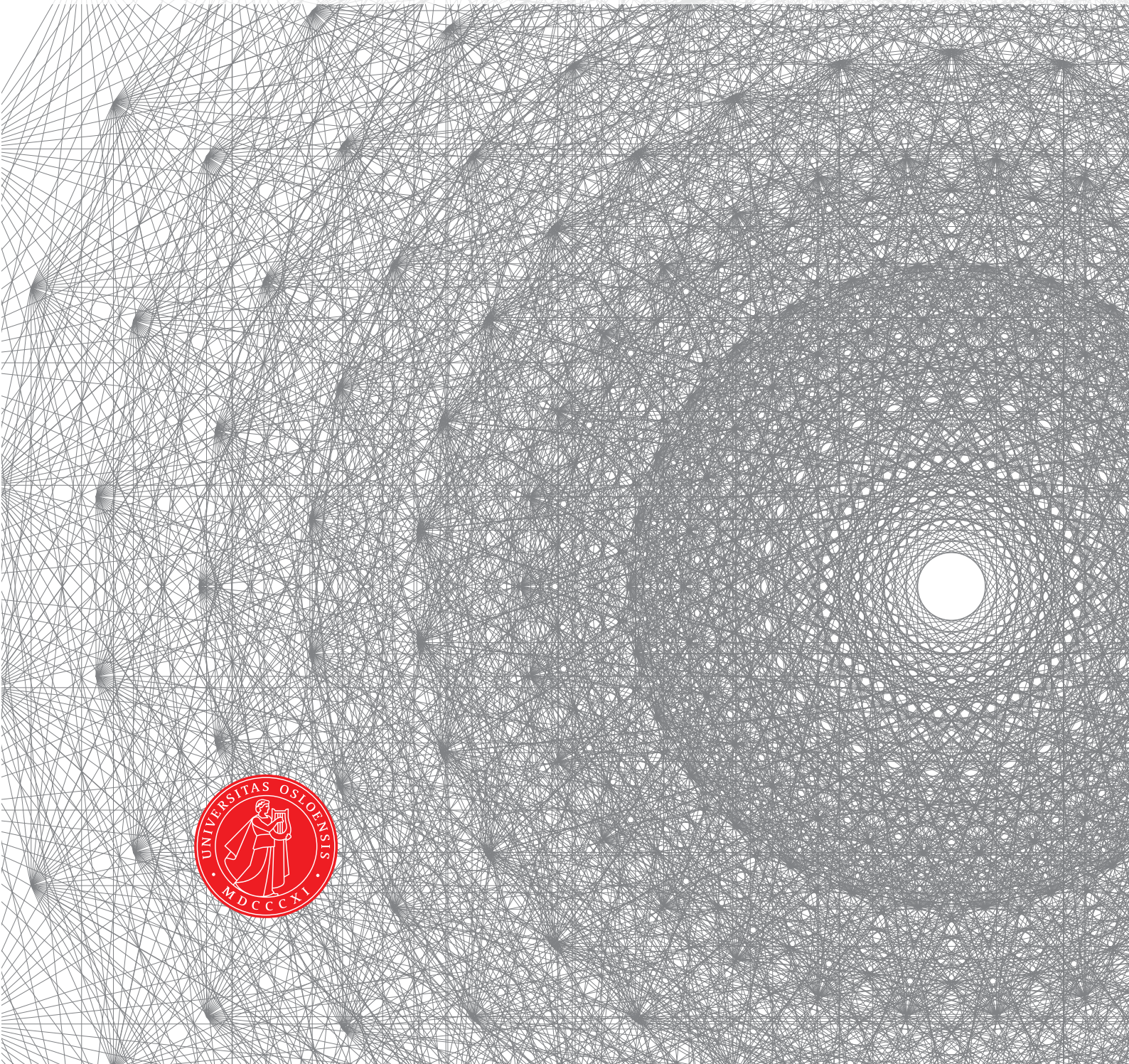
UiO : Department of Mathematics
University of Oslo

Time Evolution of Quantum Mechanical Many-Body Systems

Håkon Emil Kristiansen

Master's Thesis, Autumn 2017



This master's thesis is submitted under the master's programme *Computational Science and Engineering*, with programme option *Computational Science*, at the Department of Mathematics, University of Oslo. The scope of the thesis is 60 credits.

The front page depicts a section of the root system of the exceptional Lie group E_8, projected into the plane. Lie groups were invented by the Norwegian mathematician Sophus Lie (1842–1899) to express symmetries in differential equations and today they play a central role in various parts of mathematics.

Abstract

The solution of the time-dependent Schrödinger equation plays a central role for our understanding of quantum mechanical systems, spanning from studies and implementations of quantum circuits to nuclears reactions in stars and the synthesis of the elements. It provides invaluable information about the temporal evolution of quantum mechanical systems and their behavior when interacting with matter and/or external probes. The time-dependent Schrödinger equation (TDSE) plays thus a ubiquitous role when studying interacting quantum mechanical many-particle systems. Due to the high level of complexity involved in time-dependent studies, TDSE applications and studies have often been limited to few-body systems only.

In order to tackle the increasing complexities of different many-particle systems, we have in this thesis developed an efficient many-body approach based on the orbital-adaptive time-dependent coupled-cluster [1] (OATDCC) method. The coupled-cluster method which we base our theoretical derivations on, has been applied to a wide range of time-independent many-particle systems, from atomic and molecular physics to strongly interacting matter as seen in compact objects like neutron stars. Coupled cluster theory allows for systematic approximations to the full many-body problem and circumvents thus many of the dimensionality problems encountered in for example large-scale diagonalization problems. We refer our implementations and studies to as just the time-dependent coupled-cluster (TDCC) method. The TDCC method approximates solutions to the quantum N-body problem and for $N = 2$ TDCC is equivalent to the time-dependent configuration-interaction method (TDCI). The latter provides, within a given effective space, exact results that can be used to benchmark results from approximative methods like the TDCC developed here.

In this work, in order to compare our TDCC method with exact results, we have also developed and implemented a TDCI solver. This is shown to be correct by comparing with previous studies and is used as a foundation for verifying the implementation of the TDCC method. We demonstrate that the methods are equal to numerical precision for one and two-dimensional

quantum dot systems with two particles. The TDCC solver is then used to compute the time evolution of more complicated many-particle systems. As a proof of principle, we study a six-electron circular quantum dot in two dimensions. The properties of quantum dots play a central role in constructing quantum circuits to be used in future quantum computers.

Additionally, in order to demonstrate the feasibility of our codes and formalism, we present ground state energies for circular quantum dot systems in three dimensions using the Hartree-Fock, Configuration Interaction and Coupled Cluster methods. The results are shown to be in agreement with results obtained with other many-body methods.

Acknowledgements

I would like to thank my supervisor Morten Hjorth-Jensen for providing me with an interesting topic and giving me the freedom to explore those parts I have found most interesting. Furthermore, his enthusiasm for everything remotely related to scientific computing and profound interest in everyone's well-being is inspiring.

I am grateful to Simen Kvaal for taking interest in my work, his outspoken belief in my abilities and taking the time to discuss details of time-dependent Coupled Cluster theory. This has been a huge motivation.

A big thanks to the Computational Physics group and all the great people there for hosting me for the last two and a half years.

I have to mention a series of friends. Morten Ledum, with whom I have shared endless hours with, teaching computational physics, discussing many-body methods, our feeble attempts to establish a tradition of daily proofs, reducing problems to semi-trivial and much more. Thanks for teaching me to raise the bar and at some point you have to stop asking me questions I cannot answer.

The brilliant man that is Håvard Tveit Ihle, for all the chess games, trying to teach me Quantum Mechanics the real way and for just being a good friend. Stian Goplen, for his hospitality and sincere friendship, Oslo would not have been the same without you. My good friend through the last fifteen or so years, Knut Bjørnebye, for beer, burgers, heavy metal. Sindre, for staying in touch all these years. To Erling, for all the good times.

To my brothers, Rune and Tom, for always believing in me. Last but not least, to my mum and dad, Inger and Alf, for their unconditional love and endless support. I could not ask for more, you are the best.

Håkon Emil Kristiansen

Oslo, November 2017

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Quantum mechanics is a theory that describes the properties of microscopic systems. It postulates that given the wavefunction, $\Psi(\mathbf{r}, t)$, we can in principle compute all that there is to know about the system. Given suitable initial conditions, $\Psi(\mathbf{r}, t)$ can be determined for all future times by solving the time-dependent Schrödinger equation (TDSE),

$$i\hbar\frac{\partial}{\partial t}\Psi(\mathbf{r}, t) = \hat{H}\Psi(\mathbf{r}, t). \tag{1.1}$$

The initial conditions are typically taken as a linear combination of what is called stationary states, which in turn can be found by solving what is known as the time-independent Schrödinger (TISE) equation

$$\hat{H}\Psi(\mathbf{r}) = E\Psi(\mathbf{r}). \tag{1.2}$$

Analytical solutions to the TISE is possible only for the simplest systems and general solutions to the TDSE is even rarer. Thus, one must resort to numerical methods. Of particular interest is the so-called many-body problem where one considers systems with large numbers of interacting particles, where large can be anywhere from two to infinity.

1.1 Many-Body Methods

Several approaches for solving the many-body TISE approximatively have been devised such as Hartree-Fock (HF) theory [2], Density Functional Theory (DFT) [3, 4], Configuration Interaction (CI) theory and Coupled Cluster (CC) theory [5–7]. While efficient, Hartree-Fock and DFT are insufficient if one wants a high degree of accuracy.

Configuration Interaction theory and the various Coupled Cluster methods are hierarchical in the sense that one can systematically construct increasingly accurate approximations. If the CI method is not truncated we have what is known as Full Configuration Interaction (FCI). Full Configuration Interaction can be seen as exact (within some finite space), however it suffers from exponential scaling. Truncated CI methods which would achieve polynomial scaling are problematic since they are not size-consistent and extensive. Truncated CC methods on the other hand are size-consistent and extensive and achieve polynomial scaling [7]. Due to this fact CC theory is considered as the gold standard of many-body techniques if high accuracy is desired.

Similarly, there exist different approximations to the solution of the TDSE with the Multiconfiguration Time-Dependent Hartree-Fock method [8] (MCTDHF) being considered the most accurate. Multiconfiguration Time-Dependent Hartree-Fock, which is a combination of CI and HF generalized to the time domain, suffers from exponential scaling. In a recent article [1], Simen Kvaal proposed a method based on CC theory, the so-called Orbital-Adaptive Time-Dependent Coupled-Cluster method (OATDCC), which is a hierarchical approximation to the MCTDHF method. The OATDCC inherits size-consistency and extensivity from the CC method and achieves polynomial scaling.

1.2 Goals

The main goal of this thesis is to implement a simplified version of the OATDCC method, which we refer to as the Time Dependent Coupled Cluster (TDCC) method and apply it to one-, two- and three-dimensional quantum dot systems. Before we can compute the time evolution of a system an initial condition must be prepared. The initial condition is taken as the ground state of the system under consideration and is computed via the Coupled Cluster method in a Hartree-Fock basis. We will write Hartree-Fock and CC codes from scratch. This is advantageous since TDCC is an extension of the CC method for computing the ground state.

It turns out that TDCC is equal to the Time-Dependent Configuration Interaction [8] (TDCI) method in the special case of $N = 2$ particles. Thus, a possible way of validating the TDCC implementation is by comparing it with TDCI calculations. We demonstrate this in the present document. As such, we will also write a TDCI program. A study by Zanghellini et al. [9] studies the time evolution of a one-dimensional quantum dot system, which we will use to verify the TDCI program. The implementation may then be

split into the following steps:

a) Develop Hartree-Fock, Configuration Interaction and Coupled Cluster programs for ground state computations.

b) Expand the CI and CC program to the time domain.

c) Establish the validity of the TDCI program by comparing with Ref. [9]

d) Compare results obtained from the TDCC method with those resulting from TDCI calculations. If the results are equal to some numerical precision for $N = 2$ we take this as a strong indication that the TDCC method has been correctly implemented.

e) Having validated the implementation of the TDCC method we apply the program to more complex two- and three-dimensional quantum dot systems.

It should be noted that a working implementation of the TDCC method in the future can be expanded to the full OATDCC method.

1.3 Our contributions

There already exist many professionaly written computer codes for many-body calculations. Competing with these highly optimized implementations is in no way realistic. In order to gain valuable insight, Hartree-Fock, Coupled Cluster and Configuration Interaction codes have been written from scratch. Furthermore, the Coupled Cluster and Configuration Interaction programs have been expanded to the time domain, i.e TDCC and TDCI.

All programs have been written in both Python and C++ and the source code is available at the Github site[1][2]. C++ is more suited for high-performance computing since it can be easily combined with Message Passing Interface (MPI) and/or OpenMP for parallellization. See for example the textbook by Karniadakis and Kirby [10].

However, throughout the thesis we will focus on the Python implementation since it is considerably easier to present in text, with additional pedagogical advantages. The hope is that future Master of Science students or other interested scientists, will find the Python implementation easier to read than standard implementations in for example C++. Furthermore, we do not consider large many-particle systems, thus high-performance computing topics

[1] https://github.com/haakoek/PythonVersionMaster
[2] https://github.com/haakoek/CppVersionMaster

are not the main concern of this work. The main objective is to demonstrate a working implementation of the Time-Dependent Coupled Cluster method.

The choice of Python is due to its flexibility and extensive support for graphics, smoothing the development process. Computationally demanding parts are handled by the use of the *NumPy*[3] library which supports operations on N-dimensional arrays. In addition, the symbolic mathematics library *SymPy*[4] contains a second quantization toolbox[5] which is of great help when working with many-body methods.

[3] numpy.org/
[4] sympy.org/
[5] http://docs.sympy.org/latest/modules/physics/secondquant.html

Chapter 2

Quantum Many-Body Theory

In this chapter we will briefly review key concepts of quantum many-body theory. The material presented is based on the lecture notes [11] written by Simen Kvaal for the course Fys-Kjm4480/9480 given during autumn 2015 at the University of Oslo.

2.1 The Many-Body Problem

The goal of quantum many-body theory is to model systems of N interacting particles. It is a postulate of quantum mechanics that all information about the system is contained within a complex-valued wavefunction, $\Psi(x_1, \cdots, x_N)$, also commonly referred to as the systems *state function*. The wavefunction has a probabilistic interpretation in the sense that $|\Psi(x_1, ..., x_N)|^2$ is the probability density for locating all particles at the point $(x_1, \cdots, x_N) \in X^N$. Since the total probability must be 1 (the particles have to be somewhere) one demands that,

$$\int_{X^N} |\Psi(x_1, ..., x_N)|^2 dx_1 \cdots dx_N = 1. \tag{2.1}$$

A system is defined by a linear Hermitian operator, $\hat{H}$, commonly referred to as the system Hamiltonian.

Another fundamental postulate in quantum mechanics is that all physical observables, i.e properties of a system that we can measure, are represented by Hermitian operators $\hat{Q}$. The expectation value of an operator,

$$\langle \Psi | \hat{Q} | \Psi \rangle \equiv \int_{X^N} \Psi(x_1, \cdots, x_N)^* \hat{Q} \Psi(x_1, \cdots, x_N) dx_1 \cdots x_N, \tag{2.2}$$

represents the average of repeated measurements on an ensemble of identically prepared systems [12]. Note that if Ψ is an eigenfunction of $\hat{Q}$ with

eigenvalue q, i.e $\hat{Q}\Psi = q\Psi$, then the expectation value is certain to give q,

$$\langle\Psi|\hat{Q}|\Psi\rangle = \int_{X^N} q|\Psi|^2 dx_1 \cdots x_N = q. \tag{2.3}$$

Therefore, eigenfunctions of $\hat{Q}$ are often called determinate states since a measurement always gives the value q.

The Hamiltonian is the operator representing the total energy of a system and the eigenvalue equation for the Hamiltonian is known as the time independent Schrödinger equation (TISE),

$$\hat{H}\Psi_k(x_1, \cdots, x_n) = E_k\Psi_k(x_1, \cdots, x_N). \tag{2.4}$$

The eigenfunctions of $\hat{H}$ are commonly referred to as *energy eigenstates*. Of particular importance is the eigenstate with the lowest eigenvalue which is a systems *ground state*. The eigenfunctions of $\hat{H}$ (or any other Hermitian operator) form a basis such that any other state, $\Psi(x_1, \cdots, x_N)$, can be expanded in the eigenstates

$$\Psi(x_1, \cdots, x_N) = \sum_{k=1}^{\infty} c_k\Psi_k(x_1, \cdots, x_N). \tag{2.5}$$

The time evolution of an initial state, $\Psi(x_1, \cdots, x_N, t_0)$, is given by the time dependent Schrödinger equation (TDSE)

$$i\hbar\frac{\partial}{\partial t}\Psi(x_1, \cdots, x_N, t) = \hat{H}\Psi(x_1, \cdots, x_N, t), \tag{2.6}$$

where $\hat{H}$ is the Hamiltonian of the system. If the Hamiltonian is independent of time, $\Psi(x_1, \cdots, x_N, t)$ is given by

$$\Psi(x_1, \cdots, x_N, t) = \sum_{k=1}^{\infty} c_k\Psi_k(x_1, \cdots, x_N)e^{-iE_kt/\hbar}, \tag{2.7}$$

with Ψ_k being the eigenfunctions of $\hat{H}$ with corresponding eigenvalues E_k [12]. The coefficients c_k are determined by writing $\Psi(x_1, \cdots, x_N, t_0)$ in terms of the eigenfunctions,

$$\Psi(x_1, \cdots, x_N, t_0) = \sum_{k=1}^{\infty} c_k\Psi_k(x_1, \cdots, x_N). \tag{2.8}$$

This implies that for time independent Hamiltonians, if we can solve the TISE in Eq. (2.4), we are in principal done since the time evolution of an arbitrary state is given in terms of the eigenstates.

However, we will in general work with a time dependent Hamiltonian, $\hat{H}(t)$, meaning that we have to solve the TDSE explicitly. The general approach is to first solve the TISE for the ground state, using this as an initial state, and then compute the time evolution dictated by the TDSE. Thus, in this thesis we will study methods and write computer programs that solves both the TISE and TDSE for a time dependent Hamiltonian.

2.2 Identical particles

When working with quantum mechanics all particles of the same type must be treated as identical and indistinguishable. Suppose that we have two particles A and B which are identical. Intuitively this means that the probability of finding particle A at position x_A and particle B at position x_B is the same as the probability of finding particle A at position x_B and particle B at position x_A since we cannot tell the particles apart.

Formally, our probability density must be permutation invariant in the sense that if $\sigma \in S_N$ is a permutation of N indices and $(x_1, \cdots, x_N)$ gives the position of N particles, we must have

$$|\Psi(x_1, \cdots, x_N)|^2 = \left|\Psi(x_{\sigma(1)}, \cdots, x_{\sigma(N)})\right|^2. \tag{2.9}$$

Since $\Psi(x_1, \cdots, x_N)$ can take complex values, this is equivalent to

$$\Psi(x_1, \cdots, x_N) = e^{i\alpha(\sigma)}\Psi(x_{\sigma(1)}, \cdots, x_{\sigma(N)}), \quad \alpha(\sigma) \in \mathbb{R}. \tag{2.10}$$

Define the permutation operator $\hat{P}_\sigma$ via

$$(\hat{P}_\sigma\Psi)(x_1, \cdots, x_N) \equiv \Psi(x_{\sigma(1)}, \cdots, x_{\sigma(N)}). \tag{2.11}$$

Thus, the condition (2.10) is equivalent to Ψ being an eigenfunction of $\hat{P}_\sigma$ for every $\sigma \in S_N$ with eigenvalues possibly depending on σ.

It can be shown that either $\hat{P}_\sigma\Psi = \Psi$ for every $\sigma \in S_N$, or $\hat{P}_\sigma = (-1)^{|\sigma|}\Psi$ for every $\sigma \in S_N$, where $|\sigma|$ is the number of transpositions in σ. In the first case Ψ is said to bo totally symmetric with respect to permutations, while in the second case we say that Ψ is totally anti-symmetric.

All particles carry an intrinsic property known as *spin* which can take integer or half-integer values. Particles with integer spin are called *bosons* and have symmetric wavefunctions. On the other hand, particles with half-integer spin are called *fermions* and must have anti-symmetric wavefunctions. In this work we will restrict our attention to fermions and we will therefore require that our wavefunctions are anti-symmetric.

2.3 Slater Determinants

From now on, we consider systems consisting of N identical fermions which must have a totally anti-symmetric wavefunction $\Psi(x_1, \cdots, x_N)$. Furthermore, we must require that the wavefunction satisfies

$$\int_{X^N} |\Psi(x_1, \cdots, x_N)| dx_1 \cdots dx_N = 1. \tag{2.12}$$

It would be highly desirable to have an orthonormal basis, $\{\Phi_I(x_1, \cdots, x_N)\}_{I=1}^M$, that we could expand our wavefunction in such that[1],

$$\Psi(x_1, \cdots, x_N) = \sum_{I=1}^{M} A_I \Phi_I(x_1, \cdots, x_N). \tag{2.13}$$

Suppose now that we are given an orthonormal basis, $\{\phi_p(x)\}_{p=1}^L$ which we will refer to as *single-particle functions*, such that the wavefunction for a single particle can be written

$$\psi(x) = \sum_p c_p \phi_p(x). \tag{2.14}$$

Now, if we take the tensor product of N single-particle functions,

$$\tilde{\Phi}_{p_1 \cdots p_N}(x_1, \cdots, x_N) = \phi_{p_1}(x_1) \cdots \phi_{p_N}(x_N), \tag{2.15}$$

we get a many-body wavefunction that satisfies (2.12),

$$\begin{aligned} &\int \left| \tilde{\Phi}_{p_1 \cdots p_N}(x_1, \cdots, x_N) \right|^2 dx_1 \cdots dx_N \\ &= \int \phi^*_{p_1}(x_1) \phi_{p_1}(x_1) dx_1 \int \cdots \int \phi_{p_N}(x_N) \phi_{p_N}(x_N) dx_N = 1. \end{aligned}$$

Furthermore, if $p = (p_1, \cdots, p_N)$ and $q = (q_1, \cdots, q_N)$ we have $\langle \tilde{\Phi}_p | \tilde{\Phi}_q \rangle = \delta_{pq}$, this means that the collection of these wavefunctions form an orthonormal basis for the wavefunctions of N particles satisfying (2.12). However, these functions are not anti-symmetric and can therefore not describe fermions!

This issue is solved by introducing the *Slater determinant*

$$\Phi_{p_1 \cdots p_N}(x_1, \cdots, x_N) \equiv [\phi_{p_1}, \cdots, \phi_{p_N}](x_1, \cdots, x_N)$$

[1] In theory the basis can be infinite, but in pratice we must restrict ourselves to a finite basis since computer memory is limited.

defined by,

$$\begin{aligned}[\phi_{p_1},\cdots,\phi_{p_N}](x_1,\cdots,x_N) &= \frac{1}{\sqrt{N!}}\begin{vmatrix}\phi_{p_1}(x_1) & \phi_{p_1}(x_2) & \cdots & \phi_{p_N}(x_N)\\ \phi_{p_2}(x_1) & \phi_{p_2}(x_2) & \cdots & \phi_{p_2}(x_N)\\ \vdots & \cdots & \cdots & \vdots\\ \phi_{p_N}(x_1) & \phi_{p_N}(x_2) & \cdots & \phi_{p_N}(x_N)\end{vmatrix}\\ &= \frac{1}{\sqrt{N!}}\sum_{\sigma\in S_N}(-1)^{|\sigma|}\prod_{i=1}^{N}\phi_{\sigma(p_i)}(x_i)\\ &= \frac{1}{\sqrt{N!}}\sum_{\sigma\in S_N}(-1)^{|\sigma|}\prod_{i=1}^{N}\phi_{p_i}(\sigma(x_i)),\end{aligned}$$

where $1/\sqrt{N!}$ is a normalization factor.

Observe that interchange of single-particle indices p corresponds to an interchange of rows which gives a sign change,

$$[\phi_{p_1},\cdots,\phi_{p_j},\cdots,\phi_{p_i},\cdots,\phi_{p_N}] = -[\phi_{p_1},\cdots,\phi_{p_j},\cdots,\phi_{p_i},\cdots,\phi_{p_N}]. \quad (2.16)$$

Interchange of the coordinates corresponds to an interchange of columns which also gives a sign change,

$$\begin{aligned}&[\phi_{p_1},\cdots,\phi_{p_N}](x_1,\cdots,x_i,\cdots,x_j,\cdots,x_N)\\ &= -[\phi_{p_1},\cdots,\phi_{p_N}](x_1,\cdots,x_j,\cdots,x_i,\cdots,x_N).\end{aligned} \quad (2.17)$$

The latter observation implies that the Slater determinants are anti-symmetric.

In order for the Slater determinants to form a basis we want only those that are linearly independent. If we choose an ordering of the single-particle indices,

$$p_1 < \cdots < p_N \quad (2.18)$$

then the collection of Slater determinants $\Phi_{p_1\cdots p_N}(x_1,\cdots,x_N)$ satisfying (2.18) form an orthonormal basis for the the anti-symmetric functions of N particles which satisfies (2.12).

Thus, given a Slater determinant basis, $\{\Phi_I\}_{I=1}^{M}$, for some subspace of all normalized anti-symmetric wavenfunctions, we can write such a function in terms of our Slater determinants

$$\Psi(x_1,\cdots,x_N) = \sum_I A_I\Phi(x_1,\cdots,x_N), \quad A_I\in\mathbb{C}. \quad (2.19)$$

2.4 Second Quantization

When developing methods for solving the TISE and TDSE for fermionic systems, working directly with the definition of Slater determinants will become cumbersome in the long run. The second quantization formalism provides a framework with compact and efficient notation for manipulation of Slater determinants.

2.4.1 Creation and annihilation operators

Let L_0^2 be the one-dimensional space spanned by $|-\rangle$, the *vacuum state*, which represents the space with zero particles. Furthermore $\langle -|-\rangle = 1$ and note that $|-\rangle \neq 0$. In the following we define the so-called creation and annihilation operators $c_q^\dagger$ and c_q.

Loosely speaking $c_q^\dagger$ creates a particle by inserting a row with index q and a column with coordinate x_{N+1} in a Slater determinant. In the same way the annihilation operator destroys a particle by removing a row and a column. More precisely we define $c_q^\dagger$ by its action on the vacuum state and an arbitrary Slater determinant.

Definition 1 (The creation operator)**.** *For every single-particle index q,*

$$c_q^\dagger |-\rangle = |q\rangle\,, \quad \langle x|q\rangle = \phi_q(x). \tag{2.20}$$

Let $|p_1 \cdots p_N\rangle$ be an arbitrary Slater determinant with $N > 0$, then we define

$$c_q^\dagger |p_1 \cdots p_N\rangle \equiv |q p_1 \cdots p_N\rangle\,. \tag{2.21}$$

Observe that if there is a j such that $q = p_j$, then $c_q^\dagger |p_1 \cdots p_N\rangle = 0$. Furthermore we have that

$$c_q^\dagger |p_1 \cdots p_N\rangle = (-1)^j |p_1 \cdots p_j q p_{j+1} \cdots p_N\rangle\,, \quad q \neq p_j. \tag{2.22}$$

Recall that we defined the basis determinants to be the determinants with ordered indices. If we choose j such that the augmented index set is ordered we have created a new basis determinant. Finally, after a particle is created the new determinant has to be renormalized.

Definition 2 (The annihilation operator)**.** *The annihilation operator is the Hermitian adjoint of the creation operator.*

Using this defintion one can show that the annihilation is characterized as follows:

(I) $c_q \left|-\right\rangle = 0$, since there are no particles to remove in the vacuum state.

(II) If $q = p_j$ for some j, $c_q \left|p_1 \cdots p_N\right\rangle \equiv (-1)^{j-1} \left|p_1 \cdots p_{j-1} p_{j+1} \cdots p_N\right\rangle$.

(III) If $q \neq p_j$ for all j, $c_q \left|p_1 \cdots p_N\right\rangle = 0$.

Now, the anticommutator of two operators $\hat{A}$ and $\hat{B}$ is defined as

$$\{\hat{A}, \hat{B}\} \equiv \hat{A}\hat{B} + \hat{B}\hat{A}. \tag{2.23}$$

We want to use the creation and annihilation operators to compute the matrix representation of other operators such as the Hamiltonian. This can be realized via the following anticommutator relations for the creation and annihilation operators:

$$\{c_p^\dagger, c_q^\dagger\} = 0 \tag{2.24}$$

$$\{c_p, c_q\} = 0 \tag{2.25}$$

$$\{c_p, c_q^\dagger\} = \delta_{pq}. \tag{2.26}$$

Equation (2.26) is commonly referred to as the fundamental anticommutator.

If we want to compute matrix elements using the anticommutator relations the key obeservation is that by repeated use of the fundamental anticommutator we move creation operators to the left and annihilation operators to the right until we are left with expressions on the form,

$$\delta_{p_1 q_1} ... \delta_{p_k q_k} \left\langle - \right| \left. - \right\rangle = \delta_{p_1 q_1} ... \delta_{p_k q_k}. \tag{2.27}$$

In principle we can do this for any operator on second quantized form for arbitary N. However, the number of anticommutators grows rapidly so it quickly becomes cumbersome and error prone. The process can be simplified through what is known as Wick's theorem which we discuss later.

2.4.2 Representation of operators in Second quantization.

One can show that any one-body operator $\hat{H}_0$ can be written in second quantization as,

$$\hat{H}_0 = \sum_{i=1}^{N} \hat{h}(i) = \sum_{pq} \langle p|\hat{h}|q\rangle \, c_p^\dagger c_q, \tag{2.28}$$

where

$$h_q^p = \langle p|\hat{h}|q\rangle = \int \phi_p(x)^* \hat{h} \phi_q(x) dx.$$

Furthermore, any two-body operator can be written as

$$\hat{W} = \sum_{i<j}^{N} \hat{w}(i,j) = \frac{1}{2}\sum_{pqrs} w_{rs}^{pq} c_p^\dagger c_q^\dagger c_s c_r. \tag{2.29}$$

Here,

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

Alternatively we can write

$$\hat{W} = \frac{1}{4}\sum_{pqrs} \langle pq|\hat{w}|rs\rangle_{\mathrm{AS}}\, c_p^\dagger c_q^\dagger c_s c_r, \tag{2.30}$$

which is the form that will be used most frequently throughout this thesis. Here

$$\langle pq|\hat{w}|rs\rangle_{AS} \equiv \langle pq|\hat{w}|rs\rangle - \langle pq|\hat{w}|sr\rangle \tag{2.31}$$

is the anti-symmetrized matrix element. It is common practice to suppress the AS subscript and we will make it clear when it is done.

Thus, we can write the full Hamiltonian, $\hat{H} = \hat{H}_0 + \hat{W}$ in its second quantized form as

$$\hat{H} = \sum_{pq} \langle p|\hat{h}|q\rangle\, c_p^\dagger c_q + \frac{1}{2}\sum_{pqrs} w_{rs}^{pq} c_p^\dagger c_q^\dagger c_s c_r \tag{2.32}$$

$$= \sum_{pq} \langle p|\hat{h}|q\rangle\, c_p^\dagger c_q + \frac{1}{4}\sum_{pqrs} \langle pq|\hat{w}|rs\rangle_{\mathrm{AS}}\, c_p^\dagger c_q^\dagger c_s c_r. \tag{2.33}$$

2.4.3 Normal order and Wick's Theorem.

It is possible to compute matrix elements by the use of the fundamental anticommutator relations. However, for arbitary N, the matrix elements of the Hamiltonian is on the form,

$$\langle\Phi|\hat{H}_0|\Phi\rangle = \sum_{qr} h_r^q \langle -|c_{p_N}\cdots c_{p_1} c_q^\dagger c_r c_{p_1}^\dagger \cdots c_{p_N}|-\rangle \tag{2.34}$$

$$\langle\Phi|\hat{W}|\Phi\rangle = \frac{1}{4}\sum_{q_1q_2r_1r_2} w_{r_1r_2\mathrm{AS}}^{q_1q_2} \langle -|c_{p_N}\cdots c_{p_1} c_{q_1}^\dagger c_{q_2}^\dagger c_{r_2} c_{r_1} c_{p_1}^\dagger \cdots c_{p_N}|-\rangle\,, \tag{2.35}$$

which quickly becomes unmanageable. In the following, we describe how expressions on the above form, known as vacuum expectation values, can be simplified by introducing the normal order of an operator. In the end we state Wick's theorem, which tells us how to compute vacuum expectation values in terms of normal ordered operators.

Definition 3 (Vacuum expectation value)**.** *Let $A_1 \cdots A_n$ denote a string of n operators with $A_i \in \{c_q^\dagger\} \cup \{c_q\}$. Then we say that $\langle -|A_1 \cdots A_N|-\rangle$ is a vacuum expectation value.*

Given h_q^p and $w_{r_1 r_2 \mathrm{AS}}^{q_1 q_2}$ the problem of computing the matrix elements (of the Hamiltonian) is reduced to computing vacuum expectation values. Before Wick's theorem is stated, we first define the normal-ordered product form of $A_1 \cdots A_n$ and the contraction of two arbitary creation and annihilation operators.

Definition 4 (Normal ordering)**.** *Let $\bar{A} = A_1 \cdots A_n$ be an arbitary operator string of creation and annihilation operators. Let $\sigma \in S_n$ be a permutation such that all creation operators in $\bar{A}$ are to the left of all the annihilation operators. Then the normal order of $A_1 \cdots A_n$ denoted by $\{A_1 ... A_n\}$ is defined as,*

$$\{A_1 ... A_n\} \equiv (-1)^{|\sigma|} A_{\sigma(1)} \cdots A_{\sigma(n)} = (-1)^{|\sigma|} [\textit{creation ops.}] \times [\textit{annihilation ops.}].$$

Note that:

- *The normal order is not unique since σ can be chosen in different ways.*
- *In general $A_1 \cdots A_n \neq \{A_1 \cdots A_n\}$.*

Definition 5 (Contraction)**.** *A contraction, $\overbracket{XY}$, between two arbitary creation and annihilation operators X and Y (relative to the vacuum state, $|-\rangle$) is the number*

$$\overbracket{XY} \equiv \langle -|XY|-\rangle \,. \tag{2.36}$$

The possible contractions are given by,

$$\begin{aligned}
\overbracket{c_p^\dagger c_q^\dagger} &= 0 \\
\overbracket{c_p c_q} &= 0 \\
\overbracket{c_p^\dagger c_q} &= 0 \\
\overbracket{c_p c_q^\dagger} &= \delta_{pq}.
\end{aligned}$$

Further we can define contractions inside a normal ordered string of operators.

Definition 6. *Let $A_1 \cdots A_n$ be an operator string, and let (x, y), $x < y$, be a pair of operators. Let σ be any permutation such that $\sigma(1) = x$ and $\sigma(2) = y$. Then,*

$$\{A_1 \cdots \overbrace{A_x \cdots A_y} \cdots A_n\} \equiv (-1)^{|\sigma|}\{\overbrace{A_x A_y} A_{\sigma(3)} \cdots A_{\sigma(n)}\}. \tag{2.37}$$

In general we can contract m pairs of operators as follows: Let (x_i, y_i), $x_i < y_i$ be a pair of operators and select σ such that

$$\begin{aligned} \sigma(1) = x_1, &\quad \sigma(2) = y_1 \\ \sigma(3) = x_2, &\quad \sigma(2) = y_2 \\ &\vdots \end{aligned}$$

Then

$$\{ \overbrace{A_1 \cdots A_n}^{m \text{ contractions}} \} = (-1)^{|\sigma|}\{\overbrace{A_{x_1} A_{y_1}} \cdots \overbrace{A_{x_m} A_{y_m}} A_{\sigma(2m+1)} \cdots A_{\sigma(n)}\}. \tag{2.38}$$

Finally we can state Wick's theorem.

Theorem 1 (Wick's theorem)**.** *Let $A_1 \cdots A_n$ be an operator string, then*

$$\begin{aligned} A_1 \cdots A_n = \{A_1 \cdots A_n\} + &\sum_{\text{all single contractions}} \{ \overbrace{A_1 \cdots A_n}^{\text{one contraction}} \} \\ + \cdots + &\sum_{\text{all } \lfloor \frac{n}{2} \rfloor \text{ contractions}} \{ \overbrace{A_1 \cdots A_n}^{\lfloor \frac{n}{2} \rfloor \text{ contractions}} \}. \end{aligned}$$

Notice that when n is even, the last term is fully contracted such that there are no creation and annihilation operators left. If n is odd there is one uncontracted operator left in each term of the last sum.

So, what is the great simplifaction that we promised with Wick's theorem? Notice that the vacuum expectation value of a normal ordered operator is always zero,

$$\langle - | \{A_1 \cdots A_n\} | - \rangle = 0, \tag{2.39}$$

since all annihilation operators are to the right. Furthermore if n is odd Wick's theorem implies that

$$\langle - | A_1 \cdots A_n | - \rangle = \text{sum over } \left\lfloor \frac{n}{2} \right\rfloor \text{ terms on the form } \delta_{()} \cdots \delta_{()} \langle - | c^{(\dagger)} | - \rangle = 0,$$

since $\langle -|c^{(\dagger)}|-\rangle = 0$. Finally for even n we have that

$$\langle -|A_1 \cdots A_n|-\rangle = \sum_{\lfloor \frac{n}{2} \rfloor} \{ \overbrace{A_1 \cdots A_n}^{\text{all contracted}} \}. \tag{2.40}$$

Since $\overbrace{c_p c_q^\dagger}$ is the only non-zero contraction, the number of contractions we have to consider in this final sum is further reduced.

Another useful result is that the sign of each term in the sum is $(-1)^k$, with k the number of crossings of contraction lines.

2.4.4 Particle-hole formalism.

The starting point of a many-body treatment is to solve the TISE,

$$\hat{H} |\Psi\rangle = E |\Psi\rangle . \tag{2.41}$$

In many cases a single Slater determinant can be a good approximation to the exact eigenfunction $|\Psi\rangle$. In Hartree-Fock theory, which we discuss in more detail in chapter 3, one seeks a single Slater determinant, $|\Phi\rangle$, that minimizes the energy expectation value

$$E[\Phi] = \langle \Phi | \hat{H} | \Phi \rangle . \tag{2.42}$$

We can motivate a single determinant approximation as follows: Suppose that $|\Phi\rangle = |p_1 \cdots p_N\rangle$ is an arbitary Slater determinant, then

$$\begin{aligned} \hat{H}_0 |\Phi\rangle &= \sum_{pq} \langle \phi_p | \hat{h} | \phi_q \rangle \, c_p^\dagger c_q \, |p_1 \cdots p_N\rangle \\ &= \sum_i h_{p_i}^{p_i} \, |p_1 \cdots p_N\rangle , \quad p_i \in |p_1 \cdots p_N\rangle \\ &= \left(\sum_i \epsilon_i \right) |\Phi\rangle , \quad \epsilon_i \equiv h_{p_i}^{p_i}. \end{aligned}$$

Thus, the eigenfunctions of $\hat{H}_0$ are single Slater determinants. Now, consider

$$\hat{H} |\Psi\rangle = \hat{H}_0 |\Psi\rangle + \hat{W} |\Psi\rangle . \tag{2.43}$$

The rationale is that if $\hat{W} |\Psi\rangle$ is in some sense "small" compared to $\hat{H}_0 |\Psi\rangle$ then a single Slater determinant can be a good approximation. In particular, as a starting point we can use the eigenfunctions of $\hat{H}_0$. However, this will in

general not be good enough and we want a way to make systematic corrections to the solution. Also, while Wick's theorem simplified the evaluation of vacuum expectation values, the number of terms we have to consider grows quickly with the number of particles N. One way to adress both of these issues is through the *particle-hole* formalism.

In practice we must always restrict ourself to a finite set of basis functions $\{\phi_p\}_{p=1}^{L}$ with L the number of basis functions. We start by dividing the single-particle space into two distinct sets,

$$\{\phi_p\}_{p=1}^{L} = \{ \underbrace{\phi_i}_{\text{Occupied}} \}_{i=1}^{N} \cup \{ \underbrace{\phi_a}_{Unoccupied} \}_{a=N+1}^{L}. \tag{2.44}$$

The N first single-particle states are commonly referred to as occupied orbitals while the rest are called unoccupied orbitals. The indices $i, j, k, \cdots$ are reserved for occupied orbitals while we use $a, b, c, \cdots$ for the unoccupied ones. Finally, $p, q, r, \cdots$ denote an arbitrary orbital. We define the reference determinant, $|\Phi_{\text{ref}}\rangle$, to be the Slater determinant filled with the occupied orbitals,

$$|\Phi\rangle_{\text{ref}} \equiv \prod_{i=1}^{N} c_i^\dagger \, |-\rangle \, . \tag{2.45}$$

The reference state can be chosen in various ways, exercising some foresight we consider two references in this thesis

(I) $|\Phi\rangle_{\text{ref}} = \prod_{i \in occ} c_i^\dagger \, |-\rangle$, with $\{\phi_p\}_{p=1}^{L}$ s.t $\hat{h}\phi_p = \epsilon_p \phi_p$

(II) $|\Phi\rangle_{\text{ref}} = |\Phi_{HF}\rangle$, where $|\Phi_{HF}\rangle$ is the Hartree-Fock state.

Hartree-Fock theory will be discussed later. In the following we drop the ref-subscript and write $|\Phi\rangle_{\text{ref}} = |\Phi\rangle$.

Excitation and de-excitation operators.

A string of creation and annihilation operators on the form

$$\hat{\tau}_{ijk\cdots}^{abc\cdots} \equiv \cdots c_c^\dagger c_k c_b^\dagger c_j c_a^\dagger c_i \tag{2.46}$$

is referred to as an excitation operator. Let $X = \{ijk\cdots, abc\cdots\}$ denote a general excitation index. Then we write $\hat{\tau}_X$ for a general excitation operator. The name excitation operator is due to the fact that if we act with τ_X on a reference determinant we substitute particles with indices less than N with particles with indices larger than N. If we act with an

excitation operator on a reference determinant we write

$$\hat{\tau}_X |\Phi\rangle = \hat{\tau}^{abc\cdots}_{ijk\cdots} |\Phi\rangle = |\Phi^{abc\cdots}_{ijk\cdots}\rangle = |\Phi_X\rangle , \tag{2.47}$$

with $|\Phi_X\rangle$ being referred to as an excited determinant. Similarly one can define a de-excitation operator

$$\hat{\lambda}_{\tilde{X}} = \hat{\lambda}^{ijk\cdots}_{abc\cdots} \equiv \cdots c^\dagger_k c_c c^\dagger_j c_b c^\dagger_i c_a, \tag{2.48}$$

with $\tilde{X} = \{abc\cdots, ijk\cdots\}$ denoting a general de-excitation index. We note that $\lambda_{\tilde{X}} |\Phi\rangle = 0$ since $c_a |\Phi\rangle = 0$. Hence, de-excitation operators must act on excited determinants to give a non-zero determinant.

A complete Slater determinant basis can now be constructed by acting on a reference state with excitation operators as follows:

$$\begin{aligned} |\Phi^a_i\rangle &= c^\dagger_a c_i |\Phi\rangle \quad , \text{ single excitation} \\ |\Phi^{ab}_{ij}\rangle &= c^\dagger_b c_j c^\dagger_a c_i |\Phi\rangle \quad , \text{ double excitation} \\ &\vdots \end{aligned}$$

Note that $|\Phi^{ab}_{ij}\rangle = -|\Phi^{ba}_{ij}\rangle = -|\Phi^{ab}_{ji}\rangle = |\Phi^{ba}_{ji}\rangle$. We can now make wavefunction ansatzes to the exact solution, $|\Psi\rangle$, in a systematic way by including linear combinations of different excitation levels,

$|\Psi\rangle \approx |\Phi\rangle$, single determinant approximation,

$|\Psi\rangle \approx A_0 |\Phi\rangle + \sum_{ia} A^a_i |\Phi^a_i\rangle$, all single excitations,

$|\Psi\rangle \approx A_0 |\Phi\rangle + \sum_{ia} A^a_i |\Phi^a_i\rangle + \frac{1}{4} \sum_{ijab} A^{ab}_{ij} |\Phi^{ab}_{ij}\rangle$, all single- and double excitations,

$\vdots$

Furthermore, we can define creation and annihilation operators, $b_i, b_a, b^\dagger_i, b^\dagger_a$, relative to the reference determinant in the following manner,

$$\begin{aligned} b_i &\equiv c^\dagger_i, \quad b_a \equiv c_a \\ b^\dagger_i &\equiv c_i, \quad b^\dagger_a \equiv c^\dagger_a. \end{aligned}$$

These are commonly referred to as quasiparticle creation and annihilation operators. Notice then that since the reference state by definition contains all occupied orbitals we have,

$$b_p |\Phi\rangle = 0 \quad \forall p,$$

which means that $|\Phi\rangle$ is the vacuum for quasiparticles. We say that $b_i^\dagger$ creates a hole since it removes one of the occupied orbitals, while we say that $b_a^\dagger$ creates a particle since it inserts one of the unoccupied orbitals.

The anticommutaor relations are preserved such that we have,

$$\{b_p, b_q^\dagger\} = \delta_{pq}, \quad \{b_p, b_q\} = 0. \tag{2.49}$$

Normal ordering, contractions and Wick's theorem are defined in terms of the anticommutator relations and the fact that $c_p |-\rangle = 0$. Thus Wick's theorem is also valid for quasiparticles.

We stress that contractions now are defined relative to the reference state, since this is a quasiparticle vacuum,

$$\overbrace{XY} = \langle\Phi|XY|\Phi\rangle, \quad X, Y \in \{b_i, b_a\} \cup \{b_i^\dagger, b_a^\dagger\}. \tag{2.50}$$

The only non-zero contractions are then,

$$\begin{aligned} \overbrace{b_i b_j^\dagger} &= \langle\Phi|b_i b_j^\dagger|\Phi\rangle = \delta_{ij} \\ \overbrace{b_a b_b^\dagger} &= \langle\Phi|b_a b_b^\dagger|\Phi\rangle = \delta_{ab}. \end{aligned}$$

This is a great simplification since we now only have to consider contractions relative to the reference vacuum when using Wick's theorem!

Slater-Condon rules

Let $i, j, k, l = 1, ..., N$, $a, b, c, d = N+1, ...$ and $p, q, r, s = 1, ...$ Suppose that $|\Phi\rangle = \prod_{i=1}^N c_i^\dagger |-\rangle$ is a reference determinant and that $\hat{H}_0 = \sum_{pq} h_q^p c_p^\dagger c_q$ and $\hat{W} = \frac{1}{4}\sum_{pqrs} u_{rs}^{pq} c_p^\dagger c_q^\dagger c_s c_r$ where $u_{rs}^{pq} = \langle\phi_p\phi_q|\hat{u}|\phi_r\phi_s\rangle_{AS}$. Then the Slater-Condon rules state,

$$\langle\Phi|\left(\hat{H}_0 + \hat{W}\right)|\Phi\rangle = \sum_i h_i^i + \frac{1}{2}\sum_{ij} u_{ij}^{ij} \tag{2.51}$$

$$\langle\Phi|\left(\hat{H}_0 + \hat{W}\right)|\Phi_k^c\rangle = h_k^c + \sum_i u_{ci}^{ki} \tag{2.52}$$

$$\langle\Phi|\left(\hat{H}_0 + \hat{W}\right)|\Phi_{kl}^{cd}\rangle = u_{cd}^{kl} \tag{2.53}$$

Proof: We prove the Slater-Condon rules using the particle-hole formalism. This means that we consider $|\Phi\rangle$ as the vacuum when apply-

ing Wick's theorem. First consider the expectation value,

$$\begin{aligned}\langle\Phi|\hat{H}_0|\Phi\rangle &= \sum_{pq} h_q^p \langle\Phi|c_p^\dagger c_q|\Phi\rangle \\ &= \sum_{ij} h_j^i \langle\Phi|b_i b_j^\dagger|\Phi\rangle + \sum_{ia} h_a^i \langle\Phi|b_i b_a|\Phi\rangle \\ &+ \sum_{ai} h_i^a \langle\Phi|b_a^\dagger b_i^\dagger|\Phi\rangle + \sum_{ab} h_b^a \langle\Phi|b_a^\dagger b_b|\Phi\rangle \\ &= \sum_{ij} h_j^i \delta_{ij} = \sum_i h_i^i. \end{aligned} \tag{2.54}$$

In the following we include only terms which have a non-zero contribution when we write out the sums. Now, consider

$$\langle\Phi|\hat{W}|\Phi\rangle = \frac{1}{4}\sum_{ijkl} u_{kl}^{ij} \langle\Phi|b_i b_j b_l^\dagger b_k^\dagger|\Phi\rangle .$$

By Wick's theorem we have that

$$\begin{aligned}\langle\Phi|b_i b_j b_l^\dagger b_k^\dagger|\Phi\rangle &= \sum_{\text{all contracted}} \langle\Phi|b_i b_j b_l^\dagger b_k^\dagger|\Phi\rangle \\ &= \sum_{klij} \langle\Phi|\overbrace{b_i b_j b_l^\dagger}\, b_k^\dagger|\Phi\rangle + \sum_{klij} \langle\Phi|\overbrace{b_i b_j}\, \overbrace{b_l^\dagger b_k^\dagger}|\Phi\rangle \\ &= \sum_{klij} \left(-\delta_{il}\delta_{jk} + \delta_{ik}\delta_{jl}\right). \end{aligned}$$

Then we get,

$$\begin{aligned}\frac{1}{4}\sum_{ijkl} u_{kl}^{ij} \langle\Phi|b_i b_j b_l^\dagger b_k^\dagger|\Phi\rangle &= \frac{1}{4}\sum_{klij} u_{kl}^{ij} \left(-\delta_{il}\delta_{jk} + \delta_{ik}\delta_{jl}\right) \\ &= \frac{1}{4}\sum_{ij} \left(-u_{ji}^{ij} + u_{ij}^{ij}\right) \\ &= \frac{1}{2}\sum_{ij} u_{ij}^{ij}. \end{aligned} \tag{2.55}$$

Combining Eqs. (2.54) and (2.55) proves the first rule (2.51). For the second rule we first consider,

$$\begin{aligned}\langle\Phi|\hat{H}_0|\Phi_k^c\rangle =& \sum_{pq} h_q^p \langle\Phi|c_p^\dagger c_q c_c^\dagger c_k|\Phi\rangle \\ =& \sum_{ia} h_a^i \langle\Phi|\overbrace{b_i b_a b_c^\dagger b_k^\dagger}|\Phi\rangle \\ =& \sum_{ia} h_a^i \delta_{ik}\delta_{ac} \\ =& h_c^k. \end{aligned} \tag{2.56}$$

Next,

$$\langle\Phi|\hat{W}|\Phi_k^c\rangle = \frac{1}{4}\sum_{pqrs} u_{rs}^{pq} \langle\Phi|c_p^\dagger c_q^\dagger c_s c_r c_c^\dagger c_k|\Phi\rangle .$$

We notice that c and k are fixed, thus s or r must be larger than N to get a non-zero contraction with $c_c^\dagger$. Similarly either p or q must be less than or equal to N to have a non-zero contraction with c_k. The remaining operators to be contracted will then be on the form $c_{p'}^\dagger c_{q'}$ which is non-zero if and only if both p', q' is less than N. Finally, we consider the case where $p = i, q = j, r = a$, and $s = l$. Writing out and applying Wick's theorem we get,

$$\begin{aligned} &\frac{1}{4}\sum_{ijal} u_{al}^{ij} \langle\Phi|c_i^\dagger c_j^\dagger c_l c_a c_c^\dagger c_k|\Phi\rangle \\ &= \frac{1}{4}\sum_{ijal} u_{al}^{ij} \left(\langle\Phi|\overbrace{b_i b_j b_l^\dagger b_a b_c^\dagger b_k^\dagger}|\Phi\rangle + \langle\Phi|\overbrace{b_i b_j b_l^\dagger b_a b_c^\dagger b_k^\dagger}|\Phi\rangle \right) \\ &= \frac{1}{4}\sum_{ijal} u_{al}^{ij} \left(\delta_{ik}\delta_{jl}\delta_{ca} - \delta_{il}\delta_{jk}\delta_{ac}\right) \\ &= \frac{1}{2}\sum_{i} u_{ci}^{ki} \end{aligned}$$

If we set $r = l, s = a$ we get the same result, thus in total we have

$$\langle\Phi|\hat{W}|\Phi_k^c\rangle = \sum_i u_{ci}^{ki}, \tag{2.57}$$

which combined with (2.56) proves the second rule (2.52).

2.5 The variational principle

A fundamental result in quantum mechanics is what is known as the variational principle. The variational principle establishes a link between the expectation value of the Hamiltonian and its eigenfunctions and gives an upper bound to the exact ground state energy.

First, recall that $\hat{H}$ is an Hermitian operator. In quantum mechanics it is customary to assume that $\hat{H}$ is diagonlizable and that we in principle can find a set, $\{|\psi_k\rangle\}$, of orthonormal eigenfunctions of $\hat{H}$ such that any wavefunction can be written in terms of the eigenfunctions,

$$|\Psi\rangle = \sum_k c_k |\psi_k\rangle . \tag{2.58}$$

We assume that $|\Psi\rangle$ is normalized such that $\langle\Psi|\Psi\rangle = 1$ and $\sum_k |c_k|^2 = 1$. Now, if we consider the expectation value of $\hat{H}$ we find

$$\begin{aligned}\langle\Psi|\hat{H}|\Psi\rangle &= \sum_{kl} c_l^* c_k \langle\psi_l|\hat{H}|\psi_k\rangle \\ &= \sum_k |c_k|^2 E_k\end{aligned}$$

where we used that the $|\psi_k\rangle$ are orthonormal eigenfunctions of $\hat{H}$ with eigenvalue E_k. Now, since the ground state energy E_0 is, by definition, the smallest eigenvalue we obtain

$$\langle\Psi|\hat{H}|\Psi\rangle \geq E_0 \sum_k |c_k|^2 = E_0. \tag{2.59}$$

Thus, we have found that the expectation value of $\hat{H}$ is an upper bound on the ground state.

In particular, this inspires the Hartree-Fock method which we examine further in the next chapter. In Hartree-Fock theory we seek the single Slater determinant that minimizes the expectation value of $\hat{H}$, thus providing an upper bound on the ground state energy.

Furtheremore, one can show the following theorem [2]:

Theorem 2. *Consider the expectation value functional defined by*

$$\mathcal{E}[\Psi] = \frac{\langle\Psi|\hat{H}|\Psi\rangle}{\langle\Psi|\Psi\rangle} \tag{2.60}$$

Let $|\Psi_*\rangle$ *be given. Then* $E_* = \mathcal{E}[\Psi_*]$ *is a stationary value of* $\mathcal{E}$ *if and only if*

$$\hat{H}\,|\Psi_*\rangle = E_*\,|\Psi_*\rangle\,. \tag{2.61}$$

This is an interesting result in itself since it means that in the context of quantum mechanics, finding critical points of the expectation value of $\hat{H}$ is equivalent with the time independent Schrödinger equation! Specifically, if we construct the matrix representation H of $\hat{H}$ relative to some finite orthonormal Slater determinant basis, $\{|\Phi\rangle_I\}_{i=1}^{L}$, given in terms of its matrix elements

$$H_{IJ} = \langle\Phi_I|\hat{H}|\Phi_J\rangle \tag{2.62}$$

then diagonalization of H would give an approximation to the first L eigenpairs of $\hat{H}$. We are also guaranteed that it provides an upper bound on the L first eigenvalues.

To see this we recall that the eigenvectors of a Hermitian matrix are orthogonal with real eigenvalues [13]. We order the eigenvectors by magnitude of the corresponding eigenvalue $(E_0, \psi_0), (E_1, \psi_1), (E_2, \psi_2), \cdots$ where $E_0 \leq E_1 \leq E_2 \cdots$. Here E_0 is the ground state energy, E_1 the energy of first excited state and so on. We assume that $\langle\psi_0|\Psi\rangle = 0$ where $|\Psi\rangle$ is given by (2.58). Then we see that

$$\langle\psi_0|\Psi\rangle = \sum_k c_k\,\langle\psi_0|\psi_k\rangle = c_k \Rightarrow c_k = 0. \tag{2.63}$$

Repeating the argument for the upper bound on the ground state we obtain

$$\langle\Psi|\hat{H}|\Psi\rangle = \sum_k E_k|c_k|^2 \geq E_1 \sum_k |c_k|^2 = E_1, \tag{2.64}$$

where we used that $c_k = 0$. Thus, if we have a wavefunction that is orthogonal to the ground state the expectation value of $\hat{H}$ is an upper bound on E_1. In terms of the matrix representation the eigenpair (E_1, ψ_1) is an upper bound on the next lowest eigenvalue of $\hat{H}$. This argument can be generalized to higher eigenvalues if we assume $\langle\psi_0|\Psi\rangle = 0, \langle\psi_1|\Psi\rangle = 0, \cdots$. Thus, if we consider larger and larger Slater determinant bases we are guaranteed to get a better estimate for the eigenvalues of $\hat{H}$.

Diagonalization of H is known as the Configuration Interaction method which we discuss in more detail in chapter 4.

2.6 Density Matrices

It is hard to visualize a multivariate complex-function such as the wavefunction, $\Psi(x_1, \cdots, x_N)$ and it is therefore useful to introduce the so-called single-particle density function also referred to as the one-body density, $\rho(x, t)$,

defined by integrating over all coordinates except one

$$\rho(x_1, t) \equiv \int |\Psi(x_1, \cdots, x_N)|^2 \, dx_2 \cdots dx_N. \tag{2.65}$$

The one-body density is interpreted physically as the probability of finding a particle in dx_1 at x_1, independent of where the other electrons are.

If the wavefunction is expanded in a Slater determinant basis built from the single-particle functions, $\{\phi_p\}$, the one-body density can be written as [14]

$$\rho(x_1) = \sum_{pq} \rho_q^p \phi_p^*(x_1) \phi_q(x_1) \tag{2.66}$$

where we have defined,

$$\rho_q^p \equiv \langle \Psi | c_p^\dagger c_q | \Psi \rangle \tag{2.67}$$

Similarly, we can define the two-body density

$$\rho(x_1, x_2, t) \equiv \int |\Psi(x_1, \cdots, x_N)|^2 \, dx_3 \cdots dx_N, \tag{2.68}$$

which is interpreted as the probability of finding some particle in dx_1 at x_1 and some other particle in dx_2 at x_2. The two-body density can be written in terms of the single-particle funtions as

$$\rho(x_1, x_2) = \sum_{pqrs} \rho_{pr}^{qs} \phi_p^*(x_1) \phi_q(x_1) \phi_r^*(x_2) \phi_s(x_2) \tag{2.69}$$

where we have defined

$$\rho_{pr}^{qs} \equiv \langle \Psi | c_p^\dagger c_r^\dagger c_s c_q | \Psi \rangle \, . \tag{2.70}$$

The matrices ρ_1 and ρ_2 formed from the matrix elemetns ρ_q^p and ρ_{pr}^{qs} are referred to as the one- and two-body density matrices.

For the one- and two-body density matrices we can show the following two relations which are useful as test cases when we implement the Configuration Interaction and Coupled Cluster methods:

For the one- and two-body density matrices respectively we have the following relations,

$$\sum_q \langle \Psi | c_q^\dagger c_q | \Psi \rangle = \sum_q \rho_q^q = N \tag{2.71}$$

$$\sum_{pq} \langle \Psi | c_p^\dagger c_q^\dagger c_q c_p | \Psi \rangle = \sum_{pq} \rho_{pq}^{pq} = N(N-1), \tag{2.72}$$

given any normalized state $|\Psi\rangle$.

Proof: Let

$$|\Psi\rangle = \prod_{p=1}^{N} c_p^\dagger \, |-\rangle$$

a normalized reference determinant for a state containing N particles, i.e. $\langle\Psi|\Psi\rangle = 1$. The first relation follows immediately since,

$$\sum_{q=1}^{N} \langle\Psi|c_q^\dagger c_q|\Psi\rangle = \sum_{q=1}^{N} \langle\Psi|\Psi\rangle = N.$$

For the second relation we compute

$$\begin{aligned}
\sum_{p=1}^{N}\sum_{q=1}^{N} \langle\Psi|c_p^\dagger c_q^\dagger c_q c_p|\Psi\rangle \overset{\text{Wick's theorem}}{=} & \sum_{pq}^{N} \langle\Psi|\overbrace{b_p b_q b_q^\dagger b_p^\dagger}|\Psi\rangle + \langle\Psi|\overbrace{b_p b_q b_q^\dagger b_p^\dagger}|\Psi\rangle \\
&= \sum_{pq} -\delta_{pq}\delta_{qp} + \sum_{pq} \delta_{pp}\delta_{qq} \\
&= -\sum_{p}^{N} 1 + \sum_{pq}^{N} 1 = N^2 - N \\
&= N(N-1),
\end{aligned}$$

which shows Eq. (2.72).

Chapter 3

Hartree-Fock theory

In this chapter we will give an overview of Hartree-Fock theory, which is one of the oldest methods of computing an approximative solution to the many-body problem. In Hartree-Fock theory it is assumed that the wavefunction can be approximated by the single Slater determinant, $|\Phi\rangle$ (commonly referred to as the Hartree-Fock state), that minimizes the energy expectation value. This is in many cases a good approximation. When the Hartree-Fock approximation itself is not sufficient it turns out that it is possible to use the Hartree-Fock state as a starting point for more complicated approximations such as the Configuration Interaction or Coupled Cluster methods. The latter two methods are normally referred to as post-Hartree-Fock methods. Thus, Hartree-Fock theory is a natural starting point when discussing many-body methods.

In this chapter we will review the basics of Hartree-Fock theory. We state the so-called Hartree-Fock equations and show how they can be rewritten in terms of a self-consistent eigenvalue problem.

3.1 The Hartree-Fock equations

The starting point in Hartree-Fock theory is that we want to find an approximative solution to the exact ground state, $|\Psi_0\rangle$, of the many-body Hamiltonian $\hat{H}$ which is given by time independent Schrödinger equation,

$$\hat{H}\,|\Psi_0\rangle = E_0\,|\Psi_0\rangle\,. \tag{3.1}$$

In the following we drop the subscript for the ground state wavefunction and just write $|\Psi\rangle \equiv |\Psi_0\rangle$. As mentioned in the introduction to this chapter the fundamental idea in Hartree-Fock theory is to approximate the exact ground

state, $|\Psi\rangle$, by a single Slater determinant

$$|\Phi\rangle = \prod_{i=1}^{N} c_i^\dagger \, |-\rangle = |\phi_1, \cdots, \phi_N\rangle \, .$$

The single-particle functions $\{\phi_i\}_{i=1}^N$ must be orthonormal and $\langle\Phi|\Phi\rangle = 1$. Recall that the Hamiltonian can be written in second quantization as

$$\hat{H} = \sum_{pq} h_q^p c_p^\dagger c_q + \frac{1}{4} \sum_{pqrs} u_{rs}^{pq} c_p^\dagger c_q^\dagger c_s c_r = \hat{H}_0 + \hat{W} \tag{3.2}$$

and observe that,

$$\hat{H}_0 \, |\Phi\rangle = \sum_{pq} h_q^p c_p^\dagger c_q \, |\Phi\rangle = \left(\sum_{i=1}^{N} \epsilon_i \right) |\Phi\rangle \, , \quad \epsilon_i = h_i^i. \tag{3.3}$$

which leads us to the following observation,

$$\hat{H} \, |\Phi\rangle = \left(\hat{H}_0 + \hat{W} \right) |\Phi\rangle = \left(\sum_i \epsilon_i \right) |\Phi\rangle + \hat{W} \, |\Phi\rangle \, . \tag{3.4}$$

The intuitive idea behind the single determinant approximation is that if $\hat{W} \, |\Phi\rangle$ can be considered small in some sense, $|\Phi\rangle$ is a reasonable approximation to the exact eigenfunction $|\Psi\rangle$.

The expectation value of $\hat{H}$ is given by,

$$E[\Phi] = \langle\Phi|\hat{H}|\Phi\rangle = \sum_i \langle\phi_i|\hat{h}|\phi_i\rangle + \frac{1}{2} \sum_{ij} \langle\phi_i\phi_j|\hat{w}|\phi_i\phi_j\rangle_{AS} \, . \tag{3.5}$$

The ground state Hartree-Fock wavefunction is obtained by minimizing E under the constraint that the single-particle functions are orthonormal. In other words, we seek a set of orthonormal single-particle functions, $\{\phi_i\}_{i=1}^N$, such that $|\Phi\rangle = |\phi_1, \cdots, \phi_N\rangle$ is a minimum of E. The solution is denoted by $|\Phi_{\text{HF}}\rangle$ and is referred to as the Hartree-Fock state.

It can be shown [11] that $|\Phi_{\text{HF}}\rangle = |\phi_1, \cdots, \phi_N\rangle$ is an extremal point of $E[\Phi]$ if the single-particle functions satisfy what is known as the canonical Hartree-Fock equations,

$$\hat{f}(\phi_1, \cdots, \phi_N) \, |\phi_i\rangle = \epsilon_i \, |\phi_i\rangle \, , \quad i = 1, \cdots, N, \tag{3.6}$$

which we recognize as a non-linear eigenvalue problem. We name the one-body operator $\hat{f}$ for the Fock operator and it is defined by,

$$\hat{f}(\phi_1, \cdots, \phi_N) \equiv \hat{h} + \hat{v}^{HF}(\phi_1, \cdots, \phi_N) \tag{3.7}$$

with

$$\begin{aligned}\hat{v}^{HF}|\psi\rangle &\equiv \left(\hat{v}^{\text{direct}} - \hat{v}^{\text{exchange}}\right)|\psi\rangle \\ &= \sum_j \left[\int \phi_j^*(x_2)\hat{w}(x_1,x_2)\phi_j(x_2)dx_2\right]\psi(x_1) \\ &- \sum_j \left[\int \phi_j^*(x_2)\hat{w}(x_1,x_2)\psi(x_2)dx_2\right]\phi_j(x_1)\end{aligned}$$

The last expression can be rewritten more compactly if we define the single-particle function $\langle\cdot\phi_1|\hat{w}|\phi_2\phi_3\rangle$,

$$\langle\cdot\phi_1|\hat{w}|\phi_2\phi_3\rangle(x_1) \equiv \int \phi_1^*(x_2)[w(x_1,x_2)\phi_2(x_1)\phi_3(x_2)]dx_2, \tag{3.8}$$

where the inner product with any single-particle funtion ψ is given by,

$$\begin{aligned}\langle\psi|\langle\cdot\phi_1|\hat{w}|\phi_2\phi_3\rangle &= \langle\psi\phi_1|\hat{w}|\phi_2\phi_3\rangle \\ &= \int \psi^*(x_1)\phi_1(x_2)\hat{w}(x_1,x_2)\phi_2(x_1)\phi_3(x_2)dx_1dx_2.\end{aligned}$$

Using this notation, we can write

$$\hat{v}^{\text{HF}}|\psi\rangle = \sum_j \langle\cdot\phi_j|\hat{w}|\psi\phi_j\rangle - \sum_j \langle\cdot\phi_j|\hat{w}|\phi_j\psi\rangle\,. \tag{3.9}$$

Thus, if we can solve the so-called canonical Hartree-Fock equations (3.6) for the single-particle functions $|\phi_i\rangle$, the single Slater determinant $|\Phi\rangle = |\phi_1,\cdots,\phi_N\rangle$ is an extremal point of $E[\Phi]$. Note however that we are not guaranteed that $|\Phi\rangle$ is a global or local minimum. It could also be a saddle point.

Now, let us assume that a solution, $\{\phi_i\}_{i=1}^N$, to the canonical Hartree-Fock equations has been found. Then the Hartree-Fock energy, E_{HF}, is given by,

$$\begin{aligned}E_{HF} = \langle\Phi_{HF}|\hat{H}|\Phi_{HF}\rangle &= \sum_i \langle\phi_i|\hat{h}|\phi_i\rangle + \frac{1}{2}\sum_{ij}\langle\phi_i\phi_j|\hat{w}|\phi_i\phi_j\rangle_{AS} \\ &= \sum_i \langle\phi_i|\hat{f} - \hat{v}^{HF}|\phi_i\rangle + \frac{1}{2}\sum_{ij}\langle\phi_i\phi_j|\hat{w}|\phi_i\phi_j\rangle_{AS} \\ &= \sum_i \epsilon_i - \frac{1}{2}\sum_{ij}\langle\phi_i\phi_j|\hat{w}|\phi_i\phi_j\rangle_{AS}\,.\end{aligned}$$

To establish the last inequality we used that $\hat{f}\,|\phi_i\rangle = \epsilon_i\,|\phi_i\rangle$ by assumption and that,

$$\langle\phi_i|\hat{v}^{HF}|\phi_i\rangle = \langle\phi_i|\sum_j \langle\cdot\phi_j|\hat{w}|\phi_i\phi_j\rangle_{AS} = \sum_j \langle\phi_i\phi_j|\hat{w}|\phi_i\phi_j\rangle_{AS}\,. \tag{3.10}$$

In order to construct the Hartree-Fock state we only use the N first eigenfunctions of $\hat{f}$. In principle we can find a complete basis, $\{\phi_p\}$, of eigenvectors for $\hat{f}$. Using the notation we introduced in the section on particle-hole formalism we write

$$\{\phi_p\} = \{\phi_i\}_{i=1}^{N} \cup \{\phi_a\}_{a=N+1}^{\infty},$$

where the ϕ_i's are the occupied orbitals and the ϕ_a's are unoccupied. This means that we can use the Hartree-Fock state as a reference determinant for further corrections by building $|\Phi_i^a\rangle\,, |\Phi_{ij}^{ab}\rangle\,, \cdots$ where $|\Phi\rangle = |\Phi_{HF}\rangle$ and making the ansatz,

$$|\Psi\rangle = A_0\,|\Phi\rangle + \sum_{ia} A_i^a\,|\Phi_i^a\rangle + \frac{1}{4}\sum_{ijab} A_{ij}^{ab}\,|\Phi_{ij}^{ab}\rangle + \cdots\,. \tag{3.11}$$

We mention a result known as Brillouin's theorem [2], which is important when the Hartree-Fock state is used as a starting point for more complicated approximations such as the Configuration Interaction and Coupled Cluster method. Brillouin's theorem states that if $|\Phi\rangle = |\Phi_{\rm HF}\rangle$ then,

$$\langle\Phi_i^a|\hat{H}|\Phi\rangle = 0. \tag{3.12}$$

3.2 The Roothan-Hall equations

The canonical Hartree-Fock equations (3.6) are formulated as integral-differential equations. We would like to reformulate them as a matrix eigenvalue problem instead, which is normally easier to deal with numerically.

We do this by expanding the Hartree-Fock single-particle functions in terms of a fixed chosen orthogonal basis, $\{\psi_q\}$ (such as the Harmonic oscillator functions),

$$|\phi_p\rangle = \sum_\alpha u_{\alpha p}\,|\psi_\alpha\rangle\,. \tag{3.13}$$

Note that since $\langle\phi_p|\phi_q\rangle = \delta_{pq}$, this corresponds to a unitary transformation, $U \equiv [u_{qp}]$ with $UU^* = I$, of the HF single-particle functions $|\phi_p\rangle$. To see this

we compute,

$$\begin{aligned}\delta_{pq} &= \langle\phi_q|\phi_p\rangle \\ &= \sum_{\alpha\beta} \langle\psi_\beta|\psi_\alpha\rangle\, u^*_{\beta q} u_{\alpha p} \\ &= \sum_{\alpha\beta} \delta_{\alpha\beta} u^*_{\beta q} u_{\alpha p} \\ &= \sum_{\alpha} u_{\alpha p} u^*_{\alpha q},\end{aligned}$$

Thus, we see that $\sum_\alpha u_{\alpha p} u^*_{\alpha q} = \delta_{qp}$ which is the same as $UU^* = I$. The fact that U is a unitary transformation is important, since unitary transformations preserve norm and orthonormality, which are highly desirable features when we do numerical computations.

Inserting the expansion of Eq. (3.13) into Eq. (3.6) we have,

$$\hat{f} \sum_\alpha u_{\alpha p} |\psi_\alpha\rangle = \epsilon_p \sum_\alpha u_{\alpha p} |\psi_\alpha\rangle \,. \tag{3.14}$$

Left-projecting this equation with $\langle\psi_\beta|$ results in,

$$\begin{aligned}\sum_\alpha F_{\beta\alpha} u_{\alpha p} &= \epsilon_p \sum_\alpha u_{\alpha p} \langle\psi_\beta|\psi_\alpha\rangle \\ &= \epsilon_p \sum_\alpha u_{\alpha p} \delta_{\alpha\beta} \\ &= \epsilon_p u_{\alpha p},\end{aligned}$$

where we have defined,

$$F_{\beta\alpha} \equiv \langle\psi_\beta|\hat{f}|\psi_\alpha\rangle \,. \tag{3.15}$$

The above equation can be written on matrix form

$$F(U)U = U\epsilon, \tag{3.16}$$

where $U = [U_{\alpha p}]$ and ϵ is a diagonal matrix with ϵ_p on its diagonal. $F \equiv [F_{\beta\alpha}]$ is the so-called Fock-matrix. This equation is called the Roothan-Hall equation.

In order to solve the Roothan-Hall equations we need to choose a fixed basis, $\{\psi_\alpha\}_{\alpha=1}^L$, which in practice must be truncated after a finite number of functions L. The choice of basis functions depends on the specific problem under consideration. The quality of the solution depends on the number of basis functions included in the basis and if we at some point do not get a

significant different result when increasing the basis size we say that we have reached the Hartree-Fock limit.

Let us write out the explicit form of the elements of the Fock-matrix

$$\begin{aligned}
F_{\beta\alpha} &= \langle\psi_\beta|\hat{f}|\psi_\alpha\rangle \\
&= \langle\psi_\beta|\hat{h} + \hat{v}^{\mathrm{HF}}|\psi_\alpha\rangle \\
&= \langle\psi_\beta|\hat{h}|\psi_\alpha\rangle + \sum_{j=1}^{N} \langle\psi_\beta\phi_j|\hat{w}|\psi_\alpha\phi_j\rangle - \sum_{j=1}^{N} \langle\psi_\beta\phi_j|\hat{w}|\phi_j\psi_\alpha\rangle \\
&= \langle\psi_\beta|\hat{h}|\psi_\alpha\rangle + \sum_{\gamma\delta}^{L}\sum_{j=1}^{N} u^*_{\gamma j}u_{\delta j} \langle\psi_\beta\psi_\gamma|\hat{w}|\psi_\alpha\psi_\delta\rangle - \sum_{\gamma\delta}^{L}\sum_{j=1}^{N} u^*_{\gamma j}u_{\delta j} \langle\psi_\beta\psi_\gamma|\hat{w}|\psi_\delta\psi_\alpha\rangle \\
&= \langle\psi_\beta|\hat{h}|\psi_\alpha\rangle + \sum_{\gamma\delta}^{L} D_{\gamma\delta} \langle\psi_\beta\psi_\gamma|\hat{w}|\psi_\alpha\psi_\delta\rangle_{\mathrm{AS}}\,, \qquad (3.17)
\end{aligned}$$

where we have defined the density matrix,

$$D_{\gamma\delta} = \sum_{j=1}^{N} u^*_{\gamma j}u_{\delta j}. \qquad (3.18)$$

Thus, we see that we have to determine the one- and two-body integral elements $\langle\psi_\beta|\hat{h}|\psi_\alpha\rangle$ and $\langle\psi_\beta\psi_j|\hat{w}|\psi_\alpha\psi_j\rangle$ in the chosen basis before starting to solve Eq. (3.16).

The Roothan-Hall equation is a non-linear eigenvalue problem due to the fact the Fock matrix depends on U. Thus, it has to be solved iteratively. One possibility is to solve equation (3.16) by what is known as self-consistent field iterations. This is the strategy we have chosen and which will be described in more detail in the implementation and results chapter.

Finally, we comment that when a solution U has been found the Hartree-Fock state is given by the N first eigenvectors of F represented by the columns of U. The one- and two-body integral elements in the Hartree-Fock basis can be found by computing,

$$\langle\phi_p|\hat{h}|\phi_q\rangle = \sum_{\alpha\beta} u^*_{\alpha p}u_{\beta q} \langle\psi_p|\hat{h}|\psi_q\rangle\,, \qquad (3.19)$$

$$\langle\phi_p\phi_q|\hat{w}|\phi_r\phi_s\rangle = \sum_{\alpha\beta\gamma\delta} u^*_{\alpha p}u^*_{\beta q}u_{\gamma r}u_{\delta s} \langle\psi_p\psi_q|\hat{w}|\psi_r\psi_s\rangle \qquad (3.20)$$

which are useful as a starting point for more advanced approximations.

Chapter 4

The Configuration Interaction Method

Having introduced the Hartree-Fock method in the previous chapter, we continue our review/survey of many-body methods by introducing the Configuration Interaction (CI) method. The CI method is a conceptually simple and powerful method, but suffers from exponential scaling with respect to the number of particles N present. Thus, CI is only feasible for small systems. However, it is of great value as a way to benchmark the implementation of other approximative methods since they can be compared with results obtained with CI for small systems.

Furthermore, it is quite straightforward to extend the CI method to the time domain which serve as a starting point for time dependent studies. The time dependent version is referred to as time-dependent Configuration Interaction (TDCI) method.

4.1 Time Independent Configuration Interaction

As usual, our starting point is the time-independent Schrödinger equation,

$$\hat{H} \left|\Psi_k\right\rangle = E_k \left|\Psi_k\right\rangle, \tag{4.1}$$

and we want to find the ground state of $\hat{H}$, i.e the eigenpair $(\left|\Psi_0\right\rangle, E_0)$, corresponding to the lowest eigenvalue E_k. In Hartree-Fock theory we used a single Slater determinant as an approximation to the exact $\left|\Psi\right\rangle$. In the CI method however we write $\left|\Psi\right\rangle$ as a linear combination of Slater determinants. The idea is then to write the TISE (4.1) as a matrix eigenvalue problem which can be solved by diagonalization.

4.1.1 Full Configuration Interaction

Assume now that we are given an orthonormal Slater determinant basis $\{|\Phi_I\rangle\}$ built from a set of orthonormal single-particle functions $\{\phi_i\}$. Then we can expand an arbitrary wavefunction $|\Psi_J\rangle$ in this basis as follows,

$$|\Psi_J\rangle = \sum_K A_{KJ} |\Phi_K\rangle . \tag{4.2}$$

Inserting this expansion into (4.1) we obtain,

$$\sum_K \hat{H} |\Phi_K\rangle A_{KJ} = \sum_K E_J A_{KJ} |\Phi_K\rangle . \tag{4.3}$$

If we left-project the above equation with $\langle\Phi_I|$ we obtain

$$\begin{aligned} \sum_K \langle\Phi_I|\hat{H}|\Phi_K\rangle A_{KJ} &= \sum_K E_J A_{KJ} \langle\Phi_I|\Phi_K\rangle \\ &= E_J A_{IJ}, \end{aligned}$$

where the last equality follows by the assumption $\langle\Phi_I|\Phi_K\rangle = \delta_{IK}$. The above equation can be written as an eigenvalue problem,

$$HA = AE, \tag{4.4}$$

where $H = [H_{IJ}]$ is the Hamiltonian matrix, $A = [A_{IJ}]$ is a matrix containing the expansion coefficients and E is a diagonal matrix with the eigenvalues E_k on its diagonal. Here we have defined

$$H_{IJ} \equiv \langle\Phi_I|\hat{H}|\Phi_J\rangle . \tag{4.5}$$

Thus, the TISE can now be solved by diagonalizing H. A straightforward diagonalization of H would give all eigenpairs, $(|\Psi_k\rangle , E_k)$, not just the ground state. Notice that column k of A contains the expansion coefficients of $|\Psi_k\rangle$. This way of solving the TISE is known as the Configuration Interaction method. The naming configuration results from every possible way of ordering the single-particle states as a Slater determinant. Such an ordering is usually referred to as a configuration.

Now, if we want to solve the eigenproblem (4.4) in practice we must truncate the Slater determinant basis, $\{\Phi_I\}_{I=1}^{N_{\text{sd}}}$, such that $|\Psi_k\rangle$ is written as a linear combination of a finite number of N_{sd} determinants built from a finite number L of single-particle functions, $\{\phi_i\}_{i=1}^{L}$. As in Hartree-Fock theory the choice of single-particle functions depends on the specific system under consideration.

In order to diagonalize the matrix H we have to compute its elements H_{IJ}. If we write out the expression for the matrix elements we have,

$$\begin{aligned}\langle\Phi_I|\hat{H}|\Phi_J\rangle = &\sum_{pq}\langle\phi_p|\hat{h}|\phi_q\rangle\langle\Phi_I|c_p^\dagger c_q|\Phi_J\rangle \\ &+\frac{1}{4}\sum_{pqrs}\langle\phi_p\phi_q|\hat{w}|\phi_r\phi_s\rangle_{\mathrm{AS}}\langle\Phi_I|c_p^\dagger c_q^\dagger c_s c_r|\Phi_J\rangle\,.\end{aligned}$$

Again we see that if we can determine the integral elements $\langle\phi_p|\hat{h}|\phi_q\rangle$ and $\langle\phi_p\phi_q|\hat{w}|\phi_r\phi_s\rangle$ in the chosen basis, $\{\phi_i\}$, we can in principle solve the TISE.

Ideally we would like to include all linearly independent Slater determinants in the expansion (4.2). If we use all determinants we get what is called the Full Configuration Interaction (FCI) method. The power of FCI is that it provides an exact result within the computational space defined by the single-particle basis $\{\phi_i\}_{i=1}^L$. However, if we look at a system of N particles the number of linearly independent Slater determinants are,

$$N_{\mathrm{sd}} = \binom{L}{N} = \frac{L!}{N!(L-N)!} \tag{4.6}$$

which grows exponentially. Thus, FCI is only feasible for relatively small systems. Nevertheless, the FCI method is an invaluable tool since we can benchmark the implementation of other approximative methods for small systems with results obtained with FCI.

4.1.2 Hierarchical CI

One way to adress the exponential scaling of the FCI method is to choose the Slater determinants in a systematic way. In particular, we can choose the Slater determinants as excitations relative to a reference, $|\Phi\rangle$, such as the Hartree-Fock state.

Now, let $|\Phi\rangle$ be a reference determinant and define the operator $\hat{C}$ as,

$$\hat{C} = \hat{C}_1 + \hat{C}_2 + \cdots = \sum_{ia} A_{ia}c_a^\dagger c_i + \frac{1}{4}\sum_{ijab} A_{ijab}c_a^\dagger c_b^\dagger c_j c_i + \cdots\,, \tag{4.7}$$

where $i,j = 1,\cdots,N$ and $a,b = N+1,\cdots,L$. Then the expansion (4.2) can be written as

$$|\Psi\rangle = A_0|\Phi\rangle + \left(\sum_{ia} A_{ia}c_a^\dagger c_i + \frac{1}{4}\sum_{ijab} A_{ijab}c_a^\dagger c_b^\dagger c_j c_i + \cdots\right)|\Phi\rangle \tag{4.8}$$

$$= A_0|\Phi\rangle + \sum_{ia} A_{ia}|\Phi_i^a\rangle + \frac{1}{4}\sum_{ijab} A_{ijab}|\Phi_{ij}^{ab}\rangle + \cdots\,, \tag{4.9}$$

where

$$|\Phi_i^a\rangle = c_a^\dagger c_i \,|\Phi\rangle\,,$$
$$|\Phi_{ij}^{ab}\rangle = c_a^\dagger c_b^\dagger c_j c_i \,|\Phi\rangle\,.$$

Using this representation we get a systematic way to refine the computational space. The space defined by $\mathcal{V}_{CIS} = \text{span}\{|\Phi\rangle\,, |\Phi_i^a\rangle\}$ is known as the CI-singles (CIS) space and $\mathcal{V}_{CISD} = \text{span}\{|\Phi\rangle\,, |\Phi_i^a\rangle\,, |\Phi_{ij}^{ab}\rangle\}$ is the CI-singles-doubles (CISD) space and so on. If all excitations are included we regain the FCI method. Observe that in the special case of $N = 2$ particles, the CISD space is equivalent with the FCI space.

If we now consider the CISD space, we see that the number of linearly independent Slater determinants are given by,

$$N_{\text{CISD}} = N(L-N) + \frac{1}{4}N^2(L-N)^2 \tag{4.10}$$

which grows like $\mathcal{O}(N^2(L-N)^2)$. This is a vast improvent over the exponential scaling of the FCI method. However, it turns out that truncated CI methods are problematic since they are not size-consistent and extensive [14]. On the other hand, truncated Coupled Cluster methods which we discuss in the next chapter, are size-consistent and extensive and are therefore favored over truncated CI methods.

Furthermore, as we demonstrate in the next chapter on Coupled Cluster theory, the Configuration Interaction and Coupled Cluster spaces are equivalent for $N = 2$ particles. The CI approach (with its hierarchy of truncations) is comparatively much easier to implement for small systems than CC methods and are a great tool for verfifying an implementation of the Coupled Cluster method. This is the main motivation for introducing the CI method in this work. In the implementation chapter we describe a straightforward implementation of the CISD (FCI for $N = 2$) method, which later on is used extensively to check the validity of our implementation of the Coupled Cluster method.

4.2 Time Dependent Configuration Interaction

We have shown how to compute the eigenfunctions of $\hat{H}$ using the CI method. Recall that the time evolution of an arbitrary state, $|\Psi(t)\rangle$ with initial condition $|\Psi(t_0)\rangle$, is given by the time dependent Schrödinger equation,

$$i\hbar\frac{\partial}{\partial t}\,|\Psi(t)\rangle = \hat{H}(t)\,|\Psi(t)\rangle\,. \tag{4.11}$$

We will take the initial condition to be some (normalized) linear combination of the CI eigenfunctions. In particular, we are interested in the time evolution of the ground state.

The time-dependent Configuration Interaction (TDCI) method is obtained by placing a time-dependence on the expansion coefficients in the CI-wavefunction while keeping the single-particle basis fixed. Analogous to what we did in the previous section, we expand $|\Psi(t)\rangle$ in a given orthonormal Slater determinant basis $\{|\Phi_I\rangle\}$

$$|\Psi(t)\rangle = \sum_I A_I(t)\,|\Phi_I\rangle\,. \tag{4.12}$$

If we now insert this expansion into the time-dependent Schrödinger equation we obtain,

$$i\hbar \sum_I \frac{\partial A_I(t)}{\partial t}\,|\Phi_I\rangle = \sum_I A_I(t)\hat{H}(t)\,|\Phi_I\rangle\,. \tag{4.13}$$

Left-projecting the above equation with $\langle\Phi_J|$ and using the orthonormality of the Slater determinants, we are left with a differential equation for each expansion coefficient $A_J(t)$

$$i\hbar\frac{\partial A_J(t)}{\partial t} = \sum_I A_I(t)\,\langle\Phi_J|\hat{H}(t)|\Phi_I\rangle\,. \tag{4.14}$$

Equivalently we can write this as the matrix-vector equation

$$i\hbar\frac{\partial A(t)}{\partial t} = H(t)A(t), \tag{4.15}$$

where $H(t)$ is the time dependent Hamilton matrix defined in terms of its elements,

$$H_{IJ}(t) = \langle\Phi_I|\hat{H}(t)|\Phi_J\rangle\,, \tag{4.16}$$

and $A(t) = [A_0(t), A_1(t), \cdots]$ is the vector containing the expansion coefficients. We refer to this as the time dependent Configuration Interaction (TDCI) method.

Completely analogous to the time independent case we have a hierarchy of TDCI methods defined by our CI space. Thus we have a series of possible approximations ranging from TDCI to TDFCI. If we did not have to truncate the Slater determinant basis, TDFCI would be an exact solution to the TDSE. However, we must in practice truncate the basis and TDFCI suffers from the fact that the size of FCI-space grows exponentially. In order to obtain a good approximation to the exact solution we would need a huge fixed basis [1].

On the other hand it is quite straightforward to implement, especially when a program for computing the CI-ground state already has been written. Again, the fact that the CI and CC space are equivalent for $N = 2$ makes it worthwhile to write a TDCI program, which we can use to compare with the time dependent Coupled Cluster method discussed in the next chapter.

4.3 Density matrices

In addition to computing the energy we want to compute the single-particle density which is given in terms of the one-body density matrix and the single-particle functions as

$$\rho(\vec{r}) = \sum_{pq}^{L} \rho_p^q \phi_p(\vec{r}) \phi_q^*(\vec{r}). \tag{4.17}$$

Using a Slater determinant basis we have in turn

$$\rho_p^q = \langle \Psi | c_p^\dagger c_q | \Psi \rangle = \sum_{IJ} A_I^* A_J \langle \Phi_I | c_p^\dagger c_q | \Phi_J \rangle . \tag{4.18}$$

Chapter 5

Coupled Cluster Theory

In this section we review Coupled Cluster (CC) theory. We will look at the "classical" way of obtaining the CC equations. The presentation borrows heavily from the excellent review by Crawford and Schaefer [6].

Furthermore we look at a different approach which in addition to the usual CC equations also gives a method to solve the time dependent Schrödinger equation, the so-called Orbital Adaptive Coupled Cluster (OATDCC) method [1].

5.1 The exponential ansatz

As usual we start by considering the time-independent Schrödinger equation

$$\hat{H} \left|\Psi\right\rangle = E \left|\Psi\right\rangle .$$

In Coupled Cluster theory an exponential ansatz

$$\left|\Psi\right\rangle_{CC} \equiv e^{\hat{T}} \left|\Phi\right\rangle , \tag{5.1}$$

is used to approximate the exact solution. Here $\left|\Phi\right\rangle$ is a reference Slater determinant and $\hat{T} = \sum_i^n \hat{T}_i$ is the sum over n-orbital cluster operators.

The the one- and two-orbital cluster operators are defined as

$$\hat{T}_1 \equiv \sum_{ia} t_i^a c_a^\dagger c_i \tag{5.2}$$

and

$$\hat{T}_2 \equiv \frac{1}{4} \sum_{ijab} t_{ij}^{ab} c_b^\dagger c_j c_a^\dagger c_i . \tag{5.3}$$

Generally an n-orbital cluster operator is defined as

$$\hat{T}_n \equiv \left(\frac{1}{n!}\right)^2 \sum_{ijk\ldots abc\ldots} t_{ijk\ldots}^{abc\ldots} \ldots c_c^\dagger c_k c_b^\dagger c_j c_a^\dagger c_i. \tag{5.4}$$

We see that the cluster operators are linear combinations of excitation operators as defined by (2.46). Recalling the exponential series we can write (see the derivation below),

$$|\Psi\rangle_{CC} = e^{\hat{T}} |\Phi\rangle = \left(1 + \sum_{k=1}^{\infty} \frac{1}{k!} \hat{T}^k\right) |\Phi\rangle\,, \tag{5.5}$$

where T contains the one-body, two-body etc cluster amplitues defined above.

Properties of Cluster operators.

- $\hat{T}_{N+1} |\Phi\rangle = 0$, where $\hat{T} = \hat{T}_1 + \hat{T}_2 + \cdots$, $|\Phi\rangle$ a reference determinant and N the number of particles in $|\Phi\rangle$. This implies that for a given N the cluster operator $\hat{T}$ has a finite number of terms,

$$\hat{T} = \hat{T}_1 + \hat{T}_2 + \cdots + \hat{T}_N. \tag{5.6}$$

- The rank of an excitation/cluster operator is defined as the number of creation or annihilation operators, i.e $\text{rank}\left(\hat{T}_2\right) = 2$.

- The product of two excitation/cluster operators is an excitation/-cluster operator.

- All excitation/cluster operators commute. We can prove this by considering the action of the commutator on a reference determinant as follows,

$$\begin{aligned}
[c_b^\dagger c_j, c_a^\dagger c_i] |\Phi\rangle &= \left(c_b^\dagger c_j c_a^\dagger c_i - c_a^\dagger c_i c_b^\dagger c_j\right) |\Phi\rangle \\
&= |\Phi_{ij}^{ab}\rangle\, |\Phi_{ji}^{ba}\rangle \\
&= |\Phi_{ij}^{ab}\rangle - |\Phi_{ij}^{ab}\rangle \\
&= 0 \cdot |\Phi_{ij}^{ab}\rangle\,.
\end{aligned}$$

  Thus $c_b^\dagger c_j c_a^\dagger c_i = c_a^\dagger c_i c_b^\dagger c_j$, meaning that $c_b^\dagger c_j$ and $c_a^\dagger c_i$ commute. The result for a general excitation operator follows by induction.

- $\text{rank}\left(\hat{T}_n \hat{T}_m\right) = n + m$.
- Excitation operators are nilpotent,

$$\hat{\tau}_X^2 = \left(\cdots c_c^\dagger c_k c_b^\dagger c_j c_a^\dagger c_i\right)^2 = 0. \tag{5.7}$$

  This follows immediately from the fact that,

$$\left(c_a^\dagger c_i\right)^2 |\Phi\rangle = c_a^\dagger c_i \, |\Phi_i^a\rangle = 0$$

  since $c_i \, |\Phi_i^a\rangle = 0$.

- $\hat{\tau}_X^\dagger = \left(\cdots c_c^\dagger c_k c_b^\dagger c_j c_a^\dagger c_i\right)^\dagger = \left(c_i^\dagger c_a c_j^\dagger c_b c_k^\dagger c_c \cdots\right)$ is a de-excitation operator. Thus,

$$\langle\Phi| \, \hat{\tau}_X = \hat{\tau}_X^\dagger \, |\Phi\rangle \equiv 0. \tag{5.8}$$

- Since all cluster operators commute, $e^{\hat{T}+\hat{T}'} = e^{\hat{T}} e^{\hat{T}'}$.

We can now state a fundamental result in CC theory:

Existence of cluster expansion.

Suppose $\langle\Phi|\Psi\rangle = 1$ (normally referred to as intermediate normalization). Then there exist cluster operators $\hat{T}$ and $\hat{A}$ such that

$$|\Psi\rangle = \left(1 + \hat{A}\right) |\Phi\rangle = e^{\hat{T}} \, |\Phi\rangle \, . \tag{5.9}$$

Proof: Let X denote a general excitation index and let τ_X be a general excitation operator. Assume that $\langle\Phi|\Psi\rangle = 1$ and let

$$\{|\Phi\rangle \, , |\Phi_X\rangle \, | \text{ for all } X\}$$

be a Slater determinant basis where $\langle\Phi|\Phi_X\rangle = 0$. We can expand $|\Psi\rangle$ in the basis, giving

$$|\Psi\rangle = A_0 \, |\Phi\rangle + \sum_X A_X \, |\Phi_X\rangle \, , \quad A_0, A_X \in \mathbb{C}. \tag{5.10}$$

Note that $\langle\Phi|\Psi\rangle = 1$ implies that $A_0 = 1$. Using $|\Phi_X\rangle = \hat{\tau}_X \, |\Phi\rangle$ we get,

$$|\Psi\rangle = |\Phi\rangle + \sum_X A_X \hat{\tau}_X \, |\Phi\rangle = \left(1 + \hat{A}\right) |\Phi\rangle \, , \tag{5.11}$$

where we have defined $\hat{A} \equiv \sum_X A_X \hat{\tau}_X$. We now want to find $\hat{T}$ such that $e^{\hat{T}} = 1 + \hat{A}$. Rewrite $\hat{A}$ as,

$$\begin{aligned}\hat{A} = \sum_X A_X \hat{\tau}_X &= \sum_{ia} A_i^a \hat{\tau}_i^a + \frac{1}{2!^2} \sum_{ijab} A_{ij}^{ab} \hat{\tau}_{ij}^{ab} + \cdots \\ &= \hat{A}_1 + \hat{A}_2 + \cdots\end{aligned}$$

By expanding $e^{\hat{T}}$ as a series

$$e^{\hat{T}} = 1 + \hat{T} + \frac{1}{2}\hat{T}^2 + \cdots, \tag{5.12}$$

and note that $e^{\hat{T}} = 1 + \hat{A}$ if they have the same singles, doubles, etc. parts. Let N be given and insert $\hat{T} = \hat{T}_1 + \cdots + \hat{T}_N$ into Eq. (5.12), expand each power and group terms of equal rank together,

$$\begin{aligned}e^{\hat{T}} &= 1 + \hat{T} + \frac{1}{2}\hat{T}^2 + \frac{1}{6}\hat{T}^3 + \cdots \\ &= 1 + \left(\hat{T}_1 + \cdots + \hat{T}_N\right) + \frac{1}{2}\left(\hat{T}_1 + \cdots + \hat{T}_N\right)^2 \\ &+ \frac{1}{6}\left(\hat{T}_1 + \cdots + \hat{T}_N\right)^3 + \frac{1}{24}\left(\hat{T}_1 + \cdots + \hat{T}_N\right)^4 + \cdots \\ &= 1 + \underbrace{\hat{T}_1}_{\text{rank1}} + \underbrace{\left(\hat{T}_2 + \frac{1}{2}\hat{T}_1^2\right)}_{\text{rank2}} + \underbrace{\left(\hat{T}_3 + \frac{1}{6}\hat{T}_1^3 + \hat{T}_1\hat{T}_2\right)}_{\text{rank3}} + \cdots\end{aligned}$$

Note that in the last equation, the rank k term contains only one term from $\hat{T}_k$ while all other terms are products of cluster operators of lower rank. For the above expression to be equal to $1 + \hat{A}_1 + \hat{A}_2 + \cdots$ each rank k term must equal $\hat{A}_k$.

$$\begin{aligned}\hat{A}_1 &= \hat{T}_1 \\ \hat{A}_2 &= \hat{T}_2 + \frac{1}{2}\hat{T}_1^2 \\ \hat{A}_3 &= \hat{T}_3 + \frac{1}{6}\hat{T}_1^3 + \hat{T}_1\hat{T}_2 \\ &\vdots\end{aligned}$$

Then we can find $\hat{T}_k$ recursively,

$$\hat{T}_1 = \hat{A}_1 \tag{5.13}$$

$$\hat{T}_2 = \hat{A}_2 - \frac{1}{2}\hat{T}_1^2 = \hat{A}_2 - \frac{1}{2}\hat{A}_1^2 \tag{5.14}$$

$$\hat{T}_3 = \hat{A}_3 - \frac{1}{6}\hat{T}_1^3 - \hat{T}_1\hat{T}_2 \tag{5.15}$$

$$= \hat{A}_3 - \frac{1}{6}\hat{A}_1^3 - \hat{A}_1\left(\hat{A}_2 - \frac{1}{2}\hat{A}_1^2\right) \tag{5.16}$$

$$\vdots \tag{5.17}$$

Hence, given $|\Psi\rangle$, we can find $\hat{A}$ which in turn gives $\hat{T}$.

We notice that the expansion $|\Psi\rangle = (1 + \hat{A})|\Phi\rangle$ actually is the CI expansion. Thus, the CC expansion $e^{\hat{T}}|\Phi\rangle$ is a nonlinear reparametrization of the linear CI parametrization. In practice, for a system containing N particles, one must truncate $\hat{T} = \hat{T}_1 + \cdots + \hat{T}_N$ and $\hat{A} = \hat{A}_1 + \cdots + \hat{A}_N$. Now, truncate the expansion at the k-th operator in the sum with $1 \leq k \leq N$ such that $\hat{T} = \hat{T}_1 + \cdots + \hat{T}_k$ and $\hat{A} = \hat{A}_1 + \cdots + \hat{A}_k$. Observe that

$$|\Psi\rangle = (1 + \hat{A}_1 + \cdots + \hat{A}_k)|\Phi\rangle,$$

is a linear combination where the highest excitation contribution has rank k. The exponential ansatz on the other hand

$$e^{\hat{T}_1 + \cdots + \hat{T}_k}|\Phi\rangle = \left(1 + \left(\hat{T}_1 + \cdots + \hat{T}_k\right) + \frac{1}{2}\left(\hat{T}_1 + \cdots + \hat{T}_k\right)^2 + \cdots\right)|\Phi\rangle,$$

contains contributions from determinants of higher excitation than rank k which is immediately clear from the inclusion of the term $\hat{T}_k^2$ in the second parenthesis in the above expression. It turns out that this gives the truncated CC ansatz an advantage over the truncated CI ansatz. The latter is not size-consistent and extensive while the former is.

Loosely speaking size-consistency and extensivity means that if A and B are two noninteracting subsystems, we should get the same energy for the supersystem AB irrespective of whether we have carried out the calculations for each subsystem separately or for both subsystems simultaneously. In general truncated exponential wavefunctions have this property while truncated linear wavefunctions does not [14].

Depending on the truncation we get a sequence of more and more accurate CC approximations:

- $|\Psi_{CCS}\rangle = e^{\hat{T}_1}|\Phi\rangle$ coupled-cluster singles (CCS)
- $|\Psi_{CCD}\rangle = e^{\hat{T}_2}|\Phi\rangle$ coupled-cluster doubles (CCD)
- $|\Psi_{CCSD}\rangle = e^{\hat{T}_1+\hat{T}_2}|\Phi\rangle$ coupled-cluster singles doubles (CCSD)
- $|\Psi_{CCSDT}\rangle = e^{\hat{T}_1+\hat{T}_2+\hat{T}_3}|\Phi\rangle$ coupled-cluster singles doubles triples (CCSDT)

Equality of CC and CI for $N = 2$.

An important special case is the case $N = 2$. In that case the CC expansion equals the CI expansion. This is an invaluable information when we want to implement the CC method since we can then compare it with the corresponding CI method to verify the implementation. To see this let $N = 2$ and let $|\Phi\rangle$ be a reference determinant as usual. Observe that $\hat{A}_3|\Phi\rangle = 0$ and $\hat{T}_3 = 0$. Thus, $\hat{A} = \hat{A}_1 + \hat{A}_2$ and $\hat{T} = \hat{T}_1 + \hat{T}_2$ will give all non-zero contributions. Additionaly all products of cluster operators where the product has rank larger than 2, acting on the reference will give a zero result. Inserting $\hat{T}_1 = \hat{A}_1$ and $\hat{T}_2 = \hat{A}_2 - \frac{1}{2}\hat{A}_1^2$ into the CC expansion including only terms of at most rank 2 we obtain,

$$\begin{aligned} e^{\hat{T}_1+\hat{T}_2}|\Phi\rangle &= \left(1 + \underbrace{\hat{T}_1}_{\text{rank1}} + \underbrace{\left(\hat{T}_2 + \frac{1}{2}\hat{T}_1^2\right)}_{\text{rank2}}\right)|\Phi\rangle \\ &= \left(1 + \hat{A}_1 + \hat{A}_2 - \frac{1}{2}\hat{A}_1^2 + \frac{1}{2}\hat{A}_1^2\right)|\Phi\rangle \\ &= \left(1 + \hat{A}_1 + \hat{A}_2\right)|\Phi\rangle \end{aligned}$$

which indeed is the CI expansion. For $N = 2$, $\hat{A} = \hat{A}_1 + \hat{A}_2$ corresponds to a full configuration interaction treatment, while $\hat{T} = \hat{T}_1 + \hat{T}_2$ CCSD. For the two-particle case CCSD is actually equal to FCI. Morover, if we consider only $\hat{T} = \hat{T}_2$ (CCD), we get

$$\begin{aligned} |\Psi\rangle_{\text{CCD}} = e^{\hat{T}_2}|\Phi\rangle &= \left(1 + \hat{T}_2\right)|\Phi\rangle \\ &= \left(1 + \hat{A}_2 - \frac{1}{2}\hat{A}_1^2 + \frac{1}{2}\hat{A}_1^2\right)|\Phi\rangle \\ &= \left(1 + \hat{A}_2\right)|\Phi\rangle = |\Psi\rangle_{\text{CID}}\,. \end{aligned}$$

The importance of these two results is that one can verify a CCD implementation by comparing with a CID implementation and a CCSD implementation can be verified by comparing with an FCI/CISD implementation.

5.2 The Coupled Cluster Equations

We now proceed by inserting expression (5.1) into the time-independent Schrödinger equation

$$\hat{H}e^{\hat{T}}\left|\Phi\right\rangle = Ee^{\hat{T}}\left|\Phi\right\rangle. \tag{5.18}$$

Left-projecting with $\langle\Phi|$ we get an expression for the energy,

$$\begin{aligned}\langle\Phi|\hat{H}e^{\hat{T}}|\Phi\rangle &= E\left\langle\Phi|e^{\hat{T}}|\Phi\right\rangle = E\left\langle\Phi|\Psi_{CC}\right\rangle \\ &= E\end{aligned}$$

where intermediate normalization, $\langle\Phi|\Psi_{CC}\rangle = 1$, is assumed. Furthermore one can obtain an expression for the amplitude $t_{ij\ldots}^{ab\cdots}$ upon left-projection with $\langle\Phi_{ij\ldots}^{ab\cdots}|$,

$$\langle\Phi_{ij\ldots}^{ab\cdots}|\hat{H}e^{\hat{T}}|\Phi\rangle = E\left\langle\Phi_{ij\ldots}^{ab\cdots}|e^{\hat{T}}|\Phi\right\rangle. \tag{5.19}$$

Notice that this equations are non-linear (due to the presence of $e^{\hat{T}}$) and depend on the energy equation. This is impractical and is resolved by performing a similarity transformation of the Hamiltonian.

Multiply Eq. (5.18) with $e^{-\hat{T}}$ and left-project with $\langle\Phi|$ in order to obtain

$$\langle\Phi|e^{-\hat{T}}\hat{H}e^{\hat{T}}|\Phi\rangle = E\left\langle\Phi|e^{-\hat{T}}e^{\hat{T}}|\Phi\right\rangle = E. \tag{5.20}$$

Left-projection with $\langle\Phi_{ij\ldots}^{ab\cdots}|$ now results in,

$$\langle\Phi_{ij\ldots}^{ab\cdots}|e^{-\hat{T}}\hat{H}e^{\hat{T}}|\Phi\rangle = E\left\langle\Phi_{ij\ldots}^{ab\cdots}|\Phi\right\rangle = 0. \tag{5.21}$$

Equations (5.20) and (5.21) are commonly referred to as the Coupled Cluster equations. Notice now that the equation for the amplitudes does not depend on the energy. The amplitude equation is non-linear and has to be solved iteratively. We say that we have a self-consistent solution to CC equations if the energy converges to some pre-defined precision.

In order to implement the Coupled Cluster method we need algebraic expressions for the CC equations.

Theorem 3 (Baker-Campbell-Hausdorff expansion)**.** *For any matrices A and S the Baker-Campbell-Hausdorff (BCH) expansion is given by,*

$$e^{-S}Ae^{S} = A + [A,S] + \frac{1}{2}[[A,S],S] + \frac{1}{3!}[[[A,S],S],S] + ..., \tag{5.22}$$

where

$$[A, S] \equiv AS - SA$$

is the commutator between A and S.

Under the assumption that $\hat{H}$ contains at most a two-body operator the BCH-expansion truncates after the first five terms on the right-hand side of Eq. (5.22) [15]. This fact greatly simplifies the evaluation of the left-hand sides of the equations (5.20) and (5.21).

These expressions can be evaluated using the second quantization formalism as demonstrated in [6] or by diagrammatic techniques [5]. Alternatively one can use the second quantization toolbox in the Python library Sympy[1] to compute these expressions.

5.3 A variational CC theory?

One drawback with the CC-method is that it does not satisfy the variational principle of Eq. (2). Consider the operator $e^{-\hat{T}}\hat{H}e^{T}$ which appears in the similarity transformed energy equation (5.20). This operator is not Hermitian. In order to see this compute $\hat{T}_1^\dagger$,

$$\hat{T}_1^\dagger = \left(\sum_{ia} t_i^a c_a^\dagger c_i\right) = \sum_{ia} (t_i^a)^* c_i^\dagger c_a \neq \hat{T}_1. \tag{5.23}$$

Then it follows that

$$\left(e^{-\hat{T}}\hat{H}e^{\hat{T}}\right)^\dagger = (e^{\hat{T}})^\dagger \hat{H} (e^{-\hat{T}})^\dagger = e^{\hat{T}^\dagger}\hat{H}e^{-\hat{T}^\dagger} \neq e^{-\hat{T}}\hat{H}e^{\hat{T}}. \tag{5.24}$$

Hermiticity of the operator was a necessary condition for the variational theorem, thus Eq. (5.20) is not variational. However, if $\hat{T}$ is not truncated the spectrum of $e^{-\hat{T}}\hat{H}e^{T}$ is identical to the original Hermitian operator $\hat{H}$, justifying thereby its use in quantum mechanical models [16].

There have been attempts to construct a variational solution by deriving the amplitude equations by minimizing the functional,

$$\frac{\langle\Psi|\hat{H}|\Psi\rangle}{\langle\Psi|\Psi\rangle} = \frac{\langle\Phi|(e^{\hat{T}})^\dagger\hat{H}e^{T}|\Phi\rangle}{\langle\Phi|(e^{\hat{T}})^\dagger e^{\hat{T}}|\Phi\rangle}. \tag{5.25}$$

Notice that since $(e^{\hat{T}})^\dagger\hat{H}e^{T}$ does not conform to the Baker-Campbell-Hausdorff formula, the series has no natural truncation, complicating thereby matters greatly.

[1] http://docs.sympy.org/latest/modules/physics/secondquant.html

5.4 The Orbital Adaptive Time Dependent Coupled Cluster Method

Kvaal demonstrates in Ref. [1] that it is possible to derive the equations of motion for the CC ansatz by computing critical points of a bivariational expectation value functional. In his approach both the orbitals and CC amplitudes are allowed to vary and the method can be used to solve the time-dependent Schrödinger equation. The method is called the Orbital Adaptive Time-Dependent Coupled Cluster (OATDCC) and is a hierarchical approximation to MCTDHF. Furthermore, if the orbitals are held fixed the method gives an approximation to the Time-Dependent CI method which we refer to as the Time-Dependent Coupled Cluster (TDCC) method. While the TDFCI and MCTDHF approaches suffer from exponential scaling, the OATDCC and TDCC methods achieve polynomial scaling.

The main goal of this thesis is to implement the TDCC method for a time-dependent Hamiltonian. As such we will in the following summarize the key ingredients needed to achieve this. For technical details the interested reader is referred to the article by Kvaal [1].

5.4.1 The Bivariational Principle

As we have established, the Coupled Cluster method is not variational in the usual sense since the similarity transformed Hamiltonian,

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

is not Hermitian. Following Ref. [1], let A be an operator (possibly non-Hermitian) over Hilbert space $\mathcal{H}$ and consider the expectation value functional

$$\mathcal{E}_A : \mathcal{H}' \times \mathcal{H} \to \mathbb{C}, \quad \mathcal{E}_A[\langle \Psi' | , |\Psi\rangle] = \frac{\langle \Psi' | A | \Psi \rangle}{\langle \Psi' | \Psi \rangle}. \tag{5.26}$$

This is referred to as the the bivariational expectation value functional. In contrast to the usual variational principle $\langle \Psi' |$ and $|\Psi\rangle$ are treated as independent parameters. The conditions for $\delta \mathcal{E}_A = 0$, for all independent variations of $\langle \Psi' |$ and $|\Psi\rangle$ is

$$(A - a) |\Psi\rangle = 0 \quad \text{and} \quad \langle \Psi' | (A - a) = 0, \tag{5.27}$$

with

$$a = \mathcal{E}_A[\langle \Psi' | , |\Psi\rangle]$$

being the value of $\mathcal{E}_A$ at the critical point.

Equations of motion can be derived from so-called time-dependent variational principles. One such variational is set up using the Lagrange formulation [17],

$$\delta \int_0^T L[|\Psi\rangle]dt = 0, \tag{5.28}$$

where the Lagrangian functional is given by

$$L[|\Psi\rangle] = \langle\Psi(t)|\hat{H} - i\hbar\frac{\partial}{\partial t}|\Psi(t)\rangle \tag{5.29}$$

with the boundary conditions $\delta L(t_1) = \delta L(t_2) = 0$. The functional

$$\mathcal{S}[|\Psi\rangle] \equiv \int_0^T \langle\Psi(t)|\hat{H} - i\hbar\frac{\partial}{\partial t}|\Psi(t)\rangle\, dt \tag{5.30}$$

is referred to as the action functional and computing equations of motion from $\delta\mathcal{S} = 0$ is known as the prinicipal of least action.

By considering a bivariational generalization of the action functional (5.30),

$$\mathcal{S}[\langle\Psi'|, |\Psi\rangle] \equiv \int_0^T \frac{\langle\Psi'(t)|i\hbar\frac{\partial}{\partial t} - \hat{H}|\Psi(t)\rangle}{\langle\Psi'(t)|\Psi(t)\rangle}dt \tag{5.31}$$

$$\int_0^T i\hbar\frac{\langle\Psi'(t)|\frac{\partial}{\partial t}\Psi(t)\rangle}{\langle\Psi'(t)|\Psi(t)\rangle} - \mathcal{E}_H[\langle\Psi'(t)|, |\Psi(t)\rangle]dt. \tag{5.32}$$

Kvaal [1] derives equations of motions generalizing the Coupled Cluster method to the time domain. Here, one should note the appearance of the bivariational generalization of the energy expectation value,

$$\mathcal{E}_H[\langle\Psi'|, |\Psi\rangle] = \frac{\langle\Psi'|\hat{H}|\Psi\rangle}{\langle\Psi'|\Psi\rangle}, \tag{5.33}$$

which is required to be complex analytic as shown in Ref. [18].

5.4.2 The Coupled Cluster Ansatz

The fundamental idea in Ref. [1] is to introduce different exponential parametrizations $|\Psi\rangle$, $\langle\Psi'|$ and compute $\delta\mathcal{S} = 0$, with $\mathcal{S}$ given by (5.32), which ultimately result in time dependent Coupled Cluster equations.

In particular one makes the ansatzes

$$|\Psi\rangle = e^{\hat{T}}|\phi\rangle \tag{5.34}$$

$$\langle\Psi'| = \langle\tilde{\phi}|\, e^{\hat{T}'}, \tag{5.35}$$

where $\hat{T}$ is the familiar excitation operator

$$\hat{T} = \sum_{ia} \tau_i^a c_a^\dagger \tilde{c}_i + \frac{1}{2!^2} \sum_{ijab} \tau_{ij}^{ab} c_a^\dagger \tilde{c}_i c_b^\dagger \tilde{c}_j + \cdots, \tag{5.36}$$

and $\hat{T}'$ is a de-excitation operator on the form

$$\hat{T}' = \sum_{ia} (\tau')_a^i c_i^\dagger \tilde{c}_a + \frac{1}{2!^2} \sum_{ijab} (\tau')_{ab}^{ij} c_i^\dagger \tilde{c}_a c_j^\dagger \tilde{c}_b + \cdots. \tag{5.37}$$

Here $|\phi\rangle = |\varphi_{p_1}, \cdots, \varphi_{p_N}\rangle$ and $\langle\tilde{\phi}| = \langle\tilde{\varphi}_{p_1}, \cdots, \tilde{\varphi}_{p_N}|$ are Slater determinants built from biorthogonal single-particle functions, $\langle\tilde{\varphi}_p|\varphi_q\rangle = \delta_{pq}$, belonging to separate Hilbert spaces $\mathcal{V}$ and $\tilde{\mathcal{V}}$. The biorthogonality condition implies the anticommutator relation

$$\{\tilde{c}_p, c_q^\dagger\} \equiv \tilde{c}_p c_q^\dagger + c_q^\dagger \tilde{c}_p = \delta_{pq}, \tag{5.38}$$

which preserves Wick's theorem. Here $\tilde{c}_p, \tilde{c}_p^\dagger, c_p$ and $c_p^\dagger$ are creation and annihilation operators associated with the biorthogonal single-particle functions. It should be noted that when $\tilde{c}_p$ or $\tilde{c}_p^\dagger$ acts on the determinant $|\phi\rangle$ it is only responsible for removing or adding φ_p (not $\tilde{\varphi}_p$) from the determinant [1]. The notation $\Phi = (\varphi_1, \cdots, \varphi_L)$ and $\tilde{\Phi} = (\tilde{\varphi}_1, \cdots, \tilde{\varphi}_L)$ to denote the set of orbitals used to build the Slater determinants $|\phi\rangle$ and $\langle\tilde{\phi}|$.

In standard CC theory and virtually every other many-body method, the single-particle functions are taken to be orthogonal such that $\mathcal{V} \equiv \tilde{\mathcal{V}}'$. This relaxation of orthonormality of the single-particle functions is necessary to ensure that the bivariational functional is complex analytic if the single-particle functions are to be treated as variational parameters [1].

Next, a change of variables from (T', T) to (Λ, T), with

$$\Lambda = \sum_{ia} \lambda_a^i c_i^\dagger \tilde{c}_a + \frac{1}{2!^2} \sum_{ijab} \lambda_{ab}^{ij} c_i^\dagger \tilde{c}_a c_j^\dagger \tilde{c}_b + \cdots \tag{5.39}$$

with a subsequent application of a de-excitation operator results in

$$\langle\tilde{\Psi}| = \frac{\langle\Psi'|}{\langle\Psi'|\Psi\rangle} = \langle\tilde{\phi}|\,(1+\Lambda)e^{-T}. \tag{5.40}$$

Using this parametrization $\langle\tilde{\Psi}|\Psi\rangle = 1$. The amplitudes $\{\tau_i^a, \tau_{ij}^{ab}, \cdots\}$ are the usual Coupled Cluster amplitudes which we refer to as the excitation amplitudes or τ-amplitudes. Furthermore, we have introduced a second set of amplitudes $\{\lambda_a^i, \lambda_{ab}^{ij}, \cdots\}$ which we will refer to as the de-excitation amplitudes or λ-amplitudes.

The bivariational expectation value functional (5.33) reads

$$\mathcal{E}_H[\lambda,\tau,\tilde{\Phi},\Phi] = \langle\tilde{\phi}|(I+\Lambda)e^{-T}\hat{H}e^{T}|\phi\rangle\,. \tag{5.41}$$

Inserting the wavefunction ansatzes into the action functional of Eq. (5.32) one arrives at the following expression,

$$\mathcal{S}[\lambda,\tau,\tilde{\Phi},\Phi] = \int_0^T i\hbar\,\langle\tilde{\phi}|(I+\Lambda)e^{-T}\frac{\partial}{\partial t}e^{T}|\phi\rangle - \mathcal{E}_H[\lambda(t),\tau(t),\tilde{\Phi}(t),\Phi(t)]dt. \tag{5.42}$$

Kvaal then shows that the action can be written,

$$\mathcal{S}[\lambda,\tau,\tilde{\Phi},\Phi] = \int_0^T i\hbar\sum_\mu \lambda_\mu\dot{\tau}^\mu - \mathcal{E}_{H-i\hbar D_0}[\lambda,\tau,\tilde{\Phi},\Phi]dt \tag{5.43}$$

$$= \int_0^T i\hbar\lambda_\mu\dot{\tau}^\mu + \rho^q_p(h^p_q - i\hbar\eta^p_q) + \frac{1}{4}\rho^{qs}_{pr}u^{pr}_{qs}dt, \tag{5.44}$$

where the operator D_0 is defined as,

$$D_0 \equiv \sum_{pq}\langle\tilde{\varphi}_p|\dot{\varphi}_q\rangle\, c^\dagger_p\tilde{c}_q. \tag{5.45}$$

Here μ denotes a general excitation $\{ia, ijab, \cdots\}$. Furthermore,

$$\begin{aligned}
\rho^q_p &= \rho^q_p(\lambda,\tau) \equiv \langle\tilde{\Psi}|c^\dagger_p\tilde{c}_q|\Psi\rangle\,,\\
\rho^{qs}_{pr} &= \rho^{qs}_{pr}(\lambda,\tau) \equiv \langle\tilde{\Psi}|c^\dagger_p c^\dagger_r\tilde{c}_s\tilde{c}_q|\Psi\rangle\,,\\
h^p_q &= h^p_q(\tilde{\Phi},\Phi) \equiv \langle\tilde{\varphi}_p|h|\varphi_q\rangle\,,\\
\eta^p_q &= \eta^p_q(\tilde{\Phi},\dot{\Phi}) \equiv \langle\tilde{\varphi}_p|\dot{\varphi}_q\rangle\,,\\
u^{pr}_{qs} &= u^{pr}_{qs}(\tilde{\Phi},\Phi) \equiv \langle\tilde{\varphi}_p\tilde{\varphi}_r|u|\varphi_q\varphi_s\rangle_{AS}\,.
\end{aligned}$$

The quantities ρ^q_p and ρ^{qs}_{pr} are the CC reduced one- and two-body density matrices. Note that these do not depend explicitly on the orbitals since they are evaluated by Wick's theorem which only depends on the anti-commutator relations (2.24). Thus, they only depend on the amplitudes.

5.4.3 OATDCC equations

Finally, equations of motion are derived by demanding $\delta\mathcal{S}[\langle\tilde{\Psi}|\,,|\Psi\rangle] = 0$ for all independent variations of $\langle\tilde{\Psi}|$ and $|\Psi\rangle$. In general both the single-particle functions and the amplitudes are allowed to vary resulting in equations of

motion for both the amplitudes and the single-particle functions. A detailed derivation can be found in the supplementary material to Ref. [1].

For the amplitudes the equations of motion read,

$$i\hbar\dot{\tau}^{\mu} = \frac{\partial}{\partial\lambda_{\mu}}\mathcal{E}_{H-i\hbar D_0}[\lambda,\tau,\tilde{\Phi},\Phi] = \langle\tilde{\phi}_{\mu}|e^{-T}\left(H - i\hbar D_0\right)e^{T}|\phi\rangle\,, \tag{5.46}$$

$$-i\hbar\dot{\lambda}_{\mu} = \frac{\partial}{\partial\tau^{\mu}}\mathcal{E}_{H-i\hbar D_0}[\lambda,\tau,\tilde{\Phi},\Phi] = \langle\tilde{\phi}|(1+\Lambda)e^{-T}[H - i\hbar D_0, X_{\mu}]e^{T}|\phi\rangle\,. \tag{5.47}$$

Here X_{μ} are the excitation operators,

$$\begin{aligned} X_i^a &= c_a^{\dagger}\tilde{c}_i, \\ X_{ij}^{ab} &= c_a^{\dagger}\tilde{c}_i c_b^{\dagger}\tilde{c}_j, \\ &\vdots \end{aligned}$$

Note that in the absence of D_0 the right-hand side of (5.46) is equivalent with the usual amplitude equations (5.21).

The equations for the unknowns $\eta_q^p(\tilde{\Phi},\dot{\Phi})$ are given by

$$i\hbar\sum_{bj}A_{aj}^{ib}\eta_b^j = \sum_p\rho_p^i h_a^j - \sum_q\rho_a^q h_b^i + \frac{1}{2}\left[\sum_{prs}\rho_{pr}^{is}u_{as}^{pr} - \sum_{rqs}\rho_{ar}^{qs}u_{qs}^{ir}\right] \tag{5.48}$$

$$-i\hbar\sum_{bj}A_{bi}^{ja}\eta_j^b = \sum_p\rho_p^a h_i^b - \sum_q\rho_i^q h_j^a + \frac{1}{2}\left[\sum_{prs}\rho_{pr}^{as}u_{is}^{pr} - \sum_{rqs}\rho_{ir}^{qs}u_{qs}^{ar}\right] + i\hbar\dot{\rho}_i^a, \tag{5.49}$$

where

$$A_{aj}^{ib} \equiv \langle\tilde{\Psi}|[c_j^{\dagger}\tilde{c}_b, c_a^{\dagger}\tilde{c}_i]|\Psi\rangle = \delta_a^b\rho_j^i - \delta_j^i\rho_a^b.$$

For the single-particle functions the equations of motion are given by

$$i\hbar\sum_q\rho_p^q Q\,|\dot{\varphi}_q\rangle = \sum_q\rho_p^q Qh\,|\varphi_q\rangle + \sum_{qrs}\rho_{pr}^{qs}QW_s^r\,|\varphi_q\rangle \tag{5.50}$$

$$-i\hbar\sum_p\rho_p^q\,\langle\dot{\tilde{\varphi}}_p|\,Q = \sum_p\rho_p^q\,\langle\tilde{\varphi}_p|\,hQ + \sum_{prs}\rho_{pr}^{qs}\,\langle\tilde{\varphi}_p|\,W_s^r Q, \tag{5.51}$$

where $Q = I - P$ with P being the projection operator

$$\Phi\tilde{\Phi} \equiv \sum_p|\varphi_p\rangle\,\langle\tilde{\varphi}_p|\,. \tag{5.52}$$

The operators W_s^r are defined by,

$$W_s^r \left|\psi\right\rangle \equiv \langle \cdot \tilde{\varphi}_r | u | \psi \varphi_s \rangle \equiv \int dx_1 \int dx_2 \tilde{\varphi}_r(x_2) u(x_1, x_2) \psi(x_1) \varphi_s(x_2) \quad (5.53)$$

Equations (5.46)-(5.51) are collectively referred to as the OATDCC equations.

It is not at all obvious how one should compute the initial state for the OATDCC equations when the single-particle functions are allowed to vary and Kvaal proposes several different solutions in his article.

5.4.4 The TDCC equations

A special case of OATDCC equations arises if the orbitals are held fixed in time. Then the operator D_0 drops out and the OATDCC equations reduce to only the amplitude equations

$$i\hbar \dot{\tau}^\mu = \frac{\partial}{\partial \lambda_\mu} \mathcal{E}_H[\lambda, \tau, \tilde{\Phi}, \Phi] = \langle \tilde{\phi}_\mu | e^{-T} H e^T | \phi \rangle \quad (5.54)$$

$$-i\hbar \dot{\lambda}_\mu = \frac{\partial}{\partial \tau^\mu} \mathcal{E}_H[\lambda, \tau, \tilde{\Phi}, \Phi] = \langle \tilde{\phi} | (1 + \Lambda) e^{-T} [H, X_\mu] e^T | \phi \rangle, \quad (5.55)$$

which we refer to as the TDCC-equations. Notice now that Eqs. (5.54) and (5.21) are equivalent. Furthermore, we take $\tilde{\Phi} = \Phi^*$.

The initial amplitudes $(\lambda^{(0)}, \tau^{(0)})$ are obtained by solving the non-linear equations

$$\langle \tilde{\phi}_\mu | e^{-T} H e^T | \phi \rangle = 0 \quad (5.56)$$

$$\langle \tilde{\phi} | (1 + \Lambda) e^{-T} [H, X_\mu] e^T | \phi \rangle = 0. \quad (5.57)$$

For the time evolution one then propagates the amplitudes by solving Eqs. (5.54) and (5.55). The energy is taken as the real part of

$$\mathcal{E}_H[\lambda(t), \tau(t)] = \langle \tilde{\phi} | (I + \Lambda) e^{-T} \hat{H} e^T | \phi \rangle \,, \quad (5.58)$$

where the imaginary part should be "small".

Approximations to the TDCC equations are obtained by truncating T and Λ. For example $T = T_1 + T_2$ and $\Lambda = \Lambda_1 + \Lambda_2$ corresponds to the TDCCSD approximation while $T = T_2$ and $\Lambda = \Lambda_2$ gives the TDCCD approximation.

It is shown in Ref. [1] that the singles amplitudes are redundant when the single-particle functions are varied. We will therefore restrict ourselves to the TDCCD approximation, using the Hartree-Fock state as reference determinant. This allows for flexibility if one wants to extend the program written in this work to the full OATDCC method.

5.4.5 The TDCCD approximation

In order to write a computer program that solves the TDCC equations we need algebraic expressions for the right-hand sides of Eqs. (5.54) and (5.55). Now, in the TDCCD approximation we take $T = T_2$ and $\Lambda = \Lambda_2$. We want expressions for the amplitudes and the one- and two-body density matrices. Thus we must evaluate the vacuum expectation values,

$$\frac{\partial}{\partial \lambda_{ab}^{ij}} \mathcal{E}_H = \langle \phi_{ij}^{ab} | e^{-T_2} H e^{T_2} | \phi \rangle \tag{5.59}$$

$$\frac{\partial}{\partial \tau_{ij}^{ab}} \mathcal{E}_H = \langle \phi | (1 + \Lambda_2) e^{-T_2} [H, X_{ij}^{ab}] e^{T_2} | \phi \rangle \,, \tag{5.60}$$

$$\rho_p^q = \langle \tilde{\Psi} | c_p^\dagger \tilde{c}_q | \Psi \rangle \,, \tag{5.61}$$

$$\rho_{pr}^{qs} = \langle \tilde{\Psi} | c_p^\dagger c_r^\dagger \tilde{c}_s \tilde{c}_q | \Psi \rangle \,. \tag{5.62}$$

These are generated by the *CoupledClusterSympy.py* script, which uses the second quantization package in SymPy, and is found in the src folder at the github page[2].

Recall that the Hamiltonian in second quantization is given by,

$$\hat{H} = \hat{H}^{(1)} + \hat{H}^{(2)} = \sum_{pq} h_q^p c_p^\dagger c_q + \frac{1}{4} \sum_{pqrs} u_{qs}^{pr} c_p^\dagger c_r^\dagger c_s c_q \tag{5.63}$$

where $u_{qs}^{pr} = \langle \varphi_p \varphi_r | \hat{H}^{(2)} | \varphi_q \varphi_s \rangle_{AS}$. Evaluation of the right hand side of equation (5.59) gives,

$$\frac{\partial}{\partial \lambda_{ij}^{ab}} E_{H^{(1)}} = h_i^k \tau_{jk}^{ab} P(ij) + h_c^a \tau_{ji}^{bc} P(ab), \tag{5.64}$$

$$\begin{aligned} \frac{\partial}{\partial \lambda_{ab}^{ij}} E_{H^{(2)}} = & - \tau_{ik}^{ab} u_{jl}^{kl} P(ij) + \tau_{il}^{ab} \chi_j^l P(ij) \\ & + \tau_{kl}^{ab} \chi_{ij}^{kl} + \tau_{ij}^{ac} \chi_c^b P(ab) \\ & + \tau_{ik}^{ac} \chi_{cj}^{kb} P(ab) P(ij) - \frac{1}{2} \tau_{ji}^{dc} u_{dc}^{ab} \\ & + u_{ij}^{ab}, \end{aligned} \tag{5.65}$$

where $P(ij)$ is an anti-symmetrizer in the sense that $f(ij)P(ij) = f(ij) - f(ji)$ and similarly for $P(ab)$. We have defined the so-called intermediates,

[2]https://github.com/haakoek/PythonVersionMaster/tree/master/src

$$\chi_{ij}^{kl} = \frac{1}{4}\tau_{ij}^{cd}u_{cd}^{kl} + \frac{1}{2}u_{ij}^{kl} \tag{5.66}$$

$$\chi_{c}^{b} = \frac{1}{2}\tau_{kl}^{bd}u_{dc}^{kl} + u_{ck}^{bk} \tag{5.67}$$

$$\chi_{j}^{l} = \frac{1}{2}\tau_{jk}^{dc}u_{dc}^{kl} \tag{5.68}$$

$$\chi_{cj}^{kb} = u_{cj}^{kb} + \frac{1}{2}u_{cd}^{kl}\tau_{jl}^{bd}. \tag{5.69}$$

We use the Einstein summation convention over repeated indices of opposite vertical placement. The appearance of a $P(ij)$ or a $P(ab)$ should be ignored for the invocation of the summation convention. This means that for example the expression $h_i^k \tau_{jk}^{ab} P(ij)$ is interpreted as,

$$h_i^k \tau_{jk}^{ab} P(ij) = \sum_k \left(h_i^k \tau_{jk}^{ab} - h_j^k \tau_{ik}^{ab} \right).$$

In other words, we sum over all indices that don't appear on the left-hand side of all following expressions.

Evaluation of the right hand side of Eq. (5.60) results in,

$$\frac{\partial}{\partial \tau_{ij}^{ab}} E_{H^{(1)}} = h_a^c \lambda_{bc}^{ji} P(ab) + h_k^i \lambda_{ab}^{jk} P(ij), \tag{5.70}$$

$$\begin{aligned}\frac{\partial}{\partial \tau_{ij}^{ab}} E_{H^{(2)}} =& \lambda_{ac}^{kl} \xi_{klb}^{cji} P(ab) + \lambda_{dc}^{ik} \xi_{kab}^{dcj} P(ij) \\ &+ \lambda_{ab}^{kl} \xi_{kl}^{ij} + \lambda_{ac}^{ik} \xi_{kb}^{cj} P(ab) P(ij) \\ &+ \lambda_{ab}^{ik} \xi_{k}^{j} P(ij) + \lambda_{dc}^{ji} \xi_{ab}^{dc} \\ &+ \lambda_{ac}^{ji} \xi_{b}^{c} P(ab) - u_{ab}^{ji},\end{aligned} \tag{5.71}$$

where we have defined the intermediates,

$$\xi^{cji}_{klb} = -\frac{1}{2}\tau^{dc}_{kl}u^{ji}_{bd} \tag{5.72}$$

$$\xi^{dcj}_{kab} = -\frac{1}{2}\tau^{dc}_{kl}u^{jl}_{ab} \tag{5.73}$$

$$\xi^{ij}_{kl} = -\frac{1}{2}u^{ji}_{kl} - \frac{1}{4}\tau^{dc}_{kl}u^{ji}_{dc} \tag{5.74}$$

$$\xi^{cj}_{kb} = -\tau^{dc}_{kl}u^{jl}_{bd} + u^{jc}_{bk} \tag{5.75}$$

$$\xi^{j}_{k} = -\frac{1}{2}\tau^{dc}_{kl}u^{jl}_{dc} - u^{jl}_{kl} \tag{5.76}$$

$$\xi^{dc}_{ab} = -\frac{1}{4}\tau^{dc}_{kl}u^{kl}_{ab} - \frac{1}{2}u^{dc}_{ab} \tag{5.77}$$

$$\xi^{c}_{b} = -\frac{1}{2}\tau^{dc}_{kl}u^{kl}_{bd} - u^{ck}_{bk}. \tag{5.78}$$

If we precompute and store all intermediates Eq. (5.65) scales as

$$\mathcal{O}\left(N^2(L-N)^4\right)$$

while Eq. (5.71) scales as

$$\mathcal{O}\left(N^3(L-N)^4\right).$$

For the onebody density matrix we get

$$\rho^{j}_{i} = \delta_{ij} - \frac{1}{2}\lambda^{kj}_{ab}\tau^{ab}_{ki}, \tag{5.79}$$

$$\rho^{b}_{a} = \frac{1}{2}\lambda^{ij}_{ac}\tau^{bc}_{ij}, \tag{5.80}$$

$$\rho^{a}_{i} = \rho^{i}_{a} = 0. \tag{5.81}$$

The non-zero elements of the two-body density matrix are given by,

$$\rho^{kl}_{ij} = -\delta_{il}\delta_{jk}P(ij) - \frac{1}{2}P(ij)P(kl)\delta_{jl}\lambda^{km}_{dc}\tau^{dc}_{im} + \frac{1}{2}\lambda^{lk}_{dc}\tau^{dc}_{ji}, \tag{5.82}$$

$$\rho^{ij}_{ab} = \lambda^{ij}_{ab} \tag{5.83}$$

$$\rho^{jb}_{ia} = \frac{1}{2}\delta_{ij}\lambda^{kl}_{ac}\tau^{bc}_{kl} - \lambda^{jk}_{ac}\tau^{bc}_{ik} = -\rho^{bj}_{ia} = -\rho^{jb}_{ai} = \rho^{bj}_{ai} \tag{5.84}$$

$$\rho^{cd}_{ab} = \frac{1}{2}\lambda^{ij}_{ab}\tau^{cd}_{ij} \tag{5.85}$$

$$\begin{aligned}\rho^{ab}_{ij} = &-\frac{1}{4}\lambda^{kl}_{dc}\tau^{ab}_{ik}\tau^{dc}_{jl}P(ab)P(ij) - \frac{1}{4}\lambda^{kl}_{dc}\tau^{ab}_{kl}\tau^{dc}_{ji} - \frac{1}{4}\lambda^{kl}_{dc}\tau^{ac}_{kl}\tau^{bd}_{ji}P(ij)\\ &+ \lambda^{kl}_{dc}\tau^{ac}_{il}\tau^{bd}_{jk}P(ij) - \frac{1}{4}\lambda^{kl}_{dc}\tau^{ac}_{ji}\tau^{bd}_{kl}P(ij) - \tau^{ab}_{ji}.\end{aligned} \tag{5.86}$$

Finally, the energy is taken as the real part of the expectation value (5.58) which in the CCD approximation is given by,

$$\begin{aligned}\mathcal{E}_H[\lambda(t),\tau(t)] &= \langle\tilde{\phi}|(I+\Lambda_2)e^{-T_2}\hat{H}e^{T_2}|\phi\rangle \\ &= \langle\tilde{\phi}|e^{-T_2}\hat{H}e^{T_2}|\phi\rangle + \langle\tilde{\phi}|\Lambda_2 e^{-T_2}\hat{H}e^{T_2}|\phi\rangle \\ &= \langle\tilde{\phi}|e^{-T_2}\hat{H}e^{T_2}|\phi\rangle + \frac{1}{4}\sum_{ijab}\lambda_{ij}^{ab}\,\langle\tilde{\phi}_{ij}^{ab}|e^{-T_2}\hat{H}e^{T_2}|\phi\rangle\end{aligned}$$

where

$$\langle\tilde{\phi}|e^{-T_2}\hat{H}e^{T_2}|\phi\rangle = h_i^i + \frac{1}{2}u_{ij}^{ij} + \frac{1}{4}\tau_{ij}^{ab}u_{ab}^{ij}. \tag{5.87}$$

Notice that in the groundstate

$$\langle\tilde{\phi}_{ij}^{ab}|e^{-T_2}\hat{H}e^{T_2}|\phi\rangle = 0,$$

which means that the energy for the ground state is simply given by (5.87).

Note also that the one- and two-body density matrices depend only on the amplitudes (τ,λ). This is an advantage when we want to verify the implementation of the λ-amplitudes since we can compare the density matrices with the density matrices computed with the CI method.

In order to verify that the τ-amplitudes have been computed correctly for the ground state we compute the ground state energy and compare with previous published results.

Chapter 6

Quantum Dots

As an application of the many-methods we have introduced we will consider models of quantum dots. Electrons confined in small areas in semiconductors, so-called quantum dots, form a hot research area in modern solid-state physics, with potential applications ranging from medical imaging to quantum computing. For example, the article by Loss and DiVincenzo [19] investigates quantum computing devices with quantum dots.

The choice of quantum dots as model systems is due to the fact that there is a vast literature body with results for both ground state and time-dependent studies, providing us with invaluable benchmarks for various implementations.

6.1 The One-Dimensional Quantum Dot

In the article [9] by Zanghellini *et al.*, a one-dimensional quantum dot model is studied. The Hamiltonian, in atomic units, for two interacting electrons moving in a confining potential is given by,

$$\hat{H}_0(x_1, x_2) = -\frac{1}{2}\left(\frac{d^2}{dx_1^2} + \frac{d^2}{dx_2^2}\right) + V_{\text{con}}(x_1) + V_{\text{con}}(x_2) + V_{\text{int}}(x_1, x_2). \quad (6.1)$$

If we define the single-particle operator,

$$\hat{h}(x) \equiv -\frac{1}{2}\frac{d^2}{dx^2} + V_{\text{con}}(x), \quad (6.2)$$

the Hamiltonian can be written on second quantized form relative to a (finite) single-particle basis $\{\phi_p(x)\}_{p=1}^{L}$ as,

$$\begin{aligned}\hat{H}_0 =& \sum_{pq} \langle \phi_p | \hat{h}(x) | \phi_q \rangle \, c_p^\dagger c_q \\ &+ \frac{1}{4} \sum_{pqrs} \langle \phi_p \phi_q | \hat{V}_{\text{int}}(x_1, x_2) | \phi_r \phi_s \rangle_{\text{AS}} \, c_p^\dagger c_q^\dagger c_s c_r. \end{aligned} \tag{6.3}$$

In the above paper by Zanghellini the confining potential is taken to be the usual harmonic oscillator potential,

$$V_{\text{con}}(x) = \frac{1}{2}\omega^2 x^2 \tag{6.4}$$

with oscillator frequency ω. In three dimensions the interaction between the electrons would be given by the Coulomb interaction. One-dimensional and two-dimensional models are effective models and the interaction between the electrons is modelled by a "shielded" Coulomb potential [20]

$$V_{\text{int}}(x_1, x_2) = \frac{\lambda}{\sqrt{(x_1 - x_2)^2 + a^2}}, \tag{6.5}$$

where λ is a dimensionless constant. The screening parameter a removes the singularity at $x_1 = x_2$, while retaining the asymptotic behavior at large distances. We will frequently refer to this model as the one-dimensional quantum dot.

For the one-dimensional quantum dot we will use spin-orbitals $\{\phi_p\}_{p=1}$ built from harmonic oscillator eigenfuntions, i.e

$$\left(-\frac{1}{2}\frac{d^2}{dx^2} + \frac{1}{2}\omega^2 x^2 \right) \psi_p(x) = \epsilon_p \psi_p(x) \tag{6.6}$$

as single-particle functions. The above equation has analytic solutions given by

$$\psi_p(x) = \left(\frac{\omega}{\pi}\right)^{1/4} \frac{1}{\sqrt{2^n n!}} H_n(\sqrt{\omega}x) e^{-\omega^2 x^2/2}, \tag{6.7}$$

$$\epsilon_p = \left(p + \frac{1}{2} \right) \omega, \tag{6.8}$$

where $p = 0, 1, 2, \cdots$. Here, $H_n(y)$ are the so-called Hermite polynomials given by,

$$H_n(y) = (-1)^n e^{y^2} \left(\frac{d}{dy} \right)^n e^{-y^2}. \tag{6.9}$$

The derivation can be found in any elementary introduction to quantum mechanics such as [12]. Thus the spin orbitals are given by

$$\phi_{(p,\sigma)}(x) = \psi_p(x)\chi_\sigma(s), \tag{6.10}$$

where χ_σ are the spinor basis functions and $(p,\sigma) = \{(1,\pm1),(2,\pm2),\cdots\}$.

Following Zanghellini we will, for time-dependent studies use the time-dependent perturbation,

$$\hat{H}_1(x_1,x_2,t) = (x_1+x_2)\mathcal{E}_0\sin(\Omega t), \tag{6.11}$$

which models a laser-field with frequency Ω and amplitude $\mathcal{E}_0$. Thus, we have the total time-dependent Hamiltonian,

$$\hat{H}(x_1,x_2,t) = \hat{H}_0(x_1,x_2) + \hat{H}_1(x_1,x_2,t). \tag{6.12}$$

The parameters used by Zanghellini are $\omega = 0.25$ for the oscillator frequency. The parameters for the shielded Coulomb potential are $a = 0.25$ and $\lambda = 1$ while $\mathcal{E}_0 = 1$ and $\Omega = 8\omega$ are used as parameters for the laser.

One can also consider other confining potentials such as the double well potential

$$V_{\text{con}}(x) = \frac{1}{2d^2}\left(x-\frac{1}{2}d\right)^2\left(x+\frac{1}{2}d\right)^2, \tag{6.13}$$

where d is the distance between the wells. In this case a more suitable single-particle basis is obtained by solving,

$$\left(-\frac{1}{2}\frac{d^2}{dx^2} + \frac{1}{2d^2}\left(x-\frac{1}{2}d\right)^2\left(x+\frac{1}{2}d\right)^2\right)\psi_p(x) = \epsilon_p\psi_p(x). \tag{6.14}$$

Note however that this equation cannot be solved analytically and we have to use a numerical method for ψ_p. Relevant parameters which can be used for a numerical experiment, given in for example Sigve Bø Skattum's master thesis [21], are $d = 8$ for the distance between the wells, while $\lambda = 0.6$ and $a = 0.1$ are employed for the shielded Coulomb potential. The range of the grid is taken as $x \in [-15, 15]$. For two electrons confined to the double well using six spin-orbitals Bø Skattum reports a ground state energy of $E_0 = 1.0467$.

6.2 The Two-Dimensional Quantum Dot

We will also study the two-dimensional quantum dot. The Hamiltonian for N electrons confined by a harmonic oscillator potential is given by,

$$\hat{H} = \sum_{i=1}^{N} \left(-\frac{\hbar^2}{2m_e}\nabla_i^2 + \frac{1}{2}m_e w^2 r_i^2 \right) + \sum_{i<j}^{N} \frac{ke^2}{r_{ij}} \tag{6.15}$$

$$= \hat{H}_0 + \hat{W}. \tag{6.16}$$

Here

$$\hat{H}_0 = \sum_{i=1}^{N} \hat{h}(\vec{r}_i), \qquad \hat{h}(\vec{r}_i) = -\frac{\hbar^2}{2m_e}\nabla_i^2 + \frac{1}{2}m_e w^2 r_i^2. \tag{6.17}$$

where $r \equiv |\vec{r}|$, $r_i \equiv |\vec{r}_i|$, and $r_{ij} \equiv |\vec{r}_i - \vec{r}_j|$. Hartree atomic untis is used ($e = m_e = \hbar = k_e = k_B = 1$).

Following Bø Skattum [21] we will for time dependent studies use the following perturbation

$$\hat{H}_1(t) = \mathcal{E}_0 \sin(\Omega t) \sum_{i=1}^{N} x_i \tag{6.18}$$

which models a laser polarized along the x-axis with amplitude $\mathcal{E}_0$ and frequency Ω. Thus, during the time evolution we get an addition in the one-body operator such that

$$\hat{h}(\vec{r}_i, t) = -\frac{\hbar^2}{2m_e}\nabla_i^2 + \frac{1}{2}m_e w^2 r_i^2 + \mathcal{E}_0 \sin(\Omega t) x_i. \tag{6.19}$$

The total Hamiltonian can now be written on a second quantized form relative to a single-particle basis $\{\phi_p\}$ as,

$$\hat{H}(t) = \sum_{pq} h_q^p(t) c_p^\dagger c_q + \frac{1}{4}\sum_{pqrs} w_{rs}^{pq} c_p^\dagger c_q^\dagger c_s c_r, \tag{6.20}$$

where

$$h_q^p(t) \equiv \langle \phi_p | \hat{h}(\vec{r}, t) | \phi_q \rangle \tag{6.21}$$

$$w_{rs}^{pq} \equiv \langle \phi_p \phi_q | \hat{w} | \phi_r \phi_s \rangle \,. \tag{6.22}$$

We will use spin-orbitals based on the harmonic oscillator eigenfunctions $\psi_{nm}(r, \theta)$ in polar coordinates as our single-particle basis functions,

$$\hat{h}(\vec{r})\psi_{nm}(r, \theta) = e_{nm}\psi_{nm}(r, \theta), \tag{6.23}$$

with eigenvalue e_{nm}

$$e_{nm} = \omega(2n + |m| + 1). \tag{6.24}$$

The eigenfunctions are given by,

$$\psi_{nm}(r,\theta) = ae^{im\theta}\sqrt{\frac{n!}{\pi(n+|m|)!}}(ar)^{|m|}L_n^{|m|}(a^2r^2)\exp(-a^2r^2/2), \tag{6.25}$$

where $a = \sqrt{\omega}$ and $L_n^{|m|}$ are the so-called Laguerre polynomials.

The quantum number $n = \{0, 1, ...\}$ is the number of nodes in the radial part, while $m = \{0, \pm1, \pm2, ...\}$ is the orbital angular momentum.

The spin-orbitals become

$$\phi_{\vec{p}}(r,\theta) = \psi_{nm}(r,\theta)\chi_\alpha(\sigma), \quad \vec{p} = (n, m, \alpha) \tag{6.26}$$

where $\alpha = \pm1/2$ is the quantum number for the z-projection of the electron spin, and χ_α is the corresponding spinor basis function satisfying $\langle\chi_\alpha|\chi_\beta\rangle = \delta_{\alpha\beta}$. The spin orbitals form so-called closed shells, where each shell contains orbitals with the same eigenvalue and all both spin values. The shell structure can be visualized using the Fock-Darwin representation as in figure 6.1. We will only look at systems with full shells. The number N of spin-orbitals needed to fill k shells are called magic numbers and of which the first few are $N = \{2, 6, 12, 20, 30, 42, 56, \cdots\}$.

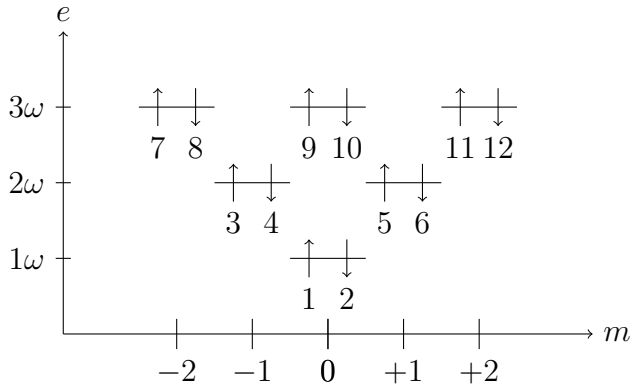


Figure 6.1: Spin-orbitals for an electron in a two-dimensional oscillator well using a so-called Fock-Darwin representation. Here we show the states and their oscillator energies for three harmonic-oscillator levels. The oscillator energies are given by $e_{nm} = \hbar\omega(2n + |m| + 1)$.

In order to study the two-dimensional quantum dot with the Hartree-Fock, the Configuration Interaction and the Coupled Cluster methods we

need to compute the one- and two-body integral elements $h_q^p(t)$ and w_{rs}^{pq} in the harmonic oscillator basis.

Writing $\vec{p} = (n_p, m_p, \alpha_p)$ and $\vec{q} = (n_q, m_q, \alpha_q)$ we have the following expression for the one-body integral,

$$\begin{aligned} h_q^p(t) &= \delta_{\alpha_p\alpha_q}\left(\langle\psi_{n_pm_p}|\hat{h}|\psi_{n_qm_q}\rangle + \mathcal{E}_0\sin(\Omega t)\langle\psi_{n_pm_p}|r\cos(\theta)|\psi_{n_qm_q}\rangle\right) \\ &= \delta_{\vec{p},\vec{q}}e_{n_pm_p} + \mathcal{E}_0\sin(\Omega t)\delta_{\alpha_p\alpha_q}\langle\psi_{n_pm_p}|r\cos(\theta)|\psi_{n_qm_q}\rangle\,. \end{aligned}$$

Notice the appearance of the term $r\cos(\theta)$ in the time dependent part since we work in polar coordinates. For ground state computations this expression reduces to,

$$h_q^p = \delta_{\vec{p},\vec{q}}e_{n_pm_p}. \tag{6.27}$$

The spatial part of the time-dependent perturbation

$$\langle\psi_{n_pm_p}|r\cos(\theta)|\psi_{n_qm_q}\rangle\,, \tag{6.28}$$

is solved analytically using the symbolic algebra package SymPy.

Furthermore the matrix elements of the Coulomb interaction $\hat{W}$ become,

$$\begin{aligned} w_{rs}^{pq} &= \langle\chi_{\alpha_p}|\chi_{\alpha_r}\rangle\langle\chi_{\alpha_q}|\chi_{\alpha_s}\rangle\langle\psi_{n_pm_p}\psi_{n_qm_q}|\hat{w}|\psi_{n_rm_r}\psi_{n_sm_s}\rangle \\ &= \delta_{\alpha_p\alpha_r}\delta_{\alpha_q\alpha_s}\langle\psi_{n_pm_p}\psi_{n_qm_q}|\hat{w}|\psi_{n_rm_r}\psi_{n_sm_s}\rangle\,, \end{aligned} \tag{6.29}$$

where,

$$\begin{aligned} &\langle\psi_{n_pm_p}\psi_{n_qm_q}|\hat{w}|\psi_{n_rm_r}\psi_{n_sm_s}\rangle = \\ &\iint r_1dr_1d\theta_1r_2dr_2d\theta_2\psi^*_{n_pm_p}(r_1,\theta_1)\psi^*_{n_qm_q}(r_2\theta_2)\frac{1}{r_{ij}}\psi_{n_rm_r}(r_1,\theta_1)\psi_{n_sm_s}(r_2,\theta_2). \end{aligned} \tag{6.30}$$

This integral can be solved analytically when using the harmonic oscillator eigenfunctions as basis with explicit expressions given in Ref. [22].

6.3 The Three-Dimensional Quantum Dot

The Hamiltonian for N electrons confined by a possibly deformed harmonic oscillator potential in three dimensions is given by,

$$\hat{H} = \sum_{i=1}^{N}\left(-\frac{\hbar^2}{2m_e}\nabla_i^2 + \frac{1}{2}m_ew^2(x^2+y^2+\alpha^2z^2)\right) + \sum_{i<j}^{N}\frac{ke^2}{r_{ij}} = \hat{H}_0 + \hat{W}, \tag{6.31}$$

where α controls the strength of the deformation in the z-direction. Vorrath [23] gives analytical expressions for both the one- and two-body integrals, using the eigenfunctions $\phi_{nml\sigma}(r, \theta, \varphi)$ of the thee-dimensional harmonic oscillator in spherical coordinates (see appendix A.3) as basis. We have done a preliminary implementation with $\alpha = 1$ where we compute the ground state energies of the three-dimensional quantum dot for $N = \{2, 8\}$ using the HF, CI and CC method for a small number of basis functions.

Chapter 7

Implementation and Results

In this chapter we give a detailed description of the implementation of the methods discussed in Chapters 4-6. Furthermore, we present and discuss in detail our results for different many-particle systems.

7.1 Overall structure of the Software Suite

7.1.1 Program flow

We want to simulate the time evolution of a quantum-mechanical system by solving the time-dependent Schrödinger equation (TDSE). Specifically, we consider the time evolution of one- and two-dimensional quantum dots. Additionally we compute ground state energies for three-dimensional quantum dots.

Before we can solve the TDSE we must define a system and an initial condition.

A system is characterized by a single-particle basis $\{\phi_i\}_{i=1}^{L}$, with L being the number of basis functions and its integral elements relative to the basis.

Depending on the system, the integral elements must either be computed numerically or they are given by analytical expressions. Thus, the first step is to set up the system and compute the integral elements.

The initial condition is taken as the ground state of the system which is found by solving the time-independent Schrödinger equation (TISE). The ground state is found by first performing a change of basis to a Hartree-Fock single-particle basis and then solving the TISE in this basis using either the Configuration Interaction method or the Coupled Cluster method.

Finally, we compute the time evolution by solving the TDSE using the ground state as initial condition. We note that since the one-body part of the

Hamiltonian in general is allowed to be time dependent we have to update the one-body integral elements during the time evolution.

7.1.2 Object-orientation: A description of the basic classes.

Using an object-oriented approach the program is split into the classes *System*, *HartreeFock*, *ConfigurationInteraction* and *CoupledCluster*. The philosophy behind the establishment of such classes is that we then can make adjustments and implement new features to separate parts without affecting the implementation of other parts of the program.

Specific systems such as the one-, two- and three-dimensional quantum dots are implemented as subclasses of the *System* class.

The Hartree-Fock, Configuration Interaction and Coupled Cluster programs are independent of the specific system in the sense that they take either a whole system or just integral elements as input. The advantage of such an approach is that the programs can be used to study a variety of systems, as long as we can provide the integral elements. One disadvantage is that it is hard to exploit properties of a specific system such as certain symmetries. If large scale simulations are desired one should probably write a program that is tailored more specifically for the system under consideration. We have however opted for a more general code, emphasing thus more overarching features of quantum-mechanical systems. Furthermore, we believe, strongly, that this approach results in a better pedagogical approach since focusing on generic aspects enhances the capability to make new abstractions.

The TDCI and TDCC methods are features of the *ConfigurationInteraction* and *CoupledCluster* respectively. A future project would be to expand these classes to handle the MCTDHF method [8] and the full OATDCC method as described in Ref. [1].

In the following sections we will discuss the details of the implementation of the classes described above.

7.2 The System class

In this section we present how to implement a system. In particular, we describe the implementation of the quantum dot systems described in chapter 6.

Common to all systems is that we want to be able to retrieve its integral elements relative to some single-particle basis. How we compute these

elements depends on the system under consideration. We introduce a superclass *System* which guarantees that it can return the integral elements. Explicit implementations of a specific system is then written as a subclass of the *System* class.

7.2.1 The One-Dimensional Quantum Dot

Consider the Hamiltonian for N electrons confined by some potential $V_{\text{con}}(x)$ in one spatial dimension relative to some single particle basis $\{\phi_p\}$,

$$\hat{H} = \sum_{pq} \langle \phi_p | \hat{h} | \phi_q \rangle \, c_p^\dagger c_q + \frac{1}{4} \sum_{pqrs} \langle \phi_p \phi_q | \hat{u} | \phi_r \psi_s \rangle_{\text{AS}} \, c_p^\dagger c_q^\dagger c_s c_r. \tag{7.1}$$

In one dimension the interaction operator $\hat{u}$ is taken to be the shielded Coulomb potential described in section 6.1. Before we can do ground state and time dependent computations we must define a single-particle basis. We must also compute the integral elements $\langle \phi_p | \hat{h} | \phi_q \rangle$ and $\langle \phi_p \phi_q | \hat{u} | \phi_r \psi_s \rangle_{\text{AS}}$. A typical choice for a single-particle basis are the eigenfunctions of the single-particle operator $\hat{h}$, which in one spatial dimension is given by

$$\hat{h} = -\frac{1}{2} \frac{d^2}{dx^2} + V_{\text{con}}(x). \tag{7.2}$$

That is, we can take our single-particle basis to be the solutions of

$$\hat{h} \phi_p(x) = \epsilon_p \phi_p(x). \tag{7.3}$$

When $V_{\text{con}}(x)$ is taken to be the harmonic oscillator potential this equation can be solved analytically. In general this is not possible, but especially when we work in one dimension it is quite straightforward to solve (7.3) numerically. When a single particle basis is found, we can compute the integral elements by numerical integration.

In the following we describe how to compute a single-particle basis for a general one-dimensional potential and how to obtain the integral elements..

Computing the single-particle functions

The most straightforward way to solve Eq. (7.3) numerically for a general confining potential is to use a uniform discretization,

$$x_{i+1} = x_i + i\Delta x, \quad i = 1, \cdots, N_{\text{grid}}, \tag{7.4}$$

where N_{grid} are the number of grid points and $x_0 = a$ and $x_{N_{\text{grid}}+1} = b$ for $x \in [a, b]$.

Let $u(x_i) = u_i$ denote the approximation of $\phi_p(x)$ at the point x_i. Using the standard central difference approximation of the second derivative we have,

$$\left(-\frac{1}{2}\frac{u_{i-1} - 2u_i + u_{i+1}}{\Delta x^2} + V_{\text{con}}(x_i)u_i\right) = \epsilon_p u_i, \tag{7.5}$$

which can be written on matrix form,

$$A\mathbf{u}_p = \epsilon_p \mathbf{u}_p, \tag{7.6}$$

where A is a tridiagonal matrix with elements,

$$A_{i,i} = -\frac{1}{2\Delta x^2} + V_{\text{con}}(x_i), \quad i = 1, \cdots, N_{\text{grid}}$$
$$A_{i,i+1} = A_{i+1,i} = -\frac{1}{2\Delta x^2}, \quad i = 1, \cdots, N_{\text{grid}} - 1,$$

and $\mathbf{u} = [\phi_p(x_1), \cdots, \phi_p(x_N)]^T$. Diagonalization of A will then give an approximation to the N first eigenfunctions of $\hat{h}$. One should note that the quality of the eigenfunctions depends on the number of grid points N_{grid} and care should be taken with how many is useful.

Computing integral elements

We now assume that a single-particle basis $\{\phi_i\}_{i=1}^{L}$ has been defined. We want to compute the integral elements

$$h_q^p = \langle \phi_p | \hat{h} | \phi_q \rangle = \int \phi_p^*(x) \hat{h} \phi_q(x) dx, \tag{7.7}$$

$$u_{rs}^{pq} = \langle \phi_p \phi_q | \hat{u} | \phi_r \phi_s \rangle = \iint \phi_p^*(x_1)\phi_q^*(x_2)\hat{u}(x_1, x_2)\phi_r(x_1)\phi_s(x_2)dx_1 dx_2. \tag{7.8}$$

The one-body operator is given by Eq. (7.2) and the two-body operator is taken to be the shielded Coulomb potential

$$\hat{u}(x_1, x_2) = \frac{\lambda}{\sqrt{(x_1 - x_2)^2 + a^2}}. \tag{7.9}$$

We assume that the single-particle functions, $\phi_i(x)$, are the eigenfunctions of the one-body operator. In this case the one-body integral elements are given by

$$h_q^p = \epsilon_p \delta_{pq}. \tag{7.10}$$

In order to compute the two-body integral elements we first rewrite Eq. (7.8) as

$$\begin{aligned} u_{rs}^{pq} &= \int \phi_p^*(x_1) \left[\int \phi_q^*(x_2) \hat{u}(x_1, x_2) \phi_s(x_2) dx_2 \right] \phi_r(x_1) dx_1 \\ &= \int \phi_p^*(x_1) W_s^q(x_1) \phi_r(x_1) dx_1, \end{aligned} \tag{7.11}$$

where we have defined the function

$$W_s^q(x_1) \equiv \int \phi_q^*(x_2) \hat{u}(x_1, x_2) \phi_s(x_2) dx_2. \tag{7.12}$$

The functions W_s^q can be computed numerically by the trapezoidal rule. In theory the integration limits are $\pm\infty$ but in practice we must choose some cut-off L_x such that $x_1, x_2 \in [-L_x, L_x]$. Choose a stepsize $\Delta x = 2L_x/N_{\text{grid}}$ where N_{grid} is the number of integration points and uniform gridpoints such,

$$x_i = -L_x + i\Delta x, \quad i = 0, \cdots, N_{\text{grid}}.$$

Applying the trapezoidal rule we get that $W_s^q(x_{1,i})$ are given by,

$$W_s^q(x_{1,i}) = \frac{\Delta x}{2} \left[f\left(x_{1,i}, x_{2,0}\right) + 2 \sum_{j=1}^{N_{\text{grid}}-1} f\left(x_{1,i}, x_{2,j}\right) + f\left(x_{1,i}, x_{2,N_{\text{grid}}}\right) \right],$$

where

$$f(x_{1,i}, x_{2,j}) = \phi_q^*(x_{2,j}) \hat{u}(x_{1,i}, x_{2,j}) \phi_s(x_{2,j}).$$

The integral elements u_{rs}^{pq} can be computed similarly by the trapezoidal rule as follows

$$u_{rs}^{pq} = \frac{\Delta x}{2} \left[g(x_{1,0}) + 2 \sum_{i=1}^{N_{\text{grid}}-1} g(x_{1,i}) + g(x_{1,N_{\text{grid}}}) \right], \tag{7.13}$$

where

$$g(x_{1,i}) = \phi_p^*(x_{1,i}) W_s^q(x_{1,i}) \phi_r(x_{1,i}).$$

Additionally, when we study the time evolution of the one-dimensional quantum dot we have the time-dependent perturbation,

$$\hat{H}_1(t) = \mathcal{E}_0 \sin(\Omega t) \sum_{pq} \langle \phi_p | x | \phi_q \rangle \, c_p^\dagger c_q \tag{7.14}$$

and we have to compute the integrals $\langle \phi_p | x | \phi_q \rangle$. This is done analogously by the trapezoidal rule.

7.2.2 Two- and Three-Dimensional Quantum Dots

For the two- and three-dimensional quantum dots we choose a less flexible approach. In particular, the computation of the two-body integrals become much more expensive in two or three dimensions.

In two dimensions the single-particle basis, $\{\phi_p\}$, is taken as spin-orbitals built from the eigenfunctions of the harmonic oscillator in polar coordinates as described in chapter 6.2,

$$\hat{h}\phi_p(r,\theta) = \epsilon_{nm}\phi_p(r,\theta), \quad p = (n, m, \sigma). \tag{7.15}$$

Then the one body integral elements are given by,

$$h_q^p = \delta_{pq}\epsilon_p. \tag{7.16}$$

Using this representation the two body integrals are given by,

$$u_{rs}^{pq} = \delta_{\alpha_p\alpha_r}\delta_{\alpha_q\alpha_s}\left\langle \psi_{n_pm_p}\psi_{n_qm_q}|\hat{w}|\psi_{n_rm_r}\psi_{n_sm_s}\right\rangle, \tag{7.17}$$

where the integrals $\langle \psi_{n_pm_p}\psi_{n_qm_q}|\hat{w}|\psi_{n_rm_r}\psi_{n_sm_s}\rangle$ are given analytically in Ref. [22].

Similarly for the three-dimensional quantum dot we use spin-orbitals built from harmonic oscillator functions in three dimensions using spherical coordinates (see appendix A.3). The integral elements are given analytically in Ref. [23].

A C++ program that evalutates the integral elements according to Refs. [22, 23] can be found in the source code associated with this work[1].

Other confining potentials.

It would be interesting to study electrons confined by some other potential than the harmonic oscillator (HO) potential. The challenge is then to have a suitable basis of single-particle functions.

For the HO potential we have analytical expressions for the eigenfunctions, $\phi_p(r)$, of the single particle operator,

$$\hat{h}_{\text{HO}} \equiv -\frac{1}{2}\nabla^2 + V_{\text{HO}}(r), \tag{7.18}$$

$$r = \sum_{i=1}^{d} x_i^2, \quad \nabla^2 = \sum_{i=1}^{d}\frac{\partial^2}{\partial x_i^2}, \quad d = \{1, 2, 3\}, \tag{7.19}$$

[1] https://github.com/haakoek/PythonVersionMaster/tree/master/src

which we use as single-particle functions. In particular, we have analytical expressions for the two-body integral elements

$$\langle\phi_p\phi_q|\frac{1}{r_{ij}}|\phi_q\phi_s\rangle$$

in two and three dimensions.

For other confining potentials of interest we can in general not solve for the eigenfunctions analytically and we also have to compute the two-body integrals. In one dimension we could just solve for the eigenfunctions by numerical diagonalization. However, in two and three dimensions the number of grid points required to get reasonable approximations with such a straightforward approach grows rapidly. If we use a quadratic grid in two dimensions with 200 grid points in the x and y-direction we would need a total of 40.000 grid points to diagonalize $\hat{h}$!

One possible approach is write the unknown eigenfunctions in terms of the HO eigenfunctions and solve for the coefficients instead. Suppose that we seek the solutions of the eigenvalue problem

$$\left(-\frac{1}{2}\nabla^2 + V_{\text{con}}(r)\right)\psi_p(r) = \epsilon_p\psi_p(r), \tag{7.20}$$

where $V_{\text{con}}(r)$ is some confining potential other than the HO potential. Since the eigenfunctions $\phi_q(r)$ of $\hat{h}_{\text{HO}}$ form a basis we can expand the unknown functions $\psi_p(r)$ in this basis

$$\psi_p(r) = \sum_q A_{qp}\phi_q(r), \quad A_{pq} \in \mathbb{C}. \tag{7.21}$$

Inserting this expansion into (7.20) and multiplying with $\phi_r^*(r)$ we obtain

$$\sum_q A_{qp}\phi_r^*(r)\left(-\frac{1}{2}\nabla^2 + V_{\text{con}}(r)\right)\phi_q(r) = \epsilon_p\sum_q A_{qp}\phi_r^*(r)\phi_q(r). \tag{7.22}$$

Integrating over the whole space and using the orthonormality of the HO-eigenfuntions we recognize this as the matrix eigenvalue problem

$$hA = A\epsilon, \quad \epsilon \equiv \text{diag}(\epsilon_0, \epsilon_1, \cdots, \epsilon_L). \tag{7.23}$$

where

$$h_{rq} = -\frac{1}{2}\langle\phi_r|\nabla^2|\phi_p\rangle + \langle\phi_r|V_{\text{con}}(r)|\phi_p\rangle \tag{7.24}$$

and $A \equiv [A_{qp}]$. Strictly speaking, this is just the CI method for a single particle!

Since we know the harmonic oscillator eigenfunctions analytically the first term in the above equation can be evaluated exactly, while the second term must, in general, be computed numerically. Diagonalization of h will give the expansion coefficients A and the eigenvalues ϵ_p. When A is known, we can compute the two-body integrals $\langle\psi_p\psi_q|1/r_{ij}|\psi_r\psi_s\rangle$ in terms of the known HO two-body integrals,

$$\langle\psi_p\psi_q|\frac{1}{r_{ij}}|\psi_r\psi_s\rangle = \sum_{\alpha\beta\gamma\delta} A^*_{\alpha p}A^*_{\beta q}A_{\gamma r}A_{\delta s}\,\langle\phi_p\phi_q|\frac{1}{r_{ij}}|\phi_r\phi_s\rangle \tag{7.25}$$

where

$$\langle\phi_p\phi_q|\frac{1}{r_{ij}}|\phi_r\phi_s\rangle \tag{7.26}$$

are the integrals given by Refs. [22,23] in two or three dimensions respectively.

7.3 Hartree-Fock theory

The implementation of the Hartree-Fock method is quite straightforward. We want to iteratively solve the non-linear Roothan-Hall equation (3.16),

$$F(U)U = U\epsilon,$$

where F is the Fock matrix, U is the matrix of expansion coefficients and ϵ is a diagonal matrix. Specifically we need a procedure for diagonalizing $F(U)$. Since F is Hermitian (symmetric when F is real) this can be done efficiently using the *eigh*[2] function in numpy which exploits the Hermiticity of F.

7.3.1 Solving the Roothan-Hall equations by SCF iterations.

One possible solution scheme is by so-called Self-Consistent Field (SCF) iterations [24]. Let $U^{(k)}$ be the matrix of expansion coefficients in iteration k. Using an initial guess $U^{(0)}$ with $(U^{(0)})^TU^{(0)} = I$, the SCF-iteration computes $U^{(k+1)}$ by solving,

$$F(U^{(k)})U^{(k+1)} = U^{(k+1)}\epsilon^{(k+1)}. \tag{7.27}$$

The process is repeated until some predefined convergence criteria is met, for example

$$\max_{1\leq i\leq L} |\epsilon^{(k+1)}_{ii} - \epsilon^{(k)}_{ii}| < \delta, \tag{7.28}$$

[2]https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.linalg.eigh.html

with δ some small number. When this criteria is met we terminate the iterations and the system is said to have reached self-consistentency.

According to Liu *et al.* [24] it is well known that the SCF-iterations converge slowly or fails to converge. However, it turns out that it is sufficient for our needs as we demonstrate in section 7.3.4. In order to speed up convergence the so-called Direct Inversion of the Iterative Subspace (DIIS), suggested in Ref. [14] could be used.

7.3.2 Implementation details

Let N denote the number of particles and L the number of single-particle functions. Furthermore, assume that we are given $h = [h_p^q]$ and $u = [u_{rs}^{pq}]$ relative to some basis $\{\phi_i\}_{i=0}^{L}$ with u anti-symmetrized. In order to write a working Hartree-Fock program we have to compute

$$D_{sr} = \sum_i U_{sj} U_{rj}^*, \tag{7.29}$$

$$F_{qp} = h_p^q + \sum_{rs} D_{sr} u_{ps}^{qr}, \tag{7.30}$$

$$E_{\text{HF}} = \sum_{pqi} U_{qi}^* h_p^q U_{pi} + \sum_{pqi} U_{qi}^* \left(\sum_{rs} D_{sr} u_{ps}^{qr} \right) U_{pi}, \tag{7.31}$$

and diagonalize F for every SCF-iteration in order to obtain U and ϵ. Note that the indices $i, j = 0, \cdots, N$ while $p, q, r, s = 0, \cdots, L$. If the method converges, we can compute the matrix elements of h_{HF} and u_{HF} in the Hartree-Fock basis as,

$$h_{q,\text{HF}}^p = \langle \psi_p | \hat{h} | \psi_q \rangle = \sum_{\alpha\beta}^{L} U_{p\alpha}^* U_{q\beta} \langle \phi_\alpha | \hat{h} | \phi_\beta \rangle \tag{7.32}$$

$$u_{rs,\text{HF}}^{pq} = \langle \psi_p \psi_q | \hat{u} | \phi_r \phi_s \rangle = \sum_{\alpha\beta\gamma\delta}^{L} U_{p\alpha}^* U_{q\alpha}^* U_{r\beta} U_{s\beta} \langle \phi_\alpha \phi_\beta | \hat{u} | \phi_\gamma \phi_\delta \rangle , \tag{7.33}$$

where

$$\psi_p = \sum_{\alpha}^{L} U_{p\alpha} | \phi_\alpha \rangle . \tag{7.34}$$

Listing (7.1) demonstrates how these expressions can be evaluated using numpy's *einsum*[3] function. This function allows for use of the Einstein summation convention by specifying the summation indices, the arrays which we

[3]https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.einsum.html

sum over and the dimension of the resulting array. This gives a very compact code and makes it a lot easier to debug. Furthermore, it is extremely useful when we implement the Coupled Cluster equations.

Listing 7.1: Einsum example

```
occ = 0:N #Indices of occupied SPF's.
D = einsum('sj,rj->sr',U, U).
F = h + einsum('sr,qrps->qp',D,u)
EHF = sum(eps[self.o])
      -einsum('qi,sr,qrps,pi->',U[:,occ],D,u,U[:,occ])
hHF = einsum('sp,rq,sr->pq',U,U,h)
uHF = einsum('ap,bq,cr,ds,abcd->pqrs',U,U,U,U,u)
```

7.3.3 Limitations of the implementation

The most computationally expensive step in the Hartree-Fock procedure is the formation of the Fock matrix, F, which requires $\mathcal{O}(L^4)$ operations since we sum $p, q, r, s = 0, \cdots, L$. The diagonalization of F is performed by a highly efficient library function and is on the order of $\mathcal{O}(L^3)$ operations.

Another limitation is that of memory. In particular, the storage of the two-body integral elements u_{rs}^{pq} is prohibitive for a large number of basis functions L. If each element is stored with double-precision it takes 8 bytes of memory. Thus, if we want to use $L = 180$ basis functions we need $180^4 \cdot 8$ bytes ≈ 8.4GB of memory. Most personal computers in 2017 have $4-8$GB of RAM, meaning that such a calculation would be in the limit of what is possible on a standard machine. However, u_{rs}^{pq} is typically sparse which can be exploited by considering certain symmetries of the specific system under consideration, see for example Ref. [25]. Storing only the non-zero elements of u_{rs}^{pq} will reduce the memory requirements drastically.

In this work we don't consider any optimization with respect to memory since we only use a modest number of basis functions. A complete Hartree-Fock program is listed in the appendix C.1.

7.3.4 Verification

In order to verify the Hartree-Fock program we compute the energy and compare with results in the article by Zanghellini *et al.* [9] for the one-dimensional quantum dot. For the two-dimensional quantum dot we compare the computed Hartree-Fock energies with those reported by Lohne, Hagen, Hjorth-Jensen, Kvaal and Pederiva [26].

For the one-dimensional quantum dot we use the parameters reported by Zanghellini *et al.* [9]. The oscillator frequency is set to $\omega = 0.25$. We use a uniform grid $[-L_x, L_x]$ with $L_x = 10$ and gridspacing $\Delta x = 0.1$. The convergence threshold for the SCF-iterations (7.28) is set to $\delta = 10^{-8}$. The two-body integrals are computed as described in chapter 7.2.1. For the shielded Coulomb potential,

$$V_c(x_1, x_2) = \frac{1}{\sqrt{(x_1 - x_2)^2 + a^2}}$$

the shielding parameter is set to $a = 0.25$.

For $N = 2$ Zanghellini *et al.* [9] obtain $E_{\text{HF}} = 1.1795$. In table 7.1 we have computed the energy as a function of spatial basis functions L and it is clear that it converges to the result reported by Zanghellini *et al.*

Table 7.1: E_{HF} computed as a function of the number of (spatial) basis functions L for the one-dimensional quantum dot with $N = 2$ and $\omega = 0.25$ using SCF-iterations (7.27). M_δ denotes the number of iterations required to reach convergence in the sense of (7.28). We have used $\delta = 10^{-8}$. The results converge to the Hartree-Fock energy reported in Ref. [9].

L	E_{HF}	M_δ
4	1.1910313	22
6	1.1795909	29
8	1.1795871	29
10	1.1795713	29
12	1.1795690	29
14	1.1795690	29

For the two-dimensional quantum dot we compare the Hartree-Fock energy with the energies reported in Ref. [26] for $N = \{2, 6\}$ and $\omega = \{0.5, 1\}$. The results summarized in table 7.2 agrees with the energies reported by Lohne *et al.* [26]. For $N = 6$ and $L = 21$ the measured cpu time of the program was 0.423 seconds averaged over 100 runs.

7.4 Configuration Interaction

We want to solve the time-independent Schrödinger equation

$$\hat{H} |\Psi_k\rangle = E_k |\Psi_k\rangle \quad k = 0, 1, 2, \cdots, \tag{7.35}$$

Table 7.2: E_{HF} computed as a function of the number of (spatial) basis functions L for the two-dimensional quantum dot for $N = \{2, 6\}$ and $w = \{0.5, 1.0\}$ using SCF-iterations. M_δ denotes the number of iterations required to reach convergence in the sense of (7.28). We have used $\delta = 10^{-8}$. The results are in agreement with the energies reported by Lohne et al.

		$N = 2$		$N = 6$	
ω	L	E_{HF}	M_δ	E_{HF}	M_δ
	6	1.799856	8	13.051619	10
	10	1.799856	8	12.357471	14
0.5	15	1.799748	10	12.325128	16
	21	1.799748	10	12.271499	18
	6	3.162691	7	21.593198	8
	10	3.162691	7	20.766919	14
1.0	15	3.161921	9	20.748402	15
	21	3.161921	9	20.720257	15

using the Configuration Interaction (CI) method. Recall that the CI ansatz was given by,

$$|\Psi_k\rangle = \sum_I A_{Ik} |\Phi_I\rangle \tag{7.36}$$

where $\{|\Phi_I\rangle\}$ is a given Slater determinant basis built from a set of single-particle functions $\{\phi_i\}$. Equation (7.35) could then be written as the matrix eigenvalue problem

$$HA = AE, \tag{7.37}$$

where

$$H_{IJ} = \langle \Phi_I | \hat{H} | \Phi_J \rangle \tag{7.38}$$

and the columns of the matrix A contain the expansion coefficients of $|\Psi_k\rangle$. The symbol E represents a diagonal matrix with the eigenvalues E_k's on its diagonal. If we can compute the matrix H we can solve equation (7.37) by diagonalization of H.

In this chapter we present how to implement the computation of H by using the so-called occupation number representation of Slater determinants. Furthermore, it is shown how the computation of the matrix elements is simplified by the use of the Slater-Condon rules from Eqs. (2.51)-(2.53).

We also show how to extend the implementation to compute the time evolution of a state using the TDCI method.

Finally, the implementation is verified by comparing results with previous studies.

7.4.1 Representation of Slater determinants

Using second quantization the matrix elements H_{IJ} are given by,

$$\begin{aligned} H_{IJ} &= \langle \Phi_I | \hat{H} | \Phi_J \rangle \\ &= \sum_{pq} h_q^p \langle \Phi_I | c_p^\dagger c_q | \Phi_J \rangle + \frac{1}{4} \sum_{pqrs} u_{rs}^{pq} \langle \Phi_I | c_p^\dagger c_q^\dagger c_s c_r | \Phi_J \rangle , \end{aligned}$$

where

$$\begin{aligned} h_q^p &= \langle \phi_p | \hat{h} | \phi_q \rangle , \\ u_{rs}^{pq} &= \langle \phi_p \phi_q | \hat{u} | \phi_r \phi_s \rangle_{\mathrm{AS}} . \end{aligned}$$

Assuming that h_q^p and u_{rs}^{pq} are available in computer memory, if we can evaluate $\langle \Phi_I | c_p^\dagger c_q | \Phi_J \rangle$ and $\langle \Phi_I | c_p^\dagger c_q^\dagger c_s c_r | \Phi_J \rangle$ we can in priniciple compute the matrix elements H_{IJ}.

One way to achieve this is by using the so-called occupation number representation of a Slater determinants and apply the rules of the creation and annihilation operators. Assume now that we have a finite number of Slater determinants $\{|\Phi_I\rangle\}_{I=0}^{N_{\mathrm{sd}}}$ built from the single-particle functions $\{\phi_i\}_{i=0}^{L}$. Then we can represent an N-particle determinant by an array with length L containing 0's and 1's. Index k is set to 1 if the determinant contains the SPF ϕ_k and 0 otherwise. As an example, let $N = 4$ and $L = 8$, and consider the determinant $|\phi_0 \phi_2 \phi_3 \phi_6\rangle$ which can be represented as,

$$|\phi_0 \phi_2 \phi_3 \phi_6\rangle \rightarrow [1, 0, 1, 1, 0, 0, 1, 0]. \tag{7.39}$$

In general we can represent an N-particle determinant $|\Phi\rangle = |\phi_{n_{i_1}} \cdots \phi_{n_{i_N}}\rangle$ where $n_{i_k} \in \{0, 1, \cdots, L\}$ as

$$|\Phi\rangle = |\phi_{n_{i_1}} \cdots \phi_{n_{i_N}}\rangle \rightarrow [n_0, n_1, n_2, \cdots, n_L] \tag{7.40}$$

where $n_k = 1$ if $\phi_{n_k} \in |\Phi\rangle$ and 0 otherwise.

The action of a creation and an annihilation operator on an arbitrary determinant in this representation can be evaluated by Algorithm 1. The evaluation of $\langle \Phi_I | c_p^\dagger c_q | \Phi_J \rangle$ and $\langle \Phi_I | c_p^\dagger c_q^\dagger c_s c_r | \Phi_J \rangle$ is then realized by repeated use of Algorithm1. If none of the operations produces the zero result, one obtains

$$\begin{aligned} c_p^\dagger c_q \, |\Phi_J\rangle &= (-1)^{\alpha_p + \alpha_q} \, |\Phi_J'\rangle , \\ c_p^\dagger c_q^\dagger c_s c_r \, |\Phi_J\rangle &= (-1)^{\alpha_p + \alpha_q + \alpha_s + \alpha_r} \, |\Phi_J''\rangle , \end{aligned}$$

Algorithm 1 Algorithm for evaluating the action of the creation or annihilation operator on a Slater determinant in the occupation number representation.

1.) Creation operator:

$$c_p^\dagger |\Phi\rangle = c_p^\dagger [n_0, n_1, n_2, \cdots, n_L] = (-1)^\alpha |\Phi'\rangle$$

- If $n_k = 1$ return the zero result.
- If $n_k = 0$ compute α as the number of 1's before n_k and set $n_k = 1$ in order to obtain $|\Phi'\rangle$.
- Return the sign $(-1)^\alpha$ and $|\Phi'\rangle$

2.) Annihilation operator:

$$c_p |\Phi\rangle = c_p [n_0, n_1, n_2, \cdots, n_L] = (-1)^\alpha |\Phi'\rangle$$

- If $n_k = 0$ return the zero result.
- If $n_k = 1$ compute α as the number of 1's before n_k and set $n_k = 0$ to obtain $|\Phi'\rangle$
- Return the sign $(-1)^\alpha$ and $|\Phi'\rangle$

and checks whether $\langle\Phi_I|\,,|\Phi'_J\rangle$ and $\langle\Phi_I|\,,|\Phi''_J\rangle$ contains the same single-particle functions. If they do contain the same single particle functions we have that

$$\langle\Phi_I|c_p^\dagger c_q|\Phi_J\rangle = (-1)^{\alpha_p+\alpha_q}$$
$$\langle\Phi_I|c_p^\dagger c_q^\dagger c_s c_r|\Phi_J\rangle = (-1)^{\alpha_p+\alpha_q+\alpha_s+\alpha_r},$$

otherwise the zero result is returned.

These algorithms are implemented in the *CreationAnnihilation* module listed in the appendix C.4.

7.4.2 Slater-Condon rules applied to the CI Hamiltonian

If we denote a Slater determinant $|\Phi_I\rangle = |\phi_{i_1},\cdots,\phi_{i_N}\rangle = |i_1,\cdots,i_N\rangle$ and apply the Slater-Condon rules to the evalutation of H_{IJ} we are left with the following non-zero matrix elements,

1.) For identical determinants we have

$$H_{II} = \sum_{i\in|\Phi_I\rangle} h_i^i + \frac{1}{2}\sum_{i\in|\Phi_I\rangle}\sum_{j\in|\Phi_I\rangle} u_{ij}^{ij} \tag{7.41}$$

2.) For determinants that differ by one single-particle function, i.e

$$|\Phi_I\rangle = |\cdots mn\cdots\rangle$$
$$|\Phi_J\rangle = |\cdots pn\cdots\rangle\,,$$

where $p \neq m$, we have

$$H_{IJ} = s\left(h_p^m + \sum_{q\in|\Phi_I\rangle} u_{pq}^{mq}\right). \tag{7.42}$$

Here $s = (-1)^{k_m+k_p}$ where k_p is the number of single-particle functions set before p while k_m is the number of single-particle functions set before m. The sign results from the fact that $\langle\Phi_I|c_m^\dagger c_q^\dagger c_q c_p|\Phi_J\rangle = (-1)^{k_m+k_p+2k_q} = (-1)^{k_m+k_p}$.

3. For determinants that differ by two single-particle functions,

$$|\Phi_I\rangle = |\cdots mn\cdots\rangle$$
$$|\Phi_J\rangle = |\cdots pq\cdots\rangle\,,$$

where $m \neq p$ and $n \neq q$, we have

$$H_{IJ} = s u_{pq}^{mn}. \tag{7.43}$$

Here $s = (-1)^{k_m+k_n+k_p+k_q}$ since we must account for $\langle \Phi_I | c_m^\dagger c_n^\dagger c_p c_q | \Phi_J \rangle = (-1)^{k_m+k_n+k_p+k_q}$.

This reduces the computational effort drastically since we do not have to perform the expensive sum, which is an $\mathcal{O}(L^4)$ operation,

$$\sum_{pqrs}^{L} u_{rs}^{pq} \langle \Phi_I | c_p^\dagger c_q^\dagger c_s c_r | \Phi_J \rangle \tag{7.44}$$

when computing H_{IJ}.

In order to take advantage of the Slater-Condon rules we have the following algorithm:

1.) Construct a CI space (CIS,CID, CISD,...) consisting of a total of N_{sd} linearly independent Slater determinants

$$\{ |\Phi_I\rangle \, | I = 0, \cdots, N_{\text{sd}} - 1 \}$$

represented in occupation number representation.

2.) Loop over all distinct pairs

$$\{ |\Phi_I\rangle, |\Phi_J\rangle \, | I = 0, \cdots, N_{\text{sd}} - 1, J = I + 1, \cdots, N_{\text{sd}} - 1 \}$$

of determinants:

- If the pair differs by one single-particle function,

$$|\Phi_I\rangle = |\cdots mn \cdots\rangle$$
$$|\Phi_J\rangle = |\cdots pn \cdots\rangle,$$

compute the sign

$$s = \langle \Phi_I | c_m^\dagger c_p | \Phi_J \rangle = (-1)^{k_m+k_p}$$

and store (I, J, m, p, s) in a table *diffByOne*.

- If the pair differs by two single-particle functions,

$$|\Phi_I\rangle = |\cdots mn \cdots\rangle$$
$$|\Phi_J\rangle = |\cdots pq \cdots\rangle,$$

compute the sign

$$s = \langle \Phi_I | c_m^\dagger c_n^\dagger c_p c_q | \Phi_J \rangle = (-1)^{k_m + k_n + k_p + k_q}$$

and store (I, J, m, n, p, q, s) in a table *diffByTwo*.

- Else: Continue

3.) Compute H_{IJ} according to the Slater-Condon rules as follows:

- Loop over $I = 0, \cdots, N_{\text{sd}} - 1$ and compute H_{II} according to (7.41).
- Loop over all elements in the table *diffByOne* and update element H_{IJ} according to (7.42).
- Loop over all elements in the table *diffByTwo* and update element H_{IJ} according to (7.43).

4.) Since H is Hermitian set $H_{JI} = H_{IJ}^*$ since we only looped over distinct pairs in step 2.).

The algorithm is implemented in the function *computeHamiltonian* in the class *ConfigurationInteraction* listed in appendix C.2.

7.4.3 Computing the CI ground state.

Let us return to the TISE method written now on matrix form,

$$HA = AE. \tag{7.45}$$

Consider a model space defined by $\{|\Phi_I\rangle \, | I = 0, \cdots, N_{\text{sd}}\}$. We can compute H by the procedure described in the previous section. A straightforward diagonalization of H will give us an approximation to the N_{sd} first eigenvectors, given by the columns of A, and eigenvalues of the operator $\hat{H}$. The ground state corresponds to the eigenpair with the lowest eigenvalue.

When we have obtained A we can compute the one-body density matrix

$$\rho_p^q = \langle \Psi | c_p^\dagger c_q | \Psi \rangle = \sum_{IJ} A_I^* A_J \langle \Phi_I | c_p^\dagger c_q | \Phi_J \rangle \tag{7.46}$$

which is done using Algorithm1. Then, we can compute the single-particle density

$$\rho(r) = \sum_{pq} \rho_p^q \phi_p^*(r) \phi_q(r). \tag{7.47}$$

7.4.4 Time evolution of the expansion coefficients.

Assume now that we have an initial condition $A(t_0)$ available. Then, the time evolution of A is given by

$$i\frac{\partial A(t)}{\partial t} = H(t)A(t) \tag{7.48}$$

where $H(t)$ is the Hamiltonian matrix given by

$$H_{IJ}(t) = \langle \Phi_I | \hat{H}(t) | \Phi_J \rangle \,. \tag{7.49}$$

We assume throughout that it is only the one-body part of the Hamiltonian $\hat{H}(t) = \hat{H}_1(t) + \hat{H}_2$ that changes in time. Thus, since we keep the single-particle functions fixed in time we only have to update the one-body part of the Hamiltonian matrix. This reduces the computational time significantly since we do not re-calculate the two-body part in every iteration.

Equation (7.48) can be solved numerically by the fourth order Runge-Kutta (RK4) method (see appendix B.1). Choose a time step Δt such that $t_n = n\Delta t$ for $n = 0, \cdots, N_t$ and let A_n denote an approximation of $A(t_n)$. Define

$$\begin{aligned}
k_1 &= H(t_n)A(t_n) \\
k_2 &= H\left(t_n + \frac{\Delta t}{2}\right)\left(A(t_n) + \frac{\Delta t}{2}k_1\right) \\
k_3 &= H\left(t_n + \frac{\Delta t}{2}\right)\left(A(t_n) + \frac{\Delta t}{2}k_2\right) \\
k_4 &= H\left(t_n + \Delta t\right)\left(A(t_n) + k_3\right)
\end{aligned}$$

Applying the RK4 method to Eq. (7.48) gives the following scheme for computing an approximation A_{n+1} of $A(t_n + \Delta t)$,

$$A_{n+1} = A_n - i\frac{\Delta t}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right). \tag{7.50}$$

7.4.5 Limitations of the implementation

How large systems can we expect to treat with the current implementation of the CI method? One limitation is the storage of two-body integral elements as we pointed out in chapter 7.3.3. Another point is that we form the Hamilton matrix, $H \in \mathbb{C}^{N_{\text{sd}} \times N_{\text{sd}}}$, explicitly. If we use the FCI space the number of linearly independent Slater determinants are given by,

$$N_{\text{sd,FCI}} = \binom{L}{N}$$

where L is the number of single-particle basis functions and N is the number of particles. If we consider a system with $N = 6$ particles and a modest number of basis functions $L = 20$ the storage of H, if we use double-precision, requires roughly

$$\binom{20}{6}^2 \cdot 8 \text{ bytes} = 38760^2 \cdot 8 \text{ bytes} \approx 12\text{GB}$$

of RAM, which is in the limit of the available memory on personal computers.

If we consider a CISD space instead, the number of linearly independent determinants are given by

$$N_{\text{sd,CISD}} = 1 + N(L-N) + \frac{N(N-1)(L-N)(L-N-1)}{4}.$$

A system with $N = 6$ particles and $L = 20$ basis functions would now require approximately

$$1450^2 \cdot 8 \text{ bytes} \approx 17\text{MB}$$

to store H.

Additonally, performing a full diagonalization of H is costly since it requires $\mathcal{O}(N_{\text{sd}}^3)$ operations. Using numpy's *eigh* function to diagonlize a random symmetric $N \times N$ matrix with $N = 1000$ takes on average 1 second on my computer. Thus, diagonalizing a matrix with $N = 30.000$ would take roughly 7 hours and 30 minutes assuming that the time used is on the order $\mathcal{O}(N^3)$.

Taking all of this into account it is clear that such a straightforward implementation of the CI method is only feasible for small systems. If one wants to study larger systems using the CI method one has to make use of iterative methods such as the Lanczos algorithm [27] which computes an approximation to the extremal eigenvalues of H. Use of the Lanczos algorithm avoids the storage of H altogether and only works with the matrix-vector product Hv. In this work the CI method is used to compare with the Coupled Cluster method for small systems and it is sufficient with such a brute-force approach.

7.4.6 Verification of ground state computations.

In order to verify the CI program we will again compare with Ref. [9] for the one-dimensional quantum dot. It should be noted that Zanghellini use the MCTDHF method [8] for both ground state computations and time evolution. Hence, the methods are not equal but for a sufficiently large number of basis functions we expect the CI result to approach the MCTDHF result.

According to Zanghellini *et al.* the exact ground state energy is given by $E_\infty = 0.8247$. In table 7.3 we have computed the FCI ground state energy for an increasing number of basis functions which clearly approaches the exact value. In addition, we have computed the one body density for $L = 5$ which is shown in figure 7.1. Visually the plot is indistinguishable from figure 1 in Ref. [9].

Table 7.3: In the table we have compared ground state energies computed with FCI for increasing number of (spatial) basis functions L with those computed by Zanghellini with MCTDHF. According to Zanghellini *et al.* [9], the exact ground state energy is $E_\infty = 0.8247$ and it is apparent that the FCI energy approaches this value.

$N = 2$	E_{FCI}	$E_{\text{Zanghellini}}$
$L = 2$	0.891597	1.0214
$L = 3$	0.827278	0.8261
$L = 4$	0.826667	0.8255
$L = 5$	0.826169	0.8250
$L = 6$	0.825835	0.8249
$L = 10$	0.825218	–
$L = 15$	0.824947	–

For the two-dimensional quantum dot we compare the ground state energy computed with FCI with those computed with CCSD by Lohne et al. [26]. As we have discussed earlier, FCI and CCSD are equivalent for $N = 2$. In table 7.4 we have computed the ground state energy for $\omega = \{0.5, 1.0\}$ and the energies are in agreement with those reported by Lohne.

Table 7.4: E_{FCI} computed as a function of the number of (spatial) basis functions L for the two-dimensional quantum dot with $N = 2$ and $\omega = \{0.5, 1\}$.

	$\omega = 0.5$	$\omega = 1.0$
L	E_{FCI}	E_{FCI}
6	1.681632	3.038604
10	1.673872	3.025231
15	1.669498	3.017606
21	1.667257	3.013626
28	1.665799	3.011020

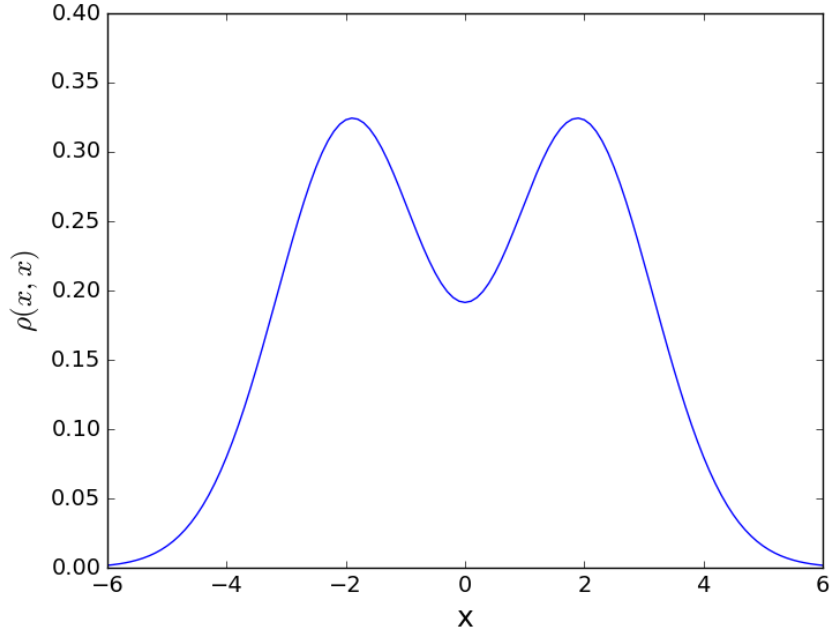


Figure 7.1: The figure shows the one-body density in the ground state for the one-dimensional quantum dot with oscillator frequency $\omega = 0.25$ computed with the FCI method which is in excellent agreement with figure 1 in Ref. [9].

7.4.7 Verification of TDCI

To verify the implementation of the TDCI method we compute the probability of being in the ground state,

$$\begin{aligned} |\langle\Psi(t)|\Psi(0)\rangle|^2 &= \left|\sum_{IJ}^{N_{\text{sd}}} A_J^*(t)A_I(0)\langle\Phi_J|\Phi_I\rangle\right|^2 \\ &\overset{\langle\Phi_J|\Phi_I\rangle=\delta_{IJ}}{=} \left|\sum_J A_J^*(t)A_J(0)\right|^2 \\ &= |\langle A(t), A(0)\rangle|^2 \end{aligned} \tag{7.51}$$

and compare with figure 2 in Ref. [9]. It should be noted that in the article MCTDHF is used, hence we cannot expect perfect agreement. However, for an increasing number of basis functions L we should approach the same answer. Figure 7.2 shows the probability for $L = \{3, 5, 8, 10\}$. For $L = 8$ it is apparent that we approach the exact value and the difference in the computed probability is relatively small when increasing to $L = 10$. To compute the

time evolution of the expansion coefficients we used RK4 with $\Delta t = 10^{-3}$. Furthermore it is observed that $\langle A(t)|A(t)\rangle = 1$ and $\mathrm{tr}(\rho(t)) = 2$ during the simulation as it should.

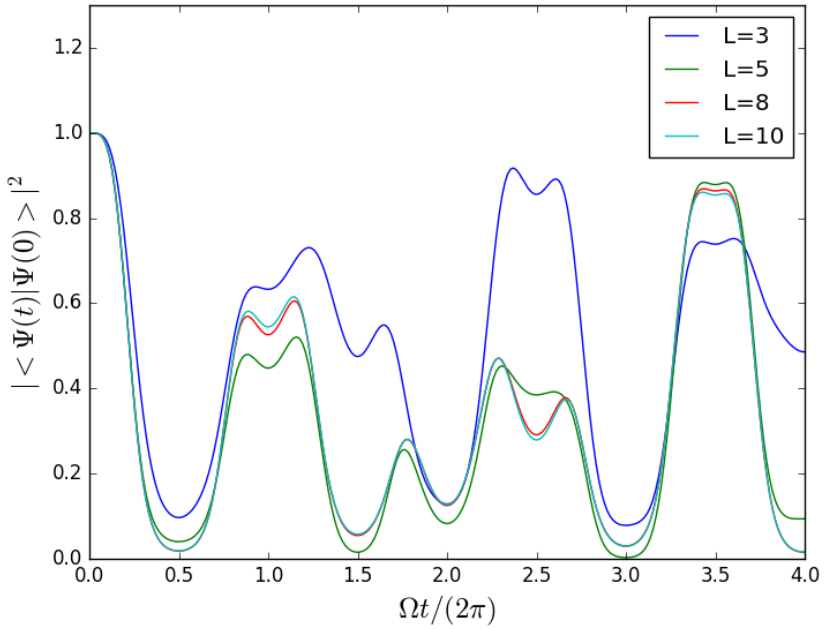


Figure 7.2: Probability of being in the ground state $|\langle\Psi(t)|\Psi(0)\rangle|^2$ for $L = \{3, 5, 8, 10\}$ computed with TD-FCI. Compared with figure 2 in [9] we see that we approach the distribution of the exact solution.

7.5 Coupled Cluster

In this section we describe how to implement the Coupled Cluster Doubles (CCD) approximation. In particular, we discuss how to solve the amplitude equations for the ground state. We show how the amplitudes can be evolved in time using the RK4 method and discuss the limitations of the current implementation. The implementation is verified by comparing the energy and one-body density in the ground state with results obtained with the CID method. The Hartree-Fock state is used as reference determinant for both methods. Similarly, we verify the time evolution by comparing the time evolved energy and one-body density computed with TDCCD and TDCID.

7.5.1 Iterative solution of the amplitude equations for the ground state.

Next, we want to write an implementation of the CCD method. Thus, $T = T_2 = \sum_{ijab} \tau_{ij}^{ab} c_a^\dagger c_i c_b^\dagger c_j$ and $\Lambda = \Lambda_2 = \sum_{ijab} \lambda_{ab}^{ij} c_i^\dagger c_a c_j^\dagger c_b$. The key component to implement is the evaluation of the right-hand sides of the amplitude equations,

$$i\hbar\dot{\tau}_{ij}^{ab} = \langle\tilde{\phi}_{ij}^{ab}|e^{-T_2} H e_2^T|\phi\rangle \tag{7.52}$$

$$-i\hbar\dot{\lambda}_{ab}^{ij} = \langle\tilde{\phi}|(1+\Lambda_2)e^{-T_2}[H, X_{ij}^{ab}]e_2^T|\phi\rangle, \tag{7.53}$$

which are given by (5.64), (5.65) and (5.70), (5.71) respectively. In addition we have to evaluate the (excitation) intermediates (5.66)-(5.69) and the (de-excitation) intermediates (5.72)-(5.78). Before we can compute the time evolution of the amplitudes we have to compute the ground state. Thus, for the ground state we seek amplitudes (λ, τ) such that the equations

$$0 = \langle\tilde{\phi}_{ij}^{ab}|e^{-T_2} H e_2^T|\phi\rangle \tag{7.54}$$

$$0 = \langle\tilde{\phi}|(1+\Lambda_2)e^{-T_2}[H, X_{ij}^{ab}]e_2^T|\phi\rangle, \tag{7.55}$$

are satisfied. The equations are non-linear so we have to solve them iteratively.

Recall from section 5.4.5 that the right-hand side of (7.54) is independent of the λ-ampltitudes. However, the right-hand side of (7.55) depends on the τ-amplitudes. Therefore, we solve for the τ-ampltitudes first. Define,

$$G_1(\tau) \equiv \sum_k (1-\delta_{ik})\, h_i^k \tau_{jk}^{ab} P(ij) + \sum_c (1-\delta_{ca})\, h_c^a \tau_{ji}^{bc}$$

$$G_2(\tau) \equiv \frac{\partial}{\partial\lambda_{ij}^{ab}} \mathcal{E}_{H^{(2)}}[\lambda, \tau],$$

with $G_2(\tau)$ given by Eq. (5.65). Equation (5.64) can be now be written as,

$$\frac{\partial}{\partial\lambda_{ij}^{ab}} \mathcal{E}_{H^{(1)}}[\lambda, \tau] = -h_i^i \tau_{ij}^{ab} - h_j^j \tau_{ij}^{ab} + h_a^a \tau_{ij}^{ab} + h_b^b \tau_{ij}^{ab} + G_1(\tau),$$

and we can rewrite Eq. (7.54) as

$$\begin{aligned}
\langle\tilde{\phi}_{ij}^{ab}|e^{-T_2} H e_2^T|\phi\rangle &= \frac{\partial}{\partial\lambda_{ij}^{ab}} \mathcal{E}_{H^{(1)}}[\lambda, \tau] + \frac{\partial}{\partial\lambda_{ij}^{ab}} \mathcal{E}_{H^{(2)}}[\lambda, \tau] \\
&= -h_i^i \tau_{ij}^{ab} - h_j^j \tau_{ij}^{ab} + h_a^a \tau_{ij}^{ab} + h_b^b \tau_{ij}^{ab} + G_1(\tau) + G_2(\tau) \\
&= 0.
\end{aligned}$$

Let $\tau^{(k)}$ denote the set of amplitudes in iteration k. Then we can compute $\tau^{(k+1)}$ as suggested in [25],

$$\left(\tau_{ij}^{ab}\right)^{(k+1)} = \frac{G_1\left(\tau^{(k)}\right) + G_2\left(\tau^{(k)}\right)}{D_{ij}^{ab}}, \quad D_{ij}^{ab} \equiv h_i^i + h_j^j - h_a^a - h_b^b, \tag{7.56}$$

with the initial condition,

$$\left(\tau_{ij}^{ab}\right)^{(0)} = \frac{u_{ij}^{ab}}{D_{ij}^{ab}}. \tag{7.57}$$

We iterate until self-consitency is reached in the sense that

$$\left|E_{\text{corr}}^{(k+1)} - E_{\text{corr}}^{(k)}\right| < \epsilon, \tag{7.58}$$

with ϵ some pre-defined convergence threshold. In the CCD approximation we have that

$$E_{\text{corr}} \equiv \frac{1}{4}\sum_{ijab} \tau_{ij}^{ab} u_{ab}^{ij}. \tag{7.59}$$

At convergence the ground state energy is given by,

$$E_{\text{CCD}} = E_{\text{ref}} + E_{\text{corr}} \tag{7.60}$$

where

$$E_{\text{ref}} \equiv \sum_i h_i^i + \frac{1}{2}\sum_{ij} u_{ij}^{ij}. \tag{7.61}$$

Notice that if we use the Hartree-Fock state as reference determinant we have

$$E_{\text{ref}} = E_{\text{HF}}.$$

Iterative solvers may run into oscillating solutions and in that case so-called *ad hoc* linear mixing can help the iterations break this cycle [25, 28],

$$\tau^{(k+1)} = \alpha\tau_{\text{no\_mixing}}^{(k+1)} + (1-\alpha)\tau^{(k)}, \quad \alpha \in [0,1].$$

When the τ amplitudes have been computed we can solve for the λ-ampltitudes in similar fashion. Define,

$$\tilde{G}_1(\lambda) \equiv \sum_c (1-\delta_{ca}) h_a^c \lambda_{bc}^{ji} P(ab) + \sum_k (1-\delta_{ki}) h_k^i \lambda_{ab}^{jk} P(ij)$$

$$\tilde{G}_2(\lambda,\tau) \equiv \frac{\partial}{\partial \tau_{ij}^{ab}} \mathcal{E}_{H^{(2)}}[\lambda,\tau]$$

where $\tilde{G}_2(\lambda,\tau)$ is given by Eq. (5.71). We can write equation (5.70) as,

$$\frac{\partial}{\partial\tau_{ij}^{ab}}\mathcal{E}_{H^{(1)}}[\lambda,\tau] = h_a^a\lambda_{ab}^{ij} + h_b^b\lambda_{ab}^{ij} - h_i^i\lambda_{ab}^{ij} - h_j^j\lambda_{ab}^{ij} + \tilde{G}_1(\lambda)$$

while the expression for equation (7.55) becomes,

$$\begin{aligned}\langle\tilde{\phi}|(1+\Lambda_2)e^{-T_2}[H,X_{ij}^{ab}]e_2^T|\phi\rangle &= \frac{\partial}{\partial\tau_{ij}^{ab}}\mathcal{E}_{H^{(1)}}[\lambda,\tau] + \frac{\partial}{\partial\tau_{ij}^{ab}}\mathcal{E}_{H^{(2)}}[\lambda,\tau] \\ &= h_a^a\lambda_{ab}^{ij} + h_b^b\lambda_{ab}^{ij} - h_i^i\lambda_{ab}^{ij} - h_j^j\lambda_{ab}^{ij} + \tilde{G}_1(\lambda) + \tilde{G}_2(\lambda,\tau) \\ &= 0.\end{aligned}$$

We want to solve this equation for the set of λ-amplitudes. Let $\lambda^{(k)}$ denote the amplitudes in iteration k and let τ_* denote the converged τ-amplitudes. Then we can compute $\lambda^{(k+1)}$ by the same procedure we used for the τ amplitudes,

$$\left(\lambda_{ab}^{ij}\right)^{(k+1)} = \frac{\tilde{G}_1\left(\lambda^{(k)}\right) + \tilde{G}_2\left(\lambda^{(k)},\tau_*\right)}{D_{ij}^{ab}}, \tag{7.62}$$

with the initial condition,

$$\left(\lambda_{ab}^{ij}\right)^{(0)} = \frac{u_{ab}^{ij}}{D_{ij}^{ab}}. \tag{7.63}$$

Again we iterate until self-consistency is reached. Note that for the τ-amplitudes we used the difference in the correlation energy as the criteria for self-consistency. The ground state energy does not depend on the λ-amplitudes, so we have to choose some other criterion. We iterate until

$$\max_{ij,ab}\left|\left(\lambda_{ab}^{ij}\right)^{(k+1)} - \left(\lambda_{ab}^{ij}\right)^{(k)}\right| < \delta, \tag{7.64}$$

with δ some pre-defined threshold. Oscillating solutions may occur and one can try to break the cycle by mixing as described for the τ-amplitudes.

In section 5.4.5 it was pointed out that the one- and two body density matrices, ρ_1, ρ_2, depend on both sets of amplitudes τ and λ. Therefore, for the computation of these matrices it is important to consider, in order to be confident, that we have computed the λ amplitudes correctly. In the special case of $N = 2$, these matrices are in theory equivalent to the corresponding density matrices computed with the CID method.

In practice, this will be true to some numerical error. Especially, since the matrices computed with CCD depend on two sets of amplitudes which are computed iteratively while in the CID case the matrices depend only on one

set of parameters, namely the CID expansion coefficients. Let ρ_{CCD} and ρ_{CID} denote density matrices computed with CCD and CID respectively. Then it seems reasonable to expect that the difference,

$$|\rho_{\text{CCD}} - \rho_{\text{CID}}| < \kappa(\epsilon, \delta), \tag{7.65}$$

where κ is some number which possibly depends on the convergence criteria ϵ and δ for τ and λ respectively. The fact that we can compare these matrices in order to verify the computation of the λ-amplitudes was, to a large extent, the main motivation for implementing the CI method at all.

7.5.2 Time evolution of the amplitudes.

Suppose now that we have computed amplitudes (λ_*, τ_*) such that equations (7.54) and (7.55) are satisfied. We now want to compute the time evolution of the amplitudes according to,

$$\begin{aligned} i\hbar\dot{\tau}_{ij}^{ab} &= \frac{\partial}{\partial\lambda_{ij}^{ab}}\mathcal{E}_{H^{(1)}}[\lambda,\tau] + \frac{\partial}{\partial\lambda_{ij}^{ab}}\mathcal{E}_{H^{(2)}}[\lambda,\tau] \\ -i\hbar\dot{\lambda}_{ab}^{ij} &= \frac{\partial}{\partial\tau_{ij}^{ab}}\mathcal{E}_{H^{(1)}}[\lambda,\tau] + \frac{\partial}{\partial\tau_{ij}^{ab}}\mathcal{E}_{H^{(2)}}[\lambda,\tau], \end{aligned}$$

where $H^{(1)}$ might depend on time. In that case we have to remember that the one-body elements must be updated. This is now an initial value problem which can be solved numerically by the class of Runge-Kutta methods. We choose to use the fourth-order Runge-Kutta (RK4) method since it is quite straightforward to implement while still being numerically accurate (see appendix B.1).

Let $\tau^{(n)}, \lambda^{(n)}$ denote the amplitudes at time $t_n = n\Delta t$ with Δt being the timestep and define,

$$f\left(t_n, \tau^n\right) \equiv -\frac{i}{\hbar}\left(\frac{\partial}{\partial\lambda_{ij}^{ab}}\mathcal{E}_{H^{(1)}}[\lambda^n,\tau^n] + \frac{\partial}{\partial\lambda_{ij}^{ab}}\mathcal{E}_{H^{(2)}}[\lambda^n,\tau^n]\right), \tag{7.66}$$

$$g\left(t_n, \lambda^n, \tau^n\right) \equiv \frac{i}{\hbar}\left(\frac{\partial}{\partial\tau_{ij}^{ab}}\mathcal{E}_{H^{(1)}}[\lambda^n,\tau^n] + \frac{\partial}{\partial\tau_{ij}^{ab}}\mathcal{E}_{H^{(2)}}[\lambda^n,\tau^n]\right). \tag{7.67}$$

Then we get the following scheme for computing the amplitudes at time t_{n+1} with the RK4 method,

$$(\tau_{ij}^{ab})^{(n+1)} = (\tau_{ij}^{ab})^n + \frac{\Delta t}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) \tag{7.68}$$

$$(\lambda_{ab}^{ij})^{(n+1)} = (\lambda_{ab}^{ij})^n + \frac{\Delta t}{6}\left(m_1 + 2m_2 + 2m_3 + m_4\right), \tag{7.69}$$

with $(\tau_{ij}^{ab})^{(0)} = (\tau_{ij}^{ab})_*$ and $(\lambda_{ab}^{ij})^{(0)} = (\lambda_{ab}^{ij})_*$. The expressions on the right hand side are given by

$$\begin{aligned}
k_1 &= f\left(t_n, \tau^n\right), \\
k_2 &= f\left(t_n + \frac{\Delta t}{2}, \tau^n + \frac{\Delta t}{2} k_1\right), \\
k_3 &= f\left(t_n + \frac{\Delta t}{2}, \tau^n + \frac{\Delta t}{2} k_2\right), \\
k_4 &= f\left(t_n + \Delta t, \tau^n + \Delta t k_3\right).
\end{aligned}$$

Similarly we have that,

$$\begin{aligned}
m_1 &= g\left(t_n, \lambda^n, \tau^n\right), \\
m_2 &= g\left(t_n + \frac{\Delta t}{2}, \lambda^n + \frac{\Delta t}{2} m_1, \tau^n + \frac{\Delta t}{2} k_1\right), \\
m_3 &= g\left(t_n + \frac{\Delta t}{2}, \lambda^n + \frac{\Delta t}{2} m_2, \tau^n + \frac{\Delta t}{2} k_2\right), \\
m_4 &= g\left(t_n + \Delta t, \lambda^n + \Delta t m_3, \tau^n + \Delta t k_3\right).
\end{aligned}$$

7.5.3 Implementation details

In the following we will discuss some details concerning the implementation of the TDCCD method.

Assume that we have N particles and L basis functions and that the one- and two-body integral elements are given. Consider the representation of the amplitudes, $\{\tau_{ij}^{ab}\}, \{\lambda_{ab}^{ij}\}$ where $i, j = 1, ..., N$ and $a, b = N+1, \cdots, L$. Since the equations of motion for the amplitudes contain the imaginary unit explicitly we have to use complex numbers. In addition to the amplitudes, we have to store the intermediates (5.66)-(5.69) and (5.72)-(5.78). In a naive implementation the amplitudes and intermediates are conveniently stored using two-, four- and six-dimensional arrays. For larger systems, the storage of especially the intermediates in Eqs. (5.72) and (5.73) would quickly be prohibitive.

However, as we have mentioned several times already, we consider only relatively small systems and such a naive storage of the amplitudes and intermediates is adequate. Another aspect is that since we use a time dependent Hamiltonian, amplitudes which are zero initially can become non-zero. Thus,

in order to avoid storage of redundant zeros one has to take the time dependent perturbation into consideration.

Using the *einsum* function, evaluation of the intermediates, one- and two-body density matrices (5.79)-(5.86), can be written in a compact and efficient manner. Listing (7.2) demonstrates how we can compute the intermediate

$$\chi_{cj}^{kb} = u_{cj}^{kb} + \frac{1}{2}\tau_{jl}^{bd} u_{cd}^{kl}.$$

Listing 7.2: Example of evaluation of Coupled Cluster Intermediate using the einsum function.

```
#u is a 4-dimensional array containing
#the anti-symmetrized integral
#elements <pq|u|rs>
o = slice(0,N) #range of indices i,j,k,l,...
v = slice(N,L) #range of indices a,b,c,d,...
Xkbcj = u[o,v,v,o] + 0.5*einsum('bdjl,klcd->kbcj',
                                t2, u[o,o,v,v])
```

Similarly the right-hand sides of the equations,

$$\left(\tau_{ij}^{ab}\right)^{(k+1)} = \frac{G_1\left(\tau^{(k)}\right) + G_2\left(\tau^{(k)}\right)}{D_{ij}^{ab}},$$
$$\left(\lambda_{ab}^{ij}\right)^{(k+1)} = \frac{\tilde{G}_1\left(\lambda^{(k)}\right) + \tilde{G}_2\left(\lambda^{(k)}, \tau_*\right)}{D_{ij}^{ab}}$$

can be implemented by repeated use of *einsum* calls. We note that equations (7.66) and (7.67) can be written in terms of $G_1(\tau), G_2(\tau)$ and $\tilde{G}_1(\lambda), \tilde{G}_2(\lambda, \tau)$ in the following manner,

$$f\left(t_n, \tau^n\right) = -\frac{i}{\hbar}\left(-D_{ij}^{ab}(\tau_{ij}^{ab})^n + G_1(\tau^n) + G_2(\tau^n)\right),$$
$$g\left(t_n, \lambda^n, \tau^n\right) = \frac{i}{\hbar}\left(-D_{ij}^{ab}(\lambda_{ab}^{ij})^n + \tilde{G}_1(\lambda^n) + \tilde{G}_2(\lambda^n, \tau^n)\right).$$

Hence, it is sufficient to write methods/functions which compute $G_1, G_2, \tilde{G}_1$ and $\tilde{G}_2$ in order to update the amplitudes both for ground state computations and for time evolution. We also note that G_2 and $\tilde{G}_2$ depend on the intermediates which must be updated in each iteration. A complete TDCCD program is listed in appendix C.3.

7.5.4 Limitations of the implementation

How large systems can we expect to treat with the current CCD/TDCCD implementation?

One limitation is the storage of the amplitudes and intermediates. As usual, let N denote the number of particles and L the number of basis functions. The total number of τ and λ-amplitudes are $2N^2(L-N)^2$. There are a total of

$$N^2(L-N)^2 + N^4 + (L-N)^2 + N^2$$

intermediates (5.66)-(5.69) while the total number of intermediates (5.72)-(5.78) are given by

$$\begin{aligned} &N^4(L-N)^2 + N^2(L-N)^4 \\ &+N^4 + N^2(L-N)^2 \\ &+N^2 + (L-N)^4 + (L-N)^2. \end{aligned}$$

The total number of amplitudes and intermediates, $S(N,L)$, is then given by

$$\begin{aligned} S(N,L) =&N^2(L-N)^4 + N^4(L-N)^2 + (L-N)^4 \\ &+4N^2(L-N)^2 + 2N^4 + 2(L-N)^2 + 2N^2 \end{aligned}$$

and is completely dominated by the storage of ξ^{dcj}_{kab} (Eq. (5.73)) totalling $N^2(L-N)^4$ elements. If we use complex numbers, with double-precision, each element requires 16 bytes of memory.

We want to study the time evolution of the two-dimensional quantum dot, where we consider closed shell-systems with the possible number of particles limited to $N = \{2, 6, 12, 20, 42, 56, 72, 90, 110, \cdots\}$ being the so-called magic numbers. In table 7.5 we list the memory requirement for different system sizes N, L. If we assume that 8GB of RAM are available on a standard desktop considering the time evolution of a quantum dot with $N = 6$ electrons and $L = 56$ basis functions or $N = 12$ and $L = 42$ is in reach with the current implementation without taking special measures to handle the storage of the amplitudes and intermediates.

We are, of course, also limited by the fact that computing new τ-amplitudes scales as $\mathcal{O}(N^2(L-N)^4)$ while computing new λ-amplitudes scales as $\mathcal{O}(N^3(L-N)^4)$. In order to estimate the duration of a simulation it is interesting to measure the time the program needs to update the τ and the λ-amplitudes, including the computation of the intermediates. We expect that the computation of the λ-amplitudes are on the order of $\mathcal{O}(N)$ more expensive to

Table 7.5: The table shows the estimated memory needed for a TDCCD calculation using with N particles and L basis functions. It is assumed that double-precision complex numbers are used to store amplitudes and intermediates.

N	L	Memory [GB]
2	72	2.35
2	90	5.85
2	110	13.24
6	42	1.08
6	56	3.92
6	72	11.77
12	30	0.37
12	42	2.24
12	56	9.53

compute than the τ-amplitudes. In table (7.6) we have measured the time it takes to compute one new set of τ and λ amplitudes including updating the intermediates for different N and L values. The timings are extremely rough and are taken just as an indication on how much time we need to compute the time evolution of different system sizes. Since we use the RK4 method, we have to compute new amplitudes four times per timestep when evolving the amplitudes in time. Running a simulation with 1000 timesteps with $N = 6$ and $L = 42$ would take approximately 10.5 hours.

7.5.5 Verification of ground state computations.

In chapter 7.4 we demonstrated that the CI/TDCI program was implemented correctly. Since the CC approximations are equal to the corresponding CI methods for $N = 2$ we can use this to verify the implementation of the CCD/TDCCD program. One of the main difficulties during the development was to be sure that the λ-amplitudes were computed correctly, since the ground state energy does not depend on these. However, the density matrices, which we can compare directly with those computed with the Configuration Interaction method depend on the λ-amplitudes.

Furthermore, we can compare the time evolved energies and density matrices computed with TDCC and TDCI respectively to be confident that the TDCC method is correctly implemented. For these computations we compare quantities computed with the CCD and CID approximation using the Hartree-Fock state as the reference determinant.

Table 7.6: The table shows the time that the current implementation of the TDCCD method needs to compute new τ and λ-amplitudes for increasing N and L.

N	L	$t_\tau[s]$	$t_\lambda[s]$	t_λ/t_τ
2	12	0.003	0.004	1.33
2	20	0.01	0.04	4
2	30	0.06	0.20	3.33
2	42	0.20	0.80	4
2	56	0.57	2.38	4.17
2	72	1.48	6.75	4.56
6	20	0.04	0.26	6.5
6	30	0.21	1.89	9
6	42	0.85	8.57	10.08
6	56	2.70	32.37	11.98
12	30	0.43	5.27	12.25
12	42	2.08	35.85	17.23

First, we compute the ground state energy of the two-electron one-dimensional quantum dot for $\omega = 0.5$ and $L = \{4, 6, 8, 10\}$. For the shielded Coulomb potential we use $a = 0.25$. The computation of integral elements was done with $x \in [-7, 7]$ with gridspacing $\Delta x = 0.1$.

The convergence criterion for the τ-amplitudes was set to $\epsilon = 10^{-6}$ and the mixing parameter is after some trial and error set to $\alpha = 0.93$ in order to obtain convergence to the chosen precision. The results are summarized in table 7.7 and we see that the difference in the CCD and CID energies is lower than ϵ.

Next, we want to compare the one-body densities computed with CCD and CID. The convergence criteria for the λ-amplitudes was set to $\delta = 10^{-5}$ while the mixing parameter was set to $\beta = 0.85$. Figure 7.3 shows the electron density and the difference, $\kappa = |\rho_{\text{CCD}} - \rho_{\text{CID}}|$, for $L = 10$. Let κ_{max} denote the maximum of the absolute diffrence. With the given parameters we get that $\kappa_{\max} = 2.58 \cdot 10^{-7}$. Previously we argued that κ depends on (ϵ, δ). In table 7.8 we compute $\kappa_{\max}$ for $L = 10$ as a function of δ keeping $\epsilon = 10^{-6}$ fixed. Keeping ϵ fixed seems reasonable due to the fact that the λ-amplitudes depends on the τ-amplitude while the converse is not true.

Table 7.7: Ground State energies computed for the two-electron one-dimensional quantum dot computed with CCD and CID using the HF state as reference determinant. For the CCD calculation the convergence threshold used in Eq. (7.58) for the τ-amplitudes was set to $\epsilon = 10^{-6}$.

$\omega = 0.5$	E_{CCD}	E_{CID}	$\lvert E_{\text{CCD}} - E_{\text{CID}} \rvert$
$L = 4$	1.4374447	1.4374443	$3.64 \cdot 10^{-7}$
$L = 6$	1.4352960	1.4352956	$3.63 \cdot 10^{-7}$
$L = 8$	1.4332307	1.4332310	$2.77 \cdot 10^{-7}$
$L = 10$	1.4321945	1.4321942	$3.29 \cdot 10^{-7}$

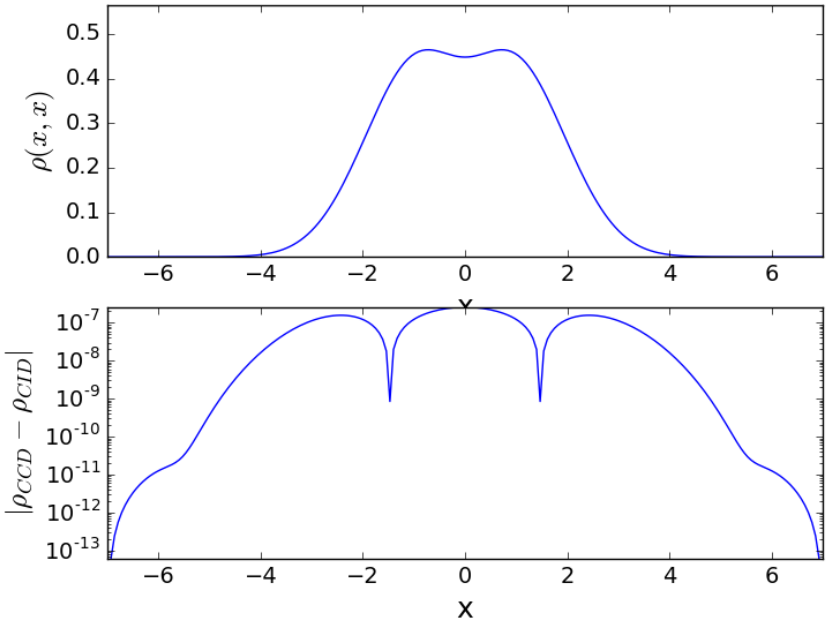


Figure 7.3: The figure shows the one-body density in the ground state for the one-dimensional quantum dot with oscillator frequency $\omega = 0.5$ and $L = 10$ computed with the CCD method.

7.5.6 Verification of TDCCD

We compute the time evolution of the two-electron one-dimensional quantum dot with $L = 4$. The frequency of the laser is set to $\Omega = 8\omega$ where ω is the frequency of the harmonic oscillator potential. For this simulation we used

Table 7.8: The maximum of the difference $\kappa = |\rho_{\text{CCD}} - \rho_{\text{CID}}|$, computed as a function of δ with $\epsilon = 10^{-6}$ held fixed where δ and ϵ are the convergence critetion for the λ- and τ-amplitudes respectively. It is apparent that lowering δ results in a lower $\kappa_{\max}$.

δ	$\kappa_{\max}(\delta)$
10^{-1}	$2.58 \cdot 10^{-3}$
10^{-2}	$4.63 \cdot 10^{-4}$
10^{-3}	$2.58 \cdot 10^{-5}$
10^{-4}	$2.35 \cdot 10^{-6}$
10^{-5}	$2.58 \cdot 10^{-7}$

$\omega = 0.5$. The timestep is $\Delta t = 10^{-3}$ and the simulation is run until,

$$\frac{\Omega t_*}{2\pi} = 5.$$

During the time evolution we monitor the energy and one-body density and compare the values obtained with the corresponding quantities obtained with TDCID. Figure 7.4 shows one-body density at the end of the simulation computed with TDCCD and TDCID and the absolute difference over the entire grid. The maximum difference is on the order of $\sim 10^{-5}$ which seems satisfactory.

Furthermore, as Kvaal points out in Ref. [1] expectation values may gain small imaginary parts during the TDCCD computation. Therefore, we also monitor $|\text{Im}E_{\text{CCD}}(t)|$. From figure 7.5 we see that the difference between the CCD and CID energy varies over the simulation with a maximum of $\sim 10^{-3}$. Furthermore, we see that we have a imaginary part in the CCD energy which is small compared to the real part.

In conclusion, the results agree well with theory in the sense that the difference between the time evolved energies and single-particle densities are small. Furthermore, we observe that the time evolved CCD method picks up a non-zero imaginary component which is several orders of magnitude smaller than the real part. In total, this really is the pinnacle of the thesis since we have demonstrated that we have, for $N = 2$, a working implementation of a time dependent Coupled Cluster method using a time dependent Hamiltonian. In principle, there should not be any difference going to higher particle numbers. However, one can never test and verify enough and to guarantee the absence of human errors is close to impossible.

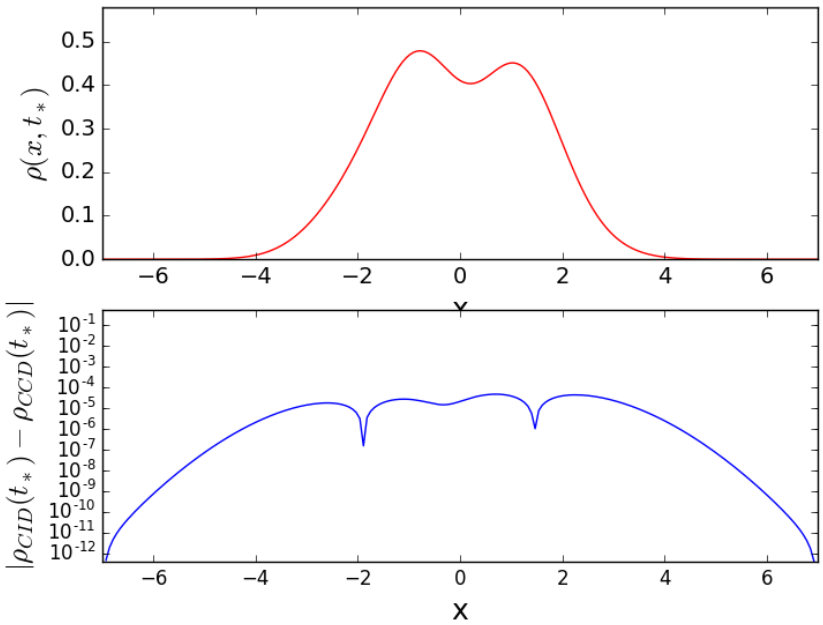


Figure 7.4: The figure shows the time evolved one-body density computed with CID and CCD using the Hartree-Fock state as reference determinant. We have also plotted the absolute difference between the two at t^* and we see that the largest difference over the entire grid is on the order of 10^{-5} as we expect since the methods should be equal for $N = 2$.

7.6 Additional results.

Now that it seems plausible that the TDCCD implementation is correct we will perform the following numerical experiments on two-dimensional quantum dot systems:

We compute the ground state with the CCD method using the Hartree-Fock state as reference determinant. Then, we compute the time evolution with the TDCCD method, using a laser frequency $\Omega = \Delta E = (E_k - E_0)$, where E_0 is the ground state energy and E_k is the energy of some excited state. According to Ref. [12] the result should be an excitation of the system. During the simulation we monitor the energy and the imaginary component of the energy expectation value given by Eq. (5.58).

If we want to compute excited states with the Coupled Cluster method we have to use the so-called Equation-of-Motion Coupled-Cluster (EOM-CC) method, see for example Refs. [28, 29]. However, we can use the CI method to compute an approximation to excited state energies E_k.

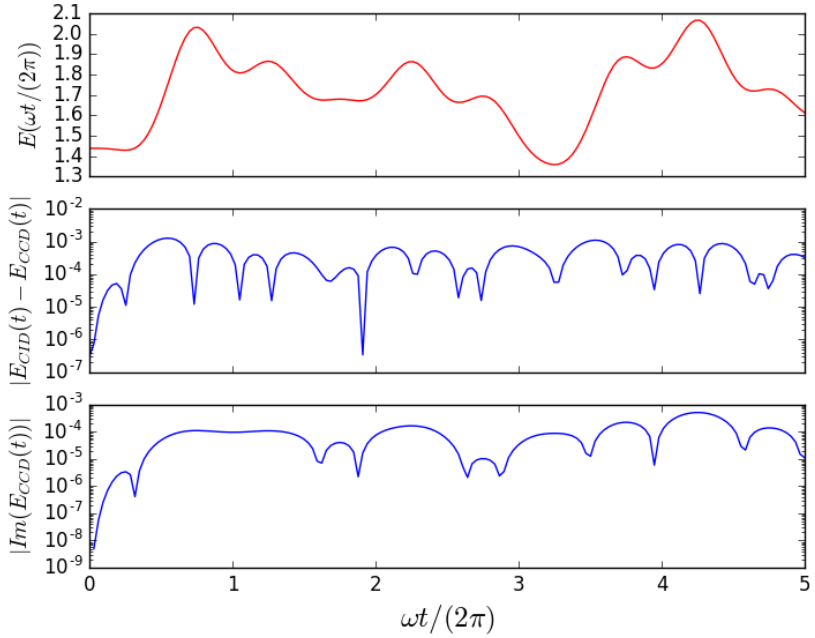


Figure 7.5: The figure shows the time evolved energy computed with CID and CCD using the Hartree-Fock state as reference determinant. Additionaly we have plotted the absolute difference between the energy over the entire simulation. Furthermore, we monitor the imaginary part of the time evolved CCD energy which we can see is small compared to the real part as expected according to Ref. [1].

Furthermore, using the expressions for the integral elements in Ref. [23] we compute ground state energies for the three-dimensional quantum dot with the HF, CISD and CCD method.

7.6.1 Time evolution of the two-dimensional quantum dot

In table (7.9) we have computed the ground state energy with the CCD method using the Hartree-Fock state as reference determinant for the two-dimensional quantum dot with $N = \{2, 6\}$ confined electrons. Additionally, we have used CID (with HF as reference determinant) to compute the energy of the first excited state of the CCD/CID wavefunction. We also include the first excited energy using the CISD approximation.

We note that the first excited state of the CID and CISD is not necessarily

the same. When using only doubles excitations we "miss" contributions from singles excitations. Using the Hartree-Fock state we know that

$$\langle \Phi_i^a | \hat{H} | \Phi \rangle = 0 \tag{7.70}$$

due to Brillouin's theorem (3.12). However, we have no guarantee that terms on the form $\langle \Phi_{ij}^{ab} | \hat{H} | \Phi_k^c \rangle$ are zero. Since we work with in the CCD approximation we will tune the laser with the difference between the ground state energy and the first excited CID energy.

Table 7.9: The table shows the ground state energies of the two-dimensional quantum dot with $N = \{2, 6\}$ confined electrons computed with the CCD method. Additionally, we have computed the first excited energy of the CID and CISD wavefunctions.

N	ω	L_{spatial}	$E_{\text{CCD-HF}}$	$E_{\text{CISD,1}}$	$E_{\text{CID-HF,1}}$
2	0.28	6	1.0330	1.1528	1.4391
2	0.5	6	1.6822	1.9243	2.4306
2	1.0	6	3.0393	3.6079	4.6142
2	0.28	10	1.0291	1.1452	1.4257
2	0.5	10	1.6742	1.9163	2.4169
2	1.0	10	3.0256	3.5996	4.6003
6	0.5	6	12.9020	13.0275	13.0275
6	1.0	6	21.4242	21.8889	22.0821
6	0.5	10	12.0575	12.4861	12.6328
6	1.0	10	20.4294	21.2175	21.6374

We will now investigate the time evolution of both a quantum dot with $N = 2$ and $N = 6$ electrons. We radiate the systems with a laser polarized along the x-axis, modelled by

$$\hat{H}_1(t) = \mathcal{E}_0 \sin(\Omega t) \sum_{i=1}^{N} x_i \tag{7.71}$$

where $\mathcal{E}_0$ is the amplitude and Ω is the laser frequency. The laser frequency is set to the energy difference between the ground state and first excited state, $\Omega = E_{\text{CID,1}} - E_{\text{CCD,0}}$, while the amplitude is set to $\mathcal{E}_0 = 1$. If the system is radiated for for a sufficiently long time we expect an excitation of the system. For the two-elctron system we use $L = 6$ spatial basisfunctions while for the six-electron system we use $L = 10$ spatial basis functions. For all simulations we used the RK4 method with timestep $\Delta t = 10^{-3}$.

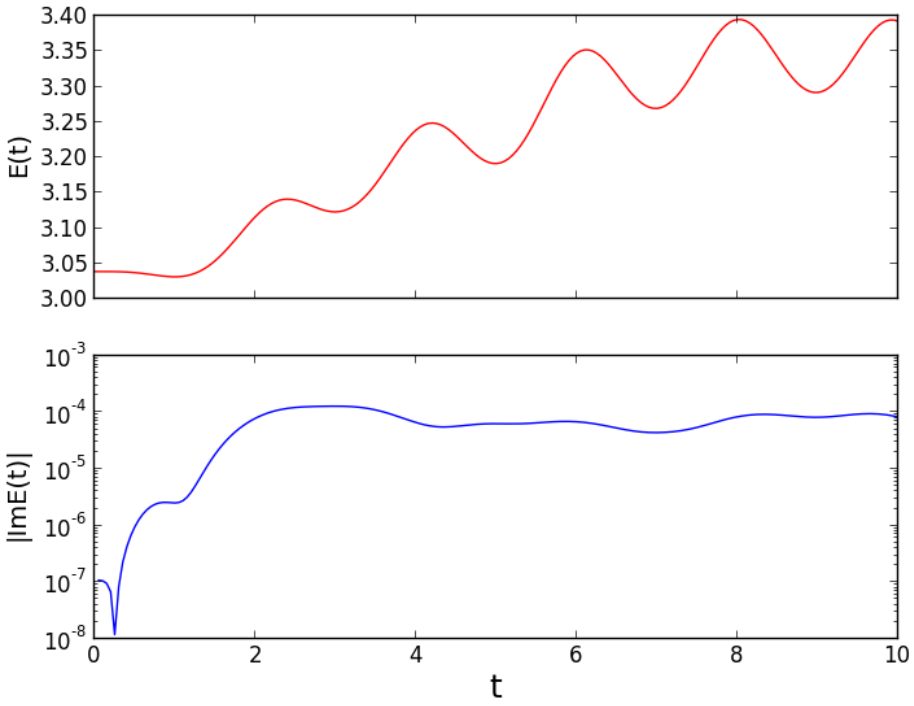


Figure 7.6: The time evolved energy of the two-dimensional quantum dot with $N = 2$ confined electrons. Here the strength of the confining potential is set to $\omega = 1$.

Figure 7.6-7.8 shows the time evolved energy of the two-electron quantum dot for $\omega = \{0.28, 0.5, 1\}$. It is clear that all systems have experienced an increase in energy by the end of the simulation. Furthermore, we see that it is harder to excite the more strongly confined system. Finally, figure 7.9 and 7.10 shows the time evolved energy for the six-electron quantum dot, where the strength of the confining potential is set to $\omega = 1$ and $\omega = 0.5$, respectively. Again, we see that both system's have been excited from the ground state. We note that for all system's the imaginary part of the energy expectation value is small compared to the real part.

7.6.2 Ground state energies of the three-dimensional quantum dot

We have computed the ground state energies of the three-dimensional isotropic quantum dot with oscillator frequency $\omega = 1$ for $N = \{2, 8\}$ confined elec-

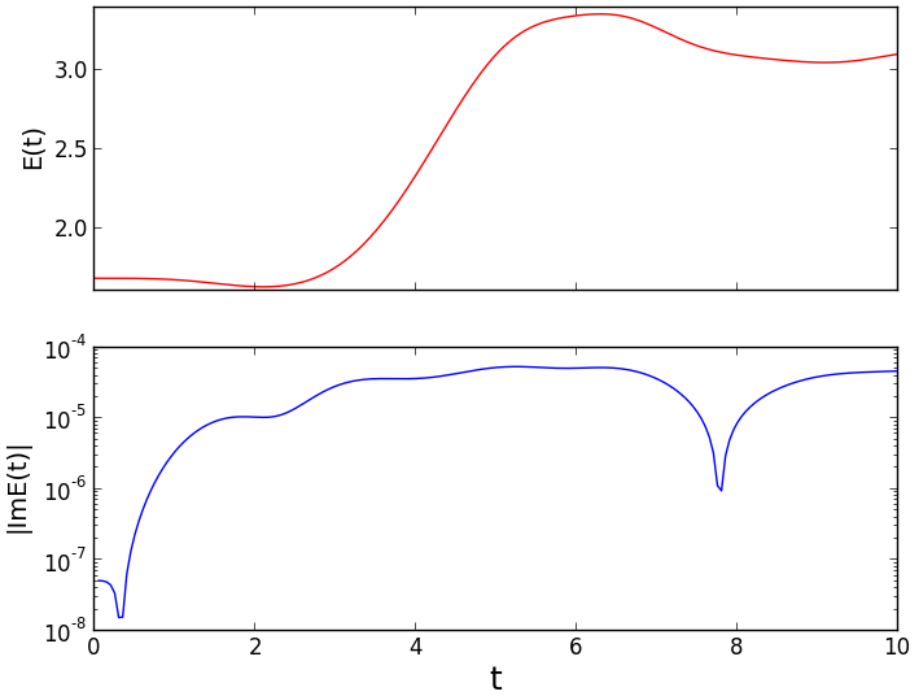


Figure 7.7: The time evolved energy of the two-dimensional quantum dot with $N = 2$ confined electrons. Here the strength of the confining potential is set to $\omega = 0.5$.

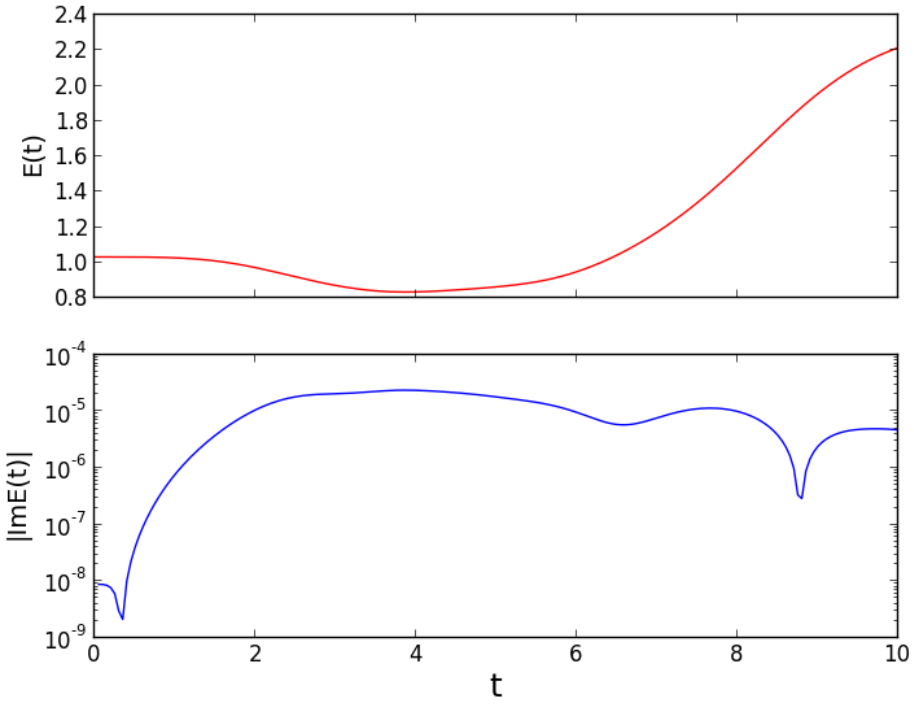


Figure 7.8: The time evolved energy of the two-dimensional quantum dot with $N = 2$ confined electrons. Here the strength of the confining potential is set to $\omega = 0.28$.

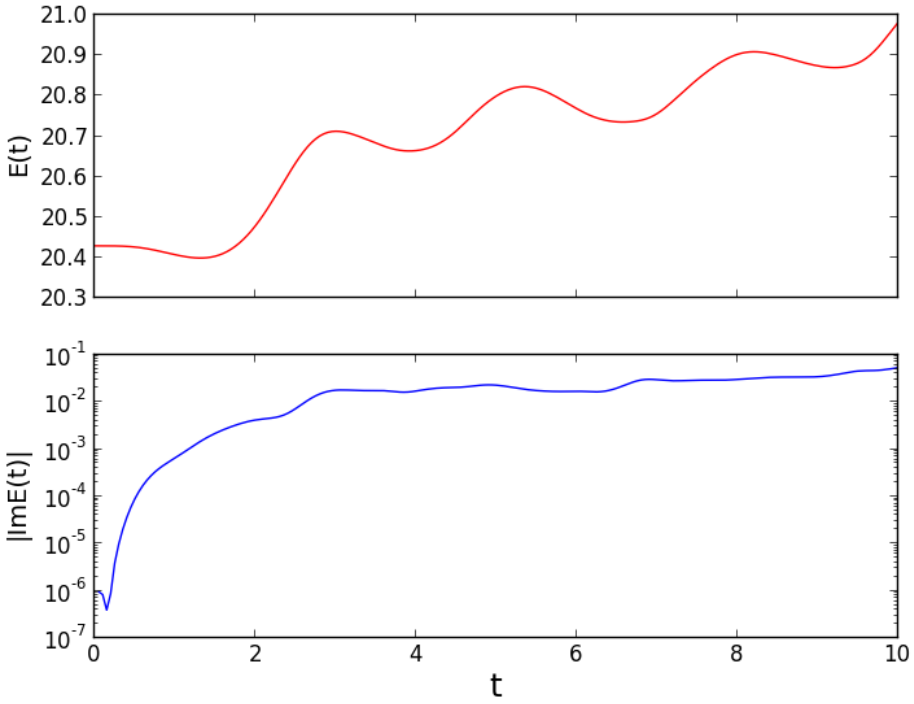


Figure 7.9: The time evolved energy of the two-dimensional quantum dot with $N = 6$ confined electrons. Here the strength of the confining potential is set to $\omega = 1$.

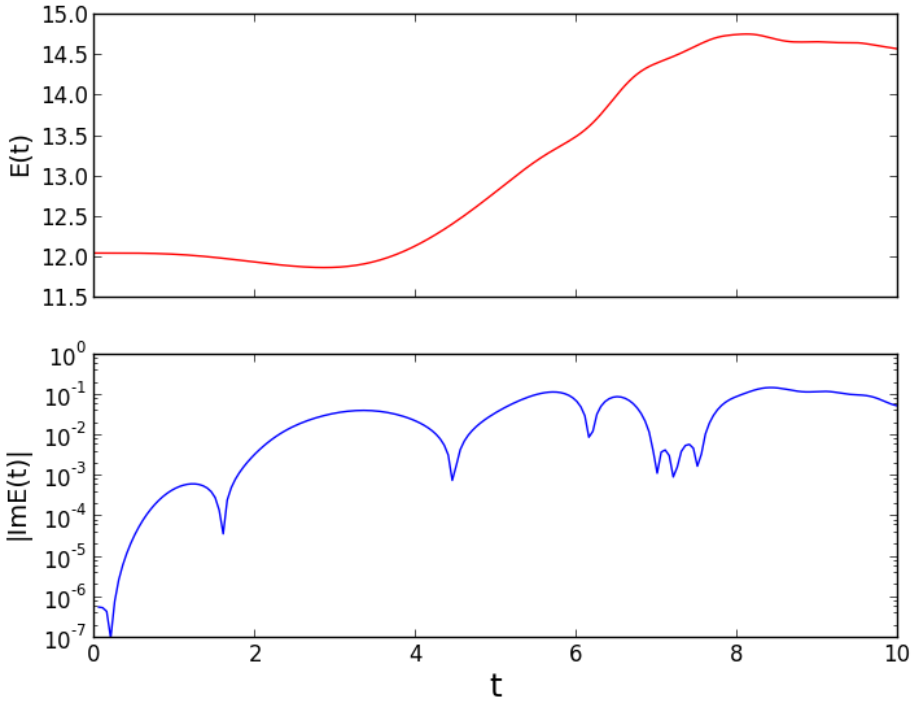


Figure 7.10: The time evolved energy of the two-dimensional quantum dot with $N = 6$ confined electrons. Here the strength of the confining potential is set to $\omega = 0.5$.

trons using HF, FCI and CCD with HF basis. The results, which are summarized in tables 7.10 and 7.11, clearly approaches the values reported by Høgberget [30] using Quantum Monte-Carlo methods.

Table 7.10: The table shows the ground state energies of a three-dimensional quantum dot with two confined electrons. L_{spatial} denotes the number of spatial basis functions used in the calculation while ω is the frequency of the confining potential.

N	ω	L_{spatial}	E_{HF}	E_{FCI}	$E_{\text{CCD-HF}}$
2	0.01	4	0.109788	0.092483	0.092483
2	0.1	4	0.552313	0.526418	0.526418
2	0.28	4	1.262200	1.235277	1.235277
2	0.5	4	2.064189	2.037123	2.037123
2	1.0	4	3.797884	3.770825	3.770825
2	0.01	10	0.093727	0.080119	0.080485
2	0.1	10	0.529042	0.501229	0.501232
2	0.28	10	1.237335	1.206433	1.206467
2	0.5	10	2.038786	2.007186	2.007225
2	1.0	10	3.772064	3.740191	3.740222
2	0.01	20	0.093727	0.079392	0.079427
2	0.1	20	0.529042	0.500716	0.500728
2	0.28	20	1.237335	1.204368	1.204368
2	0.5	20	2.038786	2.004104	2.004105
2	1.0	20	3.772064	3.736006	3.736008
2	0.01	35	0.092743	0.079203	not converged
2	0.1	35	0.529042	0.500437	0.500550
2	0.28	35	1.237180	1.203299	1.203347
2	0.5	35	2.038452	2.002486	2.002516
2	1.0	35	3.771498	3.733756	3.733775

Table 7.11: The table shows ground state energies for the three-dimensional quantum dot with $N = 8$ confined electrons computed with the Hartree-Fock and CCD methods. For increasing number of basis functions the values approach the values obtained with Variational Monte Carlo methods in Ref. [30].

N	ω	L_{spatial}	E_{HF}	$E_{\text{CCD-HF}}$
8	0.1	10	6.642160	not converged
8	0.28	10	13.306843	13.1728
8	0.5	10	20.173748	20.035253
8	1.0	10	33.995902	33.853376
8	0.1	20	6.005028	not converged
8	0.28	20	12.483030	12.3348
8	0.5	20	19.242582	19.070177
8	1.0	20	32.944312	32.748237
8	0.5	35	19.224922	19.0242
8	1.0	35	32.936945	32.717020

Chapter 8

Conclusions and Perspective

8.1 Summary

The aim of this thesis has been to study numerical methods for solving the time-dependent Schrödinger equation for many-body problems. In particular, we wanted to implement the Time-Dependent Coupled-Cluster method which is a simplified version of the Orbital-Adaptive Time-Dependent Coupled-Cluster (OATDCC) method presented in the article by Kvaal in Ref. [1].

To begin with, we reviewed many-body theory with an emphasis on the second quantization formalism. In order to solve the time-dependent Schrödinger equation it is necessary to first solve the so-called time-independent Schrödinger equation (TISE). We discussed three of the most commonly used methods for solving the TISE, namely the Hartree-Fock (HF), the Configuration Interaction (CI) and the Coupled Cluster (CC) methods. Next, we reviewed how to extend the CI and the CC methods to the time domain giving rise to the Time-Dependent Configuration Interaction (TDCI) method and the Time-Dependent Coupled-Cluster (TDCC) method.

Of particular importance is the fact that the CI and CC methods give the same results for systems with $N = 2$ particles. This fact was used as a basis for establishing the validity of the implementation of the TDCC method. Implementations of the Hartree-Fock and Configuration Interaction methods were verified by comparing with previous studies of one- and two-dimensional quantum dot systems from Refs. [9,26,28]. The article by Zanghellini *et al.* [9] examines also the time evolution of the one-dimensional quantum dot which was used to verify the implementation of the TDCI method. Finally, we could establish the validity of the implementation of the TDCCD method by comparing the time evolution with a working TDCI program. The methods were shown to produce the same results within a numerical error on the

order of $\sim 10^{-3}$ for time evolved energies when applied to a one-dimensional quantum dot system with $N = 2$ electrons.

We used the TDCC method to compute the time evolution of two-dimensional quantum dot systems with $N = 2$ and $N = 6$ electrons. Radiating the quantum dots with a laser, tuned with the energy difference between the ground state and the first excited state, we were able to produce excited states for these systems. Additionally, we used our Hartree-Fock, Configuration Interaction and Coupled Cluster codes to compute ground state energies of three-dimensional quantum dot systems using integral elements given by Vorrath [23]. Our results agree well with the Variational Monte Carlo and Diffusion Monte Carlo calculations of Ref. [30].

In summary, we have developed an extensive software suite in Python and C++ that is flexible enough to handle different quantum mechanical systems, spanning from solid state physics devices like quantum dots to atoms and molecules, or atomic nuclei. Here we have focused on quantum dots, but our software is not limited to systems described by harmonic oscillator basis sets only. A proper object orientation, as implemented in the software suite developed by us, allows for straightforward extensions and studies of other quantum mechanical systems.

8.2 Future work

We have seen that the TDCC method produces the same results as the TDCI method when applied to a system with $N = 2$ particles. Furthermore, we have demonstrated that we can simulate system's with more than two particles, using the two-dimensional quantum dot with six particles as an example.

There are several possibilities for future work. In contrast to the MCT-DHF and the TDCI methods (see for example Ref. [8]) which suffer from exponential scaling, the TDCC and OATDCC approaches achieve polynomial scaling. A more memory efficient and parallelized version of the current implementation is planned for studies of larger systems. The C++ code developed as a part of this thesis project will then form the basis for a suite of software tailored to existing High-Performance Computing facilities. The present investigations are planned to form the basis for an article in the Journal of Chemical Physics [31].

Extending the implementation to handle the full OATDCC discussed in Ref. [1] is another possibility, especially since the amplitude equations are the same for both methods.

The current implementation is general in the sense that no assumption

is made on the systems other than it is fermionic. Thus, given integral elements one can study other systems than electrons confined by the harmonic oscillator. This applies equally well to bosonic systems, that is collections of quantum mechanical systems where the total wavefunction is symmetric and the particles carry integer values of the intrinsic spin.

Using the integral elements given by Vorrath [23] a more thorough study of the three dimensional quantum dot is possible. In particular, one can study systems where the confining potential is deformed in the z-direction allowing for the inclusion of other potentials than the harmonic oscillator. The latter is of importance if one wishes to study the time evolution of quantum dots systems tailored to construct quantum gates and circuits.

Appendices

Appendix A

The Quantum Harmonic Oscillator

The quantum (isotropic) harmonic oscillator in dimension d, where $d = 1, 2$ or 3, is where we solve the Schrödinger equation for one particle confined to the potential,

$$V(r) = \frac{1}{2}\mu\omega^2 r^2, \quad r^2 = \sum_{i=1}^{d} x_i^2. \tag{A.1}$$

Here,

$$r = \sqrt{\sum_{i=1}^{d} x_i^2}$$

, μ is the mass of the particle, while ω is the oscillator frequency. The Hamiltonian of this system is given by,

$$\hat{H} = -\frac{\hbar}{2m}\nabla^2 + V(r), \quad \nabla^2 = \sum_{i=1}^{d} \frac{\partial^2}{\partial x_i^2}. \tag{A.2}$$

The TDSE for this problem,

$$\hat{H}\psi_p(r) = E_p\psi_p(r), \tag{A.3}$$

can be solved analytically for $d = 1, 2, 3$.

A.1 One dimension

The eigenstates and corresponding eigenenergies in one dimension are given by,

$$\psi_n(x) = \left(\frac{\mu\omega}{\pi\hbar}\right)^{1/4} \frac{1}{\sqrt{2^n n!}} H_n(ax) e^{-a^2x^2/2}, \tag{A.4}$$

$$E_n = \left(n + \frac{1}{2}\right)\hbar\omega, \tag{A.5}$$

where $n = 0, 1, 2, \cdots$ and we have defined

$$a \equiv \sqrt{\frac{\mu\omega}{\hbar}}. \tag{A.6}$$

Here, $H_n(y)$ are the so-called Hermite polynomials given by,

$$H_n(y) = (-1)^n e^{y^2} \left(\frac{d}{dy}\right)^n e^{-y^2}. \tag{A.7}$$

A.2 Two dimensions

In two spatial dimensions the eigenfunctions of the harmonic oscillator in polar coordinates are given by,

$$\psi_{nm}(r,\theta) = N_{nm} a e^{im\theta} (ar)^{|m|} L_n^{|m|}\left(a^2r^2\right) e^{-a^2r^2/2}, \tag{A.8}$$

$$E_{nm} = \hbar\omega\left(2n + |m| + 1\right), \tag{A.9}$$

where $n = 0, 1, 2, \cdots$ and $m = 0, \pm1, \pm2, \cdots$. N_{nm} is a normalization constant given by,

$$N_{nm} = \sqrt{\frac{n!}{\pi(n + |m|)!}}, \tag{A.10}$$

while $L_p^q(y)$ are the so-called associated Laguerre polynomials, which for arbitary real q and $p \geq 0$ is given by,

$$L_p^q(y) = y^{-q} \frac{(d/dy - 1)^p}{p!} y^{p+q}. \tag{A.11}$$

A.3 Three dimensions

In three dimensions the eigenfunctions of the harmonic oscillator in spherical coordinates are given by,

$$\psi_{klm}(r,\theta,\varphi) = N_{kl} r^l e^{-a^2r^2/2} L_k^{(l+\frac{1}{2})}(a^2r^2) Y_l^m(\theta,\varphi), \tag{A.12}$$

$$E_{kl} = \hbar\omega(2k + l + \frac{3}{2}), \tag{A.13}$$

where $k, l = 0, 1, 2, \cdots$ and $-l \leq m \leq l$. The normalization constant N_{kl} is given by,

$$N_{kl} = \sqrt{\sqrt{\frac{2\nu^3}{\pi}} \frac{2^{k+2l+3} k! \nu^l}{(2k+2l+1)!!}}, \quad \nu \equiv \frac{\mu\omega}{2\hbar}. \tag{A.14}$$

$L_p^q(y)$ is the associated Laguerre polynomials while Y_l^m is a spherical harmonic funtion,

$$Y_l^m(\theta,\varphi) = N e^{im\varphi} P_l^m(\cos\theta). \tag{A.15}$$

N is a normalization constant and $P_l^m(y)$ is the associated Legendre function,

$$P_l^m(y) = (1-y^2)^{|m|/2} \left(\frac{d}{dy}\right)^{|m|} P_l(y), \tag{A.16}$$

where $P_l(z)$ is the lth Legendre polynomial,

$$P_l(z) = \frac{1}{2^l l!} \left(\frac{d}{dz}\right)^l (z^2-1)^l \tag{A.17}$$

Appendix B

Numerical Integration

B.1 The Runge-Kutta 4 method

Consider the initial value problem:

$$\frac{dy}{dt} = f(t, y) \quad y(t_0) = y_0. \tag{B.1}$$

Pick a step-size $\Delta t > 0$ and define

$$\begin{aligned} y_{n+1} &= y_n + \frac{\Delta t}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) \\ t_{n+1} &= t_n + \Delta t \end{aligned}$$

for $n = 0, 1, 2, \cdots$, where

$$\begin{aligned} k_1 &= f(t_n, y_n) \\ k_2 &= f\left(t_n + \frac{\Delta t}{2}, y_n + \frac{\Delta t}{2} k_1\right) \\ k_3 &= f\left(t_n + \frac{\Delta t}{2}, y_n + \frac{\Delta t}{2} k_2\right) \\ k_4 &= f\left(t_n + \Delta t, y_n + \Delta t k_3\right). \end{aligned}$$

y_{n+1} is the Runge-Kutta 4 [32] (RK4) approximation of $y(t_{n+1})$. The local truncation error of the RK4 method is on the order of $\mathcal{O}(\Delta t^5)$ and the total accumulated error is on the order of $\mathcal{O}(\Delta t^4)$.

Appendix C

Code listings

C.1 Restricted Hartree Fock class

Suppose that we have the one- and two-body integrals h_q^p and $u_{rs}^{pq} = \langle pq|\hat{u}|rs\rangle_{\text{AS}}$ available, then a complete RHF program which at convergence returns the RHF energy and the matrix of expansion coefficients U can be written in Python as:

Restricted Hartree Fock Program

```
import numpy as np
class RestrictedHartreeFock:
  def __init__(self,N,L,h,u):

    #N is the number of particles
    #L is the number of spatial orbitals
    #h is the LxL array of one-body integrals <p|h|q>
    #u is the LxLxLxL array of anti-symmetrized
    #two-body integrals <pq|u|rs>

    self.nrOfParticles        = N
    self.nrOfSpatialOrbitals  = L
    self.U = np.eye(L)
    self.D = np.zeros((L,L))
    self.F = np.zeros((L,L))
    self.h = h.copy()
    self.u = u.copy()
    self.o = slice(0, N/2) #range of occupied orbitals
```

```
    def compute_Eref(self):
      occ = self.o
      return 2*(np.einsum('ii'  ,self.h[occ,occ])
                +np.einsum('ijij',self.u[occ,occ,
                                         occ,occ]))

    def computeRHFEnergy(self,eps,D,U):
      occ = self.o
      return 2*sum(eps[occ])
             -np.einsum('qi,sr,qrps,pi->',U[:,occ],D,
                        self.u, U[:,occ])

    def compute_DensityMatrix(self,U):
      occ = self.o
      return 2*np.einsum('sj,rj->sr',
                         U[:,occ], U[:,occ])

    def computeFockMatrix(self,D):
      return self.h.copy()
             +np.einsum('sr,qrps->qp',D,self.u)

    def doSCF(self,max_iters = 20, delta=1E-8):
      #delta is the convergence threshold
      #and max_iters the maximum
      #number of SCF iterations.

      ERHF     = self.compute_Eref()
      eps_old  = np.zeros(self.nrOfSpatialOrbitals)
      diff_eps = 100
      iters    = 0

      while( (diff_eps > delta) and (iters < max_iters)):

        self.D       = self.compute_DensityMatrix(self.U)
        self.F       = self.computeFockMatrix(self.D)
        eps, self.U = np.linalg.eigh(self.F)
        ERHF           = self.computeHFEnergy(eps,self.D,
                                              self.U)
        if(iters > 0):
          diff_eps = max(abs(eps-eps_old))
          eps_old = eps.copy()
          iters += 1
```

```
        if(diff_eps < delta):
          return ERHF, iters, self.U
        else:
          print "Did not convergence to given precision"
          sys.exit(1)
```

Note that program depends only on the number of particles, basisfunctions and the integral elements. Thus, in principle the program can run for any (spin-restricted) system given h and u as input in some basis $\{\phi_i\}_{i=0}^{L}$.

C.2 Configuration Interaction class

Configuration Interaction Program

```
class ConfigurationInteraction:

  def __init__(self,Np,L,h,u,system=None):
    self.Np    = Np
    self.L     = L
    self.h     = h
    self.u     = u
    self.SlaterDeterminants = []
    self.diffByOne = []
    self.diffByTwo = []
    self.nSds = 0
    self.system = system

  def TDCI(self,A,dt=1e-3,T=1,sampleFrequency=50):

    Nt = int(T/dt)
    H1, H2 = self.computeHamiltonian(self.h,self.u)
    Hamiltonian = np.zeros((self.nSds, self.nSds),
                           dtype=np.complex128)
    At = A.copy()
    overlap    = np.zeros(Nt/sampleFrequency+1)
    E_vec      = np.zeros(Nt/sampleFrequency+1)
    overlap[0] = abs(np.vdot(At,A))**2
    t0 = time.time()
    counter = 0
```

```
      k1 = np.zeros(len(At))
      k2 = np.zeros(len(At))
      k3 = np.zeros(len(At))
      k4 = np.zeros(len(At))

      for i in range(0,Nt+1):
        t = i*dt
        #RK4 step1:
        ht = self.system.
             updateOneBodyElements(self.h,t=t)
        H1 = self.computeH1(ht)
        np.add(H1,H2,out=Hamiltonian)
        k1 = -1j*np.matmul(Hamiltonian,At)

        #RK4 step2:
        A_tmp = np.add(At,dt*0.5*k1)
        ht    = self.system.
                updateOneBodyElements(self.h,t=t+0.5*dt)
        H1    = self.computeH1(ht)
        np.add(H1,H2,out=Hamiltonian)
        k2    = -1j*np.matmul(Hamiltonian,A_tmp)

        #RK4 step3:
        A_tmp = np.add(At,dt*0.5*k2)
        ht = self.system.
             updateOneBodyElements(self.h,t=t+0.5*dt)
        H1 = self.computeH1(ht)
        np.add(H1,H2,out=Hamiltonian)
        k3 = -1j*np.matmul(Hamiltonian,A_tmp)

        #RK4 step4:
        A_tmp = np.add(At,dt*k3)
        ht = self.system.
             updateOneBodyElements(self.h,t=t+dt)
        H1 = self.computeH1(ht)
        np.add(H1,H2,out=Hamiltonian)
        k4 = -1j*np.matmul(Hamiltonian,A_tmp)


        if(i%sampleFrequency == 0):
          E_tn = np.vdot(At,np.matmul(Hamiltonian,At))/
                         np.vdot(At,At)
```

```
            E_vec[counter] = E_tn.real
            overlap[counter] = abs(np.vdot(At,A))**2
            rhoCI = self.computeOneBodyDensityMatrix(At)
            counter += 1


        At = At + (dt/6.0)*(k1+2.0*(k2+k3)+k4)
        return At, overlap, E_vec


    def createCISDspace(self,Nsps,CIspace="CISD"):

        Refstate = np.zeros(self.Np)
        for i in range(0,self.Np):
            Refstate[i] = i
        self.nSds += 1
        self.SlaterDeterminants.append(
        ca.createBinaryState(Refstate,self.L))

        if(CIspace == "CISD"):
            for i in range(0,self.Np):
                for a in range(self.Np,Nsps):
                    #All singles excitations
                    state = Refstate.copy()
                    state[i] = a
                    self.SlaterDeterminants.append(
                    ca.createBinaryState(state,self.L))
                    self.nSds += 1
                    for j in range(i+1,self.Np):
                        for b in range(a+1,Nsps):
                            #All doubles excitations
                            state = Refstate.copy()
                            state[i] = a
                            state[j] = b
                            self.SlaterDeterminants.append(
                            ca.createBinaryState(state,self.L))
                            self.nSds += 1


        elif(CIspace == "CID"):
            for i in range(0,self.Np):
                for a in range(self.Np,Nsps):
```

```
            for j in range(i+1,self.Np):
              for b in range(a+1,Nsps):
                state = Refstate.copy()
                state[i] = a
                state[j] = b
                self.SlaterDeterminants.append(
                ca.createBinaryState(state,self.L))
                self.nSds += 1
      elif(CIspace == "CIS"):
        for i in range(0,self.Np):
          for a in range(self.Np,Nsps):
            state = Refstate.copy()
            state[i] = a
            self.SlaterDeterminants.append(
            ca.createBinaryState(state,self.L))
            self.nSds += 1
      else:
        print "Provide one of the following CI-spaces:
               CIS, CID or CISD"
        sys.exit(1)

      def sortCIspace():
        for I in range(0,self.nSds):
          for J in range(I+1,self.nSds):
            sI = self.SlaterDeterminants[I]
            sJ = self.SlaterDeterminants[J]
            diff, id_s1, id_s2 = ca.diffState(sI,sJ)

            if(diff == 1):
              tmp = sJ.copy()
              s1 = ca.sign(id_s2[0],tmp)
              tmp[id_s2] = False
              s2 = ca.sign(id_s1[0],tmp)
              tmp[id_s1] = True
              sign = s1*s2
              lst = [I,J,id_s1[0],id_s2[0],sign]
              self.diffByOne.append(lst)

            if(diff == 2):
              tmp = sJ.copy()
              s1 = ca.sign(id_s2[0],tmp)
              tmp[id_s2[0]] = False
```

```
            s2 = ca.sign(id_s2[1],tmp)
            tmp[id_s2[1]] = False
            s3 = ca.sign(id_s1[1],tmp)
            tmp[id_s1[1]] = True
            s4 = ca.sign(id_s1[0],tmp)
            sign = s1*s2*s3*s4
            lst = [I,J,id_s1[0],id_s1[1],
                   id_s2[0],id_s2[1],sign]
            self.diffByTwo.append(lst)

    def computeGroundState(self):
      H1, H2 = self.computeHamiltonian(self.h,self.u)
      Hamiltonian = H1+H2
      Energies, A = np.linalg.eigh(Hamiltonian)
      return Energies, A

    def computeHamiltonian(self,h,u):
      H1= np.zeros((self.nSds,self.nSds),
                   dtype=np.complex128)
      H2= np.zeros((self.nSds,self.nSds),
                   dtype=np.complex128)

      for I in range(0,self.nSds):
        sI  = self.SlaterDeterminants[I]
        idI = np.where(sI)[0]
        for m in idI:
          H1[I,I] += h[int(m),int(m)]
          for k in idI:
            if(k != int(m)):
              H2[I,I] += 0.5*u[int(m),int(k),
                               int(m),int(k)]

      for K in self.diffByOne:
        I=K[0]
        J= K[1]
        sI= self.SlaterDeterminants[I]
        sJ= self.SlaterDeterminants[J]
        idI= np.where(sI)[0]
        id_s1= K[2]
        id_s2= K[3]
        sign = K[4]
        H1[I,J] += h[id_s1,id_s2]*sign
```

```
        for q in idI:
          H2[I,J] += u[int(id_s1),int(q),
                       int(id_s2),int(q)]*sign

      for M in self.diffByTwo:
        I= M[0]
        J= M[1]
        id_s1= [M[2],M[3]]
        id_s2= [M[4],M[5]]
        sign =  M[6]
        H2[I,J] += u[int(id_s1[0]),int(id_s1[1]),
                     int(id_s2[0]),int(id_s2[1])]*sign
        for I in range(0,self.nSds):
          for J in range(I,self.nSds):
            if(I!=J):
              H1[J,I] = H1[I,J]
              H2[J,I] = H2[I,J]
      return H1,H2

  def computeH1(self,h):
    H1  = np.zeros((self.nSds,self.nSds),
                   dtype=complex)
    for I in range(0,self.nSds):
      sI = self.SlaterDeterminants[I]
      idI = np.where(sI)[0]
      for m in idI:
        H1[I,I] += h[int(m),int(m)]

    for K in self.diffByOne:
      I= K[0]
      J= K[1]
      id_s1= K[2]
      id_s2= K[3]
      sign = K[4]
      H1[I,J] += h[id_s1,id_s2]*sign

    for I in range(0,self.nSds):
      for J in range(I,self.nSds):
        if(I!=J):
          H1[J,I] = H1[I,J]
    return H1
```

```
  def computeOneBodyDensityMatrix(self,A):

    rho = np.zeros((self.L/2,self.L/2),dtype=complex)
    for p in range(0,self.L/2):
      rpp = 0
      for I in range(0,self.nSds):
        sI = self.SlaterDeterminants[I]
        if(sI[2*p] == True):
          rpp += np.conj(A[I])*A[I]
        if(sI[2*p+1] == True):
          rpp += np.conj(A[I])*A[I]
      rho[p,p] = rpp

    for K in self.diffByOne:
      sI       = self.SlaterDeterminants[K[0]]
      sJ       = self.SlaterDeterminants[K[1]]
      id_si    = K[2]
      id_sj    = K[3]
      tmp, s1  = ca.removeParticle(id_sj,sJ)
      tmp2, s2 = ca.addParticle(id_si,tmp)
      sign = s1*s2
      rho[id_si/2,id_sj/2] += np.conj(A[K[0]])
                                *A[K[1]]*sign

    for p in range(0,self.L/2):
      for q in range(p+1,self.L/2):
        rho[q,p] = rho[p,q]
    return rho
```

C.3 Coupled Cluster class

CreationAnnihilation

```
import numpy as np
import sys
import time
from memory_profiler import profile

class CoupledCluster:
  def __init__(self,N,L,h,u,system=None):
```

```
        self.N = N
        self.L = L
        self.o = slice(0, N)
        self.v = slice(N, L)
        self.h = h
        self.u = u
        self.system = system

        self.t2, self.l2 = self.initializeAmplitudes()

        Np = L-N
        self.Xklij = np.zeros((N,N,N,N),
                              dtype=np.complex128)
        self.Xbc = np.zeros((Np,Np),
                              dtype=np.complex128)
        self.Xlj = np.zeros((N,N),
                              dtype=np.complex128)
        self.Xkbcj = np.zeros((N,Np,Np,N),
                              dtype=np.complex128)
        self.Ejk     = np.zeros((N,N),
                              dtype=np.complex128)
        self.Ecb     = np.zeros((Np,Np),
                              dtype=np.complex128)
        self.Edcab   = np.zeros((Np,Np,Np,Np),
                              dtype=np.complex128)
        self.Eijkl   = np.zeros((N,N,N,N),
                              dtype=np.complex128)
        self.Ecjkb   = np.zeros((Np,N,N,Np),
                              dtype=np.complex128)
        self.Ecijkla = np.zeros((Np,N,N,N,N,Np),
                              dtype=np.complex128)
        self.Edcikab = np.zeros((Np,Np,N,N,Np,Np),
                              dtype=np.complex128)


        def initializeAmplitudes(self):
          N = self.N; L = self.L

          t2 = np.zeros((L-N,L-N,N,N),dtype=np.complex128)
          l2 = np.zeros((N,N,L-N,L-N),dtype=np.complex128)

          uNew = np.zeros((L,L,L,L),dtype=np.complex128)
```

```
        hNew = np.zeros((L,L),dtype=np.complex128)

        for p in range(0,L):
          for r in range(0,L):
            hNew[p,r] = self.h[p,r]
            for q in range(0,L):
              for s in range(0,L):
                uNew[p,r,q,s] = self.u[p,r,q,s]

        self.h = hNew.copy()
        self.u = uNew.copy()

        for i in range(0,N):
          for j in range(0,N):
            for a in range(N,L):
              for b in range(N,L):
                Dabij = self.h[i,i] + self.h[j,j]
                        -self.h[a,a] - self.h[b,b]
                t2[a-N,b-N,i,j] = self.u[a,b,i,j]/Dabij
                l2[i,j,a-N,b-N] = self.u[i,j,a,b]/Dabij
        return t2, l2

      def TDCCD(self,dt=1e-3,T=1,sampleFrequency=50):

        Nt = int(T/dt)
        h0 = self.h.copy()

        N = self.N; L = self.L
        k1 = np.zeros((L-N,L-N,N,N),dtype=np.complex128)
        k2 = np.zeros((L-N,L-N,N,N),dtype=np.complex128)
        k3 = np.zeros((L-N,L-N,N,N),dtype=np.complex128)
        k4 = np.zeros((L-N,L-N,N,N),dtype=np.complex128)

        m1 = np.zeros((N,N,L-N,L-N),dtype=np.complex128)
        m2 = np.zeros((N,N,L-N,L-N),dtype=np.complex128)
        m3 = np.zeros((N,N,L-N,L-N),dtype=np.complex128)
        m4 = np.zeros((N,N,L-N,L-N),dtype=np.complex128)

        E_vec   = np.zeros(Nt/sampleFrequency+1,
                  dtype=np.complex128)
        t_vec   = np.zeros(Nt/sampleFrequency+1)
        counter = 0
```

```
        for i in range(0,Nt+1):
          t = i*dt

          #RK4 step1:
          self.h = self.system.
                   updateOneBodyElements(h0,t=t)
          self.updateExcitationInterMediates  (self.t2)
          self.updateDeExcitationInterMediates(self.t2)

          if(i%sampleFrequency == 0):
            E_tn = self.compute_ETDCCD(self.l2,
                                       self.t2,t)
            E_vec[counter] = E_tn
            t_vec[counter] = t

          k1 = -1j*self.compute_dEdLijabH(self.t2,t=t)
          m1 =  1j*self.compute_dEtabijH
                (self.l2,self.t2,t=t)

          #RK4 step2:
          self.h = self.system.
                   updateOneBodyElements(h0,t=t+0.5*dt)
          t2_tmp = np.add(self.t2,0.5*dt*k1)
          self.updateExcitationInterMediates  (t2_tmp)
          self.updateDeExcitationInterMediates(t2_tmp)
          k2 = -1j*self.compute_dEdLijabH(t2_tmp,t=t)
          m2 =  1j*self.
                compute_dEtabijH
                (np.add(self.l2,0.5*dt*m1),
                        t2_tmp,t=t+0.5*dt)

          #RK4 step3:
          self.h = self.system.
                   updateOneBodyElements(h0,t=t+0.5*dt)
          t2_tmp = np.add(self.t2,0.5*dt*k2)
          self.updateExcitationInterMediates  (t2_tmp)
          self.updateDeExcitationInterMediates(t2_tmp)
          k3 = -1j*self.compute_dEdLijabH(t2_tmp,t=t)
          m3 =  1j*self.compute_dEtabijH
                (np.add(self.l2,0.5*dt*m2),
```

```
                        t2_tmp,t=t+0.5*dt)

        #RK4 step4:
        self.h = self.system.
                 updateOneBodyElements(h0,t=t+dt)
        t2_tmp = np.add(self.t2,dt*k3)
        self.updateExcitationInterMediates(t2_tmp)
        self.updateDeExcitationInterMediates(t2_tmp)
        k4 = -1j*self.compute_dEdLijabH(t2_tmp,t=t)
        m4 =  1j*self.compute_dEtabijH
              (np.add(self.l2,dt*m3),
                        t2_tmp,t=t+dt)

        self.t2 = self.t2
                  +(dt/6.0)*(k1+2.0*(k2+k3)+k4)
        self.l2 = self.l2
                  +(dt/6.0)*(m1+2.0*(m2+m3)+m4)

        return E_vec, t_vec, overlap

    def updateExcitationInterMediates(self,t2):
      self.build_Xklij(t2)
      self.build_Xbc(t2)
      self.build_Xlj(t2)
      self.build_Xkbcj(t2)

    def updateDeExcitationInterMediates(self,t2):
      self.build_Ejk(t2)
      self.build_Ecb(t2)
      self.build_Edcab(t2)
      self.build_Eijkl(t2)
      self.build_Ecjkb(t2)
      self.build_Ecijkla(t2)
      self.build_Edcikab(t2)

    def computeGroundState(self):
      ECC = self.computeTamplitudes()
      self.computeLamplitudes()
      return ECC.real

    def computeTamplitudes(self,max_iters,eps,alpha):
```

```
      self.updateExcitationInterMediates(self.t2)
      N = self.N; L = self.L

      Eref  =  self.compute_Eref()

      Ecorr =  self.compute_Ecorr(self.t2)
      converged = False

      diff  = 100
      iters = 1
      t_tot = 0

      while(diff > eps and iters < max_iters):


        self.updateExcitationInterMediates(self.t2)
        t2_new = self.compute_dEdLijabH(self.t2)
        t_tot += t1-t0

        for i in range(0,N):
          for j in range(0,N):
            for a in range(N,L):
              for b in range(N,L):
                Dabij = self.h[i,i] + self.h[j,j]
                      - self.h[a,a] - self.h[b,b]
                t2_new[a-N,b-N,i,j] /= Dabij

        Ecorr_new   = self.compute_Ecorr(t2_new)
        diff = abs(Ecorr_new-Ecorr)

        iters += 1
        self.t2    = alpha*t2_new + (1-alpha)*self.t2
        Ecorr      = Ecorr_new
        print ("%.4f" % (Eref+Ecorr).real)


        if(diff < eps):
          converged = True
        else:
          print (diff)
          print ("T-amplitudes did not converge.")
          print ("** Exit program **")
```

```
            sys.exit(1)
        return Eref+Ecorr


    def computeLamplitudes(self,max_iters,eps,alpha):

      N = self.N; L = self.L
      max_rhs = 100
      iters = 0
      converged = False
      t_tot = 0

      while(max_rhs > eps and iters < max_iters):


        self.updateDeExcitationInterMediates(self.t2)
        l2_new = self.compute_dEtabijH(self.l2,self.t2)

        for i in range(0,N):
          for j in range(0,N):
            for a in range(N,L):
              for b in range(N,L):
                  Dabij = self.h[i,i] + self.h[j,j]
                        - self.h[a,a] - self.h[b,b]
                  l2_new[i,j,a-N,b-N] /= Dabij

        iters += 1

        diff = np.amax(abs(l2_new - self.l2))
        RHS  = self.compute_dEtabijH(l2_new,self.t2)

        for i in range(0,N):
          for j in range(0,N):
            for a in range(N,L):
              for b in range(N,L):
                Dabij = self.h[i,i] + self.h[j,j]
                      - self.h[a,a] - self.h[b,b]
                RHS[i,j,a-N,b-N] -= l2_new[i,j,
                                           a-N,b-N]
                                    *Dabij

        self.l2 = alpha*l2_new + (1-alpha)*self.l2
```

```
            max_rhs = np.amax(abs(RHS))


        if max_rhs < eps:
            converged = True
        else:
            print max_rhs
            print ("L-amplitudes did not converge.")
            print ("** Exit program **")
            sys.exit(1)


    def compute_Eref(self):
        o = self.o; v = self.v
        Eref  = np.einsum('ii',self.h[o,o])
        Eref += 0.5*np.einsum('ijij',self.u[o,o,o,o])
        return Eref

    def compute_Ecorr(self,t2):
        o = self.o; v = self.v
        Ecorr = 0.25*np.einsum('ijab,abij->',
                               self.u[o,o,v,v],t2)
        return Ecorr

    def compute_ETDCCD(self,l2,t2,t):
        Ecorr_t = 0
        if(t > 0):
            RHS = self.compute_dEdLijabH(self.t2,t=t)
            Ecorr_t = 0.25*np.einsum('ijab,abij->',l2,RHS)
        return self.compute_Eref()
             + self.compute_Ecorr(t2)+Ecorr_t


    def compute_dEdLijabH(self,t2,t=0):
        o = self.o; v = self.v
        #dEdLijabH1

        if(t==0):
            one_dki = np.ones((self.N,self.N))
                    - np.eye(self.N)
            Pij     = np.einsum('ki,ki,abjk->abij',
                                one_dki,self.h[o,o],t2)
```

```
        t2_new  = Pij - Pij.swapaxes(2,3)
        one_dca = np.ones((self.L-self.N,self.L-self.N))
                - np.eye(self.L-self.N)
        Pab     = -np.einsum('ac,ac,bcij->abij',
                             one_dca,self.h[v,v],t2)
        t2_new += Pab - Pab.swapaxes(0,1)
      else:
        Pij     = np.einsum('ki,abjk->abij',
                            self.h[o,o],t2)
        t2_new  = Pij - Pij.swapaxes(2,3)
        Pab     = -np.einsum('ac,bcij->abij',
                             self.h[v,v],t2)
        t2_new += Pab - Pab.swapaxes(0,1)

      #dEdLijabH2
      Pab = np.einsum('acij,bc->abij',t2,self.Xbc)
      t2_new += Pab - Pab.swapaxes(0,1)

      Pij = -np.einsum('abik,kljl->abij',t2,
                       self.u[o,o,o,o])
      t2_new +=  Pij - Pij.swapaxes(2,3)

      t2_new += np.einsum('abkl,klij->abij',t2,
                          self.Xklij)

      Pijab = np.einsum('acik,kbcj->abij',t2,self.Xkbcj)
      t2_new +=  Pijab - Pijab.swapaxes(2,3)
               - Pijab.swapaxes(0,1)
               + Pijab.swapaxes(2,3).swapaxes(0,1)

      np.add(t2_new, 0.5*np.einsum('dcij,abdc->abij',t2,
             self.u[v,v,v,v]),out=t2_new)

      Pij = np.einsum('abil,lj->abij',t2,self.Xlj)
      np.add(t2_new,(Pij - Pij.swapaxes(2,3)),
             out=t2_new)

      #t2_new += self.u[v,v,o,o].copy()
      np.add(t2_new,self.u[v,v,o,o],out=t2_new)
      return t2_new

      def compute_dEtabijH(self,l2,t2,t=0):
```

```
        o = self.o; v = self.v
        #H1 part
        Pab     = -np.einsum('ca,ijbc->ijab',
                             self.h[v,v],l2)
        l2_new =  Pab - Pab.swapaxes(2,3)

        Pij      = np.einsum('ik,jkab->ijab',
                             self.h[o,o],l2)
        l2_new += Pij - Pij.swapaxes(0,1)

        if(t==0):
          Pab     =  -np.einsum('aa,ijba->ijab',
                                self.h[v,v],l2)
          l2_new += -Pab + Pab.swapaxes(2,3)

          Pij      = np.einsum('ii,jiab->ijab',
                               self.h[o,o],l2)
          l2_new += -Pij + Pij.swapaxes(0,1)

        #H2 part
        np.add(l2_new,self.u[o,o,v,v],out=l2_new)

        Pij      = np.einsum('jkdc,dcikab->ijab',l2,
                             self.Edcikab)
        l2_new += Pij - Pij.swapaxes(0,1)

        Pab      = np.einsum('klbc,cijkla->ijab',l2,
                             self.Ecijkla)
        l2_new += Pab - Pab.swapaxes(2,3)

        l2_new += np.einsum('klab,ijkl->ijab',l2,
                             self.Eijkl)

        Pabij    = np.einsum('ikac,cjkb->ijab',l2,
                             self.Ecjkb)
        l2_new += Pabij - Pabij.swapaxes(0,1)
                - Pabij.swapaxes(2,3)
                + Pabij.swapaxes(2,3).swapaxes(0,1)

        Pij = np.einsum('ikab,jk->ijab',l2,self.Ejk)
        l2_new += Pij - Pij.swapaxes(0,1)
```

```
        Pab = np.einsum('jiac,cb->ijab',l2,self.Ecb)
        l2_new += Pab - Pab.swapaxes(2,3)

        l2_new += np.einsum('jidc,dcab->ijab',l2,
                            self.Edcab)
        return l2_new

    #Excitation intermediates
    def build_Xklij(self,t2):
      o = self.o; v = self.v
      np.add(0.5*self.u[o,o,o,o],0.25
     *np.tensordot(t2,self.u[o,o,v,v],
                   axes=((0,1),(2,3))),out=self.Xklij)

    def build_Xbc(self,t2):
      o = self.o; v = self.v
      np.add(np.einsum('bkck->bc',self.u[v,o,v,o]),
         0.5*np.einsum('bdkl,kldc->bc',t2,
                       self.u[o,o,v,v]),out=self.Xbc)

    def build_Xlj(self,t2):
      o = self.o; v = self.v
      np.einsum('dcjk,kldc->lj',0.5*t2,
                self.u[o,o,v,v],out=self.Xlj)


    def build_Xkbcj(self,t2):
      o = self.o; v = self.v
      np.add(self.u[o,v,v,o],0.5
     *np.einsum('bdjl,klcd->kbcj',t2,
                self.u[o,o,v,v]),out=self.Xkbcj)

    #De-excitation intermediates
    def build_Eijkl(self,t2):
      o = self.o; v = self.v
      np.add(0.5*self.u[o,o,o,o],
             0.25*np.einsum('dckl,ijdc->ijkl',t2,
                            self.u[o,o,v,v]),
                            out=self.Eijkl)

    def build_Ecjkb(self,t2):
      o = self.o; v = self.v
```

```
      np.subtract(self.u[v,o,o,v],
      np.einsum('dckl,jlbd->cjkb',t2,
                self.u[o,o,v,v]),out=self.Ecjkb)

    def build_Ejk(self,t2):
      o = self.o; v = self.v
      np.add(-0.5*np.einsum('dckl,jldc->jk',t2,
                            self.u[o,o,v,v]),
                -np.einsum('jlkl->jk',
                            self.u[o,o,o,o]),
                            out=self.Ejk)

    def build_Edcab(self,t2):
      o = self.o; v = self.v
      np.add(-0.5*self.u[v,v,v,v],
             -0.25*np.einsum('dckl,klab->dcab',t2,
                             self.u[o,o,v,v]),
                             out=self.Edcab)

    def build_Ecb(self,t2):
      o = self.o; v = self.v
      np.add(-np.einsum('ckbk->cb',self.u[v,o,v,o]),
             -0.5*np.einsum('dckl,klbd->cb',t2,
                            self.u[o,o,v,v]),
                            out=self.Ecb)

    def build_Ecijkla(self,t2):
      o = self.o; v = self.v
      np.einsum('dckl,ijad->cijkla',-0.5*t2,
                self.u[o,o,v,v],out=self.Ecijkla)

    def build_Edcikab(self,t2):
      o = self.o; v = self.v
      np.einsum('dckl,ilab->dcikab',0.5*t2,
                self.u[o,o,v,v],out=self.Edcikab)

    def compute_spinReducedOneBodyDensityMatrix(self):
      rho = self.compute_OneBodyDensityMatrix(self.l2,
                                              self.t2)
      L_half  = int(self.L/2)
      rho_red = np.zeros((L_half,L_half),
                         dtype=np.complex128)
```

```
      for p in range(0,L_half):
        for q in range(0,L_half):
          rho_red[p,q] = rho[2*p,2*q]+rho[2*p+1,2*q+1]
      return rho_red

    def compute_OneBodyDensityMatrix(self,l2,t2):
      o = self.o; v = self.v
      L = self.L; N = self.N
      rho_ji = np.eye(N)-0.5*np.einsum('kjab,abki->ji',
                                       l2,t2)
      rho_ba = 0.5*np.einsum('ijac,bcij->ba',l2,t2)
      rho = np.zeros((L,L),dtype=np.complex128)
      rho[o,o] = rho_ji
      rho[v,v] = rho_ba
      return rho

    def compute_TwoBodyDensityMatrix(self,l2,t2):
      o = self.o; v = self.v
      L = self.L; N = self.N
      delta_oo = np.eye(N)
      Pijkl    = -0.5*np.einsum('jl,kmdc,dcim->klij',
                                delta_oo,l2,t2)
      rho_klij = Pijkl - Pijkl.swapaxes(0,1)
               - Pijkl.swapaxes(2,3)
               + Pijkl.swapaxes(0,1).swapaxes(2,3)
      Pij       = -np.einsum('il,jk->klij',
                             delta_oo,delta_oo)
      rho_klij += Pij - Pij.swapaxes(2,3)
      rho_klij += 0.5*np.einsum('lkdc,dcji->klij',l2,t2)

      rho_jbia = 0.5*np.einsum('ij,klac,bckl->jbia',
                               delta_oo,l2,t2)
                  - np.einsum('jkac,bcik->jbia',l2,t2)

      rho_cdab = 0.5*np.einsum('ijab,cdij->cdab',l2,t2)
      rho_ijab = l2.copy()

      rho_abij  = t2.copy()
      rho_abij += -0.25*np.einsum('kldc,abkl,dcji->abij'
                                  ,l2,t2,t2)
      Pabij = -0.25*np.einsum('kldc,abik,dcjl->abij'
                              ,l2,t2,t2)
```

```
    rho_abij += Pabij - Pabij.swapaxes(0,1)
              - Pabij.swapaxes(2,3)
              + Pabij.swapaxes(0,1).swapaxes(2,3)
    Pij = -0.25*np.einsum('kldc,ackl,bdji->abij',
                          l2,t2,t2)
    rho_abij += Pij - Pij.swapaxes(2,3)
    Pij = np.einsum('kldc,acil,bdjk->abij',
                    l2,t2,t2)
    rho_abij += Pij - Pij.swapaxes(2,3)
    Pij = -0.25*np.einsum('kldc,acji,bdkl->abij',
                          l2,t2,t2)
    rho_abij += Pij - Pij.swapaxes(2,3)

    rho = np.zeros((L,L,L,L),dtype=np.complex128)

    rho[o,o,o,o] = rho_klij
    rho[o,o,v,v] = rho_ijab

    rho[o,v,o,v] = rho_jbia
    rho[v,o,o,v] =-rho_jbia.transpose((1,0,2,3))
    rho[o,v,v,o] =-rho_jbia.transpose((0,1,3,2))
    rho[v,o,v,o] = rho_jbia.transpose((1,0,3,2))

    rho[v,v,v,v] = rho_cdab
    rho[v,v,o,o] = rho_abij

    return rho
```

C.4 CreationAnnihilation module

CreationAnnihilation

```
import numpy as np

def createBinaryState(state,L):
        binState = np.zeros(L+1,dtype=np.bool)
        for i in range(0,len(state)):
                binState[int(state[i])] = True
        return binState

def sign(p,binState):
        k  = np.count_nonzero(binState[0:p])
```

```
        sign = (-1)**k
        return sign

def addParticle(p,binState):

        phase    = sign(p,binState)
        newState = binState.copy()

        if newState[p] == False:
                newState[p] = True
        else:
                newState[-1] = True

        return newState, phase

def removeParticle(p,binState):

        phase    = sign(p,binState)

        newState = binState.copy()

        if newState[p] == True:
                newState[p] = False
        else:
                newState[-1] = True

        return newState, phase

def oneBodyOperator(p,q,state1,state2):

        tmp, sign1 = removeParticle(q,state2)
        if(tmp[-1] == True):
                return 0

        tmp2, sign2 = addParticle(p,tmp)
        if(tmp2[-1] == True):
                return 0

        if(np.array_equal(tmp2,state1) == False):
                return 0

        return sign1*sign2
```

```
def twoBodyOperator(p,q,r,s,state1,state2):

        tmp, sign1 = removeParticle(r,state2)
        if(tmp[-1] == True):
                return 0

        tmp2, sign2 = removeParticle(s,tmp)
        if(tmp2[-1] == True):
                return 0

        tmp3, sign3 = addParticle(q,tmp2)
        if(tmp3[-1] == True):
                return 0

        tmp4, sign4 = addParticle(p,tmp3)
        if(tmp4[-1] == True):
                return 0

        if(np.array_equal(tmp4,state1) == False):
                return 0

        return sign1*sign2*sign3*sign4

def diffState(state1,state2):

        diff       = state1.astype(np.int)
                    -state2.astype(np.int)
        abs_diff   = np.sum(abs(diff))
        indices_s1 = np.where(diff > 0)[0]
        indices_s2 = np.where(diff < 0)[0]

        return abs_diff/2, indices_s1, indices_s2
```

Bibliography

[1] S. Kvaal, "Ab initio quantum dynamics using coupled-cluster," *The Journal of Chemical Physics*, vol. 136, p. 194109, 2012.

[2] A. Szabo and N. S. Ostlund, *Modern Quantum Chemistry: Introduction To Adavanced Electronic Structure Theory*. Macmillan Publishing Co., 1982.

[3] P. Hohenberg and W. Kohn, "Inhomogeneous electron gas," *Physical Review*, vol. 136, p. B864, 1964.

[4] W. Kohn and L. J. Sham, "Self-consistent equations including exchange and correlation effects," *Physical Review*, vol. 140, p. A1133, 1965.

[5] I. Shavitt and R. J. Bartlett, *Many-Body Methods In Chemistry and Physics*, ch. 10. Cambridge University Press, 2009.

[6] T. D. Crawford and H. F. Schaefer, *An Introduction to Coupled Cluster Theory for Computational Chemists*, p. 33. John Wiley & Sons, Inc., 2007.

[7] T. Helgaker, P. Jørgensen, and J. Olsen, *Molecular Electronic Structure Theory*, ch. 10.6.2, p. 460. Wiley, 2000.

[8] D. Hochstuhl, C. Hinz, and M. Bonitz, "Time-dependent multiconfiguration methods for the numerical simulation of photoionization processes of many-electron atoms," *The European Physical Journal Special Topics*, vol. 223, p. 177, 2014.

[9] J. Zanghellini, M. Kitzler, T. Brabec, and A. Scrinzi, "Testing the multiconfiguration time-dependent hartree–fock method," *Journal of Physics B: Atomic, Molecular and Optical Physics*, vol. 37, p. 763, 2004.

[10] G. E. Karniadakis and R. M. K. II, *Parallel Scientific Computing in C++ and MPI: A Seamless Approach to Parallel Algorithms and their Implementation*. Cambridge University Press, 2003.

[11] S. Kvaal, *Lecture Notes for Fys-Kjm4480/9480; Quantum Mechanics For Many-Particle Systems*. University of Oslo, 2015.

[12] D. J. Griffiths, *Introduction to Quantum Mechanics*. Pearson Prentice Hall, 2 ed., 2005.

[13] G. H. Golub and C. F. V. Loan, *Matrix Computations*. Johns Hopkins University Press, 3 ed., 1996.

[14] T. Helgaker, P. Jørgensen, and J. Olsen, *Molecular Electronic Structure Theory*. Wiley, 2000.

[15] T. D. Crawford and H. F. Schaefer, *An Introduction to Coupled Cluster Theory for Computational Chemists*, p. 48. John Wiley & Sons, Inc., 2007.

[16] T. D. Crawford and H. F. Schaefer, *An Introduction to Coupled Cluster Theory for Computational Chemists*, p. 49. John Wiley & Sons, Inc., 2007.

[17] M. Beck, A. Jäckle, G. Worth, and H.-D. Meyer, "The multiconfiguration time-dependent hartree (mctdh) method: a highly efficient algorithm for propagating wavepackets," *Physics Reports*, vol. 324, p. 1, 2000.

[18] S. Kvaal, "Variational formulations of the coupled-cluster method in quantum chemistry," *Molecular Physics*, vol. 111, p. 1100, 2013.

[19] D. Loss and D. P. DiVincenzo, "Quantum computation with quantum dots," *Physical Review A*, vol. 57, p. 120, 1998.

[20] Q. Su and J. H. Eberly, "Model atom for multiphoton physics," *Physical Review A*, vol. 44, p. 5997, 1991.

[21] S. B. Skattum, "Time evolution in quantum dots using the multiconfiguration time-dependent hartree-fock method," Master's thesis, University of Oslo, 2013.

[22] E. Anisimovas and A. Matulis, "Energy spectra of few-electron quantum dots," *Journal of Physics: Condensed Matter*, vol. 10, p. 601, 1998.

[23] T. Vorrath and R. Blümel, "Electronic structure of three-dimensional quantum dots," *The European Physical Journal B - Condensed Matter and Complex Systems*, vol. 32, p. 227, 2003.

[24] X. Liu, X. Wang, Z. Wen, and Y. Yuan, "On the convergence of the self-consistent field iteration in kohn–sham density functional theory," *SIAM Journal on Matrix Analysis and Applications*, vol. 35, p. 546, 2014.

[25] M. Hjorth-Jensen, M. P. Lombardo, and U. van Kolck, eds., *An Advanced Course in Computational Nuclear Physics, Bridging the Scales from Quarks to Neutron Stars*. Springer, 2017.

[26] M. P. Lohne, G. Hagen, M. Hjorth-Jensen, S. Kvaal, and F. Pederiva, "Ab initio computation of the energies of circular quantum dots," *Physical Review B*, vol. 84, 2011.

[27] M. Hjorth-Jensen, *Computational Physics, Lecture Notes Fall 2015*. University of Oslo, 2015.

[28] F. Yuan, S. J. Novario, N. M. Parzuchowski, S. Reimann, S. K. Bogner, and M. Hjorth-Jensen, "Addition and removal energies of circular quantum dots," *The Journal of Chemical Physics*, vol. 147, p. 164109, 2017.

[29] Z. Wang, Z. Tu, and F. Wang, "Equation-of-motion coupled-cluster theory for excitation energies of closed-shell systems with spin–orbit coupling," *Journal of Chemical Theory and Computation*, vol. 10, p. 5567, 2014. PMID: 26583239.

[30] J. Høgberget, "Quantum monte-carlo studies of generalized many-body systems," Master's thesis, University of Oslo, 2013.

[31] H. E. Kristiansen and M. Hjorth-Jensen, "Time evolution of two-dimensional quantum dot systems using coupled cluster theory." In preparation for the Journal of Chemical Physics.

[32] H. P. Langtangen, *A Primer on Scientific Programmin with Python*. Springer, 2011.