Studies of quantum dots

Ab initio coupled-cluster analysis using OpenCL and GPU programming

by

Christoffer Hirth

THESIS
for the degree of
MASTER OF SCIENCE

(Master in Computational Physics)

Faculty of Mathematics and Natural Sciences
Department of Physics
University of Oslo

June 2012

Preface

The completion of this thesis marks the end of a story that begun more than 15 years ago. As a small boy, barely started at elementary school, I pondered everything that crossed my mind. Standing outside our cottage in the winter watching the stars, and occasionally also the northern light, I was stunned by the size and the beauty of the universe we live in. Later, as I watched wood burn in the fireplace, I enjoyed the stories told me about how the sun's warmth was stored inside the logs, to be released in winter by lighting a match.

The decision of going in the direction of natural sciences came early, initially made by myself, but it would not have been realized if I had not gotten help from people I met on my way. Whenever I had questions my dad helped feed my curiosity, and whenever my curiosity let me down my mother gave me the care and comfort I needed.

Throughout years of education I have had both positive and negative experiences. The best experience was meeting Morten Hjorth-Jensen as the teacher in computational physics. I know no other professor with the same enthusiasm and excellence, both academic and pedagogical, and there is no coincidence why I returned half a year later to discuss topics for a Master's Thesis with him. Having an office available during the Master I spent more time at the university, and I appreciate the collegues who sat close by. I would like to mention Frank Olsen (we spent some time discussing different topics regarding our similar theses), Jørgen Høgberget (your fresh grind coffee lights up the day), and Sarah Reimann (learning me German cannot have been easy, good thing we focused mainly on one word).

The rest of my family also deserves a few words here. My sisters, who have the courage to always be there for others, even in times when others should have been there for them instead, and my brother, who is always positive, even when helping me proof read this thesis. At last, but not at all the least, I appreciate how my girlfriend, soon my wife, is still there, despite all the days and nights I have spent in front of my computer. It is no exaggeration to say that I am proud of my family.

To all of you, and others who have helped me during all this time, I owe you my thanks. Hopefully have I lived up to your expectations, and I look forward to spending more time with you this summer, before I may resume my story or start a new one.

Christoffer Hirth

Oslo,
June, 2012

Contents

Chapter 1

Introduction

This is not the first time quantum dots have been studied using the coupled-cluster machinery. In fact, some of the calculations done here have also been done previously. Quantum dots, that is, strongly confined electrons show a variety of interesting properties. Of relevance in both experiments and various technical components, is the possibility to fine tune their electrical and optical properties.

trengs bedre overgang A coupled-cluster approach aims to find the ground-state energy by distributing the particles in a number of basis states. In practice working in a truncated basis, coupled cluster also defines a truncation in the possible ways to distribute the particles.

Earlier master's projects have solved the problem, through serial C++ implementations [1, 2, 3], for up to 20 electrons within 420 basis states. Programs met convergence problems for increasing number of particles and reduced potential strengths. An extension was thus natural in the direction of exploring a parallel approach to increase system sizes and at the same time try to overcome the convergence issues.

The first part of this thesis, chapter 2 and 3, serves as a theoretical introduction to the basic theory of quantum mechanics and many-body theory. Important features and terminology are discussed, with a focus on theoretical topics that are required for the understanding of later parts.

Continuing we discuss different systems, quantum dots in particular, in chapter 4. Various examples on how systems are implemented by sub-classing the 'System' base class are given, and we shed some light on how a system can be optimized, both in terms of memory and processing requirements, without compromising the flexibility and generalness of the code.

In chapter 5 the coupled-cluster method is introduced. Beginning with a more shallow outline of the method, we will eventually derive the full set of non-linear equations. In addition to explaining how to implement coupled cluster, the Hartree-Fock method is also briefly mentioned.

The last theoretical chapter is chapter 6, leaning more in a programming technical direction. OpenCL, a standard and a library for accelerated code, is introduced prior to its application, namely accelerated matrix multiplication. **tenk på bedre overgang her også** Matrix multiplication is decoupled from the main program, put in its own class, to make it easier to employ different algorithms without changing the entire code base.

Results along with a discussion and a final conclusion close this thesis. We present results for up to 56 particles in more than 900 basis functions, and for previously unobtainable weak strengths of the confining single-particle potential. Results and time usage are benchmarked, both against other coupled-cluster programs and other methods. Final remarks are made for potential extensions to this project.

Part I

Theory

Chapter 2

Quantum mechanics

During a period spanning more that 50 years, from the end of the 19th century till the late twenties in the previous century, several discoveries were made that could not be properly explained by the by then available theoretical approaches based on for example Newtonian mechanics. Phenomena like the photoelectric effect, blackbody radiation, Compton scattering, X-rays etc. were all processes which required a finer resolution of scales, leading eventually to the theory of quantum mechanics and its postulates, with Schrödinger's equation being the new mathematical framework to express the laws of motion at nano or smaller scales.

2.1 Fundamentals

Objects in classical mechanics have well-defined positions, which can be tracked over a time interval by Newton's laws of motion. Once we know the initial state and all forces present, we can predict the motion of objects until the end of time. It is of course not doable in practice, because we cannot know *all variables*, and certainly not to *endless accuracy*. We say that Newtonian mechanics is a deterministic theory. Quantum mechanics, with its postulates like its matter-wave duality and Heisenberg's uncertainty principle, introduces an approach to describe Nature that represents a probabilistic determinism. In this chapter we discuss some of the basic tools and postulates needed to describe physical systems governed by Schrödinger's equation.

2.1.1 The wave function

We begin with only one electron floating in empty space. To describe this electron we would construct a so-called wave function, typically written as $\Psi(\vec{r}, t)$. If the particle is influenced by some external potential energy V, we could find the time evolution by solving the Schrödinger equation,

$$i\hbar \frac{\partial \Psi}{\partial t} = -\frac{\hbar^2}{2m} \nabla^2 \Psi + V\Psi. \tag{2.1}$$

It is common to simplify this by defining the Hamiltonian operator,

$$\hat{H} = \hat{T} + \hat{V}, \tag{2.2}$$

where $\hat{T}$ is the operator of kinetic energy,

$$\hat{T} = \frac{\hat{p}^2}{2m} = -\frac{\hbar^2}{2m}\nabla^2, \tag{2.3}$$

and $\hat{V}$ is the potential. Our Hamiltonian thus represents the total energy for this particle. It is now possible to rewrite the full Schrödinger equation as

$$i\hbar\frac{\partial\Psi}{\partial t} = \hat{H}\Psi. \tag{2.4}$$

The Schrödinger equation serves as an analog to Newton's laws. If the initial conditions for Ψ and the exact potential $\hat{V}$ are known, we could calculate the wave function for any time later. All solutions of the Schrödinger equation reside in what is known as the Hilbert space.

A physicist's first attempt at solving partial differential equations is to see whether the technique of separation of variables can be applied or not. We assume that the potential $\hat{V}$ is time independent and the solutions consist of a time-independent factor ψ, and another factor depending only on time, τ. If multiple such products are solutions, then it is clear that any linear combination of these products is a solution too. Thus,

$$\Psi(\vec{r},t) = \sum_n c_n\psi_n(\vec{r})\tau_n(t). \tag{2.5}$$

After inserting one term from (2.5) into (2.1), we separate the two factors to appear on differing sides,

$$\frac{i\hbar}{\tau_n(t)}\frac{\partial\tau_n(t)}{\partial t} = E_n = -\frac{\hbar^2}{\psi_n(\vec{r})2m}\nabla^2\psi_n(\vec{r}) + \hat{V}. \tag{2.6}$$

Here, E_n is a constant of separation, and we should note that this may fail if our Hamiltonian has some explicit time dependency. We can solve this for τ_n, yielding

$$\frac{d\tau_n(t)}{dt} = \frac{E_n}{i\hbar}\tau_n(t) \Rightarrow \tau_n(t) = e^{-\frac{i}{\hbar}E_n}. \tag{2.7}$$

It is necessary to solve the time-independent Schrödinger equation in order to find ψ_n,

$$-\frac{\hbar^2}{2m}\nabla^2\psi_n(\vec{r}) + \hat{V}\psi_n(\vec{r}) = E_n\psi_n(\vec{r}) \Rightarrow \hat{H}\psi_n(\vec{r}) = E_n\psi_n(\vec{r}). \tag{2.8}$$

The Hamiltonian operator, $\hat{H}$, represents the energy of the wave function it is acting upon. It is thus clear why the constant of separation was called E_n.

With the knowledge on how to find the wave function at any time for a specific potential, it is still not obvious what the interpretation of a wave function is, yet more unclear how one can extract observable quantities from this construct.

The wave function is a complex function, describing the spatial distribution of a particle. Born's statistical interpretation states that the probability of finding a particle in a region Ω at a time t, is

$$\int_\Omega \Psi^*(\vec{r},t)\Psi(\vec{r},t)d^3\vec{r}. \tag{2.9}$$

In order for this to be correct it is customary to work with normalized wave functions, that is, scaling Ψ with a complex constant[1] to enforce that

$$\int_{-\infty}^{\infty} |\Psi(\vec{r},t)|^2 \, d^3\vec{r} = 1. \tag{2.10}$$

Loosely speaking this enforces that if the particle exists it has to be somewhere, and the probability to find it if we look everywhere has to be 1. The concept of not knowing where the particle is leads to many fundamental questions, both physical and philosophical. As most of these questions lead to an endless discussion with no clear answer (yet), we will not try to answer them here.

In addition to the spatial distribution varying in time, particles have an intrinsic property, possessing a magnetic dipole moment, known as spin. Despite acting similarly to a charged rotating body in classical electrodynamics, elementary particles have no known inner structure, making it a different phenomenon. Spin is quantized, as many other properties in quantum mechanics, and should be accounted for in the wave function by multiplication of a spin part, χ. Electrons, which form the key focus in this study, have a spin quantum number $s = \frac{1}{2}$ that can be projected in two directions, $\pm\frac{1}{2}$, often referred to as up and down. In fact all spatial wave functions in this chapter, ψ, should then have a total *spin-orbital*

$$\psi\chi_{\downarrow} \text{ or } \psi\chi_{\uparrow}. \tag{2.11}$$

Degeneracies, multiple particles and spin dependent Hamiltonians can complicate our theory somewhat, but neither of these effects are encountered in this chapter.

2.1.2 Observables

In quantum mechanics observables are represented by operators. As all measurements must have a real value, all such operators need to return real expectation values. The expectation value of an observable with an operator $\hat{O}$ is calculated as[2]

$$\langle \hat{O} \rangle = \int \Psi^* \hat{O} \Psi dx. \tag{2.12}$$

With the expectation value being real; $\langle \hat{O} \rangle = \langle \hat{O} \rangle^*$, and therefore also

$$\int \Psi^* \hat{O} \Psi dx = \left(\int \Psi^* \hat{O} \Psi dx \right)^* = \int \left(\hat{O} \Psi \right)^* \Psi dx. \tag{2.13}$$

All operators for observables will need to possess this property when acting on any wave function, Ψ, within the Hilbert space, referred to as Hermitian or self-adjoint operators. There are two fundamental examples of operators representing physical observables, position and momentum. The position has the simplest correspondence,

$$\hat{x} = x, \tag{2.14}$$

[1]Another possibility is to work with unnormalized wave functions, and always divide these type of integrals by $\int_{-\infty}^{\infty} |\Psi(\vec{r},t)|^2 \, d^3\vec{r}$.

[2]Here, and when convenient from now on, we will skip the integration limits and stick to dx to denote an integral over all space spanned by all variables of freedom.

whereas momentum is represented as

$$\hat{p} = -i\hbar\nabla. \tag{2.15}$$

Other quantities can be derived from the position and momentum operators using a reasoning close to classical quantities, e.g. kinetic energy,

$$\hat{T} = \frac{1}{2}m\hat{v}^2 = \frac{\hat{p}^2}{2m} = -\frac{\hbar^2}{2m}\nabla^2. \tag{2.16}$$

2.1.3 The canonical commutation relation

Similar to matrices in linear algebra, not all operators commute. Having two operators $\hat{A}$ and $\hat{B}$, the order of which they are applied may affect the result, thus in general

$$\hat{A}\hat{B} \neq \hat{B}\hat{A}. \tag{2.17}$$

One typically defines the commutator

$$[\hat{A}, \hat{B}] = \hat{A}\hat{B} - \hat{B}\hat{A}, \tag{2.18}$$

having the properties listed below:

a. Switching order between the two operators changes the sign,

$$[\hat{A}, \hat{B}] = -[\hat{B}, \hat{A}]. \tag{2.19}$$

b. All constants can safely be placed in front of the commutator,

$$[c_a\hat{A}, c_b\hat{B}] = c_a c_b[\hat{A}, \hat{B}]. \tag{2.20}$$

c. Summation of two operators inside one commutator can be carried out as a sum of two commutators,

$$[\hat{A}, \hat{B} + \hat{C}] = [\hat{A}, \hat{B}] + [\hat{A}, \hat{C}]. \tag{2.21}$$

These properties follow directly from the definition (2.18). More properties can be derived, but we restrict us to the properties of interest later in this thesis.

Even the operators for position and momentum do not commute in quantum mechanics, a fact that can be shown in a few steps. To make the derivation conceptually easier we let the operators act on an arbitrary test function Ψ_T, and concentrate on one dimension, i.e.

$$[\hat{x}, \hat{p}]\Psi_T = -i\hbar[\hat{x}, \frac{\partial}{\partial x}]\Psi_T = -i\hbar\left(\hat{x}\frac{\partial}{\partial x}\Psi_T - \frac{\partial}{\partial x}(\hat{x}\Psi_T)\right). \tag{2.22}$$

Applying the product rule on the last term we end up with

$$-i\hbar\left(\hat{x}\frac{\partial}{\partial x}\Psi_T - \Psi_T - \hat{x}\frac{\partial}{\partial x}\Psi_T\right) = i\hbar\Psi_T. \tag{2.23}$$

Dropping the test function, we have proved what is known as the canonical commutation relation,

$$[\hat{x}, \hat{p}] = i\hbar. \tag{2.24}$$

2.1.4 Eigenfunctions

Letting an observable $\hat{O}$ act on a state Ψ, one may get different results, each with its own probability. This *spectra* of results can be either continuous or discrete, and it is of interest to know if a state yielding the same value each and every time for an operator can be found. In fact such states exist, and one is already encountered in the time-independent Schrödinger equation, (2.8), where a state ψ_n will return the energy E_n when acted upon by $\hat{H}$. Such states are called eigenstates, having a corresponding eigenvalue. For any observable, its eigenstates are found by the eigenvalue equation,

$$\hat{O}\psi_n = O_n\psi_n. \tag{2.25}$$

A few properties for such functions can be proven:

- All eigenvalues of a Hermitian operator are real.

  Using the property of Hermitian operators from eq. (2.13), we have

  $$\int \psi_n^* \hat{O}\psi_n dx = \int \left(\hat{O}\psi_n\right)^* \psi_n dx \Rightarrow O_n \int \psi_n^* \psi_n dx = O_n^* \int \psi_n^* \psi_n dx, \tag{2.26}$$

  which means $O_n = O_n^*$, and thus real.

- Different eigenfunctions are orthogonal.

  Another form of the condition for Hermitian operators is

  $$\int \psi_m^* \hat{O}\psi_n dx = \int \left(\hat{O}\psi_m\right)^* \psi_n dx. \tag{2.27}$$

  Despite looking like a stronger condition it is in fact equivalent with eq. (2.13) [4], resulting in

  $$O_n \int \psi_m^* \psi_n dx = O_m^* \int \psi_m^* \psi_n dx. \tag{2.28}$$

  Since it is already known that the eigenvalues are real there is no other possibility than $\int \psi_m^* \psi_n dx = 0$ whenever $O_n \neq O_m$.

- For any operator with a finite set of eigenfunctions, the eigenfunctions are complete. They span the Hilbert space, such that any function in this space can be expressed as a linear combination of eigenfunctions. It can, in fact, not be proven in general for spectra with infinite number of eigenstates. Nonetheless, it is taken as a necessity, and thus a restriction on the observable operators.

2.1.5 Bra-ket notation

In daily work the wave functions encountered are seldom written as explicit functions. One typically refer to states instead, hiding the complexity of dealing with functions and integrals, into constructs called 'bra' and 'ket'. Dirac introduced this notation in 1930, and named it after splitting the word 'bracket' [5].

A ket state is the right-hand part, where a state Ψ would be represented as $|\Psi\rangle$. This represents a particle, with a corresponding wave function. The ket state can be viewed as a column vector,

$$|\Psi\rangle = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ ... \end{bmatrix} \text{ or } |\Psi\rangle = \begin{bmatrix} \Psi(x_0) \\ \Psi(x_1) \\ \Psi(x_2) \\ ... \end{bmatrix}, \tag{2.29}$$

where the first example has Ψ in a state with a given basis for a *finite* Hilbert space. The other example has the ket state represented in an *infinite* Hilbert space, as there are infinitely many positions x_i.

The bra state is the left-hand part of the bracket, referred to as the ket's dual, being the Hermitian transposed[3] of corresponding ket. Following the first example in eq. (2.29) the bra state would be

$$\langle\Psi| = \left[c_0^*, c_1^*, c_2^*, ...\right]. \tag{2.30}$$

The expectation values of operators can then be viewed as matrices. This is just a picture, as most of the vectors will be in infinite dimensional spaces explained by functions. The notation will, however, give us a linear algebra like syntax, a formalism that ease our daily work. Operators will in this syntax work in the same way as for wave functions. The dual part of an operator acting from the left on a ket state, will be the operator's Hermitian adjoint acting from the right on a bra state,

$$\hat{O}|\Psi\rangle \longleftrightarrow \langle\Psi|\hat{O}^\dagger. \tag{2.31}$$

Having introduced these brackets, it is time to define a particularly useful operation, the inner product. The definition is straight forward in a linear-algebra sense, except that we need to extended it to an infinite space by the integral

$$\langle\Psi_i|\Psi_j\rangle = \int \Psi_i^*\Psi_j dx. \tag{2.32}$$

Operators can be placed in between the bra and the ket states, reducing the expectation value from (2.12) to

$$\langle\hat{O}\rangle = \langle\Psi|\hat{O}|\Psi\rangle. \tag{2.33}$$

The fact that observable operators are Hermitian can now be simplified from (2.13) to

$$\langle\Psi|\hat{O}|\Psi\rangle = \langle\Psi|\hat{O}^\dagger|\Psi\rangle, \tag{2.34}$$

where $\hat{O} = \hat{O}^\dagger$ is referred to as self-adjoint.

2.1.6 A fundamental summary

The new syntax of bra-ket notation allows us to summarize quantum mechanics into a few neatly expressed postulates. Although these postulates are presented in a slightly different manner by different authors, the main concepts can be put into four postulates:

[3]Hermitian transposed means the transposed vector, where the complex conjugate is performed on each element.

Postulate 1: The state of an isolated physical system can be described by a state-vector, $|\Psi\rangle$, within a Hilbert space, $\mathcal{H}$.

Postulate 2: Every physical observable, O, has a corresponding linear Hermitian operator, $\hat{O}$, acting on vectors in $\mathcal{H}$.

Postulate 3: A measurement of the quantity O, with a corresponding operator $\hat{O}$, is guaranteed to yield one of the operator's eigenvalues, O_n, with a certain probability.

Postulate 4: The state-vector has a time evolution satisfying the Schrödinger equation,

$$i\hbar\frac{\partial}{\partial t}|\Psi(t)\rangle = \hat{H}|\Psi(t)\rangle. \tag{2.35}$$

2.2 Harmonic oscillator

In this section we will calculate, using the fundamental theory from previous section, the energy spectra along with its eigenstates for a system that is simple, but of high interest. The problem to solve in this thesis is the harmonic oscillator in two dimensions, but to begin with we will only treat the one dimensional problem, ending up with solutions for two dimensions at no extra cost.

A classical harmonic oscillator consists of a particle attached to a spring. The farther the particle moves, the larger the force from the spring, according to Hooke's law,

$$F = -kx. \tag{2.36}$$

A potential field V is simply the negative of the work done, and using this relation, we can calculate the potential energy as a simple integral,

$$V = -\int_0^x -kx dx = \frac{1}{2}kx^2 = \frac{1}{2}m\omega^2x^2, \tag{2.37}$$

where $\omega = \sqrt{k/m}$ is called the frequency. This can be inserted into the Hamiltonian by replacing x with its quantum mechanical operator, found in eq. (2.14). The total Hamiltonian reads $-\frac{\hbar^2}{2m}\frac{d^2}{dx^2} + \frac{1}{2}m\omega^2x^2$, leading to the time-independent Schrödinger equation

$$-\frac{\hbar^2}{2m}\frac{d^2}{dx^2}|\psi_n\rangle + \frac{1}{2}m\omega^2x^2|\psi_n\rangle = E_n|\psi_n\rangle. \tag{2.38}$$

It may be tempting to attack this problem in a brute force manner. That would however prove quite tedious, and better strategies exist.

2.2.1 The ladder operators

Following an, at first, unexpected path, we will introduce what is known as the ladder, or excitation, operator, defined by

$$\hat{a}^\dagger = \sqrt{\frac{m\omega}{2\hbar}}\left(\hat{x} - \frac{i\hat{p}}{m\omega}\right), \tag{2.39}$$

and its Hermitian adjoint, the de-excitation operator[4],

$$\hat{a} = \sqrt{\frac{m\omega}{2\hbar}}\left(\hat{x} + \frac{i\hat{p}}{m\omega}\right). \tag{2.40}$$

The motivation for selecting these two operators may seem unclear, but their product is of interest,

$$\hat{a}\hat{a}^\dagger = \frac{m\omega}{2\hbar}\left(\hat{x}^2 + \frac{i}{m\omega}[\hat{p},\hat{x}] + \frac{\hat{p}^2}{m^2\omega^2}\right) = \frac{m\omega}{2\hbar}\hat{x}^2 + \frac{\hat{p}^2}{2\hbar m\omega} + \frac{i}{2\hbar}[\hat{p},\hat{x}], \tag{2.41}$$

where the first two terms are found in the Hamiltonian, and the last term can be expressed by the canonical commutation relation (2.24),

$$\hat{a}\hat{a}^\dagger = \frac{1}{\hbar\omega}\hat{H} + \frac{1}{2} \Rightarrow \hat{H} = \hbar\omega\left(\hat{a}\hat{a}^\dagger - \frac{1}{2}\right). \tag{2.42}$$

Being able to rewrite our Hamiltonian, it is tempting to investigate these operators further. In particular, it is of interest to find their commutator. Using the rules of commutators shown in section 2.1.3 we find,

$$[\hat{a},\hat{a}^\dagger] = \frac{m\omega}{2\hbar}\left[\left(\hat{x} + \frac{i\hat{p}}{m\omega}\right),\left(\hat{x} - \frac{i\hat{p}}{m\omega}\right)\right] = \frac{m\omega}{2\hbar}[\hat{x},\hat{x}] + \frac{i}{\hbar}[\hat{p},\hat{x}] + \frac{1}{2\hbar m\omega}[\hat{p},\hat{p}], \tag{2.43}$$

where only the second term is nonzero,

$$[\hat{a},\hat{a}^\dagger] = \frac{i}{\hbar}[\hat{p},\hat{x}] = -\frac{i}{\hbar}i\hbar = 1. \tag{2.44}$$

Because of this, the Hamiltonian can equally well be written in one out of two forms,

$$\hat{H} = \hbar\omega\left(\hat{a}\hat{a}^\dagger - \frac{1}{2}\right) \text{ or } \hat{H} = \hbar\omega\left(\hat{a}^\dagger\hat{a} + \frac{1}{2}\right). \tag{2.45}$$

The crucial step comes when claiming that one, yet unknown, state, $|\psi_n\rangle$, is a solution of the time-independent Schrödinger equation, $\hat{H}|\psi_n\rangle = E_n|\psi_n\rangle$. With this in mind, one may ask what the energy of $\hat{a}^\dagger|\psi_n\rangle$ is,

$$\begin{aligned}\hat{H}\left(\hat{a}^\dagger|\psi_n\rangle\right) &= \hbar\omega\left(\hat{a}^\dagger\hat{a}\hat{a}^\dagger + \frac{1}{2}\hat{a}^\dagger\right)|\psi_n\rangle = \hbar\omega\hat{a}^\dagger\left(\hat{a}\hat{a}^\dagger + \frac{1}{2}\right)|\psi_n\rangle \\ &= \hbar\omega\hat{a}^\dagger\left(\hat{a}^\dagger\hat{a} + 1 + \frac{1}{2}\right)|\psi_n\rangle = \hat{a}^\dagger\left(\hat{H} + \hbar\omega\right)|\psi_n\rangle = (E_n + \hbar\omega)\,\hat{a}^\dagger|\psi_n\rangle.\end{aligned} \tag{2.46}$$

The last steps were achieved by exploiting the commutator between the excitation and de-excitation operator, and then recall that $|\psi_n\rangle$ is an eigenstate of $\hat{H}$. With the same approach one would also find that

$$\hat{H}\left(\hat{a}|\psi_n\rangle\right) = (E_n - \hbar\omega)\,\hat{a}|\psi_n\rangle. \tag{2.47}$$

[4]One can prove that the excitation and de-excitation operators are the Hermitian adjoint of each other. It will however serve no purpose to us at this stage.

With these operators we have the possibility to create states with energy at discrete steps of $\hbar\omega$, as long as we find at least one state to start out with. It seems reasonable that there should exist a lower limit, where applying $\hat{a}$ should give us no new state, viz.

$$\hat{a}|\psi_0\rangle = 0. \tag{2.48}$$

For convenience it is possible to label this state simply $|0\rangle$. By inserting the full expression for $\hat{a}$ and solving the differential equation,

$$\begin{aligned}\sqrt{\frac{m\omega}{2\hbar}}\left(\hat{x} + \frac{i}{m\omega}\left(-i\hbar\frac{\partial}{\partial x}\right)\right)|0\rangle = 0 \Rightarrow \int \frac{d|0\rangle}{|0\rangle} = -\frac{m\omega}{\hbar}\int x dx \\ \Rightarrow |0\rangle = Ce^{-\frac{m\omega}{2\hbar}x^2},\end{aligned} \tag{2.49}$$

we get an explicit expression for the lowest-lying state, up to a constant C to be determined by normalization.

The only remaining task is to find the energy of the ground state, E_0. Inserting expression (2.45) for the Hamiltonian into the time-independent equation, we find

$$\hbar\omega\left(\hat{a}^\dagger\hat{a} + \frac{1}{2}\right)|0\rangle = \hbar\omega\hat{a}^\dagger\hat{a}|0\rangle + \frac{1}{2}\hbar\omega|0\rangle = E_0|0\rangle. \tag{2.50}$$

But since $\hat{a}|0\rangle = 0$, it is clear that $E_0 = \frac{1}{2}\hbar\omega$, and using the fact that eigenstates exist at steps of $\hbar\omega$, the complete energy spectrum is

$$E_n = \left(n + \frac{1}{2}\right)\hbar\omega. \tag{2.51}$$

2.2.2 Two dimensions

So far, we have treated the oscillator problem in only one dimension. Moving to two dimensions, the actual Hamiltonian of interest changes to

$$\hat{H} = -\frac{\hbar^2}{2m}\nabla^2 + \frac{1}{2}m\omega^2 r^2 = -\frac{\hbar^2}{2m}\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) + \frac{1}{2}m\omega^2\left(x^2 + y^2\right). \tag{2.52}$$

Once again we will, as physicists, attack this by separation of variables, assuming that the new state $|n\rangle$ is a product of independent states in x and y,

$$|n\rangle = |n_x\rangle \otimes |n_y\rangle. \tag{2.53}$$

It is already clear that $\hat{H}$ can be written as a sum of two Hamiltonians, where each is only one dimensional,

$$\hat{H} = \hat{H}_x + \hat{H}_y = \left(-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + \frac{1}{2}m\omega^2 x^2\right) + \left(-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial y^2} + \frac{1}{2}m\omega^2 y^2\right). \tag{2.54}$$

The total time-independent Schrödinger equation now reads

$$\hat{H}|n\rangle = \left(\hat{H}_x|n_x\rangle\right) \otimes |n_y\rangle + |n_x\rangle \otimes \left(\hat{H}_y|n_y\rangle\right) = E_n\left(|n_x\rangle \otimes |n_y\rangle\right), \tag{2.55}$$

and since H_x has the eigenstates found by using the ladder operators, with energies $E_{n_x} = \left(n_x + \frac{1}{2}\right)\hbar\omega$, the total energy is

$$E_{n_x,n_y} = \left(n_x + \frac{1}{2}\right)\hbar\omega + \left(n_y + \frac{1}{2}\right)\hbar\omega = \left(n_x + n_y + 1\right)\hbar\omega. \tag{2.56}$$

Understanding the methods here, with the ladder operators, has a great value. When moving on to many-body methods, similar constructs, called creation and annihilation operators, will be used to simplify calculations. Whereas the ladder operators simply ‘moved’ the electron to a state with a different energy, the creation and annihilation operators will add or remove electrons from a system.

Chapter 3

Many-body theory

It is often insufficient to be able to calculate properties in systems with only one particle. One would for example be restricted to only hydrogen, if studying atoms. Methods for many-particle systems have thus been developed, often theoretically exact, but in practice we must rely on computer programs having a truncation affecting the accuracy of the results. Different types of approximations, or many-body methods, exist, where widely used techniques are; full configuration interaction, many-body perturbation theory, Hartree-Fock theory, Monte-Carlo methods and coupled-cluster theory. Even though we will focus mainly on the coupled-cluster approach, the concepts from this chapter are shared by several many-body methods based on wave functions constructed using a single-particle basis.

3.1 The non-interacting case

The natural starting point for many-body theory is to deal with non-interacting particles. In this case the Schrödinger equation holds the same form now as it did for one particle in eq. (2.35). Assuming a time-independent Hamiltonian,

$$\hat{H} = \sum_k \hat{h}_k = \sum_k \hat{t}_k + \sum_k \hat{v}_k, \tag{3.1}$$

with $\hat{t}_k$ and $\hat{v}_k$ being the operators for kinetic and potential energy for particle k, the energy is constant in time and we only need to solve the time-independent Schrödinger equation. Since the particles are not interacting, this equation is separable, and we assume a total wave function, $|\Psi^{(\lambda)}\rangle$, being a product of different single-particle spin orbitals $|\psi_k^{(\lambda)}\rangle$, with a total energy $E^{(\lambda)} = \sum_k E_k^{(\lambda)}$,

$$\hat{H}|\Psi^{(\lambda)}\rangle = \left(\sum_k \hat{h}_k\right)\left(|\psi_1^{(\lambda)}\rangle \otimes |\psi_2^{(\lambda)}\rangle \cdots \otimes |\psi_N^{(\lambda)}\rangle\right) = \sum_k E_k^{(\lambda)}|\Psi^{(\lambda)}\rangle. \tag{3.2}$$

Here $E_k^{(\lambda)}$ is the energy of particle k, satisfying the single-particle eigenvalue equation,

$$\hat{h}_k|\psi_k^{(\lambda)}\rangle = E_k^{(\lambda)}|\psi_k^{(\lambda)}\rangle. \tag{3.3}$$

It is important to note how the subscript refers to the different particles, whereas the superscript denotes different eigenstates inside a spectrum of energies.

Electrons are, although it complicates our calculations, interacting through the coulomb repulsion. For this reason we keep this simple separable calculation in memory when we move on to many-body theory which yields the correlation as a correction to the non-interacting reference energy.

3.2 Indistinguishable and identical particles

An important aspect when considering systems with more than one particle is that electrons are not only identical, they are in fact indistinguishable. In quantum mechanics it makes no sense talking about different particles, they are truly identical and impossible to track one at a time. As a consequence of this, interchanging the coordinates of two particles should not alter the probability distribution, i.e.

$$|\Psi|^2 = |\hat{P}_{ij}\Psi|^2. \tag{3.4}$$

This compact notation is due to the introduction of the permutation operator $\hat{P}_{ij}$, interchanging particle i and j. Equation (3.4) holds only if

$$\hat{P}_{ij} = \pm 1, \tag{3.5}$$

where particles with a symmetric wave function, that is $\hat{P}_{ij} = 1$, are called bosons, while particles having an antisymmetric wave function, $\hat{P}_{ij} = -1$, are called fermions.

Since electrons are fermions we need to construct wave functions that are antisymmetric. The simple product we assumed in the non-interacting case is insufficient. Antisymmetric wave functions are usually expressed as determinants, as proposed by John C. Slater, therefore called Slater determinants [6]. Having a complete, orthonormal, single-particle basis where N functions, $\phi_\alpha, \phi_\beta, \cdots \phi_\delta$, are occupied by N particles at different positions, $\vec{r_1}, \cdots, \vec{r_N}$, an N-particle wave function reads

$$\Phi_{\alpha,\beta,\cdots,\delta}(\vec{r_1}, \cdots, \vec{r_N}) = \frac{1}{\sqrt{N!}} \begin{vmatrix} \phi_\alpha(\vec{r_1}) & \phi_\beta(\vec{r_1}) & \cdots & \phi_\delta(\vec{r_1}) \\ \phi_\alpha(\vec{r_2}) & \phi_\beta(\vec{r_2}) & \cdots & \phi_\delta(\vec{r_2}) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_\alpha(\vec{r_N}) & \phi_\beta(\vec{r_N}) & \cdots & \phi_\delta(\vec{r_N}) \end{vmatrix}. \tag{3.6}$$

The notation here is of importance. Up to now we have looked at the exact solution, denoted Ψ. In this step the single particle basis ϕ can be any complete basis, and therefore Φ is in general not the exact solution. The solution can however be expressed as a combination of slater determinants,

$$|\Psi^{(\lambda)}\rangle = \sum_{\alpha,\beta,\cdots,\delta} C^{(\lambda)}_{\alpha,\beta,\cdots,\delta} |\Phi_{\alpha,\beta,\cdots,\delta}\rangle, \tag{3.7}$$

due to the completeness of our basis functions. Determinants have the property of being zero whenever two columns are equal. This is a manifestation of the exclusion principle formulated by Wolfgang Pauli in 1925 [7], two fermions cannot share the same state, which he later received the Nobel prize for in 1945 [?]. Including the two spin states available for each electron, no more than two electrons can share the same orbital.

To incorporate interactions between the electrons, we add an extra term, $\hat{v}_{kl}$, to $\hat{H}$,

$$\hat{H} = \sum_k \hat{t}_k + \sum_k \hat{v}_k + \frac{1}{2}\sum_{kl} \hat{v}_{kl}, \tag{3.8}$$

which is a two-body potential between electron k and l, and the factor of $\frac{1}{2}$ comes from the fact that all contributions are counted twice, assuming $\hat{v}_{kl} = \hat{v}_{lk}$. In the case of electron structures, these terms are simply all pairs of Coulomb interactions. It is possible to continue this, by adding three-, four-, up to N-body forces. Three-body forces are often needed in nuclear physics, but the two-body nature of the coulomb interaction limits our calculations to only two.

3.3 Second quantization

Limiting ourself to systems of electrons only, we recall the antisymmetric Slater determinant, eq. (3.6), and assuming orthonormal single-particle states,

$$\langle \phi_r | \phi_s \rangle = \delta_{rs}, \tag{3.9}$$

we fill the determinant with the N lowest lying states. This is called the reference state, or ground state, having all N particles in states with the lowest possible energies, still obeying the exclusion principle. From now on, all single-particle states within the reference determinant will be labeled $i, j, ...$, whereas states with higher energies are labeled $a, b, ...$. The border between states within the determinant and higher states is called the Fermi level. When referring to states without knowing whether they are above or below the Fermi level, we will label them $p, q, ...$. In this representation, the ground state can be written in the occupancy notation,

$$|\Phi\rangle = |ijkl...\rangle. \tag{3.10}$$

We will now introduce creation and annihilation operators, similar to the ladder operators in section 2.2.1. Instead of raising/lowering the energy of one electron, these operators add or remove one electron from the Slater determinant. Denoting an empty determinant as '$|\rangle$', we can fill it to the reference state by adding one electron at a time using creation operators,

$$|\Phi\rangle = \hat{i}^\dagger \hat{j}^\dagger \hat{k}^\dagger \cdots |\rangle. \tag{3.11}$$

This ground state is sometimes written as $|0\rangle$ for simplicity. It is also possible to remove one electron by the annihilation operator, e.g.

$$\hat{j}|\Phi\rangle = \hat{j}|ijk\cdots\rangle = -\hat{j}|jik\cdots\rangle = -|ik\cdots\rangle. \tag{3.12}$$

The minus sign here comes from the fact that these operators only alter the left-most state, and being antisymmetric one needs to multiply a factor of -1 for each permutation it takes to bring j to the left side of the determinant.

It should not be allowed to annihilate an electron that is not present in the determinant, neither create an already present one. With this in mind, it is clear that the following two statements must be true;

$$\hat{p}^\dagger|\Phi\rangle = \begin{cases} |\Phi^p\rangle & \text{if } p \in a, b, c, \cdots \\ 0 & \text{if } p \in i, j, k, \cdots \end{cases}$$
$$\hat{p}|\Phi\rangle = \begin{cases} 0 & \text{if } p \in a, b, c, \cdots \\ |\Phi_p\rangle & \text{if } p \in i, j, k, \cdots \end{cases} \tag{3.13}$$

where $|\Phi_p\rangle$ means the reference state without p, and $|\Phi^p\rangle$ means the reference state with p added. In the particle-hole formalism everything is relative to the reference, where an added electron is called a particle and a removed one referred to as a hole. Generalizing this one can have multiple particles and holes. To prevent from creating a 0-determinant, the hole states must be in $i, j, ...$, and the particle states in $a, b, ...$. An example could be

$$|\Phi^{ab}_{ijk}\rangle = \hat{a}^\dagger\hat{b}^\dagger\hat{k}\hat{j}\hat{i}|\Phi\rangle. \tag{3.14}$$

Having two particles and three holes, we call this a 2p-3h excitation.

Because of the orthonormal single particle basis (3.9), the determinants will be orthonormal too. To be consistent we define the empty vacuum state to be normalized as well,

$$\langle|\rangle = 1. \tag{3.15}$$

We see the importance of this when using the fact that creation and annihilation operators are each others adjoint,

$$\langle\Phi|\Phi\rangle = \left(\langle|\cdots\hat{j}\hat{i}\right)\left(\hat{i}^\dagger\hat{j}^\dagger\cdots|\rangle\right) = \langle|\left(\cdots\hat{j}\hat{i}\hat{i}^\dagger\hat{j}^\dagger\cdots|\rangle\right). \tag{3.16}$$

Since we first add $i, j, ...$ to the vacuum before we remove the same particles, we end out with a vacuum state again,

$$\langle|\left(\cdots\hat{j}\hat{i}\hat{i}^\dagger\hat{j}^\dagger\cdots|\rangle\right) = \langle|\rangle = 1. \tag{3.17}$$

More rigorously one may calculate this using the anti-commutation rules. Similar to the commutator the anti-commutator is defined as

$$[\hat{A}, \hat{B}]_+ = \hat{A}\hat{B} + \hat{B}\hat{A}. \tag{3.18}$$

We will start out with the annihilation operators by considering

$$\begin{aligned} &\hat{p}\hat{q}|qpij\cdots\rangle = |ij\cdots\rangle \\ &\hat{q}\hat{p}|qpij\cdots\rangle = -\hat{q}\hat{p}|pqij\cdots\rangle = -|ij\cdots\rangle. \end{aligned} \tag{3.19}$$

One permutation is required for $\hat{q}\hat{p}$ since the operators only can act on the leftmost particle. If neither p nor q is in the determinant then both of the expressions return zero. In all cases the anti-commutator should be zero, thus

$$[\hat{p}, \hat{q}]_+ = 0. \tag{3.20}$$

Considering two creation operators using the same procedure, we find that $\hat{p}^\dagger\hat{q}^\dagger = -\hat{q}^\dagger\hat{p}^\dagger$ if neither p nor q is present in the determinant. If at least one of the two is already present we get zero. Again the anti-commutator is zero,

$$[\hat{p}^\dagger, \hat{q}^\dagger]_+ = 0. \tag{3.21}$$

The last step is to look at one creation, and one annihilation operator. If these two represent two different states ($p \neq q$), we have

$$\begin{aligned} \hat{p}^\dagger\hat{q}|qij\cdots\rangle &= |pij\cdots\rangle \\ \hat{q}\hat{p}^\dagger|qij\cdots\rangle &= \hat{q}|pqij\cdots\rangle = -|pij\cdots\rangle. \end{aligned} \tag{3.22}$$

With q missing, or p already present in the determinant the anti-commutator is zero, leading to

$$[\hat{p}^\dagger, \hat{q}]_+ = 0 \quad \text{if} \quad p \neq q. \tag{3.23}$$

If $p = q$, we need to investigate both when p is already present and not,

$$\begin{aligned} \hat{p}\hat{p}^\dagger|pij\cdots\rangle &= 0 \\ \hat{p}^\dagger\hat{p}|pij\cdots\rangle &= |pij\cdots\rangle \\ \hat{p}\hat{p}^\dagger|ij\cdots\rangle &= |ij\cdots\rangle \\ \hat{p}^\dagger\hat{p}|ij\cdots\rangle &= 0, \end{aligned} \tag{3.24}$$

which leads to the relation

$$[\hat{p}^\dagger, \hat{p}]_+ = 1. \tag{3.25}$$

All together we summarize to,

$$\begin{aligned} [\hat{p}, \hat{q}]_+ &= 0 \\ [\hat{p}^\dagger, \hat{q}^\dagger]_+ &= 0 \\ [\hat{p}^\dagger, \hat{q}]_+ &= \delta_{pq}. \end{aligned} \tag{3.26}$$

Using the tools that the anti-commutators present, we could once again look at the normalized inner product

$$\langle i|i\rangle = \langle|\hat{i}\hat{i}^\dagger|\rangle = \langle| - \hat{i}^\dagger\hat{i} + 1|\rangle = \langle| - \hat{i}^\dagger\hat{i}|\rangle + \langle|\rangle = \langle|\rangle = 1. \tag{3.27}$$

The trick here was to switch place for $\hat{i}$ and $\hat{i}^\dagger$ by using the anti-commutation relation, and in the end reason that $\hat{i}|\rangle$ must be zero. For a general (wider) string of operators the same can be applied, but it is tedious, and better methods exist.

3.3.1 Operators

Suppose we rewrite the Hamiltonian from eq. (3.8) as a sum of two terms, $\hat{H}^{(0)}$ and $\hat{H}^{(1)}$, where the first term contains all one-body terms and the second incorporates only the two body potential between electrons,

$$\begin{aligned} \hat{H}^{(0)} &= \sum_{k=0}^{N} \left(\hat{t}_k + \hat{v}_k\right) = \sum_{k=0}^{N} \hat{h}^{(0)}(x_k) \\ \hat{H}^{(1)} &= \frac{1}{2}\sum_{kl}^{N} \hat{v}_{kl}(x_k, x_l). \end{aligned} \tag{3.28}$$

Introducing atomic units, that is setting $\hbar = m_e = e = 1$, these operators can be expressed neatly as

$$\begin{aligned} \hat{t}_k &= -\frac{1}{2}\nabla^2 \\ \hat{v}_k &= \frac{1}{2}\omega^2 x_k^2 \quad \text{(for harmonic oscillator)} \\ \hat{v}_{kl} &= \frac{1}{|x_k - x_l|} \quad \text{(for electrons).} \end{aligned} \tag{3.29}$$

Before we express the operators in second quantization, using the creation and annihilation operators, we will define the number operator, counting the number of occupied states in a determinant, hence the number of particles,

$$\hat{N} = \sum_p \hat{p}^\dagger \hat{p}. \tag{3.30}$$

This is the first example of a second quantized operator, and the most striking is the unrestricted sum, with p looping through all values in our basis. Extending this for the different parts of our Hamiltonian, the one-body operator becomes

$$\hat{H}^{(0)} = \sum_{pq} \langle p|\hat{h}^{(0)}|q\rangle \hat{p}^\dagger \hat{q}, \tag{3.31}$$

and the two-body operator reads

$$\begin{aligned} \hat{H}^{(1)} &= \frac{1}{2}\sum_{pqrs} \langle pq|\hat{v}|rs\rangle \hat{p}^\dagger \hat{q}^\dagger \hat{s}\hat{r} \\ &= \frac{1}{4}\sum_{pqrs} \langle pq||rs\rangle \hat{p}^\dagger \hat{q}^\dagger \hat{s}\hat{r}. \end{aligned} \tag{3.32}$$

A usual interpretation is that $\hat{H}^{(0)}$ excites one particle from q into a state p with a probability of

$$\langle p|\hat{h}^{(0)}|q\rangle = \int \phi_p^*(x_1)\hat{h}^{(0)}(x_1)\phi_q(x_1)dx_1, \tag{3.33}$$

whereas the two-body operator excites two particles from the states r and s into p and q, with a probability amplitude of

$$\langle pq|\hat{v}|rs\rangle = \int\int \phi_p^*(x_1)\phi_q^*(x_2)\hat{v}(x_1,x_2)\phi_r(x_1)\phi_s(x_2)dx_1 dx_2. \tag{3.34}$$

In the two-body case the probability amplitude is written directly as an integral, without taking $|rs\rangle$ as an antisymmetric determinant. To account for this, one often use the antisymmetric element instead, defined as

$$\langle pq||rs\rangle = \langle pq|v|rs\rangle - \langle pq|v|sr\rangle. \tag{3.35}$$

This expression will still not account for the factor $\frac{1}{\sqrt{2}}$ in front of a slater determinant, which is why we need a factor $\frac{1}{4}$ when using this in equation (3.32).

With these forms of our operators, we can calculate expectation values in a general way between two determinants. For instance $\langle ij|\hat{V}|kl\rangle$ can be calculated using anticommutators,

$$\begin{aligned}
\langle ij|\hat{V}|kl\rangle &= \frac{1}{4}\sum_{pqrs}\langle pq||rs\rangle\langle|\hat{j}\hat{i}\hat{p}^\dagger\hat{q}^\dagger\hat{s}\hat{r}\hat{k}^\dagger\hat{l}^\dagger|\rangle = \frac{1}{4}\sum_{pqrs}\langle pq||rs\rangle\langle|\hat{j}\left(\delta_{ip}-\hat{p}^\dagger\hat{i}\right)\hat{q}^\dagger\hat{s}\hat{r}\hat{k}^\dagger\hat{l}^\dagger|\rangle \\
&= \frac{1}{4}\sum_{pqrs}\langle pq||rs\rangle\langle|\delta_{ip}\hat{j}\hat{q}^\dagger\hat{s}\hat{r}\hat{k}^\dagger\hat{l}^\dagger - \hat{j}\hat{p}^\dagger\hat{i}\hat{q}^\dagger\hat{s}\hat{r}\hat{k}^\dagger\hat{l}^\dagger|\rangle \qquad (3.36) \\
&= \frac{1}{4}\sum_{pqrs}\langle pq||rs\rangle\langle|\delta_{ip}\delta_{jq}\hat{s}\hat{r}\hat{k}^\dagger\hat{l}^\dagger + \delta_{ip}\hat{q}^\dagger\hat{j}\hat{s}\hat{r}\hat{k}^\dagger\hat{l}^\dagger - \hat{j}\hat{p}^\dagger\hat{i}\hat{q}^\dagger\hat{s}\hat{r}\hat{k}^\dagger\hat{l}^\dagger|\rangle.
\end{aligned}$$

The second term on the last line of eq. (3.36) has $\hat{q}^\dagger$ as the leftmost operator. Acting from the left on a vacuum bra state, this leads to zero. The philosophy is to continue this process, moving creation operators to the left in all terms. The only contributing terms will then have Kronecker deltas only, i.e.

$$\begin{aligned}
\langle ij|\hat{V}|kl\rangle &= \frac{1}{4}\sum_{pqrs}\langle pq||rs\rangle\langle|\delta_{ip}\delta_{jq}\delta_{rk}\delta_{sl} - \delta_{ip}\delta_{jq}\delta_{sk}\delta_{rl} + \delta_{jp}\delta_{iq}\delta_{sk}\delta_{rl} - \delta_{jp}\delta_{iq}\delta_{rk}\delta_{sl}|\rangle \qquad (3.37) \\
&= \frac{1}{4}\left[\langle ij||kl\rangle - \langle ij||lk\rangle + \langle ji||lk\rangle - \langle ji||kl\rangle\right] = \langle ij||kl\rangle.
\end{aligned}$$

The last step here was to see that all terms are exactly the same, due to the antisymmetric elements, where

$$\langle ij||kl\rangle = -\langle ij||lk\rangle = \langle ji||lk\rangle = -\langle ji||kl\rangle. \qquad (3.38)$$

3.3.2 Wick's theorem

Anticommutators, as seen in the previous section, are powerfull, yet tedious, constructs for calculation of expectation values. Any state can be transformed into a string of operators acting on the vacuum, which we can transform further by anticommutators. For strings of more operators one could automate this process, by using sympy [?] or similar software, but for hand calculations simplifications exist through what is known as the time-independent Wick's theorem.

Having a string of operators $\hat{A}\hat{B}\hat{C}...$, we define the normal-ordered product

$$\left\{\hat{A}\hat{B}\hat{C}...\right\}, \qquad (3.39)$$

as a reordered product, with all creation operators moved to the left, and annihilation operators to the right. A phase phactor of -1 will arise whenever an odd number of permutations is needed in the reordering. Such a product is extremely usefull due to the fact that all expectation values in vacuum yields zero. Furthermore we define a contraction between two operators as the difference between the original ordering and the normal ordering,

$$\overbracket{\hat{A}\hat{B}} = \underbracket{\hat{A}\hat{B}} = \hat{A}\hat{B} - \left\{\hat{A}\hat{B}\right\}. \qquad (3.40)$$

With this approach, four contractions are possible;

$$
\begin{aligned}
\contraction{}{\hat{p}}{}{\hat{q}}\hat{p}\hat{q} &= \hat{p}\hat{q} - \hat{p}\hat{q} = 0 \\
\contraction{}{\hat{p}}{}{\hat{q}}\hat{p}\hat{q}^\dagger &= \hat{p}\hat{q}^\dagger - (-\hat{q}^\dagger\hat{p}) = \delta_{pq} \\
\contraction{}{\hat{p}}{^\dagger}{\hat{q}}\hat{p}^\dagger\hat{q} &= \hat{p}^\dagger\hat{q} - \hat{p}^\dagger\hat{q} = 0 \\
\contraction{}{\hat{p}}{^\dagger}{\hat{q}}\hat{p}^\dagger\hat{q}^\dagger &= \hat{p}^\dagger\hat{q}^\dagger - \hat{p}^\dagger\hat{q}^\dagger = 0.
\end{aligned}
\tag{3.41}
$$

If contractions occur within a normal product, a phase factor of -1 will arise from each permutation that is needed to bring the contracted operators beside each other.

Wick's theorem [8] states that any string of operators can be rewritten as a sum, where the first term is the normal-ordered string. The second term is a sum of all possible normal products with contractions between two operators only. The next term is a sum of possible contractions between four operators, and so on up to a sum of all possibilities where all operators are contracted. The nice feature of this theorem is that all products with normal ordered strings will not give a contribution when evaluated between vacuum states. In this case only terms that are fully contracted will contribute.

If we have a product of two already normal-ordered operator strings, this is rewritten as the normal-ordered string off all operators plus all possible contractions between the first and the second string. As an example we will return to the transition probability from equation (3.37), using Wick's theorem instead,

$$
\begin{aligned}
\langle ij|\hat{V}|kl\rangle &= \frac{1}{4}\sum_{pqrs}\langle pq||rs\rangle\langle|\hat{j}\hat{i}\hat{p}^\dagger\hat{q}^\dagger\hat{s}\hat{r}\hat{k}^\dagger\hat{l}^\dagger|\rangle \\
&= \frac{1}{4}\sum_{pqrs}\langle pq||rs\rangle\langle|
\contraction{}{\hat{j}}{\hat{i}}{\hat{p}}
\contraction[2ex]{\hat{j}}{\hat{i}}{\hat{p}^\dagger}{\hat{q}}
\contraction{\hat{j}\hat{i}\hat{p}^\dagger\hat{q}^\dagger}{\hat{s}}{\hat{r}}{\hat{k}}
\contraction[2ex]{\hat{j}\hat{i}\hat{p}^\dagger\hat{q}^\dagger\hat{s}}{\hat{r}}{\hat{k}^\dagger}{\hat{l}}
\hat{j}\hat{i}\hat{p}^\dagger\hat{q}^\dagger\hat{s}\hat{r}\hat{k}^\dagger\hat{l}^\dagger
+
\contraction{}{\hat{j}}{\hat{i}}{\hat{p}}
\contraction[2ex]{\hat{j}}{\hat{i}}{\hat{p}^\dagger}{\hat{q}}
\contraction{\hat{j}\hat{i}\hat{p}^\dagger\hat{q}^\dagger\hat{s}}{\hat{r}}{}{\hat{k}}
\contraction[2ex]{\hat{j}\hat{i}\hat{p}^\dagger\hat{q}^\dagger}{\hat{s}}{\hat{r}\hat{k}^\dagger}{\hat{l}}
\hat{j}\hat{i}\hat{p}^\dagger\hat{q}^\dagger\hat{s}\hat{r}\hat{k}^\dagger\hat{l}^\dagger
+
\contraction{\hat{j}}{\hat{i}}{}{\hat{p}}
\contraction[2ex]{}{\hat{j}}{\hat{i}\hat{p}^\dagger}{\hat{q}}
\contraction{\hat{j}\hat{i}\hat{p}^\dagger\hat{q}^\dagger\hat{s}}{\hat{r}}{}{\hat{k}}
\contraction[2ex]{\hat{j}\hat{i}\hat{p}^\dagger\hat{q}^\dagger}{\hat{s}}{\hat{r}\hat{k}^\dagger}{\hat{l}}
\hat{j}\hat{i}\hat{p}^\dagger\hat{q}^\dagger\hat{s}\hat{r}\hat{k}^\dagger\hat{l}^\dagger
+
\contraction{\hat{j}}{\hat{i}}{}{\hat{p}}
\contraction[2ex]{}{\hat{j}}{\hat{i}\hat{p}^\dagger}{\hat{q}}
\contraction{\hat{j}\hat{i}\hat{p}^\dagger\hat{q}^\dagger}{\hat{s}}{\hat{r}}{\hat{k}}
\contraction[2ex]{\hat{j}\hat{i}\hat{p}^\dagger\hat{q}^\dagger\hat{s}}{\hat{r}}{\hat{k}^\dagger}{\hat{l}}
\hat{j}\hat{i}\hat{p}^\dagger\hat{q}^\dagger\hat{s}\hat{r}\hat{k}^\dagger\hat{l}^\dagger
|\rangle \\
&= \frac{1}{4}\sum_{pqrs}\langle pq||rs\rangle\langle|\delta_{jp}\delta_{iq}\delta_{sk}\delta_{rl} - \delta_{jp}\delta_{iq}\delta_{rk}\delta_{sl} + \delta_{ip}\delta_{jq}\delta_{rk}\delta_{sl} - \delta_{ip}\delta_{jq}\delta_{sk}\delta_{rl}|\rangle.
\end{aligned}
\tag{3.42}
$$

This is not all fully contracted terms, but with a little reasoning it seems clear that the other terms have at least one contraction that is equal to zero. It is possible to count the number of crossing lines, instead of moving operators close to eachother, to get the correct phase factor. The number of crossings are $(2, 1, 0, 1)$ in the four terms, leading to a minus sign in the second and the last term, due to odd number of crossings. Comparing (3.42) with (3.37), both yield the correct result and using Wick's theorem is much less work.

To further optimize this theorem, one may redefine the normal ordering with moving all creation operators above the fermi level, and all annihilation operators below the fermi level, to the left. With this reordering, all expectation values yield zero when evaluated in the reference state, as a pure consequence of equation (3.13), e.g.

$$
\langle\Phi|\hat{a}^\dagger\cdots\hat{b}|\Phi\rangle = 0 \quad \text{or} \quad \langle\Phi|\hat{i}\cdots\hat{j}^\dagger|\Phi\rangle = 0. \tag{3.43}
$$

The contractions in eq. (3.41) are altered to only two nonzero contractions,

$$\begin{aligned}\overbracket{\hat{i}^\dagger\hat{j}} &= \hat{i}^\dagger\hat{j} - \left(-\hat{j}\hat{i}^\dagger\right) = \delta_{ij} \\ \overbracket{\hat{a}\hat{b}^\dagger} &= \hat{a}\hat{b}^\dagger - \left(-\hat{b}^\dagger\hat{a}\right) = \delta_{ab}.\end{aligned} \tag{3.44}$$

Apart from the redefinition of the normal product, Wick's theorem is unaltered.

3.4 Diagrams

The human brain is, sadly, not well suited for finding possible combinations of contractions, using Wick's theorem. As an example, we present a string of operators arising from the evaluation of the transition probability from a 1p-1h excitation to another 1p-1h excitation for the two-body potential,

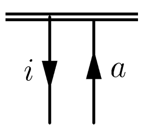
$$\langle\Phi_i^a|\hat{V}|\Phi_j^b\rangle \rightarrow \hat{i}^\dagger\hat{a}\ \hat{p}^\dagger\hat{q}^\dagger\hat{s}\hat{r}\ \hat{b}^\dagger\hat{j}. \tag{3.45}$$

Although this expression only contains eight operators, it leads to fourteen nonzero, fully contracted, terms. One needs to be focused and systematic in order to calculate all terms correctly. However, the brain seems to be good at visualizing mental images, and therefore a graphical presentation of the formulas could serve us well.

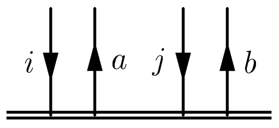
The graphical approach presented here originated in quantum field theory, developed by Richard Feynman. Although originally meant to be used on time dependent transitions from one state to another, it is presented here without a time ordering (following [9]). It does, however, restrict the order in which operators are applied.

Diagrams start out with the reference ket state, denoted by two horizontal lines at the bottom, and end out with the reference bra state, as two horizontal lines at the top. Particle operators are lines pointing upwards, whereas holes point downwards. In this fashion, determinants with excitations from the reference state can be visualized as

$$\langle\Phi_i^a| = \langle\Phi|\hat{i}^\dagger\hat{a} = \tag{3.46}$$



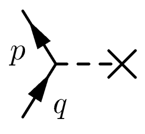
$$|\Phi_{ij}^{ab}\rangle = \hat{a}^\dagger\hat{b}^\dagger\hat{j}\hat{i}|\Phi\rangle = \tag{3.47}$$



.

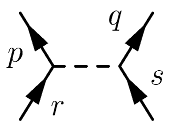
One-body operators are presented as two electron lines connected to a dashed line with a cross,

$$\hat{H^{(0)}} = \sum_{pq}\langle p|\hat{h^{(0)}}|\hat{q}\rangle\hat{p}^\dagger\hat{q} = \tag{3.48}$$



Two-body operators are represented similarly, but have two incoming and two outgoing lines due to the two-body nature,

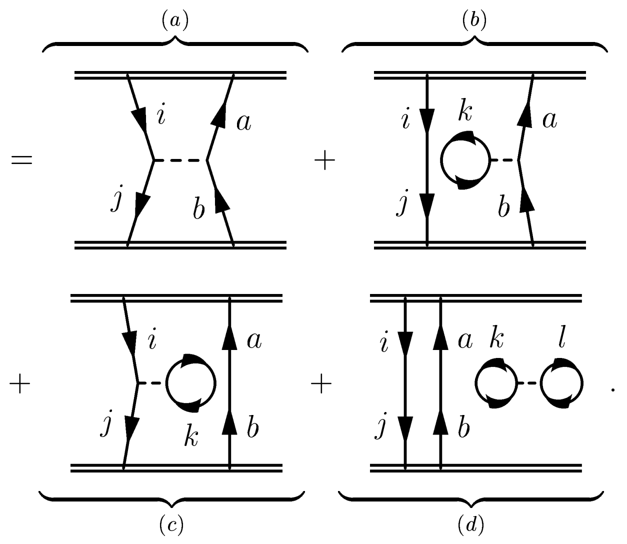
$$\hat{H}^{(1)} = \frac{1}{4}\sum_{pqrs}\langle pq||rs\rangle \hat{p}^\dagger \hat{q}^\dagger \hat{s}\hat{r} = \qquad . \qquad (3.49)$$



The idea now is to represent contractions by connecting lines, and because only fully contracted terms are nonzero within the reference state, all lines should be connected. All *free* indexes are meant to be summed over, and the matrix elements are found by replacing q with the label of incoming line and p with the outgoing label in the one-body case. In the two-body case we replace r/s with left/right incoming line and p/q with left/right outgoing line. To determine the correct phase factor, one needs to count the number of hole lines and the number of closed paths. When counting the number of closed paths we will consider associated particle-hole pairs as if they were connected in the reference states. The phase factor will in the end be $(-1)^{l+h}$, where l is the number of closed paths (loops) and h is the number of hole lines.

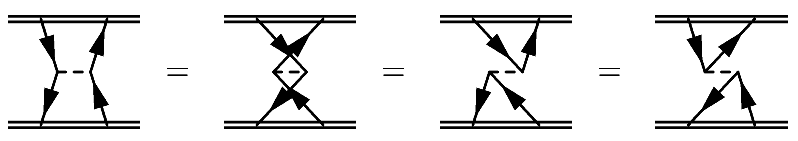
To illustrate the use of diagrams, we return to the example in the introduction, eq. (3.45). This expression is evaluated to four unique ways of connecting the diagrams;

$$\langle \Phi_i^a|\hat{V}|\Phi_j^b\rangle = \qquad (3.50)$$



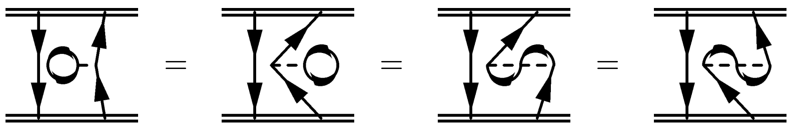
Term (3.50a) has no free indexes since all lines are connected to the particle and hole indices already defined in the reference states. The corresponding matrix element is $\langle ja||ib\rangle$. Having two hole lines, one closed loop, and in total four equal terms,

$$ \qquad , \qquad (3.51)$$

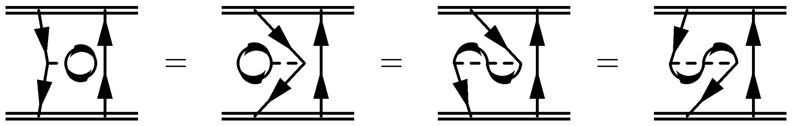


the total factor in front should be $(-1)^{2+1}\cdot 4\cdot \frac{1}{4} = -1$.

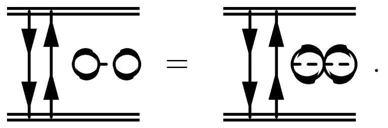
Term (3.50b) corresponds to the element $\delta_{ij}\langle ka||kb\rangle$, where the delta function follows from the contracted hole lines between i and j. There are two hole lines, two loops, and in total four equal terms,

 (3.52)

Term (3.50c) is similar to (3.50b), except how the Kronecker delta connects the particle lines a and b instead. There are now three hole lines, two loops and four equal terms;

 (3.53)

Term (3.50d) has three hole lines, three loops, but only two equal diagrams can be created;

 (3.54)

We then have in total

$$\begin{aligned}\langle\Phi_i^a|\hat{V}|\Phi_j^b\rangle &= \overbrace{(-1)^{1+2}\cdot 4\cdot\frac{1}{4}\langle ja||ib\rangle}^{(a)} + \overbrace{(-1)^{2+2}\cdot 4\cdot\frac{1}{4}\sum_k \delta_{ij}\langle ka||kb\rangle}^{(b)} \\ &\quad + \underbrace{(-1)^{2+3}\cdot 4\cdot\frac{1}{4}\sum_k \delta_{ab}\langle jk||ik\rangle}_{(c)} + \underbrace{(-1)^{3+3}\cdot 2\frac{1}{4}\sum_{kl}\delta_{ab}\delta_{ij}\langle kl||kl\rangle}_{(d)} \\ &= -\langle ja||ib\rangle + \sum_k \delta_{ij}\langle ka||kb\rangle - \sum_k \delta_{ab}\langle jk||ik\rangle + \frac{1}{2}\sum_{kl}\delta_{ab}\delta_{ij}\langle kl||kl\rangle.\end{aligned} \tag{3.55}$$

Diagrams will return when we derive the coupled cluster equations, then presented with a more explicit set of rules for interpretation.

3.5 Normal-ordered Hamiltonian

Based on the second quantized expression for the Hamiltonian from section 3.3.1, we can apply Wick's theorem. Defining $\delta_{pq<F}$ to be a delta function where p and q are below the Fermi level, the string of operators from $\hat{H}^{(0)}$ becomes

$$\hat{p}^\dagger\hat{q} = \left\{\hat{p}^\dagger\hat{q}\right\} + \overset{\sqcap}{\hat{p}^\dagger\hat{q}} = \left\{\hat{p}^\dagger\hat{q}\right\} + \delta_{pq<F}, \tag{3.56}$$

yielding a new expression for the single-particle interactions,

$$\hat{H}^{(0)} = \sum_{pq}\langle p|\hat{h}^{(0)}|q\rangle\left\{\hat{p}^\dagger\hat{q}\right\} + \sum_i\langle i|\hat{h}^{(0)}|i\rangle. \tag{3.57}$$

Similarly for $\hat{H}^{(1)}$ we get

$$
\begin{aligned}
\hat{p}^\dagger\hat{q}^\dagger\hat{s}\hat{r} = {} & \left\{\hat{p}^\dagger\hat{q}^\dagger\hat{s}\hat{r}\right\} + \left\{\contraction{}{\hat{p}^\dagger}{\hat{q}^\dagger\hat{s}}{\hat{r}}\hat{p}^\dagger\hat{q}^\dagger\hat{s}\hat{r}\right\} + \left\{\contraction{}{\hat{p}^\dagger}{\hat{q}^\dagger}{\hat{s}}\hat{p}^\dagger\hat{q}^\dagger\hat{s}\hat{r}\right\} + \left\{\contraction{\hat{p}^\dagger}{\hat{q}^\dagger}{}{\hat{s}}\hat{p}^\dagger\hat{q}^\dagger\hat{s}\hat{r}\right\} \\
& + \left\{\contraction{\hat{p}^\dagger}{\hat{q}^\dagger}{\hat{s}}{\hat{r}}\hat{p}^\dagger\hat{q}^\dagger\hat{s}\hat{r}\right\} + \left\{\contraction[2ex]{}{\hat{p}^\dagger}{\hat{q}^\dagger}{\hat{s}}\contraction{\hat{p}^\dagger}{\hat{q}^\dagger}{\hat{s}}{\hat{r}}\hat{p}^\dagger\hat{q}^\dagger\hat{s}\hat{r}\right\} + \left\{\contraction[2ex]{}{\hat{p}^\dagger}{\hat{q}^\dagger\hat{s}}{\hat{r}}\contraction{\hat{p}^\dagger}{\hat{q}^\dagger}{}{\hat{s}}\hat{p}^\dagger\hat{q}^\dagger\hat{s}\hat{r}\right\} \\
= {} & \left\{\hat{p}^\dagger\hat{q}^\dagger\hat{s}\hat{r}\right\} - \delta_{ps<F}\left\{\hat{q}^\dagger\hat{r}\right\} + \delta_{pr<F}\left\{\hat{q}^\dagger\hat{s}\right\} + \delta_{qs<F}\left\{\hat{p}^\dagger\hat{r}\right\} \\
& - \delta_{qr<F}\left\{\hat{p}^\dagger\hat{s}\right\} - \delta_{ps<F}\delta_{qr<F} + \delta_{pr<F}\delta_{qs<F},
\end{aligned}
\tag{3.58}
$$

and put back into the second-quantized operator we find

$$
\begin{aligned}
\hat{H}^{(1)} = & \frac{1}{4}\sum_{pqrs}\langle pq||rs\rangle\left\{\hat{p}^\dagger\hat{q}^\dagger\hat{s}\hat{r}\right\} - \frac{1}{4}\sum_{qri}\langle iq||ri\rangle\left\{\hat{q}^\dagger\hat{r}\right\} + \frac{1}{4}\sum_{qsi}\langle iq||is\rangle\left\{\hat{q}^\dagger\hat{s}\right\} \\
+ & \frac{1}{4}\sum_{pri}\langle pi||ri\rangle\left\{\hat{p}^\dagger\hat{r}\right\} - \frac{1}{4}\sum_{psi}\langle pi||is\rangle\left\{\hat{p}^\dagger\hat{s}\right\} - \frac{1}{4}\sum_{ij}\langle ij||ji\rangle + \frac{1}{4}\sum_{ij}\langle ij||ij\rangle.
\end{aligned}
\tag{3.59}
$$

Indices are merely dummy variables summed freely over, and together with the properties of antisymmetric interaction elements this can be compressed to

$$
\hat{H}^{(1)} = \frac{1}{4}\sum_{pqrs}\langle pq||rs\rangle\left\{\hat{p}^\dagger\hat{q}^\dagger\hat{s}\hat{r}\right\} + \sum_{pqi}\langle pi||qi\rangle\left\{\hat{p}^\dagger\hat{q}\right\} + \frac{1}{2}\sum_{ij}\langle ij||ij\rangle. \tag{3.60}
$$

Gathering all terms, we have a normal-ordered expression for the Hamiltonian,

$$
\begin{aligned}
\hat{H} = \sum_{pq}\langle p|\hat{h}^{(0)}|q\rangle\left\{\hat{p}^\dagger\hat{q}\right\} + \frac{1}{4}\sum_{pqrs}\langle pq||rs\rangle\left\{\hat{p}^\dagger\hat{q}^\dagger\hat{s}\hat{r}\right\} + \sum_{pqi}\langle pi||qi\rangle\left\{\hat{p}^\dagger\hat{q}\right\} \\
+ \sum_{i}\langle i|\hat{h}^{(0)}|i\rangle + \frac{1}{2}\sum_{ij}\langle ij||ij\rangle.
\end{aligned}
\tag{3.61}
$$

The last two terms in (3.61) are simple constants whereas the first three are all normal-ordered. Evaluating the energy within the reference state, the first three terms would be zero leaving us with the last two, i.e.

$$
E_{ref} \equiv \langle\Phi_0|\hat{H}|\Phi_0\rangle = \sum_{i}\langle i|\hat{h}^{(0)}|i\rangle + \frac{1}{2}\sum_{ij}\langle ij||ij\rangle. \tag{3.62}
$$

Normal-ordered terms are expressed through a one-particle operator $\hat{F}_N$ and a two-particle operator $\hat{V}_N$,

$$
\hat{H}_N \equiv \hat{H} - E_{ref} = \hat{F}_N + \hat{V}_N = \sum_{pq} f_{pq}\{\hat{p}^\dagger\hat{q}\} + \sum_{pqrs}\langle pq||rs\rangle\left\{\hat{p}^\dagger\hat{q}^\dagger\hat{s}\hat{r}\right\}, \tag{3.63}
$$

where

$$
f_{pq} = \langle p|\hat{h}^{(0)}|q\rangle + \sum_{i}\langle pi||qi\rangle. \tag{3.64}
$$

Chapter 4

Systems

4.1 Quantum dots

Strongly confined electrons offer a wide variety of complex and subtle phenomena which pose severe challenges to existing many-body methods. Quantum dots in particular, that is, electrons confined in semiconducting heterostructures, exhibit, due to their small size, discrete quantum levels. The ground states of, for example, circular dots show similar shell structures and magic numbers as seen for atoms and nuclei. In this thesis we study quantum dots confined in a nearly two-dimensional thin layer of a semiconductor. The potential is approximated by a radially symmetric parabolic potential, also called a harmonic-oscillator potential, for which reason they are termed parabolic or circular quantum dots.

Small confined systems, such as quantum dots, have become very popular for experimental study. Beyond their possible relevance for nanotechnology, they are highly tunable in experiments and introduce level quantization and quantum interference in a controlled way. The possibility to manufacture systems with a tailored electronic structure, may improve the electrical or optical properties, a reason why quantum dots are good candidates as components in quantum computers [?], optimized solar cells, laser technology and medical imaging, to name a few.

4.1.1 The Schrödinger equation in spherical coordinates

Circular quantum dots are said to live in only two dimensions, a consequence of precise manufacturing techniques, making the layers so thin that we can omit the third dimension in our computations. The quantum dots are, however, not truly two-dimensional and this could lead to an error in the calculated energies [?].

The harmonic-oscillator potential for two dimensions was encountered in section 2.2.2, but we will now solve the time-independent Schrödinger equation,

$$-\frac{\hbar^2}{2m}\nabla^2\psi + \frac{1}{2}m\omega^2 r^2\psi = E\psi, \tag{4.1}$$

in spherical coordinates instead of Cartesian. Assuming the wave function can be separated into two factors depending on radial distance, r, and the angle, φ, we get

$$\psi = R(r)Y(\varphi). \tag{4.2}$$

Employing polar coordinates also for the momentum operator, the Schrödinger equation reads,

$$-\frac{\hbar^2}{2m}\left(\frac{\partial^2}{\partial r^2}+\frac{1}{r}\frac{\partial}{\partial r}+\frac{1}{r^2}\frac{\partial^2}{\partial\varphi^2}\right)R(r)Y(\varphi)+\frac{1}{2}m\omega^2r^2R(r)Y(\varphi)=ER(r)Y(\varphi). \tag{4.3}$$

Dividing by $\hbar^2 R(r)Y(\varphi)$, multiplying with $-2mr^2$ and reordering terms, we obtain

$$\frac{r^2}{R(r)}\frac{\partial^2 R(r)}{\partial r^2}+\frac{r}{R(r)}\frac{\partial R(r)}{\partial r}-\frac{2mr^2}{\hbar^2}\left(\frac{1}{2}m\omega^2r^2-E\right)=m_l^2=-\frac{1}{Y(\varphi)}\frac{\partial^2 Y(\varphi)}{\partial\varphi^2}, \tag{4.4}$$

where m_l is our constant of separation. Thus, we have two equations,

$$r^2\frac{\partial^2 R(r)}{\partial r^2}+r\frac{\partial R(r)}{\partial r}-\frac{2mr^2}{\hbar^2}\left(\frac{1}{2}m\omega^2r^2-E\right)R(r)=m_l^2R(r), \tag{4.5}$$

$$\frac{\partial^2 Y(\varphi)}{\partial\varphi^2}=-m_l^2Y(\varphi). \tag{4.6}$$

Equation (4.6) is easily recognized as an exponential function,

$$Y(\varphi)=Ke^{im_l\varphi}, \tag{4.7}$$

where K is to be determined by normalization,

$$\int_0^{2\pi}Y(\varphi)d\varphi=1\Rightarrow Y(\varphi)=\frac{1}{\sqrt{2\pi}}e^{im_l\varphi}, \tag{4.8}$$

and rotational invariance determines m_l,

$$Y(\varphi)=Y(\varphi+2\pi)\Rightarrow e^{im_l2\pi}=1\Rightarrow m_l=0,\pm1,\pm2,\pm3,\dots\ . \tag{4.9}$$

Substituting $R(r)=\rho(r)r^{-\frac{1}{2}}$ eq. (4.5) is simplified to

$$-\frac{\hbar^2}{2m}\frac{\partial^2\rho(r)}{\partial r^2}+\left[\frac{1}{2}m\omega^2r^2+\frac{\hbar^2}{2m}\frac{m_l^2-\frac{1}{4}}{r^2}\right]\rho(r)=E\rho(r), \tag{4.10}$$

often named the radial equation. One should note that this is equivalent to the time-independent Schrödinger equation with an *effective* potential instead,

$$V_{eff}(r)=V(r)+\frac{\hbar^2}{2m}\frac{m_l^2-\frac{1}{4}}{r^2}, \tag{4.11}$$

and this strategy can be applied to any system with a spherical symmetric potential. The full solution after normalization,

$$\int_0^\infty|\rho(r)|^2dr=1, \tag{4.12}$$

reads

$$\psi(r,\varphi)=\frac{1}{\sqrt{2\pi}}R(r)e^{im_l\varphi}. \tag{4.13}$$

We will not solve eq. (4.10) for the one-particle quantum dot here, since it is time consuming and somewhat cumbersome. Instead we merely state the results and recommend reading the appendix of Lohne's master's thesis [2] for a full derivation. For further details regarding the harmonic oscillator and solutions to spherically symmetric problems we recommend reading [4]. The solutions have single-particle energies,

$$\epsilon_{n,m_l,m_s} = (1 + |m_l| + 2n)\,\omega, \tag{4.14}$$

with corresponding wave functions,

$$\psi_{n,m_l} = \sqrt{\frac{n!}{\pi\,(n+|m_l|)!}}\beta^{\frac{1}{2}(1+|m_l|)} r^{|m_l|} e^{-\frac{1}{2}\beta r^2} L_n^{|m_l|}\left(\beta r^2\right) e^{im_l\phi}. \tag{4.15}$$

We have used the Laguerre polynomials, $L_n^{|m_l|}(\beta r^2)$, and also defined

$$\beta = \frac{m\omega}{\hbar}. \tag{4.16}$$

Each spatial wave function from eq. (4.15) can have either spin up or spin down.

4.2 Implementation

As a part of this thesis, we have developed a library for running simulations on different types of systems. The implementation is based primarily on two major types of objects; systems and solvers. There are two solvers implemented, HF (Hartree-Fock) and CC (Coupled Cluster) to be discussed later, that should work independently of the type of system as long as the system is derived from the 'System' base class. Systems differ, from a computational point of view, only by the value of the one- and two-particle elements, f_{pq} and $\langle pq||rs\rangle$, as well as the number of hole-states, n_h, the same as the number of particles, along with the number of unoccupied, virtual, particle-states, n_p. In theory, having an infinite basis, there would be infinitely many particle-states, but in practice it is truncated to a finite set.

To implement a system we need to store the elements f_{pq} and $\langle pq||rs\rangle$ in a reasonable way. We explain the storage scheme for the one-particle elements first, but the two-particle part is postponed to section 4.2.1 due to the more complex approach. The different states p and q are assigned an integer value each, ranging from 0 to $n-1$, where n is the number of basis functions included, $n = n_h + n_p$. If p is a hole state, i, we have $i = p \in [0, n_h - 1]$. On the other hand if p is a particle state, a, we still begin indexing from zero, $a = p - n_h \in [0, n_p]$. In total we distinguish between three types of indexing:

$$\begin{aligned} p &\in [0, n-1] \quad \text{(general)},\\ i = p &\in [0, n_h - 1] \quad (p \text{ is a hole state}),\\ a = p - n_h &\in [0, n_p - 1] \quad (p \text{ is a particle state}). \end{aligned} \tag{4.17}$$

For the harmonic-oscillator basis, states are indexed as in fig. 4.1, first varying m_l in ascending order through all possible values in the lowest-lying shell, holding $|m_l| + 2n = 0$

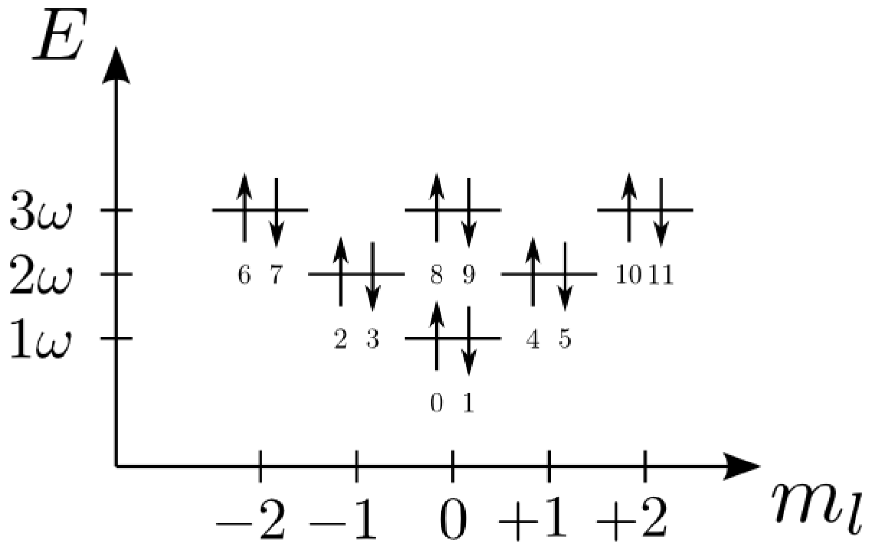


Figure 4.1: The indexing sequence used for quantum dots, implemented in 'HObasis'.

fixed. Even states have spin up whereas odd states have spin down, and after one shell is filled indexing starts in the next, $|m_l|+2n = 1, 2, \cdots$. The number of states in a basis and how they are stored is described by the 'Basis' class. In addition the mappings between integer-indexed states and the actual set of quantum numbers are also included. The actual implementation for the harmonic oscillator, 'HObasis', extends the 'Basis' class. Through 'HObasis' it is possible to query the quantum numbers one state, p, consists of by calling 'HObasis::stateMap(int p)', returning a vector of integer values. As an example for the harmonic oscillator 'HObasis::stateMap(10)' would return a vector

$$\begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix}, \tag{4.18}$$

interpreted as $n = 0$ and $m_l = 2$ with spin up ($0 =\uparrow$ and $1 =\downarrow$).

All implementations for different systems should extend the 'System' base class, which has predefined constructs for holding interaction elements. The simplest implementation of a system can be as simple as shown in listing 4.1, explained as follows;

Line 1: We declare a new subclass of 'System'. This provides storage objects for both one-particle and two-particle interactions as well as *getter*[1] methods to these storage objects.

Line 4: The most important part in the constructor is to create a sensible 'Basis', here using the default basis with 2 electrons (hole-states) and 10 virtual (particle-)states. The method 'fillMatrixElements()' is defined in 'System' and will fill all storage objects with values from 'f_elem' and 'v_elem' using the size of our basis, stored in 'basis'.

[1]In object-oriented design it is common to encapsulate a class' member objects. Data is then declared private, and can only be reached through public member functions, dubbed getter methods. The author of a class may now restrict how data is get, for example to prevent a user from changing important variables.

Listing 4.1: Example on how to implement a specific system by extending the ‘System’ super class.

```
 1 class SpecificSystem1 : public System
 2 {
 3 public:
 4     SpecificSystem1()
 5     {
 6         std::size_t n_elec = 2;
 7         std::size_t n_virtual = 10;
 8         this->basis = new Basis(n_elec, n_virtual);
 9         fillMatrixElements();
10     }
11
12     virtual double f_elem(std::size_t p, std::size_t q) const
13     {
14         double h0_pq = //Some expression for h0
15         double u_pq = 0;
16         for(size_t i = 0; i < basis->get_nH(); i++)
17             u_pq += v_elem(p, i, q, i);
18         return h0_pq + u_pq;
19     }
20
21     virtual double v_elem(std::size_t p, std::size_t q, std::
       size_t r, std::size_t s) const
22     {
23         double v_pqrs = //Some expression for <pq||rs>
24         return v_pqrs;
25     }
26 };
```

Line 12: The normal-ordered one-particle element f_{pq}, defined in eq. (3.64), is calculated by ‘f_elem’.

Line 21: The method ‘v_elem’ defines a way to calculate the antisymmetrized two-particle element $\langle pq||rs\rangle$, defined in eq. (3.35).

One should in particular note that these overridden functions are declared virtual. In this way it is possible to create solvers taking a pointer to any ‘System’, still invoking the methods from the correct subclass. Listing 4.2 illustrates this, using a ‘System’ pointer to access one single-particle element from a specific subclass.

One-particle elements are stored in three matrices: ‘f_hh’, ‘f_ph’ and ‘f_pp’. The three matrices are blocks sorted by the domain the two indices p and q belongs to. If both indices are hole states this belongs to ‘f_hh’ and if both are particle states the element is found in ‘f_pp’. When there are one particle state and one hole state the ‘f_ph’ matrix

Listing 4.2: A subclass has implemented the function ‘f_elem()’ returning the element f_{02}. The ‘System’ base class has this method declared virtual, resulting in invoking the subclass implementation even when using a super-class pointer.

```
//anySystem can point to any system,
System * anySystem = new SpecificSystem();
//and still get an element from the correct subclass
double f_02 = anySystem->f_elem(0,2);
```

Listing 4.3: Continuing listing 4.2 extracting the same element, f_{02}, now through the raw storage matrix for f. This example assumes two occupied (hole) states, thus making index 2 the first (0) particle state. With symmetric interaction matrices we expect $f_{ph} = f_{hp}^T$.

```
mat const * f_hp;
//Get element from raw storage matrix.
f_hp = trans(anySystem->get_f_ph());
f_02 = (*f_ph)(0,0);
```

is used, and, since the interaction is Hermitian, we can obtain ‘f_hp’ as the transposed matrix.

Another way of accessing the same element as in listing 4.2 is to use the storage matrices directly. The storage matrices for one-particle elements are extracted by calling ‘get_f_hh()’, ‘get_f_ph()’ or ‘get_f_pp()’. To serve an example we recall ‘SpecificSystem1’ having $n_h = 2$ and $n_p = 10$. If one would like to extract f_{02} one must note that 2 is the first particle-state, and 0 is the first hole-state. Indexing starts at zero, and because f is symmetric we see how this can be found in the ‘f_ph’ part,

$$f_{02} = f_{20} = f_{00}^{ph}, \tag{4.19}$$

as implemented in listing 4.3.

Although the implementation in listing 4.1 is very simple, counting only 26 lines of code, this would work perfectly for small systems. For larger systems one would encounter two problems. Invoking ‘fillMatrixElements()’ will force calculation of all elements every time a ‘SpecificSystem1’ is constructed, a problem discussed later in section 4.2.2. The other problem is that the dimensionality of two-particle interactions scales as the number of virtual states to the fourth power, possibly solved by exploiting symmetries in our Hamiltonian to avoid store a tremendous amount of zero-filled elements.

4.2.1 Symmetries in the Hamiltonian

Two-particle matrix elements $\langle pq||rs\rangle$ can be interpreted as the probability for two particles making a transition from states r and s into p and q. If our Hamiltonian preserves some quantum numbers, this transition is not possible, and thus zero, whenever pq does not preserve the numbers from rs. In the case of quantum dots, we encounter a Hamiltonian that is spherical symmetric, and not spin dependent. This results in preserving both the angular-momentum quantum number $m(\equiv m_l)$ as well as spin m_s. Defining $M = m^p + m^q, M' = m^r + m^s$ and $M_s = m_s^p + m_s^q, M_s' = m_s^r + m_s^s$, the following holds true,

$$\langle n^p m^p m_s^p; n^q m^q m_s^q || n^r m^r m_s^r; n^s m^s m_s^s \rangle = 0 \quad \text{if} \quad M \neq M' \quad \text{or} \quad M_s \neq M_s'. \tag{4.20}$$

More generally we define a mapping $(p, q) \leftrightarrow (\lambda, \pi)$ where λ is a transition channel, consisting of a specific set of preserved quantum numbers, and a configuration π for each possible combination of (p, q) within that channel, such that

$$\langle \lambda\pi || \lambda'\pi' \rangle = 0 \quad \text{if} \quad \lambda \neq \lambda'. \tag{4.21}$$

This re-indexing allows us to store the interaction elements as a block-diagonal matrix, one block for each channel λ, stored as a matrix $\langle \pi || \pi' \rangle_\lambda$. We further split the interaction matrices whether their indices are above (a, b, c, d) or below (i, j, k, l) the Fermi-level, resulting in six matrices,

$$\begin{aligned}
&\langle ij||kl\rangle \\
&\langle aj||kl\rangle = -\langle ja||kl\rangle = \langle kl||aj\rangle = -\langle kl||ja\rangle \\
&\langle ab||kl\rangle = \langle kl||ab\rangle \\
&\langle aj||cl\rangle = -\langle aj||lc\rangle = \langle ja||lc\rangle = -\langle ja||cl\rangle \\
&\langle ab||cl\rangle = -\langle ab||lc\rangle = \langle cl||ab\rangle = \langle lc||ab\rangle \\
&\langle ab||cd\rangle .
\end{aligned} \tag{4.22}$$

Splitting configurations into configurations for hole-hole states, particle-hole states and particle-particle states, yielding the mappings

$$\begin{aligned}
(i, j) &\leftrightarrow (\lambda, \mu), \\
(a, j) &\leftrightarrow (\lambda, \nu), \\
(a, b) &\leftrightarrow (\lambda, \xi),
\end{aligned} \tag{4.23}$$

we can use a storage scheme in ‘System’ objects splitting interactions into six parts similarly to eq. (4.22), exploiting the sparseness to store block-diagonal parts of the six matrices only,

$$\begin{aligned}
&\langle \mu||\mu'\rangle_\lambda, \quad \langle \nu||\mu\rangle_\lambda, \\
&\langle \xi||\mu\rangle_\lambda, \quad \langle \nu||\nu\rangle_\lambda, \\
&\langle \xi||\nu\rangle_\lambda, \quad \langle \xi||\xi\rangle_\lambda.
\end{aligned} \tag{4.24}$$

Mappings from states into channels and configurations are managed by the ‘Basis’ class, and one could access these mappings through the eight methods in listing 4.4. In this context two general states are encoded to one index, I, i.e.

$$\begin{aligned}
\mathrm{I}(pq) &= p + q \cdot (n_h + n_p) \\
\mathrm{I}(lm) &= l + m \cdot n_h \\
\mathrm{I}(dl) &= d + l \cdot n_p \\
\mathrm{I}(de) &= d + e \cdot n_p,
\end{aligned} \tag{4.25}$$

where n_h is the number of hole-states, and n_p the number of particle-states. Returning to our example of ‘SpecificSystem1’ we have $n_h = 2, n_p = 10$, and trying to access the matrix element $\langle 20||20\rangle$ we can use either the procedure from listing 4.5, extracting the raw matrices, or use the ‘v_elem’ method as in listing 4.6.

Listing 4.4: There are eight different mappings possible to access through the basis object.

```
//Mappings from (pq,lm,dl,de) into lmd and (pi,mu,nu,xi)
get_map_lmdPI_pq();
get_map_lmdMU_lm();
get_map_lmdNU_dl();
get_map_lmdXI_de();

//Reverse mappings
get_map_pq_lmdPI();
get_map_lm_lmdMU();
get_map_dl_lmdNU();
get_map_de_lmdXI();
```

Listing 4.5: We try to access the element $\langle 20||20\rangle$ by mapping the $|20\rangle$ particle-hole state into a channel and a configuration, $|\nu\rangle_\lambda$.

```
//Create a specific system
SpecificSystem1 sys;

//Get informations about its basis
Basis const * basis = sys.get_basis();
int nH = basis->get_nH(); //hole-states
int nP = basis->get_nP(); //particle-states

//(d,l) = (2,0) is a particle hole configuration.
int d = 2 - nH;
int l = 0;
int dl = d + l * nH;

//Map into a channel and a configuration.
umat const * map = basis->get_map_dl_lmdNU();
int lmd = (*map)(0,dl); //first row contains lambda values.
int nu = (*map)(1,dl); //second row contains configurations.

//Retreive element;
vector<mat> const * phph = sys->get_v_phph();
double elem_20_20 = phph->at(lmd)(nu,nu);
```

Listing 4.6: Continuing listing 4.5, accessing the same element again, $\langle\nu||\nu\rangle_\lambda = \langle 20||20\rangle$, now using the reverse mapping, $|\nu\rangle_\lambda \rightarrow |20\rangle$.

```
//Map back to two states d,l
vector<uvec> const * mapback = basis->get_map_lmdNU_dl();
dl = mapback->at(lmd)(nu);
d = dl % nP + nH;
l = dl / nP;

//this should return the same element
elem_20_20 = sys->v_elem(d,l,d,l);
```

The default 'Basis' base class will assume only one transition channel, resulting in storing all interaction elements. For quantum dots we have $M, M_s \leftrightarrow \lambda$, implemented in 'HOBasis'. As an example of the reduced dimensionality, we consider a harmonic oscillator basis with 20 electrons and 400 virtual states. The largest matrix would be $\langle ab||de\rangle$ with a dimensionality of 400^4, resulting in $\sim$ 200GB of storage! Exploiting the symmetries and employing an 'HOBasis' instead reduces this to $\sim$ 1.75GB. Readers are encouraged to read through this implementation as an example of subclassing 'Basis'.

4.2.2 Reading elements from file

Another time-consuming part is to calculate the matrix elements, which could be performed once and later read from file. To create a system read from file, it is convenient to create a subclass of 'TPfile', and we take the actual implementation for circular quantum-dots 'CQDot' as an example. There are three methods, shown in listing 4.7, which must be overloaded. The constructor takes three arguments; the number of filled shells, the total number of shells and the frequency, ω.

Looking at the constructor's implementation, listing 4.8, we remark how 'fillMatrix-Elements' is no longer called to fill both one- and two-particle elements. Instead we read elements $\langle pq||rs\rangle$ from file using 'readFileName', and later calculate f_{pq} by invoking 'fillF-matrix'. Both these methods are public, and can therefore be invoked from the main program. We create a basis, 'HOBasis', store the frequency in a private variable, and create a reverse statemap to be used later when reading a file. This reverse statemap assigns a unique key (unsigned int), constructed from the three quantum numbers, to each state, and it is later possible to get the single index for any state by constructing the same key.

One-particle elements are still calculated through 'f_elem()', implementing the single-particle energies from eq. (4.14) combined with a normal-ordered part as in eq. (3.64), viz

$$f_{pq} = \delta_{pq}\left(1 + |m| + 2n\right)\omega + \sum_i \langle pi||qi\rangle. \tag{4.26}$$

Depending on $\langle pi||qi\rangle$, one-particle elements are required to be filled after two-particle elements are read from file, by invoking 'fillFmatrix'.

Two-particle elements are read through the method 'readFileName', redirecting to 'readFileStream' in listing 4.9. The file-reading itself is already implemented in the base

Listing 4.7: The three most important parts a subclass of 'TPfile' would need. The actual implementation of 'CQDot' includes some extra methods and private variables.

```cpp
class CQDot : public TPfile
{
public:
  //#1- Constructor
  CQDot(int filledR, int shellsR, double omega = 1.0);

  //#2- Calculation of one-particle element f_{pq}
  virtual double f_elem(std::size_t p, std::size_t q) const
  {
    //Part arising from normal ordering
    double u_pq = 0.0;
    for (int i = 0; i < basis->get_nH(); i++)
      u_pq += v_elem(p, i, q, i);

    //Part from h0
    ivec3 p_NMMs = basis->stateMap(p);
    int n = pNMMs(0);
    int m = pNMMs(1);
    double h0 = omega * (2 * n + abs(m) + 1);

    //h0 is diagonal
    if (p == q)
      return h0 + u_pq;
    else
      return u_pq;
  }

protected:
  //#3- A recipe on how to decode tp-element file.
  virtual size_t interp_qNum(char* qNumbers) const;
};
```

Listing 4.8: Constructor for CQDot.

```cpp
CQDot::CQDot(int filledR, int totalR, double omega)
{
    //Initialize basis and frequency
    basis = new HOBasis(filledR, totalR);
    this->omega = omega;

    //Creating reversed state map, needed when reading file
    int ntot = basis->get_nH() + basis->get_nP();
    for (int state = 0; state < ntot; state++)
    {
        //Extract quantum numbers n,m,m_s
        ivec3 nmms = basis->stateMap(state);
        unsigned int n = nmms(0);
        int m = nmms(1);
        unsigned int ms = nmms(2);

        //Each state has a unique key.
        unsigned int key = n + (m + (ms + hMaxM) * maxM) *
           maxM;
        revMap[key] = state;
    }
}
```

Listing 4.9: How to read tp-elements for CQDot.

```
void CQDot::readFileStream(std::ifstream &file)
{
    //First read file as done in super class.
    TPfile::readFileStream(file);
    //Then scale by sqrt(omega)
    size_t dimLMD = basis->dim_lmd_2p();
    for (size_t lmd = 0; lmd < dimLMD; lmd++)
    {
        v_hhhh.at(lmd) = sqrt(omega) * v_hhhh.at(lmd);
        v_phhh.at(lmd) = sqrt(omega) * v_phhh.at(lmd);
        v_pphh.at(lmd) = sqrt(omega) * v_pphh.at(lmd);
        v_phph.at(lmd) = sqrt(omega) * v_phph.at(lmd);
        v_ppph.at(lmd) = sqrt(omega) * v_ppph.at(lmd);
        v_pppp.at(lmd) = sqrt(omega) * v_pppp.at(lmd);
    }
    //Fill one-particle elements.
    fillFmatrix();
}
```

class, and we expect elements on file to be for $\omega = 1$, such that we need only to scale ω after reading. Invoking 'TPfile::readFileStream' will force the base-class implementation to be run even though these functions are declared as virtual. The last steps are to scale elements[2] by $\sqrt{\omega}$ and fill the single-particle elements.

In order to successfully read elements from file the base class needs some information about the file structure, found through the sub-class implementation of 'interp_qNum'. A basic file format is expected to be binary with the content,

```
p q r s <pq||rs> p' q' r' s' <p'q'||r's'> ... ,
```

p is a number of bytes, n_b, needed to specify the state p, q is n_b bytes necessary to specify q, and so on until the element itself $\langle pq||rs\rangle$ stored as a 'double'. The implementation for quantum dots may serve as an example. Here n, m and m_s are coded as an unsigned short, a short and a char. This is repeated for all four states an element consists of, followed by the element itself,

$$\underbrace{\overbrace{n^p}^{\text{ushort (2B)}}\ \overbrace{m^p}^{\text{short (2B)}}\ \overbrace{m_s^p}^{\text{char (1B)}}}_{\text{numBytes (5B)}}\underbrace{n^q m^q m_s^q}_{\text{(5B)}}\underbrace{n^r\ m^r\ m_s^r}_{\text{(5B)}}\underbrace{n^s\ m^s\ m_s^s}_{\text{(5B)}}\underbrace{\langle pq||rs\rangle}_{\text{double (8B)}}\ . \tag{4.27}$$

On most platforms this would require 5 bytes for each state, and 8 bytes for the element, as noted in parentheses. In order to know how many bytes are expected to describe each

[2]Only standard interaction can be scaled by $\sqrt{\omega}$, forcing us to create another file-reading routine 'readEffFileStream' (and 'readEffFileName') for effective interactions. See section 7.1.2.

state, ‘interp_qNum(NULL)’ is queried. Once those bytes are read from file they are sent to the same method as a char array, and expected return is a single integer indexing the correct state. For ‘CQDot’, implementation in listing 4.10, the char array is split into an unsigned short, short and a char, and the correct state is found using the reverse state map that was initialized in the constructor.

A complete file can be read from file in a simple way, as listing 4.11 shows, reading one element at a time, adding a simple loop to read all elements in a file. The functionality to read files is already implemented in the ‘TPfile’ class, as the two methods ‘readFileName’ and ‘readFileStream’.

4.3 Other systems

Other systems may equally well be implemented, and should be compatible with the different solvers as long as they extend the ‘System’ and ‘Basis’ base classes properly. As an example we have implemented a simple class, ‘Atoms’, outlined in listings 4.12 and 4.13, that include the 8 lowest-lying s-states[3] of atomic orbitals. The single-particle energies are found in the Bohr-model to be

$$E_n = \frac{Z^2}{2n^2}, \quad \text{(in atomic units)}, \tag{4.28}$$

and the two-particle correlation elements are found by integrating the Coloumb energies for the atomic orbitals, either calculated analytically by hand or computer algebra software, or by numerical integration.

If we were to extend the Harmonic Oscillator basis to three dimension, we would have a basis often used in nuclear physics. In that case, one would need one additional quantum numbers for the orbitals, as well as quantum numbers describing different types of particles. Although not done in this thesis, it should, in the author’s opinion, not be too complicated. However, nuclear calculations often require a larger dimensionality, and further simplifications, known as jj scheme, exist.

[3]Atomic orbitals are distinguished by their angular momentum quantum number l. If $l = 0$ the state is said to be **s**harp, an ‘s-state’. Continuing, for $l = 1, 2, 3$, we have **p**rinciple, **d**iffuse and **f**undamental.

Listing 4.10: Implementation of ‘interp_qNum’ in ‘CQDot’.

```
size_t CQDot::interp_qNum(char* qNumbers) const
{
    //Quantum numbers n, m, m_s read from file.
    unsigned short n;
    short m;
    char ms;

    //Single indexed state
    size_t state;

    //The number of bytes for each state
    if (qNumbers == NULL)
        return sizeof (n) + sizeof (m) + sizeof (ms);

    //Split qNumbers!
    size_t siz_n = sizeof (n);
    memcpy(&n, qNumbers, siz_n);
    size_t siz_m = sizeof (m);
    memcpy(&m, qNumbers + siz_n, siz_m);
    size_t siz_ms = sizeof (ms);
    memcpy(&ms, qNumbers + siz_n + siz_m, siz_ms);

    //Encode spin
    if (ms == -1)
        ms = 1; //down
    else if (ms == +1)
        ms = 0; //up

    //unique key for any state
    unsigned int key = n + (m + (ms + hMaxM) * maxM) * maxM;
    //revMap is a std::map filled in the constructor
    const map<unsigned int, unsigned int>::const_iterator
       found = revMap.find(key);

    if (found == revMap.end())
        state = maxM;  //A state outside the current basis
    else
        state = found->second; //State found in current basis

    return state;
}
```

Listing 4.11: How to read two-particle elements from file.

```
void TPfile::readFileStream(std::ifstream &file)
{
  //Number of bytes to read for each state.
  size_t qNumWidth = interp_qNum(NULL);
  char * qNumP = new char[qNumWidth];
  char * qNumQ = new char[qNumWidth];
  char * qNumR = new char[qNumWidth];
  char * qNumS = new char[qNumWidth];

  //Allocate memory for tp-elements.
  //......

  while (true)
  {
    //Read one 'line' from file
    double element = 0;
    file.read(qNumP, sizeof (char) * qNumWidth);
    file.read(qNumQ, sizeof (char) * qNumWidth);
    file.read(qNumR, sizeof (char) * qNumWidth);
    file.read(qNumS, sizeof (char) * qNumWidth);
    file.read((char*) &element, sizeof (double));
    if (file.eof())
      break;  //End of file?

    //Find the index
    size_t p = interp_qNum(qNumP);
    size_t q = interp_qNum(qNumQ);
    size_t r = interp_qNum(qNumR);
    size_t s = interp_qNum(qNumS);

    //Store all antisymmetric possibilities
    storePQRS(p, q, r, s, element);
    storePQRS(p, q, s, r, -element);
    storePQRS(q, p, s, r, element);
    storePQRS(q, p, r, s, -element);
  }

  //Free the four char arrays
  //......
}
```

Listing 4.12: Simple implementation for the first sharp atomic orbitals. Continued in listing 4.13.

```cpp
class Atoms : public System
{
public:
  /** numElec is 2 for helium, 4 for beryllium */
  Atoms(int numElec = 2)
  {
    //Charge from nucleus defaults to number of electrons
    Z = numElec;
    //There are only 8 states implemented
    basis = new Basis(numElec, 8 - numElec);
    //Fill matrices
    fillMatrixElements();
  }

  virtual double f_elem(std::size_t p, std::size_t q) const
  {
    //Normal-ordered part
    double u_pq = 0;
    for (int i = 0; i < basis->get_nH(); i++)
        u_pq += v_elem(p, i, q, i);
    //single-particle energies
    int n = p / 2 + 1;   //n is the energy level
    double h0 = -pow((double) Z, 2) / (2 * n * n);
    //h0 is diagonal
    if (p == q)
      return h0 + u_pq;
    else
      return u_pq;
  }
```

Listing 4.13: Continuation of listing 4.12.

```
  virtual double v_elem(
    std::size_t p, std::size_t q,
    std::size_t r, std::size_t s) const
  {
    //Get spin of particles
    bool pUP = p % 2 == 0;
    //... same for q,r and s ...

    //Map p,q,r,s to n quantum number
    int n_p = p / 2 + 1;
    //... same for q,r and s ...

    //First term of element
    double term1 = 0;
    if (pUP == rUP && qUP == sUP)
        term1 = spatial_integral(n_p, n_q, n_r, n_s);
    //Second term
    double term2 = 0;
    if (pUP == sUP && qUP == rUP)
        term2 = spatial_integral(n_p, n_q, n_s, n_r);

    return term1 - term2;
  }

  /** The spatial integral of correlation energies can be
  /*  found analytically (and tabulated) using Mathematica
  /*  or Maple. Also possible to find through numerical
  /*  integration. (four lowest lying orbitals only) */
  double spatial_integral(int n_p, int n_q, int n_r, int n_s)
      const;

private:
  /** The total charge in the nucleus */
  double Z ;
};
```

Chapter 5

Coupled cluster theory

Coupled cluster (CC) is an *ab initio* method for solving the quantum mechanical many-body problem. It was first introduced by Coester and Kümmel during the 1950s [10, 11], originally developed for problems in nuclear physics, but later reformulated for systems of electrons by Čižek in 1966 [12]. It is one of the most popular post-Hartree-Fock methods today. Not too computationally expensive, also gaining good accuracy as well as possessing important features like size-consistency and size-extensivity.

Size-extensive, or just extensive, implies that the energy scales correctly with the number of subunits. Adding more electrons the energy should still have an error scaling proportional to the number of electrons. Size-consistency is best described by having two independent systems. Coupled cluster should now report the same result for one calculation on both systems as for adding together the results from individual calculations on the two systems. Although important, these features are not always well defined, and a more precise discussion is conducted in [9].

5.1 The exponential ansatz

The foundation for most many-body methods is to express the correct wave function by an expansion in a set of basis functions. One example is the Hartree-Fock (HF) method which employs a unitary transformation of the single-particle wave functions,

$$|\lambda\rangle = \sum_{\psi} C_{\lambda\psi} |\psi\rangle, \tag{5.1}$$

and approximates the ground state with a reference Slater determinant built up by these transformed wave functions. Another example is configuration interaction (CI) where the reference determinant is set to a linear expansion of determinants, including the initial reference determinant, 1p-1h excitations, 2p-2h excitations and so on, i.e.

$$|\Psi_0^{CI}\rangle = C_0|\Phi_0\rangle + \sum_{ia} C_i^a |\Phi_i^a\rangle + \sum_{ijab} C_{ij}^{ab} |\Phi_{ij}^{ab}\rangle + \cdots \ . \tag{5.2}$$

In all these methods one needs to solve a set of coupled equations to find the coefficients.

The coupled-cluster method also expands the exact solution in a set of Slater determinants, but employs a non-linear expansion through the exponential ansatz,

$$|\Psi_0^{CC}\rangle = e^{\hat{T}}|\Phi_0\rangle, \tag{5.3}$$

where $\hat{T}$ is the cluster operator including *all* possible excitations on the reference determinant. Sorting excitations by the number of excited electrons, we may generally express this general cluster operator as a sum of a 1p-1h operator, a 2p-2h operator, and so on,

$$\hat{T} = \hat{T}_1 + \hat{T}_2 + \hat{T}_3 + \cdots \ . \tag{5.4}$$

In the form of second-quantized operators the 1p-1h cluster operator is defined as

$$\hat{T}_1 = \sum_{ia} t_i^a \hat{a}^\dagger \hat{i}, \tag{5.5}$$

the 2p-2h cluster operator as

$$\hat{T}_2 = \frac{1}{4}\sum_{ijab} t_{ij}^{ab} \hat{a}^\dagger \hat{b}^\dagger \hat{j}\hat{i}, \tag{5.6}$$

continuing up to

$$\hat{T}_n = \left(\frac{1}{n!}\right)^2 \sum_{ij\cdots ab\cdots} t_{ij\cdots}^{ab\cdots} \hat{a}^\dagger \hat{b}^\dagger \cdots \hat{j}\hat{i}. \tag{5.7}$$

As long as we have a complete single-particle basis and include all possible excitations up to np-nh in a system with n particles, we should find the exact solution for both CI, $|\Psi_0^{CI}\rangle$, and CC, $|\Psi_0^{CC}\rangle$.

It is of course not doable to find the exact solution in practice. A complete single-particle basis would typically mean an infinite set of basis functions. In addition we would need all excitations up to np-nh determinants, yielding a huge amount of determinants. This is why all methods need a computational cut-off. Two types of truncations are used. First we truncate the number of single-particle basis functions, throwing away the ones with the highest energies, which makes sense since our interest is in the ground-state energy. Second we truncate the number of excitations we include, giving rise to different types of coupled-cluster methods. Including the singly excited cluster operator the method is labeled by S, for 'single', and including the doubly excited cluster operator the label D is added, for 'double'. This thesis has an emphasis on CCSD – 'Coupled Cluster Singles and Doubles'.

5.2 Derivation of the CCSD-equations

It is quite tedious work to derive the CCSD equations by hand, but we will present a diagrammatic approach in this section along with some explaining comments where needed.

We aim to find the solution to the time-independent energy eigenvalue equation,

$$\hat{H}|\Psi_0\rangle = E_0|\Psi_0\rangle \cong \hat{H}e^{\hat{T}}|\Phi_0\rangle = E_0 e^{\hat{T}}|\Phi_0\rangle, \tag{5.8}$$

assuming $|\Psi_0\rangle$ can be approximated by the exponential ansatz (5.3). In principle exact, we get an approximation in (5.8) whenever the single-particle basis or the number of excitations included are truncated. It is useful to project $\langle\Phi_0|e^{-\hat{T}}$ onto the eigenvalue equation to get an explicit expression for the energy,

$$\langle\Phi_0|e^{-\hat{T}}\hat{H}e^{\hat{T}}|\Phi_0\rangle = \langle\Phi_0|e^{-\hat{T}}E_0 e^{\hat{T}}|\Phi_0\rangle = E_0. \tag{5.9}$$

We may also project an excited determinant, $\langle\Phi_{exc.}|e^{-\hat{T}}$, onto the equation. Assuming orthonormal basis functions, and exploiting the fact that excited states should be orthogonal to the reference determinant, we get

$$\langle\Phi_{exc.}|e^{-\hat{T}}\hat{H}e^{\hat{T}}|\Phi_0\rangle = \langle\Phi_{exc.}|e^{-\hat{T}}E_0 e^{\hat{T}}|\Phi_0\rangle = E_0\langle\Phi_{exc.}|\hat{1}|\Phi_0\rangle = 0. \tag{5.10}$$

Further simplification is found if we use the normal-ordered Hamiltonian, (3.63), leading to

$$\begin{aligned} \langle\Phi_0|e^{-\hat{T}}\hat{H}_N e^{\hat{T}}|\Phi_0\rangle &= \langle\Phi_0|e^{-\hat{T}}\hat{H}e^{\hat{T}}|\Phi_0\rangle - E_{ref} = E_0 - E_{ref} \\ \langle\Phi_{exc.}|e^{-\hat{T}}\hat{H}_N e^{\hat{T}}|\Phi_0\rangle &= \langle\Phi_{exc.}|e^{-\hat{T}}(E_0 - E_{ref})e^{\hat{T}}|\Phi_0\rangle = 0. \end{aligned} \tag{5.11}$$

The reference energy E_{ref} is here the energy of the reference Slater determinant, as defined in eq. (3.62), not the same as the exact (within the selected truncation) ground-state energy, E_0.

In the following derivation i, j, k and a, b, c will refer to indices in the bra state, whereas $d, e, ...$ and $l, m, ...$ are free indices to be summed over, still restricting $a, b, c, d, e, ...$ to be particle states and $i, j, k, l, m, ...$ to be hole states. We define the similarity transformed Hamiltonian, denoted $\bar{H}$, which can be evaluated as a series of commutators using the Baker-Campbell-Hausdorff formula [?],

$$\bar{H} \equiv e^{-\hat{T}}\hat{H}_N e^{\hat{T}} = \hat{H}_N + \left[\hat{H}_N, \hat{T}\right] + \frac{1}{2!}\left[\left[\hat{H}_N, \hat{T}\right], \hat{T}\right] + \frac{1}{3!}\left[\left[\left[\hat{H}_N, \hat{T}\right], \hat{T}\right], \hat{T}\right] + \cdots \ . \tag{5.12}$$

There are three equations that, in the case of singles and doubles, we need to solve,

$$\begin{aligned} \langle\Phi_0|\bar{H}|\Phi_0\rangle &= E_0 - E_{ref} \\ \langle\Phi_i^a|\bar{H}|\Phi_0\rangle &= 0 \\ \langle\Phi_{ij}^{ab}|\bar{H}|\Phi_0\rangle &= 0. \end{aligned} \tag{5.13}$$

The first gives us an explicit form for the energy, whereas the other two will determine the amplitudes in the cluster operators.

Let $\hat{A}$ and $\hat{B}$ be two normal-ordered strings of operators containing an even number of creation and annihilation operators. Using Wick's generalized theorem, their commutator would be

$$\left[\hat{A}, \hat{B}\right] = \left\{\hat{A}\hat{B}\right\} - \left\{\hat{B}\hat{A}\right\} + \left\{\overset{\sqcap}{\hat{A}\hat{B}}\right\} - \left\{\overset{\sqcap}{\hat{B}\hat{A}}\right\}, \tag{5.14}$$

where the two uncontracted terms are the same, and we thus end up with

$$\left[\hat{A},\hat{B}\right] = \left\{\overbracket{\hat{A}\hat{B}}\right\} - \left\{\overbracket{\hat{B}\hat{A}}\right\}. \tag{5.15}$$

We observe that the cluster operators are already normal-ordered, and also note how no contractions between a cluster operator on the left and any other operator on the right can be non-zero. This is because the $\hat{T}$ operator contains $\hat{a}^\dagger\hat{b}^\dagger\cdots\hat{j}\hat{i}$, none of which can lead to any non-zero contraction from (3.44). The cluster operators thus commute, yielding

$$\left[\hat{T}_n,\hat{T}_m\right] = \left\{\overbracket{\hat{T}_n\hat{T}_m}\right\} - \left\{\overbracket{\hat{T}_m\hat{T}_n}\right\} = 0. \tag{5.16}$$

We may also explore how the cluster operators commute with the Hamiltonian. Once again terms with $\hat{T}_n$ on the left side are zero, and we are left with

$$\left[\hat{H}_N,\hat{T}_n\right] = \left\{\overbracket{\hat{H}_N\hat{T}_n}\right\} - \left\{\overbracket{\hat{T}_n\hat{H}_N}\right\} = \left\{\overbracket{\hat{H}_N\hat{T}_n}\right\}. \tag{5.17}$$

Applying this recursively to write out (5.12) it becomes clear that all surviving terms have $\hat{H}_N$ to the left, and all the cluster operators should have at least one contraction each to the Hamiltonian. These terms are referred to as connected terms (subscript C), and because the electronic Hamiltonian includes no more than four operators it would not be possible to find connected terms with more than four cluster operators, posing a natural truncation to $\bar{H}$,

$$\bar{H} = \left(\hat{H}_N e^{\hat{T}}\right)_C = \hat{H}_N + \left(\hat{H}_N\hat{T}\right)_C + \frac{1}{2}\left(\hat{H}_N\hat{T}^2\right)_C + \frac{1}{3!}\left(\hat{H}_N\hat{T}^3\right)_C + \frac{1}{4!}\left(\hat{H}_N\hat{T}^4\right)_C. \tag{5.18}$$

5.2.1 Diagrammatic rules

It is now possible to write out all possible terms and evaluate them diagrammatically. Writing out each term as a diagram, a number of rules exist on how to interpret them, to find the correct algebraic expressions:

1. Sum over all *free* indices. Free indices are not connected to the bra determinant.
2. Interpret one-body operators as $\langle out|\hat{f}|in\rangle \equiv f_{out,in}$, and two-body operators as $\langle lout, rout||lin, rin\rangle$.
3. Add a phase factor of $(-1)^{l+h}$, where l is the number of loops and h is the number of hole lines.
4. Multiply by a factor of $\frac{1}{2}$ for each pair of *equivalent lines*. Equivalent lines are starting and ending at the same interaction lines.
5. Each pair of *equivalent vertices* raise an additional factor of $\frac{1}{2}$. Equivalent vertices are vertices of equal type connected to the same interaction lines with equivalent connecting lines.

6. Each external, unique pair of holes or particles not connected to the same interaction line leads to an antisymmetric permutation $\hat{P}_{l_1 l_2}$.

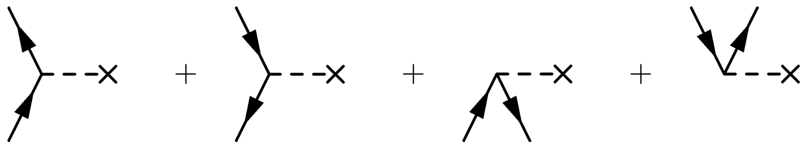
7. If there are multiple ways to connect the diagrams, only one term should exist for each configuration in the Sign-Table technique.

The different rules will be discussed in more detail where needed.

It is possible to split up the sums in the interactions, specifying whether the indices are within the fermi level or not. For the one-body part there exist four possibilities,

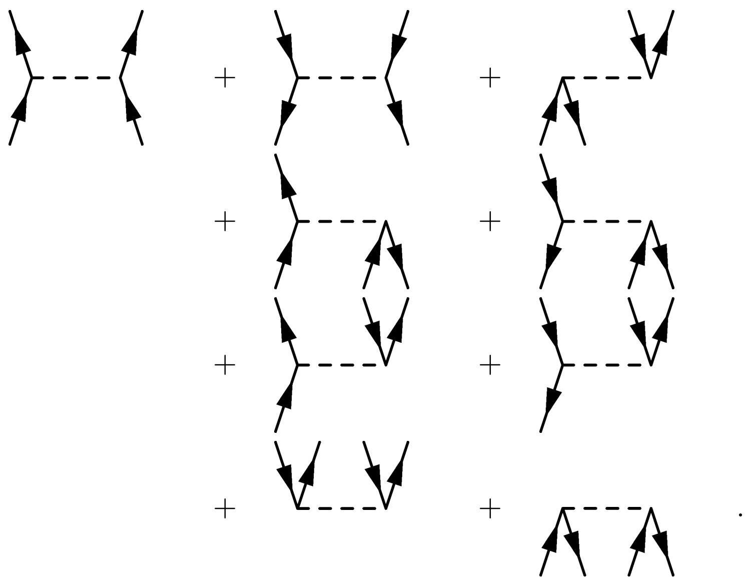
$$\hat{F}_N = \sum_{de} f_{de}\left\{\hat{d}^\dagger \hat{e}\right\} + \sum_{lm} f_{lm}\left\{\hat{l}^\dagger \hat{m}\right\} + \sum_{ld} f_{ld}\left\{\hat{l}^\dagger \hat{d}\right\} + \sum_{dl} f_{dl}\left\{\hat{d}^\dagger \hat{l}\right\}, \tag{5.19}$$

diagrammatically represented as,

$\hat{F}_N =$  (5.20)

The first two terms have the same amount of lines at the top as in the bottom, and we say that these terms have an excitation level zero. Such terms have no possibility to neither create nor annihilate a particle-hole pair. If there is an incoming 2p-2h excitation in a diagram, there will also be an outgoing 2p-2h excitation. The third term is said to have excitation level minus one. When connected in a diagram it will destroy one particle-hole pair, transforming for example an incoming 2p-2h excitation into an outgoing 1p-1h excitation. Following this reasoning, the last term has excitation level plus one.

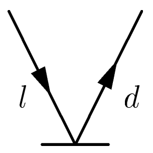
Splitting the two-body operator similarly we get

$\hat{V}_N =$  (5.21)

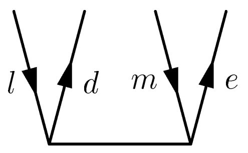
The excitation levels are 0 for the first line, -1 for the second, $+1$ for the third, and the last two terms have $+2$ and -2 respectively.

Cluster operators are represented with a solid horizontal line for the amplitude as well as electron-lines for the creation and annihilation operators, i.e.

$$\hat{T}_1 = \sum_{dl} t_l^d \hat{d}^\dagger \hat{l} =$$



$$\hat{T}_2 = \sum_{delm} t_{lm}^{de} \hat{d}^\dagger \hat{e}^\dagger \hat{m} \hat{l} = \qquad . \tag{5.22}$$



An n-body cluster operator creates an np-nh excitation, and has thus an excitation level $+n$.

5.2.2 The energy equations

The coupled cluster energy $\Delta E_{CCSD} = E_0 - E_{ref}$ is defined as

$$\langle \Phi_0 | \bar{H} | \Phi_0 \rangle = \Delta E_{CCSD}. \tag{5.23}$$

There are only three terms from $\bar{H}$ that contribute to the energy equations [9],

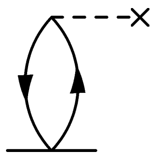
$$\bar{H} \rightarrow \left(\hat{F}_N \hat{T}_1 + \hat{V}_N \hat{T}_2 + \frac{1}{2} \hat{V}_N \hat{T}_1^2 \right)_C , \tag{5.24}$$

and we sort the terms whether they are connected to $\hat{F}_N$ or $\hat{V}_N$.

Contributions from $\hat{\mathbf{F}}_\mathbf{N}$:

We will first try to find all terms in eq. (5.24) with cluster operators connected to $\hat{F}_N$. No lines can be left unconnected, since both the bra and the ket are reference states. Only the third term in $\hat{F}_N$ can obey this, having no lines pointing upwards and an excitation level of -1. To end up with an excitation level of 0 we need to connect it with T_1 which has a $+1$ excitation level. Since no electron lines are connected to the bra, they are summed freely over, labelling the particle d, and the hole l (rule 1). We have one loop, and one hole line leading to a phase factor $(-1)^{1+1} = +1$ (rule 3). The interaction has an incoming particle line d, and an outgoing hole line l, resulting in $f_{out,in} \rightarrow f_{ld}$ (rule 2). Setting the correct indices for the amplitude as well, this term yields

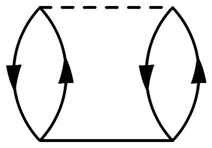
$$\hat{F}_N \hat{T}_1 \rightarrow \qquad = + \sum_{ld} f_{ld} t_l^d, \tag{5.25}$$



and is the only contribution from $\hat{F}_N$.

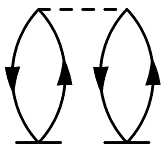
Contributions from $\hat{\mathbf{V}}_{\mathbf{N}}$:

We need to use the last term in $\hat{V}_N$ since all the other terms would leave uncontracted lines pointing upwards, defying the concept of a reference bra state. Having an excitation level of -2 we need to connect it to amplitudes with a $+2$ excitation level. There are only two possible ways to do this; $\hat{V}_N\hat{T}_2$ and $\hat{V}_N\hat{T}_1^2$. The first is

$$\hat{V}_N\hat{T}_2 \rightarrow$$

$$= +\frac{1}{4}\sum_{lmde}\langle lm||de\rangle t_{lm}^{de}, \tag{5.26}$$

a term with two hole lines as well as two loops and thus a positive phase. All indices are freely summed, but since both the pair of particle lines and the pair of hole lines starts and stops and the same interaction, they are equivalent, and should be multiplied by $\left(\frac{1}{2}\right)^2 = \frac{1}{4}$ (rule 4).

The last term in the energy has also two hole lines and two loops, but, since there are two $\hat{T}_1$ operators, the particle and hole lines are no longer equivalent. These are two equivalent vertices instead, raising a factor $\frac{1}{2}$ (rule 5) and resulting in

$$\frac{1}{2}\hat{V}_N\hat{T}_1^2 \rightarrow$$

$$= +\frac{1}{2}\sum_{lmde}\langle lm||de\rangle t_l^d t_m^e. \tag{5.27}$$

Adding these terms together we end up with the complete energy equation

$$\Delta E_{CCSD} = \sum_{ld} f_{ld}t_l^d + \frac{1}{4}\sum_{lmde}\langle lm||de\rangle t_{lm}^{de} + \frac{1}{2}\sum_{lmde}\langle lm||de\rangle t_l^d t_m^e, \tag{5.28}$$

giving us the correction to the energy as compared to E_{ref} for a given set of amplitudes t_i^a and t_{ij}^{ab}.

5.2.3 The $\hat{T}_1$ equations

The amplitudes are determined by the amplitude equations. We start with the $\hat{T}_1$ equations derived from

$$\langle\Phi_i^a|\bar{H}|\Phi_0\rangle = 0. \tag{5.29}$$

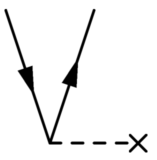
Because of the singly excited bra determinant we will need a total excitation level of $+1$. The contributing parts from $\bar{H}$ are [9],

$$\bar{H} \rightarrow \left[\hat{H}_N + \hat{H}_N\left(\hat{T}_1 + \hat{T}_2\right) + \hat{H}_N\left(\frac{1}{2}\hat{T}_1^2 + \hat{T}_1\hat{T}_2\right) + \hat{H}_N\frac{1}{3!}\hat{T}_1^3\right]_C, \tag{5.30}$$

here presented in the same order as we will follow.

Contributions from $\hat{\mathbf{H}}_{\mathbf{N}}$:

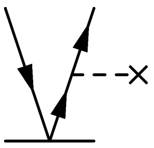
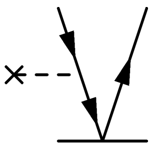
With a reference ket at the bottom and expecting a singly excited determinant bra at the top, we need a term with excitation level +1 and no electron lines pointing downwards. There is only one appropriate term for this,

$\hat{F}_N \rightarrow$  $= +f_{ai}$. (5.31)

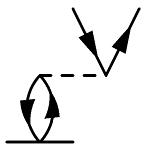
Both electron lines are connected to the singly excited bra state, labeled i, a, and thus not summed freely over. With one hole line and one loop (particle-hole excitations are 'connected' in the bra determinant) the factor in front is +1.

Contributions from $\hat{\mathbf{H}}_{\mathbf{N}}\hat{\mathbf{T}}$:

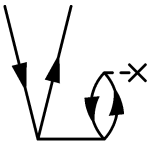
Connecting terms from $\hat{H}_N$, first with the $\hat{T}_1$ cluster operators, the total excitation level needs to be +1, already supplied by the singly excited cluster operator. We then require interaction terms that neither create nor destroy particle-hole excitations. Three possible interaction terms fit in: two from $\hat{F}_N$,

$\hat{F}_N\hat{T}_1 \rightarrow$  $+$  $= +\sum_d f_{ad}t_i^d - \sum_l f_{li}t_l^a$, (5.32)

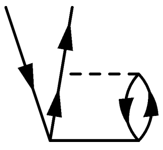
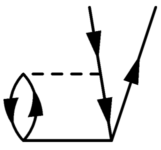
and one from $\hat{V}_N$,

$\hat{V}_N\hat{T}_1 \rightarrow$  $= +\sum_{ld} \langle la||di\rangle t_l^d$. (5.33)

There are also three terms where the interactions are connected to $\hat{T}_2$, requiring the interactions to have the ability to annihilate one particle-hole excitation. One term connects to $\hat{F}_N$,

$\hat{F}_N\hat{T}_2 \rightarrow$  $= +\sum_{ld} f_{ld}t_{il}^{ad}$, (5.34)

and the other two to $\hat{V}_N$,

$\hat{V}_N\hat{T}_2 \rightarrow$  $+$  (5.35)

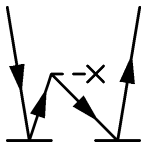
$$= +\frac{1}{2}\sum_{lde} \langle al||de\rangle t_{il}^{de} - \frac{1}{2}\sum_{lmd} \langle lm||di\rangle t_{lm}^{da}.$$

Contributions from $\mathbf{\hat{H}_N\hat{T}^2}$:

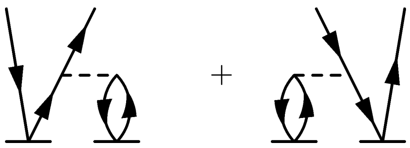
The cluster operator to second order has three terms,

$$\frac{1}{2}\hat{H}_N\left(\hat{T}_1+\hat{T}_2\right)^2 \rightarrow \frac{1}{2}\hat{H}_N\hat{T}_1^2+\hat{H}_N\hat{T}_1\hat{T}_2+\frac{1}{2}\hat{H}_N\hat{T}_2^2, \tag{5.36}$$

and we may first note how no contraction between $\hat{T}_2^2$ and any interaction term can be made without resulting in at least a doubly excited bra state. This leaves us with $\hat{T}_1^2$ and the cross term $\hat{T}_1\hat{T}_2$. For $\hat{T}_1^2$ we can connect the same interaction terms as for $\hat{T}_2$ in eq. (5.34) and (5.35),

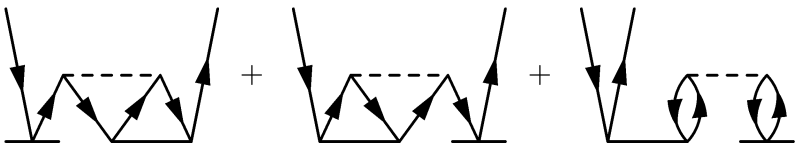
$$\frac{1}{2}\hat{F}_N\hat{T}_1^2 \rightarrow$$

$$= -\sum_{ld} f_{ld}t_i^d t_l^a, \tag{5.37}$$

and

$$\frac{1}{2}\hat{V}_N\hat{T}_1^2 \rightarrow$$


$$= +\sum_{lde}\langle al||de\rangle t_i^d t_l^e - \sum_{lmd}\langle lm||di\rangle t_l^d t_m^a. \tag{5.38}$$

For the cross term, $\hat{T}_1\hat{T}_2$, we need an operator with excitation level -2. Only the last $\hat{V}_N$ term has this level, but it can be connected in three distinct ways,

$$\hat{V}_N\hat{T}_1\hat{T}_2 \rightarrow$$


$$=\frac{1}{2}\sum_{lmde}\langle lm||de\rangle t_i^d t_{lm}^{ea} + \frac{1}{2}\sum_{lmde}\langle lm||de\rangle t_m^a t_{il}^{de} + \sum_{lmde}\langle lm||de\rangle t_m^e t_{il}^{ad}. \tag{5.39}$$

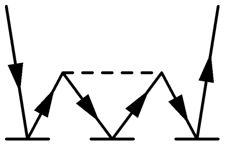
In order to find all unique ways of connecting the operators together, the sign-table technique is applied (rule 7). Denoting a plus sign for all particle lines in the interaction that are connectable to amplitudes (below the interaction line), and a minus sign for all connectable hole lines, we set up a table for what lines are connected to which amplitude. The term from $\hat{V}_N$ used in eq. (5.39) has two particle and two hole lines below the interaction line, represented by a string of four signs, $++--$. A sign table consists of one column for each of the cluster operators, and distinct terms exist for each unique way the signs are distributed. The three distinct terms from equation (5.39) are represented in table 5.1 in the same order as the terms appear in the equation. Adding more than one $+$ or more than one $-$ to $\hat{T}_1$ would be impossible here, due to its one-particle one-hole nature, and leaving $\hat{T}_1$ empty would lead to an uncontracted term. For this reason we get only the three terms seen.

Table 5.1: Sign table for the three terms in eq. (5.39).

$\hat{T}_1$	$\hat{T}_2$
$+$	$+--$
$-$	$++-$
$+-$	$+-$

Contributions from $\mathbf{\hat{H}_N\hat{T}^3}$:

No term in the normal-ordered Hamiltonian can annihilate more than a 2p-2h excitation, and since the bra determinant is singly excited, at most a triple excitation can arise from the cluster operators. The last possible term is thus connected to $\hat{T}_1^3$,

$$\frac{1}{3!}\hat{V}_N\hat{T}_1^3 \rightarrow$$

$$= +\sum_{lmde}\langle lm||de\rangle t_i^d t_l^e t_m^a. \tag{5.40}$$

Adding all terms together we get the complete $\hat{T}_1$ amplitude equations;

$$\begin{aligned}
0 =& f_{ai} + \sum_d f_{ad}t_i^d - \sum_l f_{li}t_l^a + \sum_{ld}\langle la||di\rangle t_l^d \\
&+ \sum_{ld} f_{ld}t_{il}^{ad} + \frac{1}{2}\sum_{lde}\langle al||de\rangle t_{il}^{de} - \frac{1}{2}\sum_{lmd}\langle lm||di\rangle t_{lm}^{da} - \sum_{ld} f_{ld}t_i^d t_l^a \\
&+ \sum_{lde}\langle al||de\rangle t_i^d t_l^e - \sum_{lmd}\langle lm||di\rangle t_l^d t_m^a + \frac{1}{2}\sum_{lmde}\langle lm||de\rangle t_i^d t_{lm}^{ea} \\
&+ \frac{1}{2}\sum_{lmde}\langle lm||de\rangle t_m^a t_{il}^{de} + \sum_{lmde}\langle lm||de\rangle t_m^e t_{il}^{ad} + \sum_{lmde}\langle lm||de\rangle t_i^d t_l^e t_m^a.
\end{aligned} \tag{5.41}$$

5.2.4 The $\hat{T}_2$ equations

We continue by finding algebraic expressions for the $\hat{T}_2$ equations,

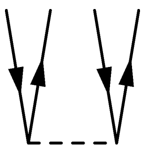
$$\langle\Phi_{ij}^{ab}|\bar{H}|\Phi_0\rangle = 0, \tag{5.42}$$

utilizing the same techniques as before, also introducing permutatation of external lines in eq (5.45). Now requiring a doubly excited particle-hole pair to accommodate the bra determinant, the possible contributions are [9],

$$\begin{aligned}
\bar{H} \rightarrow \Bigg[&\hat{H}_N + \hat{H}_N\left(\hat{T}_1 + \hat{T}_2\right) + \hat{H}_N\left(\frac{1}{2}\hat{T}_1^2 + \hat{T}_1\hat{T}_2 + \frac{1}{2}\hat{T}_2^2\right) \\
&+\hat{H}_N\left(\frac{1}{3!}\hat{T}_1^3 + \frac{1}{2}\hat{T}_1^2\hat{T}_2\right) + \hat{H}_N\frac{1}{4!}\hat{T}_1^4\Bigg]_C
\end{aligned} \tag{5.43}$$

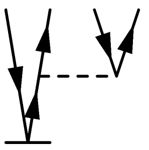
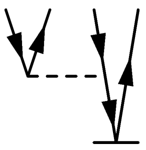
Contributions from $\hat{\mathbf{H}}_{\mathbf{N}}$:

The only interaction term to have a double particle-hole excitation comes from $\hat{V}_N$,

$$\hat{V}_N \rightarrow$$

$$= \langle ab||ij\rangle. \quad (5.44)$$

Contributions from $\hat{\mathbf{H}}_{\mathbf{N}}\hat{\mathbf{T}}$:

A consequence of having more than one pair of external particle and hole lines from the bra determinant is that two different particle-, or hole-lines may be connected to different operators. In the following two terms we have labeled the external lines from left to right, i, a, j, b,

$$\hat{V}_N\hat{T}_1 \rightarrow$$

$$+$$

$$(5.45)$$

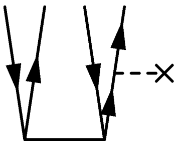
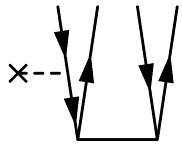
$$= \hat{P}_{ij}\sum_d \langle ab||dj\rangle t_i^d - \hat{P}_{ab}\sum_l \langle al||ij\rangle t_l^b.$$

The first term has i connected to $\hat{T}_1$, whereas j is connected to $\hat{V}_N$, and the second term has a connected to $\hat{V}_N$, whereas b is connected to $\hat{T}_1$. Whenever two such lines are connected to the same interaction a permutation is implied through the antisymmetric nature of the elements, i.e.

$$\langle pq||rs\rangle = -\langle pq||sr\rangle = \cdots \quad \text{or} \quad t_{ij}^{ab} = -t_{ji}^{ab} = \cdots \ . \quad (5.46)$$

In the cases where such a permutation is not implied, we need to enforce a permutation, as in equation (5.45).

No permutations arose in the $\hat{T}_1$ equations due to the fact that only one particle and one hole line were connected upwards to the singly excited determinant. Now, in the $\hat{T}_2$ equations, we have a doubly excited determinant, with the consequence of more terms requiring permutations, as seen also in the terms from $\hat{H}_N\hat{T}_2$:

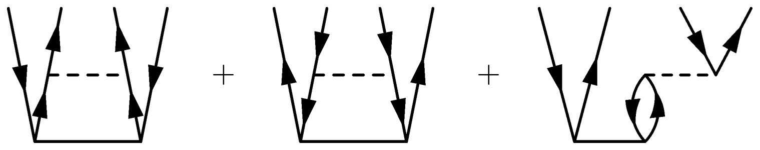
$$\hat{F}_N\hat{T}_2 \rightarrow$$

$$+$$

$$(5.47)$$

$$= \hat{P}_{ab}\sum_d f_{bd}t_{ij}^{ad} - \hat{P}_{ij}\sum_l f_{li}t_{lj}^{ab},$$

Table 5.2: Sign table for the four terms of eq. (5.50).

$\hat{T}_2$	$\hat{T}_2$
$+-$	$+-$
$+$	$+--$
$-$	$++-$
$++$	$--$

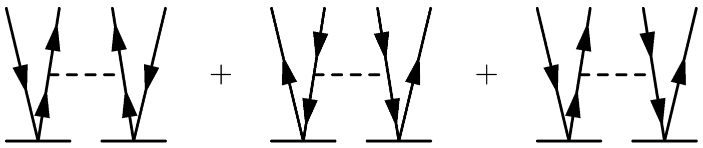
and

$\hat{V}_N\hat{T}_2 \rightarrow$ 

$$=\frac{1}{2}\sum_{de}\langle ab||de\rangle t_{ij}^{de} + \frac{1}{2}\sum_{lm}\langle lm||ij\rangle t_{lm}^{ab} + \hat{P}_{ij}\hat{P}_{ab}\sum_{ld}\langle lb||dj\rangle t_{il}^{ad}. \tag{5.48}$$

Contributions from $\mathbf{\hat{H}_N\hat{T}^2}$:

For $\hat{H}_N\hat{T}_1^2$ three terms meet the requirements of excitation levels,

$\frac{1}{2}\hat{V}_N\hat{T}_1^2 \rightarrow$ 

$$=\hat{P}_{ij}\frac{1}{2}\sum_{de}\langle ab||de\rangle t_i^d t_j^e + \hat{P}_{ab}\frac{1}{2}\sum_{lm}\langle lm||ij\rangle t \tag{5.49}$$

With an equivalent pair of particle lines, an equivalent pair of hole lines, and no equivalent lines respectively, the factors of $\frac{1}{2}$, $\frac{1}{2}$ and 1 arise.

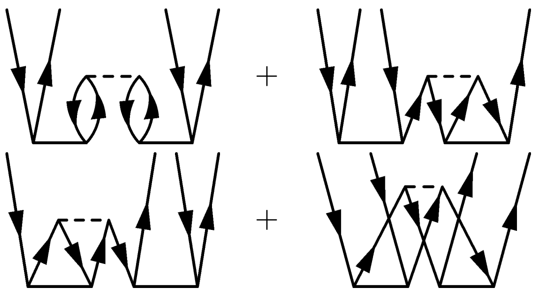
Only one term from the interaction is possible to match with $\hat{T}_2^2$, due to the need for a doubly de-excitated interaction operator. Although only one term fits, it can be

Table 5.3: Sign table for eq. (5.51)

$\hat{T}_1$	$\hat{T}_2$
$+$	$-$
$-$	$+$

connected in four different ways, found by the sign table in table 5.2,

$\frac{1}{2}\hat{V}_N\hat{T}_2^2 \rightarrow$



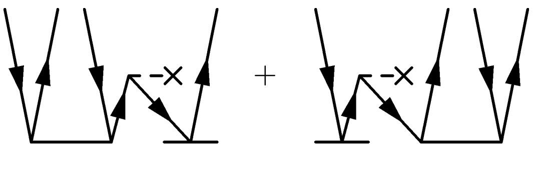
(5.50)

$$
\begin{aligned}
=&\hat{P}_{ij}\hat{P}_{ab}\frac{1}{2}\sum_{lmde}\langle lm||de\rangle t_{il}^{ad}t_{mj}^{eb} + \hat{P}_{ab}\frac{1}{2}\sum_{lmde}\langle lm||de\rangle t_{ij}^{ad}t_{lm}^{eb} \\
&+\hat{P}_{ij}\frac{1}{2}\sum_{lmde}\langle lm||de\rangle t_{il}^{de}t_{mj}^{ab} + \frac{1}{4}\sum_{lmde}\langle lm||de\rangle t_{ij}^{de}t_{lm}^{ab}.
\end{aligned}
$$

One may note how the two $\hat{T}_2$ operators are counted as equivalent, omitting configurations that already exist if we were to switch place between the two operators, i.e. $++\,|\,--$ is the same as $--\,|\,++$, thus counted only once.

The cross term $\hat{T}_1\hat{T}_2$ from $\hat{T}^2$ demands for a singly de-excited interaction operator, found in both $\hat{F}_N$ and $\hat{V}_N$. For the one-particle interaction we have two possible configurations,

$\hat{F}_N\hat{T}_1\hat{T}_2 \rightarrow$



(5.51)

$$
= -\hat{P}_{ab}\sum_{ld} f_{ld}t_{ij}^{ad}t_l^b - \hat{P}_{ij}\sum_{ld} f_{ld}t_i^d t_{lj}^{ab},
$$

as seen from table 5.3. When it comes to the part connected to $\hat{V}_N$, there are two interactions to use, one having two particle lines and one hole line connectable with the cluster operators, another having two hole lines and one particle line to be connected with the cluster operators. In table 5.4, this provides the two combinations $++-$ and $--+$,

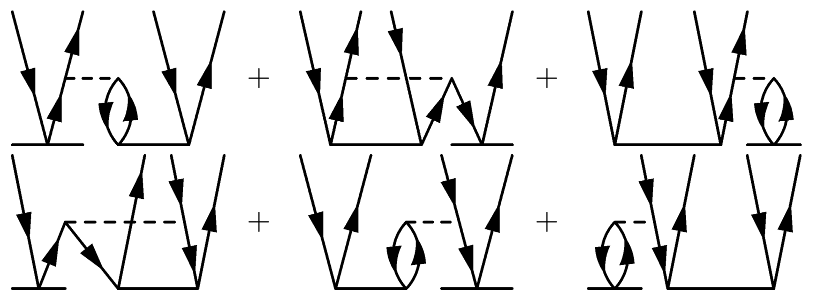
Table 5.4: Sign tables for eq. (5.52)

$\hat{T}_1$	$\hat{T}_2$
$+$	$+-$
$-$	$++$
$+-$	$+$

$\hat{T}_1$	$\hat{T}_2$
$+$	$--$
$-$	$+-$
$+-$	$-$

in all responsible for six unique configurations,

$\hat{V}_N\hat{T}_1\hat{T}_2 \rightarrow$



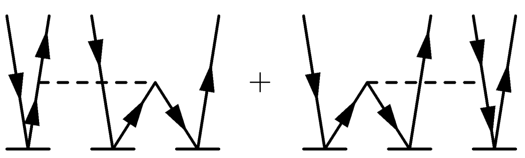
(5.52)

$$= \hat{P}_{ij}\hat{P}_{ab}\sum_{lde}\langle al||de\rangle t_i^d t_{lj}^{eb} - \hat{P}_{ab}\frac{1}{2}\sum_{lde}\langle al||de\rangle t_{ij}^{de} t_l^b + \hat{P}_{ab}\sum_{lde}\langle bl||de\rangle t_{ij}^{ad} t_l^e$$
$$+\hat{P}_{ij}\frac{1}{2}\sum_{lmd}\langle lm||dj\rangle t_i^d t_{lm}^{ab} - \hat{P}_{ij}\hat{P}_{ab}\sum_{lmd}\langle lm||dj\rangle t_{il}^{ad} t_m^b - \hat{P}_{ij}\sum_{lmd}\langle ml||di\rangle t_m^d t_{lj}^{ab}.$$

Contributions from $\hat{\mathbf{H}}_{\mathbf{N}}\hat{\mathbf{T}}^{\mathbf{3}}$:

In the case of the $\hat{T}_2$ equations, terms with higher order of cluster operators come forth, as evident for $\hat{T}^3$. As high as quadruple excitations can arise in the cluster operators and still be contracted with $\hat{V}_N$ to create a doubly excited determinant. Two terms from the two-particle interaction can ensure the correct excitation levels, and at the same time have one contraction to each of the operators in $\hat{T}_1^3$,

$\frac{1}{3!}\hat{V}_N\hat{T}_1^3 \rightarrow$



(5.53)

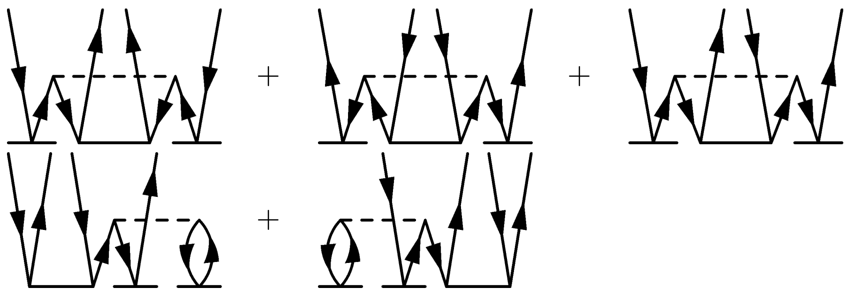
$$= -\hat{P}_{ij}\hat{P}_{ab}\frac{1}{2}\sum_{lde}\langle al||de\rangle t_i^d t_j^e t_l^b + \hat{P}_{ij}\hat{P}_{ab}\frac{1}{2}\sum_{lmd}\langle lm||dj\rangle t_i^d t_l^a t_m^b.$$

The most complicated sign table to be encountered is for the terms contracting $\hat{V}_N$ with $\hat{T}_1^2\hat{T}_2$. Five terms are found,

Table 5.5: Sign table for eq. (5.54).

$\hat{T}_1$	$\hat{T}_1$	$\hat{T}_2$
$+$	$+$	$--$
$-$	$-$	$++$
$+$	$-$	$+-$
$+-$	$-$	$+$
$+-$	$+$	$-$

$$\frac{1}{2}\hat{V}_N\hat{T}_1^2\hat{T}_2 \rightarrow$$



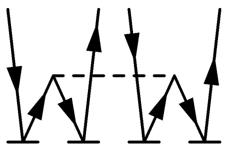
$$\begin{aligned}&= \hat{P}_{ij}\frac{1}{4}\sum_{lmde}\langle lm||de\rangle t_i^d t_j^e t_{lm}^{ab} + \hat{P}_{ab}\frac{1}{4}\sum_{lmde}\langle lm||de\rangle t_l^a t_m^b t_{ij}^{de} + \hat{P}_{ij}\hat{P}_{ab}\sum_{lmde}\langle lm||de\rangle t_i^d t_m^b t_{lj}^{ae} \\ &\quad -\hat{P}_{ab}\sum_{lmde}\langle lm||de\rangle t_m^e t_l^b t_{ij}^{ad} - \hat{P}_{ij}\sum_{lmde}\langle lm||de\rangle t_l^d t_i^e t_{mj}^{ab},\end{aligned} \tag{5.54}$$

one for each of the configurations in table 5.5.

Contributions from $\hat{\mathbf{H}}_\mathbf{N}\hat{\mathbf{T}}^4$:

The last term includes four cluster operators, all of which need to be a $\hat{T}_1$ operator to satisfy the needed excitation level. In order to be a connected diagram, all four cluster operators should be attached with at least one contraction each to $\hat{V}_N$. This is achieved in only one way due to the interaction containing exactly four operators,

$$\frac{1}{4!}\hat{V}_N\hat{T}_1^4 \rightarrow$$



$$= \hat{P}_{ab}\hat{P}_{ij}\frac{1}{4}\sum_{lmde}\langle lm||de\rangle t_i^d t_l^a t_j^e t_m^b. \tag{5.55}$$

5.3 Implementing CCSD

We revisit the $\hat{T}_1$ equations, (5.41). It is possible to relabel free indices, as long as they still sum over the same region, either hole states or particle states. Doing such a re-indexing, and at the same time factor out similar terms, we get

$$\begin{aligned}0 =& f_{ai} + \sum_{ld}\langle la||di\rangle t_l^d + \frac{1}{2}\sum_{lde}\langle al||de\rangle t_{il}^{de} + \sum_{d}\left[f_{ad} + \sum_{le}\langle al||de\rangle t_l^e\right] t_i^d \\ &- \sum_{l}\left[f_{li} + \sum_{d} f_{ld}t_i^d + \sum_{md}\langle ml||di\rangle t_m^d + \sum_{mde}\langle lm||de\rangle t_i^d t_m^e + \frac{1}{2}\sum_{mde}\langle lm||de\rangle t_{im}^{de}\right] t_l^a \\ &+ \frac{1}{2}\sum_{lmd}\left[\langle lm||id\rangle + \sum_{e}\langle lm||ed\rangle t_i^e\right] t_{lm}^{da} + \sum_{ld}\left[\sum_{me}\langle lm||de\rangle t_m^e + f_{ld}\right] t_{il}^{ad}.\end{aligned} \tag{5.56}$$

The parentheses can be defined as intermediates to be calculated and stored before we solve the complete equations. The first four parentheses are now intermediates one to four;

$$[\mathcal{I}_1]_{ad} = f_{ad} + \sum_{le}\langle al||de\rangle t_l^e, \tag{5.57}$$

$$[\mathcal{I}_2]_{ld} = f_{ld} + \sum_{me}\langle lm||de\rangle t_m^e, \tag{5.58}$$

$$[\mathcal{I}_3]_{li} = f_{li} + \sum_{md}\langle ml||di\rangle t_m^d + \frac{1}{2}\sum_{mde}\langle lm||de\rangle t_{im}^{de} + \sum_{d}[\mathcal{I}_2]_{ld}\, t_i^d, \tag{5.59}$$

$$\begin{aligned}[\mathcal{I}_4]_{id}^{lm} &= \langle lm||id\rangle + \sum_{e}\langle lm||ed\rangle t_i^e \\ &= [\mathcal{I}_5]_{id}^{lm} + \sum_{e}\frac{1}{2}\langle lm||ed\rangle t_i^e,\end{aligned} \tag{5.60}$$

where the last intermediate depends on

$$[\mathcal{I}_5]_{id}^{lm} = \langle lm||id\rangle + \sum_{e}\frac{1}{2}\langle lm||ed\rangle t_i^e. \tag{5.61}$$

Applying such a simplification the $\hat{T}_1$ amplitude equations reduce to

$$\begin{aligned}0 =& f_{ai} + \sum_{ld}\langle la||di\rangle t_l^d + \frac{1}{2}\sum_{lde}\langle al||de\rangle t_{il}^{de} + \sum_{d}[\mathcal{I}_1]_{ad}\, t_i^d \\ &- \sum_{l}[\mathcal{I}_3]_{li}\, t_l^a + \frac{1}{2}\sum_{lmd}[\mathcal{I}_4]_{id}^{lm}\, t_{lm}^{da} + \sum_{ld}[\mathcal{I}_2]_{ld}\, t_{il}^{ad}.\end{aligned} \tag{5.62}$$

We continue, repeating this procedure of simplification, with the $\hat{T}_2$ equations, relabelling indices and factoring out common terms,

$$\begin{aligned}
0 =&\langle ab||ij\rangle + \frac{1}{2}\langle ab||de\rangle t_{ij}^{de} \\
&- \hat{P}_{ij}\left[f_{li} + f_{ld}t_i^d + \langle ml||di\rangle t_m^d + \langle ml||de\rangle t_m^d t_i^e + \frac{1}{2}\langle ml||de\rangle t_{mi}^{de}\right] t_{lj}^{ab} \\
&+ \frac{1}{2}\left[\langle lm||ij\rangle + \hat{P}_{ij}\langle lm||dj\rangle t_i^d + \frac{1}{2}\langle lm||de\rangle t_{ij}^{de} + \hat{P}_{ij}\frac{1}{2}\langle lm||de\rangle t_i^d t_j^e\right] t_{lm}^{ab} \\
&+ \hat{P}_{ab}\left[f_{bd} - f_{ld}t_l^b + \langle bl||de\rangle t_l^e - \langle lm||de\rangle t_m^e t_l^b + \frac{1}{2}\langle lm||de\rangle t_{lm}^{eb}\right] t_{ij}^{ad} \\
&+ \hat{P}_{ij}\hat{P}_{ab}\left[\langle lb||dj\rangle - \langle lm||dj\rangle t_m^b + \langle bl||ed\rangle t_j^e - \langle lm||de\rangle t_j^e t_m^b + \frac{1}{2}\langle lm||de\rangle t_{mj}^{eb}\right] t_{il}^{ad} \\
&- \hat{P}_{ab}\left[\langle al||ij\rangle + \frac{1}{2}\langle al||de\rangle t_{ij}^{de} + \hat{P}_{ij}\langle al||dj\rangle t_i^d + \hat{P}_{ij}\frac{1}{2}\langle al||de\rangle t_i^d t_j^e\right. \\
&\left.+\frac{1}{2}\langle lm||ij\rangle t_m^a + \frac{1}{4}\langle lm||de\rangle t_m^a t_{ij}^{de} + \hat{P}_{ij}\frac{1}{2}\langle lm||dj\rangle t_i^d t_m^a + \hat{P}_{ij}\frac{1}{4}\langle lm||de\rangle t_i^d t_j^e t_m^a\right] t_l^b \\
&+ \hat{P}_{ij}\left[\langle ab||dj\rangle + \frac{1}{2}\langle ab||de\rangle t_j^e\right] t_i^d .
\end{aligned} \tag{5.63}$$

Once again we define intermediates, still having the five intermediates from $\hat{T}_1$ in memory. The first parenthesis from $\hat{T}_2$ is already defined in $[\mathcal{I}_3]_{li}$. The following three parentheses are $[\mathcal{I}_6]$,$[\mathcal{I}_7]$ and $[\mathcal{I}_8]$ respectively;

$$\begin{aligned}
[\mathcal{I}_6]_{ij}^{lm} =&\langle lm||ij\rangle + \hat{P}_{ij}\langle lm||dj\rangle t_i^d + \frac{1}{2}\langle lm||de\rangle t_{ij}^{de} + \hat{P}_{ij}\frac{1}{2}\langle lm||de\rangle t_i^d t_j^e \\
=&\langle lm||ij\rangle + \frac{1}{2}\langle lm||de\rangle t_{ij}^{de} - \hat{P}_{ij}\left(\langle lm||jd\rangle t_i^d + \frac{1}{2}\langle lm||ed\rangle t_i^d t_j^e\right) \\
=&\langle lm||ij\rangle + \frac{1}{2}\langle lm||de\rangle t_{ij}^{de} - \hat{P}_{ij}\,[\mathcal{I}_5]_{jd}^{lm}\, t_i^d,
\end{aligned} \tag{5.64}$$

$$\begin{aligned}
[\mathcal{I}_7]_{bd} =&f_{bd} - f_{ld}t_l^b + \langle bl||de\rangle t_l^e - \langle lm||de\rangle t_m^e t_l^b + \frac{1}{2}\langle lm||de\rangle t_{lm}^{eb} \\
=&\,[\mathcal{I}_1]_{bd} - [\mathcal{I}_2]_{ld}\, t_l^b + \frac{1}{2}\langle lm||de\rangle t_{lm}^{eb},
\end{aligned} \tag{5.65}$$

$$\begin{aligned}
[\mathcal{I}_8]_{dj}^{lb} =&\langle lb||dj\rangle - \langle lm||dj\rangle t_m^b + \langle bl||ed\rangle t_j^e - \langle lm||de\rangle t_j^e t_m^b + \frac{1}{2}\langle lm||de\rangle t_{mj}^{eb} \\
=&\,[\mathcal{I}_9]_{dj}^{lb} + \frac{1}{2}\langle bl||ed\rangle t_j^e + [\mathcal{I}_4]_{jd}^{lm}\, t_m^b + \frac{1}{2}\langle lm||de\rangle t_{mj}^{eb}.
\end{aligned} \tag{5.66}$$

A ninth intermediate, $[\mathcal{I}_9]$, is a dependency for $[\mathcal{I}_8]$,

$$[\mathcal{I}_9]_{dj}^{lb} =\langle lb||dj\rangle - \frac{1}{2}\langle lb||ed\rangle t_j^e, \tag{5.67}$$

and the two last parentheses are

$$\begin{aligned}
[\mathcal{I}_{10}]^{al}_{ij} =&\langle al||ij\rangle + \frac{1}{2}\langle al||de\rangle t^{de}_{ij} + \hat{P}_{ij}\langle al||dj\rangle t^d_i + \hat{P}_{ij}\frac{1}{2}\langle al||de\rangle t^d_i t^e_j \\
&+\frac{1}{2}\langle lm||ij\rangle t^a_m + \frac{1}{4}\langle lm||de\rangle t^a_m t^{de}_{ij} + \hat{P}_{ij}\frac{1}{2}\langle lm||dj\rangle t^d_i t^a_m + \hat{P}_{ij}\frac{1}{4}\langle lm||de\rangle t^d_i t^e_j t^a_m \\
=&\langle al||ij\rangle + \frac{1}{2}\langle al||de\rangle t^{de}_{ij} - \hat{P}_{ij}\,[\mathcal{I}_9]^{la}_{dj}\,t^d_i + \frac{1}{2}\,[\mathcal{I}_6]^{lm}_{ij}\,t^a_m,
\end{aligned} \tag{5.68}$$

and

$$[\mathcal{I}_{11}]^{ab}_{dj} = \langle ab||dj\rangle + \frac{1}{2}\langle ab||de\rangle t^e_j. \tag{5.69}$$

In its entirety we obtain simplified $\hat{T}_2$ equations,

$$\begin{aligned}
0 =&\langle ab||ij\rangle + \frac{1}{2}\langle ab||de\rangle t^{de}_{ij} - [\mathcal{I}_3]_{li}\,t^{ab}_{lj} - [\mathcal{I}_3]_{lj}\,t^{ab}_{il} \\
&+\frac{1}{2}\,[\mathcal{I}_6]^{lm}_{ij}\,t^{ab}_{lm} + [\mathcal{I}_7]_{bd}\,t^{ad}_{ij} + [\mathcal{I}_7]_{ad}\,t^{db}_{ij} \\
&+\hat{P}_{ij}\hat{P}_{ab}\,[\mathcal{I}_8]^{lb}_{dj}\,t^{ad}_{il} - \hat{P}_{ab}\,[\mathcal{I}_{10}]^{al}_{ij}\,t^b_l + \hat{P}_{ij}\,[\mathcal{I}_{11}]^{ab}_{dj}\,t^d_i.
\end{aligned} \tag{5.70}$$

It is in principle impossible to find closed-form solutions for equation (5.62) and (5.70), due to their non-linear behavior as well as the large dimensionality. Solutions can often be found anyhow, by employing iterative methods, to numerical precision when converging. We employ a 'Jacobi's method'-like iterative strategy, subtracting the 'diagonal' of the simplest terms containing $\hat{T}_1$-amplitudes from both sides of (5.62),

$$\begin{aligned}
t'^{a}_{i}\,D^a_i =&f_{ai} + \sum_{ld}\langle la||di\rangle t^d_l + \frac{1}{2}\sum_{lde}\langle al||de\rangle t^{de}_{il} + \sum_{d}\,[\mathcal{I}_1]_{ad}\,t^d_i \\
&-\sum_{l}\,[\mathcal{I}_3]_{li}\,t^a_l + \frac{1}{2}\sum_{lmd}\,[\mathcal{I}_4]^{lm}_{id}\,t^{da}_{lm} + \sum_{ld}\,[\mathcal{I}_2]_{ld}\,t^{ad}_{il} + t^a_i D^a_i,
\end{aligned} \tag{5.71}$$

where the negative diagonal is

$$D^a_i = -\,[\mathcal{I}_1]_{aa} + [\mathcal{I}_3]_{ii}\,. \tag{5.72}$$

Starting with a guess for t^a_i, usually zero or based on an earlier converged result, one calculates a new guess t'^{a}_{i}.

The same procedure[1] may equally well be applied to eq. (5.70),

$$\begin{aligned}
t'^{ab}_{ij}\,D^{ab}_{ij} =&\langle ab||ij\rangle + \frac{1}{2}\langle ab||de\rangle t^{de}_{ij} - [\mathcal{I}_3]_{li}\,t^{ab}_{lj} - [\mathcal{I}_3]_{lj}\,t^{ab}_{il} \\
&+\frac{1}{2}\,[\mathcal{I}_6]^{lm}_{ij}\,t^{ab}_{lm} + [\mathcal{I}_7]_{bd}\,t^{ad}_{ij} + [\mathcal{I}_7]_{ad}\,t^{db}_{ij} \\
&+\hat{P}_{ij}\hat{P}_{ab}\,[\mathcal{I}_8]^{lb}_{dj}\,t^{ad}_{il} - \hat{P}_{ab}\,[\mathcal{I}_{10}]^{al}_{ij}\,t'^{b}_{l} + \hat{P}_{ij}\,[\mathcal{I}_{11}]^{ab}_{dj}\,t'^{d}_{i} + t^{ab}_{ij}D^{ab}_{ij},
\end{aligned} \tag{5.73}$$

[1]One may notice the use of the new $\hat{T}_1$ amplitudes, t'^{a}_{i}, being used when iterating the $\hat{T}_2$ equations, a trick that often speeds up the convergence. This trick, however, require us to recalculate all intermediates depending on t^a_i using the new amplitudes t'^{a}_{i} instead.

where the subtracted diagonal terms are

$$D_{ij}^{ab} = -\frac{1}{2}\langle ab||ab\rangle + [\mathcal{I}_3]_{ii} + [\mathcal{I}_3]_{jj} - \frac{1}{2}[\mathcal{I}_6]_{ij}^{ij} - [\mathcal{I}_7]_{bb} - [\mathcal{I}_7]_{aa} \,. \tag{5.74}$$

After one such iteration we have found a new guess for both $\hat{T}_1$ and $\hat{T}_2$ amplitudes

$$\begin{aligned} t_i^a &= {t'}_i^a \,, \\ t_{ij}^{ab} &= {t'}_{ij}^{ab} \,, \end{aligned} \tag{5.75}$$

and we repeat the process until results are converged, typically defined by

$$\begin{aligned} |\,{t'}_i^a - t_i^a| &< \epsilon_1 \\ |\,{t'}_{ij}^{ab} - t_{ij}^{ab}| &< \epsilon_2. \end{aligned} \tag{5.76}$$

In practice it is more convenient to use the single criterion,

$$|\Delta E_{CCSD}(t') - \Delta E_{CCSD}(t)| < \epsilon_E, \tag{5.77}$$

because we now can control the precision of our results through ϵ_E.

Coupled-cluster class – 'CC'

We have implemented the iteration scheme defined in (5.71), (5.73) and (5.77) in a class, called 'CC'. This class should be able to work with any system derived from the 'System' base class, effectively exploiting the sparseness of matrix elements if defined in the system's 'Basis'.

The workhorse in 'CC' is the method 'solve_ground_state_energy()', which contains a loop iterating through the scheme. The implementation in listing 5.1 illustrates the operations needed within each iteration. It is a slightly stripped-down version as the complete implementation includes methods for printing debugging information, timing the different terms, as well as testing for convergence. The motivation to contain the different terms in multiple methods is mainly motivated by the need for a structured code, but at the same time it may ease debugging and profiling. Before starting the iteration scheme, an initial guess for the amplitudes is needed, by default zero as in listing 5.2. An important feature is how all matrices with four indices are effectively stored as block-diagonal structures, of type 'std::vector<arma::mat>'. One matrix is included for each channel, λ, indexed by configurations μ, ν or ξ.

In the notation of channels and configurations some terms are more straight forward to implement than other, as shown by the second term in the $\hat{T}_2$ equations (5.73), here stated in a slightly different notation,

$$\langle ab|t_2'|ij\rangle \leftarrow \frac{1}{2}\sum_{de}\langle ab||de\rangle\langle de|t_2|ij\rangle. \tag{5.78}$$

In terms of the diagonal blocks,

$$\langle \xi|t_2'|\mu\rangle_{(\lambda)} \leftarrow \frac{1}{2}\sum_{\xi'}\langle \xi||\xi'\rangle_{(\lambda)}\langle \xi'|t_2|\mu\rangle_{(\lambda)}, \tag{5.79}$$

Listing 5.1: The main content of the iteration loop in CC::solve_ground_state_energy().

```
//Calculate intermediates for T1
mat i1 = c_i1(t1_old);
mat i2 = c_i2(t1_old);
mat i3 = c_i3(i2, t1_old, t2_old);
vector<mat> i5 = c_i5(t1_old);
vector<mat> i4 = c_i4(i5, t1_old);

//Calculate diagonal D_i^a
mat d1 = c_d1(i1, i3);

//One iteration on T1-equations
mat t1_new = c_t1(i1, i2, i3, i4, d1, t1_old, t2_old);

//Calculate intermediates for T2, usign the new t_i^a values
i1 = c_i1(t1_new);
i2 = c_i2(t1_new);
i3 = c_i3(i2, t1_new, t2_old);
i5 = c_i5(t1_new);
i4 = c_i4(i5, t1_new);
vector<mat> i6 = c_i6(i5, t1_new, t2_old);
mat i7 = c_i7(i1, i2, t1_new, t2_old);
vector<mat> i9 = c_i9(t1_new);
vector<mat> i8 = c_i8(i4, i9, t1_new, t2_old);
vector<mat> i10 = c_i10(i6, i9, t1_new, t2_old);
vector<mat> i11 = c_i11(t1_new);

//Diagonal D_{ij}^{ab}
vector<mat> d2 = c_d2(i3, i6, i7);

//One iteration on T2-equations
vector<mat> t2_new = c_t2(i3, i6, i7, i8, i10, i11, d2,
   t1_new, t2_old);

//Store amplitudes for next iteration
t1_old = t1_new;
t2_old = t2_new;
```

Listing 5.2: An initial guess for the amplitudes is needed before doing iterations. By default all amplitudes are zero, unless another starting point is supplied in 't1_stored' and 't2_stored' at the same time as the switch 'use_t' is set to true. Amplitudes, t_{ij}^{ab}, are stored as one matrix for each channel, λ, $t_{(\lambda)}{}_{\mu}^{\xi}$.

```
mat t1_old; //T1 amplitudes
vector<mat> t2_old; //T2 amplitudes

if (use_t)
{ //Use a supplied initial guess
    t1_old = t1_stored;
    t2_old = t2_stored;

} else
{ //Use the default guess t_ij^ab = 0
    Basis const * basis = sys->get_basis();
    vector<uvec> const * mapPP = basis->get_map_lmdXI_de();
    vector<uvec> const * mapHH = basis->get_map_lmdMU_lm();

    //T1 is a (n_p x n_h) matrix
    t1_old = zeros<mat > (basis->get_nP(), basis->get_nH());

    //T2 has a (n_xi x n_mu) matrix for each channel, lambda.
    for (size_t lmd = 0; lmd < basis->dim_lmd_2p(); lmd++)
    {
        size_t dimXI = mapPP->at(lmd).size();
        size_t dimMU = mapHH->at(lmd).size();
        t2_old.push_back(zeros<mat > (dimXI, dimMU));
    }
}
```

Listing 5.3: Implementation of the second term in the $\hat{T}_2$ equations.

```
for (int lmd = 0; lmd < basis->dim_lmd_2p(); lmd++)
    t2_new.at(lmd) += 0.5 * sys->get_v_pppp()->at(lmd) *
       t2_old.at(lmd);
```

this can easily be recognized as simple matrix multiplications, one for each diagonal block, implemented as shown in listing 5.3.

Unfortunately, not all terms are on a form directly suitable for matrix multiplication. We put forth the second term from intermediate $[\mathcal{I}_{11}]$,

$$[\mathcal{I}_{11}]^{ab}_{dj} \leftarrow \frac{1}{2}\sum_e \langle ab||de\rangle t^e_j, \tag{5.80}$$

or in another notation

$$\langle ab|\mathcal{I}_{11}|dj\rangle \leftarrow \frac{1}{2}\sum_e \langle ab||de\rangle\langle e|t_1|j\rangle, \tag{5.81}$$

as an example. The main problem with this term is the summation over e, whereas elements with four indices are stored using configurations, i.e.

$$\langle \xi_{ab}|\mathcal{I}_{11}|\nu_{dj}\rangle_\lambda, \quad \text{and} \quad \langle \xi_{ab}||\xi_{de}\rangle_\lambda. \tag{5.82}$$

In its initial form the mappings $de \to \xi_{de}$ and $dj \to \nu_{dj}$ are needed for all possible combinations of d, e, j, implying poor performance. If we could reindex this term, making it suitable for efficient matrix multiplication, the number of times mappings are needed would also be reduced. We do this in a simple way,

$$\langle abd^{-1}|\mathcal{I}_{11}|j\rangle_{\lambda_{1p}} \leftarrow \frac{1}{2}\langle abd^{-1}||e\rangle_{\lambda_{1p}}\langle e|t_1|j\rangle_{\lambda_{1p}}, \tag{5.83}$$

where new channels and configurations are declared, redefining symmetries to be

$$\left.\begin{array}{ccccccc} \overbrace{m^a+m^b}^{\lambda} & = & \overbrace{m^d+m^e}^{\lambda} & \to & \overbrace{m^a+m^b-m^d}^{\lambda_{1p}} & = & \overbrace{m^e}^{\lambda_{1p}} \\ m^a_s+m^b_s & = & m^d_s+m^e_s & \to & m^a_s+m^b_s-m^d_s & = & m^e_s \end{array}\right\} \text{ for } \hat{V}_N, \tag{5.84}$$

and

$$\left.\begin{array}{ccccccc} \overbrace{m^a+m^b}^{\lambda} & = & \overbrace{m^d+m^j}^{\lambda} & \to & \overbrace{m^a+m^b-m^d}^{\lambda_{1p}} & = & \overbrace{m^j}^{\lambda_{1p}} \\ m^a_s+m^b_s & = & m^d_s+m^j_s & \to & m^a_s+m^b_s-m^d_s & = & m^j_s \end{array}\right\} \text{ for } \mathcal{I}_{11}. \tag{5.85}$$

The index of only one particle's state determines the new channels, thus labelled λ_{1p}, which are also applicable to the $\hat{T}_1$ amplitudes, t^e_j. All indices are effectively represented by configurations within each one-particle channel, making this programmable as multiplication of block-diagonal matrices. Implementing this we encounter four stages;

1. Map from $\langle \xi_{ab}||\xi_{de}\rangle_\lambda$ into $\langle abd^{-1}||e\rangle_{\lambda_{1p}}$, only required on the first iteration, or earlier, as the interaction matrices do not change.

2. Map $\langle e|t_1|j\rangle$ into $\langle e|t_1|j\rangle_{\lambda_{1p}}$ at each iteration.

3. Perform a block diagonal matrix multiplication,

$$\langle abd^{-1}|\mathcal{I}_{11}|j\rangle_{\lambda_{1p}} = \frac{1}{2}\sum_e \langle abd^{-1}||e\rangle_{\lambda_{1p}}\langle e|t_1|j\rangle_{\lambda_{1p}}. \tag{5.86}$$

4. Add the results back into the intermediate,

$$\langle ab|\mathcal{I}_{11}|dj\rangle + = \langle abd^{-1}|\mathcal{I}_{11}|j\rangle_{\lambda_{1p}}. \tag{5.87}$$

The four stages are illustrated by listings 5.4, 5.5, 5.6 and 5.7.

Listing 5.4: Mapping from $\langle \xi_{ab} || \xi_{de} \rangle_\lambda$ into $\langle abd^{-1} || e \rangle_{\lambda_{1p}}$, only needed once because the matrix elements stay constant during simulations.

```
//Interaction elements <a_b_d1||e>
vector<mat> v_abd1_e; //Block diagonal matrix

//Allocate blocks filled with zeros.
for (int lmd1 = 0; lmd1 < basis->dim_lmd_1p(); lmd1++)
{
    int dimE = map_p.at(lmd1).size();
    int dimABD1 = map_p_p_p1.at(lmd1).size();
    v_abd1_e.push_back(zeros<mat > (dimABD1, dimE));
}

//Fill elements from v_pppp
for (int lmd2 = 0; lmd2 < basis->dim_lmd_2p(); lmd2++)
{
    int dimXI = mapPP->at(lmd2).size();
    for (int xi_ab = 0; xi_ab < dimXI; xi_ab++)
        for (int xi_de = 0; xi_de < dimXI; xi_de++)
        {
            //Map configurations into indices
            int ab = mapPP->at(lmd2)(xi_ab);
            int a = ab % nP;
            int b = ab / nP;
            int de = mapPP->at(lmd2)(xi_de);
            int d = de % nP;
            int e = de / nP;

            //Find the new redefined channels and
               configurations
            int abd = a + (b + d * nP) * nP;
            int abd_idx = map_p_p_p1_inv(1, abd);
            int lmd1 = map_p_inv(0, e);
            int e_idx = map_p_inv(1, e);

            //Insert element
            v_abd1_e.at(lmd1)(abd_idx, e_idx) = sys->
               get_v_pppp()->at(lmd2)(xi_ab, xi_de);
        }
}
```

Listing 5.5: Mapping from $\langle e|t_1|j\rangle$ into $\langle e|t_1|j\rangle_{\lambda_{1p}}$, required at each iteration.

```
//Rewriting t1
vector<mat> t1_e_j;
for (int lmd1 = 0; lmd1 < dimLMD1; lmd1++)
{
    //Allocate a zero-filled matrix for each channel.
    int dimP = map_p.at(lmd1).size();
    int dimH = map_h.at(lmd1).size();
    t1_e_j.push_back(zeros<mat > (dimP, dimH));

    for (int e_idx = 0; e_idx < dimP; e_idx++)
        for (int j_idx = 0; j_idx < dimH; j_idx++)
        {
            //Find indices from configurations
            int e = map_p.at(lmd1)(e_idx);
            int j = map_h.at(lmd1)(j_idx);

            //Insert correct element
            t1_e_j.at(lmd1)(e_idx, j_idx) = t1_old(e, j);
        }
}
```

Listing 5.6: Multiply each diagonal block.

```
//Matrix mult.
vector<mat> i11_abd1_j;
for (int lmd1 = 0; lmd1 < dimLMD1; lmd1++)
    i11_abd1_j.push_back(0.5 * v_abd1_e.at(lmd1) * t1_e_j.at(
       lmd1));
```

Listing 5.7: Terms are added back into $\mathcal{I}_{11}$.

```
//Add terms back into i11
for (int lmd1 = 0; lmd1 < dimLMD1; lmd1++)
{
    int dimABD1 = map_p_p_p1.at(lmd1).size();
    int dimJ = map_h.at(lmd1).size();

    for (int abd_idx = 0; abd_idx < dimABD1; abd_idx++)
        for (int j_idx = 0; j_idx < dimJ; j_idx++)
        {
            //Map lmd1 configurations into indices
            int abd = map_p_p_p1.at(lmd1)(abd_idx);
            int a = abd % nP;
            int bd = abd / nP;
            int b = bd % nP;
            int d = bd / nP;
            int j = map_h.at(lmd1)(j_idx);

            //Map indices into lmd configurations
            int ab = a + b * nP;
            int dj = d + j * nP;
            int lmd2 = (*mapPPinv)(0, ab);
            int xi_ab = (*mapPPinv)(1, ab);
            int nu_dj = (*mapPHinv)(1, dj);

            //Add element
            i11.at(lmd2)(xi_ab, nu_dj) += i11_abd1_j.at(lmd1)
               (abd_idx, j_idx);
        }
}
```

5.4 Hartree-Fock method

The Coupled Cluster method appears as a convenient method, yielding accurate results within reasonable computation time. It depends, however, on an initial guess for the amplitudes sufficiently close to the solution. Without this guess results will converge slowly, or may not even converge at all. The usual starting point is to set all amplitudes to zero, a good guess whenever the solution can be well approximated by a simple reference determinant. If this is not sufficient one should either find a better initial guess for the amplitudes or find a better set of basis functions. A better initial guess can be hard to find, but basis functions can be transformed by a Hartree-Fock calculation.

Hartree-Fock (HF) is another *ab initio* many-body method, often used to generate a starting point for later calculations using different *post-Hartree-Fock* methods. Initial work was done by Hartree, developing the self-consistent field method, as soon as a year after Schrödinger published his derivation of the well known Schrödinger equation. Fock revised Hartree's work in 1930 by pointing out that the self-consistent field method did not fully obey Pauli's exclusion principle, and corrected the method into Hartree-Fock on a functional form.

We will introduce the Hartree-Fock method in the form of linear algebra, employing the same philosophy, following a variational approach. With a reference determinant $|\Phi_0^{(HF)}\rangle$ built out of any basis set, one could never underestimate the ground-state energy expectation value,

$$\langle\Phi_0^{(HF)}|\hat{H}|\Phi_0^{(HF)}\rangle = E_{ref} \geq \langle\Psi|\hat{H}|\Psi\rangle = E_0, \tag{5.88}$$

where E_0 is the ground state of the exact solution $|\Psi\rangle$. Starting with a basis set $|\alpha\rangle$ and performing a unitary transformation

$$|p\rangle = \sum_\alpha C_{p\alpha}|\alpha\rangle, \tag{5.89}$$

we will try to minimize the expectation value by varying the unitary matrix C. If $|\Phi_0^{(HF)}\rangle$ is built up by the N lowest-lying states in the transformed basis we would find its expectation value to be

$$E\left[\Phi_0^{(HF)}\right] = \langle\Phi_0^{(HF)}|\hat{H}|\Phi_0^{(HF)}\rangle = \sum_i \langle i|\hat{h}_0|i\rangle + \frac{1}{2}\sum_{ij}\langle ij||ij\rangle, \tag{5.90}$$

expressed in terms of our initial basis as

$$E\left[\Phi_0^{(HF)}\right] = \sum_i \sum_{\alpha\beta} C^*_{i\alpha}C_{i\beta}\langle\alpha|\hat{h}_0|\beta\rangle + \frac{1}{2}\sum_{ij}\sum_{\alpha\beta\gamma\delta} C^*_{i\alpha}C^*_{j\beta}C_{i\gamma}C_{j\delta}\langle\alpha\beta||\gamma\delta\rangle. \tag{5.91}$$

It is here understood that i and j are hole-states in the HF basis, whereas Greek letters come from the original basis whose sums loop over the entire basis set.

Introducing Lagrangian multipliers $\sum_i \omega_i \sum_\alpha C^*_{i\alpha}C_{i\alpha}$, we find the minima of the energy by

$$\begin{aligned} 0 =& \frac{\partial}{\partial C^*_{k\kappa}}\left(E\left[\Phi_0^{(HF)}\right] - \sum_i \omega_i \sum_\alpha C^*_{i\alpha}C_{i\alpha}\right) \\ =& \sum_\beta C_{k\beta}\langle\kappa|\hat{h}_0|\beta\rangle + \sum_j\sum_{\beta\gamma\delta} C^*_{j\beta}C_{k\gamma}C_{j\delta}\langle\kappa\beta||\gamma\delta\rangle - \omega_k C_{k\kappa}, \end{aligned} \tag{5.92}$$

which should hold for all k, κ, resulting in

$$\sum_{\gamma} C_{k\gamma} \left[\langle\alpha|\hat{h}_0|\gamma\rangle + \sum_{j} \sum_{\beta\delta} C^*_{j\beta} C_{j\delta} \langle\alpha\beta||\gamma\delta\rangle \right] = \omega_k C_{k\alpha}. \tag{5.93}$$

The Hartree-Fock Hamiltonian is defined as

$$\hat{h}^{HF}_{\alpha\gamma} = \langle\alpha|\hat{h}_0|\gamma\rangle + \sum_{j} \sum_{\beta\delta} C^*_{j\beta} C_{j\delta} \langle\alpha\beta||\gamma\delta\rangle, \tag{5.94}$$

in order to simplify the Hartree-Fock equations, (5.93), to

$$\sum_{\gamma} \hat{h}^{HF}_{\alpha\gamma} C_{k\gamma} = \omega_k C_{k\alpha}. \tag{5.95}$$

In terms of linear algebra the transposed coefficient matrix holds eigenvectors of the Hartree-Fock Hamiltonian (5.94), with eigenvalues ω_k,

$$\hat{h}^{HF} (C^T)_{col(k)} = \omega_k (C^T)_{col(k)}. \tag{5.96}$$

The HF Hamiltonian (5.94) depends on the transformation coefficient matrix, which in turn consist of the Hamiltonian's eigenvectors, a circular dependency that makes it hard to find exact solutions, and one needs to solve it iteratively. A typical approach is to start with an untransformed basis, being equivalent to setting $C = \hat{1}$. With this guess for C one calculates $\hat{h}^{HF}$, whose eigenvectors leads to a new 'guess' for C. After repeating this procedure a number of times, we abort the iterations when the results converges.

To obtain a HF basis suitable for the CCSD machinery one simply redefines matrix elements to be

$$\langle p|\hat{h}_0|q\rangle = \sum_{\alpha\beta} C^*_{p\alpha} C_{q\beta} \langle\alpha|\hat{h}_0|\beta\rangle, \tag{5.97}$$

and

$$\langle pq||rs\rangle = \sum_{\alpha\beta\gamma\delta} C^*_{p\alpha} C^*_{q\beta} C_{r\gamma} C_{s\delta} \langle\alpha\beta||\gamma\delta\rangle. \tag{5.98}$$

5.4.1 Implementing HF

In order to create an efficient Hartree-Fock implementation one needs to rewrite sums into matrix operations, making it as 'vectorized' as possible. The first simplification we make is to define

$$C^i_{pq} = \sum_k C_{kp} C_{kq}, \tag{5.99}$$

easily vectorized in listing 5.8. It is now possible to simplify the energy from eq. (5.91) to

$$E\left[\Phi_0^{(HF)}\right] = \sum_{\alpha\beta} C^i_{\alpha\beta} \langle\alpha|\hat{h}_0|\beta\rangle + \frac{1}{2} \sum_{\alpha\beta\gamma\delta} C^i_{\alpha\gamma} C^i_{\beta\delta} \langle\alpha\beta||\gamma\delta\rangle. \tag{5.100}$$

Listing 5.8: Vectorized procedure to obtain C^i (C_inner).

```
mat C_holeXall = C.submat(span(0, numHOLEstates - 1), span::
    all);
mat C_inner = C_holeXall.t() * C_holeXall;
```

Listing 5.9: H0 part of hf E

```
double E_ref = accu(C_inner % h0);
```

Although it is slightly simplified it is not possible to streamline the two-particle part due to indices not matching. Once again we solve the complication by remapping the matrices. The interaction is remapped to matrix blocks diagonal in a redefined channel, λ',

$$\langle\alpha\beta||\gamma\delta\rangle \to \langle\alpha\gamma^{-1}||\delta\beta^{-1}\rangle_{\lambda'}. \tag{5.101}$$

Also mapped are the coefficients into vectors in the same channels, one column vector with permuted indices (P), and one row vector that is not (N),

$$\begin{aligned} C^i_{\alpha\gamma} &\to C^N(\lambda')_{\alpha\gamma^{-1}} \\ C^i_{\beta\delta} &\to C^P(\lambda')_{\delta\beta^{-1}}. \end{aligned} \tag{5.102}$$

In total we now have

$$\begin{aligned} E\left[\Phi_0^{(HF)}\right] =& \sum_{\alpha\beta} C^i_{\alpha\beta}\langle\alpha|\hat{h}_0|\beta\rangle \\ &+\frac{1}{2}\sum_{\lambda'}\sum_{(\alpha\gamma^{-1})}\sum_{(\delta\beta^{-1})} C^N(\lambda')_{\alpha\gamma^{-1}}\langle\alpha\gamma^{-1}||\delta\beta^{-1}\rangle_{\lambda'}C^P(\lambda')_{\delta\beta^{-1}}. \end{aligned} \tag{5.103}$$

The energy from single-particle interactions can be obtained by the one simple statement in listing 5.9. Two-particle interactions are slightly more complicated since we store matrices based on the region they span,

$$\langle hh||hh\rangle, \langle ph||hh\rangle, \langle pp||hh\rangle, \langle pp||ph\rangle, \langle pp||pp\rangle, \tag{5.104}$$

but we need to account for all possibilities;

$$\begin{aligned} &\langle hh||hh\rangle, \\ &\langle ph||hh\rangle, \langle hp||hh\rangle, \langle hh||ph\rangle, \langle hh||hp\rangle, \\ &\langle pp||hh\rangle, \langle hh||pp\rangle, \\ &\langle ph||ph\rangle, \langle ph||hp\rangle, \langle hp||hp\rangle, \langle hp||ph\rangle, \\ &\langle pp||ph\rangle, \langle pp||hp\rangle, \langle ph||pp\rangle, \langle hp||pp\rangle, \\ &\langle pp||pp\rangle. \end{aligned} \tag{5.105}$$

For this reason one needs to sort the coefficients whether the span hh, ph, hp or pp, created in listing 5.10, before we calculate the energy from the two-particle interactions in (5.103). Taking into account the different permutations of the indices seen from (5.105), we get a corresponding number vector-matrix-vector products, illustrated for one channel in 5.11.

Listing 5.10: Filling the coefficients from eq (5.102). C^N is stored in 'C_xx1' whereas C^P is stored in 'C_xx1_t'.

```
//FIll C_hh1
size_t dimHH1 = map_hh1.at(lmd).size();
vec C_hh1 = zeros<vec > (dimHH1);
vec C_hh1_t = zeros<vec > (dimHH1);
for (int idx_db1 = 0; idx_db1 < dimHH1; idx_db1++)
{
    int db = map_hh1.at(lmd)(idx_db1);
    int delta = db % nH;
    int beta = db / nH;
    C_hh1_t(idx_db1) = C_inner(beta, delta);
    C_hh1(idx_db1) = C_inner(delta, beta);
}
//Fill C_pp1
size_t dimPP1 = map_pp1.at(lmd).size();
vec C_pp1 = zeros<vec > (dimPP1);
vec C_pp1_t = zeros<vec > (dimPP1);
for (int idx_db1 = 0; idx_db1 < dimPP1; idx_db1++)
{
    int db = map_pp1.at(lmd)(idx_db1);
    int delta = db % nP + nH;
    int beta = db / nP + nH;
    C_pp1(idx_db1) = C_inner(delta, beta);
    C_pp1_t(idx_db1) = C_inner(beta, delta);
}
//Fill C_ph1
size_t dimPH1 = map_ph1.at(lmd).size();
vec C_ph1 = zeros<vec > (dimPH1);
vec C_ph1_t = zeros<vec > (dimPH1);
for (int idx_ak1 = 0; idx_ak1 < dimPH1; idx_ak1++)
{
    int ak = map_ph1.at(lmd)(idx_ak1);
    int a = ak % nP + nH;
    int k = ak / nP;
    C_ph1(idx_ak1) = C_inner(a, k);
    C_ph1_t(idx_ak1) = C_inner(k, a);
}
//Fill C_hp1
size_t dimHP1 = map_hp1.at(lmd).size();
vec C_hp1 = zeros<vec > (dimHP1);
vec C_hp1_t = zeros<vec > (dimHP1);
for (int idx_lb1 = 0; idx_lb1 < dimHP1; idx_lb1++)
{
    int bl = map_hp1.at(lmd)(idx_lb1);
    int b = bl % nP + nH;
    int l = bl / nP;
    C_hp1(idx_lb1) = C_inner(l, b);
    C_hp1_t(idx_lb1) = C_inner(b, l);
}
```

Listing 5.11: Two-particle part of HF energy.

```
E_tpPart = 0;

//hhhh
E_tpPart += as_scalar(C_hh1.t()*v_ik1_lj1.at(lmd)*C_hh1_t);
//phhh
E_tpPart += as_scalar(C_ph1.t()*v_ak1_lj1.at(lmd)*C_hh1_t);
E_tpPart += as_scalar(C_ph1.t()*v_ak1_lj1.at(lmd)*C_hh1_t);
E_tpPart += as_scalar(C_ph1_t.t()*v_ak1_lj1.at(lmd)*C_hh1);
E_tpPart += as_scalar(C_ph1_t.t()*v_ak1_lj1.at(lmd)*C_hh1);
//pphh
E_tpPart += as_scalar(C_ph1.t()*v_ak1_lb1.at(lmd)*C_hp1_t);
E_tpPart += as_scalar(C_ph1_t.t()*v_ak1_lb1.at(lmd)*C_hp1);
//phph
E_tpPart += as_scalar(C_pp1.t()*v_ac1_lj1.at(lmd)*C_hh1_t);
E_tpPart += as_scalar(C_pp1.t()*v_ac1_lj1.at(lmd)*C_hh1_t);
E_tpPart -= as_scalar(C_ph1.t()*v_al1_cj1.at(lmd)*C_ph1_t);
E_tpPart -= as_scalar(C_ph1.t()*v_al1_cj1.at(lmd)*C_ph1_t);
//ppph
E_tpPart += as_scalar(C_pp1.t()*v_ac1_lb1.at(lmd)*C_hp1_t);
E_tpPart += as_scalar(C_pp1.t()*v_ac1_lb1.at(lmd)*C_hp1_t);
E_tpPart += as_scalar(C_pp1_t.t()*v_ac1_lb1.at(lmd)*C_hp1);
E_tpPart += as_scalar(C_pp1_t.t()*v_ac1_lb1.at(lmd)*C_hp1);
//pppp
E_tpPart += as_scalar(C_pp1.t()*v_ac1_db1.at(lmd)*C_pp1_t);

E_ref += 0.5 * E_tpPart;
```

Chapter 6

OpenCL

This chapter serves as an overview of OpenCL, an open and royalty-free standard for parallel programming on heterogenous systems. Topics that are important for the understanding of implementations presented in this thesis will be emphasized. In addition, a more thorough study of matrix multiplication is presented, as well as historical aspects on GPGPU computing.

6.1 General-purpose computing on GPU

With the continuous emergence of new graphic cards, high computational powers reaches consumers, ranging from regular desktop users to HPC users. Whereas regular processors have powers in the range of 10 to 100 gigaflops, graphic cards can reach more than teraflops performance. As an example the AMD radeonhd 6970, which is used extensively in this thesis, is listed with a speed of 2.7 TFLOPs for single precision, and 683 GFLOPs for double precision [13]. Having a price of a few thousand NOK, graphical processing units (GPUs) should be able to compete with larger clusters having a much higher price.

GPUs were originally, and still are, targeted mainly towards computer graphics. To be able to produce real-time graphics with a high level of detail, such a processor must be highly parallel. A large amount of pixels must be computed within a short time range, often processed independently by the same set of instructions. Some programmers did eventually see the potential of GPUs to paralellize and accelerate general purpose programs, but the lack of standards for this forced them to translate problems into a graphics-like problem in order to run them on GPUs. Video cards also lacked hardware features making it even harder to program general code on them. Even today double-precision arithmetic is supported mostly on high-end cards, and require the latest drivers.

To overcome the issues of GPU computing, a few standards arose. At the moment three major standards exists; DirectCompute [14], CUDA [15] and OpenCL [16]. All of them supply a language for writing accelerated code, as well as an API to control the execution of this code. The first two standards will not be used in this thesis, mainly because of their lack of choice. Whereas DirectCompute enforces the use of Microsoft Windows and CUDA limits itself to NVIDIA cards, OpenCL have implementations on all

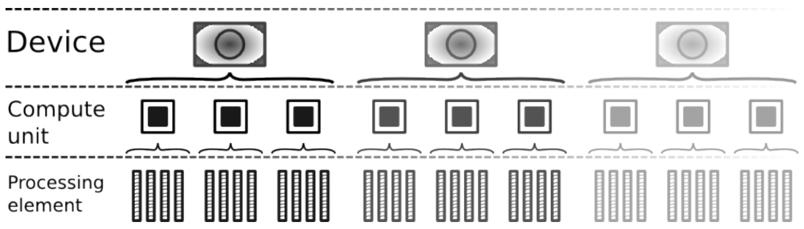


Figure 6.1: The hierarchy defined by the OpenCL platform model. A platform consists of multiple devices, each having multiple compute units. Whereas compute units can run independent threads, they can still contain multiple processing elements. The memory model (6.2.3) will also fit in this hierarchy, with (from the top) global, local and private memory residing close to corresponding layer.

common[1] operating systems for both CPUs and GPUs from Intel, AMD and NVIDIA.

6.2 The OpenCL model

The OpenCL standard [16] describes OpenCL with the following hierarchy of models:

- Platform model
- Memory model
- Execution model
- Programming model

6.2.1 Platform model

OpenCL programs consist of a host running code written in C/C++.[2] This host have access to different devices, each consisting of a number of compute units with one or more processing elements.

The simplest setup would consist of a single-threaded host program running on the CPU. By querying for a platform and a device the program can gain access to running parallel code on the same CPU. This may seem illogical, but it is an easy way to program parallel on the processor. A quad core, which is common, would typically permit four compute units (four *independent* threads), with one processing element each.

A slightly more advanced configuration could consist of the simple setup, but in addition have access to the video card. The GPU would now show up as a second device, having for example 24 compute units (SIMD cores) with 64 processing elements each. Whereas the compute units run independently, the processing elements within a compute unit will generally not. This hierarchy is defined by the platform model outlined in fig. 6.1.

[1]Implementations exists at least for recent versions of Windows, Mac and Linux

[2]The OpenCL API is written in C with bindings for C++. Third party wrappers can still be found for other languages. See for example JOCL or javaCL for java, and pyopencl for python implementations.

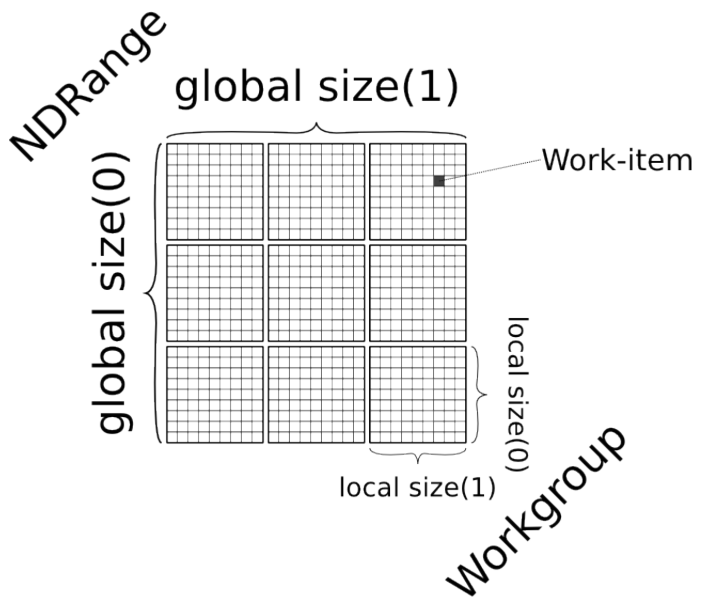


Figure 6.2: A global two dimensional 27x27 NDRange. It is divided into nine work-groups, each having 9x9 work-items.

6.2.2 Execution model

Parallel code is written in its own language '.cl', a subset of the C99 standard, where functions callable from the host are called kernels. This code is then compiled for an already initialized device, and submitted to a command queue. The command queue is now responsible for running this kernel once the device is ready. It is possible to submit multiple kernels to a queue and manage their execution via events.

Every time a kernel is queued a virtual grid, NDRange, will be defined. For every grid point, the kernel will be executed once. This is how parallelism is achieved in OpenCL. An NDRange can be one, two or three dimensional, and the size can be chosen arbitrarily up to a limit defined by the device.

The entire global grid is divided into smaller local grids called work-groups. An illustration of a 27x27 '2DRange' composed of nine 9x9 workgroups is found in fig. 6.2. During execution each work-item will fill one processing element. All work-items within the same work-group will run concurrently, but on the same compute unit. The other work-groups will run on other compute units within the same device.

6.2.3 Memory model

There are four different regions for OpenCL memory:

- **Global** memory is allocated by the host. Both the kernel and the host have read-/write access. Global memory is in general the slowest type of memory, but compensates with the largest memory size. Ideal for storing input, results or large datasets.

	Global		Constant		Local		Private	
	Host	Kernel	Host	Kernel	Host	Kernel	Host	Kernel
Allocation	Dyn	NA	Dyn	NA	Dyn	Stat	NA	Stat
Access	R/W	R/W	R/W	R	NA	R/W	NA	R/W

Table 6.1: Overview of different OpenCL memory regions. R/W: read and write. R: read only. Dyn/Stat: dynamic/static. NA: not available.

- **Constant** memory is allocated by the host, with read-only access from a kernel. Often used when passing many parameters/options as one struct to the kernel.
- **Local** memory could be allocated either dynamically by the host or statically inside a kernel. Every work group has its own independent copy of local memory, shared among the work-items inside the group. Residing close to each compute unit this memory is faster than global.
- **Private** memory is allocated statically by each work-item and not shared at all. Being close to the processing elements it is the fastest memory. Variables declared in kernels without specifying memory region are private by default.

A quick summary of the different memory regions can be found in table 6.1.

6.2.4 Programming model

OpenCL focuses on a data parallel programming model. Here we have multiple memory objects that can be calculated or altered by the same set of instructions. Instead of using loops or nested loops, our problem can be simplified by choosing appropriate global and local sizes, as we shall see for general matrix multiplication in section 6.3. Task parallel programming, and hybrids between these two, are also supported. GPUs however favors the data parallel model, due to their SIMD architecture.

SIMD, single instruction multiple data, is a type of parallel hardware where the same instruction is performed on different sets of data at the same time. A radeonhd 6970 consists of 64 processing elements within each compute unit, a SIMD core. All 64 processing elements are required to do the same operations at all times, only differing by what data they are working on. A work-group maps to one SIMD core, thus it is unfavorable to let work-items within a work-group branch by for instance an if statement.

6.3 Matrix multiplication

The coupled-cluster code has its bottleneck in matrix-matrix multiplication. Take for instance the second term of (5.70),

$$t_{ij}^{ab} \leftarrow \frac{1}{2}\langle ab||de\rangle t_{ij}^{de}. \tag{6.1}$$

Gathering indices $a, b \to \xi$, $d, e \to \xi'$ and finally $i, j \to \mu$, it can be rewritten as

$$t_{\xi\mu} \leftarrow \frac{1}{2}\sum_{\mu'}\langle\xi||\xi'\rangle t_{\xi'\mu}, \tag{6.2}$$

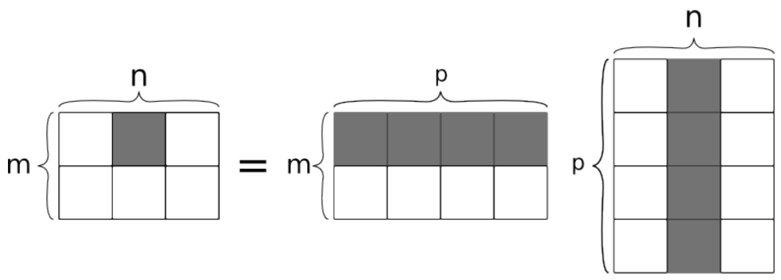


Figure 6.3: Outline of matrix multiplication, following its straight forward mathematical definition.

which is, apart from the factor $\frac{1}{2}$, identical to the mathematical definition of matrix multiplication,

$$C_{ij} = \sum_k A_{ik} B_{kj}, \qquad A \in \mathbf{R}^{m\times p}, B \in \mathbf{R}^{p\times n}, C \in \mathbf{R}^{m\times n}. \tag{6.3}$$

Element C_{ij} is here the inner product of row i in A and column j in B, as illustrated in fig. 6.3.

To illustrate the cl language we consider a simple implementation for matrix multiplication. In listing 6.1 we have a kernel taking three parameters, the matrices A, B and C, where A is already transposed. The problem is divided into one work-item for each value of i and j, each running the sum defined in (6.3). Running on a CPU this would be a good way to parallelize matrix multiplication. The GPU, however, would be penalized by slow memory access from global memory.

A way to improve memory access is to express the problem as multiplication of matrix-blocks. Figure 6.4 shows how one block in the C matrix is the sum of the matrix product of blocks in A and B. We set the block size to the same as the work-group size and use a 2DRange with the same dimensionality as C. Each work-item is now responsible for reading one element from A and one element from B into the local blocks, before summing over elements in local memory. After the sum corresponding to block multiplication is done, next block is loaded into local memory and so on. A complete implementation is included in listing 6.2. All work-items must complete loading from memory before any other work-item begin the summation, ensured by inserting a barrier. We must also prevent any work-item to start loading next block until all work-items are done using the current block. Thus we have two barriers, and one should note that barriers in OpenCL can only be applied within the same work-group. Using local memory we have reduced the number of times matrix elements need to be transfered from global memory.

The two examples of matrix multiplication shown here are not optimal solutions. The first is quite inefficient and the second is not flexible, since the matrix sizes are limited by the block size. AMD has developed its own BLAS library for GPUs, AMD APPML, which is both faster and more flexible.

Listing 6.1: Simple matrix multiplication in OpenCL. The matrices A, B and C refers to the same matrices as in eq. (6.3). The left matrix, A is already transposed, 'A_tr', to improve access pattern.

```
typedef double fp;

kernel void
matmult(global fp* A_tr, global fp* B, global fp* C)
{
    //WI global id (x,y)=(i,j)
    int2 gid = (int2) (get_global_id(0), get_global_id(1));

    //Calculate C_ij
    fp C_ij = 0;
    for (int k = 0; k < SIZE_P; k++)
        C_ij += A_tr[gid.x * SIZE_P + k]
                * B[gid.y * SIZE_P + k];

    //Store into global memory again
    C[gid.x + gid.y * SIZE_M] = C_ij;
}
```

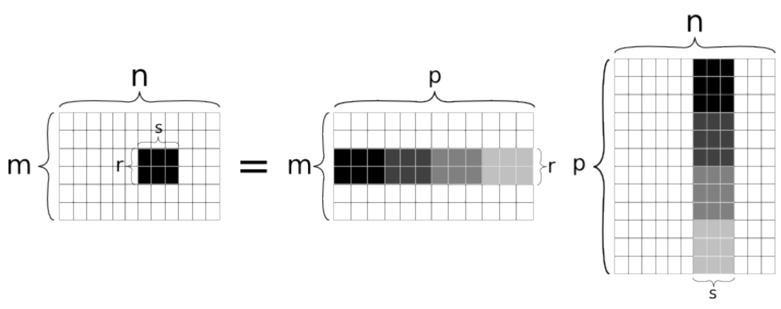


Figure 6.4: Outline of blocked matrix multiplication.

Listing 6.2: Matrix multiplication in OpenCL using local memory. The matrices A, B and C refers to the same matrices as in eq. (6.3).

```
typedef double fp;

kernel void
matmult(global fp* A_glb, global fp* B_glb, global fp* C_glb)
{
 //WI global id (x,y)=(i,j)
 int2 gid = (int2) (get_global_id(0), get_global_id(1));
 //Local id (x,y)
 int2 lid = (int2) (get_local_id(0), get_local_id(1));

 //Local storage for blocks.
 local fp A[BL_SIZ][BL_SIZ];
 local fp B[BL_SIZ][BL_SIZ];

 //Element for this WI
 fp C_ij = 0;

 //Loop over blocks
 for(int k_block = 0; k_block < SIZ_P; k_block += BL_SIZ)
 {
   //Read blocks of A and B
   barrier(CLK_LOCAL_MEM_FENCE);
   A[lid.x][lid.y] = A_glb[gid.x + (k_block+lid.y) * SIZ_M];
   B[lid.x][lid.y] = B_glb[(k_block+lid.x) + gid.y * SIZ_P];

   //Block multiplication
   barrier(CLK_LOCAL_MEM_FENCE);
   for(int k_loc = 0; k_loc < BL_SIZ; k_loc++)
     C_ij += A[lid.x][k_loc] * B[k_loc][lid.y];
 }

 //Store into global memory again
 C[gid.x + gid.y * SIZE_M] = C_ij;
}
```

6.3.1 Strassen's algorithm

The straightforward algorithm for matrix multiplication will require p multiplications and $p-1$ additions for each of the $m \times n$ elements. The total number of floating-point operations is then $mn(2p-1) \sim \mathcal{O}(mnp)$. When the matrices A and B can be divided into four equally sized blocks,

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}, \tag{6.4}$$

we get eight multiplications of smaller blocks,

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{bmatrix}. \tag{6.5}$$

Strassen discovered in 1968 how the number of multiplications could be reduced from eight to seven [17]. Following Winograd's approach we define some intermediates,

$$\begin{aligned} S_1 &= A_{21} + A_{22}, & T_1 &= B_{12} - B_{11}, \\ S_2 &= S_1 - A_{11}, & T_2 &= B_{22} - T_1, \\ S_3 &= A_{11} - A_{21}, & T_3 &= B_{22} - B_{12}, \\ S_4 &= A_{12} - S_2, & T_4 &= B_{21} - T_2, \end{aligned} \tag{6.6}$$

and need seven multiplications,

$$\begin{aligned} P_1 &= A_{11}B_{11}, & U_1 &= P_1 + P_2, \\ P_2 &= A_{12}B_{21}, & U_2 &= P_1 + P_4, \\ P_3 &= S_1T_1, & U_3 &= U_2 + P_5, \\ P_4 &= S_2T_2, & U_4 &= U_3 + P_7, \\ P_5 &= S_3T_3, & U_5 &= U_3 + P_3, \\ P_6 &= S_4B_{22}, & U_6 &= U_2 + P_3, \\ P_7 &= A_{22}T_4, & U_7 &= U_6 + P_6, \end{aligned} \tag{6.7}$$

to find the resulting C matrix as

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} U_1 & U_7 \\ U_4 & U_5 \end{bmatrix}. \tag{6.8}$$

Despite the seemingly additional work, we have reduced the number of multiplications from 8 to 7. Since the multiplications are the computational bottleneck compared to addition and subtraction, the number of flops are reduced.

In the case of square $n \times n$ matrices with n equal to a power of two, $n = 2^m$, the divided blocks will have $\frac{n}{2} = 2^{m-1}$. Letting $f(m)$ be the number of flops needed for the full matrix and applying Strassen recursively we find the total number of flops to be

$$f(m) = 7f(m-1) = 7^2 f(m-2) = \cdots = 7^m f(0), \tag{6.9}$$

where $f(0)$ is the one floating-point operation needed for multiplication of two numbers (two $2^0 \times 2^0$ matrices). For large matrices this can prove efficient, yielding a much better scaling,

$$\mathcal{O}\left(7^m\right) = \mathcal{O}\left(2^{\log_2 7^m}\right) = \mathcal{O}\left(2^{m \log_2 7}\right) = \mathcal{O}\left(n^{\log_2 7}\right) \approx \mathcal{O}\left(n^{2.807}\right), \tag{6.10}$$

effectively saving $7/8 = 12.5\%$ each time it is applied.

Listing 6.3: GEMM default implementation.

```
virtual void dgemm(
        arma::mat &res,
        arma::mat const &left,
        arma::mat const &right,
        double alpha = 1,
        double beta = 0,
        bool transL = false,
        bool transR = false)
{
    timer.tic();
    if (transL == false && transR == false)
        res = alpha * left * right + beta * res;
    else if (transL == true && transR == false)
        res = alpha * trans(left) * right + beta * res;
    else if (transL == false && transR == true)
        res = alpha * left * trans(right) + beta * res;
    else if (transL == true && transR == true)
        res = alpha * trans(left) * trans(right) + beta * res
           ;
    tot_time += timer.toc();

    return;
}
```

6.4 Implementation

Matrix multiplication has been considered an important aspect of this thesis, and all functionality has been implemented in a separate object called ‘GEMM’ – GEneral Matrix Multiplication. The default implementation, demonstrated in listing 6.3, performs the matrix operation,

$$C_{(res)} = \alpha A_{(left)} B_{(right)} + \beta C_{(res)}, \tag{6.11}$$

where the variable names used in the implementation are denoted in the subscript of the matrices. When ‘transL’ and/or ‘transR’ is set to true $A_{(left)}$ and/or $B_{(right)}$ will be transposed before used. Armadillo’s default operations will be used in ‘GEMM’, most likely translated into calls to some BLAS library. Netlib [18] and GotoBLAS2 [19] are the two implementations we have tested here.

To take advantage of different implementations all C++ statements for matrix multiplication needs to be replaced with a call to a ‘GEMM’ derived class. As an example the second term from the coupled-cluster $\hat{T}_2$ equations, eq. (5.79), will now be written as

$$\langle\xi|t_2'|\mu\rangle_{(\lambda)} = \alpha \sum_{\xi'} \langle\xi||\xi'\rangle_{(\lambda)} \langle\xi'|t_2|\mu\rangle_{(\lambda)} + \beta\langle\xi|t_2'|\mu\rangle_{(\lambda)}, \tag{6.12}$$

with $\alpha = \frac{1}{2}$ and $\beta = 1$. Implementing this listing 5.3 is altered slightly, as depicted in listing 6.4.

Listing 6.4: Second term of the coupled-cluster $\hat{T}_2$ equations, now using the matrix-multiplication framework in 'GEMM'.

```
for (int lmd = 0; lmd < basis->dim_lmd_2p(); lmd++)
    mult->dgemm(     //mult points to a GEMM derived object
       t2_new.at(lmd),               //Result
       sys->get_v_pppp()->at(lmd),   //Left
       t2_old.at(lmd),               //Right
       0.5, 1);                      //alpha, beta
```

Listing 6.5: Strassen overrides the 'dgemm' method.

```
timer.tic();
if (transL == false && transR == false)
  res = alpha * strassenMethod(left, right, 0)
       + beta * res;
else if (transL == true && transR == false)
  res = alpha * strassenMethod(trans(left), right, 0)
       + beta * res;
else if (transL == false && transR == true)
  res = alpha * strassenMethod(left, trans(right), 0)
       + beta * res;
else if (transL == true && transR == true)
  res = alpha * strassenMethod(trans(left), trans(right), 0)
       + beta * res;
tot_time += timer.toc();

return;
```

Also incorporated is a system for measuring the amount of time that is used on multiplication. A protected member, timer, of type 'arma::wall_clock', is included as well as a double, tot_time, to record how much time is spent inside the 'dgemm' method. To get this information one simply calls 'get_tot_time()', returning the number of seconds elapsed.

6.4.1 Strassen

'Strassen' subclasses 'GEMM' and adds the method 'strassenMethod' returning the matrix product of two matrices, 'left' and 'right', using the Strassen method. The Strassen method is applied recursively. An integer, 'depth', is set to 0 for the first call, and it is raised by one each time a new level of recursion is performed. This allows us to abort recursions at a specific level, helpful when timing whether blas routines or another level of recursion is the most beneficial for a specific size of the matrices.

The 'dgemm' function is overridden as in listing 6.5, essentially the same as for the default base-class implementation except now using the new 'strassenMethod'. In

Listing 6.6: Implementation of Strassen's method. Continued in listing 6.7.

```cpp
mat Strassen::strassenMethod(
    const mat& left,
    const mat& right,
    int depth)   {
//Needed integer values
int m = left.n_rows;
int m2 = m / 2;
int n = right.n_cols;
int n2 = n / 2;
int p = left.n_cols;
int p2 = p / 2;
int dp1 = depth + 1;

//Criterion for further recursion. Tau is a class member (
   double)
if (m < 2 || n < 2 || p < 2 || (3.0 * m * n * p) / (((double)
    n) * p + ((double) m) * n + ((double) m) * p) < tau)
{
  return left * right;
}
```

'strassenMethod' the criterion for adding one level of recursion or invoking an underlying blas library is

$$3mnp < \tau \left(mn + np + pm\right), \tag{6.13}$$

reduced to

$$n < \tau \tag{6.14}$$

in the case of equally sized square matrices, $m = n = p$. Such a criterion was first proposed by Higham in 1990 [20]. Other criteria exist but this was chosen because only one variable, τ, needs to be tuned. We estimate τ empirically by measuring the time a regular 'dgemm' routine uses compared to doing one Strassen recursion for different sizes m, n and p.

The first part of 'strassenMethod', listing 6.6, defines some integer values needed, 'm', 'n' and 'p' hold the size of the matrices, and 'm2', 'n2' and 'p2' store half their size. 'dp1' is the current recursion level plus one. If one of the dimensions is less than two it is not possible to split the matrices further, and if the recursion criterion, eq. (6.13), is not met it is not beneficial to split the matrices either. These conditions are tested for, and regular multiplication through armadillo will then be used instead.

The second part, in listing 6.7, deals with situations where matrices may have odd dimensions, and thus not suitable for the Strassen algorithm. We deal with these situations using dynamic peeling. The odd rows and columns are treated separately,

$$\begin{bmatrix} C_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} B_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} A_{11}B_{11} + a_{12}b_{21} & A_{11}b_{12} + a_{12}b_{22} \\ a_{21}B_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{bmatrix}, \tag{6.15}$$

where A_{11}, B_{11} and C_{11} now are the largest possible blocks with even dimensions. We then have one multiplication,

$$C_{11} = A_{11}B_{11}, \tag{6.16}$$

suitable for another level of Strassen, and in the end a few fix-up steps,

$$\begin{aligned}
C_{11} =& a_{12}b_{21} + C_{11} \\
c_{12} =& A_{11}b_{12} + a_{12}b_{22} \\
c_{21} =& a_{21}B_{11} + a_{22}b_{21} \\
c_{22} =& a_{21}b_{12} + a_{22}b_{22},
\end{aligned} \tag{6.17}$$

where all steps may not be needed if only some of the dimensions are odd.

The third and last part of our Strassen implementation, listing 6.8, carries out the actual Strassen algorithm, implemented straight forward from the equations (6.6) and (6.7).

6.4.2 CLgemm

Matrix multiplication on GPUs through OpenCL is managed by AMD's own library, APPML (Accelerated Parallel Processing Math Libraries). Again the 'dgemm' method is altered to use a separate function for the product itself, this time implemented in 'clmult'.

The overhead when using GPUs is substantial, and we need to empirically find where a CPU implementation is more efficient. We have chosen the simplest criterion, whenever the amount of floating point operations needed is less than a threshold value armadillo's underlying functionality is used instead. Listing 6.9 shows how we find the number of flops required in 'clmult', and decide whether the CPU or GPU is best suited. If there is enough work to be done data will be pushed to the graphics card followed by invoking the routine 'clAmdBlasDgemm' to calculate the matrix product, as shown in listing 6.10.

6.4.3 CLstrassen

Our 'CLgemm' implementation will meet severe problems. As the matrix size increases, the device will eventually run out of available memory. In order to solve this, the matrix needs to be partitioned into smaller blocks that can be processed one at a time. For large matrices we also experience that applying Strassen's method on top of AMD's APPML serves no purpose. The GPU is more efficient than the overhead of blocking up the matrices.

For these reasons we have combined 'Strassen' and 'CLgemm', using the Strassen algorithm with a modified criterion, now only applying another level of recursion when the blocks are too big to fit on the GPU. The new criterion is described by listing 6.11, and the parameter 'maxSizeCL' is found by querying the device, as in listing 6.12.

Listing 6.7: Implementation of Strassen's method. Continuation of listing 6.6 and continued in listing 6.7.

```
else if (m % 2 != 0 || n % 2 != 0 || p % 2 != 0)
{
  span m2Div2(0, m2 * 2 - 1); //Spanning all elements
  span n2Div2(0, n2 * 2 - 1); //except the last, if
  span p2Div2(0, p2 * 2 - 1); //the total number is odd.
  span lastM(m - 1, m - 1); //Spanning
  span lastN(n - 1, n - 1); //the last
  span lastP(p - 1, p - 1); //element.

  mat C(m, n);
  C(m2Div2, n2Div2) = strassenMethod(left(m2Div2, p2Div2),
     right(p2Div2, n2Div2), depth);

  if (p % 2 != 0) //C_11 += a_12 b_21
    C(m2Div2, n2Div2) += left(m2Div2, lastP) * right(lastP,
       n2Div2);

  if (n % 2 != 0)
  { //c_12 = A_11 b_12
    C(m2Div2, lastN) = left(m2Div2, p2Div2) * right(p2Div2,
       lastN);

    if (p % 2 != 0)  //c_12 += a_12 b_22
      C(m2Div2, lastN) += left(m2Div2, lastP) * right(lastP,
         lastN);
  }

  if (m % 2 != 0)
  { //c_21 = a_21 B_11
    C(lastM, n2Div2) = left(lastM, p2Div2) * right(p2Div2,
       n2Div2);

    if (p % 2 != 0)  //c_21 += a_22 b_21
      C(lastM, n2Div2) += left(lastM, lastP) * right(lastP,
         n2Div2);
  }

  if (m % 2 != 0 && n % 2 != 0)
  { //c_22 = a_21 b_12
    C(lastM, lastN) = left(lastM, p2Div2) * right(p2Div2,
       lastN);

    if (p % 2 != 0)  //c_22 += a_22 b_22
      C(lastM, lastN) += left(lastM, lastP) * right(lastP,
         lastN);
  }

  return C;
}
```

Listing 6.8: Implementation of Strassen's method. Continuation of listing 6.7.

```
else
{
 //left matrix
 span lR1(0, m2 - 1); //First half of rows
 span lR2(m2, m - 1); //Second half of rows
 span lC1(0, p2 - 1); //First half of columns
 span lC2(p2, p - 1); //Second half of columns
 //right matrix
 span rR1(0, p2 - 1); //First half of rows
 span rR2(p2, p - 1); //Second half of rows
 span rC1(0, n2 - 1); //First half of columns
 span rC2(n2, n - 1); //Second half of columns

 //Intermediates
 mat S1 = left(lR2, lC1) + left(lR2, lC2);
 mat S2 = S1 - left(lR1, lC1);
 mat S3 = left(lR1, lC1) - left(lR2, lC1);
 mat S4 = left(lR1, lC2) - S2;
 mat T1 = right(rR1, rC2) - right(rR1, rC1);
 mat T2 = right(rR2, rC2) - T1;
 mat T3 = right(rR2, rC2) - right(rR1, rC2);
 mat T4 = right(rR2, rC1) - T2;

 //Multiplications
 mat P1 = strassenMethod(left(lR1,lC1), right(rR1,rC1), dp1);
 mat P2 = strassenMethod(left(lR1,lC2), right(rR2,rC1), dp1);
 mat P3 = strassenMethod(S1, T1, dp1);
 mat P4 = strassenMethod(S2, T2, dp1);
 mat P5 = strassenMethod(S3, T3, dp1);
 mat P6 = strassenMethod(S4, right(rR2, rC2), dp1);
 mat P7 = strassenMethod(left(lR2, lC2), T4, dp1);

 //U matrices
 mat U1 = P1 + P2;
 mat U2 = P1 + P4;
 mat U3 = U2 + P5;
 mat U4 = U3 + P7;
 mat U5 = U3 + P3;
 mat U6 = U2 + P3;
 mat U7 = U6 + P6;

 //Fill and return the result
 mat C(m, n);
 C(lR1, rC1) = U1;
 C(lR1, rC2) = U7;
 C(lR2, rC1) = U4;
 C(lR2, rC2) = U5;
 return C;
} //End of if-else

} //End of method
```

Listing 6.9: Implementation of matrix multiplication on a GPU. Continued in listing 6.10.

```
mat CLgemm::clmult(mat const &left, mat const &right) {

int m = left.n_rows;  //Size
int n = right.n_cols; //of input
int p = left.n_cols;  //matrices.

mat res = zeros<mat > (m, n); //Result

//Flops ~O(mnp)
double work = ((double) m) * n * p;

//Don't use GPU if few flops are required.
//This threshould value is found empirically
//by testing where GPUs are quicker than CPUs.
if (work < 3e8)
  res = left * right;
```

Listing 6.10: Implementation of matrix multiplication on a GPU. Continuation of listing 6.9

```
else
{
  //memptr cannot be const in cl.
  cl_double *A_p = const_cast<double *> (left.memptr());
  cl_double *B_p = const_cast<double *> (right.memptr());
  cl_double *C_p = res.memptr();

  //Create CL buffers, copying Host memory.
  cl::Buffer A_cl(context, CL_MEM_READ_ONLY |
     CL_MEM_COPY_HOST_PTR, sizeof (*A_p) * left.n_elem, A_p);
  cl::Buffer B_cl(context, CL_MEM_READ_ONLY |
     CL_MEM_COPY_HOST_PTR, sizeof (*B_p) * right.n_elem, B_p)
     ;
  cl::Buffer C_cl(context, CL_MEM_READ_WRITE |
     CL_MEM_COPY_HOST_PTR, sizeof (*C_p) * res.n_elem, C_p);

  //Run DGEMM. armadillo uses columnmajor ordering.
  clAmdBlasDgemm(
    clAmdBlasColumnMajor, clAmdBlasNoTrans, clAmdBlasNoTrans,
    m, n, p,      //Size of matrices.
    1.0, A_cl(), m, B_cl(), p,
    0.0, C_cl(), m,
    1, &queue(), 0, NULL, NULL);

  //Read results back into C
  queue.enqueueReadBuffer(C_cl, true, 0, sizeof (*C_p) * res.
     n_elem, C_p);
}

return res;
}//End Method CLgemm::clmult
```

Listing 6.11: CLstrassen's new criterion, compared to the original from listing 6.6. Another level of Strassen's algorithm is only applied if matrices are too big to fit on the GPU. Multiplications are then sent to 'clmult' instead of armadillo's blas routines.

```
int m = left.n_rows;
int m2 = m / 2;
int n = right.n_cols;
int n2 = n / 2;
int p = left.n_cols;
int p2 = p / 2;

//Number of elements in the three matrices
size_t sizRes = m * n;
size_t sizLeft = m * p;
size_t sizRight = n * p;
size_t sizTOT = sizRes + sizRight + sizLeft;
//Number of bytes needed
sizTOT = sizeof (double) * sizTOT;

//Matrix small enough for Device?
if (sizTOT < maxSizeCL)
  return clmult.clmult(left, right);
```

Listing 6.12: How to query the max number of bytes on a GPU device available for OpenCL.

```
cl_ulong maxmem_bytes = device.getInfo<
   CL_DEVICE_MAX_MEM_ALLOC_SIZE > ();
maxSizeCL = 0.9 * maxmem_bytes;
```

Part II

Results

Chapter 7

Results

7.1 Code validation

Developing scientific software there are often numerous potential pitfalls, small mistakes that eventually lead to bugs affecting the results. The exact results are generally not known prior to running the calculations. Larger errors can be found easily, as we have an estimate for the result, but the impact of an error is sometimes small enough to leave results within the estimated range. It is then hard to determine if the deviation is a software bug or an artifact arising from the method itself.

It is of great importance to us that the implementation is free of any errors. To guarantee this we perform a number of tests, whose results are known prior to running the calculations, and demand that our code can reproduce these values. In our case earlier implementations also exist. Thus we have a foundation to build upon and test against.

7.1.1 Hand calculation

First we consider systems of non-interacting particles, where the total energy is simply the sum of the single-particle energies, as seen in eq. (3.3). The first filled shell has two electrons with an energy of $\hbar\omega$, the next filled shell has four electrons with an energy of $2\hbar\omega$ each, and so on. A shell has two more electrons than the previous, each electron with an energy of $\hbar\omega$ more than an electron from the previous shell. Filling the F first shells there are $N = F(F+1)$ electrons, with a total uncorrelated energy

$$\langle \hat{H}_0 \rangle = 2 \cdot 1\hbar\omega + 4 \cdot 2\hbar\omega + \cdots + 2F \cdot F\hbar\omega = \sum_{i=1}^{F} 2i^2\hbar\omega, \tag{7.1}$$

a series that can be recognized as

$$\langle \hat{H}_0 \rangle = \frac{2F^3 + 3F^2 + F}{3}\hbar\omega. \tag{7.2}$$

Energies for the first 7 filled shells for different values of ω are found in table 7.1. These energies must be reproduced by both the Hartree-Fock (HF) and coupled-cluster programs. In order to test this we simply use an empty two-particle element file when running our program, thus leaving all matrix elements at their initialized value, zero.

Table 7.1: Uncorrelated energies for the first seven filled shells for different values of ω. Energies are measured in Hartrees.

					F			
	E_0 [Ha]	1	2	3	4	5	6	7
	1.0	2.0	10.0	28.0	60.0	110.0	182.0	280.0
	0.9	1.8	9.0	25.2	54.0	99.0	163.8	252.0
	0.8	1.6	8.0	22.4	48.0	88.0	145.6	224.0
	0.7	1.4	7.0	19.6	42.0	77.0	127.4	196.0
ω	0.6	1.2	6.0	16.8	36.0	66.0	109.2	168.0
	0.5	1.0	5.0	14.0	30.0	55.0	91.0	140.0
	0.4	0.8	4.0	11.2	24.0	44.0	72.8	112.0
	0.3	0.6	3.0	8.4	18.0	33.0	54.6	84.0
	0.2	0.4	2.0	5.6	12.0	22.0	36.4	56.0
	0.1	0.2	1.0	2.8	6.0	11.0	18.2	28.0

For two interacting particles it is possible to do calculations by hand, if restricted to a small set of basis functions. The simplest is to have one filled shell, $N = 2$, within a basis of two shells, six basis functions. Coupled cluster should, including single and double excitations, be able to account for all possible determinants in this case, and therefore yield the same result as for eigenvalues obtained by diagonalization of the full Hamilton matrix. The determinants will now have quantum number $n = 0$, allowing us to denote the determinants by angular momentum and spin only, $|m^p m_s^p; m^q m_s^q\rangle$. The reference determinant is either

$$|0\downarrow; 0\uparrow\rangle \text{ or } |0\uparrow; 0\downarrow\rangle. \tag{7.3}$$

Only excitations preserving $M = 0$ and $M_s = 0$ from the ground state can be created, restricting possible determinants to

$$\begin{array}{ll} |0\downarrow; 0\uparrow\rangle, & |0\uparrow; 0\downarrow\rangle, \\ |-1\downarrow; +1\uparrow\rangle, & |+1\uparrow; -1\downarrow\rangle, \\ |+1\downarrow; -1\uparrow\rangle, & |-1\uparrow; +1\downarrow\rangle. \end{array} \tag{7.4}$$

The left column have the same states as in the right, except for the two particles being swapped. At most raising a negative phase factor, this allows us to skip half of the states. Omitting the explicit spin notation, and considering only the three left states in (7.4), the Hamiltonian to solve is

$$\hat{H} = \begin{bmatrix} 2\omega + \langle 0;0||0;0\rangle & \langle 0;0||-1;+1\rangle & \langle 0;0||+1;-1\rangle \\ \langle -1;+1||0;0\rangle & 4\omega + \langle -1;+1||-1;+1\rangle & \langle -1;+1||+1;-1\rangle \\ \langle +1;-1||0;0\rangle & \langle +1;-1||-1;+1\rangle & 4\omega + \langle +1;-1||+1;-1\rangle \end{bmatrix}, \tag{7.5}$$

or

$$\hat{H} = \begin{bmatrix} 3.2533 & 0.3133 & 0.3133 \\ 0.3133 & 4.8617 & 0.2350 \\ 0.3133 & 0.2350 & 4.8617 \end{bmatrix}, \tag{7.6}$$

for $\omega = 1$. This matrix has the lowest eigenvalue,

$$E_0 = 3.1523, \tag{7.7}$$

reproduced by our implementation, even when including more decimals. It is possible to do hand calculations also for three shells, as in [1], resulting in

$$E_0 = 3.0386, \tag{7.8}$$

which is also reproduced.

7.1.2 Effective interaction

For two electrons and $\omega = 1.0$ the exact result is known to be $E_0 = 3[Ha]$ [21], a result we would like the two-particle case to converge towards. The standard interaction is the integral of the Coulomb repulsion, unfortunately shown to make results converge slowly as a function of the basis size. For this reason we investigate the use of an effective interaction, previously used with great success first in nuclear physics and later also for quantum dots [22].

An effective interaction is obtained by solving the two-particle problem in an untruncated Hilbert space and project the Hamiltonian into the truncated space by a similarity transform. The lowest eigenvalue of this effective, truncated, Hamiltonian should now be the same as for the exact result with a standard interaction in an infinite space. For more than two particles we will no longer get the same eigenvalue. The idea, however, is that the contribution from two-particle interactions still are accounted more precisely, compared to a standard interaction. All two-particle elements, both for standard and effective interactions, are produced and written to file using the OpenFCI [23] library.

We test our program by running with one filled shell (two electrons) using an effective interaction, and observe how we get the exact value of $3[Ha]$ for $\omega = 1$ irrespective of the basis size. In this test we use an energy-cut model space, $\mathbb{EC}(R)$, as shown in fig. 7.1. The coupled-cluster method works in the direct-product space, $\mathbb{DP}(R)$, in principle suggesting that half of the elements are zeroed out. Despite yielding exact results for two particles this has shown convergence problems for larger systems [2], and we therefore use a basis that is twice as big, $\mathbb{EC}(2R)$, when producing the interactions, and simply omit elements not suitable for the coupled-cluster approach. At the cost of introducing an error, assumed to be fairly small in the case of a large basis, we gain better convergence.

7.1.3 Earlier results

There is also a number of previously obtained results that we should be able to reproduce with our new program. We reproduce the HF and CCSD results of Lohne et al. [22] for both a Hartree-Fock and a harmonic-oscillator basis.

Summarizing, we reproduce earlier results, the code is consistent with full diagonalization for 2 particles, and effective interactions are used correctly. All tests performed indicate our code is correct.

7.2 Efficiency

We like to believe that our implementation has pushed the boundary on what calculations are possible to run on a single node. Different optimizations are done in order to get the program efficient without the penalty of loosing flexibility. The main improvement is

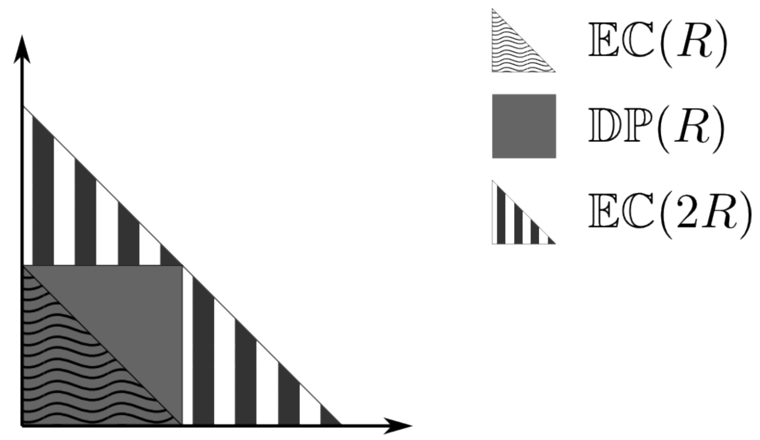


Figure 7.1: An effective interaction is obtained in an infinite basis and transformed into an energy-cut model space, $\mathbb{EC}(R)$. The coupled-cluster approach uses a direct-product model space, $\mathbb{DP}(R)$, suggesting that half of the elements are unnecessary. Also shown to have convergence problems, we use a larger energy-cut model space, $\mathbb{EC}(2R)$, and omit elements that do not fit inside the direct-product space. For a large basis, the omitted elements are assumed to be close to zero, resulting in a small induced error.

how all parts are now in terms of transition channels, reducing the dimensionality off all terms drastically. Another improvement is seen by exploring different approaches to matrix multiplications. We start by presenting benchmarks for the matrix-multiplications themselves, continued by a discussion in the efficiency of the program as a whole.

7.2.1 Optimized matrix multiplication

The main workhorse in this program is multiplication of matrices, whose dimension increases with increasing system and basis size. In the coupled-cluster machinery the most expensive term, eq. (5.79), scales as $\mathcal{O}\left(\dim(\xi)^2 \cdot \dim(\mu)\right)$ for each block of a transition channel, multiplying two matrices, A and B, with sizes of

$$A \in \mathcal{R}^{\dim(\xi)\times\dim(\xi)} \text{ and } B \in \mathcal{R}^{\dim(\xi)\times\dim(\mu)}. \tag{7.9}$$

The widest channel, $M = M_s = 0$, has 2 hole-hole configurations for two particles, 6 hole-hole configurations for six particles (see fig. 7.2), and thereafter growing. A more comprehensive list of the number of both hole-hole and particle-particle configurations in this channel is listed in table 7.2

We first test the different implementations of the 'GEMM' class calculating the matrix product of two matrices, A and B, defined in 7.9, filled with random numbers. In fig. 7.3 we see how the implementations perform for 2, 12, 30 and 56 electrons. The 'GEMM' base class is used with both Netlib and Goto's blas library as backend, Netlib running serially whereas Goto's implementation utilizes all four cores on the test machine, a stock Intel i7-920 clocked at 2.67 GHz. Both Netlib and Goto blas have also been used with the 'Strassen' sub class, conducting only one Strassen recursion, allowing us to identify whether such a recursion is beneficial or not. AMD's APPML library is also used through

Table 7.2: Listed are the number hole-hole configurations, $\max_\lambda \dim(\mu)$, and the number of particle-particle configurations, $\max_\lambda \dim(\xi)$, in the widest channel. The most expensive term for CCSD scales as ($\sim \max_\lambda \dim(\xi)^2 \max_\lambda \dim(\mu)$), equal to $n_p^4 n_h^2$ if not using block diagonal matrices.

	2	6	12	20	30	42	56
20	(2,2842)	(6,2766)	(16,2664)	(32,2528)	(58,2378)	(94,2198)	(144,2016)
22	(2,3764)	(6,3680)	(16,3566)	(32,3414)	(58,3244)	(94,3040)	(144,2830)
24	(2,4866)	(6,4774)	(16,4648)	(32,4480)	(58,4290)	(94,4062)	(144,3824)
26	(2,6164)	(6,6064)	(16,5926)	(32,5742)	(58,5532)	(94,5280)	(144,5014)
28	(2,7674)	(6,7566)	(16,7416)	(32,7216)	(58,6986)	(94,6710)	(144,6416)
30	(2,9412)	(6,9296)	(16,9134)	(32,8918)	(58,8668)	(94,8368)	(144,8046)

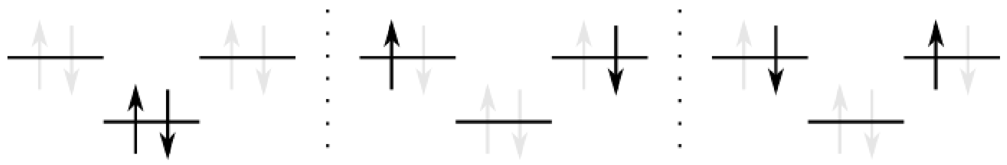


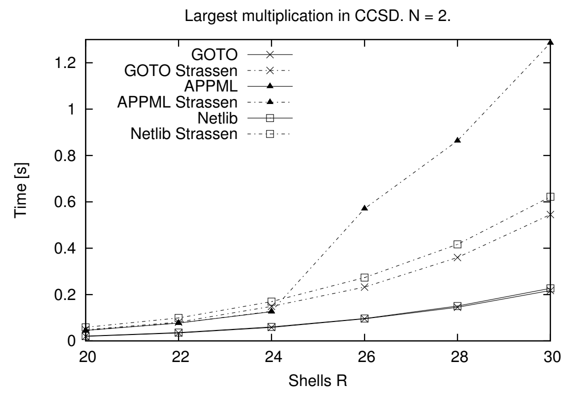
Figure 7.2: The three different hole-hole states with $M = M_s = 0$ for six particles. Counting the multiplicity of swapping the two particles there are six hole-hole states in total.

the 'CLgemm' class, which provides memory copying to and from device in addition to scheduling the blas routine. To avoid running out of memory on the GPU we split the matrices by applying a Strassen recursion whenever matrices fills more than 90% of the memory on the GPU, using the 'CLstrassen' class. This amount was chosen as a safety precaution, but could most likely be higher.

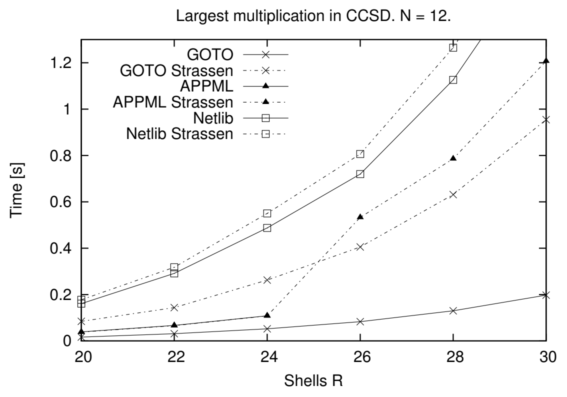
For two electrons we see clearly disadvantages of using a GPU or Strassen's method. The right matrix is no more than two elements wide, making it unsuitable for such a massive parallelization. The Strassen algorithm is neither considered beneficial for narrow matrices, as it more than doubles the wall time. Also seen is a dramatic amount of overhead performing the Strassen splitting after the GPU runs out of memory at 26 shells. Increasing the number of particles, the right matrix will widen slightly, ending at 144 elements wide for $N = 56$. Still a fairly thin matrix, neither Strassen nor GPU implementations can hold up against Goto's implementation, known to perform well in parallel over shared memory. Compared to Netlib's reference implementation, now more than ten times slower and barely included in the figure, parallel implementations prove their usefulness.

The results from the coupled-cluster benchmarks do not point toward any gain by using GPU's, most likely a consequence of overhead from data-copying. The overhead is also significant in the Strassen methods, an overhead we believe could be optimized further.

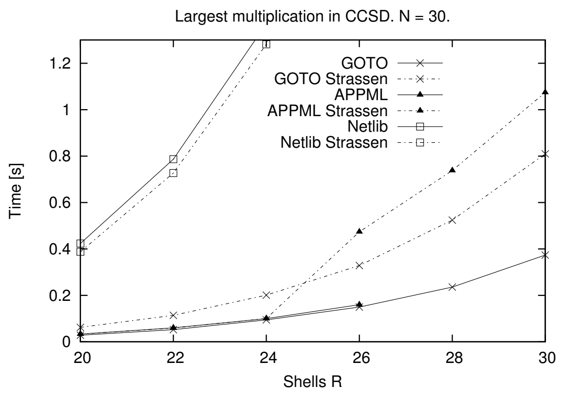
Compared to the transformation into a Hartree-Fock basis, which scales as $\max_\lambda \dim(\xi)^3$ when multiplying two square matrices of size $\dim(\xi)$, we see that the coupled-cluster



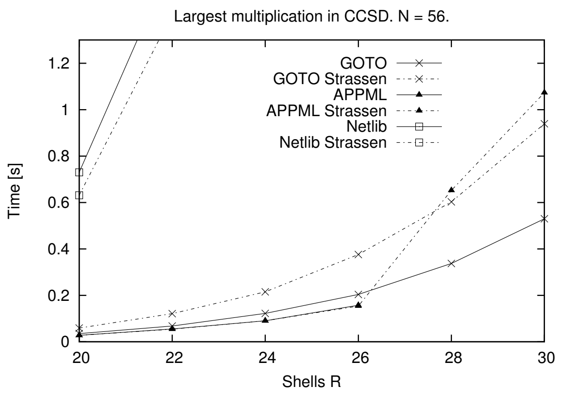
(a) $N = 2$



(b) $N = 12$

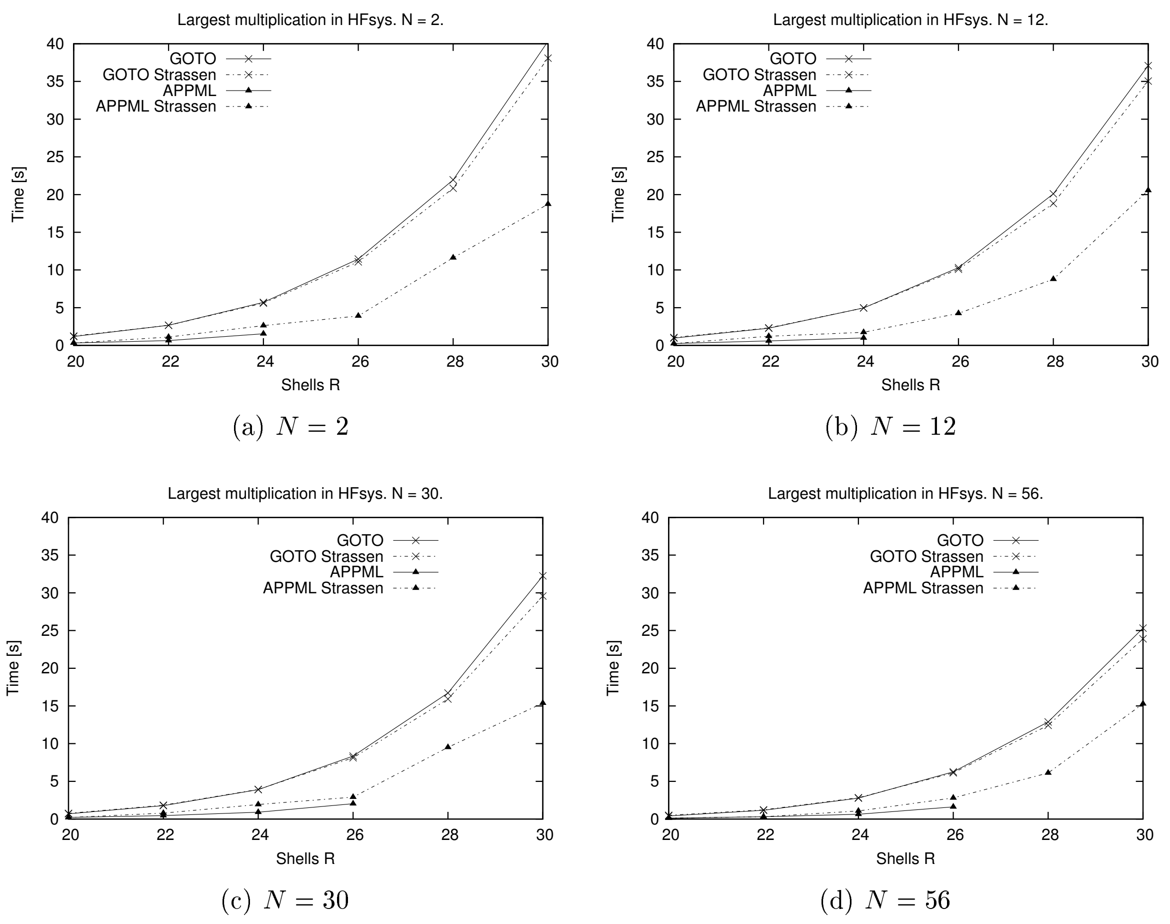


(c) $N = 30$



(d) $N = 56$

Figure 7.3: Time usage computing the widest channel in the most expensive term from coupled-cluster $\hat{T}_2$ equations.



(a) $N = 2$ (b) $N = 12$ (c) $N = 30$ (d) $N = 56$

Figure 7.4: Time usage computing the widest channel in the transformation to a Hartree-Fock basis.

calculation is no longer the bottleneck. In fig. 7.4 we study the time used to multiply two random matrices, whose size matches the largest matrices appearing in the HF-basis transformation. Now the power of exploiting graphic cards emerges. The APPML implementation is approximately 4 times faster than Goto blas. Unfortunately we once again see how unfavorable it is to apply a Strassen splitting to prevent running out of memory on the GPU. After such a splitting our GPU implementation runs at most twice as fast as Goto blas. Netlib's reference implementation is now up to 100 times slower than the best methods, and is not included in the plot.

For the narrow matrices in the amplitude equations, fig. 7.3, one level of Strassen proved a significant disadvantage. Considering time usage for the square matrices in fig. 7.4 we see a small speedup by applying Strassen's method. Only one Strassen recursion is applied in the figures, merely to see if there is a potential performance gain using such a method.

As a recursive method Strassen can be applied more than once, and we want to see how many levels of recursion that are ideal. Table 7.3(a) shows the time usage for GotoBLAS2 with different levels of Strassen when multiplying the large square matrices encountered for 2 electrons in 30 shells. One, or at most two, levels of recursion is advantageous, not

Table 7.3: Time used by Goto's and Netlib's blas multiplying two square matrices, for different levels of Strassen splitting.

(a) GOTO blas, matrix size 9412×9412.

Level	Time [s]	Size
0	40.04	9412
1	37.54	4706
2	37.42	2353
3	46.45	1176
4	52.18	588

(b) Netlib blas, matrix size 3764×3764.

Level	Time [s]	Size
0	68.31	3764
1	63.07	1882
2	47.09	941
3	41.58	470
4	40.02	235
5	39.85	117
6	35.42	58
7	36.36	29
8	53.55	14

even gaining the theoretical speedup of 12.5%. Our initial tests was using Netlib blas and showed a remarkable improvement. An example is shown in table 7.3(b), effectively reducing the time consumption by almost 50% for 3764×3764 matrices. Despite the good speedup Netlib sees with Strassen it is still more than a order of magnitude slower than the 'CLgemm' GPU implementation, or 14 times slower than GotoBLAS2.

7.2.2 Other implementations

The first Master's project on the topic of writing a C++ coupled-cluster program was completed in 2010 by M. P. Lohne [2]. His thesis involved a C++ library looping through all sums over elements stored in Blitz++ arrays. Elemental access through a linear-algebra library, such as Blitz++, proves inefficient but has the advantage of freeing the programmer of complications and bugs from memory-management. Lohne, who reached calculations up to 20 electrons and 10 shells, mentions how his code is neither optimized nor parallelized, but still he laid the ground stone for such a study, and concluded his thesis with a remark how future work may include more than 50 electrons in probably 16-20 shells.

M. H. Jørgensen [1] continued by optimizing Lohne's library, partly in collaboration with Y. M. Wang [3][1]. Their optimizations consisted mainly of replacing Blitz++ constructs with raw pointer syntax and employing a block-diagonal representation for the toughest parts of the interactions. An example of how successful the optimizations were is mentioned for 12 particles in a basis of 10 major oscillator shells. Here Lohne's program ran for 35 hours compared to their optimized library using approximately 4 minutes 30 seconds.

At the beginning of this thesis a decision was made to redesign the entire program to fit a different coding style. A linear-algebra library was reintroduced, this time Armadillo [24], and a more aggressive object orientation was conducted. Stronger encapsula-

[1]Jørgensen and Wang collaborated on optimizing a shared code base, but Wang studied a slightly different system.

Table 7.4: Wall time used for running coupled-cluster calculations with $\omega = 1.0$ and a standard interaction. Iterations stop after the energy converges to $1 \cdot 10^{-7}$. Time is measured either in seconds or [mm:ss]. Prev. is the previous program from [1], cur. is the current implementation running GPU accelerated.

(a) $N = 2$

R	Prev	Cur	Factor
10	2.9	0.9	3.2
12	8.9	2.5	3.6
14	25	6.4	3.9
16	1:01	15	4.1
18	2:12	33	4.0

(b) $N = 6$

R	Prev	Cur	Factor
10	27	1.4	19
12	1:24	3.4	25
14	3:29	8.8	24
16	7:11	19	23
18	16:45	43	23

(c) $N = 12$

R	Prev	Cur	Factor
10	5:13	3.6	87
12	17:45	9.1	117
14	51:20	19	162
16	129:05	39	199
18	287:50	1:13	237

tion was implemented, resulting in fewer classes and a cleaner syntax without loosing the flexibility. Different systems now actually follows a hierarchy, and we have successfully decoupled matrix multiplications from the main program as a separate module (class). Timing and debugging is also included, available by invoking simple switches or similar, although turned off by default.

We have compared the time consumption for a few runs in table 7.4. The run case of 12 particles in 10 shells now uses less than four seconds, 35000 times faster than the 35 hours Lohne's implementation used. Compared with the optimized code of Jørgensen and Wang we see 3-4 times speedup for two particles, more than 20 times for six particles, and some hundred times speedup for twelve electrons. Important here is how the speedup scales better with increasing number of particles, a consequence of reducing the dimensionality of all expressions in terms of channels and configurations.

In table 7.5 we have dissected the time our program uses in the different parts. Two tests are run, one for 20 particles in 20 oscillator shells and one for 56 particles in 30 shells. More particles would lead to convergence problems already at $\omega > 1.0$. Simulating 30 major oscillator shells, 930 basis functions, requires 30GB of memory for storing matrix elements. Counting also the transformed elements for a HF basis 60GB is required. In addition a remapped, temporary, copy of elements is created during simulations, resulting in almost 100GB of memory needed. To accommodate this need we have used a computer with 128GB of memory and two Intel Xeon E5620 running at 2.4GHz.

The largest test takes 160 minutes, where almost 100 of them is spent calculating the transformed HF basis. Matrix multiplication stands for more than 60% of wall time. We see that further optimizations should be targeted at the basis transformation first,

Table 7.5: Time used for different parts of the calculations. Run on a fat node with 128GB of memory. Matrix-multiplications is accelerated by the use of a GPU in the 'CLstrassen' class. As Goto blas was not successfully installed here, we can only compare to Netlib which is too slow for the largest case.

(a) $N = 20$, $R = 20$, $\omega = 1.0$

Part	Time [s]	%
Create system & read file	36.43	15.6%
Hartree-Fock calculation	20.43	8.7%
Hartree-Fock basis	99.93	42.7%
(Spent in matrix multiplication)	(79.39)	(33.9%)
Coupled cluster calculation	76.98	32.9%
(Spent in matrix multiplication)	(24.29)	(10.4%)
Total	234.05	100.0%

(b) $N = 56$, $R = 30$, $\omega = 5.0$

Part	Time [s]	%
Create system & read file	928	9.7%
Hartree-Fock calculation	449	4.7%
Hartree-Fock basis	5894	61.7%
(Spent in matrix multiplication)	(5034)	(52.7%)
Coupled cluster calculation	2281	23.9%
(Spent in matrix multiplication)	(897)	(9.4%)
Total	9556	100.0%

hereunder find a more efficient way of multiplication the large square matrices. For the coupled-cluster part, which now takes up a fourth of all execution time, there is evidence of a bottleneck other than matrix multiplication. Time usage for the coupled-cluster calculation is dissected even further in table 7.6, showing the five most time-consuming parts. Three parts stands out: the second term of the $\hat{T}_2$ equations, the second term in $[\mathcal{I}_{11}]$ as well as creating additional mappings and remapped interaction elements. The second term from $[\mathcal{I}_{11}]$, eq. (5.81), is known to depend on remapping of elements, a reason why it appears as slow. Further improvements in terms of efficiency will for the coupled-cluster part require a more efficient way of mapping as much as we need to speed up the matrix multiplications.

7.3 Convergence analysis

To be able to give a good estimate of the ground state energy we study how the energy converge as a function of the basis size. Higher shells included will contribute less to the ground state energy than lower shells. If increasing the basis from R to $R+1$ shells alters the energy by an amount $\Delta_R E_0$, we expect that a further increase of the basis will alter

Table 7.6: Thoughest terms in CCSD.

(a) $N = 20$, $R = 20$, $\omega = 1.0$.

Part	Time [s]
Init. maps	19.9
$[\mathcal{I}_{11}]$ #2	17.5
$\hat{T}_2$ #2	9.4
$\hat{T}_2$ #8	4.8
$[\mathcal{I}_{10}]$	3.7

(b) $N = 56$, $R = 30$, $\omega = 5.0$.

Part	Time [s]
$[\mathcal{I}_{11}]$ #2	656
Init. maps	455
$\hat{T}_2$ #2	329
$[\mathcal{I}_{10}]$	152
$\hat{T}_2$ #8	140

the energy by a smaller amount, i.e.

$$\Delta_{R+1}E_0 < \Delta_R E_0. \tag{7.10}$$

It is also believed that including triple excitations contributes less than double excitations, and so on, now yielding a framework to estimate how close we get to the real solution, to be obtained by full configuration interaction (FCI). FCI includes all possible excitations, finding the eigenvalues of the full Hamiltonian. For N particles distributed in n states the Hilbert space has the dimensionality

$$\dim(\mathcal{H}) = \binom{n}{N}. \tag{7.11}$$

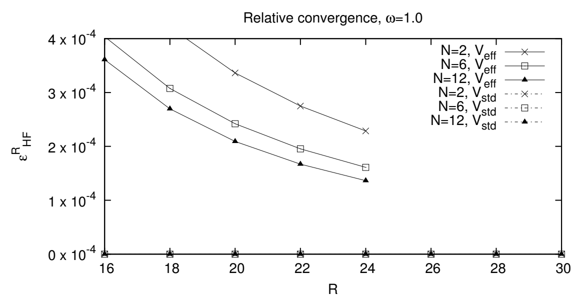
As an example 6 electrons distributed in 10 oscillator shell, that is $n = 110$, results in $\sim 2.1 \cdot 10^9$ possible determinants, already beyond the current limit for FCI solvers. For the largest system run here, we have 56 electrons and 30 oscillator shells, equivalent to 930 basis functions, yielding an enormous amount of $\sim 4.5 \cdot 10^{90}$ combinations. In this sense FCI calculations are restricted to a very limited basis size, although including all possible excitations. CC on the other hand allows a large basis size, but serves equations that are hard to derive and implement already for triples. For this reason, CC codes for quadruples or higher are seldom seen. For a comparison of CCSD and FCI, see section 7.5.1.

We label the energy obtained in a basis of R oscillator shells E_M^R, where M is the method used, either CCSD or HF. In order to study the convergence properties we define the relative difference in the energy at a specific number of shells to be,

$$\epsilon_M^R = \left| \frac{E_M^R - E_M^{R-2}}{E_M^{R_{\max}}} \right|, \tag{7.12}$$

in practice the number of significant figures the energy has converged to.

For two, six and twelve particles with $\omega = 1.0$ this measure of convergence is visualized in fig. 7.5. With few particles and a strong potential convergence is seen to be obtained quickly for the HF energy, fully converged for more than 10 shells, using the standard interaction. Only the standard interaction sees such a good convergence for the HF energy, as using V_{eff} the energy is still altered by the fourth figure for 24 shells, i.e. $\epsilon_{HF}^{24} > 10^{-4}$.


(a) HF

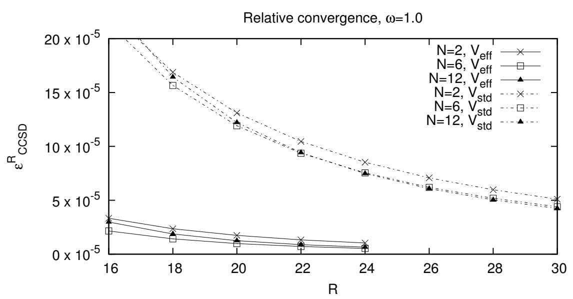
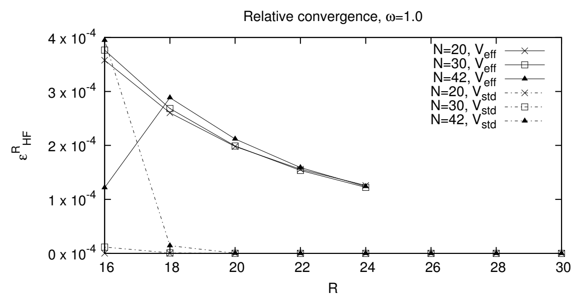

(b) CCSD

Figure 7.5: Relative convergence for $N = 2$, 6 and 12 for $\omega = 1.0$.


(a) HF

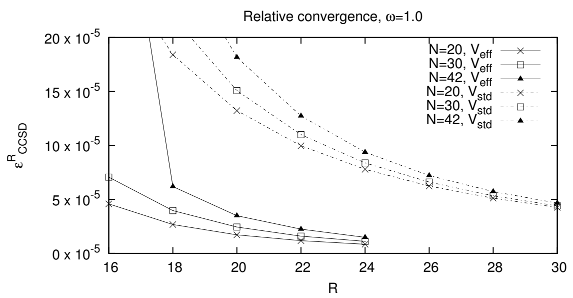

(b) CCSD

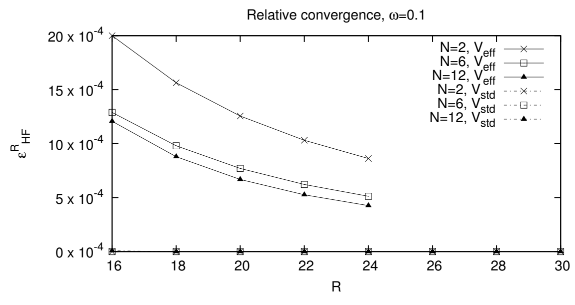
Figure 7.6: Relative convergence for $N = 20$, 30 and 42 for $\omega = 1.0$.

The CCSD energy, however, converges rapidly for an effective interaction to under $5 \cdot 10^{-5}$ before 16 shells. At least 30 shells are required to get the same convergence for the standard interaction.

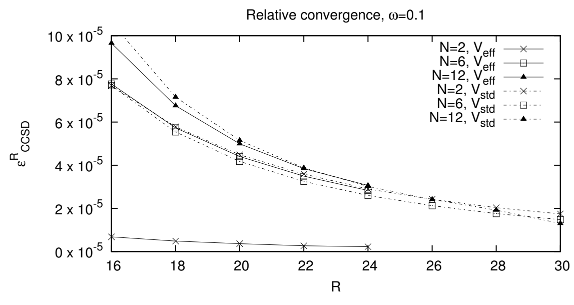
Raising the number of particles, now for 20, 30 and 42 electrons in fig. 7.6, we see the same pattern. The Hartree-Fock energy is fully converged when reaching $R = 20$ using the standard interaction. Still the effective interaction holds $\epsilon_{HF}^{24} > 10^{-4}$. The CCSD energy also behaves similar to the case with few particles, although converging slightly slower. It is remarkable to see how well coupled cluster performs up to 42 particles, with an error respective to the truncated basis in the fifth significant number.

When the frequency drops to $\omega = 0.1$, as in fig. 7.7, we still see a fully converged Hartree-Fock energy for the standard interaction, and a slightly slower convergence for the effective. Furthermore we observe that the effective interaction no longer leads to an improvement in convergence for CCSD. Except for the two-particle case, where an effective interaction yields the correct result anyhow, both approaches follow similar curves. Compared to a strongly confined system an effective interaction now performs worse and the standard interaction slightly better.

The last case we study is $N = 20$, 30 and 42 with a confinement strength of $\omega = 0.7$. This is the lowest potential strength that still converges for 42 electrons, albeit being far from the limit seen for $N = 20$. The convergence looks consistent with the other strongly

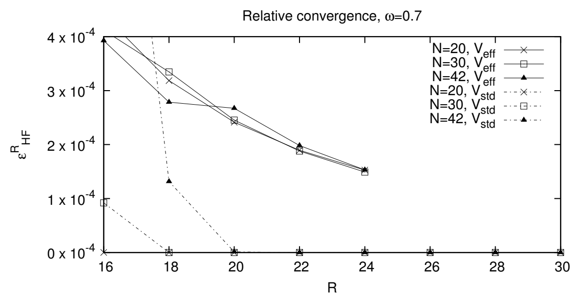


(a) HF

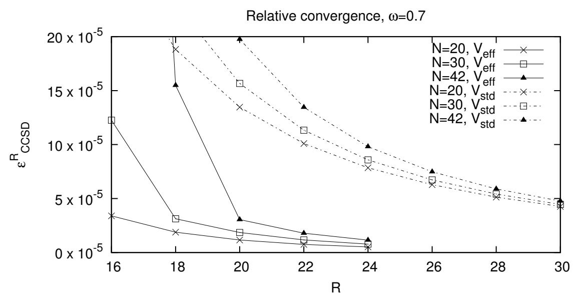


(b) CCSD

Figure 7.7: Relative convergence for $N = 2$, 6 and 12 for $\omega = 0.1$.



(a) HF



(b) CCSD

Figure 7.8: Relative convergence for $N = 20$, 30 and 42 for $\omega = 0.7$.

confined examples, approaching the same level of convergence for R_{max} although the slope appears to be somewhat steeper.

7.4 Lowering the frequency

In order to be able to reach the low oscillator frequencies, down to 0.2 or lower for systems with less than 20 electrons and $\omega \leq 1.0$ for larger systems, we need to apply a new strategy for the initial guess in the iterative schemes. No longer can we set the coefficient matrix in Hartree-Fock, C, nor the coupled-cluster amplitudes, t_i^a and t_{ij}^{ab}, to zero. In principle these values can be set to anything, and we get the energy eigenvalue of an unknown state, if converging. Such a procedure may succeed, although it has one major flaw. Setting the coefficients to zero reproduces the reference Slater determinant, the exact ground state for a non-interacting system, in the first guess. Considering the interacting system to be not too different from the uncorrelated, we assume the method to converge toward the ground state. Other values, if not carefully selected, can make the calculations to converge to other, excited, states.

For a tightly bound system we see no convergence issues, say for a strong potential $\omega = \omega_1$. The system is now determined mostly by the large single-particle contribution, leaving correlations to determine only a fraction of the total energy. Our zero-based guess for the coefficients now holds sufficient, and after a number of iterations we find an energy of $E_0(\omega_1)$ as well as coefficients $C(\omega_1)$, $t_i^a(\omega_1)$ and $t_{ij}^{ab}(\omega_1)$.

We now claim that both the energy and the coefficients vary only by a small amount whenever the frequency is also varied by a small change, Δ_ω, i.e.

$$\omega_2 \equiv \omega_1 - \Delta_\omega \to 0 \Rightarrow \begin{cases} |C(\omega_1) - C(\omega_2)| \to 0, \\ |t_i^a(\omega_1) - t_i^a(\omega_2)| \to 0, \\ |t_{ij}^{ab}(\omega_1) - t_{ij}^{ab}(\omega_2)| \to 0, \\ |E_0(\omega_1) - E_0(\omega_2)| \to 0. \end{cases} \tag{7.13}$$

Although (7.13) is left unproven here, we see clearly such a behavior in practice. Applied iteratively, we now store coefficients from one run, lower the frequency, then calculate the new coefficients and energy, taking the stored coefficients as initial values. Allowing us to reach previously unavailable frequencies, this technique also increases the CPU time by the amount of intermediate ω values needed. It is believed that we could push the lower level somewhat further by taking smaller intermediate steps, Δ_ω, but time constraints prevented us from doing so in this thesis.

In order to study the role of correlations for different number of particles when lowering the frequency, ω, we define the relative correlation energy

$$\chi = \left| \frac{E_{CCSD} - \langle \hat{H}_0 \rangle}{E_{CCSD}} \right|, \tag{7.14}$$

where $\langle \hat{H}_0 \rangle$ is the energy of an uncorrelated system. Uncorrelated energies for specific values of N and ω can be found in table 7.1. We naturally expect the two limits

$$\lim_{\omega \to 0} \chi = 1, \text{ and } \lim_{\omega \to \infty} \chi = 0, \tag{7.15}$$

as a tightly bound system is dominated by the high single-particle energy. Also, as ω tends to zero we obtain a free electron-gas with no external potential having all energy a consequence of correlations. The results for χ are shown in fig. 7.9. From this figure we see that interactions are increasingly important when the oscillator frequency is lowered. Furthermore, we see an increase in correlations whenever more particles are added to the system. Also studied are correlations with respect to the Hartree-Fock energy, defined similar to χ, except replacing $\langle \hat{H}_0 \rangle$ with E_{HF},

$$\Xi = \left| \frac{E_{CCSD} - E_{HF}}{E_{CCSD}} \right|. \tag{7.16}$$

If we recall how the Hartree-Fock method treats a many-body system as an uncorrelated system, only transforming the single-particle basis, we can interpret HF as solving an uncorrelated system where each electron is subjected to a mean field set up by the other electrons. As the system becomes less bound we see an increasing role of beyond mean-field corrections. Seen from fig. 7.10, the highest values are for few particles in a weakly

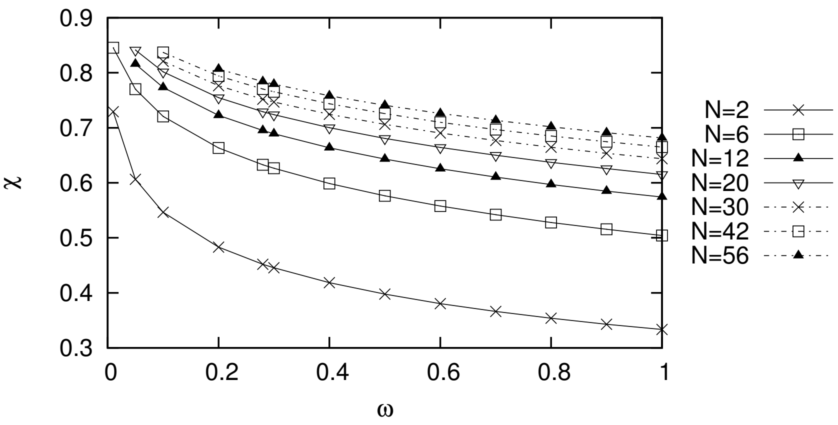


Figure 7.9: Relative correlation energy, χ, defined in eq. 7.14.

bound potential. We believe the findings here imply that a truncation in the number of excitations included in the coupled-cluster equations, here singles and doubles, affects the results less for more particles in a stronger potential. Adding more particles it is then possible to justify a mean-field approach, which serves a better scaling than more exact methods.

7.5 Comparison with other methods

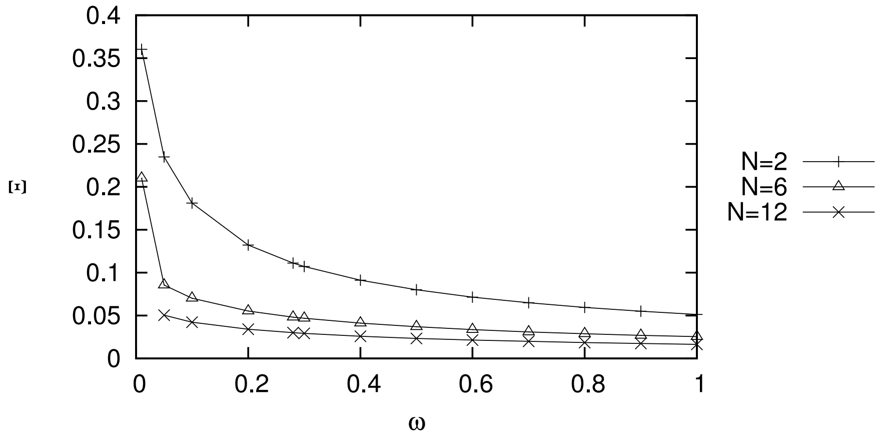
Coupled-cluster truncated to the level of singles and doubles will, as already discussed, not give us the exact ground-state eigenvalue of the Hamiltonian. We have seen how the energy changes beyond the Hartree-Fock calculation, but other methods is also of interest to compare with.

First considered are Monte Carlo methods. Variational Monte Carlo (VMC) results are here obtained by the library toffyrn::qvmc [25], and diffusion Monte Carlo (DMC) is supplied by a fellow master's student, K. R. Leikanger. VMC should be taken as a first approximation only. Taking any trial function Ψ_T and solving the integral,

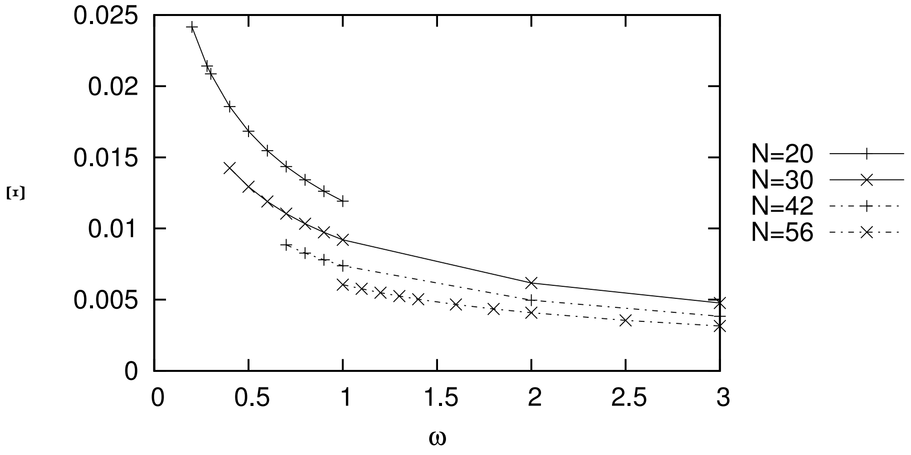
$$E\left[\Psi_T(\mathbf{a})\right] = \frac{\int \Psi_T^*(X,\mathbf{a})\mathbf{H}\Psi_T(X,\mathbf{a})dX}{\int \left|\Psi_T(X,\mathbf{a})\right|^2 dX} \geq E_0, \tag{7.17}$$

one would never underestimate the ground-state energy. The variables of freedom are denoted by X, whereas $\mathbf{a}$ is a set of variational parameters. By constructing a good trial function, and finding the optimal values for the variational parameters, one can get a result close to the real energy. In practice such a function is, unfortunately, hard to find. DMC on the other hand is considered quite accurate, but scales exponentially with the system size, and is thus not practical for larger systems.

Actual results are found in table 7.7. There is good agreement between all methods. VMC stands out as the least accurate method always overshooting the energy, a consequence of a too simple trial function. For two particles DMC and CCSD provide close to the exact result. Using an effective interaction the only error for CCSD now comes



(a) $N = 2, 6$ and 12



(b) $N = 20, 30, 42$ and 56

Figure 7.10: Relative correlation energy, Ξ, beyond the mean-field approximation, E_{HF}, as defined in eq. 7.16.

Table 7.7: Energies obtained by various methods. CCSD comes from our coupled cluster implementation using an effective interaction and 24 shells. VMC is found using the variational Monte Carlo library 'toffyrn::qvmc' [25]. DMC, diffusion Monte Carlo, results were supplied by K. R. Leikanger. The relative difference of CCSD and DMC is also listed.

N	ω	CCSD	VMC	DMC	Difference
	1.0	3.00021	3.0003(2)	3.00000(3)	0.01%
2	0.5	1.65987	1.6603(3)	1.65975(2)	0.01%
	0.1	0.44080	0.4419(2)	0.44078(2)	0.00%
	1.0	20.1734	20.194(2)	20.1597(2)	0.07%
6	0.5	11,8055	11.811(2)	11.7888(2)	0.14%
	0.1	3.5805	3.581(2)	3.5535(1)	0.76%
	1.0	65.7399	65.790(5)	65.700(1)	0.06%
12	0.5	39.2194	39.237(4)	39.159(1)	0.15%
	0.1	12.3497	12.377(4)	12.269(0)	0.66%

from the omitted elements from the double-sized energy-cut model space, as discussed in section 7.1.2. For more than two particles the relative difference is between 0.06 and 0.76 percent, the highest for low frequencies. As this is the same region where beyond mean-field correlation became important, the increasing error could point toward a need to include more than singles and doubles. Still the error is less than 1%.

7.5.1 Full configuration interaction

Table 7.8: Comparison with full configuration-interaction (FCI) results supplied by fellow master's student F. B. Olsen. The differing decimals are highlighted.

(a) $N = 2$

ω	R	CCSD	FCI	Δ
	2	3.1523280**9**	3.1523280**1**	$8 \cdot 10^{-8}$
1.0	10	3.0069**3724**	3.0069**4418**	$7 \cdot 10^{-6}$
	18	3.0034**0498**	3.0034**1108**	$6 \cdot 10^{-6}$
	30	3.00189643		$2 \cdot 10^{-3}$
	2	1.786913**64**	1.786913**53**	$1 \cdot 10^{-7}$
0.5	10	1.66353**533**	1.66353**801**	$3 \cdot 10^{-6}$
	18	1.661**59123**	1.661**60110**	$1 \cdot 10^{-5}$
	30	1.66078211		$8 \cdot 10^{-4}$
	2	0.5125**2002**	0.5125**1984**	$2 \cdot 10^{-7}$
0.1	10	0.441135**42**	0.441135**62**	$2 \cdot 10^{-7}$
	18	0.4409**5955**	0.4409**6013**	$6 \cdot 10^{-7}$
	30	0.44088382		$8 \cdot 10^{-5}$

(b) $N = 6$

ω	R	CCSD	FCI	Δ
	3	21.42**323172**	21.42**062118**	$3 \cdot 10^{-3}$
	4	20.4**2820553**	20.4**1589181**	$1 \cdot 10^{-2}$
1.0	5	20.3**3138914**	20.3**1679827**	$1 \cdot 10^{-2}$
	6	20.2**7324671**	20.2**5725213**	$2 \cdot 10^{-2}$
	30	20.18349343		$9 \cdot 10^{-2}$
	3	12.**90122455**	12.**89723259**	$4 \cdot 10^{-3}$
	4	12.0**5615503**	12.0**3695836**	$2 \cdot 10^{-2}$
0.5	5	11.9**3410621**	11.9**1311827**	$2 \cdot 10^{-2}$
	6	11.8**6345461**	11.8**4042104**	$2 \cdot 10^{-2}$
	30	11.81170403		$5 \cdot 10^{-2}$
	3	4.**208076509**	4.**149563782**	$6 \cdot 10^{-2}$
	4	3.**829020715**	3.**797449624**	$3 \cdot 10^{-2}$
0.1	5	3.6**66397817**	3.6**48168769**	$2 \cdot 10^{-2}$
	6	3.5**96913442**	3.5**68045420**	$3 \cdot 10^{-2}$
	30	3.582049703		$1 \cdot 10^{-2}$

(c) $N = 12$

ω	R	CCSD	FCI	Δ
1.0	4	70.3**2360770**	70.3**1288325**	$1 \cdot 10^{-2}$
	30	65.76965411		$5 \cdot 10^{0}$
0.5	4	43.**30832473**	43.**29209961**	$2 \cdot 10^{-2}$
	30	39.23828570		$4 \cdot 10^{0}$
0.1	4	15.**20219384**	15.**18780086**	$1 \cdot 10^{-2}$
	30	12.35523927		$3 \cdot 10^{0}$

7.6 Tables

7.6.1 Harmonic-oscillator basis

Standard interaction, $F = 1$, HO basis.

E_{CCSD}	10	12	14	16	18	20
1.0	3.0069	3.0055	3.0046	3.0039	3.0034	3.0030
0.9	2.7459	2.7446	2.7437	2.7431	2.7426	2.7423
0.8	2.4817	2.4805	2.4797	2.4792	2.4788	2.4784
0.7	2.2138	2.2128	2.2120	2.2115	2.2112	2.2109
0.6	1.9414	1.9405	1.9399	1.9395	1.9391	1.9389
0.5	1.6635	1.6628	1.6622	1.6619	1.6616	1.6614
0.4	1.3785	1.3779	1.3775	1.3772	1.3770	1.3769
0.3	1.0840	1.0835	1.0832	1.0830	1.0829	1.0828
0.28	1.0236	1.0232	1.0229	1.0227	1.0226	1.0225
0.2	0.7752	0.7749	0.7747	0.7746	0.7745	0.7745
0.1	0.4411	0.4411	0.4410	0.4410	0.4410	0.4409
0.05	0.2541	0.2541	0.2541	0.2541	0.2541	0.2541
0.01	DNC	DNC	0.5781	0.6304	DNC	DNC

Effective interaction, $F = 1$, HO basis.

E_{CCSD}	10	12	14	16	18	20
1.0	3.0009	3.0007	3.0005	3.0004	3.0003	3.0003
0.9	2.7403	2.7401	2.7400	2.7399	2.7398	2.7398
0.8	2.4766	2.4764	2.4763	2.4762	2.4762	2.4761
0.7	2.2093	2.2091	2.2090	2.2089	2.2089	2.2088
0.6	1.9375	1.9373	1.9372	1.9372	1.9371	1.9371
0.5	1.6602	1.6601	1.6600	1.6600	1.6599	1.6599
0.4	1.3759	1.3758	1.3758	1.3757	1.3757	1.3757
0.3	1.0821	1.0820	1.0820	1.0820	1.0820	1.0819
0.28	1.0218	1.0218	1.0218	1.0217	1.0217	1.0217
0.2	0.7741	0.7741	0.7740	0.7740	0.7740	0.7740
0.1	0.4408	0.4408	0.4408	0.4408	0.4408	0.4408
0.05	0.2541	0.2541	0.2541	0.2541	0.2541	0.2541
0.01	0.0738	0.0738	0.0738	0.0738	0.0738	0.0738

Standard interaction, $F = 2$, HO basis.

E_{CCSD}	10	12	14	16	18	20
1.0	20.2043	20.1947	20.1885	20.1843	20.1812	20.1789
0.9	18.6033	18.5944	18.5887	18.5848	18.5819	18.5798
0.8	16.9703	16.9621	16.9570	16.9534	16.9508	16.9488
0.7	15.2995	15.2922	15.2876	15.2844	15.2821	15.2803
0.6	13.5830	13.5767	13.5727	13.5699	13.5679	13.5664
0.5	11.8098	11.8045	11.8011	11.7988	11.7971	11.7958
0.4	9.9630	9.9588	9.9562	9.9543	9.9530	9.9520
0.3	8.0154	8.0125	8.0106	8.0093	8.0084	8.0077
0.28	7.6100	7.6073	7.6056	7.6044	7.6036	7.6030
0.2	5.9158	5.9142	5.9131	5.9124	5.9119	5.9115
0.1	3.5398	3.5394	3.5392	DNC	DNC	DNC
0.05	1.9742	DNC	DNC			
0.01	DNC					

Effective interaction, $F = 2$, HO basis.

E_{CCSD}	10	12	14	16	18	20
1.0	20.1660	20.1646	20.1639	20.1634	20.1631	20.1629
0.9	18.5676	18.5664	18.5658	18.5654	18.5652	18.5650
0.8	16.9375	16.9365	16.9360	16.9357	16.9355	16.9353
0.7	15.2699	15.2691	15.2687	15.2685	15.2683	15.2682
0.6	13.5570	13.5564	13.5561	13.5559	13.5558	13.5557
0.5	11.7877	11.7873	11.7871	11.7870	11.7869	11.7869
0.4	9.9454	9.9452	9.9450	9.9450	9.9449	9.9449
0.3	8.0028	8.0027	8.0027	8.0027	8.0027	8.0027
0.28	7.5985	7.5984	7.5984	7.5984	7.5984	7.5984
0.2	5.9089	5.9089	5.9088	5.9088	5.9088	5.9088
0.1	3.5392	3.5390	3.5389	3.5388	3.5388	3.5387
0.05	2.0215	2.0152	2.0104	DNC	DNC	DNC
0.01	0.6119	0.6021	0.5968			

Standard interaction, $F = 3$, HO basis.

E_{CCSD}	10	12	14	16	18	20
1.0	65.8065	65.7673	65.7443	65.7292	65.7186	65.7107
0.9	60.7696	60.7330	60.7116	60.6976	60.6878	60.6805
0.8	55.6144	55.5806	55.5610	55.5481	55.5391	55.5324
0.7	50.3192	50.2886	50.2709	50.2594	50.2513	50.2453
0.6	44.8551	44.8281	44.8125	44.8023	44.7952	44.7900
0.5	39.1810	39.1580	39.1447	39.1361	DNC	DNC
0.4	33.2355	33.2168	DNC	DNC		
0.3	26.9178	DNC				
0.28	25.5956					
0.2	DNC					

Effective interaction, $F = 3$, HO basis.

E_{CCSD}	10	12	14	16	18	20
1.0	65.6831	65.6745	65.6700	65.6674	65.6657	65.6645
0.9	60.6549	60.6470	60.6429	60.6405	60.6389	60.6378
0.8	55.5092	55.5020	55.4983	55.4961	55.4947	55.4937
0.7	50.2247	50.2183	50.2150	50.2132	50.2117	50.2108
0.6	44.7725	44.7670	44.7640	44.7623	44.7611	44.7603
0.5	39.1119	39.1072	39.1046	39.1030	39.1020	39.1013
0.4	33.1818	33.1777	33.1754	33.1741	DNC	DNC
0.3	26.8814	26.8782	DNC	DNC		
0.28	25.5628	DNC				
0.2	DNC					

7.6.2 Hartree-Fock basis

Standard interaction, $F = 1$.

E_{HF}	10	12	14	16	18	20	22	24	26	28	30
1.0	3.1619	3.1619	3.1619	3.1619	3.1619	3.1619	3.1619	3.1619	3.1619	3.1619	3.1619
0.9	2.8983	2.8983	2.8983	2.8983	2.8983	2.8983	2.8983	2.8983	2.8983	2.8983	2.8983
0.8	2.6311	2.6311	2.6311	2.6311	2.6311	2.6311	2.6311	2.6311	2.6311	2.6311	2.6311
0.7	2.3596	2.3596	2.3596	2.3596	2.3596	2.3596	2.3596	2.3596	2.3596	2.3596	2.3596
0.6	2.0829	2.0829	2.0829	2.0829	2.0829	2.0829	2.0829	2.0829	2.0829	2.0829	2.0829
0.5	1.7997	1.7997	1.7997	1.7997	1.7997	1.7997	1.7997	1.7997	1.7997	1.7997	1.7997
0.4	1.5080	1.5080	1.5080	1.5080	1.5080	1.5080	1.5080	1.5080	1.5080	1.5080	1.5080
0.3	1.2044	1.2044	1.2044	1.2044	1.2044	1.2044	1.2044	1.2044	1.2044	1.2044	1.2044
0.28	1.1417	1.1417	1.1417	1.1417	1.1417	1.1417	1.1417	1.1417	1.1417	1.1417	1.1417
0.2	0.8823	0.8823	0.8823	0.8823	0.8823	0.8823	0.8823	0.8823	0.8823	0.8823	0.8823
0.1	0.5256	0.5256	0.5256	0.5256	0.5256	0.5256	0.5256	0.5256	0.5256	0.5256	0.5256
0.05	0.3178	0.3178	0.3178	0.3178	0.3178	0.3178	0.3178	0.3178	0.3178	0.3178	0.3178
0.01	0.1028	0.1028	0.1028	0.1028	0.1028	0.1028	0.1028	0.1028	0.1028	0.1028	0.1028

Standard interaction, $F = 1$, HF basis.

E_{CCSD}	10	12	14	16	18	20	22	24	26	28	30
1.0	3.0069	3.0055	3.0046	3.0039	3.0034	3.0030	3.0027	3.0024	3.0022	3.0020	3.0019
0.9	2.7459	2.7446	2.7437	2.7431	2.7426	2.7423	2.7420	2.7418	2.7416	2.7414	2.7413
0.8	2.4817	2.4805	2.4797	2.4792	2.4788	2.4784	2.4782	2.4779	2.4778	2.4776	2.4775
0.7	2.2138	2.2128	2.2120	2.2115	2.2112	2.2109	2.2106	2.2104	2.2103	2.2101	2.2100
0.6	1.9414	1.9405	1.9399	1.9395	1.9391	1.9389	1.9387	1.9385	1.9384	1.9382	1.9381
0.5	1.6635	1.6628	1.6622	1.6619	1.6616	1.6614	1.6612	1.6611	1.6610	1.6609	1.6608
0.4	1.3785	1.3779	1.3775	1.3772	1.3770	1.3769	1.3767	1.3766	1.3765	1.3764	1.3764
0.3	1.0840	1.0835	1.0832	1.0830	1.0829	1.0828	1.0827	1.0826	1.0825	1.0825	1.0824
0.28	1.0236	1.0232	1.0229	1.0227	1.0226	1.0225	1.0224	1.0223	1.0222	1.0222	1.0221
0.2	0.7752	0.7749	0.7747	0.7746	0.7745	0.7745	0.7744	0.7744	0.7743	0.7743	0.7743
0.1	0.4411	0.4411	0.4410	0.4410	0.4410	0.4409	0.4409	0.4409	0.4409	0.4409	0.4409
0.05	0.2541	0.2541	0.2541	0.2541	0.2541	0.2541	0.2541	0.2541	0.2541	0.2541	0.2541
0.01	0.0738	0.0738	0.0738	0.6304	DNC	DNC	DNC	DNC	DNC	DNC	DNC

Effective interaction, $F = 1$.

E_{HF}	10	12	14	16	18	20	22	24
1.0	3.1429	3.1461	3.1484	3.1501	3.1514	3.1525	3.1533	3.1541
0.9	2.8795	2.8827	2.8849	2.8866	2.8879	2.8890	2.8898	2.8905
0.8	2.6126	2.6157	2.6179	2.6196	2.6209	2.6219	2.6227	2.6234
0.7	2.3415	2.3445	2.3467	2.3483	2.3496	2.3506	2.3514	2.3521
0.6	2.0653	2.0682	2.0703	2.0719	2.0732	2.0741	2.0749	2.0756
0.5	1.7826	1.7855	1.7875	1.7891	1.7903	1.7912	1.7920	1.7926
0.4	1.4916	1.4943	1.4963	1.4978	1.4989	1.4998	1.5006	1.5012
0.3	1.1888	1.1914	1.1933	1.1946	1.1957	1.1966	1.1973	1.1979
0.28	1.1264	1.1290	1.1308	1.1321	1.1332	1.1341	1.1347	1.1353
0.2	0.8681	0.8705	0.8721	0.8734	0.8744	0.8752	0.8758	0.8763
0.1	0.5138	0.5157	0.5171	0.5182	0.5190	0.5196	0.5202	0.5206
0.05	0.3083	0.3098	0.3109	0.3118	0.3124	0.3129	0.3134	0.3137
0.01	0.0971	0.0980	0.0987	0.0992	0.0996	0.1000	0.1002	0.1004

Effective interaction, $F = 1$, HF basis.

E_{CCSD}	10	12	14	16	18	20	22	24
1.0	3.0009	3.0007	3.0005	3.0004	3.0003	3.0003	3.0002	3.0002
0.9	2.7403	2.7401	2.7400	2.7399	2.7398	2.7398	2.7397	2.7397
0.8	2.4766	2.4764	2.4763	2.4762	2.4762	2.4761	2.4761	2.4761
0.7	2.2093	2.2091	2.2090	2.2089	2.2089	2.2088	2.2088	2.2088
0.6	1.9375	1.9373	1.9372	1.9372	1.9371	1.9371	1.9371	1.9371
0.5	1.6602	1.6601	1.6600	1.6600	1.6599	1.6599	1.6599	1.6599
0.4	1.3759	1.3758	1.3758	1.3757	1.3757	1.3757	1.3757	1.3757
0.3	1.0821	1.0820	1.0820	1.0820	1.0820	1.0819	1.0819	1.0819
0.28	1.0218	1.0218	1.0218	1.0217	1.0217	1.0217	1.0217	1.0217
0.2	0.7741	0.7741	0.7740	0.7740	0.7740	0.7740	0.7740	0.7740
0.1	0.4408	0.4408	0.4408	0.4408	0.4408	0.4408	0.4408	0.4408
0.05	0.2541	0.2541	0.2541	0.2541	0.2541	0.2541	0.2541	0.2541
0.01	0.0738	0.0738	0.0738	0.0738	0.0738	0.0738	0.0738	0.0738

Standard interaction, $F = 2$.

E_{HF}	10	12	14	16	18	20	22	24	26	28	30
1.0	20.7192	20.7192	20.7192	20.7192	20.7192	20.7192	20.7192	20.7192	20.7192	20.7192	20.7192
0.9	19.1108	19.1108	19.1108	19.1108	19.1108	19.1108	19.1108	19.1108	19.1108	19.1108	19.1108
0.8	17.4693	17.4693	17.4693	17.4693	17.4693	17.4693	17.4693	17.4693	17.4693	17.4693	17.4693
0.7	15.7884	15.7883	15.7883	15.7883	15.7883	15.7883	15.7883	15.7883	15.7883	15.7883	15.7883
0.6	14.0597	14.0597	14.0597	14.0597	14.0597	14.0597	14.0597	14.0597	14.0597	14.0597	14.0597
0.5	12.2713	12.2713	12.2713	12.2713	12.2713	12.2713	12.2713	12.2713	12.2713	12.2713	12.2713
0.4	10.4052	10.4052	10.4052	10.4052	10.4052	10.4052	10.4052	10.4052	10.4052	10.4052	10.4052
0.3	8.4314	8.4314	8.4314	8.4314	8.4314	8.4314	8.4314	8.4314	8.4314	8.4314	8.4314
0.28	8.0196	8.0196	8.0196	8.0196	8.0196	8.0196	8.0196	8.0196	8.0196	8.0196	8.0196
0.2	6.2935	6.2935	6.2935	6.2935	6.2935	6.2935	6.2935	6.2935	6.2935	6.2935	6.2935
0.1	3.8524	3.8524	3.8524	3.8524	3.8524	3.8524	3.8524	3.8524	3.8524	3.8524	3.8524
0.05	2.3791	2.3790	2.3790	2.3790	2.3790	2.3790	2.3790	2.3790	2.3790	2.3790	2.3790
0.01	0.7999	0.7947	0.7936	0.7935	0.7935	0.7935	0.7935	0.7935	0.7935	0.7935	

Standard interaction, $F = 2$, HF basis.

E_{CCSD}	10	12	14	16	18	20	22	24	26	28	30
1.0	20.2161	20.2063	20.2000	20.1957	20.1925	20.1901	20.1882	20.1867	20.1854	20.1844	20.1835
0.9	18.6162	18.6070	18.6012	18.5972	18.5943	18.5921	18.5903	18.5889	18.5877	18.5868	18.5860
0.8	16.9845	16.9761	16.9708	16.9671	16.9645	16.9624	16.9608	16.9596	16.9585	16.9576	16.9569
0.7	15.3153	15.3078	15.3030	15.2997	15.2973	15.2955	15.2940	15.2929	15.2920	15.2912	15.2905
0.6	13.6008	13.5942	13.5900	13.5871	13.5850	13.5834	13.5822	13.5812	13.5804	13.5797	13.5791
0.5	11.8301	11.8245	11.8209	11.8185	11.8167	11.8154	11.8143	11.8135	11.8128	11.8122	11.8117
0.4	9.9866	9.9820	9.9792	9.9773	9.9759	9.9748	9.9739	9.9733	9.9727	9.9723	9.9719
0.3	8.0434	8.0401	8.0380	8.0366	8.0356	8.0348	8.0342	8.0337	8.0333	8.0330	8.0327
0.28	7.6390	7.6360	7.6341	7.6328	7.6319	7.6312	7.6306	7.6302	7.6298	7.6295	7.6292
0.2	5.9501	5.9481	5.9469	5.9460	5.9454	5.9450	5.9446	5.9443	5.9441	5.9439	5.9438
0.1	3.5841	3.5835	3.5831	3.5828	3.5826	3.5825	3.5823	3.5822	3.5822	3.5821	3.5820
0.05	2.1776	2.1775	2.1774	2.1773	2.1773	2.1772	2.1772	2.1772	2.1772	2.1772	DNC
0.01	0.6267	0.6282	DNC	DNC	DNC	DNC	DNC	DNC	DNC	DNC	

Effective interaction, $F = 2$.

E_{HF}	10	12	14	16	18	20	22	24
1.0	20.6295	20.6461	20.6576	20.6659	20.6723	20.6773	20.6813	20.6847
0.9	19.0228	19.0391	19.0503	19.0585	19.0648	19.0697	19.0736	19.0769
0.8	17.3831	17.3991	17.4100	17.4181	17.4242	17.4290	17.4329	17.4361
0.7	15.7043	15.7199	15.7306	15.7384	15.7444	15.7491	15.7529	15.7560
0.6	13.9781	13.9933	14.0037	14.0112	14.0170	14.0216	14.0253	14.0283
0.5	12.1927	12.2073	12.2173	12.2246	12.2302	12.2346	12.2381	12.2411
0.4	10.3302	10.3441	10.3537	10.3606	10.3659	10.3701	10.3735	10.3763
0.3	8.3611	8.3741	8.3831	8.3896	8.3946	8.3985	8.4017	8.4043
0.28	7.9504	7.9632	7.9720	7.9785	7.9834	7.9872	7.9903	7.9929
0.2	6.2297	6.2416	6.2497	6.2556	6.2602	6.2637	6.2666	6.2689
0.1	3.7991	3.8091	3.8159	3.8208	3.8246	3.8275	3.8299	3.8319
0.05	2.3353	2.3437	2.3493	2.3534	2.3564	2.3589	2.3608	2.3624
0.01	0.7659	0.7704	0.7742	0.7772	0.7793	0.7810	0.7823	0.7834

Effective interaction, $F = 2$, HF basis.

E_{CCSD}	10	12	14	16	18	20	22	24
1.0	20.1766	20.1753	20.1746	20.1742	20.1739	20.1737	20.1735	20.1734
0.9	18.5792	18.5781	18.5775	18.5771	18.5769	18.5768	18.5766	18.5766
0.8	16.9502	16.9494	16.9489	16.9486	16.9484	16.9483	16.9482	16.9482
0.7	15.2841	15.2834	15.2831	15.2829	15.2827	15.2827	15.2826	15.2826
0.6	13.5729	13.5724	13.5722	13.5721	13.5721	13.5720	13.5720	13.5720
0.5	11.8057	11.8055	11.8055	11.8055	11.8055	11.8055	11.8055	11.8055
0.4	9.9662	9.9663	9.9664	9.9665	9.9665	9.9666	9.9666	9.9667
0.3	8.0275	8.0278	8.0280	8.0282	8.0283	8.0285	8.0285	8.0286
0.28	7.6241	7.6245	7.6247	7.6249	7.6251	7.6252	7.6253	7.6254
0.2	5.9392	5.9397	5.9400	5.9403	5.9405	5.9406	5.9408	5.9409
0.1	3.5786	3.5792	3.5796	3.5799	3.5801	3.5802	3.5804	3.5805
0.05	2.1750	2.1754	2.1757	2.1760	2.1761	2.1762	2.1763	2.1764
0.01	0.6720	0.6659	0.6610	0.6570	0.6538	0.6513	0.6492	0.6474

Standard interaction, $F = 3$.

E_{HF}	10	12	14	16	18	20	22	24	26	28	30
1.0	66.9120	66.9114	66.9113	66.9113	66.9113	66.9113	66.9113	66.9113	66.9113	66.9113	66.9113
0.9	61.8662	61.8656	61.8656	61.8656	61.8656	61.8656	61.8656	61.8656	61.8656	61.8656	61.8656
0.8	56.7002	56.6998	56.6998	56.6998	56.6998	56.6998	56.6998	56.6998	56.6998	56.6998	56.6998
0.7	51.3920	51.3918	51.3918	51.3917	51.3917	51.3917	51.3917	51.3917	51.3917	51.3917	51.3917
0.6	45.9115	45.9114	45.9114	45.9114	45.9114	45.9114	45.9114	45.9114	45.9114	45.9114	45.9114
0.5	40.2163	40.2162	40.2161	40.2161	40.2161	40.2161	40.2161	40.2161	40.2161	40.2161	40.2161
0.4	34.2426	34.2418	34.2418	34.2418	34.2418	34.2418	34.2418	34.2418	34.2418	34.2418	34.2418
0.3	27.8861	27.8827	27.8827	27.8827	27.8827	27.8827	27.8827	27.8827	27.8827	27.8827	27.8827
0.28	26.5544	26.5500	26.5500	26.5500	26.5500	26.5500	26.5500	26.5500	26.5500	26.5500	26.5500
0.2	20.9519	20.9399	20.9397	20.9397	20.9397	20.9397	20.9397	20.9397	20.9397	20.9397	20.9397
0.1	12.9699	12.9292	12.9248	12.9247	12.9247	12.9247	12.9247	12.9247	12.9247	12.9247	12.9247
0.05	8.1272	8.0531	8.0336	8.0305	8.0303	8.0303	8.0303	8.0303		8.0303	8.0303
0.01		2.8039	DNC	DNC	DNC	DNC	DNC	DNC			

Standard interaction, $F = 3$, HF basis.

E_{CCSD}	10	12	14	16	18	20	22	24	26	28	30
1.0	65.8880	65.8484	65.8250	65.8097	65.7989	65.7909	65.7847	65.7798	65.7758	65.7724	65.7697
0.9	60.8576	60.8205	60.7987	60.7845	60.7744	60.7670	60.7612	60.7566	60.7529	60.7499	60.7473
0.8	55.7099	55.6755	55.6555	55.6424	55.6332	55.6263	55.6210	55.6169	55.6135	55.6106	55.6083
0.7	50.4235	50.3923	50.3741	50.3622	50.3539	50.3477	50.3430	50.3392	50.3362	50.3336	50.3315
0.6	44.9697	44.9420	44.9259	44.9154	44.9080	44.9026	44.8984	44.8951	44.8924	44.8902	44.8883
0.5	39.3081	39.2842	39.2703	39.2614	39.2551	39.2504	39.2469	39.2440	39.2418	39.2399	39.2383
0.4	33.3779	33.3581	33.3467	33.3394	33.3343	33.3305	33.3276	33.3254	33.3235	33.3220	33.3207
0.3	27.0806	27.0642	27.0557	27.0502	27.0464	27.0436	27.0414	27.0398	27.0384	27.0373	27.0364
0.28	25.7634	25.7476	25.7396	25.7345	25.7310	25.7284	25.7264	25.7249	25.7236	25.7226	25.7217
0.2	20.2321	20.2164	20.2110	20.2075	20.2051	20.2034	20.2021	20.2011	20.2002	20.1996	20.1990
0.1	12.3887	12.3628	12.3596	12.3583	12.3574	12.3568	12.3563	12.3559	DNC	12.3554	12.3552
0.05	DNC	7.6135	7.6079	7.6074	7.6071	7.6070	7.6069	7.6068		DNC	DNC
0.01		DNC									

Effective interaction, $F = 3$.

E_{HF}	10	12	14	16	18	20	22	24
1.0	66.6596	66.7106	66.7445	66.7686	66.7867	66.8006	66.8118	66.8209
0.9	61.6186	61.6688	61.7021	61.7258	61.7435	61.7572	61.7681	61.7771
0.8	56.4580	56.5073	56.5400	56.5632	56.5805	56.5939	56.6046	56.6133
0.7	51.1558	51.2042	51.2361	51.2587	51.2756	51.2887	51.2991	51.3076
0.6	45.6821	45.7295	45.7606	45.7825	45.7989	45.8116	45.8217	45.8299
0.5	39.9948	40.0409	40.0709	40.0922	40.1080	40.1202	40.1299	40.1379
0.4	34.0304	34.0744	34.1033	34.1237	34.1388	34.1504	34.1597	34.1673
0.3	27.6849	27.7253	27.7528	27.7720	27.7862	27.7972	27.8059	27.8131
0.28	26.3556	26.3950	26.4221	26.4410	26.4551	26.4659	26.4745	26.4815
0.2	20.7635	20.7958	20.8213	20.8390	20.8521	20.8622	20.8701	20.8766
0.1	12.7940	12.8039	12.8243	12.8398	12.8511	12.8597	12.8665	12.8720
0.05	7.9512	7.9374	7.9469	7.9593	7.9692	7.9766	7.9824	7.9870
0.01	2.6952	2.6688	2.6585	DNC	DNC	DNC	DNC	DNC

Effective interaction, $F = 3$, HF basis.

E_{CCSD}	10	12	14	16	18	20	22	24
1.0	65.7552	65.7484	65.7449	65.7430	65.7417	65.7409	65.7403	65.7399
0.9	60.7325	60.7265	60.7235	60.7219	60.7209	60.7202	60.7197	60.7194
0.8	55.5931	55.5880	55.5856	55.5843	55.5835	55.5830	55.5826	55.5823
0.7	50.3158	50.3119	50.3100	50.3091	50.3085	50.3081	50.3079	50.3077
0.6	44.8722	44.8695	44.8682	44.8676	44.8672	44.8670	44.8669	44.8669
0.5	39.2218	39.2203	39.2197	39.2195	39.2194	39.2194	39.2194	39.2194
0.4	33.3043	33.3039	33.3039	33.3041	33.3042	33.3044	33.3046	33.3047
0.3	27.0211	27.0209	27.0216	27.0222	27.0226	27.0230	27.0233	27.0235
0.28	25.7069	25.7066	25.7074	25.7081	25.7085	25.7089	25.7092	25.7095
0.2	20.1881	20.1854	20.1867	20.1877	20.1884	20.1889	20.1893	20.1896
0.1	12.3612	12.3453	12.3462	12.3474	12.3482	12.3488	12.3493	12.3497
0.05	7.6379	7.6045	7.6009	7.6016	7.6024	7.6029	7.6033	7.6036
0.01	DNC	DNC	DNC					

Standard interaction, $F = 4$.

E_{HF}	10	12	14	16	18	20	22	24	26	28	30
1.0	158.0177	158.0050	158.0043	158.0043	158.0043	158.0043	158.0043	158.0043	158.0043	158.0043	158.0043
0.9	146.3052	146.2858	146.2853	146.2853	146.2853	146.2853	146.2853	146.2853	146.2853	146.2853	146.2853
0.8	134.2944	134.2650	134.2647	134.2647	134.2647	134.2647	134.2647	134.2647	134.2647	134.2647	134.2647
0.7	121.9318	121.8875	121.8874	121.8874	121.8874	121.8874	121.8874	121.8874	121.8874	121.8874	121.8874
0.6	109.1452	109.0790	109.0788	109.0787	109.0787	109.0787	109.0787	109.0787	109.0787	109.0787	109.0787
0.5	95.8333	95.7346	95.7328	95.7327	95.7327	95.7327	95.7327	95.7327	95.7327	95.7327	95.7327
0.4	81.8448	81.6973	81.6903	81.6903	81.6903	81.6903	81.6903	81.6903	81.6903	81.6903	81.6903
0.3	66.9329	66.7128	66.6905	66.6901	66.6901	66.6901	66.6901	66.6901	66.6901	66.6901	66.6901
0.28	63.8056	63.5673	63.5394	63.5388	63.5388	63.5388	63.5388	63.5388	63.5388	63.5388	63.5388
0.2	50.6360	50.3098	50.2455	50.2404	50.2403	50.2403	50.2403	50.2403	50.2403	50.2403	50.2403
0.1	31.8231	31.3597	DNC	DNC	DNC	DNC	DNC	DNC	DNC	DNC	DNC
0.05	20.3309	19.8207									
0.01	7.6477	7.2500									

Standard interaction, $F = 4$, HF basis.

E_{CCSD}	10	12	14	16	18	20	22	24	26	28	30
1.0	156.3659	156.2363	156.1671	156.1243	156.0956	156.0750	156.0594	156.0473	156.0375	156.0296	156.0229
0.9	144.6758	144.5438	144.4783	144.4382	144.4114	144.3921	144.3776	144.3663	144.3572	144.3498	144.3436
0.8	132.6918	132.5544	132.4931	132.4560	132.4312	132.4134	132.4000	132.3896	132.3813	132.3745	132.3688
0.7	120.3619	120.2145	120.1580	120.1241	120.1015	120.0853	120.0732	120.0638	120.0562	120.0501	120.0450
0.6	107.6163	107.4517	107.4006	107.3702	107.3501	107.3357	107.3249	107.3165	107.3099	107.3044	107.2999
0.5	94.3572	94.1641	94.1187	94.0922	94.0747	94.0623	94.0530	94.0458	94.0401	94.0354	94.0315
0.4	80.4391	80.2004	80.1590	80.1368	80.1223	80.1120	80.1044	80.0985	80.0939	80.0901	80.0869
0.3	65.6260	65.3157	65.2712	65.2538	65.2426	65.2348	65.2290	65.2246	65.2210	65.2182	65.2158
0.28	62.5234	62.1948	62.1476	62.1312	62.1207	62.1134	62.1080	62.1038	62.1006	62.0980	62.0958
0.2	49.4774	49.0595	48.9880	48.9741	48.9667	48.9616	48.9579	48.9550	48.9528	48.9510	48.9495
0.1	30.9173	30.3592									
0.05	19.6454	19.0378									
0.01	DNC	DNC									

Effective interaction, $F = 4$.

E_{HF}	10	12	14	16	18	20	22	24
1.0	157.4356	157.5613	157.6437	157.7002	157.7413	157.7725	157.7971	157.8170
0.9	145.7308	145.8502	145.9316	145.9873	146.0277	146.0584	146.0825	146.1020
0.8	133.7278	133.8383	133.9188	133.9735	134.0131	134.0432	134.0668	134.0858
0.7	121.3727	121.4707	121.5501	121.6037	121.6425	121.6719	121.6950	121.7135
0.6	108.5932	108.6731	108.7512	108.8036	108.8414	108.8700	108.8924	108.9105
0.5	95.2872	95.3407	95.4164	95.4676	95.5043	95.5320	95.5537	95.5711
0.4	81.3018	81.3165	81.3871	81.4369	81.4724	81.4991	81.5199	81.5365
0.3	66.3877	66.3449	66.4031	66.4512	66.4852	66.5106	66.5304	66.5462
0.28	63.2588	63.2016	63.2557	63.3032	63.3369	63.3621	63.3816	63.3972
0.2	50.0767	49.9505	49.9770	50.0203	50.0526	50.0765	50.0949	50.1095
0.1	31.2212	30.9875	DNC	DNC	DNC	DNC	DNC	DNC
0.05	19.6852	19.4099						
0.01	7.0258	6.7813						

Effective interaction, $F = 4$, HF basis.

E_{CCSD}	10	12	14	16	18	20	22	24
1.0	156.0128	155.9868	155.9740	155.9669	155.9627	155.9601	155.9582	155.9569
0.9	144.3378	144.3077	144.2962	144.2900	144.2864	144.2842	144.2826	144.2815
0.8	132.3697	132.3330	132.3230	132.3178	132.3148	132.3130	132.3118	132.3109
0.7	120.0566	120.0091	120.0009	119.9969	119.9946	119.9932	119.9923	119.9917
0.6	107.3285	107.2641	107.2580	107.2550	107.2535	107.2526	107.2520	107.2516
0.5	94.0870	93.9963	93.9921	93.9904	93.9895	93.9891	93.9889	93.9889
0.4	80.1857	80.0545	80.0503	80.0499	80.0499	80.0500	80.0502	80.0504
0.3	65.3859	65.1934	65.1827	65.1836	65.1844	65.1851	65.1857	65.1862
0.28	62.2851	62.0772	62.0634	62.0646	62.0656	62.0664	62.0671	62.0676
0.2	49.2407	48.9599	48.9221	48.9229	48.9246	48.9258	48.9268	48.9276
0.1	30.6514	30.2700						
0.05	19.3247	18.9322						
0.01	DNC	DNC						

Standard interaction, $F = 5$.

E_{HF}	10	12	14	16	18	20	22	24	26	28	30
3.0	705.5082	705.2159	705.2141	705.2131	705.2129	705.2129	705.2128	705.2128	705.2128	705.2128	705.2128
2.0	519.8257	519.2706	519.2537	519.2528	519.2527	519.2526	519.2525	519.2525	519.2525	519.2525	519.2525
1.0	313.1707	312.0104	311.8639	311.8603	311.8600	311.8599	311.8599	311.8599	311.8599	311.8599	311.8599
0.9	290.4315	289.1731	288.9904	288.9836	288.9834	288.9834	288.9834	288.9834	288.9834	288.9834	288.9834
0.8	267.0953	265.7286	265.5002	265.4877	265.4876	265.4876	265.4876	265.4876	265.4876	265.4876	265.4876
0.7	243.0553	241.5688	241.2824	241.2602	241.2602	241.2602	241.2602	241.2602	241.2602	241.2602	241.2602
0.6	218.1670	216.5484	216.1879	216.1490	216.1487	216.1487	216.1487	216.1487	216.1487	216.1487	216.1487
0.5	192.2256	190.4624	190.0072	189.9396	189.9376	189.9376	189.9376	189.9376	189.9376	189.9376	189.9376
0.4	164.9234	163.0048	162.4278	162.3113	162.3030	162.3030	162.3030	162.3030	162.3030	162.3030	162.3030
0.3	135.7536	133.6758	132.9437	DNC	DNC	DNC	DNC	DNC	DNC	DNC	DNC
0.28	129.6246	127.5162	126.7486								
0.2	103.7538	101.5381	100.6147								
0.1	66.5178	64.2883	63.1695								
0.05	43.4390	41.4092	40.2664								

Standard interaction, $F = 5$, HF basis.

E_{CCSD}	10	12	14	16	18	20	22	24	26	28	30
3.0	703.0377	702.3983	702.1784	702.0300	701.9251	701.8483	701.7902	701.7452	701.7094	701.6802	701.6560
2.0	517.4339	516.5264	516.3080	516.1745	516.0827	516.0168	515.9677	515.9300	515.9001	515.8758	515.8556
1.0	311.0017	309.4549	309.1373	309.0300	308.9623	308.9157	308.8818	308.8559	308.8355	308.8190	308.8054
0.9	288.3054	286.6549	286.3044	286.1991	286.1352	286.0915	286.0596	286.0354	286.0164	286.0010	285.9883
0.8	265.0196	263.2551	262.8617	262.7570	262.6973	262.6566	262.6271	262.6047	262.5871	262.5729	262.5612
0.7	241.0391	239.1501	238.7004	238.5935	238.5385	238.5012	238.4742	238.4537	238.4377	238.4247	238.4141
0.6	216.2221	214.1981	213.6744	213.5602	213.5102	213.4765	213.4522	213.4339	213.4196	213.4081	213.3986
0.5	190.3676	188.2003	187.5791	187.4476	187.4023	187.3726	187.3514	187.3355	187.3231	187.3131	187.3050
0.4	163.1735	160.8597	160.1105	159.9433	159.9002	159.8751	159.8574	159.8441	159.8338	159.8255	159.8188
0.3	134.1423	131.6933	130.7786								
0.28	128.0461	125.5740	124.6214								
0.2	102.3303	99.7955	98.6786								
0.1	65.3814	62.9388	61.6347								
0.05	DNC	DNC	DNC								

Effective interaction, $F = 5$.

E_{HF}	10	12	14	16	18	20	22	24
3.0	704.1042	704.1860	704.3890	704.5240	704.6209	704.6938	704.7507	704.7963
2.0	518.4603	518.2977	518.4801	518.6085	518.7002	518.7688	518.8222	518.8650
1.0	311.8164	311.1098	311.1723	311.2896	311.3732	311.4352	311.4831	311.5212
0.9	289.0720	288.2790	288.3098	288.4235	288.5062	288.5673	288.6144	288.6518
0.8	265.7280	264.8399	264.8311	264.9392	265.0209	265.0811	265.1273	265.1640
0.7	241.6767	240.6840	240.6252	240.7244	240.8051	240.8642	240.9095	240.9454
0.6	216.7722	215.6646	215.5427	215.6274	215.7067	215.7647	215.8090	215.8440
0.5	190.8081	189.5752	189.3729	189.4335	189.5104	189.5672	189.6104	189.6445
0.4	163.4743	162.1063	161.8018	161.8220	161.8931	161.9486	161.9906	162.0235
0.3	134.2609	132.7525	132.3184	132.2707	DNC	DNC	DNC	DNC
0.28	128.1216	126.5854	126.1218	126.0557				
0.2	102.2044	100.5648	99.9710	DNC				
0.1	64.9165	63.2207	62.4520					
0.05		40.2934	39.4613					

Effective interaction, $F = 5$, HF basis.

E_{CCSD}	10	12	14	16	18	20	22	24
3.0	701.8751	701.5868	701.5493	701.5173	701.4929	701.4749	701.4617	701.4519
2.0	516.3523	515.8103	515.7623	515.7338	515.7133	515.6990	515.6889	515.6816
1.0	310.0024	308.8848	308.7310	308.7092	308.6969	308.6894	308.6845	308.6810
0.9	287.3106	286.1030	285.9179	285.8955	285.8847	285.8782	285.8739	285.8710
0.8	264.0271	262.7219	262.4967	262.4722	262.4630	262.4575	262.4539	262.4515
0.7	240.0459	238.6354	238.3586	238.3294	238.3219	238.3175	238.3147	238.3128
0.6	215.2235	213.7012	213.3576	213.3187	213.3130	213.3098	213.3078	213.3065
0.5	189.3567	187.7187	187.2890	187.2313	187.2268	187.2248	187.2237	187.2231
0.4	162.1401	160.3881	159.8479	159.7549	159.7490	159.7484	159.7482	159.7483
0.3	133.0709	131.2199	130.5418	130.3827				
0.28	126.9647	125.0978	124.3890	124.2111				
0.2	101.1987	99.2928	98.4573					
0.1	DNC	62.3351	61.3827					
0.05		DNC	DNC					

Standard interaction, $F = 6$.

E_{HF}	10	12	14	16	18	20	22	24	26	28	30
3.0	1231.3362	1228.7193	1228.5692	1228.5685	1228.5679	1228.5677	1228.5676	1228.5676	1228.5675	1228.5675	1228.5675
2.0	911.9565	908.0358	907.5758	907.5651	907.5641	907.5640	907.5639	907.5638	907.5638	907.5638	907.5638
1.0	555.3932	549.3884	547.9075	547.6913	547.6834	547.6832	547.6831	547.6831	547.6831	547.6831	547.6831
0.9	516.0557	509.7985	508.1282	507.8437	507.8279	507.8278	507.8278	507.8278	507.8278	507.8278	507.8278
0.8	475.6536	469.1422	467.2581	466.8846	466.8542	466.8542	466.8542	466.8542	466.8542	466.8542	466.8542
0.7	433.9925	427.2305	425.1060	424.6166	424.5607	424.5601	424.5601	424.5601	424.5601	424.5601	424.5601
0.6	390.8076	383.8079	381.4147	DNC	DNC	DNC	DNC	DNC	DNC	DNC	DNC
0.5	345.7212	338.5124	335.8230								
0.4	298.1591	290.7981	287.7903								
0.3	247.1651	239.7646	236.4342								
0.28	236.4193	229.0335	225.6415								
0.2	190.8948	183.6960	180.0920								
0.1	124.6446	118.2783	114.6498								
0.05		77.5617	74.3017								
0.01		DNC	DNC								

Standard interaction, $F = 6$, HF basis.

E_{CCSD}	10	12	14	16	18	20	22	24	26	28	30
3.0	1228.5491	1225.0927	1224.5123	1224.2289	1224.0304	1223.8853	1223.7763	1223.6922	1223.6258	1223.5724	1223.5285
2.0	909.3486	904.5453	903.6297	903.3545	903.1726	903.0440	902.9494	902.8774	902.8213	902.7765	902.7397
1.0	553.1292	546.2805	544.2634	543.7953	543.6410	543.5423	543.4732	543.4222	543.3831	543.3520	543.3267
0.9	513.8444	506.7619	504.5465	504.0063	503.8502	503.7562	503.6909	503.6430	503.6062	503.5770	503.5533
0.8	473.5007	466.1880	463.7516	463.1161	462.9524	462.8637	462.8026	462.7578	462.7236	462.6965	462.6746
0.7	431.9049	424.3723	421.6916	420.9311	420.7491	420.6660	420.6094	420.5682	420.5368	420.5120	420.4919
0.6	388.7941	3 81.0625	378.1152								
0.5	343.7933	335.9015	332.6689								
0.4	296.3329	288.3507	284.8252								
0.3	245.4648	237.5219	233.7218								
0.28	234.7483	226.8383	222.9902								
0.2	189.3616	181.7231	177.7344								
0.1	DNC	116.7086	112.8385								
0.05		76.3210	72.9248								
0.01											

Effective interaction, $F = 6$.

E_{HF}	10	12	14	16	18	20	22	24
3.0	1228.4662	1226.8409	1227.1277	1227.3873	1227.5658	1227.6968	1227.7974	1227.8769
2.0	909.0297	906.2031	906.2118	906.4563	906.6271	906.7518	906.8468	906.9217
1.0	552.2744	547.5249	546.6254	546.6919	546.8498	546.9657	547.0528	547.1207
0.9	512.9013	507.9176	506.8492	506.8576	507.0086	507.1238	507.2098	507.2768
0.8	472.4586	467.2384	465.9789	465.9114	466.0503	466.1648	466.2499	466.3158
0.7	430.7513	425.2967	423.8217	423.6552	423.7733	423.8867	423.9708	424.0357
0.6	387.5137	381.8347	380.1183	379.8231	DNC	DNC	DNC	DNC
0.5	342.3674	336.4875	334.5036	DNC				
0.4	294.7381	288.7055	286.4314					
0.3	243.6708	237.5837	235.0093					
0.28	232.9101	226.8320	224.1989					
0.2	187.3283	181.4041	178.5581					
0.1		115.8748	112.9496					
0.05			72.5333					
0.01								

Effective interaction, $F = 6$, HF basis.

E_{CCSD}	10	12	14	16	18	20	22	24
3.0	1225.9815	1223.5313	1223.3611	1223.3115	1223.2686	1223.2343	1223.2082	1223.1883
2.0	906.7604	903.0848	902.6089	902.5544	902.5146	902.4852	902.4639	902.4483
1.0	550.4001	544.8789	543.4257	543.1884	543.1548	543.1358	543.1236	543.1155
0.9	511.0861	505.3547	503.7276	503.4261	503.3886	503.3714	503.3606	503.3536
0.8	470.7085	464.7701	462.9498	462.5641	462.5175	462.5025	462.4933	462.4873
0.7	429.0735	422.9369	420.9039	420.4086	420.3435	420.3307	420.3231	420.3183
0.6	385.9173	379.6005	377.3363	376.6996				
0.5	340.8642	334.4001	331.8901					
0.4	293.3442	286.7924	284.0311					
0.3	242.4091	235.8822	232.8853					
0.28	231.6781	225.1788	222.1401					
0.2	DNC	179.9709	176.8038					
0.1		DNC	111.7170					
0.05			DNC					
0.01								

Standard interaction, $F = 7$.

E_{HF}	10	12	14	16	18	20	22	24	26	28	30
5.0	2930.2027	2910.7234	2908.0220	2907.8927	2907.8912	2907.8906	2907.8903	2907.8902	2907.8902	2907.8901	2907.8901
4.0	2477.3370	2456.4042	2452.8519	2452.5825	2452.5805	2452.5794	2452.5792	2452.5791	2452.5791	2452.5790	2452.5790
3.0	2002.7368	1980.1845	1975.4082	1974.8343	1974.8180	1974.8170	1974.8168	1974.8167	1974.8166	1974.8166	1974.8166
2.5	1754.0613	1730.6595	1725.0645	1724.2140	1724.1696	1724.1690	1724.1687	1724.1686	1724.1685	1724.1685	1724.1685
2.0	1494.7928	1470.5630	1463.9581	1462.6794	1462.5641	1462.5629	1462.5625	1462.5624	1462.5623	1462.5623	1462.5623
1.8		1362.7550	1355.6810	1354.1685	1354.0009	1353.9975	1353.9971	1353.9970	1353.9970	1353.9969	1353.9969
1.6		1252.2857	1244.7002	1242.9055	1242.6620	1242.6534	1242.6531	1242.6530	1242.6529	1242.6529	1242.6529
1.4		1138.6746	1130.5328	1128.3960	1128.0424	1128.0218	1128.0216	1128.0215	1128.0215	1128.0214	1128.0214
1.3		1080.4965	1072.0594	1069.7248	1069.2985	1069.2673	1069.2671	1069.2670	1069.2669	1069.2669	1069.2669
1.2		1021.2709	1012.5274	1009.9745	1009.4602	1009.4136	1009.4132	1009.4130	1009.4130	1009.4130	1009.4130
1.1		960.8731	951.8130	949.0189	948.3983	948.3292	948.3280	948.3279	948.3278	948.3278	948.3278
1.0		899.1507	889.7655	886.7047	885.9553	DNC	DNC	DNC	DNC	DNC	DNC
0.9		835.9134	826.1973	822.8421	DNC						
0.8		770.9184	760.8705	757.1905							
0.7		703.8469	693.4743	689.4376							
0.6			623.5884	619.1627							
0.5			550.6180	545.7742							
0.4			473.6663	468.3870							
0.3			391.2526	385.5537							
0.28			373.9067	368.1320							
0.2			300.5487	294.5424							
0.1				DNC							

Standard interaction, $F = 7$, HF basis.

E_{CCSD}	10	12	14	16	18	20	22	24	26	28	30
5.0	2927.2937	2906.4211	2902.8719	2902.2253	2901.8536	2901.5777	2901.3664	2901.2008	2901.0686	2900.9612	2900.8728
4.0	2474.4919	2452.1866	2447.7499	2446.9646	2446.6007	2446.3340	2446.1320	2445.9749	2445.8504	2445.7499	2445.6675
3.0	1999.9767	1976.1035	1970.3971	1969.2990	1968.9324	1968.6786	1968.4892	1968.3438	1968.2297	1968.1384	1968.0640
2.5	1751.3547	1726.6785	1720.1314	1718.7433	1718.3565	1718.1113	1717.9303	1717.7927	1717.6854	1717.6001	1717.5309
2.0	DNC	1466.7153	1459.1448	1457.3034	1456.8544	1456.6190	1456.4486	1456.3207	1456.2221	1456.1442	1456.0813
1.8		1358.9732	1350.9338	1348.8441	1348.3456	1348.1134	1347.9481	1347.8249	1347.7304	1347.6561	1347.5961
1.6		1248.5791	1240.0340	1237.6448	1237.0720	1236.8409	1236.6815	1236.5636	1236.4737	1236.4032	1236.3465
1.4		1135.0548	1125.9668	1123.2160	1122.5322	1122.2966	1122.1440	1122.0322	1121.9475	1121.8814	1121.8283
1.3		1076.9251	1067.5525	1064.5938	1063.8352	1063.5935	1063.4446	1063.3361	1063.2543	1063.1906	1063.1395
1.2		1017.7520	1008.0870	1004.8998	1004.0499	1003.7976	1003.6525	1003.5476	1003.4690	1003.4078	1003.3587
1.1		957.4113	947.4476	944.0098	943.0482	942.7787	942.6373	942.5364	942.4611	942.4026	942.3558
1.0		895.7512	885.4848	881.7727	880.6746						
0.9		832.5824	822.0131	818.0014							
0.8		767.6626	756.7966	752.4594							
0.7		DNC	689.5274	684.8394							
0.6			619.7893	614.7280							
0.5			546.9927	541.5433							
0.4			470.2493	464.4148							
0.3			388.0934	381.9173							
0.28			370.8073	364.5751							
0.2			DNC	291.3593							
0.1											

Effective interaction, $F = 7$.

E_{HF}	10	12	14	16	18	20	22	24
5.0	2924.3622	2907.1863	2905.4553	2905.8337	2906.1635	2906.4007	2906.5799	2906.7204
4.0	2471.4336	2452.8813	2450.3427	2450.5860	2450.9092	2451.1403	2451.3147	2451.4510
3.0	1996.7261	1976.6505	1972.9569	1972.9141	1973.2181	1973.4426	1973.6111	1973.7423
2.5	1747.9701	1727.0999	1722.6373	1722.3377	1722.6136	1722.8348	1722.9998	1723.1279
2.0		1466.9527	1461.5444	1460.8497	1461.0596	1461.2766	1461.4377	1461.5623
1.8		1359.1135	1353.2669	1352.3571	1352.5194	1352.7332	1352.8927	1353.0157
1.6		1248.6043	1242.2801	1241.1112	1241.2051	1241.4132	1241.5710	1241.6925
1.4		1134.9422	1128.0986	1126.6162	1126.6113	1126.8080	1126.9643	1127.0841
1.3		1076.7333	1069.6140	1067.9505	1067.8806	1068.0677	1068.2233	1068.3421
1.2		1017.4729	1010.0673	1008.2039	1008.0555	1008.2289	1008.3836	1008.5015
1.1		957.0355	949.3338	947.2497	947.0065	947.1602	947.3136	947.4307
1.0		895.2681	887.2617	884.9337	884.5757	DNC	884.8524	884.9687
0.9		831.9797	823.6623	821.0648	DNC		DNC	DNC
0.8			758.2958	755.4007				
0.7			690.8492	687.6266				
0.6			620.8996	617.3182				
0.5			547.8485	543.8786				
0.4			470.7949	466.4146				
0.3			388.2552	383.4666				
0.28				366.0162				
0.2				292.2890				
0.1				186.1034				
0.05								

Effective interaction, $F = 7$, HF basis.

E_{CCSD}	10	12	14	16	18	20	22	24
5.0	2921.7134	2903.2085	2900.6233	2900.4615	2900.3992	2900.3408	2900.2909	2900.2498
4.0	2468.8654	2449.0180	2445.5934	2445.2945	2445.2305	2445.1728	2445.1247	2445.0858
3.0	1994.2639	1972.9619	1968.3464	1967.7491	1967.6723	1967.6164	1967.5713	1967.5359
2.5	DNC	1723.5348	1718.1368	1717.2677	1717.1664	1717.1123	1717.0695	1717.0365
2.0		1463.5481	1457.2036	1455.9139	1455.7494	1455.6970	1455.6573	1455.6276
1.8		1355.7871	1349.0109	1347.4925	1347.2802	1347.2276	1347.1895	1347.1615
1.6		1245.3662	1238.1257	1236.3328	1236.0502	1235.9951	1235.9591	1235.9329
1.4		1131.8049	1124.0672	1121.9446	1121.5585	1121.4957	1121.4621	1121.4382
1.3		1073.6520	1065.6540	1063.3425	1062.8874	1062.8171	1062.7849	1062.7622
1.2		1014.4517	1006.1866	1003.6683	1003.1292	1003.0474	1003.0164	1002.9951
1.1		954.0794	945.5414	942.7970	942.1559	942.0564	942.0267	942.0069
1.0		892.3825	883.5681	880.5768	879.8119		879.6566	879.6386
0.9		DNC	820.0796	816.8198				
0.8			754.8382	751.2877				
0.7			687.5338	683.6708				
0.6			617.7467	613.5518				
0.5			544.8834	540.3430				
0.4			468.0504	463.1638				
0.3			DNC	380.5729				
0.28				363.2048				
0.2				289.8539				
0.1				DNC				
0.05								

Chapter 8

Conclusions

In this master's thesis we aimed towards a more flexible and faster coupled-cluster code. The main goal was to study the use of alternative methods for matrix-multiplication, especially accelerated by the use of graphics cards, GPUs. Initially a previous C++ code ([1, 3]) was considered to be extended, but the decision fell on a complete redesign and rewrite of the program. This was partially to fit a different coding style, but also to redesign parts that were bottlenecks. In fact, most of the speedup acquired here was due to a simplification of the problem, rewriting all large matrices on a block-diagonal form, not by the accelerated multiplications.

Over now three master's projects, the boundary has been pushed from 20 particles in 110 basis functions [2], first increasing the basis size to up to 420 functions [1], and now up to 56 particles in 930 basis functions. The current limit is now the memory requirements of up to 100 gigabytes and not the long run times anymore. A consequence of the current state is that the next area of focus should no longer be aimed at a pure speed up. Further work on the program could include triples, leading to either perturbative triples correction, CCSD(T), or full inclusion of $\hat{T}_3$, CCSDT. In order to move to larger systems it is also needed to either distribute matrix elements across different nodes, or calculate them on the fly, in order to reduce memory usage.

When it comes to results, we have managed to expand the range of frequencies, ω, toward less bound systems, using a step-by-step approach where the initial guess is now based on a previous calculation. Although this technique opens up a new area of convergence, it is not sufficient for larger systems. We are for instance not able to break the barrier of $\omega = 1.0$ for 56 particles. From the more physical point of view, it is of great interest to study more difficult systems such as a double well quantum dot, along the lines of Wang's studies [3]. In our opinion it should not be hard to extend the developed library to include such a system. A reduction in the wall time used for simulations of a double dot would then most likely be seen as well, if the symmetries of the new Hamiltonian is successfully described in the 'Basis' class.

Another interesting topic to investigate is simulating the time-dependent Schrödinger equation. A time-dependent coupled-cluster framework is already tried to some extent. See for example [26]. Including a variable magnetic field in the calculations, if possible, one could simulate spin-flipping and other phenomena. Such simulations are not done previously, and would probably require a rethinking of the approach.

We are well satisfied with the work done on implementing the coupled cluster equations. To the best of our knowledge, coupled-cluster calculations using GPU programming have not been studied before. Although only a speedup of 2-4 times was seen compared to a quad core for the matrix-multiplications itself, the impact on the full program was not dramatic for such a small system. We predict that GPUs could be more effective if we could circumvent the memory transfer between the host and the GPU. Perhaps could GPUs play a more important role if elements were calculated on the fly, and these calculations were parallelized. Seen from a cost-effective perspective, the use of multiple GPUs is worth a try, as multiple GPUs can be attached to the same node.

Developing this library was time consuming as the project rapidly grew in size. The current code base consists of more than 6000 lines of code along with 2000 lines of comments. We therefore hope that sometime someone will continue along the path we have partly paved here, by creating creating a library that breaks the barriers we met.

- Efficiency gain for HFsys
- Efficiency due to channel/conf
- Thin vs. square matrices
- Strassen not optimal (no or little gain)
- Scaling is now better than previous program.
- Mappings are too slow
- Effective 24 shells conv $< 5 \cdot 10^{-5}$.
- Correlations dominate for many particles and low frequencies.
- Beyond HF important for few particles and low frequencies.
- Compares well to DMC.
- FCI excitations vs. shells?

Bibliography

[1] M. H. Jørgensen, "Many-body approaches to quantum dots," Master's thesis, University of Oslo, 2011. [Online]. Available: http://urn.nb.no/URN:NBN:no-28880

[2] M. P. Lohne, "Coupled-cluster studies of quantum dots," Master's thesis, University of Oslo, 2010. [Online]. Available: http://urn.nb.no/URN:NBN:no-26102

[3] Y. M. Wang, "Coupled-cluster studies of double quantum dots," Master's thesis, University of Oslo, 2011. [Online]. Available: http://urn.nb.no/URN:NBN:no-28527

[4] D. Griffiths, *Introduction to Quantum Mechanics*, 2nd ed. Pearson, 2005.

[5] P. A. M. Dirac, "A new notation for quantum mechanics," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 35, pp. 416–418, 1939.

[6] J. C. Slater, "The theory of complex spectra," *Phys. Rev.*, vol. 34, pp. 1293–1322, Nov 1929. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRev.34.1293

[7] W. Pauli, "Über den zusammenhang des abschlusses der elektronengruppen im atom mit der komplexstruktur der spektren," *Zeitschrift für Physik A Hadrons and Nuclei*, vol. 31, pp. 765–783, 1925, 10.1007/BF02980631. [Online]. Available: http://dx.doi.org/10.1007/BF02980631

[8] G.-C. Wick, "The evaluation of the collision matrix," *Phys. Rev.*, vol. 80, pp. 268–272, Oct 1950. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRev.80.268

[9] I. Shavitt and R. J. Bartlett, *Many-body methods in chemistry and physics: MBPT and coupled-cluster theory*, ser. Cambridge molecular science series. Cambridge University Press, 2009. [Online]. Available: http://books.google.no/books?id=gU1eHAAACAAJ

[10] F. Coester, "Bound states of a many-particle system," *Nuclear Physics*, vol. 7, no. 0, pp. 421 – 424, 1958. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0029558258902803

[11] F. Coester and H. Kümmel, "Short-range correlations in nuclear wave functions," *Nuclear Physics*, vol. 17, no. 0, pp. 477 – 485, 1960. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0029558260901401

[12] J. Čížek, "On the correlation problem in atomic and molecular systems. calculation of wavefunction components in ursell-type expansion using quantum-field theoretical methods," *The Journal of Chemical Physics*, vol. 45, no. 11, pp. 4256–4266, 1966. [Online]. Available: http://link.aip.org/link/?JCP/45/4256/1

[13] (2011, June) Radeon HD 6970 specifications. Advanced Micro Devices, Inc. [Online]. Available: ”http://www.amd.com/us/products/desktop/graphics/amd-radeon-hd-6000/hd-6970/”

[14] Microsoft, *Programming Guide for Direct3D 11*, Microsoft Std. [Online]. Available: http://msdn.microsoft.com/en-us/library/windows/desktop/ff476345(v=vs.85).aspx

[15] NVIDIA, *NVIDIA CUDA C Programming Guide*, NVIDIA Std. Version 4.2, 4 2012. [Online]. Available: http://developer.nvidia.com/nvidia-gpu-computing-documentation

[16] Khronos OpenCL Working Group, *OpenCL Specification*, Khronos Group Std. Version 1.1, Rev. 36.

[17] V. Strassen, “Gaussian elimination is not optimal,” *Numerische Mathematik*, vol. 13, pp. 354–356, 1969, 10.1007/BF02165411. [Online]. Available: http://dx.doi.org/10.1007/BF02165411

[18] [Online]. Available: http://www.netlib.org/blas/

[19] K. Goto and R. Van De Geijn, “High-performance implementation of the level-3 blas,” *ACM Trans. Math. Softw.*, vol. 35, no. 1, pp. 4:1–4:14, Jul. 2008. [Online]. Available: http://doi.acm.org/10.1145/1377603.1377607

[20] N. J. Higham, “Exploiting fast matrix multiplication within the level 3 blas,” *ACM Trans. Math. Softw.*, vol. 16, no. 4, pp. 352–368, Dec. 1990. [Online]. Available: http://doi.acm.org/10.1145/98267.98290

[21] M. Taut, “Two electrons in a homogeneous magnetic field: particular analytical solutions,” *Journal of Physics A: Mathematical and General*, vol. vol. 27, no. 3, pp. pp. 1045–1055, 1994.

[22] M. Pedersen Lohne, G. Hagen, M. Hjorth-Jensen, S. Kvaal, and F. Pederiva, “*Ab initio* computation of the energies of circular quantum dots,” *Phys. Rev. B*, vol. 84, p. 115302, Sep 2011. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevB.84.115302

[23] S. Kvaal, “Open source fci code for quantum dots and effective interactions,” 2008, arXiv:0810.2644v1.

[24] C. Sanderson, “Armadillo: An open source c++ linear algebra library for fast prototyping and computationally intensive experiments,” NICTA, Tech. Rep., 2010.

[25] C. Hirth. toffyrn::qvmc. [Online]. Available: https://orion.toffyrn.net/trac/physics/wiki/qvmc

[26] S. Kvaal, “Ab initio quantum dynamics using coupled-cluster,” *The Journal of Chemical Physics*, vol. 136, no. 19, p. 194109, 2012. [Online]. Available: http://link.aip.org/link/?JCP/136/194109/1