phys.*, 114(2):661–681, Jan 2001.

[131] Martin Schütz and Hans-Joachim Werner. Local perturbative triples correction (T) with linear cost scaling. *Chem. Phys. Lett.*, 318(4):370–378, Feb 2000.

[132] James W. Boughton and Peter Pulay. Comparison of the boys and Pipek–Mezey localizations in the local correlation approach and automatic virtual basis selection. *J. Comput. Chem.*, 14(6):736–740, Jun 1993.

[133] Ricardo A. Mata, Hans-Joachim Werner, Stephan Thiel, and Walter Thiel. Toward accurate barriers for enzymatic reactions: QM/MM case study on p-hydroxybenzoate hydroxylase. *J. Chem. Phys.*, 128(2):025104, Jan 2008.

[134] Frank Neese, Andreas Hansen, and Dimitrios G. Liakos. Efficient and accurate approximations to the local coupled cluster singles doubles method using a truncated pair natural orbital basis. *J. Chem. Phys.*, 131(6):064103, Aug 2009.

[135] Frank Neese, Frank Wennmohs, and Andreas Hansen. Efficient and accurate local approximations to coupled-electron pair approaches: An attempt to revive the pair natural orbital method. *J. Chem. Phys.*, 130(11):114108, Mar 2009.

[136] Andreas Hansen, Dimitrios G. Liakos, and Frank Neese. Efficient and accurate local single reference correlation methods for high-spin open-shell molecules using pair natural orbitals. *J. Chem. Phys.*, 135(21):214102., Dec 2011.

[137] Christoph Riplinger and Frank Neese. An efficient and near linear scaling pair natural orbital based local coupled cluster method. *J. Chem. Phys.*, 138(3):034106, Jan 2013.

[138] Jun Yang, Yuki Kurashige, Frederick R. Manby, and Garnet K. L. Chan. Tensor factorizations of local second-order Møller–Plesset theory. *J. Chem. Phys.*, 134(4):044123, Jan 2011.

[139] Gunnar Schmitz, Benjamin Helmich, and Christof Hättig. A scaling PNO–MP2 method using a hybrid OSV–PNO approach with an iterative direct generation of OSVs†. *Mol. Phys.*, 111(16-17):2463–2476, Sep 2013.

[140] Matthew C. Strain, Gustavo E. Scuseria, and Michael J. Frisch. Achieving Linear Scaling for the Electronic Quantum Coulomb Problem. *Science*, 271(5245):51–53, Jan 1996.

[141] Matt Challacombe, Eric Schwegler, and Jan Almlöf. Fast assembly of the Coulomb matrix: A quantum chemical tree code. *J. Chem. Phys.*, 104(12):4685–4698, Mar 1996.

[142] Yihan Shao and Martin Head-Gordon. An improved J matrix engine for density functional theory calculations. *Chem. Phys. Lett.*, 323(5):425–433, Jun 2000.

[143] Yihan Shao, Christopher A. White, and Martin Head-Gordon. Efficient evaluation of the Coulomb force in density-functional theory calculations. *J. Chem. Phys.*, 114(15):6572–6577, Apr 2001.

[144] David S. Hollman, Henry F. Schaefer, and Edward F. Valeev. Semi-exact concentric atomic density fitting: Reduced cost and increased accuracy compared to standard density fitting. *J. Chem. Phys.*, 140(6):064109, Feb 2014.

[145] Eric Schwegler, Matt Challacombe, and Martin Head-Gordon. Linear scaling computation of the Fock matrix. II. Rigorous bounds on exchange integrals and incremental Fock build. *J. Chem. Phys.*, 106(23):9708–9717, Jun 1997.

[146] Christian Ochsenfeld, Christopher A. White, and Martin Head-Gordon. Linear and sublinear scaling formation of Hartree–Fock-type exchange matrices. *J. Chem. Phys.*, 109(5):1663–1669, Aug 1998.

[147] Robert Polly, Hans-Joachim Werner ∗, Frederick R. Manby, and Peter J. Knowles. Fast Hartree–Fock theory using local density fitting approximations. *Mol. Phys.*, 102(21-22):2311–2321, Nov 2004.

[148] Daniel Mejía-Rodríguez and Andreas M. Köster. Robust and efficient variational fitting of Fock exchange. *J. Chem. Phys.*, 141(12):124114, Sep 2014.

[149] Frank Neese, Frank Wennmohs, Andreas Hansen, and Ute Becker. Efficient, approximate and parallel Hartree–Fock and hybrid DFT calculations. A 'chain-of-spheres' algorithm for the Hartree–Fock exchange. *Chem. Phys.*, 356(1):98–109, Feb 2009.

[150] Róbert Izsák and Frank Neese. An overlap fitted chain of spheres exchange method. *J. Chem. Phys.*, 135(14):144105, Oct 2011.

[151] Manuel Guidon, Jürg Hutter, and Joost VandeVondele. Auxiliary Density Matrix Methods for Hartree-Fock Exchange Calculations. *J. Chem. Theory Comput.*, 6(8):2348–2364, Aug 2010.

[152] Jaehoon Kim and Yousung Jung. A perspective on the density matrix purification for linear scaling electronic structure calculations. *Int. J. Quantum Chem.*, 116(8):563–568, Apr 2016.

[153] R. McWeeny. Hartree-Fock Theory with Nonorthogonal Basis Functions. *Phys. Rev.*, 114(6):1528–1529, Jun 1959.

[154] Adam H. R. Palser and David E. Manolopoulos. Canonical purification of the density matrix in electronic-structure theory. *Phys. Rev. B*, 58(19):12704–12711, Nov 1998.

[155] Anders M. N. Niklasson, C. J. Tymczak, and Matt Challacombe. Trace resetting density matrix purification in O(N) self-consistent-field theory. *J. Chem. Phys.*, 118(19):8611–8620, May 2003.

[156] X.-P. Li, R. W. Nunes, and David Vanderbilt. Density-matrix electronic-structure method with linear system-size scaling. *Phys. Rev. B*, 47(16):10891–10894, Apr 1993.

[157] Murray S. Daw. Model for energetics of solids based on the density matrix. *Phys. Rev. B*, 47(16):10895–10898, Apr 1993.

[158] R. W. Nunes and David Vanderbilt. Generalization of the density-matrix method to a nonorthogonal basis. *Phys. Rev. B*, 50(23):17611–17614, Dec 1994.

[159] Yihan Shao, Chandra Saravanan, Martin Head-Gordon, and Christopher A. White. Curvy steps for density matrix-based energy minimization: Application to large-scale self-consistent-field calculations. *J. Chem. Phys.*, 118(14):6144–6151, Apr 2003.

[160] Emanuel H. Rubensson, Elias Rudberg, and Paweł Sałek. Density matrix purification with rigorous error control. *J. Chem. Phys.*, 128(7):074106, Feb 2008.

[161] Marco Häser. Møller-Plesset (MP2) perturbation theory for large molecules. *Theor. Chim. Acta*, 87(1):147–173, Nov 1993.

[162] Gustavo E. Scuseria and Philippe Y. Ayala. Linear scaling coupled cluster and perturbation theories in the atomic orbital basis. *J. Chem. Phys.*, 111(18):8330–8343, Nov 1999.

[163] Jan Almlöf. Elimination of energy denominators in Møller—Plesset perturbation theory by a Laplace transform approach. *Chem. Phys. Lett.*, 181(4):319–320, Jun 1991.

[164] Philippe Y. Ayala and Gustavo E. Scuseria. Linear scaling second-order Moller–Plesset theory in the atomic orbital basis for large molecular systems. *J. Chem. Phys.*, 110(8):3660–3671, Feb 1999.

[165] Daniel S. Lambrecht, Bernd Doser, and Christian Ochsenfeld. Rigorous integral screening for electron correlation methods. *J. Chem. Phys.*, 123(18):184102, Nov 2005.

[166] Daniel S. Lambrecht and Christian Ochsenfeld. Multipole-based integral estimates for the rigorous description of distance dependence in two-electron integrals. *J. Chem. Phys.*, 123(18):184101, Nov 2005.

[167] Bernd Doser, Daniel S. Lambrecht, and Christian Ochsenfeld. Tighter multipole-based integral estimates and parallel implementation of linear-scaling AO–MP2 theory. *Phys. Chem. Chem. Phys.*, 10(23):3335–3344, Jun 2008.

[168] Jan Zienau, Lucien Clin, Bernd Doser, and Christian Ochsenfeld. Cholesky-decomposed densities in Laplace-based second-order Møller–Plesset perturbation theory. *J. Chem. Phys.*, 130(20):204112, May 2009.

[169] Arne Luenser, Henry F. Schurkus, and Christian Ochsenfeld. Vanishing-Overhead Linear-Scaling Random Phase Approximation by Cholesky Decomposition and an Attenuated Coulomb-Metric. *J. Chem. Theory Comput.*, 13(4):1647–1655, Apr 2017.

[170] Simon A. Maurer, Lucien Clin, and Christian Ochsenfeld. Cholesky-decomposed density MP2 with density fitting: Accurate MP2 and double-hybrid DFT energies for large systems. *J. Chem. Phys.*, 140(22):224112, Jun 2014.

[171] Michael Glasbrenner, Daniel Graf, and Christian Ochsenfeld. Efficient Reduced-Scaling Second-Order Møller–Plesset Perturbation Theory with Cholesky-Decomposed Densities and an Attenuated Coulomb Metric. *J. Chem. Theory Comput.*, 16(11):6856–6868, Nov 2020.

[172] Peter Pulay. Localizability of dynamic electron correlation. *Chem. Phys. Lett.*, 100(2):151–154, Sep 1983.

[173] Peter Pulay and Svein Saebø. Orbital-invariant formulation and second-order gradient evaluation in Møller-Plesset perturbation theory. *Theor. Chim. Acta*, 69(5):357–368, Jun 1986.

[174] Svein Saebo/ and Peter Pulay. Fourth-order Mo/ller–Plessett perturbation theory in the local correlation treatment. I. Method. *J. Chem. Phys.*, 86(2):914–922, Jan 1987.

[175] Svein Saebo and Peter Pulay. The local correlation treatment. II. Implementation and tests. *J. Chem. Phys.*, 88(3):1884–1890, Feb 1988.

[176] Peter Pinski and Frank Neese. Communication: Exact analytical derivatives for the domain-based local pair natural orbital MP2 method (DLPNO-MP2). *J. Chem. Phys.*, 148(3):031101, Jan 2018.

[177] Danylo Kats, Denis Usvyat, and Martin Schütz. On the use of the Laplace transform in local correlation methods. *Phys. Chem. Chem. Phys.*, 10(23):3430–3439, Jun 2008.

[178] Egil A. Hylleraas. Neue Berechnung der Energie des Heliums im Grundzustande, sowie des tiefsten Terms von Ortho-Helium. *Z. Phys.*, 54(5-6):347–366, May 1929.

[179] Georg Hetzer, Peter Pulay, and Hans-Joachim Werner. Multipole approximation of distant pair energies in local MP2 calculations. *Chem. Phys. Lett.*, 290(1):143–149, Jun 1998.

[180] Guntram Rauhut, James W. Boughton, and Peter Pulay. Modeling localized electron pair correlation energies. *J. Chem. Phys.*, 103(13):5662–5673, Oct 1995.

[181] Peter Pinski, Christoph Riplinger, Edward F. Valeev, and Frank Neese. Sparse maps—A systematic infrastructure for reduced-scaling electronic structure methods. I. An efficient and simple linear scaling local MP2 method that uses an intermediate basis of pair natural orbitals. *J. Chem. Phys.*, 143(3):034108, Jul 2015.

[182] Hans-Joachim Werner, Gerald Knizia, Christine Krause, Max Schwilk, and Mark Dornbach. Scalable Electron Correlation Methods I.: PNO-LMP2 with Linear Scaling in the Molecular Size and Near-Inverse-Linear Scaling in the Number of Processors. *J. Chem. Theory Comput.*, 11(2):484–507, Feb 2015.

[183] Marius S. Frank, Gunnar Schmitz, and Christof Hättig. The PNO–MP2 gradient and its application to molecular geometry optimisations. *Mol. Phys.*, 115(3):343–356, Feb 2017.

[184] Martin Schütz. Low-order scaling local electron correlation methods. V. Connected triples beyond (T): Linear scaling local CCSDT-1b. *J. Chem. Phys.*, 116(20):8772–8785, May 2002.

[185] Martin Schütz and Hans-Joachim Werner. Local perturbative triples correction (T) with linear cost scaling. *Chem. Phys. Lett.*, 318(4):370–378, Feb 2000.

[186] Martin Schütz. Low-order scaling local electron correlation methods. III. Linear scaling local perturbative triples correction (T). *J. Chem. Phys.*, 113(22):9986–10001, Dec 2000.

[187] Péter R. Nagy, Gyula Samu, and Mihály Kállay. Optimization of the Linear-Scaling Local Natural Orbital CCSD(T) Method: Improved Algorithm and Benchmark Applications. *J. Chem. Theory Comput.*, 14(8):4193–4215, Aug 2018.

[188] Zoltán Rolik, Lóránt Szegedy, István Ladjánszki, Bence Ladóczki, and Mihály Kállay. An efficient linear-scaling CCSD(T) method based on local natural orbitals. *J. Chem. Phys.*, 139(9):094105, Sep 2013.

[189] Yang Guo, Christoph Riplinger, Ute Becker, Dimitrios G. Liakos, Yury Minenkov, Luigi Cavallo, and Frank Neese. Communication: An improved linear scaling perturbative triples correction for the domain based local pair-natural orbital based singles and doubles coupled cluster method [DLPNO-CCSD(T)]. *J. Chem. Phys.*, 148(1):011101, Jan 2018.

[190] Gunnar Schmitz and Christof Hättig. Accuracy of Explicitly Correlated Local PNO-CCSD(T). *J. Chem. Theory Comput.*, 13(6):2623–2633, Jun 2017.

[191] Filippo Sacchetta and Christian Ochsenfeld. 2021.

[192] Tatiana Korona and Hans-Joachim Werner. Local treatment of electron excitations in the EOM-CCSD method. *J. Chem. Phys.*, 118(7):3006–3019, Feb 2003.

[193] T. Daniel Crawford and Rollin A. King. Locally correlated equation-of-motion coupled cluster theory for the excited states of large molecules. *Chem. Phys. Lett.*, 366(5):611–622, Dec 2002.

[194] Danylo Kats and Martin Schütz. A multistate local coupled cluster CC2 response method based on the Laplace transform. *J. Chem. Phys.*, 131(12):124117, Sep 2009.

[195] Dávid Mester, Péter R. Nagy, and Mihály Kállay. Reduced-cost linear-response CC2 method based on natural orbitals and natural auxiliary functions. *J. Chem. Phys.*, 146(19):194102, May 2017.

[196] Arie Landau, Kirill Khistyaev, Stanislav Dolgikh, and Anna I. Krylov. Frozen natural orbitals for ionized states within equation-of-motion coupled-cluster formalism. *J. Chem. Phys.*, 132(1):014109, Jan 2010.

[197] Ashutosh Kumar and T. Daniel Crawford. Frozen Virtual Natural Orbitals for Coupled-Cluster Linear-Response Theory. *J. Phys. Chem. A*, 121(3):708–716, Jan 2017.

[198] Dávid Mester, Péter R. Nagy, and Mihály Kállay. Reduced-cost second-order algebraic-diagrammatic construction method for excitation energies and transition moments. *J. Chem. Phys.*, 148(9):094111, Mar 2018.

[199] Benjamin Helmich and Christof Hättig. Local pair natural orbitals for excited states. *J. Chem. Phys.*, 135(21):214106, Dec 2011.

[200] Benjamin Helmich and Christof Hättig. A pair natural orbital implementation of the coupled cluster model CC2 for excitation energies. *J. Chem. Phys.*, 139(8):084114, Aug 2013.

[201] Benjamin Helmich and Christof Hättig. A pair natural orbital based implementation of ADC(2)-x: Perspectives and challenges for response methods for singly and doubly excited states in large molecules. *Comput. Theor. Chem.*, 1040-1041:35–44, Jul 2014.

[202] Marius S. Frank and Christof Hättig. A pair natural orbital based implementation of CCSD excitation energies within the framework of linear response theory. *J. Chem. Phys.*, 148(13):134102, Apr 2018.

[203] T. Daniel Crawford, Ashutosh Kumar, Alexandre P. Bazanté, and Roberto Di Remigio. Reduced-scaling coupled cluster response theory: Challenges and opportunities. *WIREs Comput. Mol. Sci.*, 9(4):e1406, Jul 2019.

[204] Achintya Kumar Dutta, Frank Neese, and Róbert Izsák. Towards a pair natural orbital coupled cluster method for excited states. *J. Chem. Phys.*, 145(3):034102, Jul 2016.

[205] Pablo Baudin and Kasper Kristensen. LoFEx — A local framework for calculating excitation energies: Illustrations using RI-CC2 linear response theory. *J. Chem. Phys.*, 144(22):224106, Jun 2016.

[206] Michael Wormit. *Development and Application of Reliable Methods for the Calculation of Excited States: From Light-Harvesting Complexes to Medium-Sized Molecules.* PhD thesis, Johann Wolfgang Goethe–Universitaet Frankfurt, 2009.

[207] Yihan Shao, Laszlo Fusti Molnar, Yousung Jung, Jörg Kussmann, Christian Ochsenfeld, Shawn T. Brown, Andrew T. B. Gilbert, Lyudmila V. Slipchenko, Sergey V. Levchenko, Darragh P. O'Neill, A. DiStasio Robert, Jr., Rohini C. Lochan, Tao Wang, Gregory J. O. Beran, Nicholas A. Besley, John M. Herbert, Ching Yeh Lin, Troy Van Voorhis, Siu Hung Chien, Alex Sodt, Ryan P. Steele, Vitaly A. Rassolov, Paul E. Maslen, Prakashan P. Korambath, Ross D. Adamson, Brian Austin, Jon Baker, Edward F. C. Byrd, Holger Dachsel, Robert J. Doerksen, Andreas Dreuw, Barry D. Dunietz, Anthony D. Dutoi, Thomas R. Furlani, Steven R. Gwaltney, Andreas Heyden, So Hirata,

Chao-Ping Hsu, Gary Kedziora, Rustam Z. Khalliulin, Phil Klunzinger, Aaron M. Lee, Michael S. Lee, WanZhen Liang, Itay Lotan, Nikhil Nair, Baron Peters, Emil I. Proynov, Piotr A. Pieniazek, Young Min Rhee, Jim Ritchie, Edina Rosta, C. David Sherrill, Andrew C. Simmonett, Joseph E. Subotnik, H. Lee Woodcock Iii, Weimin Zhang, Alexis T. Bell, Arup K. Chakraborty, Daniel M. Chipman, Frerich J. Keil, Arieh Warshel, Warren J. Hehre, Henry F. Schaefer Iii, Jing Kong, Anna I. Krylov, Peter M. W. Gill, and Martin Head-Gordon. Advances in methods and algorithms in a modern quantum chemistry program package. *Phys. Chem. Chem. Phys.*, 8(27):3172–3191, Jul 2006.

[208] Evgeny Epifanovsky, Michael Wormit, Tomasz Kuś, Arie Landau, Dmitry Zuev, Kirill Khistyaev, Prashant Manohar, Ilya Kaliman, Andreas Dreuw, and Anna I. Krylov. New implementation of high-level correlated methods using a general block tensor library for high-performance electronic structure calculations. *J. Comput. Chem.*, 34(26):2293–2309, Oct 2013.

[209] Danylo Kats, Tatiana Korona, and Martin Schütz. Transition strengths and first-order properties of excited states from local coupled cluster CC2 response theory with density fitting. *J. Chem. Phys.*, 127(6):064107, Aug 2007.

[210] LAPACK — Linear Algebra PACKage, Aug 2021. [Online; accessed 28. Aug. 2021].

[211] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[212] PBLAS, Mar 1995. [Online; accessed 28. Aug. 2021].

[213] Lynn Elliot Cannon. A cellular computer to implement the Kalman filter algorithm. *Montana State University - Bozeman, College of Engineering*, pages 1–229, 1969.

[214] ScaLAPACK — Scalable Linear Algebra PACKage, Apr 2021. [Online; accessed 28. Aug. 2021].

[215] Jürg Hutter, Marcella Iannuzzi, Florian Schiffmann, and Joost VandeVondele. cp2k: atomistic simulations of condensed matter systems. *WIREs Comput. Mol. Sci.*, 4(1):15–25, Jan 2014.

[216] Ole Schütt, Peter Messmer, Jürg Hutter, and Joost VandeVondele. GPU-Accelerated Sparse Matrix–Matrix Multiplication for Linear Scaling Density Functional Theory. In *Electronic Structure Calculations on Graphics Processing Units*, pages 173–190. John Wiley & Sons, Ltd, Chichester, England, UK, Mar 2016.

[217] Ilia Sivkov, Alfio Lazzaro, and Juerg Hutter. DBCSR: A Library for Dense Matrix Multiplications on Distributed GPU-Accelerated Systems. *arXiv*, Oct 2019.

[218] Maximilien Ambroise. Designing a modern c++/fortran interface by example. Presented at FortranCon 2020, University of Zurich, Switzerland, 2020.

[219] Péter Pulay. Convergence acceleration of iterative sequences. the case of scf iteration. *Chem. Phys. Lett.*, 73(2):393–398, Jul 1980.

[220] Ernest R. Davidson. The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices. *J. Comput. Phys.*, 17(1):87–94, Jan 1975.

[221] Sonia Coriani, Thomas Fransson, Ove Christiansen, and Patrick Norman. Asymmetric-Lanczos-Chain-Driven Implementation of Electronic Resonance Convergent Coupled-Cluster Linear Response Theory. *J. Chem. Theory Comput.*, 8(5):1616–1628, May 2012.

[222] Jeppe Olsen, Poul Jørgensen, and Jack Simons. Passing the one-billion limit in full configuration-interaction (FCI) calculations. *Chem. Phys. Lett.*, 169(6):463–472, Jun 1990.

[223] Ronald B. Morgan. Davidson's method and preconditioning for generalized eigenvalue problems. *J. Comput. Phys.*, 89(1):241–245, Jul 1990.

[224] C. B. Moler, I. Shavitt, and Lawrence Berkeley Laboratory. National Resource for Computation In Chemistry. *Report on the Workshop Numerical Algorithms in Chemistry: Algebraic Methods: August 9-11, 1978*. Lawrence Berkeley Laboratory, University of California, Berkeley, CA, USA, 1978.

[225] W. Butscher and W. E. Kammer. Modification of Davidson's method for the calculation of eigenvalues and eigenvectors of large real-symmetric matrices: "root homing procedure". *J. Comput. Phys.*, 20(3):313–325, Mar 1976.

[226] Jianwei Xiao and Ming Gu. Spectrum-Revealing Cholesky Factorization for Kernel Methods. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 1293–1298. IEEE, Dec 2016.

[227] Jianwei Xiao, Ming Gu, and Julien Langou. Fast Parallel Randomized QR with Column Pivoting Algorithms for Reliable Low-Rank Matrix Approximations. In *2017 IEEE 24th International Conference on High Performance Computing (HiPC)*, pages 233–242. IEEE, Dec 2017.