Quantum-mechanical systems in traps and Similarity Renormalization Group theory

by

Sarah Reimann

THESIS
for the degree of
MASTER OF SCIENCE

(Master in Computational Physics)

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

April 2013

Abstract

We apply the Similarity Renormalization Group (SRG) method to quantum dots, using the same methodology that has recently had great success in the nuclear physics community. The SRG method can be realized in two different ways: In free space with respect to the physical vacuum state, or in-medium using the particle-hole formalism of second quantization. We start with the free-space approach, which is computationally less effective but has the advantage that no truncation occurs. We analyse the ground state using two different generators, Wegner's canonical generator and a modified version of that one, and meet numerical problems by replacing the standard Coulomb by an effective interaction and the harmonic oscillator by a Hartree-Fock basis. Afterwards, we apply the recently evolved in-medium SRG approach to our electronic systems. Here we choose the IM-SRG(2) method, meaning that all operators are truncated on a two-body level. Again, we apply two different generators, Wegner's canonical generator and White's generator. We demonstrate that Wegner's generator leads to numerical instabilities and stiff equation systems, especially for systems with comparatively high correlations. Computations with White's generator, on the other hand, are shown to be much more efficient and require less CPU time, especially as the size of the basis is increased. This enables us to look at systems up to $N = 42$ particles. To analyse the capabilities of IM-SRG(2), we compare our results for the ground state energy with other *ab initio* many-body methods, including Hartree-Fock, Coupled Cluster, Diffusion Monte Carlo and Full Configuration Interaction. Finally, we use the IM-SRG(2) results to study the role of correlations in two-dimensional quantum dots.

Preface

The completion of this thesis does not only denote the end of my master studies in physics, but also of a time of getting in touch with quantum many-body physics for the first time. Since it was a completely new area for me, it required lots of reading and understanding in the beginning - but at the same time I loved the balance between theory and programming and I knew that it was that field I wanted to do my master's thesis in.
However, the completion of my project does also denote the end of something else, namely two years of getting to know Norway as a new country to live in. Having always been fascinated by the marvellous fjords and mountains of the Norwegian nature, living in Oslo gave me the opportunity to spend many weekends on cabin trips in untouched nature and skiing in open mountain areas. It were those experiences that gave me each time again lots of energy to spend on my studies. Having available high-performance computers for quantum physics during the week and on the other hand no electricity at all in the mountains during weekends, is no contradiction at all, but definitely an amazing complement.

Regarding my thesis, I want above all thank my supervisor Morten Hjorth-Jensen, who taught me amazingly much. The discussions with him were always extremely motivating and his enthusiasm for many-body physics swept over to me, such that I usually came out of his office with an extreme eagerness to continue working and plenty of new ideas. Furthermore, I appreciate the great amount of freedom he gave me by setting the framework for the thesis and always coming up with proposals, but at the same time letting me explore and include those aspects in my thesis that I was interested in.
Moreover, I want to thank my co-supervisor Scott Bogner for the really constructive and elucidating discussions we had at Michigan State University. Together with all the talks I had with Titus Morris regarding theory and implementation of the SRG method, this was an enormous productive period.
It must be mentioned that this thesis would not be what it is today, had it not been for the unique collaboration at my department. Having an office available, I spent very much time at university, and I want at this point thank all the people in the Computational Physics group for the great way we worked together. Particularly, I would like to mention Jørgen Høgberget, who I shared my office with from the beginning on, and whose fooling around always gave me the necessary cheering-up, even in times of desperate code-debugging. Apart from that, I want to mention Karl Leikanger, whom I had very constructive discussions with regarding Diffusion Monte Carlo. Moreover, Sigve Bøe Skattum deserves a special thank for always helping immediately with one or the other question and being a great companion at MSU.
Last but not least I want to to thank my mother who has always supported me, my grandma for the many letters filling my postbox at least twice a month, and my granddad for all the inspirations I got from him about life and universe.
To all of you who helped me during this time, I owe you my thanks. The things I learned during these two years, from quantum many-body physics over object-oriented programming up to making fire in Norwegian DNT cabins, will definitely enter into many future projects, with respect to academics and beyond.

Sarah Reimann
Oslo, April 2013

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

Understanding the behaviour of strongly confined electrons is of fundamental interest for solving many-body problems. Quantum dots, e.g. electrons confined in semiconducting heterostructures, are of particular interest since they exhibit, due to their small size, discrete quantum levels. Under these conditions, typical quantum phenomena like tunnelling, entanglement and magnetization can all be observed [1,2]. Since quantum dots can be manufactured and designed artificially, their quantum levels can be tuned to one's needs by changing for instance the external field, or the size and shape of the system. As a consequence, quantum dots provide a high level of control for the dynamics and correlation of the electrons, which makes them perfectly suited to study quantum effects in practice.
Since the ground state of circular dots shows similar shell structures and magic numbers as seen for atoms and nuclei [3], these systems give the opportunity to study electronic systems without the presence of a nucleus affecting the electrons.
Apart from their relevance for theoretical research in quantum physics, quantum dots offer a wide variety of applications: In particular, their electrical and optical properties make them attractive for the use in laser technology [4,5] and solar cells [6, 7], but they are also used in quantum computers [8] and medical imaging [9], to give some examples.
In order to properly understand the properties of quantum dots and make theoretical predictions to their behaviour in various applications, it is necessary to study features like ground state energies and correlation effects. Since apart from quantum dots consisting of only two electrons or with specific values of the external field, no analytical solutions exist [10], the development of appropriate few- and many-body methods is required.
Several *ab initio* methods have been applied to these systems, in particular variational and diffusion Monte Carlo [11–14], large-scale diagonalization [15–18], Coupled Cluster theory [13, 19, 20] and Density Functional Theory [21].
Exact diagonalization has the great advantage of being accurate within a given model space, but the size of the problem grows very rapidly with the number of electrons and basis functions, and the computational cost gets exceedingly large. In practice, up to eight electrons have been studied [18], and in her Master of Science thesis, V.K.B. Olsen performed some first calculations with twelve electrons, however, with a rather limited basis size [22]. With Diffusion Monte Carlo, closed-shell quantum dots up to $N = 20$ particles have been treated and in his Master of Science thesis, J. Høgberget is looking at even larger systems. Though, as the system size increases, the DMC computations start to get rather time-consuming. Moreover,

for large systems the validity of the method as exact benchmark solution gets questionable, since the error made by approximations like the fixed-node approximation depends on a good choice of the trial wave function [23]. Coupled Cluster calculations, on the other hand, allow studying larger systems with much less required CPU time. However, with increasing number of particles and correlations between them, the method faces problems to converge [24], apart from the fact that the electron correlation is only approximated, with the error depending on the specific method (CCSD,CCSDT,...) [25].

Additionally to the above approaches, another very promising first-principle method has recently been introduced [26,27]. This is the Similarity Renormalization Group (SRG) method, which drives the Hamiltonian to a band- or block-diagonal form using a continuous series of unitary transformations. Especially in nuclear theory, it has successfully been applied to study systems with different underlying potentials, and it has been used to analyse their binding energy and other observables [28–30]. Apart from the free-space approach, where the Hamiltonian is set up with respect to a zero vacuum state, in recent times another interesting alternative has been worked out: Making use of the technique of normal-ordering, the SRG evolution can be applied to many-body problems with a new reference vacuum defined by occupied single-particle states. This approach is normally called in-medium SRG (IM-SRG) and makes numerical calculations much more efficient.

The aim of this thesis is to apply the same methodology that has been so successfully employed in nuclear physics to study the ground state of closed-shell systems of quantum dots in two dimensions. In particular, we focus on the different realizations, free-space and IM-SRG, both with different generators, and analyse the accuracy of the results, numerical challenges, as well as the computational cost of each specific method. Although this thesis focuses on systems of confined electrons, our code is written in such a general way that it can easily be extended to other systems, too.

Structure of the thesis

This thesis is structured the following way:

- The first part, chapters 2-5, presents the underlying theoretical models. In particular, chapters 2 and 3 introduce the basic concepts of quantum mechanics and many-body theory, focusing on those features that are relevant for the following parts of this work and introducing the used notation. Since an essential part of this thesis is to solve a set of coupled ordinary differential equations, chapter 4 explains the corresponding theoretical aspects and presents in particular the solution method which is used for our calculations. The last chapter of the first part, chapter 5, sets up the mathematical framework needed to deal with the systems modelled by us, namely quantum dots.

- The second part of this thesis, chapters 6-7, serves to explain the methods we use to study our systems. Chapter 6 introduces our main method, SRG. After exposing the general ideas and formalisms of the method, we discuss two different realizations, free-space and in-medium SRG. In particular, we present the full sets of equations and point out differences due to different generators. In chapter 7, we present the two other many-body methods implemented by us: the Hartree-Fock method, which precedes many of

our SRG calculations, and Diffusion Monte Carlo, which we use to benchmark our SRG results.

- In the third part of this thesis, which includes chapters 8-10, we present our implementation and results. Chapter 8 shows how we translated the different methods into source code, how our code is structured and we point out encountered problems and optimization techniques. Our numerical results are presented in chapter 9, where we analyse and discuss them in detail. In particular, we start with free-space SRG and make out computational challenges and methods to improve convergence. Then we continue with the in-medium calculations, making use of the results free-space SRG, benchmark the results against other many-body methods and use them to study the role of correlations in quantum dots. Chapter 10 concludes this thesis and provides suggestions for possibles extensions.

Part I

THEORY

Chapter 2

Quantum mechanical background

Quantum mechanics, also referred to as *quantum physics* in general, is a theory in physics which deals with the description of matter and its laws and properties. In contrast to classical physics, which deals with macroscopic systems, quantum mechanics allows the calculation of physical properties at typical length scales of 10^{-6} -10^{-7} m or smaller. Hence it is one of the main foundations of modern physics and forms the basis for atomic physics, condensed matter physics, nuclear physics and elementary particle physics, as well as related disciplines such as quantum chemistry.

For the quantum systems considered in this thesis, the theories and methods of quantum physics are needed, too, and for this reason, we want to explain the basic underlying concepts of quantum mechanics. This chapter deals with those basic aspects and especially introduces the notations we use in this thesis. Since quantum physics is a very large area, and even introductory text books often cover several hundred pages, we only focus on the most fundamental aspects that are relevant for the following parts of this work. Unless explicit references are stated, we base our explanations on [31–34].

2.1 Historical overview

In the 19th century, physics was based on what we nowadays refer to as 'classical physics':
The essential foundations were classical mechanics (following Newton), electrodynamics (following Maxwell) and thermodynamics (following Boltzmann). However, in the end of the 19th and particularly in the beginning of the 20th century, a number of experiments cast doubts on those former concepts, since the results could not be properly explained with the available theories.
In 1900, to derive his law of radiation, Max Planck made the hypothesis that an oscillator absorbs and emits energy only as multiples of an energy quantum

$$\Delta E = h\nu,$$

where h is Planck's constant and ν the oscillator frequency. In 1905, Albert Einstein went one step further and explained the photoelectric effect, stating that light consists of discrete particles of the same energy E. Further developments include the atom model by Rutherford

(1911), the quantum theory of spectra by Bohr (1913) and the scattering of photons, studied by Compton (1922).
Numerous experiments made in this period showed that light waves sometimes behave as if they were particles. In 1924, de Broglie finally proposed the reversal, namely that particles can exhibit wave characteristics, too. In particular, he suggested that each particle with momentum p corresponds to a wave with wave length λ and frequency ω, given by

$$\lambda = \frac{h}{p}, \qquad \omega = \frac{E}{\hbar}, \tag{2.1}$$

where $\hbar$ is the reduced Planck constant $\hbar = h/2\pi$. This hypothesis has been confirmed by several experiments, for instance the Davison-Germer experiment (1927), studying the reflection of electron beams on crystal surfaces.

Thus quantum mechanics had gradually come into the focus of scientists, and during the first half of the 20th century, further scientists, including Schrödinger, Hilbert and Dirac, helped to put the new observations and concepts into a mathematical framework.

2.2 Hilbert space and Dirac notation

All physical states we will consider in the following, lie in a complex vector space, which we refer to as *Hilbert space* $\mathcal{H}$, named after David Hilbert [35]. To be a Hilbert space, $\mathcal{H}$ must hold a positive-definite inner product and be complete with respect to its norm. The inner product $\langle\cdot|\cdot\rangle$ is a mapping

$$\langle\cdot|\cdot\rangle : \mathcal{H} \times \mathcal{H} \to \mathbb{C}$$

with the following properties:

1. The inner product is linear in the second argument,

$$\langle\psi|\alpha\chi_1 + \beta\chi_2\rangle = \alpha\langle\psi|\chi_1\rangle + \beta\langle\psi|\chi_2\rangle. \tag{2.2}$$

2. Forming the complex conjugate of the inner product gives

$$\langle\psi|\chi\rangle^* = \langle\chi|\psi\rangle. \tag{2.3}$$

   In particular, the inner product is anti-linear in the first argument,

$$\langle\alpha\chi_1 + \beta\chi_2|\psi\rangle = \alpha^*\langle\chi_1|\psi\rangle + \beta^*\langle\chi_2|\psi\rangle.$$

   Therefore an inner product is not a bilinear, but a *sesquilinear* form.

3. The inner product is positive definite,

$$\langle\psi|\psi\rangle \geq 0, \qquad \text{and} \qquad \langle\psi|\psi\rangle = 0 \Rightarrow \psi = 0. \tag{2.4}$$

Here ψ, χ are elements of $\mathcal{H}$ and $\alpha, \beta \in \mathbb{C}$. Each inner product defines a norm by

$$\|\psi\| = \sqrt{\langle\psi|\psi\rangle}. \tag{2.5}$$

A complex vector space $\mathcal{H}$ with an inner product is now called *Hilbert space*, if $\mathcal{H}$ is complete with respect to the norm (2.5). This means that each Cauchy series of vectors $\phi_n \in \mathcal{H}$ converges to an element in $\mathcal{H}$ [1],

$$\lim_{n\to\infty} \phi_n = \phi \in \mathcal{H}.$$

Loosely speaking, it means that $\mathcal{H}$ has enough restrictions such that calculations with vectors $\psi \in \mathcal{H}$ produce results also lying in $\mathcal{H}$. The easiest example of a Hilbert space is the n-dimensional complex vector space $\mathbb{C}^n$, with the inner product defined as

$$\left\langle \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \middle| \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \right\rangle = \sum_{i=1}^{n} x_i^* y_i. \tag{2.6}$$

To apply the concept of states to wave functions, which describe our quantum mechanical states and will be discussed in detail in section 2.4, we use the *bra-ket* notation developed by the physicist Paul Dirac [36]. It is named after splitting the word 'bracket' and is a standard notation for describing quantum states. Instead of dealing with functions ψ, one refers to ket-states $|\psi\rangle$ and their dual states $\langle\psi|$. For a finite-dimensional Hilbert space, the ket-state $|\psi\rangle$ can be viewed as a column vector,

$$\psi = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \end{bmatrix}$$

and its dual bra-state as the Hermitian transpose

$$\langle\psi| = [c_1^*, c_2^*, \dots] .$$

The connection between the bra- and ket-state is given by the inner product, in bra-ket notation compactly written as

$$\langle\psi_\alpha|\psi_\beta\rangle = \int dx\ \psi_\alpha^*(x)\psi_\beta^*(x). \tag{2.7}$$

With this definition of the inner product, we summarize some definitions which will be frequently used later on:

- A function $\psi \in \mathcal{H}$ is said to be *normalized* if the inner product with itself equals one,

$$\langle\psi|\psi\rangle = 1.$$

- Two functions $\psi, \chi \in \mathcal{H}$ are *orthogonal* if their inner product is zero,

$$\langle\psi|\chi\rangle = 0.$$

- A set of two or more functions is called *orthonormal* if each of the functions is normalized and each pair of functions is orthogonal.

[1] A Cauchy series ϕ_n is a series with the following property: For each $\epsilon > 0$, there exists an $N \in \mathbb{N}$, such that for all $n, m > N$, $\|\phi_n - \phi_m\| < \epsilon$. For more mathematical details about Cauchy series, we refer to standard textbooks in calculus.

Assuming a d-dimensional Hilbert space, a discrete orthonormal basis $\mathcal{B} = \{|\phi_i\rangle\}_{i=1}^{d}$ is given by a set of functions $\{\phi_1, \phi_2, \dots\}$ with orthonormality condition

$$\langle\phi_i|\phi_j\rangle = \delta_{ij} = \begin{cases} 0, & i \neq j \\ 1, & i = j. \end{cases} \tag{2.8}$$

Moreover, for the basis to be complete, it must fulfil the completeness relation

$$\sum_{i}^{d} |\phi_i\rangle\langle\phi_i| = \mathbf{1}. \tag{2.9}$$

That way, each function in $\mathcal{H}$ can be expressed as linear combination of the basis vectors,

$$|\Psi\rangle = \sum_{i=1}^{d} |\phi_i\rangle\langle\phi_i|\Psi\rangle = \sum_{i=1}^{d} c_i|\phi_i\rangle. \tag{2.10}$$

2.3 Observables and operators

In quantum physics, each physical observable A is associated with an operator $\hat{A}$, which acts on wave functions ψ to yield the expectation value of A:

$$\langle A\rangle = \int dx\ \psi^*(x)\hat{A}\psi(x). \tag{2.11}$$

Note that x and dx here for simplicity contain all degrees of freedom, such that integration is understood to be over all dimensions, not only one.
Since all measurements must yield real values, the operators must be *Hermitian* (or *self-adjoint*). This means

$$\hat{A} = \hat{A}^\dagger,$$

where $\hat{A}^\dagger$ is the Hermitian conjugate of $\hat{A}$, defined by

$$\langle\chi|\hat{A}\psi\rangle^* = \langle\hat{A}^\dagger\psi|\chi\rangle.$$

With these properties, the expectation value of an observable A can in bra-ket notation easily be expressed by

$$\langle A\rangle = \langle\psi|\hat{A}\psi\rangle = \langle\hat{A}\psi|\psi\rangle \equiv \langle\psi|\hat{A}|\psi\rangle. \tag{2.12}$$

Two fundamental examples of operators are the position operator in one dimension,

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

and the momentum operator

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

2.3.1 Commutation relations

At a later stage of this thesis, we will frequently encounter so-called *commutation relations* of operators. The point is that the order in which two operators $\hat{A}$ and $\hat{B}$ are applied to a function ψ, generally makes a difference, suggesting that

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

The commutator is defined as

$$[\hat{A}, \hat{B}] = \hat{A}\hat{B} - \hat{B}\hat{A}, \tag{2.13}$$

and has the properties

$$\begin{aligned}
[\hat{A}, \hat{B}] &= -[\hat{B}, \hat{A}], \\
[\hat{A}, a\hat{B}] &= [a\hat{A}, \hat{B}] = a[\hat{A}, \hat{B}], \qquad a \in \mathbb{C} \\
[\hat{A} + \hat{B}, \hat{C}] &= [\hat{A}, \hat{C}] + [\hat{B}, \hat{C}], \\
[\hat{A}\hat{B}, \hat{C}] &= \hat{A}[\hat{B}, \hat{C}] + [\hat{A}, \hat{C}]\hat{B},
\end{aligned}$$

all of which can easily be proved by applying definition (2.13). This list is not complete and summarizes just those relations that are most relevant for this thesis. For more properties, we refer to [31].
In the case that the order in which two operators act on a function ψ makes no difference, the two operators are said to *commute*, i.e.

$$[\hat{A}, \hat{B}] = \hat{A}\hat{B} - \hat{B}\hat{A} = 0.$$

As stated before, this is not the case in general, and even the well-known position and momentum operator do not commute, but follow the canonical commutation relation[2]

$$[\hat{x}, \hat{p_x}] = i\hbar.$$

2.3.2 Eigenvalues and eigenfunctions

If the action of an operator $\hat{A}$ on a function ψ yields the following relation,

$$\hat{A}\psi = a\psi, \tag{2.14}$$

then the constant a is called *eigenvalue* of $\hat{A}$ with corresponding eigenfunction ψ. Equation (2.14) is referred to as *eigenvalue equation*.
Eigenvalues and eigenfunctions have several useful properties, which we shortly summarize here following [32], which we also refer to for the corresponding proofs.[3]
Assuming that $\hat{A}$ is Hermitian, we have that:

- All eigenvalues are real.

[2] For a detailed derivation, we refer to [32].

[3] Note that we restrict us to discrete spectra, i.e. the eigenvalues are separated from each other. If the spectrum is continuous, the eigenfunctions are not normalizable and the first two properties do not hold. However, in this thesis we will only deal with discrete spectra.

- Eigenfunctions belonging to distinct eigenvalues are orthonormal.
- For any operator with a finite set of eigenfunctions, the eigenfunctions are complete and span the full Hilbert space $\mathcal{H}$. This makes it possible to express any arbitrary function in this space as linear combination of eigenfunctions,

$$\Psi = \sum_{i}^{d} c_i \psi_i,$$

where d is the dimension of $\mathcal{H}$. For infinite-dimensional Hilbert spaces, this property cannot be proven in general. However, since it is essential for the internal consistency of quantum mechanics, it is taken as restriction on operators representing observables.

2.4 Wave mechanics

In this section we discuss in more detail how quantum mechanical systems can be represented by functions Ψ, referred to as *wave functions*, and how the formalisms of the previous sections can be applied to describe the evolution of a system.

2.4.1 Properties of the wave function

According to de Broglie, each particle with momentum p is associated with a wave of wave length λ and frequency ω, as stated in Eq. (2.1), and we will denote this wave function with $\Psi(\mathbf{r}, t)$. Following Born's statistical interpretation, we understand the square $|\Psi(\mathbf{r}, t)|^2$ as the probability distribution for finding the particle at time t at position $\mathbf{r}$. More generally, the probability of finding a particle at a time t in a region $\Omega \subset \mathcal{H}$ is

$$P_\Omega(t) = \int_\Omega d\mathbf{r}\ \Psi^*(\mathbf{r}, t)\Psi(\mathbf{r}, t), \tag{2.15}$$

where Ω is a subspace of the full Hilbert space $\mathcal{H}$. In order for this interpretation to be correct, $\Psi(\mathbf{r}, t)$ must be normalized, suggesting that for all t

$$\int_{\mathcal{H}} d\mathbf{r}\ |\Psi(\mathbf{r}, t)|^2 = 1. \tag{2.16}$$

An alternative approach is to work with unnormalized wave functions and normalize the integrals themselves, which means to divide them by $\int_{\mathcal{H}} d\mathbf{r}\ |\Psi(\mathbf{r}, t)|^2$.

2.4.2 Time-dependent Schrödinger equation

To get a concrete expression for the wave function, let us first consider the easy case that our system only consists of one single particle. An easy constructable wave function with the above mentioned parameters, momentum p and wave length λ, is given by

$$\Psi(x, t) = \Psi(0, 0)e^{i\frac{2\pi x}{\lambda} - i\omega t}, \tag{2.17}$$

where $\Psi(0,0)$ is a constant determining the amplitude of the wave. Taking the derivative with respect to time and space, we obtain

$$\frac{\partial}{\partial x}\Psi(x,t) = i\frac{2\pi}{\lambda}\Psi(x,t) = i\frac{p}{h}\Psi(x,t) \tag{2.18}$$

$$\frac{\partial}{\partial t}\Psi(x,t) = -i\omega\Psi(x,t) = -i\frac{E}{h}\Psi(x,t). \tag{2.19}$$

In the non-relativistic limit, the energy of a free particle with momentum p and mass m is

$$E = \frac{p^2}{2m}.$$

Combining equations (2.18) and (2.19) with this relation yields

$$i\hbar\frac{\partial}{\partial t}\Psi(x,t) = E\Psi(x,t) = -\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2}\Psi(x,t). \tag{2.20}$$

If the particle is not free, but moving in an external potential $V(x)$, we have to add that contribution to the time evolution and obtain

$$i\hbar\frac{\partial}{\partial t}\Psi(x,t) = \left(-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + \hat{V}(x)\right)\Psi(x,t). \tag{2.21}$$

We use the notation $\hat{V}(x)$ to emphasize that the potential acts as an operator, possibly containing derivatives etc. Equation (2.21) is called *time-dependent Schrödinger equation*, which is one of the main foundations of quantum mechanics and regarded as the quantum-mechanical analogue to Newton's laws of motion. Generalized to three dimensions, it reads

$$i\hbar\frac{\partial}{\partial t}\Psi(\mathbf{r},t) = \left(-\frac{\hbar^2}{2m}\nabla^2 + \hat{V}(\mathbf{r},t)\right)\Psi(\mathbf{r},t). \tag{2.22}$$

Defining the Hamiltonian operator,

$$\hat{H} = \hat{T} + \hat{V}, \tag{2.23}$$

Schrödinger's equation can be simplified to

$$i\hbar\frac{\partial}{\partial t}\Psi(\mathbf{r},t) = \hat{H}\Psi(\mathbf{r},t). \tag{2.24}$$

Here $\hat{T}$ denotes kinetic energy operator,

$$\hat{T} = \frac{\hat{p}^2}{2m} = -\frac{\hbar^2}{2m}\nabla^2, \tag{2.25}$$

and $\hat{V}$, as before, is the operator of the potential energy. Hence the Hamiltonian (2.23) represents the total energy of the particle. For systems consisting of more than just one particle, the Hamiltonian is extended to correspond to the total energy of the system by including interaction energies etc.

2.4.3 Time-independent Schrödinger equation

To get a more specific expression for the wave function, let us assume that the potential V is time-independent, a reasonable first approach. In this case, Schrödinger's equation can be solved by the separation of variables, with the ansatz

$$\Psi(\mathbf{r}, t) = \psi(\mathbf{r})\chi(t), \tag{2.26}$$

which decouples space and time. To account for the case that multiple products are solutions, we extend our ansatz to

$$\Psi(\mathbf{r}, t) = \sum_n c_n \psi_n(\mathbf{r})\chi_n(t), \tag{2.27}$$

which is possible since any linear combination of solutions to Schrödinger's equation is a solution, too.
For each of the solutions, Schrödinger's equation now implies

$$i\hbar\psi_n(\mathbf{r})\frac{d}{dt}\chi_n(t) = \chi_n(t)\left(-\frac{\hbar^2}{2m}\nabla^2\psi_n(\mathbf{r}) + \hat{V}(\mathbf{r})\psi_n(\mathbf{r})\right). \tag{2.28}$$

Formally, we can divide Eq. (2.28) by $\psi_n(\mathbf{r})\chi_n(t)$, which yields

$$i\hbar\frac{1}{\chi_n}\frac{d\chi_n}{dt} = -\frac{\hbar^2}{2m}\frac{1}{\psi_n}\nabla^2\psi_n + \hat{V}\psi_n. \tag{2.29}$$

Note that we have dropped the t- and $\mathbf{r}$-dependence for better readability. We observe that the left side of Eq. (2.29) is a function depending only time t, whereas the right side depends only on space $\mathbf{r}$. The equation can only hold true if both expressions equal a constant, and we denote that one by E_n. That way, we have divided the time-dependent Schrödinger equation into two separate equations,

$$i\hbar\frac{d\chi_n}{dt} = E_n\chi_n, \tag{2.30}$$

$$\hat{H}\psi_n(\mathbf{r}) = E_n\psi_n(\mathbf{r}), \tag{2.31}$$

where $\hat{H}$ is the Hamiltonian operator of Eq. (2.23). The first equation can easily be solved, giving for the time-dependent part of the wave function:

$$\chi_n(t) = \exp\left(-i\frac{E_n}{\hbar}t\right). \tag{2.32}$$

The spatial part $\psi_n(\mathbf{r})$ can be obtained by solving Eq. (2.31), which is also called *time-independent Schrödinger equation*. Since the Hamiltonian operator represents the energy of the wave function, the constants E_n correspond to the energy eigenvalues of the functions ψ_n. The full, time-dependent Schrödinger equation (2.24) is now solved by

$$\Psi(\mathbf{r}, t) = \sum_n \psi_n(\mathbf{r})\exp\left(-i\frac{E_n}{\hbar}t\right). \tag{2.33}$$

2.5 The postulates of quantum mechanics

With the concepts and formalisms of the previous sections, the basics of quantum mechanics can be summarized in a few postulates. Depending on the author, they are presented in a slightly different manner, and we will here closely follow [31].

Postulate I To each well-defined observable A in physics, there exists an operator $\hat{A}$, such that measurements of A yield values a, which are eigenvalues of $\hat{A}$. In particular, the values a are those values for which the equation

$$\hat{A}\psi = a\psi$$

has solution ψ. The function ψ is called *eigenfunction* with *eigenvalue* a.

Postulate II Consider the set of eigenvalue equations

$$\hat{A}\psi_i = a_i\psi_i, \qquad i = 1, 2, \ldots$$

The operator $\hat{A}$ has different eigenvalues a_i with corresponding eigenfunctions ψ_i. If the measurement of observable A yields a value a_i, then the system is left in the state ψ_i, with the eigenfunction corresponding to eigenvalue a_i.

Postulate III At any instance of time, the state of a system may be represented by a wave function Ψ, which is continuous and differentiable, and contains all information regarding the state of the system. In particular, if the state of a system is described by a wave function $\Psi(\mathbf{r}, t)$, then the average of any physical observable A at time t is

$$\langle A \rangle = \int d\mathbf{r}\ \Psi^*(\mathbf{r}, t)\hat{A}\ \Psi(\mathbf{r}, t).$$

The average $\langle A \rangle$ is called the *expectation value* of $\hat{A}$.

Postulate IV The time development of a wave function $\Psi(\mathbf{r}, t)$ is given by the *time-dependent Schrödinger equation*

$$i\hbar\frac{\partial}{\partial t}\Psi(\mathbf{r}, t) = \left(-\frac{\hbar^2}{2m}\nabla^2 + \hat{V}(\mathbf{r}, t)\right)\Psi(\mathbf{r}, t).$$

2.6 Special case: Harmonic oscillator

One quantum-mechanical system of highest interest is the harmonic oscillator. Not only is it easy to obtain closed-form solutions, it allows also to demonstrate the concepts of the previous sections. Many complex problems can be reduced to harmonic oscillator problems and get thereby exactly solvable. Serving as basis for the Hamiltonian, the harmonic oscillator has an important role in this thesis, too, and we will therefore discuss it in more detail.

In classical mechanics, a harmonic oscillator is a system where a mass m experiences a restoring force F when displaced from its equilibrium position. The force is proportional to the displacement Δx and described by Hooke's law,

$$F = m\frac{d^2x}{dt^2} = -k\Delta x, \tag{2.34}$$

where $k > 0$ is the spring constant. Solving Eq. (2.34) for x yields the periodic function

$$x(t) = A\sin\omega t + B\cos\omega t, \tag{2.35}$$

where A and B are constants determined by the initial conditions, and the oscillator frequency ω describes the periodicity of the motion,

$$\omega = \sqrt{\frac{k}{m}}. \tag{2.36}$$

The potential energy can easily be obtained by integration,

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

where we assume $x = 0$ to be the equilibrium point.

For the quantum-mechanical analogue, we use Eqs. (2.23) and (2.25) combined with the oscillator potential (2.37), and obtain for one dimension

$$\hat{H} = -\frac{\hbar^2}{2m}\frac{d^2}{dx^2} + \frac{1}{2}m\omega^2x^2. \tag{2.38}$$

The time-independent Schrödinger equation (2.31) is then given by

$$\left(-\frac{\hbar^2}{2m}\frac{d^2}{dx^2} + \frac{1}{2}m\omega^2x^2\right)\psi_n = E_n\psi_n. \tag{2.39}$$

In the following, we will introduce two different approaches to solve this equation.

2.6.1 Conventional solution

The conventional approach is rather tedious and we will therefore only sketch the main steps. To make life a bit easier, we go over to dimensionless variables,

$$x \leftarrow \sqrt{\frac{m\omega}{\hbar}}x, \qquad \hat{p} \leftarrow -i\sqrt{m\hbar\omega}\frac{d}{dx}, \tag{2.40}$$

which simplify the eigenvalue problem to

$$\left(\frac{d^2}{dx^2} + \lambda - x^2\right)\psi_n = 0, \qquad \lambda = \frac{2E_n}{\hbar\omega}. \tag{2.41}$$

Since the leading term for $x \to \infty$ is

$$\left(\frac{d^2}{dx^2} - x^2\right)\psi_n = 0,$$

the wave function ψ_n must asymptotically approach

$$\psi_n \propto e^{-x^2/2}.$$

In this case, we have that $\frac{d}{dx}\psi_n = -x\psi_n$, suggesting that $\frac{d^2}{dx^2}\psi_n = -\psi_n + x^2\psi_n \approx x^2\psi_n$ in the limit $x \to \infty$.
For the wave function, we make the ansatz $\psi_n(x) = F(x)e^{-x^2/2}$ and get the following differential equation for the functions $F(x)$:

$$\left(\frac{d^2}{dx^2} - 2x\frac{d}{dx} + (\lambda - 1)\right) F(x) = 0. \tag{2.42}$$

For the solution, we use Fuchs' ansatz

$$F(x) = x^s \sum_{n\in\mathbb{N}} a_n x^n, \tag{2.43}$$

with $a_0 \neq 0$ and $s \geq 0$. After comparison of coefficients and some additional mathematical considerations[4], we get for each $n \in \mathbb{N}$ the differential equation

$$F''(x) - 2xF'(x) + 2nF(x).$$

This equation is solved by the Hermite polynomials $H_n(x)$, which fulfil the orthogonality relation

$$\int_{-\infty}^{\infty} H_m(x)H_n(x)e^{-x^2}dx = 2^n n!\sqrt{\pi}\delta_{nm}, \tag{2.44}$$

and the recurrence relation

$$H_{n+1}(x) = 2xH_n(x) - 2nH_{n-1}(x). \tag{2.45}$$

The first few polynomials are

$$\begin{aligned} H_0(x) &= 1, \\ H_1(x) &= 2x, \\ H_2(x) &= (2x)^2 - 2, \\ H_3(x) &= (2x)^3 - 6(2x). \end{aligned} \tag{2.46}$$

We have now found our solution ψ_n expanded in Hermite polynomials,

$$\psi_n(x) = N_n H_n(x)e^{-x^2/2}. \tag{2.47}$$

The corresponding eigenvalues are $\lambda_n = 2n + 1$, suggesting that

$$E_n = \hbar\omega\left(n + \frac{1}{2}\right), \tag{2.48}$$

and the normalization factors can easily be shown to be

$$N_0 = 1/\pi^{1/4}, \qquad N_n = N_0/\sqrt{2^n n!}.$$

The energies E_n represent the one-particle harmonic oscillator spectrum, are quantized and equally spaced, with spacing $\frac{1}{2}\hbar\omega$. Note that the lowest possible energy state is given by $E_0 = \frac{1}{2}\hbar\omega$ and not by 0, a result of vacuum fluctuations.

The solution method that we have shown here represents the standard approach for solving an eigenvalue problem like Eq. (2.39). However, additionally there exists a more elegant way, which is also of conceptual importance and will therefore be presented in the following.

[4] See [33] for details.

2.6.2 Elegant solution with ladder operators

The second solution approach is based on an operator technique with creation and annihilation operators.
We define the creation (rising) operator

$$a^\dagger = \sqrt{\frac{m\omega}{2\hbar}}\left(\hat{x} - \frac{i\hat{p}}{m\omega}\right) \tag{2.49}$$

and its Hermitian adjoint, the annihilation (lowering) operator[5]

$$a = \sqrt{\frac{m\omega}{2\hbar}}\left(\hat{x} + \frac{i\hat{p}}{m\omega}\right). \tag{2.50}$$

Moreover, we define the number operator

$$\hat{N} = a^\dagger a, \qquad \hat{N}|\psi\rangle = n|\psi\rangle, \tag{2.51}$$

where n is an integer eigenvalue, and obtain the commutation relations[6]

$$[a, a^\dagger] = 1, \qquad [\hat{N}, a^\dagger] = a^\dagger, \qquad [\hat{N}, a] = -a. \tag{2.52}$$

Reversing the latter operators yields

$$\hat{x} = \sqrt{\frac{\hbar}{2m\omega}}(a + a^\dagger), \tag{2.53}$$

$$\hat{p} = i\sqrt{\frac{\hbar m\omega}{2}}(a^\dagger - a). \tag{2.54}$$

Inserting this into our Hamiltonian (2.38), we get

$$\hat{H} = \hbar\omega\left(a^\dagger a + \frac{1}{2}\right) = \hbar\omega\left(\hat{N} + \frac{1}{2}\right). \tag{2.55}$$

That way, the eigenvalue problem $\hat{H}|\psi_n\rangle = E_n|\psi_n\rangle$ reduces to

$$\hat{N}|\psi_n\rangle = n|\psi_n\rangle,$$

suggesting that $\hat{H}$ and $\hat{N}$ have common eigenstates. Obviously, the eigenvalues are the same ones as in Eq. (2.48), $E_n = \hbar\omega\left(n + \frac{1}{2}\right)$. The states

$$a^\dagger|\psi_n\rangle, \qquad a|\psi_n\rangle$$

define new eigenvectors for $\hat{N}$ with eigenvalues $n+1$ and $n-1$, respectively:

$$\hat{N}a^\dagger|\psi_n\rangle = (a^\dagger\hat{N} + [\hat{N}, a^\dagger])|\psi_n\rangle = a^\dagger n|\psi_n\rangle + a^\dagger|\psi_n\rangle = (n+1)a^\dagger|\psi_n\rangle$$
$$\hat{N}a|\psi_n\rangle = (a\hat{N} + [\hat{N}, a])|\psi_n\rangle = an|\psi_n\rangle - a|\psi_n\rangle = (n-1)a|\psi_n\rangle,$$

[5] Depending on the author, the operators are often called *ladder* operators (explicitly *rising* and *lowering* operator) in connection with the representation theory of Lie algebras, whereas in quantum field and many-body theory, they are referred to as *creation* and *annihilation* operator, respectively. To be consistent with our next chapter, we use the latter terms.

[6] For the more or less straightforward proofs in this section, we refer to [32].

where we make use of the commutation relations (2.52). We observe that the operators $a^\dagger$ and a increase/decrease the eigenvalue n of eigenstates $|\psi_n\rangle$ by 1, which explains the terms *creation* and *annihilation* operator, respectively.
To stop the iterations, one defines for the lowest value $n = 0$

$$a|\psi_0\rangle = 0. \tag{2.56}$$

Starting from this one, all eigenstates $|\psi_n\rangle$ can be computed by applying the creation operator $a^\dagger$,

$$|\psi_n\rangle = \frac{\left(\hat{a}^\dagger\right)^n}{\sqrt{n!}}|\psi_0\rangle. \tag{2.57}$$

The lowest-lying state can be obtained by solving Eq. (2.56) explicitly:

$$\begin{aligned}
&\langle x|a|\psi_0\rangle = 0 \\
&\langle x|\sqrt{\frac{m\omega}{2\hbar}}\left(\hat{x} + \frac{i}{m\omega}\left(-i\hbar\frac{d}{dx}\right)\right)|\psi_0\rangle = 0 \\
&\Rightarrow \int \frac{d\psi_0(x)}{\psi_0(x)} = -\frac{m\omega}{\hbar}\int dx\; x \\
&\Rightarrow \psi_0(x) = Ne^{-\frac{m\omega}{2\hbar}x^2},
\end{aligned}$$

where we go over to coordinate representation $\langle x|\psi\rangle = \psi(x)$. Specifying N such that $\psi_0(x)$ is properly normalized, we obtain

$$\psi_0(x) = \left(\frac{m\omega}{\pi\hbar}\right)^{1/4} e^{-\frac{m\omega}{2\hbar}x^2}.$$

Applying Eq. (2.57), the solution for an arbitrary $n \in \mathbb{N}$ can be shown to be

$$\psi_n(x) = \langle x|\psi_n\rangle = \left(\sqrt{\frac{m\omega}{\pi\hbar}}\frac{1}{2^n n!}\right)^{1/2} H_n\left(\sqrt{\frac{m\omega}{\hbar}}x\right) e^{-\frac{m\omega x^2}{2\hbar}}, \tag{2.58}$$

where $H_n(x)$ are the previously defined Hermite polynomials. Comparing with Eq. (2.47), we note that we have gotten exactly the same expression, provided that we rewrite Eq. (2.47) from dimensionless units.

The here discussed method of creation and annihilation operators is of fundamental importance for further proceedings in quantum theory, as well as in pure mathematics. In quantum field theory, an expansion in creation and annihilation operators forms the foundation of percolation theory and is related to *second quantization*, a concept we will come back to in the next chapter.

2.6.3 The harmonic oscillator in $d > 1$ dimensions

For the harmonic oscillator potential, moving from $d = 1$ to higher dimensions is rather straightforward, since the Hamiltonian operator can be decomposed into a sum of contributions for each dimension. The general expression for the Hamiltonian is

$$\hat{H} = -\frac{\hbar^2}{2m}\nabla^2 + \frac{1}{2}m\omega^2 r^2, \tag{2.59}$$

which for $d = 2$ dimensions explicitly reads

$$\hat{H} = -\frac{\hbar^2}{2m}\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) + \frac{1}{2}m\omega^2(x^2 + y^2), \tag{2.60}$$

and for $d = 3$ dimensions

$$\hat{H} = -\frac{\hbar^2}{2m}\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}\right) + \frac{1}{2}m\omega^2(x^2 + y^2 + z^2). \tag{2.61}$$

For the example $d = 2$, which will be needed in this thesis, we rewrite the Hamiltonian as the following sum,

$$\begin{aligned}\hat{H} &= \hat{H}_x + \hat{H}_y \\ &= \left(-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + \frac{1}{2}m\omega^2 x^2\right) + \left(-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial y^2} + \frac{1}{2}m\omega^2 y^2\right),\end{aligned}$$

which enables us to approach the eigenvalue problem by separation of variables. In particular, we assume that each state $|\psi_n\rangle \equiv |n\rangle$ is a product of independent states in each dimension,

$$|n\rangle = |n_x\rangle \otimes |n_y\rangle.$$

The time-independent Schrödinger equation for eigenstates $|n\rangle$ now reads

$$\hat{H}|n\rangle = \left(\hat{H}_x|n_x\rangle\right) \otimes |n_y\rangle + |n_x\rangle \otimes \left(\hat{H}_y|n_y\rangle\right) \stackrel{!}{=} E_n\left(|n_x\rangle \otimes |n_y\rangle\right), \tag{2.62}$$

where the eigenvalue E_n needs to be determined. Inserting the well-known solution for one dimension, $E_{n_i} = \hbar\omega\left(n_i + \frac{1}{2}\right)$ for $i \in \{x, y\}$, this eigenvalue is

$$\begin{aligned}E_n(n_x, n_y) &= \hbar\omega\left(n_x + \frac{1}{2}\right) + \hbar\omega\left(n_y + \frac{1}{2}\right) \\ &= \hbar\omega\left(n_x + n_y + 1\right).\end{aligned} \tag{2.63}$$

In other words, the eigenvalues are simply added for each of the dimensions. For $d = 3$ dimensions, an analogue derivation yields

$$\begin{aligned}E_n(n_x, n_y, n_z) &= \hbar\omega\left(n_x + \frac{1}{2}\right) + \hbar\omega\left(n_y + \frac{1}{2}\right) + \hbar\omega\left(n_z + \frac{1}{2}\right) \\ &= \hbar\omega\left(n_x + n_y + n_z + \frac{3}{2}\right).\end{aligned} \tag{2.64}$$

Chapter 3

Many-body theory

When studying real physical systems, for instance nucleons in a nucleus, electrons in atoms or atoms in a molecule, one usually considers more than one particle. The degrees of freedom of the system increase with the number of particles, and the many-body Schrödinger equation includes more terms than simply the sum of the single-particle contributions: The particles interact, and since each particle influences each other one's motion, the problem gets almost impossibly complicated. In the general case, one neither knows the exact form of the Hamiltonian, nor is one able to solve Schrödinger's equation with conventional methods. It is therefore necessary to simplify the problem and make approximations, and several many-body methods have been developed to understand the behaviour of interacting systems.
In this thesis, the focus lies on interacting electrons and the following sections serve to explain the basic aspects of many-body theory, especially concentrating on second quantization. Unless explicit references are given, we follow the explanations in [25, 37].

3.1 The many-body problem

The problem of interest is an isolated system consisting of N particles. The evolution is described by Schrödinger's equation, which for one particle has been given in Eq. (2.24). For more than one particle, the many-body wave function

$$\Psi(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N; t) \equiv \Psi(\mathbf{R}, t) \tag{3.1}$$

is an N-dimensional vector in the composite Hilbert space

$$\mathcal{H}_N = \mathcal{H}_1^{(1)} \otimes \mathcal{H}_1^{(2)} \cdots \otimes \mathcal{H}_1^{(N)}.$$

Here the single-particle Hilbert space $\mathcal{H}_1^{(i)}$ denotes the space of square integrable functions over spatial as well as spin degrees of freedom, and the basis of $\mathcal{H}_N$ is given by direct products of the corresponding single-particle basis states:

$$|\Phi_N^{(\alpha)}(\mathbf{R}, t)\rangle = |\phi_{\alpha_1}(\mathbf{r}_1, t)\rangle \otimes |\phi_{\alpha_2}(\mathbf{r}_2, t)\rangle \otimes \cdots \otimes |\phi_{\alpha_N}(\mathbf{r}_N, t)\rangle, \tag{3.2}$$

where $\mathbf{r}_i$ contains spin in addition to spatial degrees of freedom. Each general N-particle wave function $\Psi(\mathbf{R}, t)$ can now be expanded in terms of those basis functions. In bra-ket notation, this can be formulated as

$$\begin{aligned} |\Psi(\mathbf{R},t)\rangle &= \sum_{\alpha} C(\alpha)|\Phi_N^{(\alpha)}\rangle \\ &= \sum_{\alpha_1\cdots\alpha_N} C(\alpha_1\cdots\alpha_N)|\phi_{\alpha_1}\rangle\otimes|\phi_{\alpha_2}\rangle\otimes\cdots\otimes|\phi_{\alpha_N}\rangle \\ &\equiv \sum_{\alpha_1\cdots\alpha_N} C(\alpha_1\cdots\alpha_N)|\phi_{\alpha_1}\phi_{\alpha_2}\cdots\phi_{\alpha_N}\rangle, \end{aligned} \tag{3.3}$$

where we skipped the **r**- and t-dependence on the right-hand side for better readability. With the single-particle functions ϕ_{α_i} normalized as explained in chapter 2, $|C(\alpha_1\cdots\alpha_N)|^2$ represents the probability with which a measurement of a observable in state $|\Psi(\mathbf{R},t)\rangle$ will yield the eigenvalue of $|\phi_{\alpha_1}\phi_{\alpha_2}\cdots\phi_{\alpha_N}\rangle$.

Apart from the wave function, also the Hamiltonian of Eq. (2.24) has to be extended to include the contributions from all particles. A first approach is to start with the non-interacting case, where the Hamiltonian of the N-particle system is given as the sum of the single-particle Hamiltonians $\hat{h}_i^{(0)}$,

$$\hat{H}_0 = \sum_i \hat{h}_i^{(0)} = \sum_i \left(-\frac{\hbar^2}{2m}\nabla_i^2 + \hat{v}_i\right), \tag{3.4}$$

where $\hat{v}_i$ denotes the external single-particle potential. With this Hamiltonian, the time-independent Schrödinger equation, $\hat{H}|\Psi(\mathbf{R},t)\rangle = E|\Psi(\mathbf{R},t)\rangle$, is separable, with solution

$$\hat{H}|\Psi(\mathbf{R},t)\rangle = \left(\sum_i \hat{h}_i^{(0)}\right)|\phi_1\rangle\otimes|\phi_2\rangle\otimes\cdots\otimes|\phi_N\rangle = \sum_i \epsilon_i|\phi_i\rangle. \tag{3.5}$$

The single-particle energies ϵ_i are the solutions to the associated one-particle problems

$$\hat{h}^{(0)}|\phi_i\rangle = \epsilon_i|\phi_i\rangle.$$

Taking the interaction between the particles into account, the potential energy has to be extended by an interaction term $\hat{V}_{int}$, such that the total Hamiltonian reads

$$\hat{H} = \sum_i \hat{h}_i^{(0)} + \hat{V}_{int} = \sum_i \left(-\frac{\hbar^2}{2m}\nabla_i^2 + \hat{v}_i\right) + \hat{V}_{int}. \tag{3.6}$$

The explicit form of $\hat{V}_{int}$ is usually unknown, and depending on the many-body method, there exist different ways to model it.

3.1.1 Fermionic systems

In this thesis, we deal with electrons, which are fermions. Fermions are particles with half-integer spin and follow the Pauli exclusion principle, stating that two fermions cannot simultaneously occupy the same quantum state. In the case that two fermions have the same spatial probability distribution, at least one other property, for instance spin, must be different.

Moreover, our electrons behave as identical particles, meaning that under similar physical conditions, they behave exactly the same way and therefore cannot be distinguished by any objective measurement. While in classical mechanics, due to a computable orbit, particles are always identifiable, in quantum mechanics the principle of indistinguishability holds. Resulting from the uncertainty relation, the particles have no sharply defined orbit. Therefore the occupation probabilities of mutually interacting identical particles overlap, making their identification impossible.
As a consequence, the probability distribution of a system should not be altered when interchanging the coordinates of two particles i and j. Introducing the permutation operator $\hat{P}_{ij}$, with the property

$$\hat{P}_{ij}|\phi_1 \cdots \phi_i \cdots \phi_j \cdots \phi_N\rangle = |\phi_1 \cdots \phi_j \cdots \phi_i \cdots \phi_N\rangle,$$

we can express this fact by

$$|\Psi(\mathbf{R},t)|^2 = |\hat{P}_{ij}\Psi(\mathbf{R},t)|^2. \tag{3.7}$$

Equation (3.7) has two solutions, namely

$$\hat{P}_{ij}\Psi(\mathbf{R},t) = \Psi(\mathbf{R},t), \qquad \hat{P}_{ij}\Psi(\mathbf{R},t) = -\Psi(\mathbf{R},t).$$

The first solution results in a symmetric wave function, describing bosons, whereas the second solution corresponds to an antisymmetric wave function and describes fermions.

To construct such an antisymmetric wave function for electrons, one usually expresses the wave function as so-called *Slater determinant*, named after J.C. Slater who first proposed this model in 1929 [38]. For an N-particle function, the entries of this determinant are N single-particle functions $\phi_{\alpha_1}, \phi_{\alpha_2}, \cdots, \phi_{\alpha_N}$, forming a complete, orthonormal basis. With the positions and spin degrees of freedoms of the particles given by $\mathbf{r}_1, \cdots, \mathbf{r}_N$, such a determinant reads

$$\Phi_{\alpha_1,\alpha_2,\cdots,\alpha_N}(\mathbf{r}_1,\cdots,\mathbf{r}_N) = \frac{1}{\sqrt{N!}} \begin{vmatrix} \phi_{\alpha_1}(\mathbf{r}_1) & \phi_{\alpha_2}(\mathbf{r}_1) & \cdots & \phi_{\alpha_N}(\mathbf{r}_1) \\ \phi_{\alpha_1}(\mathbf{r}_2) & \phi_{\alpha_2}(\mathbf{r}_2) & \cdots & \phi_{\alpha_N}(\mathbf{r}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{\alpha_1}(\mathbf{r}_N) & \phi_{\alpha_2}(\mathbf{r}_N) & \cdots & \phi_{\alpha_N}(\mathbf{r}_N) \end{vmatrix}, \tag{3.8}$$

where the factor $1/\sqrt{N!}$ accounts for the indistinguishability of the particles and ensures normalization of the wave function. Since determinants have the property to vanish whenever two of their rows or columns are equal, the Pauli exclusion principle is respected. The feature that determinants change their sign whenever two of their rows or columns are exchanged ensures antisymmetry. It is important to note that two electrons are allowed to have the same position, as long as their spin is different. In particular, each of the single-particle functions $\phi_{\alpha_i}(\mathbf{r}_j)$ is strictly speaking composed of two parts, namely a spatial and a spin part:

$$\phi_{\alpha_i}(\mathbf{r}_j) = \tilde{\phi}_{\alpha_i}(x_j, y_j, z_j) \otimes \chi(\sigma_j), \tag{3.9}$$

where σ_j denotes the spin orientation of particle j.

The basic idea now is that any arbitrary wave function can be expressed as a linear combination of such Slater determinants,

$$\Psi(\mathbf{R}) = \sum_{\alpha_1,\alpha_2\cdots\alpha_N} C_{\alpha_1,\alpha_2\cdots\alpha_N} \Phi_{\alpha_1,\alpha_2,\cdots,\alpha_N}(\mathbf{r}_1,\cdots,\mathbf{r}_N), \tag{3.10}$$

where the Slater determinants $\Phi_{\alpha_1,\alpha_2,\cdots,\alpha_N}(\mathbf{r}_1,\cdots,\mathbf{r}_N)$ are defined by Eq. (3.8). Such an expansion is possible for all times t, such that we have suppressed the t-dependence.

3.2 Second Quantization

The formalism of second quantization involves a reformulation of the original Schrödinger equation, allowing a considerable simplification of the many-body problem. Tedious constructions of wave functions as products of single-particle wave functions get redundant when making use of the creation and annihilation operators introduced in the previous chapter. The overall statistical properties are then contained in fundamental commutation relations, and complicated interactions between particles can be modelled in terms of creation and annihilation of particles.
In the following sections, we will give a short introduction to the formalisms of second quantization and demonstrate how these formalisms can be applied to fermionic systems. Since we are interested in electrons, we will restrict us to antisymmetric wave functions constructed of Slater determinants.

3.2.1 The basic formalism

A first useful tool is to introduce the *occupancy notation* for Slater determinants (SDs),

$$\Phi_{\alpha_1,\alpha_2,\cdots,\alpha_N} \equiv |\alpha_1\alpha_2\cdots\alpha_N\rangle, \tag{3.11}$$

which specifies which basis states ϕ_{α_i} are occupied in the determinant. The creation and annihilation operators, $a^\dagger$ and a, respectively, are defined in terms of their action on the SDs,

$$a^\dagger_{\alpha_0}|\alpha_1\alpha_2\cdots\alpha_N\rangle = |\alpha_0\alpha_1\alpha_2\cdots\alpha_N\rangle \tag{3.12}$$
$$a_{\alpha_1}|\alpha_1\alpha_2\cdots\alpha_N\rangle = |\alpha_2\cdots\alpha_N\rangle. \tag{3.13}$$

In Eq. (3.12), the creation operator $a^\dagger_{\alpha_0}$ adds a state ϕ_{α_0} to the Slater determinant, creating an $(N+1)$- from an N-particle state. On the other hand, in Eq. (3.13), the annihilation operator a_{α_1} removes a state ϕ_{α_1}, thus transforming an N- to an $(N-1)$-particle state.
We define the vacuum state $|0\rangle$ as state where none of the orbitals are occupied, and have

$$a_{\alpha_i}|0\rangle = 0, \qquad \forall i \in \mathbb{N}.$$

Each N-particle state can now be generated by applying a product of creation operators on the vacuum state,

$$|\alpha_1\alpha_2\cdots\alpha_N\rangle = a^\dagger_{\alpha_1}a^\dagger_{\alpha_2}\cdots a^\dagger_{\alpha_N}|0\rangle. \tag{3.14}$$

Note that the orbitals are given an ordering, guaranteeing antisymmetry by

$$|\alpha_1\cdots\alpha_i\alpha_j\cdots\alpha_N\rangle = -|\alpha_1\cdots\alpha_j\alpha_i\cdots\alpha_N\rangle. \tag{3.15}$$

Carried over to the creation operators, this means

$$a^\dagger_{\alpha_i}a^\dagger_{\alpha_j} = -a^\dagger_{\alpha_j}a^\dagger_{\alpha_i}.$$

The same holds true for the annihilation operators. Defining the anticommutator of two operators $\hat{A}, \hat{B}$,

$$\{\hat{A},\hat{B}\} = \hat{A}\hat{B} + \hat{B}\hat{A}, \tag{3.16}$$

we obtain the following basic anticommutation relations[1]:

$$\begin{aligned} \{a_\alpha^\dagger, a_\beta^\dagger\} &= 0, \\ \{a_\alpha, a_\beta\} &= 0, \\ \{a_\alpha^\dagger, a_\beta\} &= \delta_{\alpha\beta}. \end{aligned} \tag{3.17}$$

3.2.2 Second quantization with reference state

When the system consists of a large amount of particles, it gets often rather cumbersome to work with the physical vacuum state $|0\rangle$. A large number of creation operators has to be taken into account and worked with, often resulting in long equations.
To reduce the dimensionality of the problem, we introduce a reference state $|\Phi_0\rangle$, which is the state where N particles occupy the N single-particle states with the lowest energy. This reference state is also referred to as *Fermi vacuum*, and the level of the highest occupied orbital is called *Fermi level*.
In the following, we will refer to the single-particle states up to the Fermi level as *hole states* and label them with $i, j, k, \dots$, and to the ones above the Fermi level as *particle states*, labelled with $a, b, c, \dots$ General single-particle states that can lie above or below the Fermi level will be labelled with $p, q, r, \dots$

With this notation, the N-particle reference state $|\Phi_0\rangle$ is obtained by applying all N creation operators corresponding to hole states to the vacuum state,

$$|\Phi_0\rangle = a_i^\dagger a_j^\dagger a_k^\dagger \cdots |0\rangle. \tag{3.18}$$

We assume that the energies of the single-particle states are arranged in lexical order,

$$\cdots \geq \epsilon_k \geq \epsilon_j \geq \epsilon_i.$$

When applying the creation and annihilation operators to this reference state $|\Phi_0\rangle$, we have to guarantee that particles can only be annihilated if they are present in the determinant, and that we cannot create particles that are contained already.
For the creation operator $a^\dagger$, this suggests

$$a_p^\dagger |\Phi_0\rangle = \begin{cases} |\Phi^p\rangle, & \text{if } p \in \{a, b, c, \dots\} \\ 0 & \text{if } p \in \{i, j, k, \dots\} \end{cases}, \tag{3.19}$$

where $|\Phi^p\rangle$ denotes the reference state with particle p added. Similarly, we have for the annihilation operator

$$a_p |\Phi_0\rangle = \begin{cases} |\Phi_p\rangle, & \text{if } p \in \{i, j, k, \dots\} \\ 0 & \text{if } p \in \{a, b, c, \dots\} \end{cases}, \tag{3.20}$$

where $|\Phi_p\rangle$ denotes the reference state with particle p removed, or, equivalently, with hole p created.
Creating a number of n_p particles and n_h holes, one obtains so-called n_p-particle-n_h-hole excitations. For example, the determinant

$$|\Phi_{ij}^{ab}\rangle = a_a^\dagger a_b^\dagger a_i a_j |\Phi_0\rangle$$

is called a two-particle-two-hole excitation.

[1] For a proof of the relations, we refer to [25].

3.2.3 Wick's theorem

When computing the inner product between two determinants, the straightforward way is to transform the states into strings of operators acting on the vacuum or a reference state, and transform them further using anticommutation relations.
For example, to compute the inner product $\langle pq|rs\rangle$, we first rewrite

$$\langle pq| = \langle 0|a_p a_q, \qquad |rs\rangle = a_r^\dagger a_s^\dagger |0\rangle,$$

where we make use of the fact that creation and annihilation operator are Hermitian conjugate to each other. Afterwards, we aim to bring all annihilation operators to the right and all creation operators to the left, where we apply the anticommutation relations (3.17):

$$\begin{aligned}
\langle pq|rs\rangle &= \langle 0| \left(a_p a_q a_r^\dagger a_s^\dagger \right) |0\rangle \\
&= \langle 0| \left(a_p (\delta_{qr} - a_r^\dagger a_q) a_s^\dagger \right) |0\rangle \\
&= \langle 0| \left(a_p a_s^\dagger \delta_{qr} - a_p a_r^\dagger a_q a_s^\dagger \right) |0\rangle \\
&= \langle 0| \left((\delta_{ps} - a_p a_s^\dagger)\delta_{qr} - (\delta_{pr} - a_r^\dagger a_p)(\delta_{qs} - a_s^\dagger a_q) \right) |0\rangle \\
&= \langle 0| \left(\delta_{ps}\delta_{qr} - a_p^\dagger a_s^\dagger \delta_{qr} - \delta_{pr}\delta_{qs} + a_s^\dagger a_q \delta_{pr} + a_r^\dagger a_p \delta_{qs} - a_r^\dagger a_q \delta_{ps} + a_r^\dagger a_s^\dagger a_p a_q \right) |0\rangle \\
&= \delta_{ps}\delta_{qr} - \delta_{pr}\delta_{qs}
\end{aligned}$$

Only those two terms with only Kronecker deltas give a non-zero contribution, since in all other terms an annihilation operator acts on the vacuum state. As the number of particles is increasing, these computations get more and more lengthy, and one commonly applies Wick's theorem, which simplifies the calculations based on the concepts of *normal-ordering* and *contractions*.

For a string of operators $\hat{A}, \hat{B}, \hat{C}, \ldots$, the *normal-ordered* product[2] $\left\{ \hat{A}\hat{B}\hat{C}\ldots \right\}$ is defined as that rearrangement where all creation operators are moved to the left, and all annihilation operators to the right. In addition, a phase factor of (-1) arises for each permutation of nearest neighbour operators. Since creation and annihilation operators can permute among themselves, the normal-ordered form is not uniquely defined.
The most important property of normal-ordered products is that the expectation value with respect to the vacuum state vanishes,

$$\langle 0| \left\{ \hat{A}\hat{B}\ldots \right\} |0\rangle = 0. \tag{3.21}$$

A *contraction* between two operators is defined as

$$\overbrace{\hat{A}\hat{B}} \equiv \hat{A}\hat{B} - \left\{ \hat{A}\hat{B} \right\}. \tag{3.22}$$

[2] We start with normal-ordering *with respect to the vacuum state*, as originally used by Wick in [39].

The four possible types of contractions are

$$\begin{aligned}
\overbracket{a_p^\dagger a_q^\dagger} &= a_p^\dagger a_q^\dagger - a_p^\dagger a_q^\dagger = 0, \\
\overbracket{a_p a_q} &= a_p a_q - a_p a_q = 0, \\
\overbracket{a_p^\dagger a_q} &= a_p^\dagger a_q - a_p^\dagger a_q = 0, \\
\overbracket{a_p a_q^\dagger} &= a_p a_q^\dagger - (-a_q^\dagger a_p) = \{a_p, a_q^\dagger\} = \delta_{pq}.
\end{aligned} \tag{3.23}$$

The concept of normal-ordering can be extended from the physical vacuum state $|0\rangle$ to the reference state $|\Phi_0\rangle$. In this case, the normal-ordered product $\left\{\hat{A}\hat{B}\hat{C}\ldots\right\}$ requires that all creation operators above and all annihilation operators below the Fermi level are moved to the left, whereas all creation operator below and all annihilation operators above the Fermi level are moved to the right. With this reordering, one obtains again the useful property that the expectation value with respect to the reference state vanishes,

$$\langle\Phi_0|\left\{\hat{A}\hat{B}\ldots\right\}|\Phi_0\rangle = 0. \tag{3.24}$$

Labelling the indices as before, the only two non-zero contractions are

$$\begin{aligned}
\overbracket{a_i^\dagger a_j} &= a_i^\dagger a_j - (-a_j a_i^\dagger) = \delta_{ij}, \\
\overbracket{a_a a_b^\dagger} &= a_a a_b^\dagger - (-a_b^\dagger a_a) = \delta_{ab}.
\end{aligned} \tag{3.25}$$

The second relation is analogous to the vacuum case. This makes sense, since loosely speaking, with respect to the vacuum state all indices correspond to particles.

Statement of the theorem Wick's theorem states that any product of creation and annihilation operators can be expressed as normal-ordered product plus the sum of all possible normal-ordered products with contractions. Symbolically, this means

$$\begin{aligned}
\hat{A}\hat{B}\hat{C}\hat{D}\hat{E}\hat{F}\ldots = &\left\{\hat{A}\hat{B}\hat{C}\hat{D}\hat{E}\hat{F}\ldots\right\} \\
&+ \left\{\overbracket{\hat{A}\hat{B}}\hat{C}\hat{D}\hat{E}\hat{F}\ldots\right\} + \left\{\overbracket{\hat{A}\hat{B}\hat{C}}\hat{D}\hat{E}\hat{F}\ldots\right\} + \ldots \\
&+ \left\{\overbracket{\hat{A}\hat{B}}\overbracket{\hat{C}\hat{D}}\hat{E}\hat{F}\ldots\right\} + \left\{\overbracket{\hat{A}\hat{B}\hat{C}}\overbracket{\hat{D}\hat{E}}\hat{F}\right\} + \ldots \\
&+ \ldots \\
&+ \left\{\overbracket{\hat{A}\hat{B}}\overbracket{\hat{C}\hat{D}}\overbracket{\hat{E}\hat{F}}\ldots\right\} + \ldots
\end{aligned} \tag{3.26}$$

In words, the first term is the normal-ordered string, followed by all possible normal-ordered products with contractions between two operators. Afterwards, there come all possible contractions between four operators and this scheme is continued, up to the terms where all

operators are contracted.
When calculating the expectation value with respect to the vacuum or a reference state, this theorem brings considerable simplifications: As suggested by Eqs. (3.21) and (3.24), only the last terms of relation (3.26), namely where all operators are contracted, can give a non-zero contribution. Regarding the previous example, this means

$$\begin{aligned}\langle pq|rs\rangle &= \langle 0|\left(a_p a_q a_r^\dagger a_s^\dagger\right)|0\rangle \\ &= \langle 0|\left(\overbrace{a_p \overbrace{a_q a_r^\dagger}^{} a_s^\dagger}^{} + \overbrace{a_p \overbrace{a_q a_r^\dagger a_s^\dagger}^{}}^{}\right)|0\rangle \\ &= \delta_{ps}\delta_{qr} - \delta_{pr}\delta_{qs}.\end{aligned}$$

The result is the same as before, but obtained with much less effort, demonstrating the power of Wick's theorem.
Another useful feature is that a product of two already normal-ordered operator strings can be rewritten as the normal-ordered product of the total group of operators plus all possible contractions between the first and the second string. In particular, there are no internal contractions inside each of the strings. This statement is often referred to as *Wick's generalized theorem*.

3.2.4 Hamiltonian in second quantization

To make use of the machinery of second quantization when computing expectation values, it is necessary to find a representation for the quantum-mechanical operators.
To compute expectation values of the form $\langle\Phi_1|\hat{A}|\Phi_2\rangle$, we utilize the fact that the single-particle functions of our basis are orthonormal, such that for $\langle\Phi_1|\Phi_2\rangle$ to be 1, the determinants $|\Phi_1\rangle$ and $|\Phi_2\rangle$ must have an identical occupation scheme.
To count the number of occupied orbitals, we introduce the number operator

$$\hat{N} = \sum_p a_p^\dagger a_p. \tag{3.27}$$

Applied to a determinant $|\Phi_N\rangle = |\alpha_1\alpha_2\ldots\alpha_N\rangle$, we get

$$\begin{aligned}\hat{N}|\alpha_1\alpha_2\ldots\alpha_N\rangle &= \sum_p a_p^\dagger a_p\ |\alpha_1\alpha_2\ldots\alpha_N\rangle \\ &= a_1^\dagger|\alpha_2\alpha_3\ldots\alpha_N\rangle + a_2^\dagger|\alpha_1\alpha_3\ldots\alpha_N\rangle + \ldots \\ &= |\alpha_1\alpha_2\ldots\alpha_N\rangle + |\alpha_1\alpha_2\ldots\alpha_N\rangle + \ldots \\ &= N|\alpha_1\alpha_2\ldots\alpha_N\rangle.\end{aligned} \tag{3.28}$$

In words, when acting on a determinant, the number operator loops over all particles, each time removing and adding the same particle subsequently. With relation (3.28), the expectation value is

$$\langle\Phi_N|\hat{N}|\Phi_N\rangle = \langle\Phi_N|N|\Phi_N\rangle = N.$$

Let us now demonstrate how the more complex operators included in the Hamiltonian can be expressed in a similar manner:

Restricted to maximally two-body interactions, the Hamiltonian of Eq. (3.6) is given by

$$\begin{aligned}\hat{H} &= \sum_i \hat{h}_i^{(0)} + \hat{V} \\ &= \sum_i \left(-\frac{\hbar^2}{2m}\nabla_i^2 + \hat{v}_i \right) + \frac{1}{2}\sum_{ij} \hat{v}_{ij}(\mathbf{r}_i, \mathbf{r}_j) \\ &\equiv \hat{H}_0 + \hat{H}_I.\end{aligned}$$

The non-interacting part of the Hamiltonian, $\hat{H}_0$, acts just on one particle at a time and represents therefore a *one-body operator*. In second quantization, it reads[3]

$$\hat{H}_0 = \sum_{pq} \langle p|\hat{h}^{(0)}|q\rangle a_p^\dagger a_q. \tag{3.29}$$

Similar to the number operator, its role as operator is translated into creation and annihilation operators. In addition to the simple number operator, we now encounter transition amplitudes

$$\langle p|\hat{h}^{(0)}|q\rangle = \int d\mathbf{r}\ \phi_p^*(\mathbf{r})\hat{h}^{(0)}\phi_q(\mathbf{r}), \tag{3.30}$$

which represent the probability that the operator moves a particle from single-particle state $|q\rangle$ to $|p\rangle$.
For the two-body part of the Hamiltonian, we have analogously

$$\hat{V} = \frac{1}{2}\sum_{pqrs} \langle pq|\hat{v}|rs\rangle a_p^\dagger a_q^\dagger a_s a_r, \tag{3.31}$$

$$\langle pq|\hat{v}|rs\rangle = \int\int d\mathbf{r}_1 d\mathbf{r}_2\ \phi_p^*(\mathbf{r}_1)\phi_q^*(\mathbf{r}_2)\hat{v}(\mathbf{r}_1,\mathbf{r}_2)\phi_r(\mathbf{r}_1)\phi_s(\mathbf{r}_2). \tag{3.32}$$

The interpretation is that particles are removed from states $|r\rangle$ and $|s\rangle$ and created in states $|p\rangle$ and $|q\rangle$, respectively, with probability $\frac{1}{2}\langle pq|\hat{v}|rs\rangle$. Note that with definition (3.32), one does not account for the antisymmetry of states $|rs\rangle$, as stated in Eq. (3.15). This relation suggests that

$$\langle pq|\hat{v}|rs\rangle = -\langle pq|\hat{v}|sr\rangle, \tag{3.33}$$

where on the left-hand side, the first particle is moved from $|r\rangle$ to $|p\rangle$ and the second one from $|s\rangle$ to $|q\rangle$, whereas on the right-hand side, the first particle is moved from $|r\rangle$ to $|q\rangle$ and the second one from $|s\rangle$ to $|p\rangle$. To account for this antisymmetry, one commonly employs so-called *antisymmetric elements*, defined by

$$\langle pq||rs\rangle = \langle pq|\hat{v}|rs\rangle - \langle pq|\hat{v}|sr\rangle. \tag{3.34}$$

With these elements, the two-body interaction operator reads

$$\hat{V} = \frac{1}{4}\sum_{pqrs} \langle pq||rs\rangle a_p^\dagger a_q^\dagger a_s a_r, \tag{3.35}$$

[3] For a profound motivation of this representation, see [25].

and the full Hamiltonian

$$\begin{aligned}\hat{H} &= \hat{H}_0 + \hat{V} \\ &= \sum_{pq} \langle p|\hat{h}^{(0)}|q\rangle a_p^\dagger a_q + \frac{1}{4}\sum_{pqrs}\langle pq||rs\rangle a_p^\dagger a_q^\dagger a_s a_r. \end{aligned} \tag{3.36}$$

To demonstrate how the expectation value of the Hamiltonian, $\langle\Phi_1|\hat{H}|\Phi_2\rangle$, is computed making use of Wick's theorem, let us consider two determinants

$$\begin{aligned}|\Phi_1\rangle &= |\alpha_1\alpha_2\ldots\alpha_N\rangle, \\ |\Phi_2\rangle &= |\beta_1\beta_2\ldots\beta_N\rangle,\end{aligned}$$

and start with the interaction part. Since we have restricted the Hamiltonian to two-body operators, maximally two of the states in $|\Phi_1\rangle$ and $|\Phi_2\rangle$ can be different. If more than two states are different, our Hamiltonian can not link all the states and the expectation value vanishes.
If two states are different, the expectation value can be simplified to the one between two two-particle determinants,

$$\begin{aligned}\langle\Phi_1|\hat{V}|\Phi_2\rangle &= \langle\alpha_N\ldots i\ldots j\ldots\alpha_2\alpha_1|\hat{V}|\alpha_1\alpha_2\ldots k\ldots l\ldots\alpha_N\rangle \\ &= \underbrace{\langle\alpha_N|\alpha_N\rangle}_{1}\cdots\underbrace{\langle\alpha_2|\alpha_2\rangle}_{1}\underbrace{\langle\alpha_1|\alpha_1\rangle}_{1}\langle ij|\hat{V}|kl\rangle \\ &= \langle ij|\hat{V}|kl\rangle.\end{aligned}$$

Afterwards Wick's theorem can be applied:

$$\begin{aligned}\langle ij|\hat{V}|kl\rangle &= \frac{1}{4}\langle 0|a_j a_i \sum_{pqrs}\langle pq||rs\rangle a_p^\dagger a_q^\dagger a_s a_r a_k^\dagger a_l^\dagger|0\rangle \\ &= \frac{1}{4}\sum_{pqrs}\langle pq||rs\rangle\langle 0|a_j a_i a_p^\dagger a_q^\dagger a_s a_r a_k^\dagger a_l^\dagger|0\rangle \\ &= \frac{1}{4}\sum_{pqrs}\langle pq||rs\rangle\langle 0|\overbrace{a_j a_i a_p^\dagger a_q^\dagger}\overbrace{a_s a_r a_k^\dagger a_l^\dagger} + \overbrace{a_j a_i a_p^\dagger a_q^\dagger a_s a_r a_k^\dagger a_l^\dagger} \\ &\quad + \overbrace{a_j a_i a_p^\dagger a_q^\dagger}\overbrace{a_s a_r a_k^\dagger a_l^\dagger} + \overbrace{a_j a_i a_p^\dagger a_q^\dagger a_s a_r a_k^\dagger a_l^\dagger}|0\rangle \\ &= \frac{1}{4}\sum_{pqrs}\langle pq||rs\rangle\langle 0|\delta_{jp}\delta_{iq}\delta_{sk}\delta_{rl} - \delta_{jq}\delta_{ip}\delta_{sk}\delta_{rl} - \delta_{jp}\delta_{iq}\delta_{sl}\delta_{rk} + \delta_{jq}\delta_{ip}\delta_{sl}\delta_{rk}|0\rangle \\ &= \frac{1}{4}\left(\langle ji||lk\rangle - \langle ij||lk\rangle - \langle ji||kl\rangle + \langle ij||kl\rangle\right) \\ &= \langle ij||kl\rangle.\end{aligned}$$

Note that we have only written down those fully contracted terms where all contractions are non-zero. To get the correct phase, it is possible to count the number of crossings between the contraction lines, instead of permuting the operators to get contracted pairs next to each other. An even number of crossings gives a positive, an odd number a negative phase. In the last step, we have made use of the antisymmetry relations, giving four equal terms.

If less than two states in the determinants $|\Phi_1\rangle$ and $|\Phi_2\rangle$ are different, the operator can link several possible pairs of states. If one of the states is different, here chosen as transition from state $|k\rangle$ to $|j\rangle$, this means

$$\begin{aligned}\langle\Phi_1|\hat{V}|\Phi_2\rangle &= \langle\alpha_1 j||\alpha_1 k\rangle + \langle\alpha_2 j||\alpha_2 k\rangle + \dots \\ &= \sum_i \langle\alpha_i j||\alpha_i k\rangle,\end{aligned}$$

where i sums over all occupied single-particle states. In the case that both determinants are equal, complete free summation is possible:

$$\langle\Phi_1|\hat{V}|\Phi_1\rangle = \sum_{i<j}\langle ij||ij\rangle.$$

The restriction $i < j$ makes sure that equivalent configurations are not counted twice.
Analogous to the fact that the two-body operator can maximally link two determinants with two different states in their occupancy scheme, the one-body operator can maximally change one state. Thus the contribution from the non-interacting part of the Hamiltonian simplifies to

$$\begin{aligned}\langle\Phi_1|\hat{H}_0|\Phi_2\rangle &= \langle\alpha_N \dots i \dots \alpha_2\alpha_1|\hat{H}_0|\alpha_1\alpha_2 \dots j \dots \alpha_N\rangle \\ &= \underbrace{\langle\alpha_N|\alpha_N\rangle}_{1} \cdots \underbrace{\langle\alpha_2|\alpha_2\rangle}_{1}\underbrace{\langle\alpha_1|\alpha_1\rangle}_{1}\langle i|\hat{H}_0|j\rangle \\ &= \langle i|\hat{H}_0|j\rangle.\end{aligned}$$

Again, we apply Wick's theorem and get

$$\begin{aligned}\langle i|\hat{H}_0|j\rangle &= \langle 0|a_i \sum_{pq}\langle p|\hat{h}^{(0)}|q\rangle a_p^\dagger a_q a_j^\dagger|0\rangle \\ &= \sum_{pq}\langle p|\hat{h}^{(0)}|q\rangle\langle 0|\overbrace{a_i a_p^\dagger}\overbrace{a_q a_j^\dagger}|0\rangle \\ &= \sum_{pq}\langle p|\hat{h}^{(0)}|q\rangle\delta_{ip}\delta_{qj} \\ &= \langle i|\hat{h}^{(0)}|j\rangle.\end{aligned}$$

In the case that both determinants have the same occupancy scheme, the operator has several possibilities to link the states:

$$\langle\Phi_1|\hat{H}_0|\Phi_1\rangle = \sum_i\langle i|\hat{h}^{(0)}|i\rangle.$$

Although in this thesis we work with a Hamiltonian that is restricted to two-body interactions, the interaction part can in principle contain higher-order interactions, too. In this case, it is straightforwardly extended to

$$\hat{H}_I = \frac{1}{2!}\sum_{pqrs}\langle pq|\hat{v}|rs\rangle + \frac{1}{3!}\sum_{pqrstu}\langle pqr|\hat{v}^{(3)}|stu\rangle + \dots + \frac{1}{N!}\sum_{\substack{pqr\dots\\stu\dots}}\langle pqr\dots|\hat{v}^{(N)}|\dots stu\rangle, \quad (3.37)$$

where $v^{(n)}$ denotes the interaction potential for interactions between n particles.

In-medium formulation of the Hamiltonian

To make use of the particle-hole formalism introduced in subsection 3.2.2 and the advantages calculations with a reference state $|\Phi_0\rangle$ bring, it is useful to express the Hamiltonian in terms of normal-ordered strings of operators. In particular, starting with the second-quantized Hamiltonian

$$\begin{aligned}\hat{H} = &\sum_{pq}\langle p|\hat{h}^{(0)}|q\rangle a_p^\dagger a_q + \frac{1}{4}\sum_{pqrs}\langle pq||rs\rangle a_p^\dagger a_q^\dagger a_s a_r \\ &+ \frac{1}{36}\sum_{pqrstu}\langle pqr|\,\hat{v}^{(3)}\,|stu\rangle\, a_p^\dagger a_q^\dagger a_r^\dagger a_u a_t a_s + \ldots,\end{aligned} \tag{3.38}$$

Wick's theorem can be used to convert all operator strings $a_1 a_2 \cdots a_{n-1}^\dagger a_n^\dagger$ into sums of normal-ordered expressions. Defining $\delta_{pq<F}$ to be the Kronecker delta function where states p and q are restricted to be hole states below the Fermi level, the one-body operator can be rewritten the following way:

$$a_p^\dagger a_q = \{a_p^\dagger a_q\} + \overbrace{a_p^\dagger a_q} = \{a_p^\dagger a_q\} + \delta_{pq<F}. \tag{3.39}$$

With our convention of labelling the states, in particular the use of $\{i, j, \ldots\}$ for hole states below the Fermi level, the non-interacting part of $\hat{H}$ thus reads

$$\hat{H}_N^{(0)} = \sum_{pq}\langle p|\hat{h}^{(0)}|q\rangle\{a_p^\dagger a_q\} + \sum_i \langle i|\hat{h}^{(0)}|i\rangle, \tag{3.40}$$

where the subscript N denotes normal-ordering. The same procedure can be applied to the two-body part of $\hat{H}$:

$$\begin{aligned}a_p^\dagger a_q^\dagger a_s a_r = &\{a_p^\dagger a_q^\dagger a_s a_r\} + \left\{\overbrace{a_p^\dagger a_q^\dagger a_s a_r}\right\} + \left\{\overbrace{a_p^\dagger a_q^\dagger a_s a_r}\right\} + \left\{a_p^\dagger \overbrace{a_q^\dagger a_s} a_r\right\} \\ &+ \left\{a_p^\dagger \overbrace{a_q^\dagger a_s a_r}\right\} + \left\{\overbrace{a_p^\dagger a_q^\dagger} a_s a_r\right\} + \left\{\overbrace{a_p^\dagger a_q^\dagger a_s} a_r\right\} \\ = &\{a_p^\dagger a_q^\dagger a_s a_r\} - \delta_{ps<F}\{a_q^\dagger a_r\} + \delta_{pr<F}\{a_q^\dagger a_s\} + \delta_{qs<F}\{a_p^\dagger a_r\} \\ &- \delta_{qr<F}\{a_p^\dagger a_s\} - \delta_{ps<F}\delta_{qr<F} + \delta_{pr<F}\delta_{qs<F}.\end{aligned} \tag{3.41}$$

Inserting this into the expression for the two-body operator $\hat{V}$, we obtain

$$\begin{aligned}\hat{V} &= \frac{1}{4}\sum_{pqrs}\langle pq||rs\rangle a_p^\dagger a_q^\dagger a_s a_r \\ &= \frac{1}{4}\sum_{pqrs}\langle pq||rs\rangle\{a_p^\dagger a_q^\dagger a_s a_r\} - \frac{1}{4}\sum_{qri}\langle iq||ri\rangle\{a_q^\dagger a_r\} + \frac{1}{4}\sum_{qsi}\langle iq||is\rangle \\ &\quad + \frac{1}{4}\sum_{pri}\langle pi||ri\rangle\{a_p^\dagger a_r\} - \frac{1}{4}\sum_{psi}\langle pi||is\rangle\{a_p^\dagger a_s\} - \frac{1}{4}\sum_{ij}\langle ij||ji\rangle + \frac{1}{4}\sum_{ij}\langle ij||ij\rangle.\end{aligned} \tag{3.42}$$

With the antisymmetry relation (3.15) suggesting that

$$\langle pq||rs\rangle = -\langle qp||rs\rangle = -\langle pq||sr\rangle = \langle qp||sr\rangle, \tag{3.43}$$

expression (3.42) can be simplified to

$$\hat{V}_N = \frac{1}{4}\sum_{pqrs}\langle pq||rs\rangle\{a_p^\dagger a_q^\dagger a_s a_r\} + \sum_{pqi}\langle pi||qi\rangle\{a_p^\dagger a_q\} + \frac{1}{2}\sum_{ij}\langle ij||ij\rangle. \tag{3.44}$$

The subscript N denotes again normal-ordering. For higher-order interactions, one proceeds analogously. Calculations for the three-body operator yield

$$\begin{aligned}\hat{V}^{(3)} &= \frac{1}{36}\sum_{pqrstu}\langle pqr|\,\hat{v}^{(3)}\,|stu\rangle\, a_p^\dagger a_q^\dagger a_r^\dagger a_u a_t a_s \\ &= \frac{1}{36}\sum_{pqrstu}\langle pqr|\,\hat{v}^{(3)}\,|stu\rangle\,\{a_p^\dagger a_q^\dagger a_r^\dagger a_u a_t a_s\} + \frac{1}{4}\sum_{i}\langle pqi|\,\hat{v}^{(3)}\,|rsi\rangle\,\{a_p^\dagger a_q^\dagger a_s a_r\} \\ &\quad + \frac{1}{2}\sum_{ij}\langle pij|\,\hat{v}^{(3)}\,|pij\rangle\,\{a_p^\dagger a_q\} + \frac{1}{6}\sum_{ijk}\langle ijk|\,\hat{v}^{(3)}\,|ijk\rangle\,.\end{aligned} \tag{3.45}$$

One usually collects all one-body contributions in the one-body operator $\hat{F}_N$, all two-body contributions in the two-body operator $\hat{V}_N$ and all higher-order contributions in $\hat{H}_N^{(3)}, \hat{H}_N^{(4)}, \ldots$ A Hamiltonian restricted to three-body interactions then reads

$$\begin{aligned}\hat{H}_N &= \hat{F}_N + \hat{V}_N + \hat{H}_N^{(3)} \\ &= \sum_{pq} f_{pq}\{a_p^\dagger a_q\} + \frac{1}{4}\sum_{pqrs}\Gamma_{pqrs}\{a_p^\dagger a_q^\dagger a_s a_r\} + \frac{1}{36}\sum_{pqrstu}W_{pqrstu}\{a_p^\dagger a_q^\dagger a_r^\dagger a_u a_t a_s\},\end{aligned} \tag{3.46}$$

with the amplitudes for the one-body, two-body and three-body operator given by

$$f_{pq} = \langle p|\,\hat{h}^{(0)}\,|q\rangle + \sum_i\langle pi||qi\rangle + \frac{1}{2}\sum_{ij}\langle pij|\,\hat{v}^{(3)}\,|pij\rangle\,, \tag{3.47}$$

$$\Gamma_{pqrs} = \langle pq||rs\rangle + \sum_i\langle pqi|\,\hat{v}^{(3)}\,|rsi\rangle\,, \tag{3.48}$$

$$W_{pqrstu} = \langle pqr|\,\hat{v}^{(3)}\,|stu\rangle\,, \tag{3.49}$$

respectively. When restricting the Hamiltonian to two-body operators, as will be used in this thesis, the normal-ordered Hamiltonian $\hat{H}_N$ simplifies to

$$\begin{aligned}\hat{H}_N &= \hat{F}_N + \hat{V}_N \\ &= \sum_{pq} f_{pq}\{a_p^\dagger a_q\} + \sum_{pqrs}\langle pq||rs\rangle\{a_p^\dagger a_q^\dagger a_s a_r\},\end{aligned} \tag{3.50}$$

with

$$f_{pq} = \langle p|\hat{h}^{(0)}|q\rangle + \sum_i\langle pi||qi\rangle. \tag{3.51}$$

Note that the normal-ordered Hamiltonian does by definition not contain any scalar terms, which represent the ground state energy $E_0 = \langle\Phi_0|\hat{H}|\Phi_0\rangle$. In particular, the following relation holds:

$$\hat{H}_N = \hat{H} - E_0. \tag{3.52}$$

When collecting those scalar terms of Eqs. (3.40), (3.44) and (3.45), the ground state energy of the three-body Hamiltonian is given by

$$E_0 = \sum_i \langle i| \hat{h}^{(0)} |i\rangle + \frac{1}{2} \sum_{ij} \langle ij||ij\rangle + \frac{1}{6} \sum_{ijk} \langle ijk| \hat{v}^{(3)} |ijk\rangle \,,$$

and the one of the two-body Hamiltonian by

$$E_0 = \sum_i \langle i| \hat{h}^{(0)} |i\rangle + \frac{1}{2} \sum_{ij} \langle ij||ij\rangle \,. \tag{3.53}$$

Chapter 4

Ordinary Differential Equations

An essential part of this thesis is to solve a set of coupled ordinary differential equations. Although we do not implement the corresponding functions by ourselves, but use a library [40] instead, it is necessary to understand the basic formalism and adjust it to our needs. For that reason, this chapter explains the basic theoretical aspects of ordinary differential equations. The algorithm we use in our code has been developed by Shampine and Gordon [41] and bases on Adams methods, which we therefore put focus on in the following. Unless otherwise stated, we follow the explanations in [42, 43].

4.1 Basic concepts

Ordinary differential equations (ODEs) are equations involving the derivative of an unknown function with respect to a single variable x. In particular, they are usually given in the form

$$y'(x) = f(x, y(x)), \tag{4.1}$$

where x is defined on an interval $[a, b]$. More generally,

$$\begin{aligned} \mathbf{y} &= (y_1, \ldots, y_n), \\ \mathbf{f}(x, \mathbf{y}) &= (\mathbf{f}_1(x, y_1, \ldots, y_n), \ldots, \mathbf{f}_n(x, y_1, \ldots, y_n)) \end{aligned} \tag{4.2}$$

are vectors in an n-dimensional Euclidean space. This gives a set of *coupled* ordinary differential equations,

$$\begin{aligned} y_1'(x) &= f_1(x, y_1(x), y_2(x), \ldots, y_n(x)) \\ y_2'(x) &= f_2(x, y_1(x), y_2(x), \ldots, y_n(x)) \\ &\vdots \\ y_n'(x) &= f_n(x, y_1(x), y_2(x), \ldots, y_n(x)), \end{aligned}$$

which can be expressed compactly by

$$\mathbf{y}'(x) = \mathbf{f}(x, \mathbf{y}(x)). \tag{4.3}$$

For well-behaved $\mathbf{f}$, any point $(x_0, \mathbf{y}_0)$ completely determines a trajectory satisfying Eq. (4.3). This defines the *initial value problem* (IVP):

$$\begin{aligned} \mathbf{y}'(x) &= \mathbf{f}(x, \mathbf{y}(x)), \\ \mathbf{y}(x_0) &= \mathbf{y}_0. \end{aligned} \tag{4.4}$$

4.2 Solution methods

For problems arising in practice, it is generally not possible to find explicit solutions to the IVP. Instead, one approximates the solution using various numerical methods. One fundamental class, which we will present in the following, is based on *discrete variables*.

Consider the initial value problem given in Eq. (4.4) on an interval $x \in [a, b]$. This interval can be divided by a set of mesh points $\{x_0, x_1, ...\}$ with mesh spacing h between them. In the general case, where the mesh points are separated by unequal spacings $\{h_1, h_2, \dots\}$, we have that

$$\begin{aligned} x_0 &= a, \\ x_{i+1} &= x_i + h_i, \qquad i = 0, 1, 2, \dots \end{aligned}$$

In order to solve the initial value problem (4.4) numerically, the solution $\mathbf{y}(x)$ is approximated at each mesh point x_i:

$$\mathbf{y}_i \equiv \mathbf{y}(x_i).$$

4.2.1 One-step methods

The simplest methods of this discrete type are so-called *one-step methods*, where the value of $\mathbf{y}_{i+1}$ is computed only using $\mathbf{y}_i$, but no other preceding values. One possibility, for example, is to employ a Taylor series, such that

$$\mathbf{y}_{i+1} = \mathbf{y}_i + h_i \mathbf{f}(x_i, \mathbf{y}_i) + \frac{h^2}{2}\mathbf{f}^{(1)}(x_i, \mathbf{y}_i) + \cdots + \frac{h^p}{p!}\mathbf{f}^{(p-1)}(x_i, \mathbf{y}_i). \tag{4.5}$$

For $p = 1$, this yields the well-known method of Euler,

$$\mathbf{y}_{i+1} = \mathbf{y}_i + h_i \mathbf{f}(x_i, \mathbf{y}_i).$$

Runge-Kutta methods

One disadvantage of classical Taylor-series methods is that they require exact formal differentiation of $\mathbf{f}(x, \mathbf{y})$, which may be very inefficient or even impossible. Instead of computing these derivatives formally, the family of Runge-Kutta methods aims to produce an approximation

to the Taylor-series by evaluating $\mathbf{f}(x, \mathbf{y})$ at values between x_i and x_{i+1}. In general, one sets

$$\mathbf{k}_j = \mathbf{f}(x_i + \alpha_j h_i, \mathbf{y} + h_i \sum_{l=1}^{j-1} \beta_{jl} \mathbf{k}_l), \qquad j = 1, 2, \ldots J$$

$$\mathbf{y}_{i+1} = \mathbf{y}_i + h_i \sum_{j=1}^{J} \gamma_k \mathbf{k}_j.$$

The constants $\alpha_j, \beta_{jl}, \gamma_j$ are chosen in such a way, that the series expansion of $\mathbf{y}_{i+1}$ matches the Taylor-series expansion to as high a degree as possible, at the same time aiming at small computational complexity.

The most widely used Runge-Kutta method is of fourth order, with the usual form

$$\begin{aligned} \mathbf{k}_1 &= \mathbf{f}(x_i, \mathbf{y}_i), \\ \mathbf{k}_2 &= \mathbf{f}(x_i + \frac{1}{2}h_i, \mathbf{y}_i + \frac{1}{2}h_i \mathbf{k}_1), \\ \mathbf{k}_3 &= \mathbf{f}(x_i + \frac{1}{2}h_i, \mathbf{y}_i + \frac{1}{2}h_i \mathbf{k}_2), \\ \mathbf{k}_4 &= \mathbf{f}(x_i + h_i, \mathbf{y}_i + h_i \mathbf{k}_3), \\ \mathbf{y}_{i+1} &= \mathbf{y}_i + \frac{1}{6}h_i(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4). \end{aligned} \tag{4.6}$$

From a geometrical point of view, the derivative is evaluated at four points: once at the initial point, two times at trial midpoints at $x_{i+1/2}$ and once at the trial endpoint at x_{i+1}. All four derivatives are part of one single Runge-Kutta step, yielding the final value of $\mathbf{y}_{i+1}$. Since $\mathbf{k}$ can be interpreted as slope used to predict solutions of $\mathbf{y}$ that are afterwards corrected, the Runge-Kutta methods belong to the so-called *predictor-corrector methods*. Compared to Euler's method, which runs with a mathematical truncation of $\mathcal{O}(h)$, fourth-order Runge-Kutta has a global truncation error which goes like $\mathcal{O}(h^4)$.

4.2.2 Multi-step methods

Although the one-step methods give fairly good results, they do only use the information provided by the last point, and do not use the ones before. As an improvement, other methods have been developed, so-called *multi-step methods*. One class of these methods are *Adams methods*, which we will present in the following.

Adams methods

Rigorously, any solution of Eq. (4.4) can be written as

$$\mathbf{y}_{i+1} = \mathbf{y}_i + \int_{x_i}^{x_{i+1}} \frac{d\mathbf{y}}{dt} dt = \mathbf{y}_i + \int_{x_i}^{x_{i+1}} \mathbf{f}(t, \mathbf{y}(t)) dt. \tag{4.7}$$

Adams methods are based on the idea of approximating the integrand with a polynomial on the interval (x_i, x_{i+1}). A polynomial of order k results in a $(k+1)$th order method.

The algorithm of Adams consists of two parts:

- A *starting procedure* provides the approximate solutions $\mathbf{y}_1, \ldots, \mathbf{y}_{k-1}$ at points $x_1, \ldots, x_{k-1}$.
- A multi-step formula is used to obtain $\mathbf{y}_k$. One can then proceed recursively to obtain $\mathbf{y}_{k+1}, \mathbf{y}_{k+2}, \ldots$, based on numerical approximation of the k previous successive steps.

To obtain the missing starting points, there exist several possibilities. One way is to employ a Taylor-series expansion, as done by the developer of the method, J.C. Adams, himself. Another possibility is the use of any one-step method, for instance a Runge-Kutta method.
The Adams methods exist in two types, the explicit type (*Adams-Bashforth)* and the implicit type (*Adams-Moulton).* While explicit methods calculate functions at later points from previous ones only, implicit methods find a solution by solving equations involving both previous and successive points.

Adams-Bashforth method The explicit Adams-Bashforth method is one of the multi-step methods to solve the IVP (4.4). Assuming that the k preceding points $\mathbf{y}_i, \mathbf{y}_{i-1}, \ldots, \mathbf{y}_{i-k+1}$ are known, the values

$$\mathbf{f}_n = \mathbf{f}(x_n, \mathbf{y}_n), \qquad \text{for } n = i-k+1, \ldots, i$$

are available, too. That way, the function $\mathbf{f}(t, \mathbf{y}(t))$ in Eq. (4.7) can be replaced by the interpolation polynomial through the points $\{(t_n, \mathbf{f}_n) | n = i-k+1, \ldots, i\}$. Employing Newton's interpolation formula, this polynomial can be expressed as follows:

$$\mathbf{p}(t) = \mathbf{p}(t_i + sh) = \sum_{j=0}^{k-1} (-1)^j \binom{-s}{j} \nabla^j \mathbf{f}_i, \tag{4.8}$$

with the relations

$$\nabla^0 \mathbf{f}_i = \mathbf{f}_i, \qquad \nabla^{j+1} \mathbf{f}_i = \nabla^j \mathbf{f}_i - \nabla^j \mathbf{f}_{i-1}.$$

Thus the practically used analogue to Eq. (4.7) is given by

$$\begin{aligned} \mathbf{y}_{i+1} &= \mathbf{y}_i + \int_{x_i}^{x_{i+1}} \mathbf{p}(t) dt \\ &= \mathbf{y}_i + \sum_{j=0}^{k-1} \gamma_j \nabla^j \mathbf{f}_i, \end{aligned} \tag{4.9}$$

where one usually takes the same step length h for k consecutive points, and the coefficients γ_j satisfy

$$\gamma_j = (-1)^j \int_0^1 \binom{-s}{j} dt.$$

For $k = 1$, one obtains the explicit Euler method, $\mathbf{y}_{i+1} = \mathbf{y}_i + h\mathbf{f}_i$.

Adams-Moulton method The explicit Adams method is based on integrating the interpolation polynomial (4.8) from x_i to x_{i+1}, which means outside the interpolation interval (x_{i-k+1}, x_i). However, usually an interpolation is a rather poor approximation outside this interval.
The algorithm of Shampine and Gordon[1], that is used in this thesis, therefore bases on the implicit Adams-Moulton method, where the interpolation polynomial (4.8) uses an additional point $(x_{i+1}, \mathbf{f}_{i+1})$. This suggests

$$\mathbf{p}^*(t) = \mathbf{p}^*(t_i + sh) = \sum_{j=0}^{k} (-1)^j \binom{-s+1}{j} \nabla^j \mathbf{f}_{i+1}. \tag{4.10}$$

Equation (4.7) can then be approximated by

$$\mathbf{y}_{i+1} = \mathbf{y}_i + h \sum_{j=0}^{k} \gamma_j^* \nabla^j \mathbf{f}_{i+1}, \tag{4.11}$$

with coefficients

$$\gamma_j^* = (-1)^j \int_0^1 \binom{-s+1}{j} ds.$$

The formulas obtained with Eq. (4.11) are of the form

$$\mathbf{y}_{i+1} = \mathbf{y}_i + h(\beta_k \mathbf{f}_{i+1} + \cdots + \beta_0 \mathbf{f}_{i-k+1}), \tag{4.12}$$

with $\sum_i \beta_i = 1$. The first examples are:

$$\begin{aligned}
k = 0: &\quad \mathbf{y}_{i+1} = \mathbf{y}_i + h\mathbf{f}_{i+1} \\
k = 1: &\quad \mathbf{y}_{i+1} = \mathbf{y}_i + h\left(\frac{1}{2}\mathbf{f}_{i+1} + \frac{1}{2}\mathbf{f}_i\right) \\
k = 2: &\quad \mathbf{y}_{i+1} = \mathbf{y}_i + h\left(\frac{5}{12}\mathbf{f}_{i+1} + \frac{8}{12}\mathbf{f}_i - \frac{1}{12}\mathbf{f}_{i-1}\right) \\
k = 3: &\quad \mathbf{y}_{i+1} = \mathbf{y}_i + h\left(\frac{9}{24}\mathbf{f}_{i+1} + \frac{19}{24}\mathbf{f}_i - \frac{5}{24}\mathbf{f}_{i-1} + \frac{1}{14}\mathbf{f}_{i-1}\right).
\end{aligned}$$

The special case $k = 0$ corresponds to the *implicit Euler method*, and the case $k = 1$ to the *trapezoidal rule*. Both methods are actually one-step methods.

In general, Eq. (4.11) gives a more accurate approximation to the exact solution than Eq. (4.9) does and is subject to less numerical instability for relatively large values of the step length. However, these benefits bring the disadvantage that $\mathbf{y}_{i+1}$ is only implicitly defined, which in general results in non-linear equations for each step. To solve these equations, J.C. Adams himself used Newton's method, which is still done when encountering stiff equations [43]. Another possibility are predictor-corrector methods, which we already mentioned in the previous section. For the Adams-Moulton method, the algorithm can be summarized as follows:

P: Using the explicit Adams method (4.9), compute a reasonable approximation to $\mathbf{y}_{i+1}$:

$$\tilde{\mathbf{y}}_{i+1} = \mathbf{y}_i + h \sum_{j=0}^{k-1} \gamma_j \nabla^j \mathbf{f}_i.$$

[1] See subsection 8.4.3.

E: At this point x_{i+1}, evaluate the derivative $\tilde{\mathbf{f}}_{i+1} = \mathbf{f}(x_{i+1}, \tilde{\mathbf{y}}_{i+1})$.

C: Apply the corrector formula

$$\mathbf{y}_{i+1} = \mathbf{y}_i + h(\beta_k \tilde{\mathbf{f}}_{i+1} + \beta_{k-1}\mathbf{f}_i + \cdots + \beta_0 \mathbf{f}_{i-k+1})$$

to obtain the final point $\mathbf{y}_{i+1}$.

E: Evaluate the derivative again: $\mathbf{f}_{i+1} = f(x_{i+1}, \tilde{\mathbf{y}}_{i+1})$.

This most common procedure, denoted by PECE, is also used by Shampine and Gordon's algorithm of this thesis. Other often encountered versions are PECECE, with two fixed-point iterations per step, and PEC, where subsequent steps use $\tilde{\mathbf{f}}_{i+1}$ instead of $\mathbf{f}_{i+1}$.

Due to the variable order of the Adams methods, they are the method of choice if accuracy is needed over a wide range of tolerances. Moreover, they outperform classical one-step methods when output at many points is needed and function evaluations are expensive. On the other hand, if moderate accuracy is required and memory for storage is limited, Runge-Kutta methods are usually preferred.

Chapter 5

The modelled system

In this thesis, we apply our many-body methods to quantum dots, which are electrons confined in a potential. In accordance to popular manufacturing techniques, we use the common model of a two-dimensional parabolic quantum dot, where N electrons are confined in an isotropic harmonic oscillator potential in two spatial dimensions.
This chapter serves to set up the mathematical framework for the description of those quantum dots, and introduces in particular the employed Hamiltonian and model space.

5.1 The model Hamiltonian

As discussed in chapter 3, we can express the Hamiltonian as a sum of a non-interacting and an interaction part. The non-interacting part is given as a sum of the single-particle Hamiltonians $\hat{h}^{(0)}$,

$$\hat{H}_0 = \sum_i \hat{h}_i^{(0)} = \sum_i \left(-\frac{\hbar^2}{2m}\nabla_i^2 + \hat{v}_i \right), \tag{5.1}$$

and composed of a kinetic energy and a potential energy part. The specific expression depends on the choice of the external potential $\hat{v}_i$.

5.1.1 Harmonic oscillator basis

In our case, the external potential is a harmonic oscillator one, which in two dimensions reads

$$\hat{v} = \frac{1}{2}m\omega^2\mathbf{r}^2 = \frac{1}{2}m\omega^2(x^2 + y^2). \tag{5.2}$$

The mass m is given by $m = m^* m_e$, where m^* relates the bare electron mass to an effective mass. The parameter ω denotes the oscillator frequency and is related to the trap size a by $\omega = \hbar/ma^2$, with the latter being our length unit.

As derived in chapter 2, the eigenfunctions of $\hat{h}^{(0)}$, specified by

$$\hat{h}^{(0)}\phi_n = \epsilon_n\phi_n,$$

are given as product of the one-dimensional solution in each dimension,

$$\phi_n \equiv \phi_{n_x,n_y} = \phi_{n_x}\phi_{n_y}.$$

Inserting the expression derived in Eq. (2.58), we obtain

$$\phi_{n_x,n_y}(x,y) = \sqrt{\frac{m\omega}{\pi\hbar}}(2^{(n_x+n_y)}n_x!\, n_y!)^{-\frac{1}{2}} H_n\left(\frac{m\omega}{\hbar x}\right) H_n\left(\frac{m\omega}{\hbar y}\right) e^{-\frac{m\omega}{2\hbar}(x^2+y^2)}, \tag{5.3}$$

where H_n denotes the nth Hermite polynomial. The corresponding eigenvalues have been derived in Eq. (2.63), such that

$$\epsilon_n \equiv \epsilon_{n_x,n_y} = \hbar\omega(n_x + n_y + 1). \tag{5.4}$$

Since the single-particle Hamiltonians $\hat{h}^{(0)}$ are invariant under orthogonal transformations, their eigenvalues do not depend on the chosen spatial coordinates. Considering that our chosen oscillator potential is spherically symmetric, i.e. the oscillator frequency ω is the same for both x- and y-dimension, it is common to express the single-particle orbitals in polar coordinates, instead of the Cartesian representation in Eq. (5.3).
For $d = 2$ dimensions, one obtains the so-called *Fock-Darwin orbitals* [44],

$$\phi_{n,m_l}(r,\theta) = \sqrt{\frac{2n!}{(n+|m_l|)!}}\frac{e^{im_l\theta}}{\sqrt{2\pi}} L_n^{|m_l|}(r^2)e^{-r^2/2}, \tag{5.5}$$

where n is the nodal quantum number ($n = 0, 1, 2, \dots$) and equals the number of nodes in the radial part, m_l is the orbital angular momentum quantum number ($m_l = 0, \pm1, \pm2, \dots$), and $L_n^{|m_l|}(x)$ are *generalized Laguerre polynomials*. These ones are related to Laguerre polynomials as follows:

$$L_{q-p}^{p}(x) = (-1)^p \frac{d^p}{dx^p} L_q(x), \tag{5.6}$$

where the latter ones are polynomial sequences, defined by

$$L_q(x) = e^x \frac{d^q}{dx^q}\left(e^{-x}x^q\right). \tag{5.7}$$

For further properties, we refer to [31, 32]. The Fock-Darwin orbitals are by construction eigenfunctions of the angular-momentum operator $L_z = -i\frac{\partial}{\partial\theta}$ with eigenvalue m_l. This gives the advantage of having a basis that is diagonal in angular momentum. With quantum numbers $\{n, m_l\}$ instead of $\{n_x, n_y\}$, the eigenvalues of the single-particle Hamiltonians are given by

$$\epsilon_{n,m_l} = \hbar\omega(2n + |m_l| + 1). \tag{5.8}$$

Since the single-particle orbitals ϕ_{n,m_l} are denumerable, we can choose an ordering and identify each orbital with an integer. Recalling from Eq. (3.9) that each single-particle state is modelled as product of spatial and spin part, we must take into account that each orbit ϕ_{n,m_l} can be occupied by two particles, and specify the first one to have spin down, $m_s = -\frac{1}{2}$, and the second one to have spin up, $m_s = +\frac{1}{2}$. With this convention, our single-particle states can be labelled as shown in figure 5.1.
Each index i defines a specific set of quantum numbers $\{n, m_l, m_s\}$, where the relation between

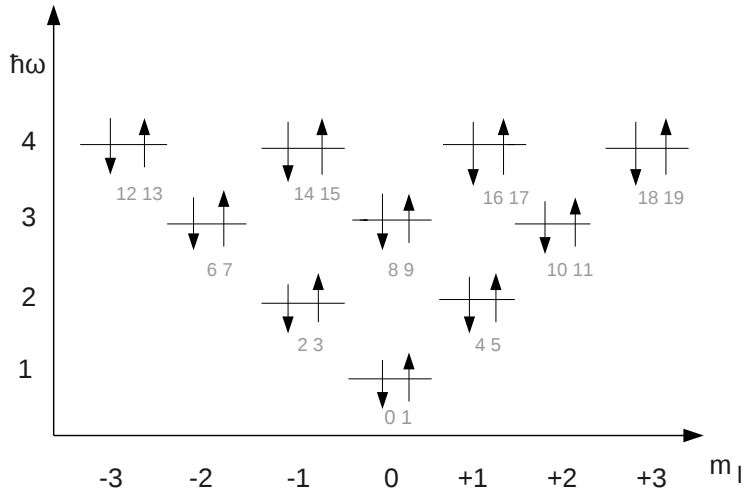

Figure 5.1: Labelling of the single-particle states of the two-dimensional harmonic oscillator. The x-axis shows the angular momentum quantum number, the y-axis the energy in units of $\hbar\omega$. Each energy level defines a shell R, with degenerate eigenvalues ϵ_i for the corresponding single-particle states. The shown arrangement is also referred to as *shell structure*.

R	Degeneracy	Shell filling
1	2	2
2	4	6
3	6	12
4	8	20
5	10	30

Table 5.1: Degeneracy for the first 5 shells and number of particles it takes to have a closed-shell system up to this level.

energy and quantum numbers n and m_l is determined by Eq. (5.8). All eigenfunctions with the same energy span a so-called *shell*. We define the shell number R as[1]

$$R = 2n + |m_l| + 1. \tag{5.9}$$

Each shell corresponds to an energy $\hbar\omega R$, and the degeneracy of each shell is given by $2R$, where the factor of 2 accounts for spin. If all single-particle states up to a certain shell are occupied, one has so-called *closed-shell systems*.
These are the systems we will consider in this thesis, suggesting that our number of particles is restricted to $2, 6, 12, 20, \ldots$ The outstanding feature of closed-shell systems is their symmetry, since none of the degenerate energy levels is in a way more or less preferred.

To choose the eigenfunctions of the single-particle Hamiltonian as basis is a natural starting point with respect to the underlying physics, since the interaction can be considered as perturbation from the non-interacting case. Moreover, the basis allows an easy symmetrization of the many-fermion wave function and results in a fast convergence of the ground state energy as function of the basis size [18].

5.1.2 Choice of interaction

We model the interaction between the electrons by a two-body Coulomb potential,

$$v(r_{ij}) = \frac{e^2}{4\pi\epsilon_0\epsilon}\frac{1}{r_{ij}},$$

where $r_{ij} = \|\mathbf{r}_i - \mathbf{r}_j\|$ denotes the distance between two particles, ϵ the dielectric constant of the semiconductor, and $e^2/4\pi\epsilon_0 \approx 1.440$ eV $\cdot$ nm.
For typical GaAs quantum dots, we have that $a = 20$ nm, $m^* = 0.067$ and $\epsilon = 12.4$. In the following, we will use effective atomic units, setting $\hbar = m_e = e = 1/4\pi\epsilon_0 = 1$. This system of units is especially convenient for atomic physics calculations. On the one hand, it simplifies the equations, on the other hand it avoids orders of magnitude that give rise to numerical truncation and round-off errors.
The unit of energy is then Hartrees[2] times m^*/ϵ^2, which corresponds to effective Hartrees $E_h^* \approx 11.86$ meV. The oscillator frequency ω is given in units of $1.79 \times 10^{13}\,\mathrm{s}^{-1}$.
In this dimensionless form, our total Hamiltonian reads

$$\hat{H} = \sum_{i=1}^{N}\left(-\frac{1}{2}\nabla_i^2 + \frac{1}{2}\omega^2 r_i^2\right) + \sum_{i<j}\frac{1}{r_{ij}}. \tag{5.10}$$

5.2 Model space

The basis functions of the N-particle Hilbert space are Slater determinants, as defined in Eq. (3.8). Since there in principle exist infinitely many single-particle states for the N parti-

[1] Note that another common convention is to define the shell number as $R = 2n + |m_l|$. However, to directly compare our results with Ref. [13] and theses of other current and previous master's students, we use the definition used in those works.

[2] One Hartree is given by $E_h = \frac{m_e e^4}{(4\pi\epsilon_0\hbar)^2} = 4.359744 \times 10^{-18} J = 27.21138$ eV.

cles to occupy, the number of possible Slater determinants is infinite, too. For numerical calculations, one therefore uses a finite-dimensional subspace $\mathcal{P} \subset \mathcal{H}$ of the full Hilbert space $\mathcal{H}$, called *model space*. This model space has a basis $\mathcal{B}$ with a finite number of Slater determinants, and its orthogonal projector is given by

$$P = \sum_{|\Phi_b\rangle \in \mathcal{B}} |\Phi_b\rangle\langle\Phi_b|. \tag{5.11}$$

In practice, one truncates the number of included single-particle states by specifying a maximal shell number R, such that the N particles can maximally occupy $n = R(R+1)$ states:

$$\mathcal{B} = \mathcal{B}_R = \left\{ |\Phi_b\rangle : \max_{i=1\ldots N} R_i \leq R \right\}, \tag{5.12}$$

The shell R is also called energy-cut[3]. As $R \to \infty$, the whole Hilbert space is spanned, such that the eigenvalues of $P\hat{H}P$ converge to the ones of $\hat{H}$.

5.3 Symmetries of the Hamiltonian

Our Hamiltonian exhibits several symmetries, making it possible to reduce the complexity of the computations. First of all, we have that

$$[\hat{H}, \hat{L}_z] = [\hat{H}, \hat{S}_z] = 0. \tag{5.13}$$

Here $\hat{L}_z$ is the angular momentum operator with eigenvalue[4] $M = \sum_{i=1}^{N} m_i$:

$$\sum_{i=1}^{N} \hat{L}_z(i)|\Phi\rangle = M|\Phi\rangle. \tag{5.14}$$

The spin projection operator $\hat{S}_z$ is given by

$$\hat{S}_z = \frac{1}{2} \sum_{p,\sigma} \sigma a^\dagger_{p,\sigma} a_{p,\sigma}, \tag{5.15}$$

with eigenvalues $M_s = \sum_{i=1}^{N} \sigma_i/2$. Since the Slater determinants are eigenvectors of both $\hat{L}_z$ and $\hat{S}_z$, our model space $\mathcal{P}$ is naturally split into subspaces with constant angular momentum M and spin projection M_s. In other words, the Hamiltonian is block-diagonal in M and M_s, such that a diagonalization of $\hat{H}$ can be done within each block separately. In particular, when interested in the ground state energy only, it is sufficient to set up the block with $M = M_S = 0$ and diagonalize that one, which reduces the computational effort enormously.

[3] Another common practice is to cut the global shell number instead of the single-particle shell number, $\mathcal{B}_R = \left\{ |\Phi_b\rangle : \sum_{i=1}^{N} R_i \leq R \right\}$, see [44].

[4] Since we are now using dimensionless units and not dealing with masses m any more, which could cause confusion, we take the widely used notation $m_l \to m$ for the azimuthal quantum number in the remainder of this thesis.

In principle, the matrix blocks could be even smaller, since our Hamiltonian (5.10) also commutes with the total spin $\hat{S}^2$, given by

$$\hat{S}^2 = \hat{S}_z^2 + \frac{1}{2}(\hat{S}_+\hat{S}_- + \hat{S}_-\hat{S}_+). \tag{5.16}$$

The spin raising and lowering operators, $\hat{S}_+$ and $\hat{S}_-$ respectively, are defined as

$$\hat{S}_\pm = \hat{S}_x \pm i\hat{S}_y = \sum_p a^\dagger_{p\pm} a_{p\mp}. \tag{5.17}$$

However, since the Slater determinants are not eigenfunctions of $\hat{S}^2$, we would have to take a linear combination of them to obtain block-diagonality in the total spin [18]. For simplicity, we therefore restrict us to identities (5.13).

Part II

METHODS

Chapter 6

The Similarity Renormalization Group method

In this chapter, we introduce the similarity renormalization group (SRG) method. In the first two sections, we expose the general ideas and formalisms of the method. The subsequent parts discuss two different realizations: first an application to free space, where the Hamiltonian is set up with respect to the physical vacuum state, and afterwards we explain how to model SRG in a many-body medium, making use of the technique of normal-ordering.

6.1 General aspects

The SRG method was introduced independently by Glazek and Wilson [45, 46] and Wegner [47, 48] as a new way to implement the principle of energy scale separation. While Glazek and Wilson developed it under the name *similarity renormalization scheme* in the context of high-energy physics, Wegner evolved it under the name *flow equations* in the context of condensed matter theory.
The SRG method uses a continuous sequence of unitary transformations to decouple the high- and low-energy matrix elements of a given interaction, thus driving the Hamiltonian towards a band- or block-diagonal form.

Let us consider the initial Hamiltonian

$$\hat{H} = \hat{H}^{\mathrm{d}} + \hat{H}^{\mathrm{od}},$$

where $\hat{H}^{\mathrm{d}}$ and $\hat{H}^{\mathrm{od}}$ denote its "diagonal" and "off-diagonal" parts, namely

$$\langle i|\, \hat{H}^{\mathrm{d}}\, |j\rangle \equiv \begin{cases} \langle i|\, \hat{H}\, |i\rangle & \text{if } i = j, \\ 0 & \text{otherwise,} \end{cases}$$

and similarly

$$\langle i|\, \hat{H}^{\mathrm{od}}\, |j\rangle \equiv \begin{cases} \langle i|\, \hat{H}\, |j\rangle & \text{if } i \neq j, \\ 0 & \text{otherwise.} \end{cases}$$

Introducing a flow parameter s, there exits a unitary transformation U_s, such that

$$\hat{H}_s = U_s \hat{H} U_s^\dagger \equiv \hat{H}_s^{\mathrm{d}} + \hat{H}_s^{\mathrm{od}}, \tag{6.1}$$

with the relations $U_{s=0} = \mathbf{1}$, and $\hat{H}_{s=0} = \hat{H}$. In particular, U_s is parametrized as

$$\begin{aligned} U_s &= T_s \exp\left(\int_0^s ds' \hat{\eta}_{s'}\right) \\ &= 1 + \sum_{n=1}^{\infty} \frac{1}{n!} \int_0^s ds_1 \dots ds_n T_s(\hat{\eta}_{s_1} \dots \hat{\eta}_{s_n}), \end{aligned}$$

where we introduced the anti-hermitian operator $\hat{\eta}_s$ as so-called *generator* of the transformation. With T_s we denote s-ordering, which is defined equivalently to usual time-ordering: The generator $\hat{\eta}_{s_i}$ with the smallest s_i is permuted to the right, the one with next smallest s_i one step further left etc. [49]. This suggests that

$$T_s(\hat{\eta}_{s_1} \dots \hat{\eta}_{s_n}) \equiv \hat{\eta}_{s_{\pi(1)}} \dots \hat{\eta}_{s_{\pi(n)}},$$

with permutations $\pi \in S_n$ such that $s_{\pi(1)} \geq s_{\pi(2)} \geq \dots \geq s_{\pi(n)}$. Taking the derivative of $\hat{H}_s$ with respect to s gives

$$\frac{d\hat{H}_s}{ds} = \frac{dU_s}{ds} \hat{H} U_s^\dagger + U_s \hat{H} \frac{dU_s^\dagger}{ds}. \tag{6.2}$$

Utilizing that for our particular form of U_s, we have that

$$\hat{\eta}_s = \frac{dU_s}{ds} U_s^\dagger = -U_s \frac{dU_s^\dagger}{ds} = -\hat{\eta}_s, \tag{6.3}$$

we obtain that

$$\frac{d\hat{H}_s}{ds} = \hat{\eta}_s \hat{H}_s - \hat{H}_s \hat{\eta}_s = \left[\hat{\eta}_s, \hat{H}_s\right]. \tag{6.4}$$

This is the key expression of the SRG method, describing the flow of the Hamiltonian.

6.2 Choice of generator

The specific unitary transformation is determined by the choice of the generator. Through different choices of $\hat{\eta}_s$, the SRG evolution can be adapted to the features of a particular problem.

6.2.1 Canonical generator

The original choice for $\hat{\eta}_s$, suggested by Wegner [47], reads

$$\hat{\eta}_s = \left[\hat{H}_s^{\mathrm{d}}, \hat{H}_s\right] = \left[\hat{H}_s^{\mathrm{d}}, \hat{H}_s^{\mathrm{od}}\right],$$

and has been extensively applied in condensed matter physics. As commutator between two Hermitian operators, $\hat{\eta}_s$ fulfils the criterion of antihermiticity and can be shown to suppress the off-diagonal matrix elements, provided that the two conditions

$$\mathrm{Tr}\left(\hat{H}_s^{\mathrm{d}} \hat{H}_s^{\mathrm{od}}\right) = 0 \tag{6.5}$$

and

$$\mathrm{Tr}\left(\frac{d\hat{H}_s^{\mathrm{d}}}{ds}\hat{H}_s^{\mathrm{od}}\right) = 0 \tag{6.6}$$

are met [49]. Due to its many successful applications in condensed matter and nuclear physics, it will be one of the choices in this thesis, too.
However, other choices are possible and might exhibit better numerical and computational efficiency. In our case of quantum dots, the initial Hamiltonian is given in the center-of-mass frame,

$$\hat{H} = \hat{T}_{\mathrm{rel}} + \hat{V},$$

where $\hat{T}_{\mathrm{rel}} = \sum_i \hat{t}_i$ is the relative kinetic energy and $\hat{V} = \sum_{i<j} \hat{v}_{ij}$ describes the interaction part. In this case it seems desirable to express the unitary transformation as

$$\hat{H}_s = U_s \hat{H} U_s^\dagger = \hat{T}_{\mathrm{rel}} + \hat{V}_s,$$

which means that all dependence on the flow parameter s is stored in the potential part of the Hamiltonian and $\hat{T}_{\mathrm{rel}}$ is constant during the whole computation.

A simple generator which fulfils this criterion and eliminates the off-diagonal elements during the flow is

$$\hat{\eta}_s = \left[\hat{T}_{\mathrm{rel}}, \hat{H}_s\right].$$

Since $\hat{T}_{\mathrm{rel}}$ obviously commutes with itself, this is equivalent to

$$\hat{\eta}_s = \left[\hat{T}_{\mathrm{rel}}, \hat{V}_s\right]. \tag{6.7}$$

This generator has been successfully applied in nuclear physics [28–30, 50], too, and in section 9.1, we compare its effect on the flow equations with Wegner's canonical generator.

6.2.2 White's generator

Apart from this canonical generator, there exist several other ones in literature. One of them is White's choice [51], which has been shown to make numerical approaches much more efficient.

The problem with Wegner's generator are the widely varying decaying speeds of the elements, removing first terms with large energy differences and then subsequently those ones with smaller energy separations. That way, the flow equations become a stiff set of coupled differential equations, and often a large number of integration steps is needed to obtain the result up to a desired precision.
White takes another approach, which is especially suited for problems where one is interested in the ground state of a system. Instead of driving all off-diagonal elements of the Hamiltonian to zero, he focuses solely on those ones that are connected to the reference state $|\Phi_0\rangle$, aiming to decouple the reference state from the remaining Hamiltonian. With a suitable transformation, the elements get similar decaying speeds, which solves the problem of stiffness of the flow equations.

To derive an expression for the generator, consider the Hamiltonian operator in second quantization

$$\hat{H} = \sum_{pq} \langle p| \hat{h}^{(0)} |q\rangle \, a_p^\dagger a_q + \frac{1}{4} \sum_{pqrs} \langle pq||rs\rangle \, a_p^\dagger a_q^\dagger a_s a_r + \dots \tag{6.8}$$

More generally, Eq. (6.8) can be rewritten as

$$\hat{H} = \sum_\alpha a_\alpha h_\alpha, \tag{6.9}$$

where h_α is a product of α creation and α annihilation operators, and a_α the corresponding coefficient. In Eq. (6.8), we thus have that $a_1 = \langle p| \hat{h}^{(0)} |q\rangle$ with $h_1 = a_p^\dagger a_q$, $a_2 = \frac{1}{4} \langle pq| |rs\rangle$ with $h_2 = a_p^\dagger a_q^\dagger a_s a_r$, etc.
Moreover, introducing the flow parameter s, Eq. (6.9) becomes

$$\hat{H}(s) = \sum_\alpha a_\alpha(s) h_\alpha. \tag{6.10}$$

According to White, the generator can be expressed in terms of $a_\alpha(s)$ and h_α the following way:

$$\hat{\eta}(s) = \sum_\alpha \hat{\eta}_\alpha(s) h_\alpha, \qquad \text{where} \quad \hat{\eta}_\alpha(s) = b_\alpha a_\alpha(s). \tag{6.11}$$

The b_α are fixed parameters ensuring that those $a_\alpha(s)$ that correspond to off-diagonal elements are driven to zero. For diagonal elements and others that are not connected to the reference state $|\Phi_0\rangle$, and are therefore not needed to be zeroed out, the parameter b_α is set to zero. For the remaining elements, b_α is chosen in such a way that all $a_\alpha(s)$ have approximately the same decaying speed.
White's suggestion is to set

$$b_\alpha = (E_l^\alpha - E_r^\alpha)^{-1}, \tag{6.12}$$

where

$$E_l^\alpha = \langle L^\alpha| \hat{H} |L^\alpha\rangle, \quad E_r^\alpha = \langle R^\alpha| \hat{H} |R^\alpha\rangle.$$

The so-called *left state* $|L^\alpha\rangle$ and *right state* $|R^\alpha\rangle$ are defined as that pair of states fulfilling $\langle L^\alpha| \hat{V} |R^\alpha\rangle \neq 0$ that is closest to the reference state $|\Phi_0\rangle$. In order to specify this statement, we introduce the quasi-particle operators d and $d^\dagger$, satisfying

$$d_p |\Phi_0\rangle = 0, \qquad \forall p. \tag{6.13}$$

They are related to the standard creation and annihilation operators $a_p^\dagger$ and a_p in such a way that $d_p^\dagger = a_p^\dagger$ for an unoccupied state p and $d_p^\dagger = a_p$ for an occupied state p. That way, they satisfy the same anticommutation relations

$$\begin{aligned} \{d_p^\dagger, d_q^\dagger\} = \{d_p, d_q\} &= 0 \\ \{d_p^\dagger, d_q\} &= \delta_{p,q}. \end{aligned}$$

With those operators, the above mentioned pair of states $|L^\alpha\rangle$ and $|R^\alpha\rangle$ closest to $|\Phi_0\rangle$ can be specified as that pair having the fewest number of quasi-particle creation operators $d^\dagger$ acting on $|\Phi_0\rangle$.

As an example, consider

$$a_\alpha h_\alpha = a_\alpha d_i^\dagger d_j d_k d_l,$$

where a_α is the coefficient appearing in Eq. (6.9) and should not be confused with the standard annihilation operator. In this case, the states closest to the reference state $|\Phi_0\rangle$ that satisfy $\langle L^\alpha| \hat{V} |R^\alpha\rangle \neq 0$ are

$$\begin{aligned} |R^\alpha\rangle &= d_m^\dagger d_k^\dagger d_j^\dagger |\Phi_0\rangle \\ |L^\alpha\rangle &= d_i^\dagger |\Phi_0\rangle . \end{aligned}$$

6.3 Free-space SRG

The idea behind SRG in free space is to choose a basis with respect to the physical vacuum state, set up the full Hamiltonian matrix in this basis and solve Eq. (6.4) as a set of coupled first-order differential equations. As stated above, the specific expression for the derivatives depends on the choice of the generator η.
In order to get an idea of how the Hamiltonian is driven towards diagonal form with the SRG method, we start with the easy generator $\eta_1 = \left[\hat{T}_{\text{rel}}, \hat{V}\right]$. Note that whenever using this notation, we assume an implicit dependence on the flow parameter s, since $V = V(s)$.

Choosing an appropriate basis, where $\hat{T}_{\text{rel}}$ is diagonal with matrix elements ϵ_i corresponding to the single-particle energies, the matrix products simplify to

$$\eta_{ij}(s) = (\epsilon_i - \epsilon_j)\, V_{ij}(s). \tag{6.14}$$

With this expression for the elements of the generator, the flow of the Hamiltonian is computed as follows:

$$\begin{aligned} \frac{dH_{ij}(s)}{ds} &= \langle i| \left(\hat{\eta}_s \hat{H}_s - \hat{H}_s \hat{\eta}_s\right) |j\rangle \\ &= \langle i| \hat{\eta}_s \hat{T}_{\text{rel}} |j\rangle + \langle i| \hat{\eta}_s \hat{V}_s |j\rangle - \langle i| \hat{T}_{\text{rel}} \hat{\eta}_s |j\rangle - \langle i| \hat{V}_s \hat{\eta}_s |j\rangle \\ &= \epsilon_j \eta_{ij}(s) - \epsilon_i \eta_{ij}(s) + \langle i| \left(\hat{\eta}_s \hat{V}_s - \hat{V}_s \hat{\eta}_s\right) |j\rangle \\ &= -(\epsilon_i - \epsilon_j)\eta_{ij}(s) + \sum_k \left\{(\epsilon_i - \epsilon_k) V_{ik}(s) V_{kj}(s) - (\epsilon_k - \epsilon_j) V_{ik}(s) V_{kj}(s)\right\}, \end{aligned}$$

which finally gives

$$\frac{dH_{ij}(s)}{ds} = \frac{dV_{ij}(s)}{ds} = -(\epsilon_i - \epsilon_j)^2 V_{ij}(s) + \sum_k \left(\epsilon_i + \epsilon_j - 2\epsilon_k\right) V_{ik}(s) V_{kj}(s). \tag{6.15}$$

To obtain the same expression one frequently encounters in literature (e.g. [29]), one can rewrite this equation in the space of relative momentum k (with normalization $1 = \frac{2}{\pi}\int_0^\infty |q\rangle q^2 dq \langle q|$), which gives

$$\frac{dV_s(k,k')}{ds} = -(k^2 - k'^2)^2 V_s(k,k') + \frac{2}{\pi}\int_0^\infty q^2 dq (k^2 + k'^2 - 2q^2) V_s(k,q) V_s(q,k'). \tag{6.16}$$

Equations (6.15) and (6.16) represent a non-linear system of first-order coupled differential equations, with the initial condition that $\hat{V}_s$ equals the initial potential at the first value of the flow parameter s .

To get an idea of how the off-diagonal elements are suppressed, let us approximate the flow of the Hamiltonian in Eq. (6.15) by

$$\frac{dV_{ij}(s)}{ds} \approx -(\epsilon_i - \epsilon_j)^2 V_{ij}(s).$$

The solution can easily be obtained by integration and is given by

$$V_{ij}(s) \approx V_{ij}(0)\mathrm{e}^{-s(\epsilon_i-\epsilon_j)^2}. \tag{6.17}$$

Thus all off-diagonal elements with $i \neq j$ decrease to zero during the flow, with the energy difference $(\epsilon_i - \epsilon_j)$ controlling how fast a particular element is suppressed. Matrix elements far off the diagonal, where the Hamiltonian connects states with large energy differences, are in general suppressed much faster than elements closer to the diagonal.
Instead of measuring the progress of the flow in terms of s, it is convenient to do it in terms of the parameter $\lambda \equiv s^{-1/2}$, which provides at the same time a measure for the width of the diagonal band [47, 49]. While it is in principle required to go to $s = \infty$ ($\lambda = 0$) in order to obtain a diagonal Hamiltonian, Eq. (6.17) demonstrates that one in practice only needs to proceed until all off-diagonal elements are that small that the result does not change up to a certain tolerance. In this case, we say that convergence has been reached within the desired numerical accuracy.

The same argumentation holds for Wegner's original choice of the generator. The matrix elements of $\hat{\eta}$ are only slightly changed to

$$\eta_{ij} = (\epsilon_i + V_{ii} - \epsilon_j - V_{jj})\, V_{ij}^{od}, \tag{6.18}$$

with off-diagonal interaction elements $V_{ij}^{od} = V_{ij}$ for $i = j$ and $V_{ij}^{od} = 0$, otherwise. This yields the following flow equations:

$$\begin{aligned}\frac{dV_{ij}(s)}{ds} =& -(\epsilon_i + V_{ii}(s) - \epsilon_j - V_{jj}(s))^2\, V_{ij}(s) \\ &+ \sum_k (\epsilon_i + V_{ii}(s) + \epsilon_j + V_{jj}(s) - 2\epsilon_k - 2V_{kk}(s))\, V_{ik}^{od}(s) V_{kj}^{od}(s).\end{aligned} \tag{6.19}$$

In section 9.1, we analyse how this small difference between the two generators affects the results and numerical stability.

6.4 In-medium SRG

Instead of performing SRG in free space, the evolution can be done at finite density, i.e. directly in the A-body system [49]. This approach has recently been applied very successfully in nuclear physics [26, 27] and is called in-medium SRG (IM-SRG). The method allows the evolution of $3, ..., A$-body operators using only two-body machinery, with the simplifications arising from the use of normal-ordering with respect to a reference state.

To explain the concept, let us consider the Hamiltonian used in this thesis, a second-quantized Hamiltonian with maximally two-body interactions:

$$\hat{H} = \sum_{pq} \langle p| \hat{h}^{(0)} |q\rangle \, a_p^\dagger a_q + \frac{1}{4} \sum_{pqrs} \langle pq||rs\rangle \, a_p^\dagger a_q^\dagger a_s a_r. \tag{6.20}$$

As demonstrated in chapter 3, normal-ordering with respect to a reference state $|\Phi_0\rangle$ yields

$$\begin{aligned} \hat{H_N} &= \hat{F}_N + \hat{V}_N \\ &= \sum_{pq} f_{pq} \{a_p^\dagger a_q\} + \frac{1}{4} \sum_{pqrs} v_{pqrs} \{a_p^\dagger a_q^\dagger a_s a_r\}. \end{aligned}$$

The amplitudes for the one-body operator read

$$f_{pq} = \langle p| \hat{h}^{(0)} |q\rangle + \sum_i \langle pi||qi\rangle \,,$$

and the ones for the two-body operator

$$v_{pqrs} = \langle pq||rs\rangle \,.$$

As in chapter 3, we use the notation that indices $\{a, b, c, ...\}$ denote particles states above the Fermi level, $\{i, j, k, ...\}$ denote hole states below the Fermi level, and $\{p, q, r, ...\}$ can be used for both particle and hole states.
In relation to the full Hamiltonian, $\hat{H}_N$ is given by[1]

$$\hat{H}_N = \hat{H} - E_0,$$

where the energy expectation value between reference states, E_0, is

$$\begin{aligned} E_0 &= \langle \Phi_0| \hat{H} |\Phi_0\rangle \\ &= \sum_i \langle i| \hat{h}^{(0)} |i\rangle + \frac{1}{2} \sum_{ij} \langle ij| \, |ij\rangle \,. \end{aligned}$$

Exactly as in free space, we want to compute the flow equations

$$\frac{d\hat{H}_s}{ds} = \left[\hat{\eta}_s, \hat{H}_s\right].$$

The difference is that we now formulate the derivatives and generator $\hat{\eta}$ in the language of second quantization and normal-ordering, too.
With this procedure, one faces one of the major challenges of the SRG method, namely the generation of higher and higher order interaction terms during the flow. With each evaluation of the commutator, the Hamiltonian gains terms of higher order, and these induced contributions will in subsequent integration steps contribute to terms of lower order. In principle, this continues to infinity.
To make the method computationally possible, one is therefore forced to truncate the flow equations after a certain order. This affects of course the accuracy of the result, and the

[1] See explanations in chapter 3.

fewer orders one includes, the higher the truncation error is. On the other hand, improving the result is quite straightforward and means to include higher orders, too. If we included all generated terms until the result has converged within the desired accuracy, we would obtain the same results as with SRG in free space.

In this thesis we choose IM-SRG(2), which means that we truncate all operators on a two-body level. This normal-ordered two-body approximation seems to be sufficient in many cases and has yielded excellent results for several nuclei [27, 52, 53].
With this truncation, the generator $\hat{\eta}$ can be written as

$$\hat{\eta} = \sum_{pq} \eta^{(1)}_{pq} \{a^\dagger_p a_q\} + \frac{1}{4} \sum_{pqrs} \eta^{(2)}_{pqrs} \{a^\dagger_p a^\dagger_q a_s a_r\},$$

where $\eta^{(1)}_{pq}$ and $\eta^{(2)}_{pqrs}$ are the one- and two-body operator of the generator, respectively. Including up to two-body interactions, the flow equations are given by

$$\begin{aligned}
\frac{d\hat{H}_s}{ds} &= \left[\hat{\eta}_s, \hat{H}_s\right] \\
&= \left[\sum_{pq} \eta^{(1)}_{pq}(s) \{a^\dagger_p a_q\} + \frac{1}{4} \sum_{pqrs} \eta_{pqrs}(s)^{(2)} \{a^\dagger_p a^\dagger_q a_s a_r\}, \right. \\
&\qquad \left. \sum_{pq} f_{pq}(s) \{a^\dagger_p a_q\} + \frac{1}{4} \sum_{pqrs} v_{pqrs}(s) \{a^\dagger_p a^\dagger_q a_s a_r\} \right].
\end{aligned}$$

Using the commutation relations presented in Appendix A and collecting the constants in E_0, the one-body terms in f and the two-body terms in v, we obtain Eqs. (6.22)-(6.24), where we make use of the permutation operator

$$\hat{P}_{pq} f(p,q) = f(q,p). \tag{6.21}$$

The derivative of E_0 is given by

$$\frac{dE_0}{ds} = \sum_{ia} \left(\eta^{(1)}_{ia} f_{ai} - \eta^{(1)}_{ai} f_{ia} \right) + \frac{1}{2} \sum_{ijab} \eta^{(2)}_{ijab} v_{abij}, \tag{6.22}$$

the one of the one-body operator by

$$\begin{aligned}
\frac{df_{pq}}{ds} &= \sum_{r} \left(\eta^{(1)}_{pr} f_{rq} + \eta^{(1)}_{qr} f_{rp} \right) + \sum_{ia} \left(1 - \hat{P}_{ia} \right) \left(\eta^{(1)}_{ia} v_{apiq} - f_{ia} \eta^{(2)}_{apiq} \right) \\
&\quad + \frac{1}{2} \sum_{aij} \left(1 + \hat{P}_{pq} \right) \eta^{(2)}_{apij} v_{ijaq} + \frac{1}{2} \sum_{abi} \left(1 + \hat{P}_{pq} \right) \eta^{(2)}_{ipab} v_{abiq},
\end{aligned} \tag{6.23}$$

and the derivative of the two-body operator by

$$\begin{aligned}
\frac{dv_{pqrs}}{ds} &= \sum_{t} \left(1 - \hat{P}_{pq} \right) \left(\eta^{(1)}_{pt} v_{tqrs} - f_{pt} \eta^{(2)}_{tqrs} \right) - \sum_{t} \left(1 - \hat{P}_{rs} \right) \left(\eta^{(1)}_{tr} v_{pqts} - f_{tr} \eta^{(2)}_{pqts} \right) \\
&\quad + \frac{1}{2} \sum_{ab} \left(\eta^{(2)}_{pqab} v_{abrs} - v_{pqab} \eta^{(2)}_{abrs} \right) - \frac{1}{2} \sum_{ij} \left(\eta^{(2)}_{pqij} v_{ijrs} - v_{pqij} \eta^{(2)}_{ijrs} \right) \\
&\quad - \sum_{ia} \left(1 - \hat{P}_{ia} \right) \left(1 - \hat{P}_{pq} \right) \left(1 - \hat{P}_{rs} \right) \eta^{(2)}_{aqis} v_{ipar}.
\end{aligned} \tag{6.24}$$

Note that on the right-hand side the s-dependence of all the elements has been skipped for better readability. From a computational point of view, most of the work of the SRG method is to compute those derivatives for each integration step. Table 6.1 gives some examples of how in our case of quantum dots, the number of equations is quickly increasing with the number of shells R.

R	# eqs.
2	5
5	6531
10	624149
15	9600152
18	33187537
20	68138690

Table 6.1: Number of ordinary differential equations to be solved for $N = 2$ particles, as function of the number of shells R.

The commutator in the flow equations (6.4) guarantees that the IM-SRG wave function $U(s)|\Phi\rangle$ can be expanded in terms of linked diagrams only [25,54]. This suggests that IM-SRG is size-extensive. Regarding the quality of the SRG results, it means that the error introduced by truncating the many-body expansions scales linearly with the number of particles N.

6.4.1 IM-SRG(2) with Wegner's canonical generator

To determine the specific form of the equations, one needs to specify the concrete generator $\hat{\eta}$. As introduced in section 6.2, one possibility is Wegner's generator

$$\hat{\eta} = \left[\hat{H}^{\mathrm{d}}, \hat{H}^{\mathrm{od}}\right] = \left[\hat{H}^{\mathrm{d}}, \hat{H}\right]. \tag{6.25}$$

In second quantization, we thereby have to compute

$$\hat{\eta} = \left[\sum_{pq} f^d_{pq}\{a^\dagger_p a_q\} + \frac{1}{4}\sum_{pqrs} v^d_{pqrs}\{a^\dagger_p a^\dagger_q a_s a_r\}, \sum_{pq} f_{pq}\{a^\dagger_p a_q\} + \frac{1}{4}\sum_{pqrs} v_{pqrs}\{a^\dagger_p a^\dagger_q a_s a_r\}\right],$$

where the amplitudes of the diagonal Hamiltonian are defined as

$$f^d_{pq} = f_{pq}\delta_{pq}, \qquad v^d_{pqrs} = v_{pqrs}\left(\delta_{pr}\delta_{qs} + \delta_{ps}\delta_{qr}\right). \tag{6.26}$$

Using the commutation relations presented in Appendix A, we collect the first-order terms in $\hat{\eta}^{(1)}$ and the second-order terms in $\hat{\eta}^{(2)}$. Using the standard notation

$$n_p = \begin{cases} 1, & \text{if } p < \epsilon_F \quad (p \text{ is hole state}) \\ 0, & \text{if } p > \epsilon_F \quad (p \text{ is particle state}), \end{cases}$$

the corresponding matrix elements are

$$\eta^{(1)}_{pq} = \sum_r \left(f^d_{pr} f_{rq} - f_{pr} f^d_{rq} \right) + f_{pq} v^d_{qppq} \left(n_q - n_p \right) \tag{6.27}$$

$$\begin{aligned} \eta^{(2)}_{pqrs} =& - \sum_t \left\{ \left(1 - \hat{P}_{pq} \right) f_{pt} v^d_{tqrs} - \left(1 - \hat{P}_{rs} \right) f_{tr} v^d_{pqts} \right\} \\ & + \sum_t \left\{ \left(1 - \hat{P}_{pq} \right) f^d_{pt} v_{tqrs} - \left(1 - \hat{P}_{rs} \right) f^d_{tr} v_{pqts} \right\} \\ & + \frac{1}{2} \sum_{tu} (1 - n_t - n_u) \left(v^d_{pqtu} v_{turs} - v_{pqtu} v^d_{turs} \right) \\ & + \sum_{tu} \left(n_t - n_u \right) \left(1 - \hat{P}_{pq} \right) \left(1 - \hat{P}_{rs} \right) v^d_{tpur} v_{uqts} \end{aligned} \tag{6.28}$$

Considering relations (6.26), the sums can be simplified to

$$\eta^{(1)}_{pq} = (f_{pp} f_{pq} - f_{pq} f_{qq}) + f_{pq} v_{qppq} \left(n_q - n_p \right) \tag{6.29}$$

$$\begin{aligned} \eta^{(2)}_{pqrs} =& f_{ps} v^d_{sqsq} \delta_{qr} + f_{qr} v^d_{rprp} \delta_{pr} - f_{pr} v^d_{rqrq} \delta_{qs} - f_{qs} v^d_{spsp} \delta_{pr} \\ & - \left(f_{qr} \delta_{ps} + f_{ps} \delta_{qr} - f_{pr} \delta_{qs} - f_{qs} \delta_{pr} \right) v^d_{pqpq} \\ & + \left(f^d_{pp} + f^d_{qq} - f^d_{rr} - f^d_{ss} \right) v_{pqrs} \\ & + \Big(v^d_{pqpq} (1 - n_p - n_q) - v^d_{rsrs} (1 - n_r - n_s) - v^d_{rprp} (n_r - n_p) \\ & - v^d_{rqrq} (n_r - n_q) - v^d_{spsp} (n_s - n_p) - v^d_{sqsq} (n_s - n_q) \Big) v_{pqrs}. \end{aligned} \tag{6.30}$$

With this simplification, the matrix elements $\eta^{(1)}_{pq}$ and $\eta^{(2)}_{pqrs}$ contain no sums over indices, which is of great importance regarding computational efficiency.

It should again be mentioned that in general, the generator $\hat{\eta}$ also includes terms of higher order, even if the Hamiltonian itself is truncated to two-body level. These terms $\eta^{(3)}_{pqrstu}$ then induce higher order interaction terms in the Hamiltonian, making the Hamiltonian more and more complex with each evaluation. However, in the IM-SRG(2) approach both, the generator $\hat{\eta}$ and the final flow equations, are truncated to terms with maximal two creation and two annihilation operators.

6.4.2 IM-SRG(2) with White's generator

White's generator for the in-medium SRG approach is explicitly derived for nuclear systems in Ref. [52]. These expressions can be applied to our systems of quantum dots, too, and we will therefore work out the equations in the following.

In Eq. (6.11), White's generator has been given as

$$\hat{\eta}(s) = \sum_\alpha \eta_\alpha(s) h_\alpha, \tag{6.31}$$

with

$$\eta_\alpha(s) = b_\alpha a_\alpha(s), \quad b_\alpha = (E^\alpha_l - E^\alpha_r)^{-1}.$$

Since the goal is to rotate those elements to zero that are connected to the reference state $|\Phi_0\rangle$, we want to eliminate the coefficients $a_\alpha(s)$ of the following sets of creation and annihilation operators:

$$h_\alpha \in \left\{ \{a_p^\dagger a_h\}, \{a_h^\dagger a_p\}, \{a_{p_1}^\dagger a_{p_2}^\dagger a_{h_2} a_{h_1}\}, \{a_{h_1}^\dagger a_{h_2}^\dagger a_{p_2} a_{p_1}\} \right\}. \tag{6.32}$$

Here, indices h denote hole states, whereas indices p denote particle states. Since IM-SRG(2) is restricted to one-particle-one-hole and two-particle-two-hole excitations, higher excitations are not considered, and these are all terms to be taken into account.
The terms in Eq. (6.32) can be divided into two types: When applied to $|\Phi_0\rangle$, the expressions $\{a_p^\dagger a_h\}$ and $\{a_{p_1}^\dagger a_{p_2}^\dagger a_{h_2} a_{h_1}\}$ contain only operators of $d^\dagger$-type (see section 6.2.2). Therefore they are assigned to the left state. On the other hand, the sets $\{a_h^\dagger a_p\}$ and $\{a_{h_1}^\dagger a_{h_2}^\dagger a_{p_2} a_{p_1}\}$ correspond to d-operators when applied to $|\Phi_0\rangle$ and are therefore assigned to the right state.

Corresponding to the sets of creation and annihilation operators in Eq. (6.32), White's generator has only two types of non-zero elements:
The non-zero one-body elements are limited to combinations of one particle and one hole and of the form $\eta_{ph}^{(1)}$. Due to anti-hermiticity of the generator, elements $\eta_{hp}^{(1)}$ are obtained by the relation $\eta_{hp}^{(1)} = -\eta_{ph}^{(1)}$.
For the non-zero two-body elements, corresponding to the sets $h_\alpha = \{a_{p_1}^\dagger a_{p_2}^\dagger a_{h_2} a_{h_1}\}$ and $h_\alpha = \{a_{h_1}^\dagger a_{h_2}^\dagger a_{p_2} a_{p_1}\}$, combinations of two particles and two holes are needed. Anti-hermiticity limits this again to one relevant term, since $\eta_{p_1 p_2 h_1 h_2}^{(2)} = -\eta_{h_1 h_2 p_1 p_2}^{(2)}$.

To derive the concrete expressions, we follow Ref. [52]. For the one-body elements $\eta_{ph}^{(1)}$, the left and right state are defined as

$$\begin{aligned} |R^{(1)}\rangle &= |\Phi_0\rangle \\ |L^{(1)}\rangle &= \{a_p^\dagger a_h\} |\Phi_0\rangle, \qquad \langle L^{(1)}| = \langle \Phi_0| \{a_h^\dagger a_p\}, \end{aligned}$$

respectively.

This yields for the corresponding energies

$$\begin{aligned} E_r^{(1)}(s) &= \langle R^{(1)}|\hat{H}(s)|R^{(1)}\rangle = E_0(s) \\ E_l^{(1)}(s) &= \langle L^{(1)}|\hat{H}(s)|L^{(1)}\rangle \\ &= E_0(s) + \sum_{ij} f_{ij}(s) \langle \Phi_0| \{a_k^\dagger a_a\}\{a_i^\dagger a_j\}\{a_a^\dagger a_k\} |\Phi_0\rangle \\ &\quad + \frac{1}{4} \sum_{ijkl} v_{ijkl}(s) \langle \Phi_0| \{a_m^\dagger a_a\}\{a_i^\dagger a_j^\dagger a_l a_k\}\{a_a^\dagger a_m\} |\Phi_0\rangle \\ &= E_0 + f_{aa}(s) - f_{mm}(s) - v_{amam}(s), \end{aligned}$$

Note that in his original article [51], White suggests to take simply the initial values of E_r^α and E_l^α and mentions the s-dependent values only as a further possibility. In [52], however, $E_r^\alpha(s)$ and $E_l^\alpha(s)$ are not set to constants, but evolved during the whole flow.

With $b_\alpha(s) = (E_l^\alpha(s) - E_r^\alpha(s))^{-1} = f_{aa}(s) - f_{mm}(s) - v_{amam}(s)$, we obtain for the one-body elements of the generator

$$\eta_{ph}^{(1)}(s) = \frac{1}{f_{pp}(s) - f_{hh}(s) - v_{phph}(s)} f_{ph}(s), \tag{6.33}$$

where we have renamed the indices appropriately. For the two-body elements $\eta^{(2)}_{p_1p_2h_1h_2}$, the left and right state are needed in the following form

$$\begin{aligned}
&|R^{(2)}\rangle = 0 \\
&|L^{(2)}\rangle = \{a^\dagger_{p_1} a^\dagger_{p_2} a_{h_2} a_{h_1}\} \, |\Phi_0\rangle \,, \qquad \langle L^{(2)}| = \langle \Phi_0| \, \{a^\dagger_{h_1} a^\dagger_{h_2} a_{p_2} a_{p_1}\}.
\end{aligned}$$

This yields the following left and right energies:

$$\begin{aligned}
E_r^{(2)}(s) &= \langle R^{(2)}|\hat{H}(s)|R^{(2)}\rangle = E_0(s), \\
E_l^{(2)}(s) &= \langle L^{(2)}|\hat{H}(s)|L^{(2)}\rangle \\
&= E_0(s) + \sum_{ij} f_{ij}(s) \, \langle \Phi_0| \, \{a^\dagger_k a^\dagger_l a_b a_a\}\{a^\dagger_i a_j\}\{a^\dagger_a a^\dagger_b a_l a_k\} \, |\Phi_0\rangle \\
&\quad + \frac{1}{4} \sum_{ijkl} v_{ijkl}(s \, \langle \Phi_0| \, \{a^\dagger_m a^\dagger_n a_b a_a\}\{a^\dagger_i a^\dagger_j a_l a_k\}\{a^\dagger_a a^\dagger_b a_n a_m\} \, |\Phi_0\rangle \\
&= E_0(s) + f_{aa}(s) + f_{bb}(s) - f_{mm}(s) - f_{nn}(s) \\
&\quad + v_{abab}(s) + v_{mnmn}(s) - v_{amam}(s) - v_{anan}(s) - v_{bmbm}(s) - v_{bnbn}(s) \\
&\equiv E_0(s) + f_{aa}(s) + f_{bb}(s) - f_{mm}(s) - f_{nn}(s) + A_{abmn}(s),
\end{aligned}$$

where we introduced

$$A_{abmn}(s) = v_{abab}(s) + v_{mnmn}(s) - v_{amam}(s) - v_{anan}(s) - v_{bmbm}(s) - v_{bnbn}(s). \tag{6.34}$$

Renaming the indices appropriately, the non-zero two-body elements become

$$\eta^{(2)}_{p_1p_2h_1h_2}(s) = \frac{1}{f_{p_1p_1}(s) + f_{p_2p_2}(s) - f_{h_1h_1}(s) - f_{h_2h_2}(s) + A_{p_1p_2h_1h_2}(s)} \, v_{p_1p_2h_1h_2}(s). \tag{6.35}$$

Inserting the elements of Eqs. (6.33) and (6.35) in Eq. (6.31) , White's generator, truncated to second order, can in total be written as

$$\hat{\eta} = \sum_{ai} \frac{f_{ai}}{f_a - f_i - v_{aiai}} \{a^\dagger_a a_i\} - \text{hc} + \sum_{abij} \frac{v_{abij}}{f_a + f_b - f_i - f_j + A_{abij}} \{a^\dagger_a a^\dagger_b a_j a_i\} - \text{hc}, \tag{6.36}$$

with the common notation $f_p \equiv f_{pp}$ and 'hc' denoting the Hermitian conjugate. When summing over indices, we use as before indices $\{a, b\}$ for particle states and indices $\{i, j\}$ for hole states.

Since the energy denominators are constructed using the diagonal matrix elements of $\hat{H}$, the generator is naturally regularized if the difference between single-particle energies might become small during the flow [27].
Compared to Wegner's canonical generator (6.25), where the final flow equations involve third order powers of the f- and v-elements, these elements contribute only linearly with White's generator, which reduces the stiffness of the flow equations significantly.

Chapter 7

Other many-body methods

Apart from the SRG method, there exist several other popular many-methods that can be used to solve the problem of interacting electrons. Two of these methods have been implemented by us in the course of this thesis, and will therefore present them in more detail:
The first method is the Hartree-Fock (HF) method, which converts the problem of interacting fermions to an effective single-particle problem. The second method is the Diffusion Monte Carlo (DMC) method, a quantum Monte Carlo method that solves Schrödinger's equation by employing a Green's function.

7.1 Hartree-Fock

The Hartree-Fock method is an *ab initio* method which was first introduced as self-consistent field method by Hartree, and later corrected and extended by Fock [55]. Its main assumption is that each particle of the system moves in a mean field potential which is set up by all the other particles in the system. That way, the complicated two-body potential is replaced by an effective single-particle potential, which is much easier to handle. This simple approximation is often the first starting point in many-body calculations and used as input for more complex methods, such as Coupled Cluster (see e.g. [13]) and variational Monte Carlo methods. Since in this thesis, we concentrate on closed-shell systems where all orbitals are doubly occupied, we will only present the Restricted Hartree-Fock method. Open-shell systems, where some of the electrons are not paired, can be treated with the Unrestricted Hartree-Fock method, see [55].

As an ansatz, we assume that the wave function can be modelled as single Slater determinant. Based on the variational principle, stating that with an arbitrary wave function, the expectation value of the Hamiltonian can never be smaller than the real ground state energy E_0,

$$E[\Phi] = \frac{\langle \Phi | \hat{H} | \Phi \rangle}{\langle \Phi | \Phi \rangle} \geq E_0, \tag{7.1}$$

the ansatz wave function is assigned a set of parameters. These parameters are used to minimize the energy $E[\Phi]$. In this thesis, we use the approach to expand the single-particle

states $|p\rangle$, which we refer to as *Hartree-Fock orbitals*, in terms of a known basis,

$$|p\rangle = \sum_{\alpha} C_{p\alpha}|\alpha\rangle. \tag{7.2}$$

The elements of the unitary matrix C are used as variational parameters. For a two-body Hamiltonian, as given in Eq. (3.36), the ground state energy is given by

$$E\left[\Phi_0^{HF}\right] = \langle\Phi_0^{HF}|\hat{H}|\Phi_0^{HF}\rangle = \sum_i \langle i|\hat{h}^{(0)}|i\rangle + \frac{1}{2}\sum_{ij}\langle ij||ij\rangle. \tag{7.3}$$

The wave function Φ_0^{HF} is a Slater determinant of HF orbitals, and inserting relation (7.2), we obtain

$$E\left[\Phi_0^{HF}\right] = \sum_i \sum_{\alpha\beta} C_{i\alpha}^* C_{i\beta}\langle\alpha|\hat{h}^{(0)}|\beta\rangle + \frac{1}{2}\sum_{ij}\sum_{\alpha\beta\gamma\delta} C_{i\alpha}^* C_{j\beta}^* C_{i\gamma} C_{j\delta}\langle\alpha\beta||\gamma\delta\rangle. \tag{7.4}$$

As in the previous chapters, the indices $\{i, j\}$ are assumed to sum over all hole states below the Fermi level. Note that the sums over greek indices run over the complete set of basis functions, which in principle is infinitely large.
To minimize the energy functional (7.3), we employ the technique of Lagrange multipliers, using the constraint

$$\delta_{pq} = \langle p|q\rangle = \sum_{\alpha\beta} C_{p\alpha}^* C_{q\beta} = \sum_{\alpha} C_{p\alpha}^* C_{q\alpha}. \tag{7.5}$$

The function to be minimized reads

$$E\left[\Phi_0^{HF}\right] - \sum_i \omega_i \sum_{\kappa} C_{i\kappa}^* C_{i\kappa},$$

and minimizing with respect to $C_{k\alpha}^*$, we obtain

$$\begin{aligned}
0 &= \frac{\partial}{\partial C_{k\alpha}^*}\left[E\left[\Phi_0^{HF}\right] - \sum_i \omega_i \sum_{\kappa} C_{i\kappa}^* C_{i\kappa}\right] \\
&= \frac{\partial}{\partial C_{k\alpha}^*}\left[\sum_i \sum_{\kappa\beta} C_{i\kappa}^* C_{i\beta}\langle\kappa|\hat{h}^{(0)}|\beta\rangle + \frac{1}{2}\sum_{ij}\sum_{\kappa\beta\gamma\delta} C_{i\kappa}^* C_{j\beta}^* C_{i\gamma} C_{j\delta}\langle\kappa\beta||\gamma\delta\rangle - \sum_i \omega_i \sum_{\kappa} C_{i\kappa}^* C_{i\kappa}\right] \\
&= \sum_{\beta} C_{k\beta}\langle\alpha|\hat{h}^{(0)}|\beta\rangle + \sum_j \sum_{\beta\gamma\delta} C_{j\beta}^* C_{k\gamma} C_{j\delta}\langle\alpha\beta||\gamma\delta\rangle - \omega_k C_{k\alpha}.
\end{aligned}$$

Rewriting this identity as

$$\sum_{\gamma} C_{k\gamma}\left[\langle\alpha|\hat{h}^{(0)}|\gamma\rangle + \sum_j \sum_{\beta\delta} C_{j\beta}^* C_{j\delta}\langle\alpha\beta||\gamma\delta\rangle\right] = \omega_k C_{k\alpha},$$

we define the Hartree-Fock Hamiltonian as

$$\hat{h}_{\alpha\gamma}^{HF} = \langle\alpha|\hat{h}_0|\gamma\rangle + \sum_j \sum_{\beta\delta} C_{j\beta}^* C_{j\delta}\langle\alpha\beta||\gamma\delta\rangle. \tag{7.6}$$

So we obtain the simplified Hartree-Fock equations

$$\sum_{\gamma} \hat{h}^{HF}_{\alpha\gamma} C_{k\gamma} = \omega_k C_{k\alpha}. \tag{7.7}$$

Solving these equations is an eigenvalue problem and corresponds to the diagonalization of the Hartree-Fock matrix

$$\hat{h}^{HF} = \begin{pmatrix} h^{HF}_{00} & h^{HF}_{01} & \cdots \\ h^{HF}_{10} & h^{HF}_{11} & \cdots \\ \cdots & \cdots & \cdots \end{pmatrix}.$$

Note that $\hat{h}^{HF}$ only links one-particle-one-hole excitations, which matches the initial aim to replace the complicated two-body by an effective one-body potential.

7.2 Diffusion Monte Carlo (DMC)

Diffusion Monte Carlo (DMC) is a quantum Monte Carlo method which, via a transformation to imaginary time, makes the solution of Schrödinger's equation similar to a classical diffusion problem. The advantage compared to other Monte Carlo methods, e.g. Variational Monte Carlo (VMC), is that the solution does not directly depend on a trial wave function, thereby restricting the quality of the result. Instead, within the limits of the algorithm, DMC can in principle reproduce the exact ground state of the system.

7.2.1 Fundamentals of DMC

The basic philosophy can be summarized as follows: Transforming Schrödinger's equation to imaginary time, $it \to t$, it reads (using atomic units)

$$\frac{\partial \Psi(\mathbf{R}, t)}{\partial t} = \hat{H} \Psi(\mathbf{R}, t),$$

where $\mathbf{R}$ contains all degrees of freedom of the system. Expanding the state $|\Psi(\mathbf{R}, t)\rangle$ in terms of the eigenstates $|\phi_n\rangle$ of the Hamiltonian, the solution is given by

$$\begin{aligned} |\Psi(\mathbf{R}, t)\rangle &= e^{-\hat{H}t} |\Psi(\mathbf{R}, 0)\rangle \\ &= \sum_n e^{-\hat{H}t} |\phi_n\rangle\langle\phi_n|\Psi(\mathbf{R}, 0)\rangle \\ &= \sum_n e^{-E_n t} |\phi_n\rangle\langle\phi_n|\Psi(\mathbf{R}, 0)\rangle. \end{aligned}$$

For $t \to \infty$, all eigenstates with negative energy blow up while the ones with positive energy vanish.
If one is only interested in projecting the ground state component of Ψ out, a constant energy shift E_T, called *trial energy*, is introduced to the potential part of the Hamiltonian. Since the physical properties of a system are generally independent of the zero point of the energy, the physics of the system remains unchanged. The state $|\Psi(\mathbf{R}, t)\rangle$ can thus be expressed by

$$|\Psi(\mathbf{R}, t)\rangle = \sum_n e^{-(E_n - E_t)t} |\phi_n\rangle\langle\phi_n|\Psi(\mathbf{R}, 0)\rangle. \tag{7.8}$$

Considering the ideal situation where $E_T = E_0$, the contributions from the excited states vanish in the limit $t \to \infty$, projecting out the ground state Φ_0,

$$\lim_{t\to\infty} e^{-(\hat{H}-E_0)t}\Psi(\mathbf{R},t) \propto \Phi_0.$$

To have a practical scheme to do the time propagation, we expand Eq. (7.8) in the eigenstates $|\mathbf{R}_i\rangle$ of the position operator, which are referred to as *walkers*. In this basis, the time evolution is given by

$$|\Psi(\mathbf{R},t)\rangle = \sum_i e^{-(\hat{H}-E_T)t}|\mathbf{R}_i'\rangle\langle\mathbf{R}_i'|\Psi(\mathbf{R}',0)\rangle. \tag{7.9}$$

In terms of Green's functions, Eq. (7.9) can be written as

$$\Psi(\mathbf{R},t) = \int d\mathbf{R}'\, G(\mathbf{R}',\mathbf{R};t)\Psi(\mathbf{R}',0).$$

The Green's function represents the probability that the system moves from $\mathbf{R}$ to $\mathbf{R}'$ in an imaginary time interval t and is given by

$$G(\mathbf{R}',\mathbf{R};t) = \langle\mathbf{R}'|e^{-(\hat{H}-E_T)t}|\mathbf{R}\rangle \tag{7.10}$$

$$= \langle\mathbf{R}'|e^{-\left(\hat{T}+\hat{V}-E_T\right)t}|\mathbf{R}\rangle, \tag{7.11}$$

where $\hat{T}$ and $\hat{V}$ are the kinetic and potential energy operator, respectively. Writing out the imaginary time Schrödinger equation in atomic units, we get

$$\begin{aligned}\frac{\partial}{\partial t}\Psi(\mathbf{R},t) &= -\hat{T}\Psi(\mathbf{R},t) - \left(\hat{V}(\mathbf{R}) - E_T\right)\Psi(\mathbf{R},t)\\ &= \frac{1}{2}\nabla^2\Psi(\mathbf{R},t) - \left(\hat{V}(\mathbf{R}) - E_T\right)\Psi(\mathbf{R},t).\end{aligned} \tag{7.12}$$

Note that this equation is of the form of an extended diffusion equation.
The basic idea now is to represent the initial state by an ensemble of random walkers and propagate them iteratively in imaginary time. The propagation occurs according to probabilities defined by the Green's function G, which is subject to the controlled diffusion process of Eq. (7.12). After a large number of generations, the population density will represent the ground state wave function.
Interpreting the terms of Eq. (7.12) separately, we see that the first term is a standard *diffusion term* with diffusion constant $D = \frac{1}{2}$. The second term is a *rate term*, also called *branching term*, and describes a potential-dependent increase or decrease in the density of walkers.
In order to perform the diffusion and branching separately, the Baker-Campbell-Haussdorff formula [25] is applied in the limit $\tau \to 0$, $t = n\tau$. This yields

$$e^{-(\hat{H}-E_T)\tau} = e^{-\hat{T}\tau}e^{-(\hat{V}-E_T)\tau}, \tag{7.13}$$

with an error to second order in τ. Since it is only valid for small time steps τ, Eq. (7.13) is called *short time approximation* of the Green's function. Ignoring normalization factors, the Green's function reads with this factorization [56]

$$G(\mathbf{R}',\mathbf{R};t) = e^{-(\mathbf{R}'-\mathbf{R})^2/4D\tau}e^{-\left(\frac{1}{2}V(\mathbf{R})+\frac{1}{2}V(\mathbf{R}')-E_T\right)\tau}.$$

Due to divergencies in the potential, the algorithm needs to be modified. In practice, the sampled distribution is multiplied by a trial wave function $\Psi_T(\mathbf{R})$ obtained from VMC calculations, giving the distribution

$$\begin{aligned} f(\mathbf{R},t) &= \Psi_T(\mathbf{R})\Psi(\mathbf{R},t) \\ &= \int d\mathbf{R}'\, G(\mathbf{R}',\mathbf{R};t)\frac{\Psi_T(\mathbf{R})}{\Psi_T(\mathbf{R}')}\Psi_T(\mathbf{R}')\Psi(\mathbf{R}',0). \end{aligned} \tag{7.14}$$

That way, a drift velocity is added to the diffusion equation and applying the Fokker-Planck formalism[1], the modified diffusion term reads

$$G_{\text{diff}}(\mathbf{R}',\mathbf{R};\tau) = \frac{1}{(4\pi D\tau)^{3N/2}} e^{-(\mathbf{R}'-\mathbf{R}-D\tau F(\mathbf{R}))^2/4D\tau},$$

and the branching term is changed to

$$G_b(\mathbf{R}',\mathbf{R};\tau) = e^{-\left(\frac{E_L(\mathbf{R}')+E_L(\mathbf{R})}{2}-E_T\right)\tau}. \tag{7.15}$$

Thus the potential is replaced by an expression depending on the local energy, which gives a greatly reduced branching effect. In particular, as $\Psi_T \to \Phi_0$ and $E_T \to E_0$, the local energy remains constant and no branching occurs, corresponding to a stable distribution of walkers.
A final point, which will be taken up later again, is the fact that the above described procedure is only well defined in the case of a totally symmetric ground state. For fermionic systems, there are additional divergencies at the nodes of the wave function. One way to overcome this, is to additionally enforce the boundary condition that the wave function vanishes at the nodes of Ψ_T, an approach that is called *fixed-node approximation*.

7.2.2 Modelling of the trial wave function

The trial wave function encountered in Eq. (7.14) should approximate the real ground state Φ_0 as well as possible. One the one hand, the fixed-node approximation requires both wave functions to have the same nodes, one the other hand, the trial energy E_T should be close enough to E_0 to be smaller than the first excited energy. Only that way, just those walkers corresponding to the ground state are selected. In other words, one should bring in as much knowledge about the physics of the system as possible.
However, at the same time, numerical computations should not become too extensive, which means that the wave function should still keep a rather simple form. In practice, the trial wave function Ψ_T is usually obtained by running a VMC calculation and taking the obtained, with respect to parameters optimized, wave function as input for subsequent DMC runs.

Our ansatz for the wave function

In our considered quantum systems, the electrons are confined in a two-dimensional harmonic oscillator potential

$$V(\mathbf{R}) = \frac{1}{2}m\omega^2 r^2,$$

[1] For details, see [56].

where ω is the oscillator frequency. As derived in chapter 5, the single-particle functions are thus given by a product of Hermite polynomials, namely

$$\phi_{n_x,n_y}(\mathbf{R}) = AH_{n_x}\left(\sqrt{\omega}x\right) H_{n_y}\left(\sqrt{\omega}y\right) e^{-\frac{\omega}{2}\left(x^2+y^2\right)}, \tag{7.16}$$

where A is a normalization constant. Following [57], the complete trial wave function of our fermionic system consists of two parts: The first is a totally antisymmetric part, the Slater determinant. This one is the exact solution of the non-interacting system and ensures the indistinguishability of the electrons by considering all possible configurations. As explained in chapter 3, it generally reads

$$\psi_S(\mathbf{R}_1,...,\mathbf{R}_N) = \frac{1}{\sqrt{N!}} \begin{vmatrix} \phi_1(\mathbf{R}_1) & \phi_1(\mathbf{R}_2) & \cdots & \phi_1(\mathbf{R}_N) \\ \phi_2(\mathbf{R}_1) & \phi_2(\mathbf{R}_2) & \cdots & \phi_2(\mathbf{R}_N) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_N(\mathbf{R}_1) & \phi_N(\mathbf{R}_2) & \cdots & \phi_N(\mathbf{R}_N) \end{vmatrix}, \tag{7.17}$$

where the functions ϕ_i are the single-particle orbitals (7.16) and the vector $\mathbf{R}$ is assumed to contain both spatial and spin degrees of freedom.
A computationally smarter solution is to create one Slater determinant for the spin up electrons, and one for the spin down ones. It can been shown [58] that if one uses the product of these two half-sized determinants instead of the full Slater determinant, one gets exactly the same energy, provided that the Hamiltonian is spin independent:

$$\psi_S(\mathbf{R}_1,...,\mathbf{R}_N) = \frac{1}{\sqrt{N!}} |D_\uparrow| |D_\downarrow|$$

with

$$|D_\uparrow| = \begin{vmatrix} \phi_{1\uparrow}(\mathbf{R}_1) & \phi_{1\uparrow}(\mathbf{R}_2) & \cdots & \phi_{1\uparrow}\left(\mathbf{R}_{\frac{N}{2}}\right) \\ \phi_{2\uparrow}(\mathbf{R}_1) & \phi_{2\uparrow}(\mathbf{R}_2) & \cdots & \phi_{2\uparrow}\left(\mathbf{R}_{\frac{N}{2}}\right) \\ \vdots & & \ddots & \vdots \\ \phi_{\frac{N}{2}\uparrow}(\mathbf{R}_1) & \cdots & & \phi_{\frac{N}{2}\uparrow}\left(\mathbf{R}_{\frac{N}{2}}\right) \end{vmatrix}$$

and

$$|D_\downarrow| = \begin{vmatrix} \phi_{1\downarrow}\left(\mathbf{R}_{\frac{N}{2}+1}\right) & \phi_{1\downarrow}\left(\mathbf{R}_{\frac{N}{2}+2}\right) & \cdots & \phi_{1\downarrow}(\mathbf{R}_N) \\ \phi_{2\downarrow}\left(\mathbf{R}_{\frac{N}{2}+1}\right) & \phi_{1\downarrow}\left(\mathbf{R}_{\frac{N}{2}+2}\right) & \cdots & \phi_{2\downarrow}(\mathbf{R}_N) \\ \vdots & & \ddots & \vdots \\ \phi_{\frac{N}{2}\downarrow}\left(\mathbf{R}_{\frac{N}{2}+1}\right) & \cdots & & \phi_{\frac{N}{2}\downarrow}(\mathbf{R}_N) \end{vmatrix}.$$

Although the wave function now no longer is antisymmetric, the eigenvalues for a spin independent Hamiltonian are unchanged.
The computational strength is that when only one particle is moved at a time, only one of the determinants is changed and the other one keeps its value. Since the calculation of the Slater determinants costs quite a lot of CPU time, this makes the program much more efficient.

In principle, our wave function could be expressed as infinite linear combination of such Slater determinants. However, since this is practically not possible, we have to cut the single-particle basis at some point and include an extra correlation function instead.

In this thesis, we use only the ground-state Slater determinant for the non-interacting part. This is a reasonable ansatz, since only closed-shell systems are considered and there is a comparatively high energy difference between the highest energy level in one shell and the lowest one in the next shell. Hence we can assume that it is very hard to excite a particle from the outer-most filled shell and that it therefore is rather unlikely to have Slater determinants with orbitals of higher energy.
Since our Slater determinant does not include any correlation effects, it is crucial to have a correlation term. This one must fulfil an important cusp condition, namely it has to take care of the singularity which arises when having zero distance between two particles.
In this thesis, we use the *Pade-Jastrow* function,

$$J = \prod_{i<j}^{N} \exp\left(\frac{a r_{ij}}{1+\beta r_{ij}}\right),$$

where $a = \frac{1}{3}$ for particles with equal spin and $a = 1$, else. In this function, a simple parameter β describes the whole strength of the correlation: If β is large, the Jastrow factor is close to one, meaning that the effect of the interaction is small. On the other hand, the smaller β becomes, the more central the role of correlations in the system is.

For the Slater determinant, we introduce the variational parameter α and include it in the single-particle wave functions the following way:

$$\phi_{n_x,n_y}(x,y;\alpha) = H_{n_x}\left(\sqrt{\omega\alpha}x\right) H_{n_y}\left(\sqrt{\omega\alpha}y\right) e^{-\frac{\omega\alpha}{2}\left(x^2+y^2\right)}. \tag{7.18}$$

Note that we omit all normalization constants since only ratios between wave functions will be considered. The parameter α serves as a scaling factor of the oscillator frequency. The closer it is to one, the closer the system is to a perfect harmonic oscillator.

Bringing it all together, the total trial wave function reads

$$\Psi_T(\alpha,\beta) = |D_\uparrow(\alpha)|\,|D_\downarrow(\alpha)|\,J(\beta). \tag{7.19}$$

Extracting the parameters using Variational Monte Carlo (VMC)

Our ansatz for the trial wave function Ψ_T includes two variational parameters, α and β, which shall be optimized to approximate the real ground state Φ_0 as well by Ψ_T as possible. Practically, we base this search for parameters on the variational principle, stating that the energy calculated from any trial wave function can never be below the true ground state energy. In particular, we compute the integral

$$\langle\hat{H}\rangle = \frac{\langle\Psi_T|\hat{H}|\Psi_T\rangle}{\langle\Psi_T|\Psi_T\rangle} = \frac{\int d\mathbf{R}\Psi_T^*(\mathbf{R})\hat{H}(\mathbf{R})\Psi_T(\mathbf{R})}{\int d\mathbf{R}\Psi_T^*(\mathbf{R})\Psi_T(\mathbf{R})} \tag{7.20}$$

and minimize it with respect to the variational parameters. To compute the integral and determine those variational parameters that yield a minimum, we use the standard VMC approach. This means that we combine Monte-Carlo integration with the Metropolis algorithm, where the latter one governs the transition of states.

Monte Carlo integration Monte Carlo integration is a very useful method of integration, especially for integrals of higher dimensions. Its basic philosophy is rather simple:
Consider a function $f(x)$ which shall be integrated over some interval $[a,b]$:

$$I = \int_a^b f(x)dx. \tag{7.21}$$

From statistics we know that the expectation value of the function $f(x)$ on $[a,b]$ is calculated by multiplying it with a probability distribution function (PDF) and integrating over the desired interval:

$$\langle f(x)\rangle = \int_a^b P(x)f(x)dx. \tag{7.22}$$

The main idea now is to bring integral (7.21) into the form of Eq. (7.22). This is done by rewriting

$$I = \int_a^b P(x)\left(\frac{f(x)}{P(x)}\right)dx = \langle\frac{f(x)}{P(x)}\rangle.$$

Hence we replace the integral (7.21) by the average of $f(x)$ divided by some PDF. The idea of Monte Carlo integration now is to choose random numbers in the interval $[a,b]$, and approximate the expectation value by averaging over all contributions:

$$I = \langle\frac{f(x)}{P(x)}\rangle \simeq \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{P(x_i)}. \tag{7.23}$$

The points x_i are randomly generated from the probability distribution $P(x)$.
Monte Carlo integration is a statistical method and yields obviously only an approximation to the real expectation value. Now matter how large the samples are chosen, there will always be a statistical error and one needs a measure of how statistically precise the obtained estimate is. This is provided by the so-called *variance*, which is given by

$$\sigma^2\left(\langle f\rangle\right) = \frac{\sigma^2(f)}{N}.$$

The quantity $\sigma^2(f)$ is the sample variance

$$\sigma^2(f) = \langle f^2\rangle - \langle f\rangle^2.$$

The smaller the variance is, the closer the obtained expectation value is to the true average.

Theory behind Variational Monte Carlo In principle, one could use any arbitrary PDF for Eq. (7.23) and compute the expectation value. A smarter choice, however, is to take a distribution which behaves similar to the original function and results in a smoother curve to be integrated. In particular, we want to have a large number of integration points in regions where the function varies rapidly and has large values, whereas fewer points are needed in regions where the function is almost constant or vanishes.

For our wave functions, a smart choice of the probability distribution is

$$P(\mathbf{R}) = \frac{|\Psi_T|^2}{\int d\mathbf{R}|\Psi_T|^2}.$$

This is very intuitive, since the quantum mechanical interpretation of $|\Psi_T|^2$ is nothing else than a probability distribution. The expectation value of the Hamiltonian can then be rewritten as

$$\begin{aligned}\langle \hat{H} \rangle_T &= \frac{\langle \Psi_T | \hat{H} | \Psi_T \rangle}{\langle \Psi_T | \Psi_T \rangle} = \frac{\int d\mathbf{R} \Psi_T^*(\mathbf{R}) \hat{H}(\mathbf{R}) \Psi_T(\mathbf{R})}{\int d\mathbf{R} |\Psi_T|^2} \\ &= \frac{1}{\int d\mathbf{R} |\Psi_T|^2} \int d\mathbf{R} \Psi_T^* \Psi_T \left(\frac{1}{\Psi_T} \hat{H} \Psi_T \right) \\ &= \int d\mathbf{R} P(\mathbf{R}) E_L(\mathbf{R}) = \langle E_L \rangle_T,\end{aligned}$$

where

$$E_L = \frac{1}{\Psi_T} \hat{H} \Psi_T$$

is the so-called *local energy*. In other words, we have replaced the integral (7.20) by the expectation value of the local energy. The latter one is specific for a given trial wave function and therefore dependent on the set of variational parameters.

In terms of the local energy, the sample variance can be expressed by

$$\begin{aligned}\sigma^2(\hat{H}) &= \langle \hat{H}^2 \rangle - \langle \hat{H} \rangle^2 \\ &= \frac{1}{N} \sum_{i=1}^{N} E_L^2(\mathbf{R}_i) - \left(\frac{1}{N} \sum_{i=1}^{N} E_L(\mathbf{R}_i) \right)^2\end{aligned}$$

and is an indicator of how close the trial wave function is to the true eigenstate of $\hat{H}$.

The Metropolis algorithm To compute the integral

$$E[\Psi_T] = \int d\mathbf{R}\, E_L(\mathbf{R}) P(\mathbf{R}), \tag{7.24}$$

we employ the *Metropolis algorithm*, which is that part of the VMC machinery which governs the transition of states. Obviously, the sample points for Eq. (7.24) must be chosen with care: In order to get the true expectation value, we have to follow the PDF in a correct way.

For that reason, we employ a *Markov chain*, which is a random walk with a selected probability for making a move: All random walkers start out from an initial position and, as time elapses, spread out in space, meaning that they occupy more and more states. After a certain amount of time steps, the system reaches an equilibrium situation, where the most likely state has been reached.
Two conditions must be fulfilled in order to sample correctly: *Ergodicity* and *detailed balance*. The ergodic hypothesis states that if the system is simulated long enough, one should be able to trace through all possible paths in the space of available states to reach the equilibrium situation. In other words, no matter from which state one starts, one should be able to reach any other state provided the run is long enough. Markov processes fulfil this requirement because all moves are independent of the previous history, suggesting that for each time step the random walkers start "with a clean memory" to explore the space of all available states.

To explain the concept of detailed balance, we have to get a bit more technical. From transport theory we have the famous *master equation* [59], which relates the transition probabilities of all states,

$$\frac{dw_i(t)}{dt} = \sum_j \left[W(j \to i)w_j - W(i \to j)w_i\right],$$

where w is the PDF and $W(i \to j)$ the transition matrix from state i to state j. In equilibrium, the probability distribution should not change, therefore we demand $\frac{dw}{dt} = 0$ and are left with

$$\sum_j \left[W(j \to i)\, w_j - W(i \to j)\, w_i\right] = 0. \tag{7.25}$$

Detailed balance now ensures the generation of the correct distribution by demanding that the system follows the trivial solution of Eq. (7.25), namely

$$W(j \to i)w_j = W(i \to j)w_i.$$

This in turn means that in equilibrium, we have the following condition

$$\frac{W(j \to i)}{W(i \to j)} = \frac{w_i}{w_j}, \tag{7.26}$$

where the left-hand side is in general unknown. As an ansatz, the transition probability $W(i \to j)$ from state i to state j is modelled as product of the selection probability g to choose a certain state, times the probability A of actually performing this move,

$$W(i \to j) = g(i \to j)A(i \to j).$$

Plugging this into Eq. (7.26) gives

$$\frac{g(j \to i)A(j \to i)}{g(i \to j)A(i \to j)} = \frac{w_i}{w_j}. \tag{7.27}$$

In the standard *Metropolis* algorithm, one assumes that the walker's probability of picking the transition from i to j should not differ from picking the opposite direction j to i. This simplifies Eq. (7.27) to

$$\frac{A(j \to i)}{A(i \to j)} = \frac{w_i}{w_j}.$$

In our specific case, w is the square of the wave function, leading to

$$A(j \to i) = \left|\frac{\psi_i}{\psi_j}\right|^2 A(i \to j). \tag{7.28}$$

The probability for accepting a move to a state with higher probability is set to 1. For that reason, if in Eq. (7.28) state i has a higher probability than state j, then $A(j \to i) = 1$, and we get

$$A(j \to i) = \begin{cases} \left|\frac{\psi_i}{\psi_j}\right|^2 & \text{if } |\psi_i|^2 < |\psi_j|^2 \\ 1 & \text{else.} \end{cases} \tag{7.29}$$

Since it is the probability ratio between new and old state that decides whether a move is accepted or not, we ensure that the walkers follow the path of the PDF. Although moves to more probable states are more likely to be accepted, also transitions to less probable states are possible, which is necessary to ensure ergodicity.

Generalized Metropolis sampling The simple approach presented thus far is not very optimal, since the positions are chosen completely randomly and not adjusted to the shape of the wave function. Many sample points are in small regions of the wave function and get rejected. It is therefore instructive to go back to Eq. (7.27) and choose a more advanced model for g, which pushes the walkers into the direction of higher probabilities.
As starting point we use the *Fokker-Planck equation*, which describes an isotropic diffusion process where the particles are effected by an external potential:

$$\frac{\partial \rho}{\partial t} = D\nabla(\nabla - F)\rho. \tag{7.30}$$

The parameter D denotes the diffusion constant, which, considering Schrödinger's equation in atomic units, in our case is $D = \frac{1}{2}$. The variable F denotes the drift term, the so-called *quantum force*.
Let i denote the component of the probability distribution related to particle i. Then we can rewrite

$$\frac{\partial \rho(\mathbf{R}, t)}{\partial t} = \sum_{i=1}^{N} D\nabla_i \left[\nabla_i - \mathbf{F}_i(\mathbf{R})\right] \rho(\mathbf{R}, t). \tag{7.31}$$

The quantum force is determined by solving the Fokker-Planck equation (7.30) for stationary densities and in its simplest form, where not just the whole expression but each single term in the sum equals zero:

$$0 = D\nabla_i \left[\nabla_i - \mathbf{F}_i(\mathbf{R})\right] \rho(\mathbf{R}, t).$$

For our $\rho(\mathbf{R}, t) = |\psi_T|^2$, the solution is given by

$$\mathbf{F}_i(\mathbf{R}) = 2\frac{1}{\Psi_T}\nabla_i \Psi_T.$$

This expression tells very well what the quantum force actually is doing: The gradient of the wave function determines in which direction the walker should move to reach a region of higher interest. If the walker is far away from such a region, i.e. the wave function is currently very small, then the quantum force gets extra large and pushes the walker more intense into the right direction than in regions that already have a high probability (with large values of Ψ_T).

The Fokker-Planck equation can now be used as input for the Monte Carlo algorithm, with the aim of modelling Eq. (7.27) more properly than the standard Metropolis algorithm does. In statistical mechanics, Fokker-Planck trajectories are generated via the *Langevin equation*, which for the concrete expression of Eq. (7.31) reads

$$\frac{\partial \mathbf{R}(t)}{\partial t} = D\mathbf{F}_i\left(\mathbf{R}(t)\right) + \eta. \tag{7.32}$$

In this equation, the components of η are random variables following a Gaussian distribution with mean zero and a variance of $2D$.

To make this equation numerically practicable, we have to discretize it in time $t \to t_n = n\Delta t$. Integrating over the short time interval Δt, we obtain an expression for generating the new trial positions,

$$\mathbf{R}' = \mathbf{R} + D\Delta t\mathbf{F}\left(\mathbf{R}\right) + \chi, \tag{7.33}$$

where χ is a random Gaussian variable with mean zero and variance $2D\Delta t$.
The solution to the Fokker-Planck equation (in two dimensions with N particles) is given in form of a Green's function [56],

$$G\left(\mathbf{R},\mathbf{R}';\Delta t\right)=\frac{1}{(4\pi D\Delta t)^N}e^{-\frac{1}{4D\Delta t}(\mathbf{R}'-\mathbf{R}-D\Delta t\mathbf{F}(\mathbf{R}))^2}.$$

For the improvement of the standard Metropolis algorithm, the so-called *Metropolis-Hastings* algorithm, we use this solution for the selection probability, instead of setting $g(i\to j)=g(j\to i)$. Hence Eq. (7.27) takes the form

$$\frac{w_i}{w_j}=\frac{G\left(\mathbf{R},\mathbf{R}';\Delta t\right)}{G\left(\mathbf{R}',\mathbf{R};\Delta t\right)}\frac{A(j\to i)}{A(i\to j)}.$$

The probability to perform a move is now given by

$$A(j\to i)=\begin{cases}R=\frac{G(\mathbf{R}',\mathbf{R};\Delta t)}{G(\mathbf{R},\mathbf{R}';\Delta t)}\left|\frac{\psi_i}{\psi_j}\right|^2 & \text{if } R<1\\ 1 & \text{else.}\end{cases}\tag{7.34}$$

The fraction involving the Greens functions can be simplified to

$$\frac{G\left(\mathbf{R}',\mathbf{R};\Delta t\right)}{G\left(\mathbf{R},\mathbf{R}';\Delta t\right)}=e^{-\frac{1}{2}(\mathbf{F}(\mathbf{R})+\mathbf{F}(\mathbf{R}'))\left(\frac{\Delta t}{4}(\mathbf{F}(\mathbf{R})-\mathbf{F}(\mathbf{R}'))\right)+(\mathbf{R}-\mathbf{R}')}.\tag{7.35}$$

Conjugate Gradient method & DFP The main aim of the VMC algorithm is to find optimal parameters for the trial wave function that minimize the expectation value of the energy. We proceed in such a way that the we first close in the region of contemplable parameters α and β by setting up a coarse grid. Afterwards, we employ a linear algebra method called *Davidon-Fletcher-Powell* algorithm (DFP) [60]. This method is based on the *Conjugate Gradient method* (CGM).

The CGM is an algorithm for iteratively solving particular systems of linear equations

$$\mathbf{Ax}=\mathbf{b},\tag{7.36}$$

namely those where the matrix $\mathbf{A}$ is square, symmetric and positive-definite. To solve Eq. (7.36) iteratively, the residual $\mathbf{r}=\mathbf{b}-\mathbf{Ax}$ is minimized. It gets zero when the minimum of the quadratic equation

$$P(\mathbf{x})=\frac{1}{2}\mathbf{x^T Ax}-\mathbf{x^T b}+c$$

is reached. To find this minimum, a sequence of vectors $\{\mathbf{x}_k\}$ is generated, specified by

$$\mathbf{x}_{k+1}=\mathbf{x}_k+\alpha_k\mathbf{d}_k,\tag{7.37}$$

where α_k is a scalar determining the step length and $\mathbf{d}_k$ a search direction. The search directions $\mathbf{d}_k$ need to fulfil the requirement to be conjugate to the subsequent error vector $\mathbf{e}_k=\mathbf{x}-\mathbf{x}_{k+1}$, where $\mathbf{x}$ is the exact solution. Note that two vectors $\mathbf{u},\mathbf{v}$ are said to be conjugate if they fulfil

$$\mathbf{u}^T\mathbf{Av}=0.$$

Since the solution $\mathbf{x}$ is not known, this requirement cannot be applied directly. However, it can be shown to be equivalent (see [61]) to finding the minimum point along the search direction $\mathbf{d}_k$ with the step length given by

$$\alpha_k = -\frac{\mathbf{d}_k^T \mathbf{A} \mathbf{e}_k}{\mathbf{d}_k^T \mathbf{A} \mathbf{d}_k} = \frac{\mathbf{d}_k^T \mathbf{r}_k}{\mathbf{d}_k^T \mathbf{A} \mathbf{d}_k}.$$

With the additional condition to span the same Krylov subspace as the corresponding residuals, the search directions $\mathbf{d}_k$ are obtained recursively by

$$\mathbf{d}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{d}_k, \qquad \beta_k = \frac{\mathbf{r}_{k+1}^T \mathbf{r}_{k+1}}{\mathbf{r}_k^T \mathbf{r}_k}.$$

The DFP algorithm now takes an arbitrary function $f(\mathbf{x})$ of variational parameters, that are stored in the vector $\mathbf{x}$, and approximates this function by a quadratic form. This one is based on a Taylor series of f around some point $\mathbf{x_i}$ of variational parameters:

$$f(\mathbf{x}) = f(\mathbf{x}_i) + (\mathbf{x} - \mathbf{x}_i)\nabla f(\mathbf{x}_i) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_i)\mathbf{A}(\mathbf{x} - \mathbf{x}_i),$$

where the matrix $\mathbf{A}$ is the Hessian matrix of the function f at x_i. This one is generally hard to compute and therefore approximated iteratively, where the Conjugate Gradient method is employed for generating directions.
The gradient of the function f is given by

$$\nabla f(\mathbf{x}) = \nabla f(\mathbf{x}_i) + \mathbf{A}(\mathbf{x} - \mathbf{x}_i).$$

Using Newton's method, one sets $\nabla f = 0$ to find the next iteration point, which yields

$$\mathbf{x} - \mathbf{x}_i = -\mathbf{A}^{-1} \cdot \nabla f(\mathbf{x}_i).$$

The minimization of the energy thus requires that the VMC program also computes the derivatives of the energy with respect to the variational parameters.
Once the optimal parameter set $\{\alpha, \beta\}$, corresponding to the lowest ground state energy, has been determined, the trial wave function (7.19) can be used as input for the subsequent DMC calculation.

Part III

IMPLEMENTATION AND RESULTS

Chapter 8

Implementation

In this chapter, we explain the code that we developed for this thesis. We discuss its structure, show how we implemented the methods of the two last chapters in detail and explain how the different items are related to each other to make the code as flexible and efficient as possible. To make the reader familiar with the terminology used in the following sections, we start with a short introduction to object-orientation in the programming language C++. Afterwards, we will first explain the main code of this thesis, the SRG method, before we outline the implementation of the other two used many-body methods, Hartree-Fock and Diffusion Monte Carlo.

8.1 Object-orientation in C++

The programming language C++ is built on the C programming language, and was developed with the main purpose of adding object-orientation to the latter one. Analogously to C, C++ is statically typed and compiled, which means that type-checking is performed during compile-time, as opposed to run-time. This is of great importance regarding high-level performance, where the code shall run as fast as possible.
Object-orientation provides the user with tools to write general codes that can without much effort be adapted to the demands of different problems. In physics problems, this is a great benefit, since it allows to treat different variations of the same problem, e.g. different potentials or Euclidean dimensions, without having to write a complete new code for each instance. Instead, so-called *classes* form reusable building blocks that can unify groups of problems in a structured way, always opening up the possibility for extension.
In this section, we will explain the central features of C++ which allow object-oriented programming. We will concentrate on those aspects that are really necessary for understanding the structure of the code in this thesis, and we assume that the reader is familiar with the syntax and basic functionalities of C++. For details, we refer to [62].

8.1.1 Classes and objects

To combine data structures and methods for data manipulation handily in one package, C++ provides so-called *classes*. A class is used to specify the form of an object, and all data and functions within a class are called *members* of a class.

Definition of a C++ class

Defining a class, the prototype for a data type is specified. In particular, it is determined what objects the class consist of and what kinds of operations one can perform on such objects. As an example, we give the definition of our class `SPstate`, which is a class for holding single-particle states:

```
1  class SPstate
2  {
3    private:
4      int index; // label of the state
5      ...
6
7    public:
8      SPstate();
9       ~SPstate();
10     void create(int i_qn);
11     ...
12 };
```

Each class definition starts with the keyword `class`, after which the name of the class and the class body are given. The class body contains the members of the class, and the keywords `public, private` and `protected` determine their access attributes. For example, a public member can be accessed from anywhere outside the class, whereas private members can only be accessed within the class itself. By default, members are assumed to be private.
In the example of `SPstate`, we have a private variable called `index`, which is of type integer, and in line 10, the public function `create` is declared, with explicitly specified syntax. Note that member functions of a class are defined within the class definition like any other variable.

Defining a C++ object

Instances of a class are called *objects*, and contain all the members of the class. To declare an object, one can use one of the following two alternatives:

```
SPstate SP();
SPstate* SP = new SPstate();
```

In the first case, we create an object `SP` of type `SPstate`, whereas in the second case, a pointer is created. The latter alternative we use in our code to generate arrays containing objects of type `SPstate` as items:

```
SPstate* singPart = new SPstate[number_states];
```

Each time a new object of a class is created, a special function, called *constructor*, is called. In our example of the class `SPstate`, this is the function declared in line 8. A *destructor*, see line 9 of the above example, is another special function, and called when a created object is deleted.

8.1.2 Inheritance

The main concept in object-orientation is the one of inheritance, providing the opportunity to reuse code functionality. Inheritance allows to define a class in terms of another class, such that the new class inherits the members of an existing one, which avoids writing completely new data members and functions. The existing class is called *base* class, whereas the new class is referred to as *derived class* or *subclass*.
To define a subclass, one uses a class derivation list to specify the base class (or several ones). This list names one or several base classes and has the following form:

```
class subclass: access-specifier baseClass
```

Here `baseClass` is the name of any previously defined class, and `access-specifier` must be one of `public, protected, private`. By default, `private` is used. The specifier `protected` means that data members can only be accessed within the class itself and all derived subclasses.

As an example, we consider the class `System` of our SRG code, which has the derived class `System_2DQdot`:

```
1  class System {
2
3    protected:
4      int R, numpart, sp_states;
5
6    public:
7      System(){};
8      virtual void mapping(double omega) = 0;
9      void setup(bool hfbasis, double omega, int label);
10     ...
11 };
12
13 class System_2DQdot: public System{
14
15   public:
16     System_2DQdot(int numpart, int R);
17     void mapping(double omega);
18 };
```

When an object of class `System_2DQdot` is created, it inherits all data members of `System`, in particular all the protected data members declared in line 4, and the public member functions, which we will come back to in the following.

Polymorphism When deriving more than one subclass from a base class, the C++ functionality of *polymorphism* is very handy. Polymorphism allows the base class and its subclasses

to have functions of the same name, and a call to such functions will cause different lines of code to be executed, depending on the type of object that invokes the function.

As an example, consider the function `mapping` of the previous listing. This function is declared in the class `System`, as well as in `System_2DQdot`, and if we had further base classes of `System`, each of the classes could have one such function with a specific implementation, too. To make it possible to select the specific function to be called, based on the kind of calling object, we have defined `mapping` to be *virtual* in the base class `System`. Defining a virtual function in a base class, with a specific version in derived classes, signals the compiler not to use static linkage for this function.
In our case, the function `mapping` is even set to zero, since there is no meaningful general definition for it in the base class. In this case, it is referred to as *pure virtual function*. To demonstrate usage, consider the following extract of our code:

```
void System::setup(bool hfbasis, double omega, int label) {
        ...
    mapping(omega);
    ...
}
```

Here the class `System` contains a member function `setup`, which calls the function `mapping`. As already explained, the latter one represents a pure virtual function in the class `System`. We can now call:

```
...
System_2dQdot QuantumDot(numpart, R);
QuatumDot.setup(hfbasis, omega, label);
```

The compiler understands that `QuantumDot` is derived of the base class `System`, and therefore has a function `setup`, and calls automatically that implementation of `mapping` that is specified in the subclass `System_2DQDot`. This demonstrates the power of object-orientation and its great features for writing general, well-structured codes.

8.2 Structure of the SRG code

Our complete SRG code is written in object-oriented C++ and kept as general as possible. On the one hand, this makes it easy to switch between different options of the code (e.g. use of different generators $\hat{\eta}$, harmonic oscillator or Hartree-Fock basis). On the other hand, the code is easy to extend to other systems, e.g. atoms and nuclei, and it is uncomplicated to add additional generators, potentials etc.

The specific implementations for the free-space and in-medium approach of the SRG method are quite different. The Hamiltonian, for example, is in free space stored as a complete matrix, with the matrix elements obtained via the action of creation and annihilation operators. In medium, on the other hand, it is necessary to store the elements f_{pq} and v_{pqrs}, at the same time keeping track whether the indices correspond to hole or particle states. Since storage system, access to elements, etc. are so different for the two approaches, it makes little sense to put them into a common class. Even the implementation as subclasses of a common class `Hamiltonian` seems rather artificial and forced to us. Since similar argumentations hold for

other parts of the code, e.g. the classes `Basis` and `SRG`, we decided on two separate codes. Nevertheless, we have tried to find as many common data structures and methods as possible, enabling us to have common classes that can be used by both SRG codes. To keep everything as structured and transparent as possible, we chose the same class names also for those classes that are implemented differently in both codes. The following list summarizes the classes we designed, and gives a short explanation regarding their purpose.

Classes specifying the quantum mechanical system

- Class `System`: A class for holding all data structures and methods of a specific system. It serves as interface for solver classes.
- Class `Hamiltonian`: A class for handling the Hamiltonian matrix in a given basis.
- Class `Basis`: A class containing the basis the Hamiltonian is set up in. For in-medium SRG, this class is particularly responsible for creating and administering a two-particle basis.
- Class `SPstate`: A class for holding the single-particle states.

Solver classes

- Class `SRG`: Our main many-body solver. It accepts an object of type `System` and uses the SRG method to determine the ground state energy.
- Class `HartreeFock`: Our second many-body solver. A class for performing a Hartree-Fock calculation and transforming a given basis to Hartree-Fock basis. The class can be used separately for solely determining the Hartree-Fock energy, or combined with other many-body solvers if a Hartree-Fock basis is desired.

Organization of code Our complete code lies in a folder called `SRG`. This folder has the following items:

- Folder `src`: This folder contains one subfolder for each of the above mentioned classes. Each subfolder has the name of the class and contains one header (**.h*) and one (or several) source (**.cpp*) files.
- Folder `lib`: This folder contains header files with our specifications for use of libraries and parallelization.
- Folder `output`: This folder receives all output files that are created during a run.
- File `main.cpp`: The main file. It runs our code for one specific choice of input parameters.
- File `makeElements.py`: Python script for creating the input files with two-particle elements using the OpenFCI [17] library. The corresponding folder `OpenFCI` is placed next to the `SRG` folder.
- File `Makefile`: Compiles our code by simply calling *make*.

- File `runSerial.py`: Python script for running the SRG code with one specific choice of input parameters at a time, usually several different runs after each other. The script calls the code of the OpenFCI library, creates the input files and places them in an appropriate folder, compiles the SRG code by calling the `Makefile` and runs the code with correct input parameters for the main file. That way, the user does not have to deal explicitly with the preparations of each SRG run.
- File `runParallel.py`: Python script with the same functionality as `runSerial.py`. Additionally, it opens up the possibility to run simulations with different input parameters (e.g. different oscillator frequencies ω) in parallel.
- File `mainParallel.cpp`: Wrapper that runs the main program in parallel, with the parameters specified by `runParallel.py`. The code is parallelized with MPI (Message Passing Interface).

8.3 Implementation of SRG - general aspects

In the following, we will present those aspects of our code that the free-space and the in-medium implementations have in common, before we move on and look at those parts that are specific for each implementation.

8.3.1 Class `System`

The first class used by both SRG codes is the class `System`, whose task it is to hold all data structures and methods that characterize a specific system. That way, it shall serve as communication point for the solver, in our case the SRG method, and make the program clearly structured and organized. It contains several other classes, like `Hamiltonian` and `Basis`, and is responsible for appropriate communication between those classes.
An important point is that the class `System` is supposed to embed those aspects that are specific for a system, whereas `Hamiltonian`, `Basis`, etc., are general classes that can be used for all kinds of systems, e.g. quantum dots, atoms and nuclear systems, and in arbitrarily many Euclidean dimensions.
In order to have a general program that can easily be adopted to different kinds of physical problems, we have constructed `System` as a virtual base class, which gives the possibility to implement system-specific functions in derived subclasses. One example is the labelling of single-particle states: For two-dimensional quantum dots with a harmonic oscillator basis, this looks as in figure 5.1, but already for a three-dimensional quantum dot, we would need a different mapping scheme. Therefore, the function `mapping` is virtual in the class `System`, and we implemented the concrete mapping of figure 5.1 in the derived subclass `System_2DQdot`.

8.3.2 Class `SPstate`

Another general class of our code, that can be used for arbitrary systems and bases, is the class `SPstate`. One instance of this class represents one specific single-particle state, characterized by specific quantum numbers and a single-particle energy. The basis of the Hamiltonian is

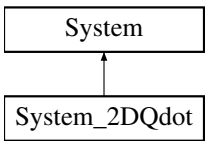

Figure 8.1: The class `System` is a virtual base class, and needs to be extended by subclasses containing system-specific methods and data structures.

then constructed as an array of those single-particle states where the number of states is chosen according to the size of the basis.
Since the algorithm specifying the quantum numbers is specific for each considered system, we have chosen to put the function mapping between a particular single-particle state and the associated quantum numbers, into the subclasses of `System`, rather than putting it into `SPstate` which should be as general as possible.
In the following, we will demonstrate how the mapping is performed for two-dimensional quantum dots, implemented in the subclass `System_2DQdot`.

Mapping the single-particle states for two-dimensional quantum dots Considering the shell structure of quantum dots illustrated in figure 5.1, we observe that each shell R corresponds to $R(R+1)$ possible single-particle states where at least one of the quantum numbers n, m, m_s is different.
Beginning from the lowest shell, each single particle state can be assigned an index α. We now perform a mapping

$$|\alpha\rangle \rightarrow |n, m, m_s\rangle \,,$$

such that each index α corresponds to a specific set of quantum numbers $n(\alpha), m(\alpha), m_s(\alpha)$. As before, the quantum number n is the nodal quantum number, m the azimuthal quantum number and m_s is the spin projection. An example for a two-dimensional quantum dot and the first four shells is given in table 8.1. To make our code compatible with the OpenFCI library [17], which generates our interaction elements, we have compared to figure 5.1 slightly changed the indexing within a shell R.
Studying this table, we observe a pattern which makes it possible to automatize the assignment of quantum numbers through a mapping algorithm. The first observation is that all states appear in pairs with positive and negative m_s, such that m_s can simply be modelled by an alternating sequence. The second, more interesting point is that for each shell R, there exist exactly R states with the same single-particle energy ϵ. Looping over those R states, the quantum number m starts with $m = 1 - R$ and increases in steps of two, always first with negative, then with positive sign. That way, we end up with the algorithm demonstrated in listing 8.1, which performs the mapping of quantum numbers for two-dimensional quantum dots and additionally computes the single-particle energies ϵ_i.

Concerning generalizability of our code, we first note that this mapping algorithm is specific for each considered system. As mentioned previously, designing the function `mapping` as virtual function allows to adopt it to the specific expressions for each subclass of `System`. Moreover, we have not hardcoded the quantum numbers $\{n, m, m_s\}$ of each single-particle state as three integers, but collected them in the array `qnumbers`, whose usage is demonstrated in listing 8.1.

α	n	m	m_s	R	α	n	m	m_s	R
0	0	0	-1	1	10	1	0	-1	
1	0	0	1		11	1	0	1	
2	0	-1	-1	2	12	0	-3	-1	4
3	0	-1	1		13	0	-3	1	
4	0	1	-1		14	0	3	-1	
5	0	1	1		15	0	3	1	
6	0	-2	-1	3	16	1	-1	-1	
7	0	-2	1		17	1	-1	1	
8	0	2	-1		18	1	1	-1	
9	0	2	1		19	1	1	1	

Table 8.1: Overview of the mapping between single-particle states α and corresponding quantum numbers n, m, m_s for a two-dimensional quantum dot with harmonic oscillator basis. Here we illustrate the first four shells. Each state α is assigned a specific set of quantum numbers, with allowed values given by $n = 0, 1, ...$ for the nodal quantum number, $m = 0, \pm 1, \pm 2, \ldots$ for the azimuthal quantum number and $m_s = \pm\frac{1}{2}$ for the spin projection, which we have transformed to $m_s = \pm 1$ such that only integers have to be stored in our program.

Listing 8.1: Code segment performing the mapping $|\alpha\rangle \rightarrow |n, m, m_s\rangle$. The array `qnumbers` contains the quantum numbers in the order n, m, m_s. For example, this means that in line 12, we access quantum number m_s of single particle state α, contained in the Basis **Bas**.

```
1  // Mapping between |alpha> and |n,m,m_s>
2
3  // Loop over all shells
4  for(int shell = 1; shell <=R; shell ++){
5
6          m_count = 1-shell;
7
8          for( int i = 0; i< shell; i++){
9
10             Bas->singPart[alpha].qnumbers[1] =  Bas->singPart[alpha+1].qnumbers
                   [1] = m_count; // m
11             Bas->singPart[alpha].qnumbers[0]  = Bas->singPart[alpha+1].qnumbers
                   [0] = i/2; // n
12             Bas->singPart[alpha].qnumbers[2] = -1; // m_s
13             Bas->singPart[alpha].eps = Bas->singPart[alpha+1].eps = shell*omega
                   ; //sp_energy
14             alpha++;
15             Bas->singPart[alpha].qnumbers[2] = +1; // m_s
16             alpha++;
17             if(i%2 == 0){
18                  m_count *= -1;
19             }
20             else{
21                  m_count = -m_count +2;
22             }
23         }
24     }
```

The dimension of this array is variable, such that we easily can include further quantum numbers, like the parity τ for nuclear systems.

8.4 Implementation specific for free-space SRG

As mentioned above, there are fundamental differences in the implementation of the free-space and the in-medium SRG method. This section serves to describe how we implemented the free-space case, which algorithms we used and which computational challenges we met in order to make the code as effective as possible.

8.4.1 Classes for the free-space case

Of the above mentioned classes of our code, we have so far described the classes `System` and `SPstate` as general classes. In the following, we will come to the classes `Hamiltonian`, `Basis` and `SRG`, that are different for the free-space and in-medium implementation. First, we give an overview of the structure of the classes `Hamiltonian` and `Basis`. To demonstrate the data flow and how the classes of our code work together, we give a detailed description of how a system is set up. Afterwards, we demonstrate how the solver, in our case the SRG method, is applied to the system. In the course of this, we describe the class `SRG`, which as many-body solver is independent of the previously described classes.

Class `Hamiltonian`

The main purpose of the class `Hamiltonian` is to set up the Hamiltonian matrix in a given basis and store it. When an instance of this class is created, memory for two arrays is allocated, one called `HO` for the elements corresponding to the non-interacting part of the Hamiltonian, and one called `HI` for the ones arising from the interaction part. Apart from an organized structure, the main reason for this subdivision is that during the flow of the Hamiltonian, only the interaction elements are changed, suggesting that the `HO`-array can be stored permanently. The member functions of `Hamiltonian` can be subdivided into two classes: The first group of functions serves to read in the one-particle and two-particle elements from file, and they must be provided in the form

$$a \quad b \quad \kappa \qquad \text{and} \tag{8.1}$$

$$a \quad b \quad c \quad d \quad \kappa, \tag{8.2}$$

respectively. Here indices a, b, c, d are integers denoting the single-particle states as illustrated in figure 5.1, whereas κ is a floating-point number with the value of the corresponding matrix element.
After the elements have been read in, the task of the second group of member functions is to use those elements to compute the initial Hamiltonian matrix. With the Hamiltonian operator given in second quantization, those functions mainly handle the action of creation and annihilation operators, including various bit operations. A detailed description is given in section 8.4.2, where we explain how a system is set up.

Class `Basis`

The class `Basis` handles everything concerning the basis in which the Hamiltonian matrix is set up. The two main data structures are an array holding the Slater determinant basis for the Hamiltonian, and an array containing all considered single-particle states until the made cut-off. That array contains objects of the type `SPstate`, which themselves contain all relevant quantum numbers and the single-particle energy. The single-particle states are in our case the ones of a harmonic oscillator basis or a Hartree-Fock basis based on harmonic-oscillator orbitals. The only difference with a Hartree-Fock basis is that the one-body and two-body elements (8.1) and (8.2) are in this case not directly obtained using the OpenFCI library, but from a preceding Hartree-Fock calculation.
Since the `Basis`-class is responsible to hold the Slater determinant basis, it has several member functions for setting this basis up, transforming it to binary representation, choosing the right states for a given channel etc. All these algorithms are explained in the following section, where we explain in detail how a specific system is set up.

8.4.2 Setting a system up

Setting a system up in free space involves two steps: First, the basis of Slater determinants is established, then the Hamiltonian matrix is set up in this basis.

Slater determinant basis

In principle, our Slater determinant basis consists of all possibilities to place a number of N particles in n_{sp} single-particle states (sp-states). For example, for two particles and four sp-states, we have the basis states

$$|0,1\rangle, |0,2\rangle, |0,3\rangle, |1,2\rangle, |1,3\rangle, |2,3\rangle,$$

where $|i,j\rangle$ denotes a determinant where single-particle state i and j are occupied by an electron and the remaining ones are unoccupied. In general, there are $\binom{n_{sp}}{N}$ possible Slater determinants and we need an algorithm to systematically create all combinations. For this purpose, we employ the so-called *odometer* algorithm, which works as follows:

Odometer algorithm

1. Start with the first N states occupied. (N= number of particles)
2. Repeat the following recursively until all N particles occupy the last N sp-states:
   (a) Loop with the last particle over all remaining sp-states.
   (b) Increase the position of the second last particle by 1.
   (c) ... Repeat steps (a) and (b) until end reached ...
   (d) Increase the position of the third last particle by 1.
   (e) ...

Following this procedure, we make sure to include all possible combinations. For $N = 3$ particles and $n_{sp} = 5$ sp-states, for example, the Slater determinants are produced in the following order:

$$|0,1,2\rangle, |0,1,3\rangle, |0,1,4\rangle, |0,2,3\rangle, |0,2,4\rangle, |0,3,4\rangle.$$

To obtain this series, we use the function `odometer`, which is part of the class `Basis`. Given an initial occupation scheme, the function moves the 'odometer' one step further.

Listing 8.2: The function `odometer` accepts the N-dimensional array `occ` with the current occupancy scheme of the N particles and returns the next occupancy scheme, using the odometer algorithm.

```
void Basis::odometer(ivec& occ, int numpart) {
    int l;
    // Loop over all partices, beginning with the last one
    for (int j = numpart - 1; j >= 0; j--) {
        if (occ(j) < sp_states - numpart + j) {
            l = occ(j);
            for (int k = j; k < numpart; k++) {
                occ(k) = l + 1 + k - j;
            }
            break;
        }
    }
}
```

To set up the whole Hamiltonian matrix, one theoretically has to store all possible Slater determinants. The problem is that the size of this matrix increases rapidly with the number of particles and sp-states, requiring large amounts of memory and CPU time in a subsequent diagonalization. Moreover, in the case of quantum dots, this matrix would is very sparse: Since the encountered Hamiltonian operator preserves spin and angular momentum, the matrix is block diagonal in these two quantities from the beginning on. Defining

$$M = \sum_i^N m(i), \qquad M_s = \sum_i^N m_s(i)$$

as the total angular momentum and total spin, respectively, the Hamiltonian does only link states $|\alpha\rangle$ and $|\beta\rangle$ with the same value for M and M_s. Hence, if we are only interested in the ground state energy, it would be an enormous overkill to diagonalize the whole Hamiltonian matrix.
Since we study only closed-shell systems in this thesis, we can assume that the ground state fulfils $M = M_s = 0$ and therefore we have to diagonalize solely that block of the Hamiltonian matrix that fulfils these two requirements. This simplification reduces the dimensionality of the problem significantly and makes it possibly to treat larger systems within the restrictions set by the limited memory of a given machine. For the case of $N = 6$ particles and $R = 4$ shells, for example, there exist $\binom{20}{6} = 38760$ possible Slater determinants. With double precision, the whole Hamiltonian matrix would require $8 \times 38760^2 \approx 12$GB of RAM. For ordinary computers having four nodes, each with 2GB of RAM, already this system with $R = 4$ shells would exceed the available memory. Restricting us to the states with $M = M_s = 0$, however, the Slater determinant basis involves only $n = 1490$ states, which decreases the required memory to about

17MB. Thus the memory requirement has been drastically reduced and the computation is possible even on ordinary laptops.

We implemented the setup of the Slater determinant basis as follows: Using the odometer algorithm, all possible Slater determinants are created. Each time a new occupation is computed, we check if that one satisfies the requirements on M and M_s. Only in this case, the Slater determinant is added to the array which saves the basis states. To make the code as general as possible, the function checking for the right values of M and M_s is not restricted to only those two quantities, but could also check for further quantum numbers.
In a more general context, one can have a system with q relevant quantum numbers, out of which q_λ define a block of the Hamiltonian matrix. Then there exists a mapping

$$(q_1, q_2, \ldots, q_n) \leftrightarrow (\lambda, \pi),$$

where λ is a transition channel, consisting of a specific set of preserved quantum numbers, and π denotes the configuration with specific values of the remaining quantum numbers. That way, for two general states $|\alpha\rangle$ and $|\beta\rangle$, the following relation holds:

$$_{\lambda',\pi'}\langle\alpha|\,\hat{H}\,|\beta\rangle_{\lambda,\pi} = 0, \qquad \text{if } \lambda' \neq \lambda.$$

This is just the mathematical formulation of saying that the Hamiltonian is block-diagonal. In the case of quantum dots, a channel λ is specified by the two quantum numbers M and M_s. However, for nuclear systems for example, we could add the parity τ as the third preserved quantum number. In order to treat those kinds of systems, too, we have opened up the possibility of including more quantum numbers in our code.

Especially with regard to systems with many particles, we do not store the Slater determinant basis in occupation representation, i.e. in the form

$$\begin{pmatrix} |0,1,2,3\rangle \\ |0,1,2,6\rangle \\ |0,1,3,8\rangle \\ \cdots \end{pmatrix},$$

but in binary representation, with the mapping

$$\begin{pmatrix} |0,1,2,3\rangle \\ |0,1,2,6\rangle \\ |0,1,3,8\rangle \\ \cdots \end{pmatrix} \leftrightarrow \begin{pmatrix} |2^0+2^1+2^2+2^3\rangle \\ |2^0+2^1+2^2+2^6\rangle \\ |2^0+2^1+2^3+2^8\rangle \\ \cdots \end{pmatrix} \leftrightarrow \begin{pmatrix} |15\rangle \\ |71\rangle \\ |267\rangle \\ \cdots \end{pmatrix}.$$

On the one hand, this approach saves memory since instead of N integers, just one integer per Slater determinant is saved. On the other hand, it makes the usage of time saving bit manipulation operations possible when the Hamiltonian acts on those basis states, as we will demonstrate in the next section.
However, the storage as integers has its limitations since an integer is restricted to 32 or 64 bits, depending on hardware and the operating system. In order to have the possibility to treat systems with more than $R = 7$ or $R = 10$ shells, respectively, we save the Slater determinants therefore saved in the form of *bitsets*, see [62]. This C++ class, which is part of the standard library, is a special container class designed to store bits. The number of stored bits can be assigned to an arbitrary value, in our case $R(R+1)$, with R denoting the number of shells. Further advantages of this class are the numerous methods for bit manipulation, such as setting and removing bits, or testing whether a specific bit is set.

Listing 8.3: Function to check if a Slater determinant fulfils the requirements of a specific channel. Input parameters are the vector `occ`, containing the occupied sp-states, the number of particles `numpart` and the vector `spes_channel`. This vector contains the values of the preserved quantum numbers in the same order as they appear in the array `qnumbers` (which holds the quantum numbers of each sp-state), starting with the second entry of `qnumbers` (the first entry we assume not to be conserved). In our case, `qnumbers` contains the quantum numbers n, m, m_s and if we are interested in the channel with $M = M_s = 0$, then the vector `spes_channel` has to equal $(0, 0)$. For other systems than quantum dots, with more quantum numbers, we can simply extend the array `qnumbers` and, if those quantum numbers define the channel, also the vector `spes_channel`.

```
1
2  // Return whether the occupation scheme "occ" has the correct channel
3  bool Basis::correct_channel(ivec& occ, int numpart,ivec& spes_channel) {
4      ...
5      int fixed_channel = spes_channel.size(); // number of quantum numbers to be
           checked is specified by the input vector
6
7      for(int i = 1; i<= fixed_channel; i++){
8          sum = 0;
9          for( int j = 0; j< numpart; j++){
10             sum += singPart[occ(j)].qnumbers[i];
11         }
12
13         if(sum != spes_channel(i-1))
14             return false;
15     }
16     return true;
17 }
```

Setting up the Hamiltonian matrix

In order to set up the Hamiltonian matrix, we have to compute all matrix elements

$$\langle\alpha|\,\hat{H}\,|\beta\rangle = \langle\alpha|\left(\sum_{pq}\langle p|\,\hat{h}^{(0)}\,|q\rangle\,a_p^\dagger a_q + \frac{1}{4}\sum_{pqrs}\langle pq||rs\rangle\,a_p^\dagger a_q^\dagger a_s a_r\right)|\beta\rangle$$

in the given basis of Slater determinants. To do this as effectively as possible, we proceed as summarized for the interaction elements in the box below. The procedure for the non-interacting part of the Hamiltonian is similar. In the case that $\hat{H}_0$ is diagonal, it is even easier since the machinery with creation and annihilation operators can be skipped and we simply have to add contributions on the diagonal.

Algorithm: Setup of the Hamiltonian matrix (interaction part)

Loop over all ket-states (right side of the Hamiltonian operator)

- Loop over all indices $i < j$ and $k < l$.
  1. Act with creation/annihilation operators $a_i^\dagger a_j^\dagger a_l a_k$ on the ket-state $|\beta\rangle$. If not zero, this yields a bra-state $\langle\alpha|$.
  2. Determine the index of $\langle\alpha|$ in the basis of Slater determinants.
  3. If the state is contained in our basis, extract the transition amplitude $\langle\alpha\,|\hat{v}|\,\beta\rangle$ from the elements that have been read in.
  4. Add this contribution to the matrix element $H_{\alpha\beta} = \langle\alpha|\,\hat{H}\,|\beta\rangle$ of the Hamiltonian matrix.

The step of acting with $a_i^\dagger a_j^\dagger a_l a_k$ on a ket-state $|\beta\rangle$ to obtain the bra-state $\langle\alpha|$ might need some additional explanation:
As explained above, we save all Slater determinants as a bit pattern. Acting with $a_i^\dagger a_j^\dagger a_l a_k$ on such a bit pattern means to remove bits k and l, and to add bits i and j. At the same time, it is necessary to keep track of the phase since for $i \neq j$ we have that $a_i a_j = -a_j a_i$ and $a_i^\dagger a_j^\dagger = -a_j^\dagger a_i^\dagger$. Our concrete procedure is therefore as follows:
First, we check whether both bits k and l are occupied, using the corresponding method of the C++ class `bitset`. If not, no further calculations are necessary since $a_p|\Phi\rangle = a_p a_{q_1}^\dagger a_{q_2}^\dagger \ldots a_{q_N}^\dagger|0\rangle = 0$ for $p \notin \{q_1, q_2, \ldots q_n\}$. If both bits are set, first bit k, then bit l are removed from the bit pattern. To keep track of the correct sign, each time a bit is removed, we count the number of occupied bits before that one and multiply the overall sign with (-1) for each occupied bit. Afterwards, the two creation operators $a_i^\dagger a_j^\dagger$ have to act on the state. Similar to the annihilation process, we first check whether the two bits i and j already are contained in the bit pattern. In this case, we return zero, since $a_p^\dagger|\Phi\rangle = a_p^\dagger a_{q_1}^\dagger a_{q_2}^\dagger \ldots a_{q_N}^\dagger|0\rangle = 0$ for $p \in \{q_1, q_2, \ldots q_n\}$. Otherwise, the two bits are added at the correct place using simple bit operations. Again we keep track of the overall sign by counting the number of occupied bits before the created ones. The results are the bra-state $\langle\alpha|$ in bit representation as well as the phase, which we afterwards have to multiply with the transition amplitude.

Determination of $|\alpha\rangle = \pm a_i^\dagger a_j^\dagger a_l a_k |\beta\rangle$

1. Test if bit k and l are occupied in the bit pattern of state $|\beta\rangle$. If not, return zero.
2. Remove first bit k, then bit l from state $|\beta\rangle$, keeping track of the phase as explained in the text.
3. Check if bit i or j are occupied in $|\beta\rangle$.
4. If yes, then return zero, otherwise add first bit j, then i, again following the phase.

Figure 8.2: Summary of the algorithm to determine the bra-state when acting with creation/annihilation operators on a ket-state.

8.4.3 Applying the SRG solver

When the Hamiltonian matrix is set up, the SRG method can be used to (block-)diagonalize it. Since the flow equations (6.15) represent a set of coupled ordinary differential equations (ODEs), we need an ODE solver that performs the integration. In this thesis, we use a solver based on the algorithm by Shampine and Gordon [41], provided as C++ version in [40].

ODE algorithm by Shampine and Gordon

In the following, we explain the basic concepts of the Shampine and Gordon ODE algorithm. Note that the complete algorithm is much more advanced and involves dealing with discontinuities and stiffness criteria, controls for propagated roundoff errors and detects requests for high accuracies. This makes it a very powerful tool for the solution of ordinary differential equations, but the code is rather difficult to read and the reader can easily get lost in all the details. Therefore, we will summarize the main procedure and ideas of the algorithm, without mentioning all the minor functions used for error control etc. For a detailed explanation, we refer instead to [41].

The implementation of the ODE algorithm as given in [40] involves three major functions: the core integrator `step`, a method `intrp` for interpolation and a driver `ode`.
To demonstrate the tasks of the different functions, suppose we have to solve a general ODE problem of the form

$$\begin{aligned} y_1'(t) &= f_1(t, y_1(t), y_2(t), \ldots, y_n(t)) \\ y_2'(t) &= f_2(t, y_1(t), y_2(t), \ldots, y_n(t)) \\ &\vdots \\ y_n'(t) &= f_n(t, y_1(t), y_2(t), \ldots, y_n(t)), \end{aligned}$$

with given initial conditions $y_1(a), y_2(a), \ldots, y_n(a)$. In a simplified notation, this can be summarized as

$$\mathbf{y}'(t) = \mathbf{f}(t, \mathbf{y}(t)), \tag{8.3}$$
$$\mathbf{y}(a) = \mathbf{y}_0. \tag{8.4}$$

In order to solve such a problem with an ODE solver, it is necessary to provide the solver with the complete specification of the problem. This involves the concrete equations (8.3), which should be supplied as a subroutine, the initial conditions $y_1(a), y_2(a), \ldots, y_n(a)$, the interval of integration $[a, b]$, as well as the expected accuracy and how the error should be measured. The integrator should then be able to return the solution at b, or, in the case that the integration failed, the solution up to the point where it failed and should report why it failed.
All this tasks, getting the input parameters, piecing all the minor functions together to a working code and at the same time taking care of the above mentioned criteria like stiffness and error propagation, are carried out by the driver function `ode`. It is intended for problems in which the solution is only desired at some endpoint b or a sequence of output points b_i, and the user gets no insight into the complicated calculations regarding the integration within $[a, b]$. However, for the user to know about success of the integration or how to possibly attain it in a next attempt, `ode` returns a flag which explicitly states why an integration failed. Possible causes that we encountered in our integrations, and that helped us to tune the input parameters, are too small error tolerances, too many required steps to reach the output point and the warning that the equations appear to be stiff. In the ideal case, the integration converges and we only have to specify the input parameters, call the `ode` function and collect the output results in the end.

The specific equations (8.3) must be supplied as subroutine of the form

$$f(t, y, dy), \tag{8.5}$$

where t is the current integration point, y an array containing the values $y_i(t), i = 1 \ldots n$ and dy an array with the same dimension as y, holding the derivatives. In the case of the flow equations, we have the function

```
void SRG::derivative(double lambda, double** v, double** dv),
```

for both the free space and the in-medium case, where `lambda` specifies the integration point, `v` holds the interaction elements and `dv` the corresponding derivatives.
On input, `ode` must be provided with all initial conditions, which means that the array `v` must contain the interaction elements of the initial Hamiltonian. On output, this array contains the (hopefully) converged interaction elements at the output integration point s.

$$\begin{aligned} \text{Input: } & v \leftarrow \langle pq||rs\rangle \\ \text{Output: } & v = v(s). \end{aligned}$$

The mathematically most fundamental routine of the ODE code is the function `step`, which is the basic integrator and advances the solution of the differential equations exactly one step at a time, i.e. from y_i to $y_{i+1} = y_i + h$. It uses a variable order version of the Adams method, using a predictor-corrector algorithm of type PECE (see section 4.2.2). Here the predictor is of order k and the corrector of order $k + 1$ [41].
To use Adams methods most effectively, the algorithm aims at using the largest step size h yielding the requested accuracy. Note that many ODE codes require the user to guess an initial step size. This can be a source of error if the user does not know how to estimate a suitable value. The ODE algorithm by Shampine and Gordon therefore includes a function to estimate this step size automatically, trying to limit the problem to a minimum of function evaluations,

but at the same time maintaining stability. The determination of the step size is motivated by experimentation done by F.T.Krogh [64] and C.W.Gear [65] and bases on various error predictions and interpolation. Concerning selection of the right order, the algorithm tends to use lower orders since they exhibit better stability properties. The order is only raised if this is associated with a lower predicted error.
Starting with lowest order k, the typical behaviour is that after the next order $k+1$ has been reached, the step size will double on successive steps until a step size appropriate to the order $k+1$ is attained. Then $k+1$ steps of this size are taken and the order is raised. This procedure is repeated until an order appropriate to the problem and with lowest possible error has been found.
After the step size as well as the order have been specified, the function `step` advances the solution of the differential equation one step further. During the propagation, the local error is controlled according to a criterion that bases on a generalized error per unit step. For details we refer to [41].

With a step size determined by error propagation and stability of the integration process, it is very likely that the integration points of the algorithm do not coincide with the desired output point s. Moreover, results are most accurate if the equations are integrated just beyond that point s and then interpolated. For that reason, the code contains additionally the function `intrp`, which performs an interpolation to obtain the solution at the specified output points.

Class `SRG`

All parts of our program related to the flow of the Hamiltonian are handled by the class `SRG`. The task of this class is to be given a system with a Hamiltonian and solve the flow equations (6.15) as a system of coupled ODEs.
The most important class member is of type `System`, which serves as communication point to system-specifying objects of type `Hamiltonian`, `Basis` etc. The central function `run_algo` performs the integration, employing the ODE-solver of Shampine and Gordon. We adopted the original version of the C++ code, see [40], to our needs, such that it fits our system of data storage and our derivative function. The modified functions of the code we included in our class `SRG`.

As stated above, the ODE-solver needs to be supplied with initial conditions, integration interval, specifications of error limits, as well as the concrete derivatives (8.3). In our code, the initial conditions are stored in the Hamiltonian matrix and the integration interval, as well as the limits for the relative and absolute error, are transferred as input parameters to the function `run_algo`. As already mentioned in the previous section, we have a separate function, called `derivative`, that computes the derivatives at a each point of integration.
Concerning the integration interval, we theoretically have to integrate down to $\lambda = 0$. Practically, we can stop the integration when the ground state energy E_0 does not change any more within a user-defined tolerance. For this reason, we specify for each run a sufficiently small value for λ, after which the integration shall stop in certain steps of length $\Delta\lambda$ and we make out whether the change of E_0 lies within the given bounds.
On the one hand, this step size $\Delta\lambda$ must be chosen small enough that the integration is not performed unnecessarily close to the zero point, because the required CPU time increases drastically with the length of integration, as discussed in the next chapter. On the other hand, it

Listing 8.4: The function `derivative` serves as a wrapper, picking the generator-specific derivative-computing function from an array. All functions in this array must have the same signature as in Eq. (8.5). Here `lambda` denotes the integration point, and `v` and `dv` are arrays containing the interaction elements and corresponding derivatives, respectively.

```
void SRG::derivative(double lambda, double** v, double** dv) {
    (this->*eta)(lambda, v, dv);
}
```

Listing 8.5: Demonstration how we practically implemented the array containing the derivative-functions `deriv_eta1` and `deriv_eta2` for two different generators. When an instance of the class `SRG` is created, the integer `eta_choice` determines which of the functions shall be used when `derivative` is called.

```
typedef void (SRG::*fptr)(double k_lambda, double **v, double **dv);
    static const fptr eta_table[2];
    fptr eta;

const SRG::fptr SRG::eta_table[] = {
    &SRG::deriv_eta1, &SRG::deriv_eta2
};

// in constructor:
SRG::SRG(..., int eta_choice) {
    ...
    eta = eta_table[eta_choice];
    ...
}
```

should be chosen large enough that as long as convergence has not been reached, differences in E_0 exceed the tolerance for convergence. Additionally, the step size should be sufficiently large such that the integration does not need to stop too often for interpolation.
We solved these requirements in such a way, that we usually start with a step length of $\Delta\lambda = 0.1$ after we stop for the first check and integrate to maximally $\lambda = 0.1$. If the ground state has still not converged, we decrease the step length and approach the zero point in smaller steps.

As explained in chapter 6, there are different possible generators $\hat{\eta}$ driving the Hamiltonian to diagonal form, or at least decoupling the ground state. For SRG in free space, we consider the simple generator $\hat{\eta}_1 = \left[\hat{T}_{\mathrm{rel}}, \hat{V}\right]$ and Wegner's canonical generator $\hat{\eta}_2 = \left[\hat{H}^{\mathrm{d}}, \hat{H}^{\mathrm{od}}\right]$. When running our code, the only computational difference between both generators occurs when the derivatives (6.15) are computed. Therefore, we designed the derivative-function as a kind of wrapper that chooses the generator-specific function from an array containing the functions for all considered generators. This structure makes it enormous easy to include further generators: One simply has to extend the array by a further derivative-function. Listings (8.4) and (8.5) demonstrate how we practically implemented this choosing of generator.
The concrete derivative-functions, the first one for $\hat{\eta}_1 = \left[\hat{T}_{\mathrm{rel}}, \hat{V}\right]$ and the second one for $\hat{\eta}_2 = \left[\hat{H}^{\mathrm{d}}, \hat{H}^{\mathrm{od}}\right]$, are presented in listings (8.6) and (8.7). Since there is only a minimal

Listing 8.6: Derivative function for generator $\hat{\eta}_1$. The function receives the current interaction elements at `lambda` in the array `v`, and returns the corresponding derivatives in the array `dv`. The calculations are performed according to Eq. (6.15). Via the object `Sys` of type `System`, the Hamiltonian `H` can effectively be accessed. Further explanations are given in the text.

```
void SRG::deriv_eta1(double lambda, double** v, double** dv) {

    ...

    k3 = lambda * lambda*lambda;

#pragma omp parallel for private(i,k12,j,sum,k) schedule(dynamic)
    for (i = 0; i < n; i++) {
         for (j = i; j < n; j++){

            sum = 0.0;
            k12 = Sys->H->H0[i] + Sys->H->H0[j];
            for (k = 0; k < i; k++)
                sum += (k12 - 2.0 * Sys->H->H0[k]) * v[k][j] * v[k][i];
            for (k = i; k < j; k++)
                sum += (k12 - 2.0 * Sys->H->H0[k]) * v[k][j] * v[i][k];
            for (k = j; k < n; k++)
                sum += (k12 - 2.0 * Sys->H->H0[k]) * v[j][k] * v[i][k];

            dv[i][j] = sum - (Sys->H->H0[j] - Sys->H->H0[i])*(Sys->H->H0[j] -
                Sys->H->H0[i]) * v[i][j];
            dv[i][j] *= -2.0 / k3;
        }
    }

    return;
}
```

Listing 8.7: Derivative function for Wegner's generator $\hat{\eta}_2$. The function has the same signature as the one for generator $\hat{\eta}_1$. Analogously, it receives the current interaction elements at the integration point `lambda` in the array `v` and returns the corresponding derivatives in the array `dv`. The calculation is performed according to Eq. (6.19).

```
void SRG::deriv_eta2(double lambda, double** v, double** dv) {

    ...

    k3 = lambda * lambda*lambda;

#pragma omp parallel for private(i,k12,j,sum,k) schedule(dynamic)
    for (i = 0; i < n; i++) {
         for (j = i; j < n; j++){

            sum = 0.0;
            k12 = Sys->H->H0[i] + Sys->H->H0[j] + v[i][i] + v[j][j];
            for (k = 0; k < i; k++)
                sum += (k12 - 2.0 * (Sys->H->H0[k] + v[k][k])) * v[k][j] * v[k
                    ][i];
            for (k = i+1; k < j; k++)
                sum += (k12 - 2.0 * (Sys->H->H0[k] + v[k][k])) * v[k][j] * v[i
                    ][k];
            for (k = j+1; k < n; k++)
                sum += (k12 - 2.0 * (Sys->H->H0[k] + v[k][k])) * v[j][k] * v[i
                    ][k];

            dv[i][j] = sum - (Sys->H->H0[j] + v[j][j] - Sys->H->H0[i] - v[i][i
                ])*(Sys->H->H0[j] + v[j][j] - Sys->H->H0[i] - v[i][i]) * v[i][j
                ];
            dv[i][j] *= -2.0 / k3;
        }
    }
    return;
}
```

difference between the final flow equations (6.15) and (6.19), our derivative functions look quite similar, too. The only difference lies in some additional terms for Wegner's generator in lines 14,16,18 and 20.
These two functions are well suited to demonstrate how the `SRG`-class communicates with the Hamiltonian via the class `System`: If we want to access the kth element of the array `H0`, containing the diagonal elements of the Hamiltonian, we call `Sys->H->H0[k]`. This means that we enter the Hamiltonian `H` of our system `Sys`, and of this Hamiltonian the array `H0` is called. All accessing happens via pointers, minimizing overhead.
Concerning lines 13-18, it might seem unclear why we split the sum into three parts. The reason is that for optimal efficiency, we make use of our Hamiltonian being symmetric and therefore only store and consider its upper triangular part in our calculations. This reduces the number of coupled differential equations to nearly one half, which lowers the required CPU time of our program.
Line 7 demonstrates how we parallelized these functions with OpenMP, selecting a dynamic schedule to ensure equal work balance in spite of the triangular form. For more information about syntax and scheduling in OpenMP, we refer to [66].

8.5 Implementation specific for in-medium SRG

Having explained our implementation for free-space SRG, we will now proceed in a similar manner for the in-medium implementation. This section will discuss all classes that differ from the free-space case, present the algorithms we used and point out the steps we have taken to make the code as effective as possible, at the same time maintaining generalizability.

8.5.1 Classes for the in-medium case

For consistency, we will present the relevant classes in a similar order as we did for the free-space case. First, we will present the two classes `Hamiltonian` and `Basis`, that are components of each `System`. In particular, we will demonstrate how a smart arrangement of the basis can be used to store the matrix elements in a way that saves memory space of several orders of magnitude and at the same time leads to a considerable reduction of the problem's dimension. Afterwards, we will turn to the implementation of the SRG method, performed by the class `SRG`, and show how the structure of our basis additionally saves a great number of floating point operations when evaluating the flow equations.

Class `Basis`

The purpose of the class `Basis` is to create and administer a two-particle basis (tp-basis), where one such a basis state is determined by two single-particle states (sp-states). For this idea, we have been inspired by [24, 67], since we made the observation that from a computational point of view, many functions of the SRG algorithm resemble ones used in Coupled Cluster implementations. Since M.H. Jørgensen and C. Hirth demonstrated in their theses the enormous speed-up gained by such a tp-basis, we decided to use a similar basis and storage system for our matrix elements, enabling us to make us of the associated benefits.

Similar to the free-space case, one central data structure of the class `Basis` is an array holding all sp-states for the number of considered shells R. As before, the items are of the type `SPstate`, containing all quantum numbers, single-particle energy etc. At a later point of this section, we will demonstrate how this array is needed when setting up the tp-basis.
The next important members of our class `Basis` are three arrays holding this tp-basis. The idea is to have a mapping

$$(M, M_s) \leftrightarrow \lambda,$$

where $M = m_{(1)} + m_{(2)}$ and $M_s = m_{s(1)} + m_{s(2)}$ are the overall angular momentum and spin projection of the two particles, respectively. Each value of the so-called *channel* λ corresponds to one specific combination of M and M_s, and we tabulate those combinations of two sp-states that fulfil the requirements on M and M_s. Hence, per two-particle state, we save the indices of two sp-states. Since needed later, we differentiate whether the indices lies above or below the Fermi level and therefore have three arrays: `hhbasis` for both indices corresponding to holes lying below the Fermi level, `ppbasis` for both indices corresponding to particles lying above the Fermi level and `phbasis` if one of the indices lies above and the other one below the Fermi level. For efficiency reasons, i.e. to keep the basis as small as possible, we just store one of the possible combinations in the case that both indices refer to particles or holes.

Listing 8.8: Declaration of the three arrays saving the two-particle basis. Since we do not know the number of states in each channel in advance, we make use of the class `vector` of the standard C++ library. The number of single-particle states, however, is always fixed to two, enabling us to use the type `ivec2` of the Armadillo library.

```
std::vector<arma::ivec2> *hhbasis, *ppbasis, *phbasis;
```

The algorithm for the mapping is based on the one presented in [67], and we demonstrate it for the case of one particle and one hole in listing 8.9. In this code excerpt, we make use of the maximal M-value, which for two particles is given by

$$M_{max} = 2 \cdot (R - 1).$$

With

$$\begin{aligned} M &= 0, \pm 1, \pm 2, \ldots, \pm M_{max} \\ M_s &= -1, 0, +1, \end{aligned}$$

the total number of channels λ is

$$\texttt{lbd\_dim} = 2 \cdot M_{max} \cdot 3 + 3.$$

In line 7 of listing 8.9, we now allocate memory for storing the tp-basis for each possible channel λ. Then, we loop over all possible values for M and M_s to determine those two sp-states that fulfil the requirements for each channel. Since the demonstrated function creates the `ph`-basis, the first index i must correspond to a particle, and the second index j to a hole. Using the quantum numbers stored in the array `singPart`, which contains all sp-states, we obtain M and M_s (see lines 19-20), and if they correspond to the ones of the current channel λ, we add the particle-hole combination to our basis (lines 24-25). For the `hh`- and `pp`-basis, we proceed analogously.

Listing 8.9: Setting up the two-particle basis for the case of one particle and one hole. A detailed explanation is given in the text.

```
void Basis::create_ph() {
    ...
    int counter = 0;
    int lbd = 0;

    // Allocating space for each channel
    ph_basis = new std::vector<arma::ivec2>[lbd_dim];

    // Loop over all possible values of M
    for (int l = -Mmax; l <= Mmax; l++) {
        channel(0) = l;

        // Loop over all possible values of M_s (since m_s is stored as integer
            , here -2,0,+2)
        for (channel(1) = -2; channel(1) <= 2; channel(1)+=2) {

            for (int i = hole_states; i < sp_states; i++)  // i = particle
                for (int j = 0; j < hole_states; j++) {  // j = hole

                    M = singPart[i].qnumbers[1] + singPart[j].qnumbers[1];
                    Ms = singPart[i].qnumbers[2] + singPart[j].qnumbers[2];

                    // If channel is correct, add particle-hole combination
                    if (M == channel(0) && Ms == channel(1)) {
                        contr << i << j << endr;
                        ph_basis[lbd].push_back(contr);
                    }
                }
            lbd++;
        }
    }
}
```

Listing 8.10: Mapping between a pair of one particle and one hole and the corresponding index in the two-particle basis. For the case that the channel λ is not known in advance, it can be obtained by the function `get_lbd`, which has to be provided with the quantum numbers M and M_s and returns the channel λ.

```
int Basis::map_ph(int p, int h, int lbd){

    for(int i = 0; i< ph_basis[lbd].size(); i++)
        if( (p == ph_basis[lbd][i](0)) && (h == ph_basis[lbd][i](1))){
            return i;
        }
    ...
}

int Basis::get_lbd(int M, int Ms){
    return (M+Mmax)*3+(Ms/2)+1;   // Note: for efficiency reasons Ms/2={-1,0,1}
}
```

Those three functions establishing the tp-basis constitute one group of member functions in our class `Basis`. Another important group of functions is responsible to map between a certain particle and/or hole combination and the corresponding index in the two-particle basis. As shown later in this section, this is needed when we want to access a specific matrix element of the Hamiltonian.
All three functions are structured the way

Input: sp-index 1, sp-index 2, channel λ
Output: index in tp-basis.

An example for the `ph`-basis is given in listing 8.10.
Using the tp-basis, we often have to loop over all basis states, and this looping should therefore be as effective as possible. Studying figure 5.1, it gets intuitively clear that just for the case of the `pp`-basis, all possible channels λ are exploited. For the `ph`- and especially the `hh`-basis, a much smaller, central range is needed. To increase efficiency when looping over the states, we therefore save for each of the three bases the first and last occupied channel λ. An extract of the corresponding function for the `ph`-basis is given in listing 8.11.

Class `Hamiltonian`

The task of the class `Hamiltonian` is to set up the initial Hamiltonian matrix and administer the access to its elements.
The two central data structures are an object of type `Basis`, by which the Hamiltonian communicates with the basis, as well as an array called `mat_elems`, which contains all the matrix elements.
Considering the computational parallels to Coupled Cluster calculations, we have, as mentioned before, been inspired by [24,67] and save the elements in a special, very effective way: First of all, we aim to make use of the symmetries of the interaction elements,

$$\langle pq||rs\rangle = -\langle qp||rs\rangle = -\langle pq||sr\rangle = \langle qp||sr\rangle, \tag{8.6}$$

Listing 8.11: Getting the first and last occupied channel λ, here an excerpt for the `ph`-basis. The boundaries are saved in an integer-matrix, which is returned by the function `lbd_limits`.

```
imat Basis::lbd_limits(){

    ...

    // Lower bound ph_basis
    lbd = 0;
    while(ph_basis[lbd].size() == 0)
        lbd++;
    mat_ret(1,0) = lbd;

    // Upper bound ph_basis
    lbd = lbd_dim-1;
    while(ph_basis[lbd].size() == 0)
        lbd--;
    mat_ret(1,1) = lbd;

    ...
}
```

enabling us to store only one of four possibilities. This reduces the required memory space, and later also the number of flow equations, considerably. Second, we utilize that our two-body interaction is modelled by a Coulomb interaction, which is spherically symmetric and independent of spin. For that reason the angular momentum, as well as the spin, are conserved and we obtain an additional criterion for the two-particle elements $\langle pq||rs\rangle$:
Defining

$$M = m_{(p)} + m_{(q)}, \quad M' = m_{(r)} + m_{(s)}$$
$$M_s = m_{s(p)} + m_{s(q)}, \quad M'_s = m_{s(r)} + m_{s(s)},$$

each two-particle state can be characterized by the two quantum numbers M and M_s:

$$|pq\rangle \leftrightarrow |M, M_s\rangle.$$

Accounting for the conservation of quantum numbers, we then have

$$\langle M, M_s|\hat{v}|M', M'_s\rangle = 0 \quad \text{if } M \neq M' \text{ or } M_s \neq M'_s.$$

In other words, only transitions between two states of the same channel λ yield a non-zero result. For that reason, we store the Hamiltonian in block-diagonal matrices, one block for each channel λ. To make us of the tp-basis and the fact that the flow equations (6.22)-(6.24) depend on whether the indices correspond to particle or hole states, we differentiate between the following combinations:

$$\begin{aligned} v_{hhhh} &\mathrel{\widehat{=}} \langle ij||kl\rangle, \\ v_{phhh} &\mathrel{\widehat{=}} \langle aj||kl\rangle, \\ v_{pphh} &\mathrel{\widehat{=}} \langle ab||kl\rangle, \\ v_{phph} &\mathrel{\widehat{=}} \langle aj||cl\rangle, \\ v_{ppph} &\mathrel{\widehat{=}} \langle ab||cl\rangle, \\ v_{pppp} &\mathrel{\widehat{=}} \langle ab||cd\rangle. \end{aligned} \tag{8.7}$$

Here p denotes particles above the Fermi level, h holes below the Fermi level and, as before, indices $\{a, b, c, d\}$ and $\{i, j, k, l\}$ designate particle and hole states, respectively. These six combinations cover all basic possibilities for the interaction elements. For other ones, e.g. v_{hphh}, we exploit the symmetry relations (8.6), and obtain

$$\begin{aligned} & v_{hhhh}, \\ & v_{phhh} = -v_{hphh} = -v_{hhhp} = v_{hhph}, \\ & v_{pphh} = v_{hhpp}, \\ & v_{phph} = -v_{hpph} = -v_{phhp} = v_{hphp}, \\ & v_{ppph} = -v_{pphp} = -v_{hppp} = v_{phpp}, \\ & v_{pppp}. \end{aligned} \tag{8.8}$$

That way, we have all $2^4 = 16$ possible combinations. Tabulating only the six matrices of Eq. (8.7) therefore provides all needed information.
The v-elements are stored in the tp-basis established in the class `Basis`. As an example, take an element of type $v_{phph} \mathrel{\widehat{=}} \langle aj||cl\rangle$. Since we store the Hamiltonian in block-form, the states $|aj\rangle$ and $|cl\rangle$ need to belong to the same channel λ. For each λ, the matrix v_{phph} is of size $s_{ph(\lambda)} \times s_{ph(\lambda)}$, where $s_{ph(\lambda)}$ denotes the number of particle-hole combinations the `ph`-basis holds for this value of λ. For a specific channel, the matrix element $v_{phph}(0,0)$ then contains the transition amplitude from the first to the first item in the corresponding `ph`-basis, the element $v_{phph}(0,1)$ the one from the first to the second, etc.

For the f-elements of the Hamiltonian, see Eq. (3.51), we have the symmetry relation

$$f_{pq} = f_{qp}.$$

To make use of this relation, we store the f-elements in three matrices:

$$\begin{aligned} f_{hh} &\mathrel{\widehat{=}} f_{ij}, \\ f_{ph} &\mathrel{\widehat{=}} f_{ai}, \\ f_{pp} &\mathrel{\widehat{=}} f_{ab}, \end{aligned} \tag{8.9}$$

with the same notation for the indices as before, and where f_{hh} and f_{pp} are symmetric such that only the upper triangular part has to be computed explicitly. Elements of the type f_{hp} can be obtained via $f_{hp} = f_{ph}$.

We store the complete Hamiltonian in the array `mat_elems`, whose items are matrices, handled by the Armadillo library.

```
std::vector<arma::mat> mat_elems;
```

The first item of this array holds the ground state energy E_0, the three next ones the matrices f_{hh}, f_{ph} and f_{pp}. The remaining items contain the v-elements. Starting with the lowest channel λ, we have for each channel the six matrices of Eq. (8.7).

To give an example of how this storage pattern reduces the required memory space by taking into account sparsity and symmetry relations, consider the case of $N = 2$ particles and $R = 15$ shells, corresponding to 110 single-particle states. If stored in a straightforward way, with the

Listing 8.12: Access to f-elements via the function `get_f_elem`. The main task is to determine which of the three possible matrices f_{hh}, f_{ph} and f_{pp} has to be used.

```
1  double Hamiltonian::get_f_elem(int p, int q, std::vector<arma::mat>& mat_elems)
       const {
2
3      // f_hh
4      if ((p < Bas->hole_states) && (q < Bas->hole_states))
5          return mat_elems[1](p, q);
6
7      // f_ph
8      if ((p >= Bas->hole_states) && (q < Bas->hole_states))
9          return mat_elems[2](p - Bas->hole_states, q);
10
11     // f_hp
12     if ((p < Bas->hole_states) && (q >= Bas->hole_states))
13         return mat_elems[2](q - Bas->hole_states, p);
14
15     // f_pp
16     return mat_elems[3](p - Bas->hole_states, q - Bas->hole_states);
17
18 }
```

f-elements in one two-dimensional and the v-elements in one four-dimensional matrix, one would need $110^4 + 110^2$ elements, corresponding to approximately 1.2 GB of memory. The required memory for our array, however, is only about 76 MB. The benefit is larger the more shells R the system includes. Apart from a significant reduction of space in memory, the number of ODEs to be solved is thereby significantly reduced, too.

To obtain access to a specific matrix element, the class `Hamiltonian` provides appropriate functions, called `get_f_elem` and `get_v_elem`. Obtaining the f-element f_{pq} is rather straightforward: Having determined whether the indices correspond to particle or holes states, the element can easily be extracted from one of the matrices f_{hh}, f_{ph} or f_{pp}, as shown in listing 8.12.

Accessing the v-elements is a bit more complicated since the elements are stored in the two-particle basis and according to channels λ. An excerpt of the function `get_v_elem` is given in listing 8.13. To obtain a specific interaction element v_{pqrs}, the function receives the four sp-state indices p, q, r, s. In the function, we first check whether the quantum numbers M and M_s are conserved, see lines 5-13. If not, the element has not been stored anyway, and we can return zero. Afterwards (see lines 15-23) we bring the indices into right order, since we only store one of the possible combinations for the `hh`- and `pp`-basis, as explained before. If, for example, we have stored the combinations $|p_1p_2\rangle$ and $|h_1h_2\rangle$ in our basis and are interested in the matrix element $v_{p_2p_1h_1h_2} = \langle p_2p_1||h_1h_2\rangle$, we have to make use of the symmetry relations of Eq. (8.8) and ask for the element $v_{p_1p_2h_1h_2} = -v_{p_2p_1h_1h_2}$.
As a next step, we extract the right channel λ, see line 25. Knowing the channel and having the indices in the right order, we can then access the matrix elements of our array `mat_elems`. However, before we finally get the elements, the challenge is to extract which of the six combinations of Eq. (8.7) has to be used. We therefore have to know explicitly which of the indices correspond to hole and which ones to particle states. With this information, we can make use of the methods `map_hh`, `map_ph` and `map_pp` of our class `Basis`, giving us the index in the

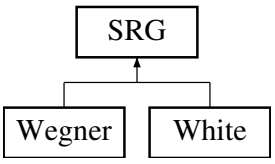

Figure 8.3: The class `SRG` is a virtual base class and needs to be extended by subclasses containing the specific expressions for the flow equations, depending on the generator $\hat{\eta}$.

tp-basis. Finally, we have enough knowledge to really extract the element. The many `if-else` statements in listing 8.13 might look a bit clumsy, but they enable us to formulate the function in such a way, that we have as few checks concerning the correspondence to particle or hole states, as possible.
For our example $v_{p_2p_1h_1h_2}$, we would now end up in the loop of lines 49-53, where p and q are particles, and r and s are holes. There, the function `map_pp` gives the tp-basis index for $|p_1p_2\rangle$, the function `map_hh` the one for $|h_1h_2\rangle$, and considering the change of sign due to the ordering $|p_2p_1\rangle \to |p_1p_2\rangle$, the element can be accessed.

Apart from storing the Hamiltonian matrix, the class `Hamiltonian` is responsible for setting this matrix up. The v-elements can be stored the way they are, whereas the f-elements and ground state energy E_0 have to be computed according to Eqs. (3.51) and (3.53), respectively.

Class `SRG`

As explained for the free-space case, the class `SRG` is the solver-class of our program. Given a `System` with an initialized Hamiltonian, it solves the flow equations (6.22)-(6.24). Since the concrete terms of these flow equations depend on the generator $\hat{\eta}$, we designed the class `SRG` as virtual base class, similar to the class `System`. It needs to be extended by subclasses for each used generator $\hat{\eta}$, containing the specific expressions for the flow equations. This idea allows us to make use of the benefits that object-orientation in C++ offers:
On the one hand, it is uncomplicated to switch between different generators. On the other hand, the program can easily be extended by further generators. In this thesis, we consider Wegner's and White's generator and have therefore implemented the corresponding subclasses `Wegner` and `White`.

The central data structures of the base class `SRG` are an object of type `System`, containing all system-specific information, and an array `eta`, where the values of the generator are stored during the integration. The size and structure of this array depend on the specific generator and are specified in the subclasses.
The function `run_algo` performs the integration, using the ODE-solver of Shampine and Gordon. It is analogous to the free-space SRG, and concerning the determination of integration limits, step length etc., we therefore refer to subsection 8.4.3.

Listing 8.13: Function for accessing the v-elements. See text for detailed explanation.

```
double Hamiltonian::get_v_elem(int p, int q, int r, int s, std::vector<arma::
    mat>& mat_elems) const {
    ...
    int sign = 1;

    int M1 = Bas->singPart[p].qnumbers[1] + Bas->singPart[q].qnumbers[1];
    int M2 = Bas->singPart[r].qnumbers[1] + Bas->singPart[s].qnumbers[1];

    if (M1 != M2) return 0; // Check conservation of angular momentum

    int Ms1 = Bas->singPart[p].qnumbers[2]+ Bas->singPart[q].qnumbers[2];
    int Ms2 = Bas->singPart[r].qnumbers[2]+ Bas->singPart[s].qnumbers[2];

    if (Ms1 != Ms2) return 0; // Check conservation of spin

    // Bring the indices into the right order
    if (p < q) {
        tmp = p;
        p = q;
        q = tmp;
        sign *= -1;
    }

    ... // The same for indices r and s

    int lbd = Bas->get_lbd(M1, Ms1); // Extract correct channel

    if (p < Bas->hole_states) {
        if (q < Bas->hole_states) {
            if (r < Bas->hole_states) {
                if (s < Bas->hole_states) {

                    // v_hhhh
                    comb1 = Bas->map_hh(p, q, lbd);
                    comb2 = Bas->map_hh(r, s, lbd);
                    if (comb1 < 0 || comb2 < 0) return 0;
                    return sign * mat_elems[4 + 6 * lbd](comb1, comb2);
                } else {

                    // v_hhhp -> v_phhh
                    comb1 = Bas->map_ph(s, r, lbd);
                    comb2 = Bas->map_hh(p, q, lbd);
                    if (comb1 < 0 || comb2 < 0) return 0;
                    return -sign * mat_elems[5 + 6 * lbd](comb1, comb2);
                }
            }
        } else { ...
            } else { ...
            ...
                    // v_pphh
                    comb1 = Bas->map_pp(p, q, lbd);
                    comb2 = Bas->map_hh(r, s, lbd);
                    if (comb1 < 0 || comb2 < 0) return 0;
                    return sign * mat_elems[6 + 6 * lbd](comb1, comb2);}
        }
    }
}
```

Class `Wegner`

The class `Wegner` is the subclass of `SRG` which is designed for Wegner's canonical generator $\hat{\eta} = \left[\hat{H}^{\mathrm{d}}, \hat{H}^{\mathrm{od}}\right]$. It extends `SRG`, inheriting data structures and functions like the central function `run_algo`, and adds those parts that are generator-dependent. Dependency on the generator $\hat{\eta}$ occurs in two stages: First, for each integration step, the elements of $\hat{\eta}$ have to be updated and stored; then the derivatives (6.22)-(6.24) are computed with these values.

In the following, we will explain those two stages in detail, including how elements are stored and accessed in our program. Afterwards we will demonstrate how we improved efficiency, both with respect to CPU time and memory.

Updating $\hat{\eta}$ In principle, there exist two possibilities to handle the generator $\hat{\eta}$ when computing the flow of the Hamiltonian: One possible way is to determine the elements under way, which means to start evaluating the flow equations (6.22)-(6.24) straightforwardly, and each time an element of $\hat{\eta}$ is needed, that one is computed according to Eqs. (6.29) and (6.30). The advantage would be not to use any memory for storing the $\hat{\eta}$-elements. However, since the flow equations require many elements several times, lots of redundant calculations would be the consequence. In fact, in one of our very first implementations to test the SRG method, we proceeded in such a way and realized that the redundant calculations made the program unacceptably slow. The second, now used approach, is to first compute all elements of $\hat{\eta}$, store them, and then continue with the derivatives using these elements.

To store the $\hat{\eta}$-elements most efficiently, and with an uncomplicated way of looping over and accessing them, we decided on the same structure that we store the matrix elements of the Hamiltonian in. The reasons are as follows:
First of all, the organization of the elements is similar, with the one- and two-body elements of the generator $\hat{\eta}$ corresponding to the f- and v-elements of the Hamiltonian $\hat{H}$, respectively. Moreover, similar to the v-elements of $\hat{H}$, we aim to store the two-body elements $\eta^{(2)}_{pqrs}$ in our tp-basis. Since the generator, basing on the Hamiltonian via the relation $\hat{\eta} = \left[\hat{H}^{\mathrm{d}}, \hat{H}^{\mathrm{od}}\right]$, conserves angular momentum and spin, too, we intend to minimize memory for storage by saving $\hat{\eta}$ in the same block form. As a last motivation, required memory can further be reduced by making use of the relation $\hat{\eta}^{\dagger}_{s} = -\hat{\eta}_{s}$, which apart from the sign is analogous to the symmetry of $\hat{H}$, giving the possibility to use the same storage system.

In analogy to the f-elements of $\hat{H}$, we save the one-body elements $\eta^{(1)}_{pq}$ in three matrices $\eta^{(1)}_{hh}, \eta^{(1)}_{ph}, \eta^{(1)}_{pp}$, depending on whether the indices correspond to particle or hole states. Elements of the form $\eta^{(1)}_{hp}$ can be obtained via $\eta^{(1)}_{hp} = -\eta^{(1)}_{ph}$.

For the two-body elements $\eta^{(2)}_{pqrs}$, we save analogous to relations (8.7) the six combinations

$$\begin{aligned}
\eta^{(2)}_{hhhh} &\widehat{=} \langle ij|\hat{\eta}|kl\rangle, \\
\eta^{(2)}_{phhh} &\widehat{=} \langle aj|\hat{\eta}|kl\rangle, \\
\eta^{(2)}_{pphh} &\widehat{=} \langle ab|\hat{\eta}|kl\rangle, \\
\eta^{(2)}_{phph} &\widehat{=} \langle aj|\hat{\eta}|cl\rangle, \\
\eta^{(2)}_{ppph} &\widehat{=} \langle ab|\hat{\eta}|cl\rangle, \\
\eta^{(2)}_{pppp} &\widehat{=} \langle ab|\hat{\eta}|cd\rangle,
\end{aligned} \tag{8.10}$$

with the symmetry relations of Eq. (8.8) changed to

$$\begin{aligned}
&\eta^{(2)}_{hhhh}, \\
&\eta^{(2)}_{phhh} = -\eta^{(2)}_{hphh} = \eta^{(2)}_{hhhp} = -\eta^{(2)}_{hhph}, \\
&\eta^{(2)}_{pphh} = -\eta^{(2)}_{hhpp}, \\
&\eta^{(2)}_{phph} = -\eta^{(2)}_{hpph} = -\eta^{(2)}_{phhp} = \eta^{(2)}_{hphp}, \\
&\eta^{(2)}_{ppph} = -\eta^{(2)}_{pphp} = \eta^{(2)}_{hppp} = -\eta^{(2)}_{phpp}, \\
&\eta^{(2)}_{pppp}.
\end{aligned} \tag{8.11}$$

The class `Wegner` has two groups of member functions that are related to the generator $\hat{\eta}$: One group is used to get access to the $\hat{\eta}$-elements, where the analogy to the matrix elements of $\hat{H}$ enabled us to use nearly the same functions, only modified by some signs.
The other important function is responsible for computing and storing the $\hat{\eta}$-elements in the corresponding array. The calculations are based on Eqs. (6.29) and (6.30), but for efficiency reasons split according to which indices correspond to hole and which to particle states. That way, we try to keep the number of floating point operations to a minimum and avoid computing contributions that are known to be zero in advance. As an example, the first term for the two-body elements $\eta^{(2)}_{pqrs}$, the term $f_{ps}v^d_{sqsq}\delta_{qr}$ of Eq. (6.30), can only give a contribution if q and r are both particles or holes. Therefore it needs not to be included for terms of the form $\eta^{(2)}_{pphh}$ and $\eta^{(2)}_{phph}$, and similar argumentations hold for the other terms of Eq. (6.30).

Computing the derivatives The flow equations are in principle computed according to Eqs. (6.22)-(6.24), using the stored $\hat{\eta}$-elements in order to avoid redundant computations. To keep the number of floating point operations to a minimum, we proceed similar to updating $\hat{\eta}$ and try to circumvent evaluating those terms that are known to be zero in advance. As we will demonstrate, this involves several adaptations of our code, making it sometimes a bit harder to read. However, this increases effectiveness enormously, which is crucial for the code's overall performance.

To illustrate how we use our matrix elements with a tp-basis and how our classes work together, listing 8.14 shows how the second term of Eq. (6.22),

$$\frac{1}{2}\sum_{ijab}\eta^{(2)}_{ijab}v_{abij}, \tag{8.12}$$

Listing 8.14: Excerpt of the function `E0_deriv:`, which evaluates Eq. (6.22). Here the second term of Eq. (6.22) is demonstrated. For detailed explanations, see text.

```
1 for (int lbd = lbd_limits(0, 0); lbd <= lbd_limits(0, 1); lbd++)
2       for (int ij = 0; ij < Sys->H->Bas->hh_basis[lbd].size(); ij++)
3           for (int ab = 0; ab < Sys->H->Bas->pp_basis[lbd].size(); ab++)
4                 contr += -eta[6 + 6 * lbd](ab, ij) * mat_elems[6 + 6 * lbd](ab,
                      ij);
5
6     d_ret += 2 * contr;
```

is evaluated in our program. In principle, the term loops over four indices: hole states i, j and particle states a, b. The tp-basis enables us to reduce the four loops to two loops, which for N particles reduces the number of floating point operations from order $\mathcal{O}(N^4)$ to $\mathcal{O}(N^2)$.
Instead of summing over the two hole indices i, j, we loop over our `hh_basis` as demonstrated in line 2 of listing 8.14. Furthermore, instead of summing over particle states a, b, we simply take all elements of our `pp_basis` (see line 3). Since term (8.12) sums over all indices i, j, a, b, we need to consider all channels λ of the tp-basis, resulting in the additional loop of line 1. Because we loop over the `pp_basis` inside the loop of the `hh_basis`, we can hold the number of considered channels λ to a minimum by only taking into account the smaller range of the `hh_basis`.
The contributions from the $\hat{\eta}$- and v-elements are added as shown in line 4, where we have to select the right two-body elements of Eqs. (8.7) and (8.10). In the case of term (8.12), this means to pick elements $\eta^{(2)}_{pphh}$ and v_{pphh}. For the first channel λ, these elements have been stored at the sixth position of the arrays `eta` and `mat_elems`, respectively, afterwards the elements occupy every sixth position, due to six possible two-body elements. Hence, as listing 8.14 shows, the indices of the required elements are accessed using the index `6+6*lbd`.

The considerations about which of the possible elements in (8.7) and (8.10) have to be accessed, have not only to be made for term (8.12), but for each single term in the flow equations (6.22)-(6.24). This results often in splitting sums up, in order to distinguish between particle and hole states, and makes the code much more complex. However, as stated before, each loop over the tp-basis, instead of over two indices, reduces the number of floating point operations from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$, which is worth all the extra work on the code.

Another advantage of our tp-basis is that for each of the two possible two-particle states $|pq\rangle$ and $|qp\rangle$, we only store one of the configurations. As can be seen in listing 8.14, we nevertheless sum only once over all elements in the tp-basis. The reason is that, due to symmetry relations (8.8) and (8.11), only half of the indices has to be summed over, given that the final result is multiplied by two. Hence the tp-basis naturally halves the number of terms.

Improving efficiency Compared to one of the first versions of our code, where we implemented the flow equations absolutely straightforwardly, we have obtained a speedup of three orders of magnitude with our final code. This stresses how important optimization is to make the SRG method computationally affordable. Some of the means to improve efficiency have already been demonstrated, e.g. the use of the tp-basis, other ones concerning the class `Wegner` will be shown in the following.

Listing 8.15: Term (8.13) implemented straightforwardly, i.e. close to mathematical notation, and therefore easy to read.

```
for(int p_ind = 0; p_ind< part_states ; p_ind++){
        p = p_ind +hole_states;

        for(int q = 0; q < hole_states; q++){
            for(int r = 0; r < all_states; r++)
               dv[2](p_ind,q) += get_eta1(p,r)*H->get_f_elem(r,q,mat_elems) +
                   get_eta1(q,r)*H->get_f_elem(r,p,mat_elems);
         ...
         }
}
```

One especially useful tool is to express as many terms of the flow equations (6.22)-(6.24) as possible in terms of matrix-matrix multiplication. In many cases, this reduces the effective number of indices to be summed over and enables us to profit from the highly optimized matrix tools of LAPACK and BLAS, which we access via the Armadillo library.
As an example, consider the term

$$\sum_r \left(\eta^{(1)}_{pr} f_{rq} + \eta^{(1)}_{qr} f_{rp}\right) \tag{8.13}$$

of Eq. (6.23) for the matrix of form f_{ph}. In a first, straightforward approach, we expressed the term as shown in listing 8.15. The code segment is closely related to the mathematical equation and easy to read, but computationally ineffective since the elements are not accessed directly, but through 'getter-functions'.

A first step of optimization was therefore to reduce the number of calls to those 'getter-functions' to a minimum and access the elements as often as possible directly. We took this step of course not only in Eq. (6.23), but in all functions dealing with arrays. To know exactly how to access the arrays, this involved making out for every single index whether this one corresponds to a particle or a hole state. Often this required sums to be split up, as demonstrated for our example (8.13) in listing 8.16.
Apart from taking the right particle or hole indices, another demand is to pick the correct stored matrix by considering symmetry relations. For example in term (8.13), in the case that r corresponds to a particle, the element $\eta^{(1)}_{qr}$ needs to be transformed from the form $\eta^{(1)}_{hp}$ to $\eta^{(1)}_{hp} = -\eta^{(1)}_{ph}$. The usage is demonstrated in lines 13-14 of listing 8.16, where also the signs are changed correspondingly.
As mentioned before, some expressions, like our example term (8.13), even admit to be further optimized by using matrix-matrix multiplications. Considering basic matrix properties for the elements of f_{ph}, the contribution of term (8.13) can be rewritten as

$$\begin{aligned}\sum_r \left(\eta^{(1)}_{pr} f_{rq} + \eta^{(1)}_{qr} f_{rp}\right) \to \eta^{(1)}_{ph} \cdot f_{hh} + \left(\eta^{(1)}_{hh} \cdot f_{ph}\right)^T \quad &\text{(for hole states } r\text{)} \\ + \eta^{(1)}_{pp} \cdot f_{ph} - \left(\eta^{T}_{ph} \cdot f_{pp}\right)^T, \quad &\text{(for particle states } r\text{)}\end{aligned}$$

where '$\cdot$' denotes matrix-matrix multiplication and T the matrix transpose. The corresponding implementation can be found in listing 8.17. Matrices f_{hh} and f_{pp} are treated analogously.

Listing 8.16: Improvement of the implementation in listing 8.15 by accessing all elements directly, without getter functions.

```
for(int p_ind = 0; p_ind< part_states ; p_ind++){
        p = p_ind +hole_states;
        for(int q = 0; q < hole_states; q++){

            // r is hole state
            for(int r = 0; r < hole_states; r++)
                dv[2](p_ind,q) += eta[2](p_ind,r)*mat_elems[1](r,q)
                        + eta[1](q,r)*mat_elems[2](p_ind,r);

            // r is particle state
            for(int r = hole_states; r < all_states; r++)
                r_ind = r - hole_states; // r as index in matrices
                dv[2](p_ind,q) += eta[3](p_ind,r_ind)*mat_elems[2](r_ind,q)
                        - eta[2](r_ind,q)*mat_elems[3](r_ind,p_ind);
        ...
        }
}
```

Listing 8.17: The expression (8.13) can further be optimized by using matrix-matrix multiplication. The function `strans` belongs to the Armadillo library and returns simple matrix transposes, without taking the conjugate of the elements.

```
    // r is hole state
    dv[2] += eta[2] * mat_elems[1] + mat_elems[2] * strans(eta[1]);
    // r is particle state
    dv[2] += eta[3] * mat_elems[2] - strans(mat_elems[3]) * eta[2];
```

Listing 8.18: Effective way for adding the two contributions of Eq. (8.15). The variables `v_comb1` and `v_comb2` are used to access the right matrix elements, and the variable `v_ind` to use the right matrix of the array `mat_elems`. Since different symmetry relations of $\hat{H}$ and $\hat{\eta}$ sometimes result in different signs, the variable `v_sign` conveys the correct sign. The same explanations hold for the variables of the other function, which all start with the prefix `eta_`.

```
for (int i = 0; i < hole_states; i++)
    for (int a = hole_states; a < all_states; a++) {

        dv[dv_ind](ai, kl) -= get_eta2_comb(a, q, i, s, v_comb1, v_comb2, v_ind
            , v_sign) * get_v_elem_comb(i, p, a, r, mat_elems, eta_comb1,
            eta_comb2, eta_ind, eta_sign);

        if (abs(v_sign) > 0 && abs(eta_sign) > 0)
                dv[dv_ind](ai, kl) += eta_sign * eta[eta_ind](eta_comb1,
                    eta_comb2) * v_sign * mat_elems[v_ind](v_comb1, v_comb2);

        }
```

Apart from this, our code involves many smaller optimizations for special expressions. One further example are cases where using the 'getter-functions' is unavoidable but the two-body elements $\eta^{(2)}_{pqrs}$ and v_{pqrs} are needed with the same order of indices $pqrs$. Due to the specific form of the flow equations, $\frac{d\hat{H}_s}{ds} = \left[\hat{\eta}_s, \hat{H}_s\right]$, such terms occur rather frequently. Since obtaining an element with specific indices involves first finding the correct channel λ, then the actual index in the array, and we store $\hat{\eta}$ and $\hat{H}$ in arrays of the same structure, it makes sense to avoid finding the index twice and reuse it from the first computation.

Therefore, we designed the two functions

```
    double get_eta2_comb(int p, int q, int r, int s, int& comb1, int& comb2,
        int& v_ind, int& v_sign);

    double get_v_elem_comb(int p, int q, int r, int s, std::vector<arma::mat>&
        mat_elems, int& comb1, int& comb2, int& eta_ind, int& eta_sign);
```

which return, similar to ordinary getter functions, $\eta^{(2)}_{pqrs}$ and v_{pqrs}, respectively, but save additionally all information needed to access the other element with same indices. For example, consider the last line in Eq. (6.24),

$$-\sum_{ia}\left(1-\hat{P}_{ia}\right)\left(1-\hat{P}_{pq}\right)\left(1-\hat{P}_{rs}\right)\eta^{(2)}_{aqis}v_{ipar}, \tag{8.14}$$

which when multiplied out contains the contribution

$$-\eta^{(2)}_{aqis}v_{ipar} + v^{(2)}_{aqis}\eta_{ipar}. \tag{8.15}$$

Instead of searching for the index corresponding to $aqis$ and $ipar$ twice, we do it only for the first term by using the two introduced functions, as demonstrated in listing 8.18.

Class `White`

The class `White` is our second subclass of `SRG` and used for White's generator (6.31). Since it is derived from the same base class as the class `Wegner` and overrides the same virtual functions, it has exactly the same structure. This is one of the advantages of inheritance in C++ and contributes to a clear structure of our code.
The overriding functions for computing the generator $\hat{\eta}$ and the flow equations (6.22)-(6.24) have been adopted correspondingly. In particular, White's generator allows great simplifications with respect to the complexity of the functions, as well as the number of stored matrix elements. In the following, we show how we adapted the data structures and functions explained for Wegner's generator to White's generator. We proceed in the same order, starting with the generator itself and then continuing with the derivatives, at the same time presenting issues concerning efficiency of the code.

Updating $\hat{\eta}$ Similar to the class `Wegner`, the one- and two-body elements of the generator $\hat{\eta}$ are stored in an array and used when evaluating the flow equations. However, as explained in subsection 6.4.2, the only non-zero elements are of the form η_{ph} and $\eta_{p_1p_2h_1h_2}$, which means that storage is drastically reduced. In particular for the two-body elements, only one instead of six matrices per channel is needed.
The computational advantage does not only involve reduced storage, but also the number of $\hat{\eta}$-elements to be updated each integration step is considerably reduced.
Listing 8.19 shows how we implemented Eqs. (6.33) and (6.35) for updating the $\hat{\eta}$-elements. As explained for Wegner's generator, efficiency can considerably be increased by accessing as many elements as possible directly, without the use of getter functions. This is for example applied in the code segment shown in listing 8.19, although the code gets harder to read than the excerpt in listing 8.20, which shows our first, straightforward implementation.
In spite of the great advantages of direct matrix access, the final version of our code still relies on the use of getter functions, since it is not always possible to know an element's exact place in the array in advance. For example, consider lines 33-38 in listing 8.19:
Since we are interested in all elements $\eta^{(2)}_{pqrs}$ of the form $\eta^{(2)}_{pphh}$, we loop in our `pp_basis` over states $|pq\rangle$ and in the `hh_basis` over states $|rs\rangle$. The indices of states $|pq\rangle$ and $|rs\rangle$ in the tp-basis are therefore well known, which makes it straightforward to access v-elements v_{pqpq} and v_{rsrs} directly in the storing array (see lines 33-34). However, also terms $v_{prpr}, v_{qrqr}, v_{psps}$ and v_{qsqs} are needed and we do not now the indices of tp-states $|pr\rangle, |qr\rangle, |ps\rangle$ and $|qs\rangle$ in advance. Not even the channel λ is necessarily the same as for $|pq\rangle$ and $|rs\rangle$. In these cases, we are dependent on our getter functions, which supply the required elements by extracting both the channel and the indices in the tp-basis.

Computing the derivatives The derivatives (6.22)-(6.24) are independent of exact expressions of the $\hat{\eta}$-elements, which means that we principally could use the same implementation as for Wegner's generator. However, in this case we would not profit of the simplifications White's generator involves:
Since the only non-zero elements of the generator $\hat{\eta}$ must have the form $\eta^{(1)}_{ph}$ or $\eta^{(2)}_{pphh}$, respectively, all terms in the flow equations involving $\hat{\eta}$-elements not in this form can be removed. This reduces the complexity of the flow equations, and hence the number of floating point

Listing 8.19: Updating all elements of the generator $\hat{\eta}$. The matrix elements of the Hamiltonian are received in the array `mat_elems`. The $\hat{\eta}$-elements stored in the class variable `eta`. As for Wegner's generator, we have the possibility of including the loop terms of three-body terms.

```
void White::update_eta(std::vector<arma::mat>& mat_elems) {

    ...
    ////////////////////////////////////////////////////////////////////
    //                        Update eta_ph

    for (int p_ind = 0; p_ind < part_states; p_ind++) {
        p = p_ind + hole_states;
        for (int q = 0; q < hole_states; q++) {
            eta[0](p_ind, q) = mat_elems[2](p_ind,q) /
                    (mat_elems[3](p_ind,p_ind) - mat_elems[1](q,q)
                    - Sys->H->get_v_elem(p, q, p, q, mat_elems));
        }
    }

    ////////////////////////////////////////////////////////////////////
    //                    Update eta_pphh

    for (int lbd = lbd_limits(0, 0); lbd <= lbd_limits(0, 1); lbd++)

        for (int ab = 0; ab < Sys->H->Bas->pp_basis[lbd].size(); ab++) {
            p = Sys->H->Bas->pp_basis[lbd][ab](0);
            q = Sys->H->Bas->pp_basis[lbd][ab](1);

            for (int kl = 0; kl < Sys->H->Bas->hh_basis[lbd].size(); kl++) {
                r = Sys->H->Bas->hh_basis[lbd][kl](0);
                s = Sys->H->Bas->hh_basis[lbd][kl](1);

                A = mat_elems[9+6*lbd](ab,ab)   // v_pqpq
                        + mat_elems[4+6*lbd](kl,kl) // v_rsrs
                        - Sys->H->get_v_elem(p, r, p, r, mat_elems) // v_prpr
                        - Sys->H->get_v_elem(q, r, q, r, mat_elems) // v_qrqr
                        - Sys->H->get_v_elem(p, s, p, s, mat_elems) // v_psps
                        - Sys->H->get_v_elem(q, s, q, s, mat_elems); // v_qsqs

                eta[1 + lbd](ab, kl) += mat_elems[6+6*lbd](ab,kl) / // v_pqrs
                        ( mat_elems[3](p-hole_states,p-hole_states) // f_pp
                        + mat_elems[3](q-hole_states,q-hole_states) // f_qq
                        - mat_elems[1](r,r) // f_rr
                        - mat_elems[1](s,s) + A); // f_ss

            }
        }
}
```

Listing 8.20: Straightforward implementation of the update of the one-body elements of $\hat{\eta}$, close to mathematical notation.

```
//                          Update eta_ph

    for (int p_ind = 0; p_ind < part_states; p_ind++) {
        p = p_ind + hole_states;
        for (int q = 0; q < hole_states; q++) {
            eta[0](p_ind, q) = Sys->H->get_f_elem(p, q, mat_elems) /
                    (Sys->H->get_f_elem(p, p, mat_elems)
                    - Sys->H->get_f_elem(q, q, mat_elems)
                    - Sys->H->get_v_elem(p, q, p, q, mat_elems));
        }
    }
```

operations per integration step, considerably.
For example, consider the derivative of v-elements v_{klmn} of form v_{hhhh}, i.e. all four indices correspond to hole states. Explicitly writing out the permutation operators in Eq. (6.24), the following expression has to be evaluated:

$$\begin{aligned}\frac{dv_{klmn}}{ds} = &\sum_t \left(\eta^{(1)}_{kt} v_{tlmn} - \eta^{(1)}_{lt} v_{tkmn} - f_{kt}\eta^{(2)}_{tlmn} + f_{lt}\eta^{(2)}_{tkmn}\right) \\ &- \sum_t \left(\eta^{(1)}_{tm} v_{kltn} - \eta^{(1)}_{tn} v_{kltm} - f_{tm}\eta^{(2)}_{kltn} + f_{tn}\eta^{(2)}_{kltm}\right) \\ &+ \frac{1}{2}\sum_{ab} \left(\eta^{(2)}_{klab} v_{abmn} - v_{klab}\eta^{(2)}_{abmn}\right) - \frac{1}{2}\sum_{ij} \left(\eta^{(2)}_{klij} v_{ijmn} - v_{klij}\eta^{(2)}_{ijmn}\right) \\ &- \sum_{ia} \Big(\eta^{(2)}_{alin} v_{ikam} - \eta^{(2)}_{akin} v_{ilam} - \eta^{(2)}_{alim} v_{ikan} + \eta^{(2)}_{akim} v_{ilan} \\ &- \eta^{(2)}_{ilan} v_{akim} + \eta^{(2)}_{ikan} v_{alim} + \eta^{(2)}_{ilam} v_{akin} - \eta^{(2)}_{ikam} v_{alin}\Big) . \end{aligned} \tag{8.16}$$

Eliminating all terms containing $\hat{\eta}$-elements of form $\eta^{(1)}_{hh}, \eta^{(1)}_{pp}, \eta^{(2)}_{hhhh}, \eta^{(2)}_{phhh}, \eta^{(2)}_{phph}, \eta^{(2)}_{ppppp}, \eta^{(2)}_{pppp}$, Eq. (8.16) gets simplified to

$$\begin{aligned}\frac{dv_{klmn}}{ds} = &\sum_a \left(\eta^{(1)}_{ka} v_{almn} - \eta^{(1)}_{la} v_{akmn}\right) - \sum_a \left(\eta^{(1)}_{am} v_{klan} - \eta^{(1)}_{an} v_{klam}\right) \\ &+ \frac{1}{2}\sum_{ab} \left(\eta^{(2)}_{klab} v_{abmn} - v_{klab}\eta^{(2)}_{abmn}\right) . \end{aligned} \tag{8.17}$$

Obviously, this involves considerably fewer floating point operations. One should also notice that the first two loops now only sum over particle states a, instead of over all single-particle states t. These simplifications demonstrate how computationally advantageous it is to differentiate the matrix elements according to whether their indices correspond to particle or hole states and to treat each of the possible particle-hole combinations separately, instead of implementing the flow equations (6.22)-(6.22) completely generally without this differentiation. Apart from those simplifications, the code for White's generator is structured analogously to Wegner's one and we took the same means to improve efficiency, as for example using matrix-matrix multiplications whenever possible.

8.6 Implementation of Hartree-Fock

Our code contains a class, called `HartreeFock`, which allows the user to prepend a Hartree-Fock (HF) to the SRG calculation. The purpose is to transform an ordinary harmonic oscillator to a Hartree-Fock basis, which can be used as input for the following computations. To gain as much efficiency as possible, we structured the code to be in line with the data structures of IM-SRG, such that a direct use of the transformed elements is possible. However, the class can without any problems be used for calculations preceding free-space SRG, too.
In practice, the class receives the untransformed one- and two-body elements $\langle\alpha|\hat{h}^{(0)}|\gamma\rangle$ and $\langle\alpha\beta||\gamma\delta\rangle$ of the Hamiltonian and transforms them using a coefficient matrix. In our case, the one-body elements are given by the single-particle energies and the tp-elements are obtained using the OpenFCI library [17].
The transformation occurs in two steps: First, we perform a HF calculation based on the convergence of the HF energy. This yields a coefficient matrix C, which afterwards can be used to transform the basis elements.

In the following, we will explain both steps in detail and present the most important aspects of our implementation.

8.6.1 Hartree-Fock calculation

Our Hartree-Fock calculation is based on the explanations of section 7.1, in particular on Eqs. (7.4), (7.6) and (7.7). The main algorithm is implemented in one function, which takes as input an untransformed single-particle basis. It returns the HF energy, as well as the C-matrix which can be used for a subsequent basis transformation. As explained in section 7.1, the HF orbitals are linear combinations of single-particle functions,

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

where $|\alpha\rangle$ is the set of orthonormal single-particle functions that spans the chosen model space. In our case, that one is given by a harmonic oscillator basis. The idea is to vary the coefficients in this expansion, such that the HF energy given in Eq. (7.3) is minimized. For this purpose, the HF equations (7.7) are solved iteratively. The complete algorithm is summarized in figure 8.4. In the following, we will look at the details of the algorithm and demonstrate how we implemented the different stages in our class `HartreeFock`.

The first step is to initialize the coefficient matrix C. For a pure HF calculation, where one is only interested in the HF energy, the dimension of this matrix is required to be only $N \times n_{sp}$, where N and n_{sp} denote the number of particles and single-particle states, respectively. The orbital of electron k is then associated with the kth column of the matrix and this vector,

$$C_k = \begin{pmatrix} C_{1k} \\ C_{2k} \\ \dots \\ C_{nk} \end{pmatrix},$$

is used for the HF equation (7.7). However, since for a basis transformation all eigenvectors of the HF-matrix are needed, our C-matrix has dimension $n_{sp} \times n_{sp}$.

Hartree-Fock algorithm

Input: Untransformed basis elements $\langle\alpha|\hat{h}^{(0)}|\gamma\rangle$ and $\langle\alpha\beta||\gamma\delta\rangle$.

1. Initialize the coefficient matrix to $C = \hat{1}$.
2. While not converged:
   (a) Use C to compute $\hat{h}^{HF}$ according to Eq. (7.6).
   (b) Diagonalize $\hat{h}^{HF}$ to obtain its eigenvectors.
   (c) Let those eigenvectors define the coefficient vectors for the next guess of C.
   (d) Compute the new HF energy $E\left[\Phi_0^{HF}\right]$ according to Eq. (7.4).
   (e) Determine the difference between $E\left[\Phi_0^{HF}\right]$ of current and previous iteration.

Output: HF energy $E\left[\Phi_0^{HF}\right]$ and coefficient matrix C.

Figure 8.4: Summary of the Hartree-Fock algorithm.

At the beginning of the calculation, the coefficient matrix is initialized to $C = \hat{1}$, which is equivalent to starting with an untransformed basis.
For each HF iteration, we then proceed as follows: First, we use the C-matrix to compute the HF-matrix $\hat{h}^{HF}$ according to Eq. (7.6). The challenge hereby is to store $\hat{h}^{HF}$ in a way that minimizes required space in memory and allows an efficient diagonalization. Inspired by the procedure of M.H. Jørgensen and C. Hirth [24, 67], we store $\hat{h}^{HF}$ in block-form. Similar to storing the Hamiltonian in our SRG code, we use that quantum numbers M and M_s are conserved, suggesting that $\hat{h}^{HF}$ is block-diagonal in M and M_s. Therefore we store the HF-matrix as an array of those blocks, which allows to perform blockwise diagonalization later on. For setting up the blocks, we proceeded similar as M.H. Jørgensen [67], although we adjusted the implementation to our code structure and optimized it. We have adopted the computational trick presented there, namely introducing another matrix, called `rho`, to speed up convergence. For details, we refer to [67].
Having computed all blocks of $\hat{h}^{HF}$ using the C-matrix, our next step is to diagonalize those blocks. In our code, we employ the function `eig_sym` of the Armadillo library, which refers to LAPACK functions for diagonalization.
As summarized in the algorithm in figure 8.4, we let those eigenvectors define the coefficient vectors for the next guess of C. The next step is to compute the new HF energy $E\left[\Phi_0^{HF}\right]$. Since the non-interacting part $\hat{H}_0$ of our Hamiltonian is diagonal, Eq. (7.4) can be simplified to

$$E\left[\Phi_0^{HF}\right] = \sum_i \sum_\alpha C_{i\alpha}^2 \epsilon_\alpha + \frac{1}{2} \sum_{ij} \sum_{\alpha\beta\gamma\delta} C_{i\alpha}^* C_{j\beta}^* C_{i\gamma} C_{j\delta} \langle\alpha\beta||\gamma\delta\rangle, \tag{8.18}$$

where $\epsilon_\alpha = \langle\alpha|\hat{h}_0|\alpha\rangle$ are the single-particle energies. Each iteration, the new HF energy is compared with the one of the previous iteration and if the energy has not changed within a certain tolerance interval, it is said to have converged and we end the iterations. The final result of all iterations are the HF energy $E\left[\Phi_0^{HF}\right]$ and the coefficient matrix C.

Improving efficiency To make the Hartree-Fock implementation as effective a possible, we took two major steps:
First, we have rewritten as many sums as possible into matrix operations, similar to what we

explained for our SRG code. Inspired by the code of C. Hirth [24], we define a new matrix

$$C^i_{pq} = \sum_k C_{kp} C_{kq}, \tag{8.19}$$

which can be expressed by the following matrix-matrix multiplication:

$$C^i = C^T_{h\times all} \cdot C_{h\times all}.$$

Here '$\cdot$' denotes matrix multiplication and T the matrix transpose. The matrix $C_{h\times all}$ has dimension $N \times n_{sp}$, where n_{sp} denotes the number of single-particle states and N the number of particles. It contains the first N columns of the C-matrix. With this matrix C^i, Eq. (7.4) can be simplified to

$$E\left[\Phi_0^{HF}\right] = \sum_{\alpha\beta} C^i_{\alpha\beta} \langle\alpha|\hat{h}_0|\beta\rangle + \frac{1}{2}\sum_{\alpha\beta\gamma\delta} C^i_{\alpha\gamma} C^i_{\beta\delta} \langle\alpha\beta||\gamma\delta\rangle. \tag{8.20}$$

That way, the first term is reduced from $\mathcal{O}(N n_{sp}^2)$ to $\mathcal{O}(n_{sp}^2)$ and the second one from $\mathcal{O}(N^2 n_{sp}^4)$ to $\mathcal{O}(n_{sp}^4)$. This saves an enormous number of floating-point operations, especially for large numbers of particles and considered shells R.

Additionally, we found another feature which allows considerable simplifications: Both matrices C and C^i are extremely sparse, i.e. most of their elements are zero, as illustrated in figure 8.5. When updating the HF energy according to Eq. (8.20), this means that most of the contributions are zero.
We now tried to find a way to make use of this sparsity and avoid adding contributions that are known to be zero in advance. For this reason, we have introduced a new array, which saves for each index i those indices j that correspond to non-zero matrix elements C_{ij}. Instead of summing over all indices $\alpha, \beta, \gamma, \delta$ in the second term of Eq. (8.20), this allows to sum over only those combinations α, γ and β, δ where the corresponding matrix elements $C^i_{\alpha\gamma}$ and $C^i_{\beta\delta}$ are different from zero. The respective implementation is given in listing 8.21.

8.6.2 Transformation of basis

The coefficient matrix C from the Hartree-Fock calculation can be used to transform the one- and two-body elements according to

$$\langle p|\hat{h}^{(0)}|q\rangle \leftarrow \sum_{\alpha\beta} C^*_{p\alpha} C_{q\beta} \langle\alpha|\hat{h}^{(0)}|\beta\rangle \tag{8.21}$$

$$\langle pq||rs\rangle \leftarrow \sum_{\alpha\beta\gamma\delta} C^*_{p\alpha} C^*_{q\beta} C_{r\gamma} C_{s\delta} \langle\alpha\beta||\gamma\delta\rangle. \tag{8.22}$$

Improving efficiency Equations (8.21) and (8.22) are computationally very expensive and should be optimized to keep the overall code efficient. Regarding Eq. (8.21), one complete summation loop can be omitted by taking into account that the non-interacting part of our Hamiltonian, $\hat{H}_0$, is diagonal. The result is the simplified equation

$$\langle p|\hat{h}^{(0)}|q\rangle \leftarrow \sum_{\alpha} C^2_{p\alpha} \epsilon_\alpha. \tag{8.23}$$

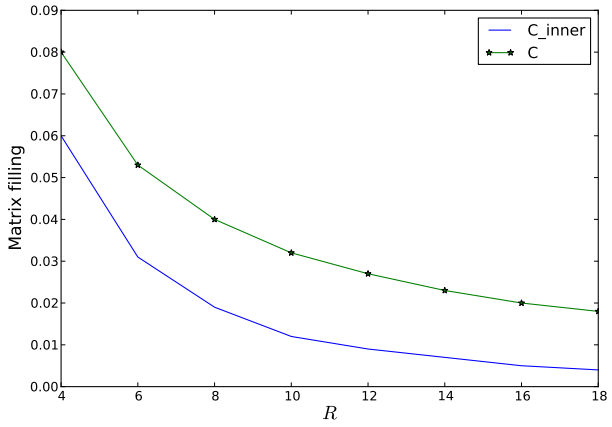

Figure 8.5: Sparsity of the coefficient matrices for systems with $N = 2$ particles, where R denotes the number of shells. The filling of the matrices is given by the percentage of non-zero elements.

Listing 8.21: Calculation of the two-body contribution to the Hartree-Fock energy as given in Eq. (8.20). To minimize the number of terms, the array `cCombs` contains those combinations of indices p and q where the matrix elements C^i_{pq} are different from zero. Parallelization is performed with OpenMP, where the clause `reduction` ensures that all contributions are properly summed up.

```
#pragma omp parallel for private(alpha, gamma, beta, delta) reduction(+:hf_ret)
    for (alpha = 0; alpha < nsp; alpha++)
        for (int c = 0; c < cCombs[alpha].size(); c++) {

            gamma = cCombs[alpha][c];
            for (beta = 0; beta < nsp; beta++)

                for (int d = 0; d < cCombs[beta].size(); d++) {
                    delta = cCombs[beta][d];
                    hf_ret +=  C_inner(alpha, gamma) * C_inner(beta, delta)
                            * Sys->H->get_v_elem(alpha, beta, gamma, delta, Sys
                                ->H->mat_elems);
                }
        }

    hf_ret *= 0.5;
```

Listing 8.22: Instead of summing over all arrangements of indices as in Eq. (8.22), we make use of our two-particle basis and transform the two-body elements block-wise and according to particle and hole indices. In each of those functions, Eq. (8.22) is implemented similar to the example of v_{hhhh}, which is given in listing 8.23.

```
1 // Transformation of two-body elements
2     transform_vhhhh();
3     transform_vphhh();
4     transform_vpphh();
5     transform_vphph();
6     transform_vppph();
7     transform_vpppp();
```

Here $\epsilon_\alpha = \langle\alpha|\hat{h}^{(0)}|\alpha\rangle$ denotes the energy of single-particle state α. However, most of the time is not required by transforming the one-body elements, but by treating the two-body elements as in Eq. (8.22). When all elements $\langle pq||rs\rangle$ are to be computed, this equation demands theoretically looping over eight indices. This corresponds to a number of floating-point operations of order $\mathcal{O}(n_{sp}^8)$, which should definitely be avoided.
As for the SRG method, we make therefore use of the block-form of the Hamiltonian, i.e. its diagonality in M and M_s. As explained before, this enables us to employ a two-particle basis, which reduces the number of relevant elements to combinations (8.7). Instead of looping over all possible arrangements, we therefore consider only those ones corresponding to non-zero elements. This means that we do not loop over single indices but over the particle-hole combinations of our two-particle basis. The procedure is analogous to the one of our class `Hamiltonian` in IM-SRG, and we therefore refer to section 8.5 for further details. The function responsible for the basis transformation then calls six functions, one for each of the possible combinations (8.7), see listing 8.22.

Our second optimization, which can be found in listing 8.23, is that analogous to the update of the HF energy in Eq. (8.20), most of the contributions are zero due to the great sparsity of the C-matrix. For not looping unnecessarily over most of the combinations, we again make use of an array that saves those indices i and j corresponding to non-zero matrix elements C_{ij}. The implementation, here given for elements of form v_{hhhh}, i.e. where all four indices correspond to hole states, can be found in listing 8.23. The code excerpt is structured as follows:

- Line 1: Pragma for parallelization with OpenMP.
- Line 2: We loop over all channels.
- Lines 4-11: We loop two times over all combinations in our `hh_basis`, one time for the bra- and one time for the ket-side of elements $\langle pq||rs\rangle$.
- Lines 13-29: We perform the inner loop for the two-body elements, where we make use of the sparsity of the C-matrix. As explained above, this is achieved by saving non-zero index combinations in an array, here called `cCombs`.

Listing 8.23: Transformation of two-body elements of the form v_{hhhh}, i.e. where all four indices correspond to hole states. The array `tr_elems` contains the transformed elements and is structured analogously to the array holding the matrix elements of the Hamiltonian. Array `cCombs` contains those combinations i, j that correspond to non-zero elements C_{ij} of the coefficient matrix. This allows us to make use of the sparsity of the C-matrix and perform the loops most effectively. All functions performing the basis transformation are parallelized with OpenMP.

```
#pragma omp parallel for private(tp_HF, a,b,c,d,alpha,beta,gamma,delta,tmp)
    schedule(dynamic)
    for (int lbd = lbd_limits(0, 0); lbd <= lbd_limits(0, 1); lbd++) {
        tr_elems[4 + 6 * lbd].zeros();
        for (int ab = 0; ab < Sys->H->Bas->hh_basis[lbd].size(); ab++) {
            a = Sys->H->Bas->hh_basis[lbd][ab](1);
            b = Sys->H->Bas->hh_basis[lbd][ab](0);

            for (int cd = 0; cd < Sys->H->Bas->hh_basis[lbd].size(); cd++) {
                c = Sys->H->Bas->hh_basis[lbd][cd](1);
                d = Sys->H->Bas->hh_basis[lbd][cd](0);
                tp_HF = 0.0;

                // Inner loop
                for (int p = 0; p < cCombs[a].size(); p++) {
                    alpha = cCombs[a][p];
                    for (int q = 0; q < cCombs[b].size(); q++) {
                        beta = cCombs[b][q];

                        for (int r = 0; r < cCombs[c].size(); r++) {
                            gamma = cCombs[c][r];
                            for (int s = 0; s < cCombs[d].size(); s++) {
                                delta = cCombs[d][s];
                                tp_HF += C(alpha, a) * C(beta, b) * C(gamma, c)
                                        * C(delta, d) * Sys->H->get_v_elem(
                                            alpha,
                                        beta, gamma, delta, Sys->H->mat_elems);
                            }
                        }
                    }
                }
                tr_elems[4 + 6 * lbd](ab, cd) = tp_HF;
            }
        }
    }
```

8.7 Implementation of Diffusion Monte Carlo

We developed the Diffusion Monte Carlo code independently of our SRG code, as part of the project in [68]. The complete code is written object-oriented in C++ and kept as general as possible. That way, it is on the one hand easy to switch between different methods, e.g. brute-force or importance sampling. On the other hand, the code can easily be extended, for example with another potential than the harmonic oscillator one.

8.7.1 The Variational Monte Carlo part

Prior to each DMC calculation, the VMC algorithm has to be run in order to determine the optimal parameters α, β for the trial wave function. Since the VMC part serves as input for the subsequent DMC calculation and the main procedure, calculating the average of the local energy, is the same, we have designed `DMC` and `VMC` as two classes of the same code, and they use the same classes for Hamiltonian, wave function etc. For that reason, the explanations for the latter classes hold for the DMC part, too, and will not be repeated there.

VMC algorithm

The VMC algorithm is implemented in the class `VMC`. This one has to subclasses, which makes it easy to switch between the two methods: `Metropolis` and `Metropolis_Hastings`.

The main algorithm is the same for both methods and summarized in figure 8.6. The central difference lies only in the determination of the trial position $\mathbf{R}'$ and the acceptance ratio R. Table 8.2 compares those two functions of our code for both subclasses. Note that for the DMC calculations later on, we will only employ the more effective Metropolis-Hastings sampling. However, our program includes both algorithms, which is also a practical feature for debugging the code.

Hamiltonian and local energy

All computations related to the Hamiltonian of the system are implemented in a separate class, `Hamiltonian`, which to make it easily adjustable, is composed of three components:
The class `Kinetic` is responsible for computing the kinetic part of the local energy, the classes

1. Initialize the system
2. Loop over the MC cycles and particles.
   (a) Find new trial position $\mathbf{R}'$.
   (b) Calculate acceptance ratio R.
   (c) If $R < Y$, $Y \in [0, 1] \rightarrow$ Accept the move. Else reject.
   (d) Update the expectation values.

Figure 8.6: The main algorithm of Variational Monte Carlo calculations.

	Metropolis	Metropolis_Hastings
Trial position	$\chi \in [-1,1]$	$\chi \in gaussian()$
		$\mathbf{F}_i(\mathbf{R}) = 2\frac{1}{\Psi_T}\nabla_i \Psi_T$
	$\mathbf{R}' = \mathbf{R} + \chi\delta$	$\mathbf{R}' = \mathbf{R} + D\Delta t\mathbf{F}(\mathbf{R}) + \chi$
Acc. ratio		$G(\mathbf{R},\mathbf{R}';\Delta t) = e^{-\frac{(\mathbf{R}'-\mathbf{R}-D\Delta t\mathbf{F}(\mathbf{R}))^2}{4D\Delta t}}$
	$\mathrm{R} = \left\lvert\frac{\psi_i}{\psi_j}\right\rvert^2$	$R = \frac{G(\mathbf{R}',\mathbf{R};\Delta t)}{G(\mathbf{R},\mathbf{R}';\Delta t)}\left\lvert\frac{\psi_i}{\psi_j}\right\rvert^2$

Table 8.2: The technical differences between the Metropolis and Metropolis-Hastings algorithm lie only in the determination of the trial position and computation of the acceptance ratio.

`Potential` and `Interaction` for each its part of the potential energy.
To make it easy to switch between an interacting and a non-interacting system, we compute the unperturbed and interaction part of the Hamiltonian separately. If the interaction is switched on, we use both contributions to compute the local energy, otherwise we simply leaves out the one from the interaction part.
In the class `Kinetic`, one can choose whether the Laplacian $\left(-\frac{1}{2}\nabla^2\right)$ for the kinetic energy shall be computed analytically or numerically.
The numerical version uses a simple form of the discretized second derivative

$$f''(x) = \frac{f(x+h) + f(x-h) - 2f(x)}{h^2},$$

leading to

$$\frac{1}{\Psi_T}\nabla_i^2\Psi_T = \frac{\Psi_T(\mathbf{r}_i + h) + \Psi_T(\mathbf{r}_i - h) - 2\Psi_T(\mathbf{r}_i)}{h^2\Psi_T(\mathbf{r}_i)},$$

where $\mathbf{r}_i$ is the position vector of the ith particle.

The analytical expression involves some more mathematics. First of all,

$$\begin{aligned}\frac{\nabla_i^2\Psi_T}{\Psi_T} &= \frac{\nabla_i^2\left(|D_\uparrow|\,|D_\downarrow|\,J\right)}{|D_\uparrow|\,|D_\downarrow|\,J} \\ &= \frac{\nabla_i^2|D_\uparrow|}{|D_\uparrow|} + \frac{\nabla_i^2|D_\downarrow|}{|D_\downarrow|} + \frac{\nabla_i^2 J}{J} + 2\frac{(\nabla_i|D_\uparrow|)(\nabla_i|D_\downarrow|)}{|D_\uparrow|\,|D_\downarrow|} \\ &\quad + 2\frac{(\nabla_i|D_\uparrow|)(\nabla_i J)}{|D_\uparrow|\,J} + 2\frac{(\nabla_i|D_\downarrow|)(\nabla_i J)}{|D_\downarrow|\,J},\end{aligned}$$

where as introduced in section 7.2, $|D_\uparrow|$ and $|D_\downarrow|$ denote the spin up and spin down part of the Slater determinant, respectively, and J is the Jastrow function, Since only one of two determinants contains particle i, the gradient of the other determinant vanishes and the above equation reduces to

$$\frac{\nabla_i^2\Psi_T}{\Psi_T} = \frac{\nabla_i^2|D|}{|D|} + \frac{\nabla_i J}{J} + 2\frac{\nabla_i^2|D|}{|D|}\frac{\nabla_i J}{J},$$

where $|D|$ is the determinant with particle i. The needed ratios are given in Ref. [57]: For the Slater determinant, we have

$$\frac{\nabla_i|D|}{|D|} = \sum_{j=1}^{N}\left(\nabla_i\phi_j(\mathbf{r}_i)\right)D_{ji}^{-1}, \tag{8.24}$$

where the gradient of the single particle orbital $\phi_j(\mathbf{r}_i)$ is

$$\nabla\phi_{n_x,n_y} = \left\{ \left(2n_x\sqrt{\omega\alpha}\frac{H_{n_x-1}}{H_{n_x}} - x\omega\alpha \right)\vec{e}_x + \left(2n_y\sqrt{\omega\alpha}\frac{H_{n_y-1}}{H_{n_y}} - y\omega\alpha \right)\vec{e}_y \right\}\phi_{n_x,n_y}.$$

The gradient of the Jastrow factor is straightforward,

$$\frac{1}{J}\frac{\partial J}{\partial x_i} = \sum_{k=1,k\neq i}^{N}\frac{a_{ki}(x_i - x_k)}{r_{ki}(1+\beta r_{ki})^2}. \tag{8.25}$$

Note that we have written a_{ki} as matrix element. Since calculating the matrix a for every run in the loop would request unnecessary CPU time, we specify the elements only in the beginning and store them in a matrix once and for all.
The Laplacians are as follows:

$$\frac{\nabla_i^2|D|}{|D|} = \sum_{j=1}^{N}\left(\nabla_i^2\phi_j(\mathbf{r}_i)\right)D_{ji}^{-1}, \tag{8.26}$$

where the Laplacian of the single-particle orbitals is

$$\begin{aligned}\nabla^2\phi_{n_x,n_y} =& \omega\alpha\phi_{n_x,n_y}\left(4n_x(n_x-1)\frac{H_{n_x-2}}{H_{n_x}}\right.\\ &+4n_y(n_y-1)\frac{H_{n_y-2}}{H_{n_y}} + \omega\alpha\left(x^2+y^2\right)\\ &\left.-4\sqrt{\omega\alpha}n_x x\frac{H_{n_x-1}}{H_{n_x}} - 4\sqrt{\omega\alpha}n_y y\frac{H_{n_y-1}}{H_{n_y}} - 2\right).\end{aligned}$$

For the Jastrow factor, we have

$$\frac{\nabla_i^2 J}{J} = \left(\frac{\nabla_i J}{J}\right)^2 + \sum_{k=1,k\neq i}^{N}\frac{a_{ki}(1-\beta r_{ik})}{r_{ki}(1+\beta r_{ki})^2}, \tag{8.27}$$

with the gradient given by Eq. (8.25).

Implementation of the wave function

As introduced in section 7.2, our wave function has the structure

$$\Psi_T(\alpha,\beta) = |D_\uparrow(\alpha)|\,|D_\downarrow(\alpha)|\,J(\beta). \tag{8.28}$$

In order to easily switch between wave functions with and without correlation, it is favourably that each instance of the class `Wavefunction` is composed of two objects: A Slater determinant and a Jastrow factor. If interested in how good the Slater determinant without correlation factor approximates the true wave function, we can simply leave out the Jastrow object. On the other hand, this structure opens up the possibility to create several subclasses with different forms of the correlation factor that can be studied.

j	n_x	n_y	ϵ
1	0	0	ω
2	1	0	2ω
3	0	1	2ω
4	2	0	3ω
5	1	1	3ω
6	0	2	3ω

Table 8.3: Assignment of single-particle levels.

j	2 particles		6 particles		12 particles	
1	1	2	1	4	1	7
2			2	5	2	8
3			3	6	3	9
4					4	10
5					5	11
6					6	12

Table 8.4: Assignment of the particles to the single-particle levels for different values of N.

Running the VMC algorithm, we are actually not interested in the values of the wave function itself, but only in ratios between new and old wave function. With only one particle i moved at a time, this ratio is computationally efficient given by [57]

$$R_D = \frac{|D^{new}|}{|D^{old}|} = \sum_{j=1}^{n} \phi_j\left(\mathbf{r}_{i,new}\right)\left(D_{ji}^{old}\right)^{-1}.$$

The index j denotes the indices of the single-particle states and runs to $n = N/2$, the position vector $\mathbf{r}_i$ is of particle i .
Since we have two Slater determinants (spin up and down), we also need two inverses. An efficient way to avoid all if-tests on whether or not to access spin up or down, is to merge the two Slater matrices and inverses into one,

$$D = [D_\uparrow D_\downarrow] \qquad D^{-1} = \left[D_\uparrow^{-1} D_\downarrow^{-1}\right].$$

Provided that we order the single-particle states slightly different from the ones presented in figure 5.1, the correct matrix is then accessed automatically when calling a column corresponding to a specific particle. Tables 8.3 and 8.4 show our new procedure when ordering the states:
Each single-particle state is assigned an index j referring to the spatial part, i.e. the single-particle level without considering spin. Now the first half of the particles is assumed to have spin up, such that each of these particles is mapped to one of the levels j. When half of the particles is used, the other half is assigned the same indices j again, this time assumed to have spin down.

To update the inverse of the Slater matrix after a particle i has been moved, we use the

following relation,

$$\left(D_{jk}^{new}\right)^{-1} = \begin{cases} \left(D_{jk}^{old}\right)^{-1} - \frac{S_k \left(D_{jk}^{old}\right)^{-1}}{R_D} & \text{if} \quad k \neq i \\ \frac{\left(D_{ji}^{old}\right)^{-1}}{R_D} & \text{if} \quad k = i, \end{cases}$$

where

$$S_k = \sum_{l=1}^{n} \phi_l \left(\mathbf{r}_{i,new}\right) \left(D_{lk}^{old}\right)^{-1}.$$

The ratio between new and old Jastrow factor after movement of particle i is

$$R_J = \frac{J^{new}}{J^{old}} = \exp\left[\sum_{j=1, j\neq i}^{N} \left(g_{ji}^{new} - g_{ji}^{old}\right)\right].$$

The final ratio is then simply the product of the two components:

$$\frac{\Psi_{new}}{\Psi_{old}} = R_D R_J.$$

Simplification of the quantum force

As stated in section 7.2, the quantum force is given by

$$\mathbf{F}_i(\mathbf{R}) = 2\frac{1}{\Psi_T}\nabla_i \Psi_T.$$

To split it up into Slater and Jastrow part, we rewrite

$$\begin{aligned} \mathbf{F}_i(\mathbf{R}) &= 2\frac{\nabla_i \left(|D_\uparrow|\,|D_\downarrow|\,J\right)}{|D_\uparrow|\,|D_\downarrow|\,J} \\ &= 2\left(\frac{\nabla_i |D_\uparrow|}{|D_\uparrow|} + \frac{\nabla_i |D_\downarrow|}{|D_\downarrow|} + \frac{\nabla_i J}{J}\right). \end{aligned}$$

Since particle i is only contained in one of the Slater determinants (spin up or spin down), the expression simplifies to

$$\mathbf{F}_i(\mathbf{R}) = 2\left(\frac{\nabla_i |D|}{|D|} + \frac{\nabla_i J}{J}\right),$$

where $|D|$ is the determinant with the spin of particle i.

Optimization: Storing positions and distances

Since a lot of the calculations require the radial positions and relative distances between the particles, it would be a waste of CPU time to compute them again and again. We save much time by calculating and storing them once after each change of position.
The radial positions of the particles, $r_i = \sqrt{x_i^2 + y_i^2}$, are simply stored in a vector, or, to be more accurate, it is the square r_i^2 which we save. Since that one is needed more often than r_i

it is more efficient not to compute the square root additionally.
We store the relative differences between all particles in a symmetric matrix,

$$\mathbf{r}_{int} = \begin{pmatrix} 0 & r_{12} & r_{13} & \cdots & r_{1N} \\ & 0 & r_{23} & \cdots & r_{2N} \\ & & \ddots & \ddots & \vdots \\ & \cdots & & 0 & r_{(N-1)N} \\ & & & & 0 \end{pmatrix}.$$

The useful feature of this matrix is that when moving one particle i, only parts of the matrix (one row and one column) have to be updated. This means that we can reuse those elements not depending on particle i from the previous update of the matrix.

DFP algorithm for minimization

To implement the DFP algorithm, we use the function `dfpmin` of [69], which is a function that utilizes the DFP algorithm to find the minimum of a function f.

As discussed in section 7.2, the algorithm does not only require the function to be minimized, but also the gradient of this function. The partial derivative with respect to one of the parameters α_i is given by

$$\frac{\partial\langle E\rangle}{\partial\alpha_i} = \left\langle \frac{\Psi_i}{\Psi_T}E_L + \frac{\hat{H}\Psi_i}{\Psi_T} - 2\langle E\rangle\frac{\Psi_i}{\Psi_T} \right\rangle \tag{8.29}$$

$$= 2\left\langle \frac{\Psi_i}{\Psi_T}(E_L - \langle E\rangle) \right\rangle \qquad \text{(by Hermiticity)} \tag{8.30}$$

$$= 2\left\langle \frac{\Psi_i}{\Psi_T}E_L \right\rangle - 2\left\langle \frac{\Psi_i}{\Psi_T} \right\rangle\langle E\rangle, \tag{8.31}$$

with the definition $\Psi_i = \partial\Psi_T/\partial\alpha_i$.

To compute the derivatives with respect to the variational parameters numerically, we use the standard approximation for the first derivative,

$$\frac{1}{\Psi_T}\frac{\partial\Psi_T}{\partial\alpha} = \frac{\Psi_T(\alpha+h,\beta) - \Psi_T(\alpha-h,\beta)}{2h\Psi_T} + \mathcal{O}(h^2),$$

and an analogous expression for β,

$$\frac{1}{\Psi_T}\frac{\partial\Psi_T}{\partial\beta} = \frac{\Psi_T(\alpha,\beta+h) - \Psi_T(\alpha,\beta-h)}{2h\Psi_T} + \mathcal{O}(h^2).$$

The variable h denotes the step length, here chosen as $h = 0.002$.
It is important to note that this numerical evaluation of both derivatives has a rather bad efficiency. For each iteration, the complete wave function has to be evaluated four times, including the very time consuming Slater determinant.
For that reason, our second alternative uses an analytical approach, which is not only more

exact, but also way more efficient.
The derivative with respect to β is straightforward,

$$\frac{1}{\Psi_T}\frac{\partial \Psi_T}{\partial \beta} = \sum_{i<j} \frac{-a_{ij} r_{ij}^2}{(1+\beta r_{ij})^2}.$$

To compute the derivative of the Slater determinant with respect to α, we follow [57] and use for both parts, spin up and down,

$$\frac{1}{|D|}\frac{\partial |D|}{\partial \alpha} = \mathrm{tr}\left(D^{-1}\frac{\partial D}{\partial \alpha}\right) = \sum_{i,j=1}^{n} D_{ij}^{-1}\frac{\partial D_{ji}}{\partial \alpha}.$$

This means that we have to take the derivative of each single-particle wave function of the Slater matrix with respect to the variational parameter and finally take the trace of $D^{-1}(\alpha)\dot{D}(\alpha)$.
The derivatives of the single-particle orbitals are

$$\frac{\partial}{\partial \alpha}\phi_{n_x,n_y}(x,y;\alpha) = \left(\frac{\partial H_{n_x}}{\partial \alpha}H_{n_y} + H_{n_x}\frac{\partial H_{n_y}}{\partial \alpha} - \frac{\omega}{2}\left(x^2+y^2\right)H_{n_x}H_{n_y}\right)e^{-\frac{\omega}{2}\alpha(x^2+y^2)}.$$

The derivatives of the Hermite polynomials are given by

$$\frac{\partial H_{n_x}}{\partial \alpha}\left(\sqrt{\omega\alpha}x\right) = \frac{1}{2}\sqrt{\frac{\omega}{\alpha}}x\dot{H}_{n_x}\left(\sqrt{\omega\alpha}x\right)$$

for H_{n_x} and analogously for H_{n_y}.

To improve convergence, we performed two changes in the provided functions `dfpmin` and `lnsrhc`:
First of all, we noticed that it sometimes happened that during the search for minima, one of the variational parameters got negative, which caused the whole algorithm to diverge. We therefore check each new computed variational parameter and in case it gets negative, we reset it to the starting guess, which should already be quite a good choice.

Second, we recognized that after the gradient has been computed the first time, the first new trial parameters are often quite a lot away from the good starting parameters. This slows the algorithm down, such that it requires more iterations then actually are necessary and does not profit that well from the starting guess. For that reason, we normalize the first computed gradient to a norm equal 1. That way, the trial parameters stay close to the starting guess and we observe fewer iterations for convergence and better and more stable final results.

Parallelizing the code

We have parallelized the whole VMC algorithm using MPI (Message Passing Interface). Since in the Monte Carlo algorithm only averages have to be computed, the different jobs do not need to communicate, which makes it really easy:
The number of MC cycles is equally distributed among the processors and each processor calculates separately its contribution to the local energy and variance. At the end of the MC sampling, the master node collects the local sums using `MPI_Reduce`, adds them up and computes the final integral.

1. Loop over all cycles:
   (a) **WALK:** Loop over all walkers:
      i. Loop over all particles
         A. Calculate new trial position $\mathbf{r}' = \mathbf{r} + D\mathbf{F}(\mathbf{r})\tau + \chi$.
         B. Compute acceptance ratio R according to Eq. (7.34)
         C. Make Metropolis test: If $R < \epsilon, \epsilon \in (0,1) \rightarrow$ Metropolis-test=`true`.
         D. Check that no node has been crossed: If $\frac{\Psi_{\text{new}}}{\Psi_{\text{old}}} > 0 \rightarrow$ Node-test = `true`.
         E. If both tests are positive, accept the move. Else reject.
      ii. Compute E_{local}
   (b) **BRANCH**
      i. Compute branching factor G_B according to Eq. (7.15).
      ii. Decide which walkers will be killed and which ones will be cloned. Number in the next generation is $n_{\text{next}} = \text{int}(G_B + \epsilon)$, with ϵ random number $\epsilon \in (0,1)$.
      iii. Perform the killing/cloning.
2. During equilibration phase: Update the reference energy E_T.
3. Update statistics

Figure 8.7: Our DMC algorithm. We follow this procedure for each of the samples.

8.7.2 The Diffusion Monte Carlo part

In order to explain most efficiently how we implemented the DMC algorithm, we will first summarize the basic procedure and afterwards take up those parts that require special attention to secure correct convergence.

After initializing and thermalizing all walkers, we proceed for each sample as demonstrated in figure 8.7. Note that during the sampling phase, the reference energy E_T is updated after the loop over all cycles, i.e. one time per sample.
The following subsections will explain important parts of the algorithm in more detail.

Equilibration versus sampling phase

First of all, it is very important to differentiate between an equilibration and a sampling phase: At the beginning, all walkers are distributed randomly in configuration space. Therefore, before sampling, a steady state has to be reached, such that the distribution of walkers really represents the desired distribution. All the time there is a steady flow: Walkers are created in regions with low local energy and killed in the ones with higher local energy.
If the walkers are initially distributed very unlike the real distribution, there might occur unwanted population explosions or implosions. To retain the total weight of all walkers approximately stationary, we adjust the reference energy E_T after each cycle as in [70]:

$$E_T(t + \Delta t) = E_{\text{est}}(t) - \frac{1}{g\Delta t} \log \frac{W_t}{W_0}.$$

Here $E_{\text{est}}(t)$ is an estimate of the energy at time t, which we have chosen to be the average of the mixed estimator of the previous sample (see next paragraph for the mixed estimator). The second term attempts to reset the current number of walkers W_t to the target number W_0 after a number of g generations. As in [70], we choose $g = 1/\Delta t$, which is of the order of the correlation time of $e^{-\hat{H}t}$.
The main goal of this energy adjustment is not to obtain the correct estimate for the energy, as desired in the sampling phase, but to stabilize the number of walkers and obtain the right steady-state distribution.
In the following sampling phase, where the walkers are in a more or less stationary state, we update the trial energy less frequently and set it to the average of the previous sample, $E_T = E_{\text{old}}$.

Updating the energy: Mixed estimator

As stated in section 7.2, the wave function of fermionic systems exhibits nodes and in the vicinity of these nodes, the local energy shows a non-analytic behaviour. However, as shown explicitly in [70], the order of the error is not altered if the so-called *mixed estimator* is used in connection with an energy cut-off. The mixed estimator is defined as

$$\begin{aligned} E_{\text{mix}} &= \frac{\int d\mathbf{R}\ \Psi \hat{H} \Psi_T}{\int d\mathbf{R}\ \Psi \Psi_T} = \frac{\int d\mathbf{R}\ \Psi_T \Psi \frac{1}{\Psi_T} \hat{H} \Psi_T}{\int d\mathbf{R}\ \Psi \Psi_T} \\ &= \frac{\int d\mathbf{R}\ E_L f(\mathbf{R}, t)}{\int d\mathbf{R}\ f(\mathbf{R}, t)}. \end{aligned}$$

In our case,

$$f(\mathbf{R}, t) = \sum_{i=1}^{N} w_i \delta(\mathbf{R} - \mathbf{R_i}),$$

where N is the number of particles and w_i the weight associated with the walkers in the branching phase. That way, the energy contribution for each step is weighted with the branching factor and in the limit $t \to \infty$, where we approach the true ground state wave function $\Psi \to \Phi_0$, we have that $E = E_{\text{mix}}$.
We combine this mixed estimator with an energy cut-off, accounting for divergencies in the vicinity of nodes. It is commonly chosen as

$$E_L(\mathbf{R}) \to E_{\text{var}} + \frac{2}{\sqrt{\Delta t}} \text{sgn}\{E_L(\mathbf{R}) - E_{\text{var}}\},$$

for $|E_L(\mathbf{R}) - E_{\text{var}}| > 2/\sqrt{\Delta t}$, where E_{var} is the VMC energy obtained with Ψ_T.

Modifications to classical importance sampling

Utilizing the fixed-node approximation, we require that the walkers do not move across nodal surfaces of the trial wave function. To implement this requirement, we set $G(\mathbf{R}', \mathbf{R}; \Delta t)$ to zero if $\mathbf{R}'$ and $\mathbf{R}$ are on different sides of the nodal surface, meaning that moves attempting to cross nodes will always be rejected. In contrast to the common practice of killing all walkers straying across nodes, that way detailed balance is preserved.

N	ω	$E_{\text{analytical}}$	E_{DMC}
2	1.0	2	2.00000000(0)
2	0.28	0.56	0.56000000(0)
6	1.0	10	10.0000000(0)
6	0.28	2.8	2.80000000(0)
12	1.0	28	28.0000000(0)
12	0.28	7.84	7.84000000(0)

Table 8.5: Results of the ground state energy E_0 (in $[E_H]$) for systems without interaction. The second column shows the analytical results (no rounding).

Since DMC uses a statistical average of the local energy, there are always fluctuations resulting in energies that are lower than the true fixed-node energy. The problem is that, because the whole DMC algorithm is based on the selection of configurations with low energies, the number of walkers with those configurations will increase, until the trial energy E_T adjusts to stabilize the total population. Those *persistent configurations* result in a negatively biased energy. They may disappear due to fluctuations, but unfortunately it is more likely that they are replaced by other configurations that are even more strongly persistent, which produces a cascade of ever decreasing energies. This problem occurs most likely in the vicinity to nodes and has been observed by several authors [23, 71], who solved the problem by choosing very small time steps. If Δt is small, the acceptance ratio is always close to one, leading away from the persistent configurations. Using a small time step is also our strategy, in addition to moving only one electron at a time, which makes the acceptance probability even higher. The disadvantage is that small time steps make subsequent configurations more correlated and therefore increase the statistical error. For solving the problem of persistent configurations using larger time steps, we refer to the solution methods discussed in [70].
An additional suggestion of [70] is to replace the time step Δt by an effective time step $\Delta t_{\text{eff}} = A_r \Delta t$, where A_r is the acceptance ratio. The motivation is that each time a move is rejected, we make a small error in the time evolution, since time goes on without a diffusion step happening. However, since in our calculations the acceptance rate is always close to one, there is no observable difference.

8.7.3 Validation of code

In order to validate that our DMC code is working properly, we run the code first without interaction between the particles. This means that we exclude the interaction potential from the local energy, and include only the Slater determinant part in our ansatz for the trial wave function,

$$\Psi_T(\alpha, \beta) \to \Psi_T(\alpha) = |D_\uparrow(\alpha)|\,|D_\downarrow(\alpha)|\,.$$

Moreover, we set $\alpha = 1.0$, since this Slater determinant is known to represent the analytically correct wave function. Without interaction, we expect the ground state energy E_0 to be simply the sum of the single-particle energies. Table 8.5 confirms our expectations, which suggests that the tested parts of our program are working correctly.

To have a benchmark for the complete DMC program, with interaction, we compare our results with [13], where exactly the same systems, two-dimensional parabolic quantum dots,

N	ω	E_{DMC}	E_{DMC} in [13]
2	0.5	1.65976(2)	1.65975(2)
	1.0	3.00000(3)	3.00000(3)
6	0.28	7.6001(2)	7.6001(1)
	0.5	11.7855(8)	11.7888(2)
	1.0	20.1598(4)	20.1597(2)
12	0.28	25.636(1)	25.6356(1)
	0.5	39.162(2)	39.159(1)
	1.0	65.699(3)	65.700(1)

Table 8.6: Comparison of our DMC results with the ones of Lohne, Hagen, Hjorth-Jensen, Kvaal and Pederiva [13].

have been studied with DMC and the Coupled-Cluster method. Table 8.6 compares those of our results that are listed in [13], too. That our results are in very good agreement with the reference suggests that our DMC code produces correct results with reasonable precision and hence, it can be used as reliable benchmark for our SRG results.

Chapter 9

Computational results and analysis

In this chapter, we present the numerical results of our SRG calculations and analyse and discuss them. To get an idea about the performance of the SRG method, we start with the free-space case. After having sorted out computational challenges and methods to improve convergence, we continue with the in-medium calculations. Here we start with Wegner's canonical generator and identify the problems associated with this generator in our in-medium calculations. Eventually, we devote a whole section to IM-SRG(2) with White's generator. With this generator we perform our final large computations, compare IM-SRG(2) with other many-body methods and study the role of correlations in quantum dots.
Note that all quantities in this chapter are given in atomic units, as introduced in chapter 5. In particular, all energies are given in effective Hartrees $E_h^* \approx 11.86\,\mathrm{meV}$.

9.1 Free-space SRG

As a first test for the SRG method, we perform the evolution in free space. Obviously, this is computationally much less effective than the in-medium approach, since, at least for the lowest channel, the full Hamiltonian matrix needs to be set up. However, in contrast to the in-medium evolution, the flow equations are not truncated. This in turn means that we obtain a result which is correct within the SRG approach and can therefore be used as benchmark to measure the truncation error later on.

9.1.1 Code validation

In order to guarantee that our implementation is free of errors, we perform numerous tests where we know the exact result. Reproducing these results, we can be sure that the tested parts of our program work as they are expected to do and can be used as a reliable input for further calculations.

Hamiltonian matrix The free-space approach of the SRG method relies on a proper setup of the Hamiltonian matrix. It is therefore essential to test that this matrix is correct. One

of the major error sources is an incorrect phase after applying the creation and annihilation operators. Another one is a wrong interaction element due to incorrect spin considerations when computing the antisymmetric interaction elements.
To test the setup of our matrix, we take the example of $N = 2$ particles and $R = 3$ shells. The correct Hamiltonian matrix is presented in [67], which makes it possible to compare every single element.

The considered system contains twelve singe-particle states with the following mappings $|\alpha\rangle \rightarrow |n, m, m_s\rangle$:

$$\begin{aligned}
|0\rangle &\rightarrow |0, 0, -1\rangle, & |6\rangle &\rightarrow |0, -2, -1\rangle \\
|1\rangle &\rightarrow |0, 0, +1\rangle, & |7\rangle &\rightarrow |0, -2, +1\rangle \\
|2\rangle &\rightarrow |0, -1, -1\rangle, & |8\rangle &\rightarrow |0, +2, -1\rangle \\
|3\rangle &\rightarrow |0, -1, +1\rangle, & |9\rangle &\rightarrow |0, +2, +1\rangle \\
|4\rangle &\rightarrow |0, +1, -1\rangle, & |10\rangle &\rightarrow |1, 0, -1\rangle \\
|5\rangle &\rightarrow |0, +1, +1\rangle, & |11\rangle &\rightarrow |1, 0, +1\rangle.
\end{aligned}$$

As explained in chapter 5, the Hamiltonian does only connect states with the same quantum numbers M and M_s, resulting in a block-diagonal form. Since we are only interested in the ground state energy, we set $M = M_s = 0$ and obtain the following possible Slater determinants as basis for the Hamiltonian matrix:

$$|0, 1\rangle, \quad |0, 11\rangle, \quad |1, 10\rangle, \quad |2, 5\rangle, \quad |3, 4\rangle, \quad |6, 9\rangle, \quad |7, 8\rangle, \quad |10, 11\rangle.$$

In this basis, the Hamiltonian matrix has the following form:

$$H = \begin{bmatrix}
\langle 0,1|\hat{H}|0,1\rangle & \langle 0,1|\hat{H}|0,11\rangle & \langle 0,1|\hat{H}|1,10\rangle & \langle 0,1|\hat{H}|2,5\rangle & \langle 0,1|\hat{H}|3,4\rangle & \langle 0,1|\hat{H}|6,9\rangle & \langle 0,1|\hat{H}|7,8\rangle & \langle 0,1|\hat{H}|10,11\rangle \\
\langle 0,11|\hat{H}|0,1\rangle & \ddots & \cdots & \cdots & \cdots & \cdots & \cdots & \langle 0,11|\hat{H}|10,11\rangle \\
\langle 1,10|\hat{H}|0,1\rangle & \vdots & \ddots & & & & & \langle 1,10|\hat{H}|10,11\rangle \\
\langle 2,5|\hat{H}|0,1\rangle & \vdots & & \ddots & & & & \langle 2,5|\hat{H}|10,11\rangle \\
\langle 3,4|\hat{H}|0,1\rangle & \vdots & & & \ddots & & & \langle 3,4|\hat{H}|10,11\rangle \\
\langle 6,9|\hat{H}|0,1\rangle & \vdots & & & & \ddots & & \langle 6,9|\hat{H}|10,11\rangle \\
\langle 7,8|\hat{H}|0,1\rangle & \vdots & & & & & \ddots & \langle 7,8|\hat{H}|10,11\rangle \\
\langle 10,11|\hat{H}|0,1\rangle & \langle 10,11|\hat{H}|0,11\rangle & \langle 10,11|\hat{H}|1,10\rangle & \langle 10,11|\hat{H}|2,5\rangle & \langle 10,11|\hat{H}|3,4\rangle & \langle 10,11|\hat{H}|6,9\rangle & \langle 10,11|\hat{H}|7,8\rangle & \langle 10,11|\hat{H}|10,11\rangle
\end{bmatrix}$$

With our Hamiltonian (6.20) and oscillator frequency $\omega = 1.0$, this matrix should read [67]

$$H = \begin{bmatrix}
3.2533 & 0.3133 & -0.3133 & 0.3133 & -0.3133 & 0.1175 & -0.1175 & 0.2350 \\
0.3133 & 4.8617 & -0.2350 & -0.0783 & 0.0783 & -0.0881 & 0.0881 & 0.1371 \\
-0.3133 & -0.2350 & 4.8617 & 0.0783 & -0.0783 & 0.0881 & -0.0881 & -0.1371 \\
0.3133 & -0.0783 & 0.0783 & 4.8617 & -0.2350 & 0.3035 & -0.1469 & 0.1371 \\
-0.3133 & 0.0783 & -0.0783 & -0.2350 & 4.8617 & -0.1469 & 0.3035 & -0.1371 \\
0.1175 & -0.0881 & 0.0881 & 0.3035 & -0.1469 & 6.7160 & -0.1285 & 0.1395 \\
-0.1175 & 0.0881 & -0.0881 & -0.1469 & 0.3035 & -0.1285 & 6.7160 & -0.1395 \\
0.2350 & 0.1371 & -0.1371 & 0.1371 & -0.1371 & 0.1395 & -0.1395 & 6.7491
\end{bmatrix}.$$

This matrix is exactly reproduced by our code, suggesting that we set up the Hamiltonian correctly.

Diagonalization In order to make statements about the accuracy of the SRG method, we will compare our results with the ones obtained by exact diagonalization. For the small systems considered with the free-space approach, it is efficient enough to do this with the standard function `eig_sym` of the Armadillo library [73]. For the above Hamiltonian matrix, this function returns the lowest eigenvalue

$$E_0 = 3.0386,$$

which is in accordance to [67]. For two shells, we expect [24]

$$E_0 = 3.1523,$$

which is also reproduced. This implies that we apply the diagonalizing function correctly and can rely on it when benchmarking our SRG results.

9.1.2 Numerical results

Having verified that the SRG solver obtains the correct input matrix, we perform runs for systems of different number of particles N and oscillator frequencies ω.

As a first approach, we take a harmonic oscillator basis and study the convergence behaviour using a standard Coulomb interaction. In atomic units, the Hamiltonian of the two-dimensional parabolic quantum dot with this standard interaction reads

$$\hat{H} = \sum_{i=1}^{N} \left(-\frac{1}{2}\nabla_i^2 + \frac{1}{2}\omega^2 r_i^2 \right) + \sum_{i<j} \frac{1}{r_{ij}}, \tag{9.1}$$

see section 5.1. The oscillator frequency ω is used to tune the confining potential, which means the higher ω is, the more confined the particles are. Note that the complete non-interacting energy $E_{non} = \sum_i \epsilon_i = \omega \sum_i (2n_i + |m|_i + 1)$ is proportional to ω, whereas the oscillator frequency has no significant impact on the interaction energy. This means that the influence of the interaction energy, contributing to the potential energy, diminishes for larger values of ω, which in turn leads to a relative increase of the kinetic energy. Experimentally, the frequency can be tuned, too, and we perform all our numerical calculations with different values of ω. This allows to study how the frequency is affecting the results, as well as the numerical stability of the SRG method.

As introduced in section 6.2, the two most fundamental generators are

$$\hat{\eta}_1(s) = \left[\hat{T}_{\mathrm{rel}}, \hat{V}(s) \right] \tag{9.2}$$

and Wegner's canonical generator

$$\hat{\eta}_2(s) = \left[\hat{H}^{\mathrm{d}}(s), \hat{H}^{\mathrm{od}}(s) \right]. \tag{9.3}$$

Note that we label the generators with $\hat{\eta}_1, \hat{\eta}_2$ for easy access in the following.
In order to compare their effect on the flow of the Hamiltonian, we have performed calculations

ω	R	$\hat{\eta}_1$	$\hat{\eta}_2$	FCI
0.1	2	0.5125198414	0.5125198414	0.5125198414
	4	0.4418679942	0.4418679942	0.4418679942
	6	0.4414466720	0.4414466720	0.4414466720
	8	0.4412461536	0.4412461536	0.4412461536
	10	0.4411351270	0.4411351270	0.4411351270
0.28	2	1.127251038	1.127251038	1.127251038
	4	1.028803672	1.028803672	1.028803672
	6	1.025448813	1.025448813	1.025448813
	8	1.024199606	1.024199606	1.024199606
	10	1.023550577	1.023550577	1.023550577
0.5	2	1.78691353	1.78691353	1.78691353
	4	1.673872389	1.673872389	1.673872389
	6	1.667257181	1.667257181	1.667257181
	8	1.664806939	1.664806939	1.664806939
	10	1.663535219	1.663535219	1.663535219
1.0	2	3.152328007	3.152328007	3.152328007
	4	3.025230582	3.025230582	3.025230582
	6	3.013626129	3.013626129	3.013626129
	8	3.009235721	3.009235721	3.009235721
	10	3.006937178	3.006937178	3.006937178

Table 9.1: The ground state energy E_0 (in atomic units) for $N = 2$ particles is compared with the results from generator $\hat{\eta}_1 = \left[\hat{T}_{\text{rel}}, \hat{V}\right]$, generator $\hat{\eta}_2 = \left[\hat{H}^{\text{d}}, \hat{H}^{\text{od}}\right]$ and exact diagonalization (FCI). The large number of specified digits shall demonstrate that exactly the same results are obtained. This table is just an excerpt, the full table B.1 is given in Appendix B.

N	ω	R	$\hat{\eta}_1$	$\hat{\eta}_2$	FCI
6	0.1	3	4.149558313	4.149558313	4.149558313
		4	3.797435926	nc	3.797435926
	0.28	3	nc	nc	8.518319500
		4	7.851831187	nc	7.851831187
	0.5	3	12.89722859	12.89722859	12.89722859
		4	12.03694209	nc	12.03694209
	1.0	3	21.42058830	21.42058830	21.42058830
		4	20.41582765	nc	20.41582765
12	1.0	4	70.31250219	70.31250218	70.31250218

Table 9.2: Comparison of the ground state energy E_0 as in table 9.1. The label "nc" denotes non-converging runs.

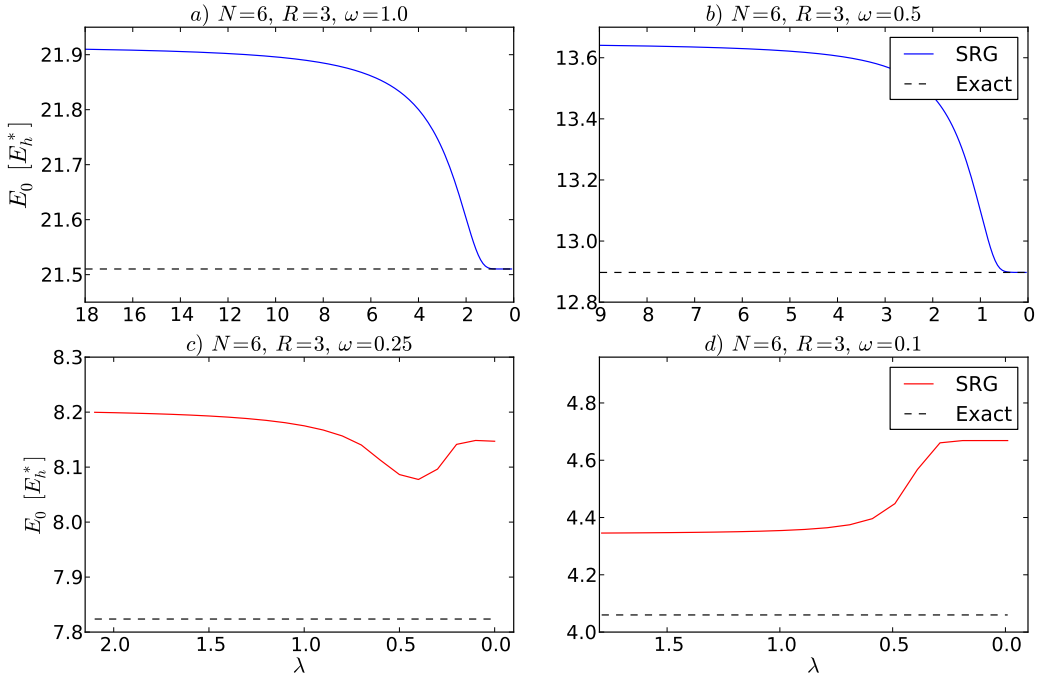

Figure 9.1: Typical plots demonstrating the behaviour of the ground state energy for converging (a, b) and non-converging (c, d) cases. For these plots, generator $\hat{\eta}_2 = \left[\hat{H}^{\mathrm{d}}, \hat{H}^{\mathrm{od}}\right]$ has been used, but converging/non-converging runs look similar with generator $\hat{\eta}_1$.

with both generators. Tables 9.1 and 9.2 display the results for systems with two, six and twelve electrons and compare them to the result obtained with exact diagonalization (FCI).

We observe that both generators reproduce to very high precision the ground state energy which is obtained by exact diagonalization of the Hamiltonian matrix. In other words, the ground state seems to get completely decoupled from the other matrix elements. However, Wegner's generator $\hat{\eta}_2 = \left[\hat{H}^{\mathrm{d}}, \hat{H}^{\mathrm{od}}\right]$ seems to be numerically more unstable and results more often in a non-converging ground state energy.

Since this is also of great importance later on, it should be demonstrated what we mean by converging and non-converging simulations:
Figure 9.1 shows some typical examples: In plots (a) and (b), the ground state energy E_0 is monotonically decreasing during the whole integration procedure and finally stabilizing at a constant value. Just if we integrate extremely close to $\lambda = 0$, in some cases numerical instabilities let E_0 suddenly explode to very large values again. However, this happens after a clear convergence on a longer interval and is not of physical relevance. On the other hand, with an increasing number of particles and a lower oscillator frequency ω, we often obtain results like examples (c) and (d) in figure 9.1. In plot (c), the energy increases again without having stabilized at a constant value before, and in plot (d), it does not decrease at all. Since the method theoretically should converge, we suspect numerical instabilities to be responsible for this effect. The convergence behaviour is discussed more thoroughly in subsection 9.1.3.

Having found out that in general, both generators $\hat{\eta}_1$ and $\hat{\eta}_2$ yield the same result, we analyse

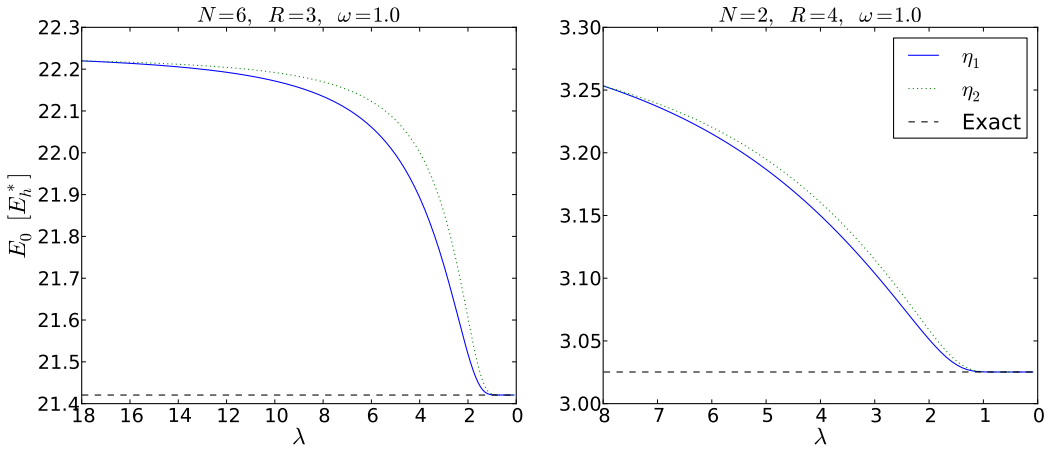

Figure 9.2: Comparison of the convergence behaviour with the two generators $\hat{\eta}_1 = \left[\hat{T}_{\text{rel}}, \hat{V}\right]$ and $\hat{\eta}_2 = \left[\hat{H}^{\text{d}}, \hat{H}^{\text{od}}\right]$. Although both generators lead to a converging result and convergence sets in at approximately the same value of λ, decoupling appears to be faster for $\hat{\eta}_1$.

whether there exist qualitative differences in the flow.
Figure 9.2 shows two typical plots: Although the final point of convergence is reached at approximately the same value of λ, it is noticeable that the curve of generator $\hat{\eta}_1$ constantly lies below the one of $\hat{\eta}_2$. This in turn means that even in the converging case, generator $\hat{\eta}_1$ seems to result in a better and faster decoupling for the studied quantum dots.

To verify the decoupling of the ground state by the SRG method and to illustrate how the Hamiltonian is driven to a diagonal form during the flow, we have made snapshots of the Hamiltonian matrix at different stages of the integration process. Since the non-interacting part of the Hamiltonian already is in diagonal form and those elements dominate the plots, we have chosen to restrict the figures to the interaction elements. A typical evolution for the case of $N = 2$ particles and $R = 4$ shells is shown in figure 9.3.
The Hamiltonian matrix behaves exactly as expected: At the beginning of the flow, we have non-zero elements at all places of the matrix. During the flow, first the elements which are far off the diagonal are zeroed out, followed by the ones closer to the diagonal. This is especially well illustrated in the transition from the fourth to the fifth plot, corresponding to $\lambda = 2.0$ and $\lambda = 1.0$, respectively. Note that throughout this thesis, the use of atomic units results in $[\lambda] = [E] = E_h^*$ due to $\lambda = s^{-1/2}$. In the case of $\lambda = 2.0$, only matrix elements connecting states with the highest possible energy differences seem to be close to zero, while in the plot for $\lambda = 1.0$, this is true for a much greater area of the matrix's upper and lower triangular part. Finally, in the last plot, the ground state seems to be completely decoupled from the remaining states. Even if not the whole interaction is completely diagonal, this decoupling should mean that the energy should have converged to its final value.
In order to confirm this statement, we have plotted the evolution of the ground state energy for $N = 2$ and $R = 4$ in figure 9.4. From $\lambda = 8.0$ down to $\lambda = 1.0$, we know from figure 9.3 that many matrix elements are zeroed out and, correspondingly, a large drop can be observed for the ground state energy E_0. Afterwards, the matrix in figure 9.3 does only change slightly and seems to have converged to its final shape for $\lambda = 0.1$. Correspondingly, we observe a

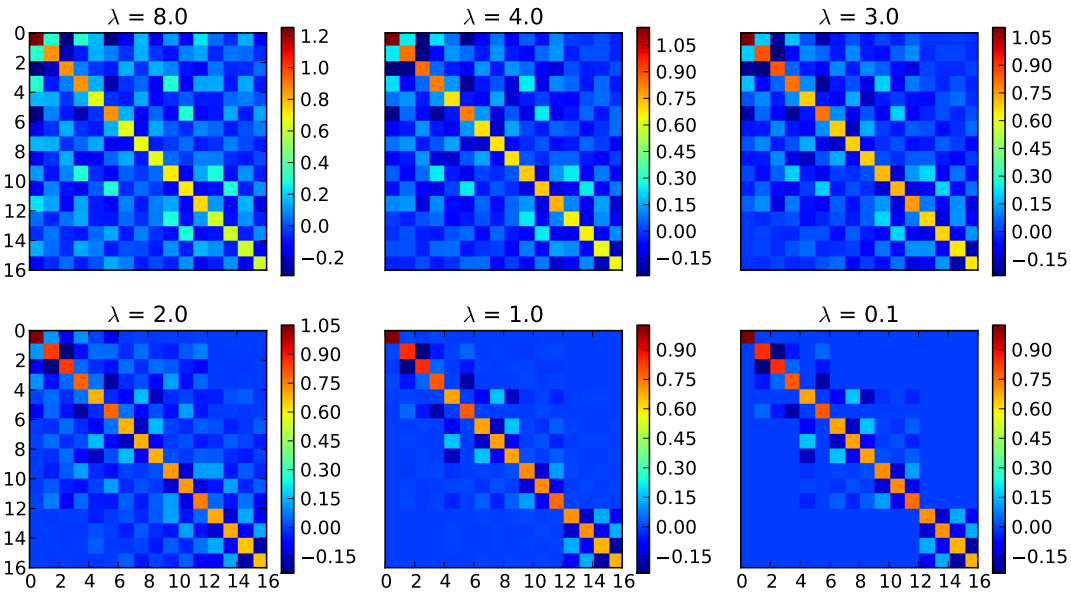

Figure 9.3: Snapshots of the interaction elements of the Hamiltonian matrix at different stages of the flow, specified by the flow parameter λ. The evolution is show for a system of $N = 2$ particles, $R = 4$ shells, oscillator frequency $\omega = 1.0$ and generator $\hat{\eta}_1 = [\hat{T}_{\text{rel}}, \hat{V}]$.

more or less constant value for E_0.
An interesting aspect of figure 9.3 are the groups of three to four states which the SRG method seems not to be able to zero out. To explain this phenomenon, it is necessary to look at the involved Slater determinants: Table 9.3 lists for each Slater determinant $|\alpha\rangle$ the two occupied singe-particle states, as well as the sum of both single-particle energies $\Sigma_\alpha = \epsilon_1 + \epsilon_2$. We observe that many of the Slater determinants are energetically degenerate. Beginning with the states with highest energy, the first degeneracy involves the last four states. The effect on figure 9.3 is that the last 4×4 block of the Hamiltonian does not converge to diagonal form. Furthermore, the next set of degenerate states is formed by the six determinants $|3, 18\rangle$ to $|10, 11\rangle$. Relating this to the plots of figure 9.3, we see that also these six states result in a non-zero block, which additionally is much more pronounced for clusters of 3×3 blocks. Regarding the shell structure of figure 5.1, we see that for the first three states of this 6×6 block, the particles occupy single-particle states of different shells, namely $R = 2$ and $R = 4$. For the remaining three states, all particles occupy the same shell $R = 3$.
All the named correspondences between table 9.3 and figure 9.3 show that the final shape of the Hamiltonian matrix depends very much on the energetic structure of the system and that energy degeneracies result in non-diagonal blocks.

In order to see the block-diagonal form in a more intuitive way, we need to exchange states $|2, 19\rangle$ and $|3, 4\rangle$, which the odometer algorithm unfortunately has been arranged in the wrong way regarding the energy Σ_α. In this energetically better arrangement, the final interaction part of the Hamiltonian looks as in figure 9.5, clearly illustrating the block-diagonal form due to energy degeneracies.

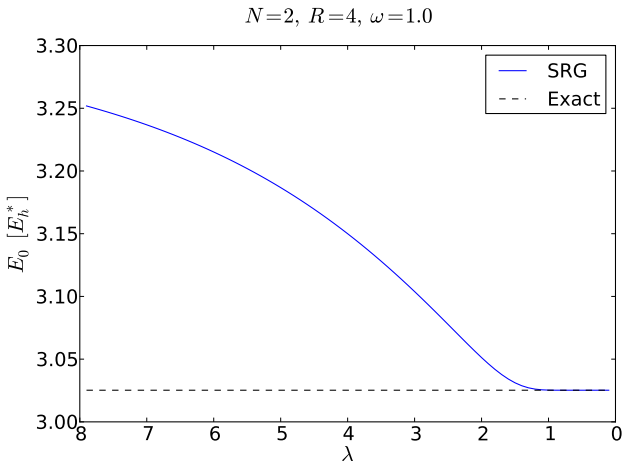

Figure 9.4: Evolution of the ground state energy for a system with $N = 2$ particles, $R = 4$ shells, oscillator frequency $\omega = 1.0$ and generator $\hat{\eta}_1 = [\hat{T}_{\text{rel}}, \hat{V}]$.

$\|\alpha\rangle$	Σ_α	$\|\alpha\rangle$	Σ_α
$\|0,1\rangle$	2ω	$\|5,16\rangle$	6ω
$\|0,11\rangle$	4ω	$\|6,9\rangle$	6ω
$\|1,10\rangle$	4ω	$\|7,8\rangle$	6ω
$\|2,5\rangle$	4ω	$\|10,11\rangle$	6ω
$\|2,19\rangle$	6ω	$\|12,15\rangle$	8ω
$\|3,4\rangle$	4ω	$\|13,14\rangle$	8ω
$\|3,18\rangle$	6ω	$\|16,19\rangle$	8ω
$\|4,17\rangle$	6ω	$\|17,18\rangle$	8ω

Table 9.3: Slater determinant basis for a system with $N = 2$ particles and $R = 4$ shells. The sum of the single-particle energies of both particles, $\Sigma_\alpha = \epsilon_1 + \epsilon_2$, is given in atomic units. The single-particle states are indexed as in figure 5.1

.

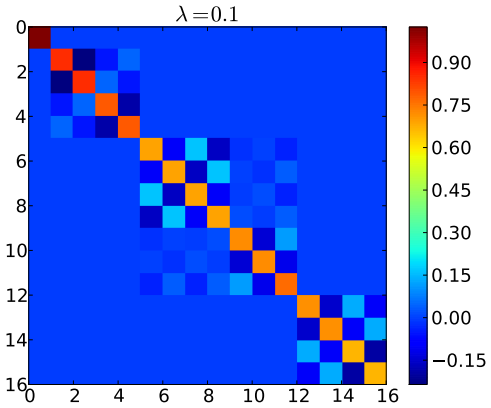

Figure 9.5: Interaction elements of the Hamiltonian for a system of $N = 2$ particles, $R = 4$ shells and $\omega = 1.0$. Snapshot of the evolution at $\lambda = 0.1$. Compared to figure 9.3, the basis states are explicitly ordered by increasing energy, pointing out the block-diagonal form of the final Hamiltonian.

The behaviour can easily be explained by looking at the first-order solution of the flow equations, Eq. (6.17), which we restate here:

$$V_{ij}(s) \approx V_{ij}(0)\mathrm{e}^{-s(\epsilon_i-\epsilon_j)^2}. \tag{9.4}$$

Until now, we argued that the off-diagonal elements are zeroed out the faster, the larger the energy difference $(\epsilon_i - \epsilon_j)$ between the corresponding bra- and ket-state is. In figures 9.3 and 9.5 the extreme case is illustrated, namely that energy degeneracies of two or more states prevent the interaction elements connecting those states to converge to zero. This is valid for Wegner's generator $\hat{\eta}_2$ as well as for $\hat{\eta}_1$.
In the case of quantum dots, however, we do not expect this to impact the ground state energy E_0, since it is uniquely defined. The fact that E_0 converges with SRG to the same value as an exact diagonalization, underlines and verifies this assumption. For more complicated systems, however, one should keep this fact in mind, and possibly draw on different generators.

9.1.3 Improving convergence: Effective interaction and Hartree-Fock basis

Especially as the frequency ω is lowered, we often encounter cases where the ground state energy E_0 does not converge. In the following, we will introduce two methods to improve convergence and analyse how the result is affected. First, we will exchange the standard Coulomb interaction by a more advanced effective interaction and compare the results. Afterwards, we will make use of a Hartree-Fock basis and examine how E_0 converges with this basis.

Effective interaction

Up to this point, we have used the Coulomb interaction as so-called *standard interaction*. Since this one has a divergency at $r = 0$, convergence is rather slow as function of R if a

harmonic oscillator basis is used.
A common way to solve this problem is to introduce a renormalized Coulomb interaction, called *effective interaction*, which aims to speed up the convergence rate as function of the number of oscillator shells. In this work, we utilize the same effective interaction $\hat{V}_{eff}$ that has already been very successfully applied in Full Configuration Interaction [17] and Coupled Cluster [13] calculations with two-dimensional quantum dots.

The basic idea is to look at a model space $\mathcal{P}$ of smaller dimension m than the full n-dimensional Hilbert space $\mathcal{H}$. This model space is spanned by a few eigenvectors $|e_k\rangle$ of the non-interacting Hamiltonian $\hat{H}_0$. Its orthogonal projector $\hat{P}$ is given by

$$\hat{P} = \sum_{i=1}^{m} |e_i\rangle \langle e_i| . \tag{9.5}$$

If we define $\mathcal{Q} \subset \mathcal{H}$ as orthogonal complement of $\mathcal{P}$, with orthogonal projector

$$\hat{Q} = 1 - \sum_{i=1}^{m} |e_i\rangle \langle e_i| = \sum_{i=m+1}^{n} |e_i\rangle \langle e_i| ,$$

then the Hamiltonian can be rewritten in the following block matrix form:

$$\hat{H} = \begin{bmatrix} \hat{P}\hat{H}\hat{P} & \hat{P}\hat{H}\hat{Q} \\ \hat{Q}\hat{H}\hat{P} & \hat{Q}\hat{H}\hat{Q} \end{bmatrix}.$$

Introducing the complex parameter z, we consider the interaction operator $\hat{V}$ as perturbation of the Hamiltonian, such that we can rewrite

$$\hat{H}(z) = \hat{H}_0 + z\hat{V}. \tag{9.6}$$

The approach in effective interaction theory is to find a unitary transformation

$$\hat{H}'(z) = e^{-X(Z)} \hat{H}(z) e^{X(z)}$$

that decouples the model space $\mathcal{P}$ from the complement space $\mathcal{Q}$, viz.

$$\hat{P}\hat{H}'\hat{Q} = \hat{Q}\hat{H}'\hat{P} = 0.$$

The effective Hamiltonian $\hat{H}_{\text{eff}}(z) = \hat{P}\hat{H}'(z)\hat{P}$ acts on a smaller space $\mathcal{P}$ than the full Hilbert space $\mathcal{H}$. Since a unitary transformation preserves an operator's eigenvalues, $\hat{H}_{\text{eff}}$ has $m = \dim(\hat{P}\mathcal{H})$ eigenvalues identical to the ones of $\hat{H}$. Our specific transformation is given by

$$X = \operatorname{artanh}(\tilde{\omega} - \tilde{\omega}^{\dagger}),$$

where the operator $\tilde{\omega}$ ensures that $\hat{H}'(z)$ is Hermitian and that the effective eigenvectors are as close as possible to the exact eigenvectors, see [17]. Moreover, we truncate $\hat{H}'(z)$ at the two-body level. Thus for $N = 2$ particles, exact diagonalization yields the exact ground state energy with respect to the full Hilbert space. For more than two particles, this is no longer true since we miss many-body correlations, but we still expect the overall interaction to be modelled more accurately.

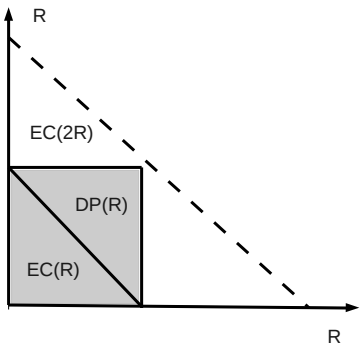

Figure 9.6: Illustration of the different spaces that are involved in forming the effective interaction. The standard approach generates the interaction elements in an energy-cut model space $\mathbb{EC}(R)$, corresponding to a cut in the global shell number R (and energy). The SRG method acts in a direct product space $\mathbb{DP}(R)$, where the single-particle shell number R (and energy) is limited to a maximum. For an *effective interaction without energy cut*, we use a larger energy-cut model space, $\mathbb{EC}(2R)$. The term 'without energy cut' arose because the whole direct product space $\mathbb{DP}(R)$ is included in $\mathbb{EC}(2R)$, such that there is no cut in $\mathbb{DP}(R)$.

The one-body part of $\hat{H}'$ is the unperturbed Hamiltonian $\hat{H}_0$, such that our effective interaction is given by

$$\hat{V}_{\text{eff}} = \hat{H}_{\text{eff}} - \hat{H}_0. \tag{9.7}$$

In our code, all interaction elements are generated by the OpenFCI library [17].
To be well-defined, the effective interaction must be generated in an energy-cut model space $\mathbb{EC}(R)$, which cuts the global shell number, i.e. $\sum_i^N R_i \leq R$. In contrast to the direct product space $\mathbb{DP}(R)$, where the single-particle shell number R is cut ($\max R_i \leq R$), the space $\mathbb{EC}(R)$ guarantees that all symmetries of the effective Hamiltonian are preserved. A use of $\mathbb{DP}(R)$ would break essential symmetries, resulting in an ill-behaved effective interaction, see [17]. However, although an application of the effective interaction in $\mathbb{EC}(R)$ is the correct approach and yields exact results for two particles, it seems to cause convergence problems when applied to quantum dots consisting of larger numbers of electrons [74]. Therefore the common practice arose [22,24,67,74] to use a basis that is twice as big, namely $\mathbb{EC}(2R)$. We will refer to the interaction in this basis as *effective interaction without energy cut*. Since our SRG method works in $\mathbb{DP}(R)$, whereas this effective interaction without energy cut acts in $\mathbb{EC}(2R)$, we simply restrict ourselves to the interaction elements in $\mathbb{DP}(R)$, in order to work in a model space of correct size. In the case of a large basis, the error is assumed to be fairly small, but as we will show convergence is considerably improved.

Tables 9.4 and 9.5 compare the results obtained with standard Coulomb interaction and effective interaction. As before, all converging runs yield precisely the same result as an exact diagonalization of the Hamiltonian in the same model space. We therefore omitted the comparison with exact diagonalization in the tables and rather focus on the relation between standard and effective interaction.
As stated above, the effective interaction applied to $N = 2$ particles should give the same result as a standard interaction in an infinite space. Exactly as expected, we therefore observe

$N = 2$

		Standard	Effective	
ω	R		E-cut	No E-cut
0.1	2	0.5125198414	0.440791888	0.4716227724
	3	0.4421887603	0.440791888	0.4408841339
	4	0.4418679942	0.440791888	0.4408938347
	5	0.4416137068	0.440791888	0.4408701421
0.28	2	1.127251038	1.021644014	1.066937680
	3	1.032681412	1.021644014	1.023735246
	4	1.028803672	1.021644014	1.022974149
	5	1.026588059	1.021644014	1.022380493
0.5	2	1.78691353	1.65977215	1.713577410
	3	1.681631996	1.65977215	1.664345671
	4	1.673872389	1.65977215	1.662657612
	5	1.669498218	1.65977215	1.661389243
1.0	2	3.152328007	3.000000000	3.063440415
	3	3.038604576	3.000000000	3.008602484
	4	3.025230582	3.000000000	3.005518845
	5	3.01760623	3.000000000	3.003199310

Table 9.4: Comparison of the ground state energy E_0 obtained with standard and effective interaction, for $N = 2$ particles. The runs with effective interaction have been performed with and without energy cut, see text for further explanation. The energy is given in atomic units. All runs have been performed with free-space SRG using generator $\hat{\eta}_1 = \left[\hat{T}_{\text{rel}}, \hat{V}\right]$.

$N = 6$

		Standard	Effective	
ω	R		E-cut	No E-cut
0.1	3	4.149558313	nc	nc
	4	3.797435926	nc	nc
0.28	3	nc	nc	8.086703549
	4	7.851831187	nc	nc
0.5	3	12.89722859	nc	12.37361922
	4	12.03694209	nc	11.86576513
1.0	3	21.42058830	nc	20.86307299
	4	20.41582765	nc	20.21066463

Table 9.5: Comparison of the ground state energy E_0 (in atomic units), analogously to table 9.4, this time for $N = 6$ particles. The label 'nc' denotes non-converging runs.

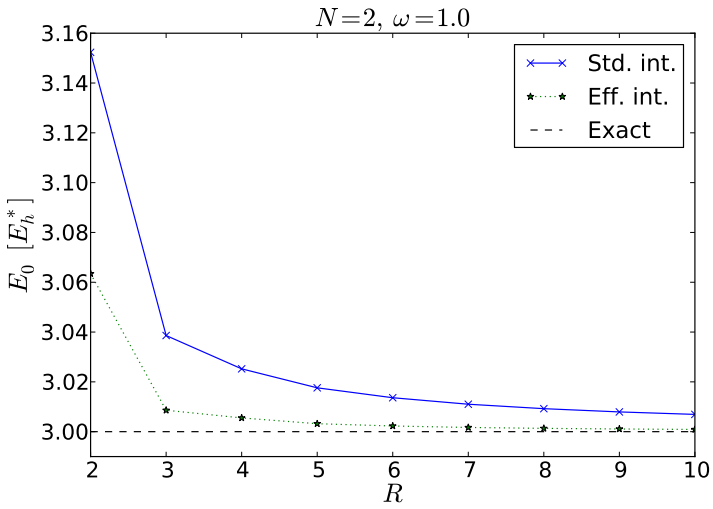

Figure 9.7: Comparison between standard and effective interaction (without energy cut) for $N = 2$ particles. With increasing number of shells R, the ground state energy E_0 converges towards the analytically exact result. The convergence with effective interaction is faster and the result for a given R a better approximation for E_0 than with standard interaction.

that the result for $N = 2$ particles with energy cut does not change as the number of shells R is increased. Moreover, we reproduce $E_0 = 3$ (in atomic units) for $N = 2$ electrons and $\omega = 1.0$, which is the exact result that is analytically derived in [10].
Although the effective interaction without energy cut is less exact, table 9.4 shows that it is numerically much more stable. Since the convergence behaviour with energy cut turns out to be much worse than with standard interaction, we will use the effective interaction only without energy cut from now on. This decision has also been made by V.K.B. Olsen studying two-dimensional quantum dots with Full Configuration Interaction [22] and C. Hirth using Coupled Cluster [24], which will therefore enable us to directly compare results in section 9.2.

Table 9.4 and figure 9.7 illustrate that for the ground state energy as function of the basis size, the effective interaction gives a better convergence than the standard Coulomb interaction. However, it does not seem to lead to a considerable improvement of the integrator's numerical stability. The convergence is getting worse as the oscillator frequency ω is decreasing and as the number of particles N is getting larger, which suggests that higher correlations between the electrons are the reason for the numerical instabilities.

Hartree-Fock basis

Since changing the interaction has not resulted in an improved numerical stability, we replace the harmonic oscillator by a Hartree-Fock basis. Note that after a Hartree-Fock calculation, the one-particle-one-hole excitations are zeroed out. Since these are assumed to have the greatest impact regarding the electron-electron interaction, we hope to have fewer problems caused by correlations.

A first important result is that runs with a Hartree-Fock basis give the same result as runs

$N = 2$, $\omega = 1.0$

	Standard			Effective		
R	2	5	10	2	5	10
HO basis	3.152328007	3.01760623	3.006937178	3.063440415	3.00319931	3.000894294
HF basis	3.152328007	3.01760623	3.006937178	3.063440415	3.00319931	3.000894294

$N = 2$, $\omega = 0.1$

	Standard			Effective		
R	2	5	10	2	5	10
HO basis	0.5125198414	0.4416137068	0.441135127	0.4716227724	0.4408701421	0.4408186851
HF basis	0.5125198414	0.4416137068	0.441135127	0.4716227724	0.4408701421	0.4408186851

Table 9.6: Comparison of the ground state energy E_0 (in atomic units) obtained with harmonic oscillator (HO) and Hartree-Fock (HF) basis. The large number of specified decimals shall demonstrate that the Hartree-Fock basis gives exactly the same results.

(a) Standard Interaction, $N = 6$

ω	R	HO basis	HF basis
0.1	3	nc	nc
	4	nc	nc
0.15	3	nc	nc
0.2	3	nc	nc
0.28	3	nc	nc

(b) Effective Interaction, $N = 6$

ω	R	HO basis	HF basis
0.1	3	nc	nc
	4	nc	nc
0.15	3	nc	nc
0.2	3	nc	nc
0.28	4	nc	7.703042

Table 9.7: Runs not converging with a harmonic oscillator (HO) basis are repeated with a Hartree-Fock (HF) basis. For the converging cases, the ground state energy E_0 is given in atomic units.

with a harmonic oscillator basis. This is demonstrated in table 9.6 , where we performed tests with $N = 2$ particles and different values of R and ω.
Since a basis transformation should preserve physical observables like energy, this confirms our expectations and implies that the Hartree-Fock method is correctly implemented. Additionally, it shows that the SRG method is also stable with a Hartree-Fock basis and converges to the correct result.

Having verified that, in the case of convergence, the basis transformation reproduces the earlier obtained results, we focus on the so far non-converging results and analyse whether convergence is improved with a Hartree-Fock basis. Since the generator $\hat{\eta}_1 = \left[T_{\text{rel}}, \hat{V}\right]$ yielded better numerics, we will limit ourselves to this generator.

Table 9.7 summarizes the results: With a standard interaction, we have not gotten any improvement. For low oscillator frequencies ω, the numerical integration is still unstable, resulting in a non-converging ground state energy. With an effective interaction, the stability has not considerably improved, either. Just in one case, the basis transformation could resolve the numerical problem.

From studies of two-dimensional quantum dots with the Coupled Cluster method [24, 67, 74], we know that convergence often gets better as the number of shells R is increased. Since due to an inclusion of higher excitations, an increase of R also corresponds to a better result, these

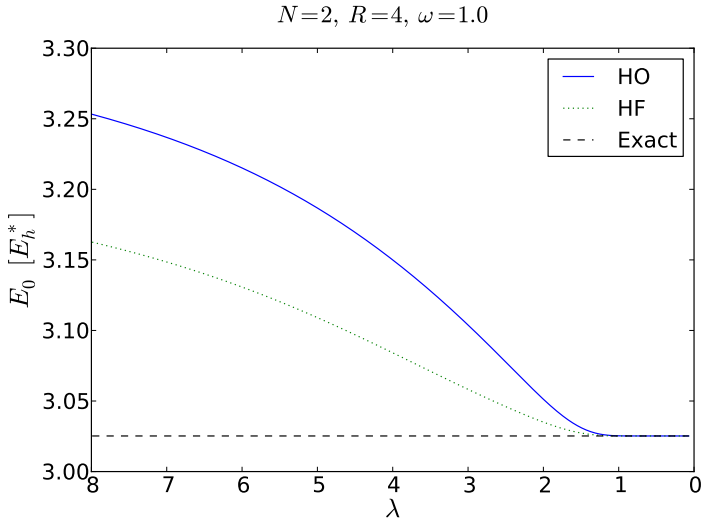

Figure 9.8: Comparison of the convergence behaviour with harmonic oscillator (HO) and Hartree-Fock (HF) basis. With HF basis, the one-particle-one-hole excitations are already zeroed out from the beginning, such that the ground state energy E_0 starts with a lower value. The plot shows a system with standard interaction and $\hat{\eta}_1 = \left[\hat{T}_{\text{rel}}, \hat{V}\right]$.

cases are more relevant, anyway. We therefore hope to obtain a better convergence pattern by including more basis states. However, due to its high demands on CPU time and memory, the free-space SRG method as applied here is not the method of choice in these cases. We will rather take benefit of the advantages of in-medium SRG and use this method to study systems with a larger number of basis states.
It should be mentioned that for practical calculations, where one is interested in numerical results for the system, one usually does not use free-space SRG as complete diagonalization method as we do here, but rather stops the flow at a certain value of λ and utilizes the transformed Hamiltonian for further calculations. However, in our analyses here, we are interested what the pure SRG method itself is accomplishing for our systems and we therefore study the complete decoupling of the ground state via SRG.

9.1.4 Time analysis

The advantage of the SRG method in free space is that no truncation is made and we therefore obtain the same result as exact diagonalization in the same model space. However, this requires to set up the full Hamiltonian matrix, which with increasing number of basis states gets exceedingly large.
Table 9.8 illustrates how the number of Slater determinants with the constraint $M = M_s = 0$ is rapidly increasing with the number of particles N and shells R. One the one hand, it might quickly exceed the available memory to store the full $n \times n$ Hamiltonian matrix. On the other hand, the computational costs get unacceptably high since each call of the derivative function (6.15) is of order $\mathcal{O}(n^3/2)$. The latter reason made us limit ourselves to rather small systems in table 9.2. To get an impression of the required CPU time of such a run, table 9.9

N	$R = 3$	$R = 5$	$R = 7$
2	8	29	72
6	64	16451	594118
12	-	1630953	579968

Table 9.8: Number n of relevant basis states in M-scheme with constraint $M = M_s = 0$.

λ	E_0	CPU time in s
20.0	**2**2.20366072	660
10.0	**2**2.05448854	1756
3.0	**2**0.99839817	2908
2.0	**20**.57836962	3363
1.4	**20.4**3153289	4608
1.0	**20.416**04434	6813
0.8	**20.41582**988	9126
0.6	**20.41582765**	14039

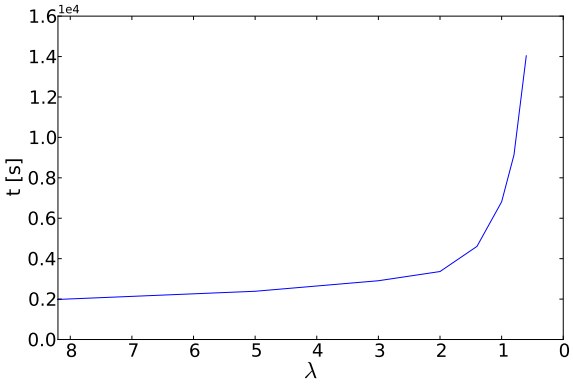

Table 9.9: Required CPU time for different values of the flow evolution parameter λ. Both the corresponding ground state energy E_0 and λ are given in atomic units. The studied system consists of $N = 6$ particles, with $\omega = 1.0$, $R = 4$, and we use generator $\hat{\eta}_1 = \left[\hat{T}_{\text{rel}}, \hat{V}\right]$. The interaction is a standard Coulomb one, and the basis is modelled by a harmonic oscillator basis. The bold letters indicate correct digits with respect to exact diagonalization.

lists how much time is needed for a system with $N = 6$ particles and $R = 4$ shells in order to obtain the ground state energy E_0 up to a certain accuracy. It gets obvious that especially the last part of the flow, i.e. the last correct digits, are very CPU demanding.

Analysing our SRG program explicitly, we have found out that for larger systems more than 90% of the CPU time is spent computing the derivatives. This percentage is even increasing as the dimension n of the Hamiltonian gets larger.
The problem with the free-space approach as used up to this point, is that we in diagonalize the full Hamiltonian matrix, but with a method that demands much more floating point operations than advanced iterative diagonalization methods like the Lanczos algorithm do. We remind that each computation of the flow derivatives (6.15) is of order $\mathcal{O}(n^3/2)$.
If we diagonalize the example of table 9.9 with the standard diagonalization function of the Armadillo library, the needed CPU time is about 3 seconds, which is five orders of magnitude smaller than the corresponding SRG time to obtain an accuracy of ten correct digits.
As mentioned before, one therefore usually does not use free-space SRG as complete diagonalization method in practice. However, when applying SRG in medium, considerable simplifications can be achieved.

9.2 In-medium SRG: Wegner's generator

Now that we have verified that the SRG evolution yields the correct result in free space, we perform the runs in medium. We focus especially on how the truncation error effects the results. Due to its greater computational efficiency, the in-medium approach allows us to study systems with much larger single-particle bases, which enables us to analyse how the ground state energy E_0 behaves as the number of particles N and shells R is increased.
In this section, we start with Wegner's canonical generator $\hat{\eta} = \left[\hat{H}^{\mathrm{d}}, \hat{H}^{\mathrm{od}}\right]$, which is the standard generator, and known to drive the Hamiltonian to a diagonal form in many applications. We will identify numerical problems and challenges connected to this generator, before we exchange it with White's generator in the next section.

9.2.1 Code validation

Analogous to free-space SRG, we need to check that the code gives reliable results and is free of bugs. An important benchmark is the free-space evolution, which does not contain any truncation errors. For $N = 2$ particles, we expect to be able to obtain the same result as with the free-space machinery. The reason is that interactions beyond the two-body level, which are omitted with IM-SRG(2), do not give any contribution with two particles, anyway. However, it is important to note that if we apply our equations straightforwardly, we do not get exactly the same result. The particle-hole formalism with the f-elements, including loop terms of two-body interactions, results in adding and subtracting some loop terms which are actually not present for two particles. In IM-SRG(3), IM-SRG(4) etc., these terms cancel each other and are therefore actually irrelevant, since the final result is correct. However, when truncating to IM-SRG(2), we have implicitly some of their leftovers. To test our in-medium code with the aim to get the exact untruncated result, we therefore modify our code slightly and suppress those terms explicitly.
Table B.3 in Appendix B shows that we now obtain exactly the same results. First of all, this implies that our in-medium implementation with the particle-hole machinery is working correctly. Second, we see that the in-medium approach does not introduce greater numerical roundoff errors, which would give a deviating result. Dealing with numerical methods, one should always check this since there exist many methods that are theoretically exact, but still give unstable results due to numerical issues.

9.2.2 Convergence analysis

We start our analysis of the ground state energies using a harmonic oscillator basis and a standard Coulomb interaction. For the two-particle system, we have already verified that the IM-SRG(2) approach gives the same result as exact diagonalization in the same model space, provided that one restricts oneself to contributions corresponding to one- and two-body diagrams. The more interesting cases, which we put focus on in this work, are the cases with $N \geq 6$ particles since they really profit from the in-medium machinery. Starting with $N = 6$ particles, we encounter the problem that only in the cases of $\omega = 1.0$ and $\omega = 0.5$ our results are converging. For lower values of ω, corresponding to higher correlations in the system,

	Harm. oscillator basis				Hartree-Fock basis			
R	$\omega = 0.1$	$\omega = 0.28$	$\omega = 0.5$	$\omega = 1.0$	$\omega = 0.1$	$\omega = 0.28$	$\omega = 0.5$	$\omega = 1.0$
3	nc	nc	12.87740	21.40509	nc	nc	12.89292	21.41679
5	nc	7.617524	11.88042	20.29962	3.57515	7.68673	11.90040	20.31070
7	nc	nc	11.79448	20.21363	3.54587	7.609189	11.81643	20.22692
9	nc	nc	11.77924	20.18864	3.54941	7.60536	11.80349	20.20133
11	nc	nc	11.77510	20.17660	3.55046	7.60300	11.79714	20.18868
DMC	-	7.6001(2)	11.7855(8)	20.1598(4)	3.5539(1)	7.6001(2)	11.7855(8)	20.1598(4)

Table 9.10: Ground state energies (in atomic units) obtained with IM-SRG(2) for $N = 6$ electrons. All calculations have been performed with Wegner's generator and bare Coulomb interaction. The label 'nc' denotes non-converging runs. For benchmarking, we also included the Diffusion Monte Carlo (DMC) results, where the number in brackets denotes the standard error. The full table with all results can be found in table B.7 in Appendix B.

numerical instabilities cause our integration algorithm to diverge. As we increase the number of particles to $N = 12, 20 \ldots$, this is even true for larger values of ω.

Hartree-Fock basis

To circumvent the problem high correlations cause, we introduce in line with our free-space SRG approach a Hartree-Fock (HF) basis. It is expected to be a much better starting point than the harmonic oscillator basis, since it produces a mean-field solution which is already closer to the fully correlated many-body solution. The Hartree-Fock calculation corresponds to a diagonalization of the part of the Hamiltonian which corresponds to one-particle-one-hole excitations. That way, the Hamiltonian is already "more diagonal" from the beginning on, which facilitates the work required of the SRG method. Especially for Wegner's generator, this approach is expected to be very helpful, since the one-body-one-hole excitations often link states with comparably low energy difference. Therefore those matrix elements are during the SRG flow usually the last ones to be zeroed out, which might result in a stiff and numerical unstable system of differential equations. Moreover, the Hartree-Fock calculation does not introduce a truncation error like SRG does. With this statement we mean that starting with a Hartree-Fock basis, exact diagonalization should yield the same result as with a harmonic oscillator basis. The Hartree-Fock calculation can thus be regarded as first step of diagonalization, but without the truncation error of the SRG method. For this reason, we expect the results to be closer to the exact ones.
As demonstrated in Table 9.10, this change of basis does for six particles indeed solve our numerical problems, and also runs with frequencies $\omega = 0.28$ and even $\omega = 0.1$ are converging.

Effect on CPU time Since the Hartree-Fock calculation does already diagonalize those parts of the Hamiltonian matrix corresponding to one-particle-one-hole excitations, we expect the SRG flow with HF basis to require fewer integration steps than with a HO basis. In table 9.11, we list the ratio between the required CPU time needed with a HF basis and a HO basis. The chosen configurations of N and R may seem a bit randomly, but the problem is that in most cases the calculation with HO basis do not converge. To compute the ratio, we

N	6	12	20
R	5	5	6
r	0.90	0.71	0.48

Table 9.11: Ratio between the required CPU time for IM-SRG(2) calculations with Hartree-Fock (HF) basis and harmonic oscillator (HO) basis. The time does only include the solution of the flow equations, not setup of the system, HF calculation etc. The ratio between both times is given by $r = t_{\mathrm{HF}}/t_{\mathrm{H0}}$. All calculations use Wegner's generator and oscillator frequency $\omega = 1.0$.

therefore had to pick out those runs where both calculations give a converging result.
The main observation is that that the calculations with a HF basis require less time than those with a HO basis. Moreover, the time difference gets more pronounced as the number of particles N is increased. This behaviour meets our expectations and supports the assumption that zeroing out the one-particle-one-hole excitations is time-consuming and needs many integration steps, possibly due to the stiffness of the equation system. With a large number of particles, there are obviously more of those excitations possible, which explains the larger speedup for more particles.

Effective interaction

Although numerical stability has improved applying a Hartree-Fock basis, the convergence of the energies as function of the number of oscillator shells is slow. As explained for free-space SRG, this is mainly because the harmonic oscillator basis, and linear expansions of it, do not properly account for the divergency of the Coulomb interaction at $r = 0$. To solve this problem, we apply again a renormalized Coulomb interaction.

Figure 9.9 shows for the example of $N = 6$ particles how convergence is accelerated as function of the number of oscillator shells. In particular, performance with respect to the DMC result is best when combining the effective interaction with Hartree-Fock basis. This is exactly what we hoped for and supports our assumption that calculations with HF basis introduce smaller errors than the ones with HO basis.
Moreover, figure 9.9 shows that for a low number of shells R, the effective interaction yields much better results than the standard interaction does. This meets also our expectations, since the effective interaction speeds up the convergence as function of R.

We subsequently performed calculations with higher numbers of particles N. Here we encountered the next problem: Especially as the oscillator frequency ω is lowered, the system of equations gets stiff, which causes difficulties in the integration process. For well converging runs, it is usually sufficient to integrate maximally to $\lambda = 0.1$ until E_0 has stabilized. However, for calculations with $N \geq 12$ particles, we need to integrate much further until convergence is reached. For the example of $N = 12, R = 6, \omega = 0.1$, the ground state energy E_0 has still not converged for $\lambda = 0.008$, see listing B.1 in the Appendix, whereas usually integration to $\lambda = 0.2$ or $\lambda = 0.3$ is sufficient. As we already observed for the free-space SRG calculations and now again see for IM-SRG in figure 9.10, the required CPU time blows enormously up as λ is approaching zero, even if the calculations have a comparably good convergence behaviour.

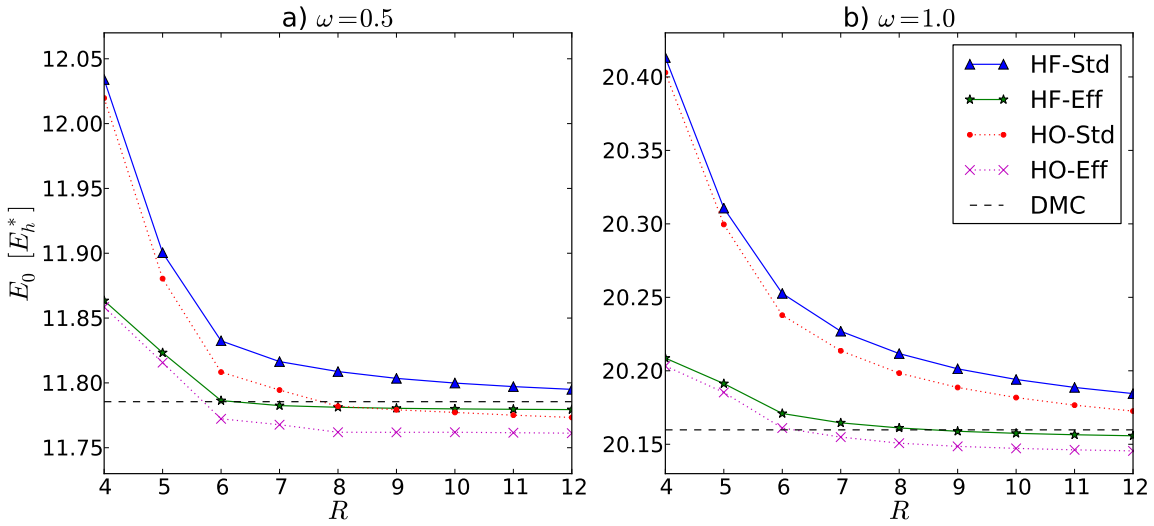

Figure 9.9: Ground state energies for a circular quantum dot with $N = 6$ particles, obtained with IM-SRG(2) using Wegner's generator. Results are displayed for a Hartree-Fock basis, both with standard interaction (HF-Std) and effective interaction (HF-Eff), and for a harmonic oscillator basis, also with standard (HO-Std) and effective interaction (HO-Eff). For comparison, we have also included results obtained with Diffusion Monte Carlo (DMC). The figure uses the data of table B.7 and B.8 in Appendix B.

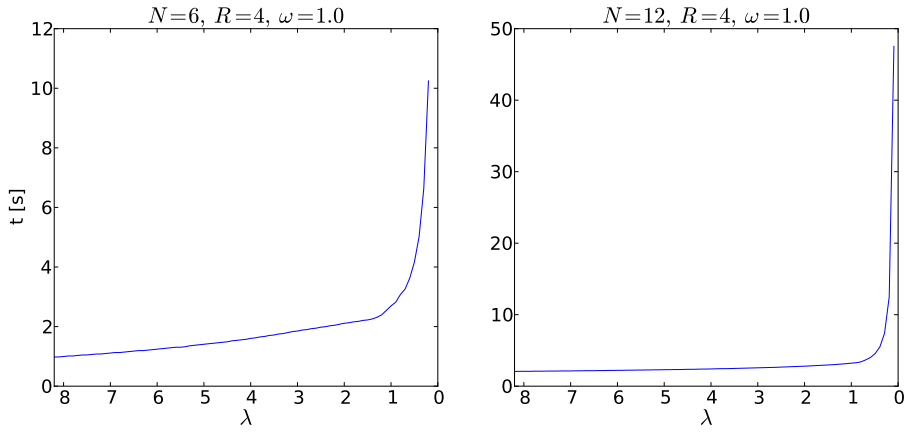

Figure 9.10: Required CPU time as function of the flow parameter λ (in atomic units). The example is with HF basis and effective interaction. The given time includes only the integration, not setup of the system, Hartree-Fock calculation etc.

ω	0.1	0.28	0.5	1.0	2.0	3.0	5.0
t_{Std}	nc	nc	45.0	15.1	13.8	7.8	11.2
t_{Eff}	nc	nc	nc	26.9	10.9	6.0	9.1

Table 9.12: Required CPU time (in s) such that the energy E_0 converges to six decimals. The first line lists the results with standard Coulomb interaction, the second one with effective interaction. The label 'nc' denotes non-converging runs.

Stiffness of the equations thus results in massive computational costs.
Moreover, for larger numbers of particles, the calculations are not converging any more. When performing runs with $N = 20, 30, \ldots$ particles, almost no calculation with $\omega \in \{0.1, 0.28, 0.5, 1.0\}$ is converging. However, numerical stability improves as we increase the oscillator frequency ω, and runs with $\omega \in \{2.0, 3.0, \ldots\}$ are converging and generally with fewer required integration steps, see also table 9.12.
The fact that these numerical instabilities arise for larger values of N as well as smaller values of ω, indicates that higher correlations between the particles cause the problem and set limits to the applicability of our method.

In the following section we analyse whether a change from Wegner's to White's generator yields better numerical properties.

9.3 In-medium SRG: White's generator

9.3.1 Motivation

As explained in chapter 6, White's generator introduces similar decaying speeds for all matrix elements, making numerical approaches much more efficient. Two main motivations made us use this generator:
First of all, we hope to solve the problem of stiffness of the equations and to be able to get converging results for systems with higher correlations, too. Second, as we have shown in section 8.5, White's generator is computationally much more efficient. Since the generator focuses on zeroing out the one-particle-one-hole and two-particle-two-hole excitations connected to the reference state $|\Phi_0\rangle$, the only non-vanishing terms are one-body terms of the form $\eta^{(1)}_{ph}$ and two-body terms of the form $\eta^{(2)}_{pphh}$, where p denotes particles and h holes. Making use of these properties, an effective implementation can simplify Eqs. (6.22)-(6.24) considerably, reducing the number of required floating point operations. Apart from better stability, we therefore expect our calculations to take less CPU time, which is of special importance as we want to increase the number of particles N and single-particle states in our basis.

This section is structured as follows: After validating our code, we compare the convergence behaviour of IM-SRG(2) with both Wegner's and White's generator . Particularly, we look at how the required CPU time is changed, indicating the number of integration steps and therefore being a measure for the stiffness of the equations. Afterwards, we analyse how numerical stability is affected. If the results are satisfying, we continue our calculations with larger ranges of particles N and shells R and study how well applicable our method is for those systems. Finally, we compare our results with other many-body methods.

R	n	R	n
5	6640	17	22274038
7	57602	20	67647814
10	613572	22	130003774
12	2092438	25	312789734
15	9495322		

Table 9.13: Number of coupled ordinary differential equations n that have to be solved for $N = 2$ particles.

9.3.2 Code validation

To be sure that the code is free of errors, we first performed a number of test calculations. As for Wegner's generator, we take the case of $N = 2$ particles. Provided that we restrict ourselves to contributions corresponding to one- and two-body diagrams, we expect to obtain with the IM-SRG(2) machinery the same results as with free-space SRG, which is not subject to any truncation error.
The results are listed in table B.4 in Appendix B, where we performed calculations with HO as well as HF basis. We obtain exactly the same results, indicating that the code is working correctly.

9.3.3 Comparison with Wegner's generator

As mentioned above, we expect the calculations with White's generator to require less CPU time than the ones with Wegner's generator. The first reason is that the flow equations simplify due to the structure of the generator $\hat{\eta}$, which reduces the number of floating-point operations per integration step. The second reason is that we assume the equations not to be stiff any more, reducing the number of integration steps itself.
Figure 9.11 shows that our assumptions are right: The calculations with White's generator do indeed require much less time, which seems to be even more pronounced as the number of shells R is increasing. This can be explained by the fact that the simplification of the flow equations has greater impact as the number of single-particle states is increasing.
Another important observation is that the huge blow up of time for small values of λ, which we faced with Wegner's generator, no longer exists to the same extent. This indicates that the equation system is not stiff any more and exhibits better numerical properties. This makes it more straightforward to solve with the ODE solver.

Since the aim of our SRG calculations is to extract the ground state energy in the limit $s \to \infty$, both generators must, up to truncation errors, give the same result. Numerical comparisons by us yield energy differences of the order of $0.05\,E_h^* \approx 0.6\,\mathrm{meV}$ or less. As illustrated in Fig. (9.12), these differences are rather independent of the number of oscillator shells R. Since the flow equations with Wegner's generator get stiff and often do not converge for $N \geq 6$ particles, we continue our larger computations with White's generator. The full results of the IM-SRG(2) calculations with White's generator can be found in Appendix B. Since calculations with a HF basis generally result in a smaller error, all the runs have been performed with a preceding Hartree-Fock calculation.

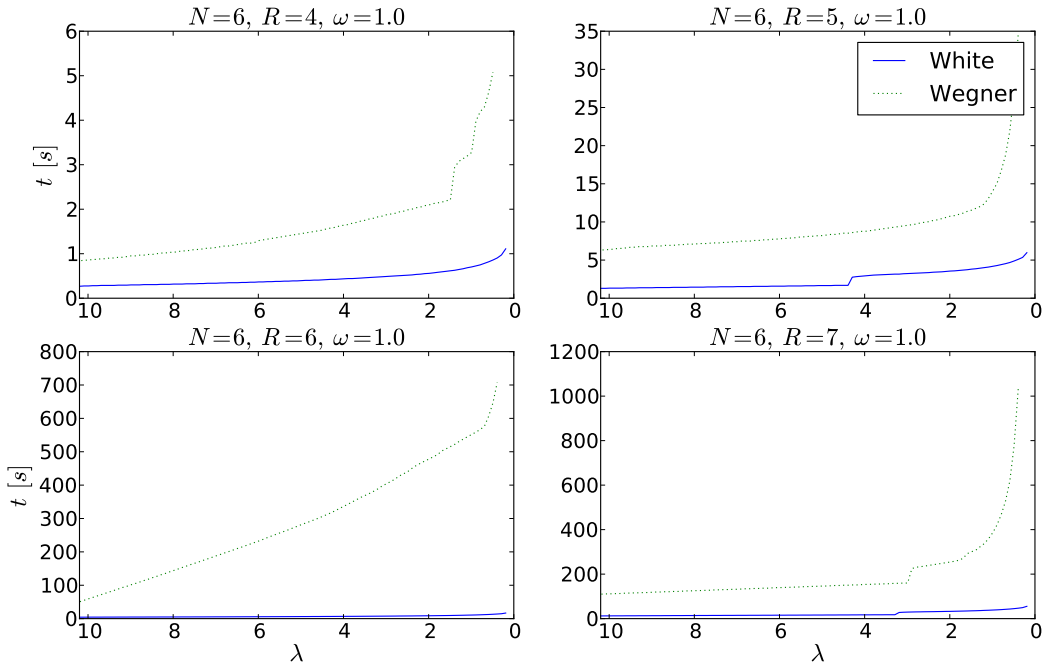

Figure 9.11: Required CPU time for SRG calculations performed with White's and Wegner's generator. The runs have been performed with HF basis and standard interaction. To give a reasonable comparison, the time does only consider the integration of the flow equations, not setup of the system, Hartree-Fock calculation etc. The flow parameter λ is given in atomic units.

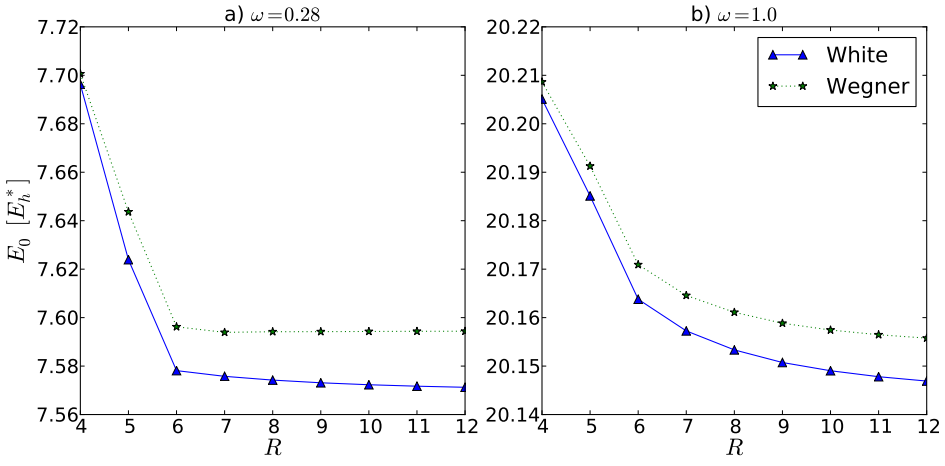

Figure 9.12: Ground state energies for a circular quantum dot with $N = 6$ particles. We compare the results obtained with White's and Wegner's generator, respectively. All calculations have been performed with Hartree-Fock basis and effective interaction.

N	$R = 10$	$R = 14$	$R = 16$	$R = 20$
2	0.36	0.9	2.3	10.2
6	0.9	5.1	11.9	74.6
12	2.8	22.7	50.6	182.7
20	11.8	30.9	108.4	659.8

Table 9.14: Required CPU time (in hours) for runs with $\omega = 1.0$, Hartree-Fock basis and effective interaction. Here, we used the computing cluster "Abel" at the University of Oslo.

To get an impression of the dimension of the problem to be solved, table 9.13 lists the number of ordinary differential equations to be solved for the example of $N = 2$ particles. It gets evident that for each additional included shell R, the number of equations is considerably increased, which of course is reflected in the required CPU time. For this reason, we restricted us to maximally $N = 20$ shells, although we in principle could have treated larger basis sizes, too. However, as we would neither get any new insights into the physics of the systems, nor in the SRG method itself, we think that $R = 20$ shells is a reasonable limit. It provides all information needed for further analyses, which we will come back to in the next section.
It should be mentioned that the number of differential equations is nearly independent of the number of particles N, and that more or less just the number of included shells R determines the size of the problem. However, for a larger number of particles, our model space is larger, too, suggesting that we need to integrate over a longer interval. Therefore the number of integration steps is increased, which requires additional CPU time, see table 9.14.

9.3.4 Comparison with other many-body methods

In the following, we examine how our IM-SRG(2) results behave as function of shells R and rate them with respect to other many-body methods. Doing so, we consider the following methods:

- **Hartree-Fock** (HF): The HF code is written by us, as explained in section 8.6. Since we use a Hartree-Fock basis, each SRG calculation is preceded by a Hartree-Fock calculation. This means that the Hartree-Fock energy is readily available, anyway.

- **Diffusion Monte Carlo** (DMC): As described in section 7.2, we have developed an own DMC code independently of the SRG implementation. Since DMC, apart from statistical and systematic errors, provides in principle the exact solution and is not dependent on a basis size like the other methods, it is a valuable benchmark for our SRG results.

- **Full Configuration Interaction** (FCI): The FCI data we use for comparison are from [22]. For a given number of single-particle states n_{sp}, FCI includes all possible Slater determinants, which are obtained by exciting particles from the ground state configuration to all possible virtual orbitals. Within this basis, the eigenvalues of the full Hamiltonian are computed.
  Since for a given Hamiltonian, the method is exact within the space spanned by the basis functions, FCI is a valuable reference for our results. However, only systems with a rather small number of included shells R can be treated with FCI, since the number

of determinants in the basis grows factorially with the number of particles and single-particle orbitals. For N electrons and n_{sp} single-particle states, the Hilbert space has dimension

$$\dim(\mathcal{H}) = \binom{n_{sp}}{N}.$$

For an example of $N = 12$ electrons and $R = 10$ shells, which is still one of the smaller systems considered by us, one would have about $3.5 \cdot 10^{15}$ possible Slater determinants, which is beyond the limit of current FCI calculations [75]. The largest system we run in this thesis, $N = 42$ particles and $R = 20$ shells, corresponding to 420 basis functions, yields approximately $1.3 \cdot 10^{58}$ determinants, which is way too large for an exact diagonalization.

- **Coupled Cluster**: The Coupled Cluster data we compare our results with are from C. Hirth [24]. The method aims to solve the time-independent Schrödinger equation using an exponential ansatz for the wave function,

$$|\Psi\rangle \equiv e^{\hat{T}}|\Phi_0\rangle, \tag{9.8}$$

where $|\Phi_0\rangle$ is a reference Slater determinant and $\hat{T}$ the cluster operator including all possible excitations. If the excitations are sorted by the number of excited electrons, $\hat{T}$ can be expressed as sum of a one-particle-one-hole operator (single excitations), two-particle-two-hole operator (double excitations) etc.

$$\hat{T} = \hat{T}_1 + \hat{T}_2 + \hat{T}_3 + \dots$$

A common approach, called CCSD and used by C. Hirth, is to only include singles and doubles, such that Eq. (9.8) simplifies to

$$|\Psi\rangle \approx e^{\hat{T}_1 + \hat{T}_2}|\Phi_0\rangle.$$

This ansatz for the wave function is now used to iteratively solve Schrödinger's equation. For more details concerning the setup and solution of the Coupled Cluster equations, we refer to [25]. The advantage of CCSD over FCI is that much larger systems can be treated, which gives us a reference method where no FCI data are available. Moreover, CCSD is subject to truncation, similar to IM-SRG(2), giving a good starting point to compare the truncation errors made by the different methods.

Figures 9.14 to 9.18 show our IM-SRG(2) results as function of the shell number R and compare them with other many-body methods. In the first plots, we consider oscillator frequency $\omega = 1.0$, afterwards we look at lower values for ω, corresponding to higher correlations. Note that all DMC results up to $N = 12$ particles have been obtained with our own code, whereas the results for $20 - 42$ particles are provided by the fellow master student J. Høgberget, who is developing a computationally highly optimized DMC code in his master's project.
Let us in the following discuss the different many-body methods, starting with Hartree-Fock: Evidently, SRG performs much better. The curve of the Hartree-Fock results lies considerably more off the ones of DMC than our SRG curve does. This meets our expectations, since Hartree-Fock as mean-field calculation only considers one-particle one-hole excitations, while we take higher excitations into account, too. The comparison to Hartree-Fock is of particular interest since Hartree-Fock serves as starting point for our IM-SRG(2) calculations. Hence the

improvement from the Hartree-Fock to the SRG curve arises from those correlations included in IM-SRG(2) that are beyond the mean-field approximation.
A next interesting point is how SRG compares to FCI, which can be regarded as exact for a given value of R. Since due to enormous computational costs, FCI is limited to systems of rather small numbers of N and R, we have unfortunately not that many data for comparison available. In all cases where we have data available, also for systems not plotted here, our ground state energies lie systematically slightly below the FCI results. This suggests that a positive term is omitted when truncating to IM-SRG(2), which is possible since the method is not variational.
For larger values of N and R, we can evaluate our results with respect to DMC and CCSD calculations. As discussed in [27], both IM-SRG and Coupled Cluster can be interpreted as re-summation of the perturbation series for the ground state energy. Similar to CCSD, the IM-SRG(2) energy contains (at least) the complete third-order expansions, and with a superficial similarity between the cluster operator and our generator $\hat{\eta}$, one can expect a similar performance. However, in contrast to traditional Coupled Cluster methods, the effective Hamiltonian in IM-SRG is always Hermitian. This implies that IM-SRG resembles rather unitary Coupled Cluster (UCC) than CCSD, with the first one converging more rapidly [76].
For our quantum systems, figures 9.14 to 9.18 demonstrate that for all cases with $N > 6$ particles, IM-SRG(2) lies closer to the DMC result than CCSD does. Since DMC is expected to lie really close to the true ground state energy, we can thus conclude that for $N > 6$ electrons, IM-SRG(2) performs better than the Coupled Cluster method limited to singles and doubles.
An explicit numerical comparison between SRG and CCSD, up to $N = 42$ particles, is given in table B.19 in Appendix B. The table confirms quantitatively the behaviour observed in the plots: As the number of particles is increased, IM-SRG(2) approximates the DMC value steadily better than CCSD does. This is a very important result, demonstrating the power of IM-SRG(2).
To compare our results directly with DMC, we introduce the relative difference

$$\epsilon_{IM-SRG} = \left| \frac{E_{IM-SRG} - E_{DMC}}{E_{DMC}} \right|. \tag{9.9}$$

This quantity is plotted in figure 9.13, summarizing the trend which can also be observed in figures 9.14 to 9.18: Generally, as the number of particles N is increasing, the IM-SRG(2) ground state energy approximates the DMC energy better and better. Only for $N = 42$ particles, the difference is increasing again. This can be explained by the fact that for smaller numbers of particles, a model space with $R = 20$ shells gives a much better accuracy than for $N = 42$ particles. To obtain the same degree of excitations, we would have to run the calculations for higher numbers of particles with more shells.
For lower values of the oscillator frequency ω, the deviations of the IM-SRG(2) to the DMC result get larger. This can be explained by the increasing significance of correlations for lower frequencies ω, an effect which we will study in the next section. However, as already mentioned, for $N > 6$ electrons the deviations of IM-SRG(2) to DMC stay are still smaller than the corresponding CCSD ones.

Concerning the error of our IM-SRG calculations, we want to note two error sources: The first one comes from the finite size of the single-particle basis, which means that our model space does not include all possible particle-hole excitations. This error can be decreased by including

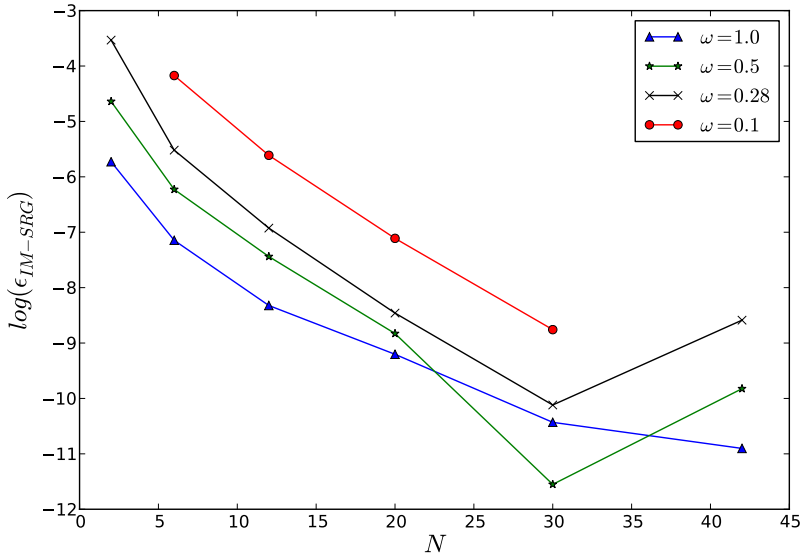

Figure 9.13: Relative difference between IM-SRG(2) and DMC, as defined in Eq. (9.9), for different number N of particles and oscillator frequencies ω. We use our optimal IM-SRG(2) results, with effective interaction, Hartree-Fock basis and basis size $R = 20$.

more shells R. Assuming that the DMC result lies very close to the exact solution, we observe correspondingly a smaller difference between the IM-SRG(2) and DMC ground state energy as we increase the number of oscillator shells R. The second error comes from the truncation of the flow equations, which in the case of IM-SRG(2) means that all operators are truncated on a two-body level. This error is intrinsic to the specific IM-SRG method itself and can only be decreased by including higher-body interactions, for example with IM-SRG(3), IM-SRG(4) etc. A full listing of all our results, up to $N = 42$ particles, can be found in Appendix A.

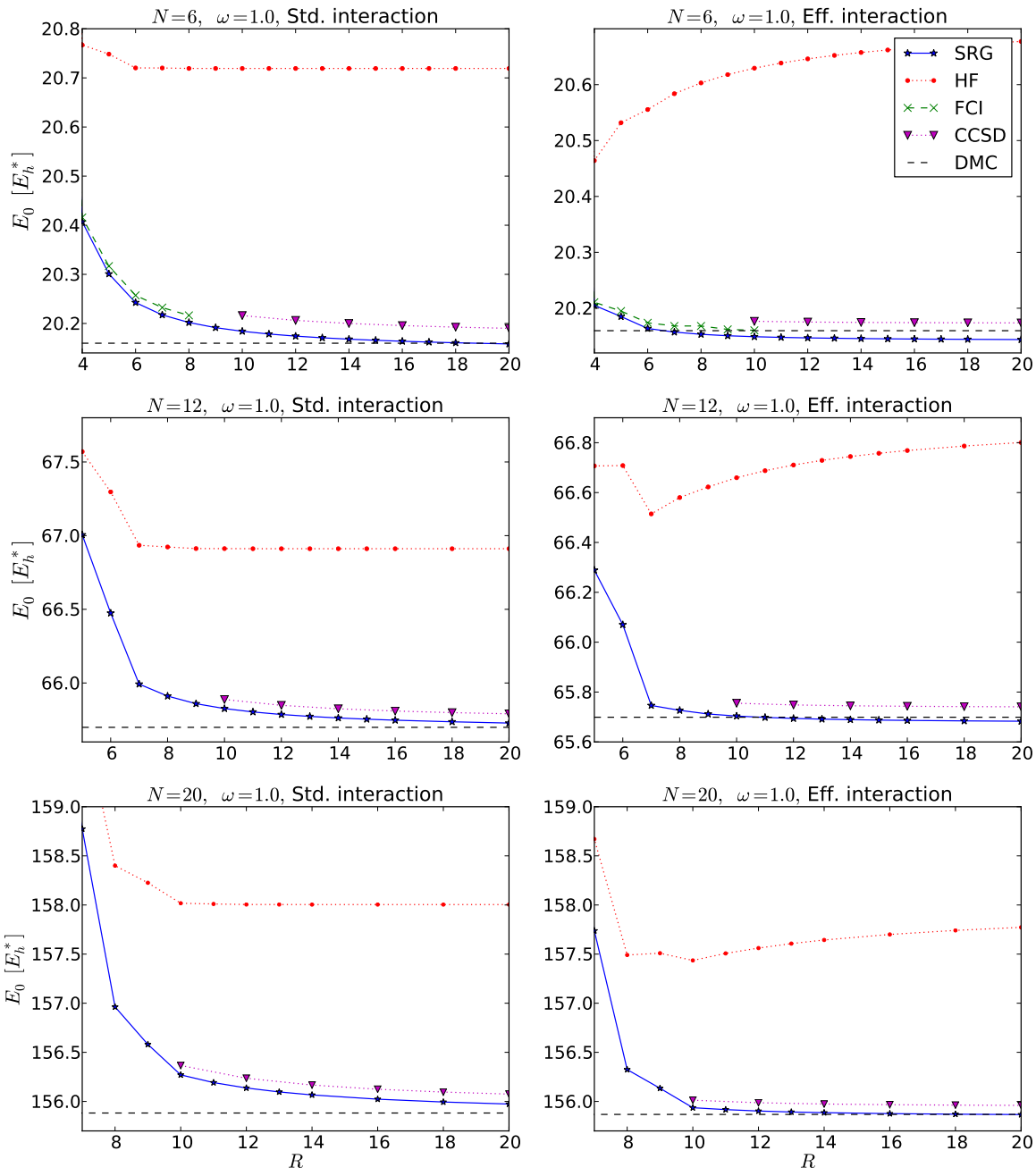

Figure 9.14: Comparison of our IM-SRG(2) ground state energies with other many-body methods. We performed the IM-SRG(2) calculations (SRG) with White's generator and Hartree-Fock basis. The Hartree-Fock (HF) and the Diffusion Monte Carlo (DMC) results up to $N = 12$ particles have been obtained with our own code. For $N \geq 20$, the DMC results are provided by the fellow master's student J. Høgberget. The Full Configuration Interaction (FCI) results are taken from [22] and the Coupled Cluster (CCSD) results from [24]. The Coupled Cluster calculations include singles and doubles and use a Hartree-Fock basis.

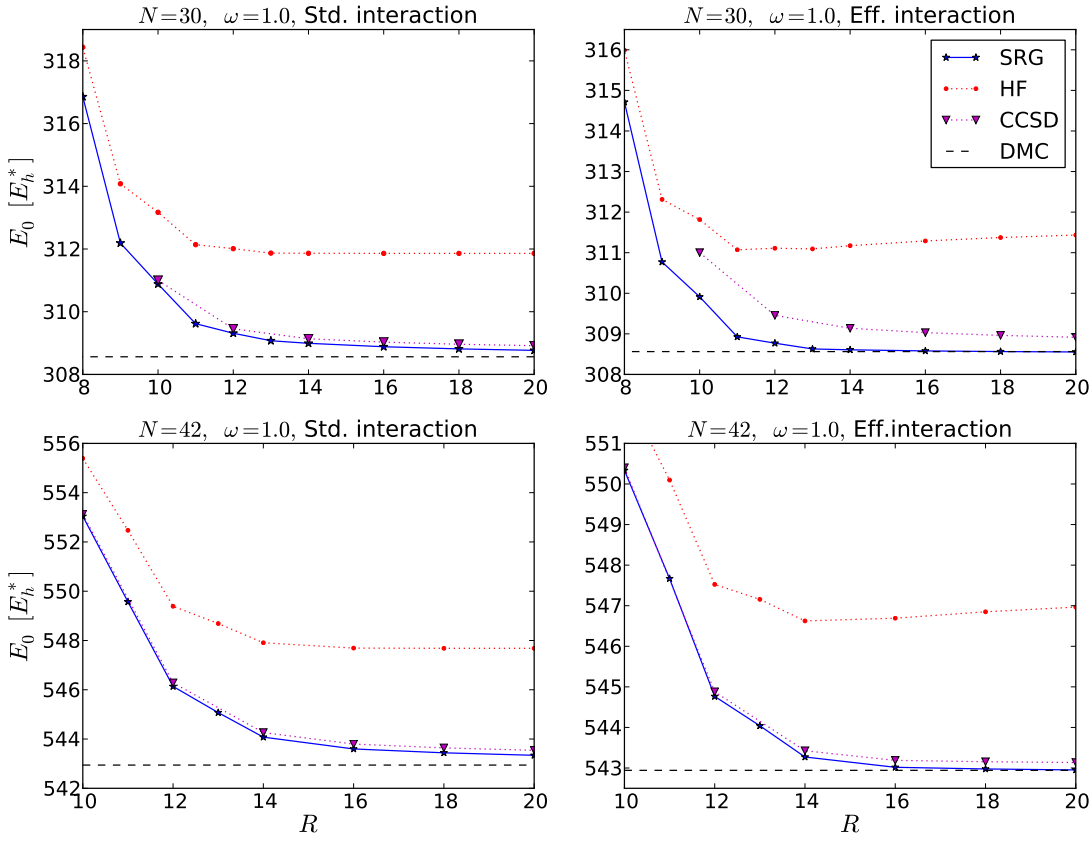

Figure 9.15: Same caption as figure 9.14. Continuation of the results for oscillator frequency $\omega = 1.0$ with $N = 30$ and $N = 42$ particles.

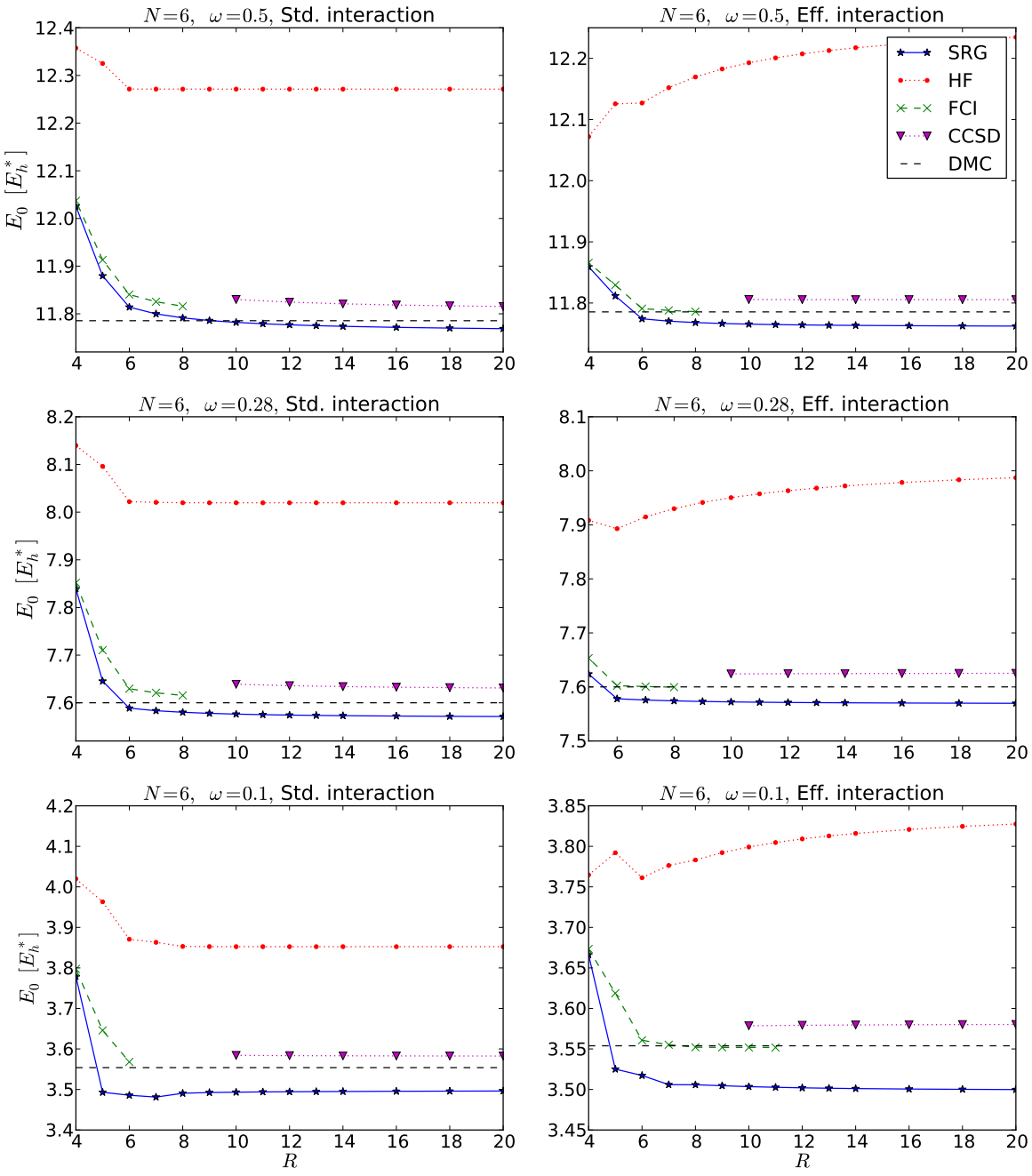

Figure 9.16: Same caption as figure 9.14. Here the results for $N = 6$ particles are presented with different values of the oscillator frequency ω.

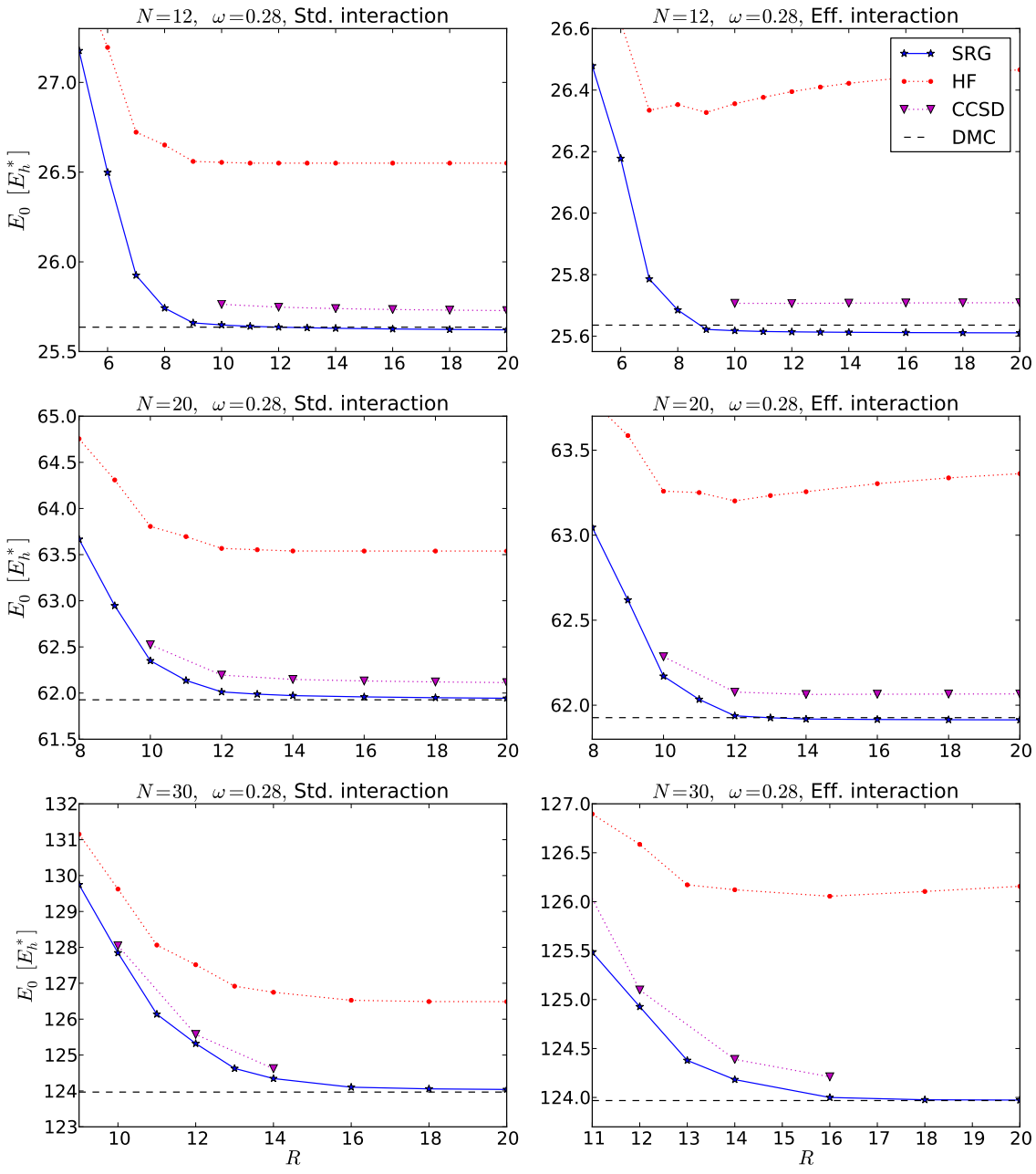

Figure 9.17: Same caption as figure 9.14. Here we present the results for oscillator frequency $\omega = 0.5$ with different numbers of particles N.

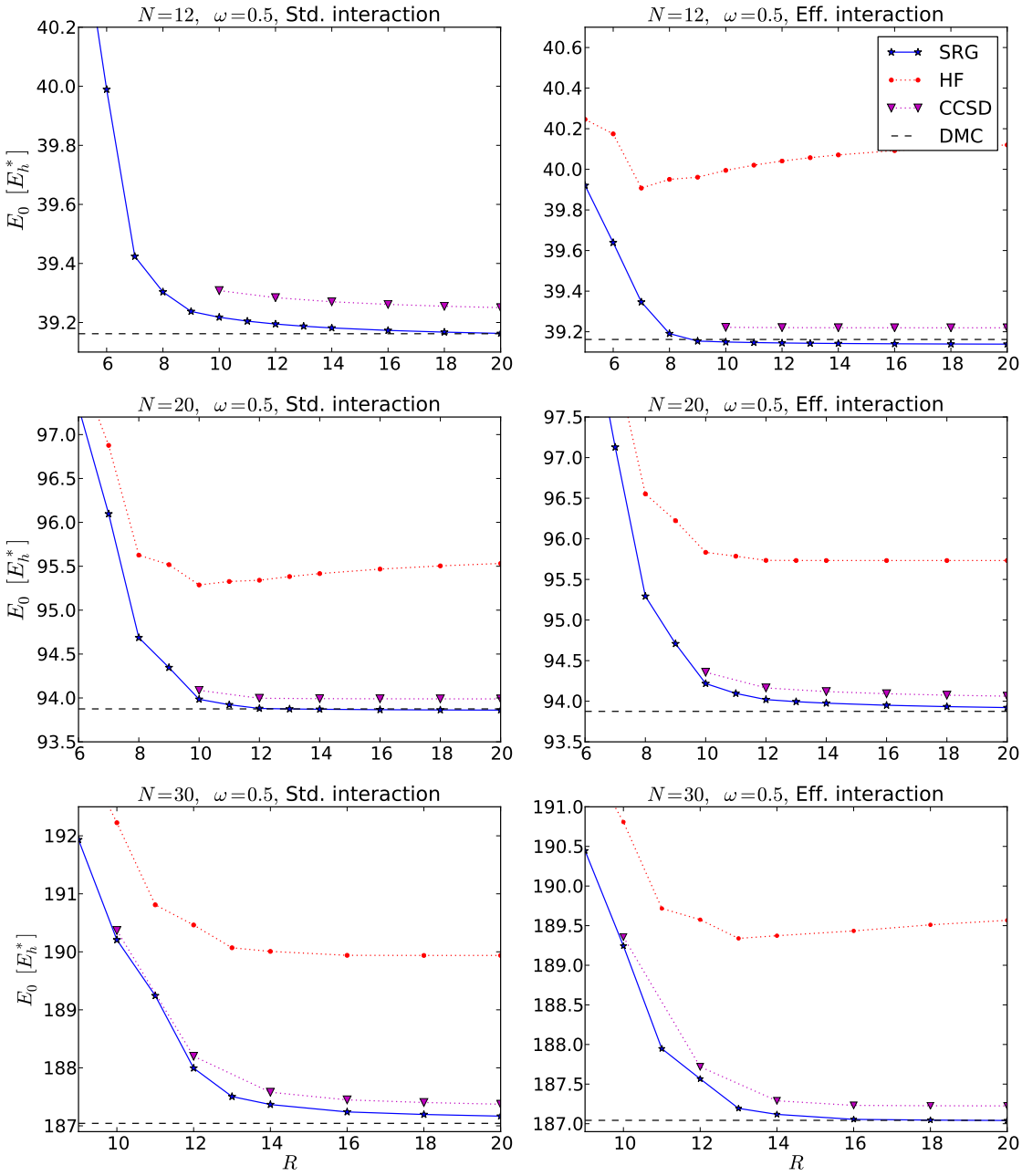

Figure 9.18: Same caption as figure 9.14. Here we present the results for oscillator frequency $\omega = 0.28$ with different numbers of particles N.

9.3.5 Study of correlation effects

Having verified that our IM-SRG(2) energies compare well with the corresponding CCSD and DMC energies, we use our results to study the physical properties of circular quantum dots. In particular, we are interested in the role of correlations between the electrons.
As a first step, we study the role of correlations as function of the number of particles N and the oscillator frequency ω. Similar to [13], we define the relative energy with respect to the unperturbed part $\hat{H}_0$ of the Hamiltonian,

$$\epsilon = \left| \frac{E_{IM-SRG(2)} - \langle \hat{H}_0 \rangle}{E_{IM-SRG(2)}} \right|. \tag{9.10}$$

The expectation value of the one-body operator, $\langle \hat{H}_0 \rangle$, is simply given as the sum of the single-particle energies, see Eq. (3.4). In atomic units, this corresponds to $\langle \hat{H}_0 \rangle = \{2\omega, 10\omega, 28\omega, \dots\}$ for $N = \{2, 6, 12, \dots\}$. The quantity ϵ measures the role of the two-body interaction included by IM-SRG(2).
The results for ϵ as function of the number of particles N and for different oscillator frequencies ω are shown in figure 9.19. As expected, the significance of interactions is increasing with the number of particles. Moreover, the effect gets more important for smaller values of the oscillator frequency ω. This meets our expectations, since the non-interacting energy $E_{non} = \sum_i \epsilon_i = \omega \sum_i (2n_i + |m|_i + 1)$ is proportional to ω, such that for lower frequencies, the two-body interaction gives comparatively a higher contribution.

Another interesting aspect is to analyse to which extent IM-SRG(2) is able to account for contributions beyond the mean-field approximation. For this reason, we introduce the relative correlation energy with respect to Hartree-Fock,

$$\chi = \left| \frac{E_{IM-SRG(2)} - E^{HF}}{E_{IM-SRG(2)}} \right|, \tag{9.11}$$

with the Hartree-Fock energy $E^{HF} = E\left[\Phi_0^{HF}\right]$ defined as in Eq. (7.3). The quantity χ measures the role of correlations beyond the mean-field approximation. For the numerical analysis, we use our optimal IM-SRG(2) results, i.e. with $R = 20$ shells, Hartree-Fock basis and effective interaction. The results are shown in figure 9.20 and reproduce what was found for CCSD in Ref. [13]:
A first observation is that the role of correlations is greater for low values of the oscillator frequency ω, similar to the previous plot. Another, even more interesting aspect is that correlations beyond the Hartree-Fock level are more important for fewer particles. To explain this observation, note that in the case of many particles, the single-particle wave functions around the Fermi level have a larger number of nodes, such that matrix elements involving states around the Fermi level usually are smaller. For that reason, particle excitations across the Fermi level get less significant as the number of particles is increasing. More illustratively stated, when increasing the number of particles, those close to the Fermi level are farther apart from each other, such that the effect of correlations between them is less important. Consequently, mean-field methods work better as the number of particles is increased. For smaller systems, correlations beyond one-particle-one-hole excitations, as included in IM-SRG(2), are needed to get a good approximation.

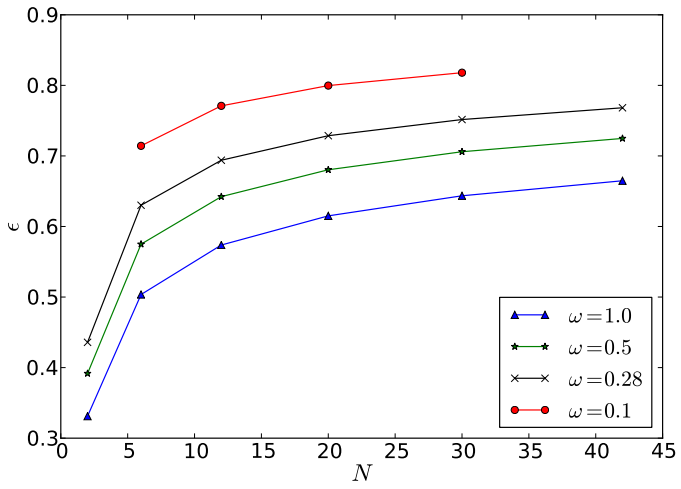

Figure 9.19: Relative correlation energy ϵ, as defined in Eq. (9.10), for different number of particles N and oscillator frequencies ω. We use our optimal IM-SRG(2) results, with effective interaction, Hartree-Fock basis and basis size $R = 20$.

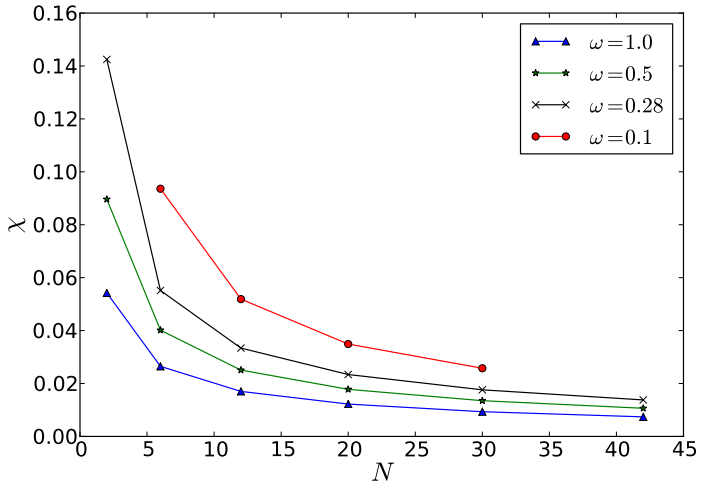

Figure 9.20: Relative correlation energy χ, as defined in Eq. (9.11), for different number of particles N and oscillator frequencies ω. We use our optimal IM-SRG(2) results with effective interaction, Hartree-Fock basis and basis size $R = 20$.

Chapter 10

Conclusions

The aim of this thesis has been to apply the Similarity Renormalization Group (SRG) method, which has recently had great success in nuclear physics [27–29], to electronic systems, in particular quantum dots, which are idealized as electrons confined in two-dimensional structures. The results have been benchmarked against other *ab initio* many-body techniques, which are used to study the physical properties of electronic systems.

To study the ground state of a system, the SRG method can be realized in different ways: The most straightforward way, which has been used in the majority of previous publications (e.g. [28–30]), is to set up the Hamiltonian with respect to a zero vacuum state and apply the flow equations to drive it to diagonal form. This approach is computationally less effective than the in-medium approach (IM-SRG), which benefits of the advantages related to normal-ordering. However, it has the advantage that no truncation occurs and thus, within a given model space, the exact result should be obtained.
To get an idea about the performance of the SRG method when applied to quantum dots, the first goal of this thesis was to implement free-space SRG and analyse the effects of the flow equations on the evolution of the Hamiltonian. In particular, we did not stop the integration at a certain resolution scale λ, as often done in practice. Instead, to test the method's capabilities for our quantum dot systems, we performed the complete diagonalization using SRG. We applied two different generators, $\hat{\eta}_1(s) = [\hat{T}_{\mathrm{rel}}, \hat{V}_s]$ and Wegner's canonical generator $\hat{\eta}_2(s) = [\hat{H}_s^{\mathrm{d}}, \hat{H}_s^{\mathrm{od}}]$. In our calculations, both generators yielded exactly the same ground state energy as obtained from exact diagonalization methods. We examined not only the decoupling of the ground state, but made also snapshots of the whole Hamiltonian matrix at different stages of the integration process. Here we observed, as expected, that energy degeneracies result in non-diagonal blocks, and that the ground state gets completely decoupled from the remaining Hamiltonian matrix.
As a next step, we aimed at identifying computational challenges and methods to deal with them. Replacing the harmonic oscillator by a Hartree-Fock basis and the standard Coulomb by an effective interaction, we were able to improve the convergence of the ground state energy as function of the basis size. However, we often encountered numerical instabilities, that we, in analogy to Coupled Cluster calculations [24, 67], expect to diminish with increasing basis size. Due to the high computational costs of the free-space approach, the IM-SRG method is computationally much more advantageous when many particles and larger effective Hilbert

spaces are involved.
Thus a next major goal of this thesis was to apply the recently evolved in-medium SRG approach to our fermionic systems. We decided on the IM-SRG(2) method, meaning that all operators are truncated at a two-body level. As a first starting point, we chose Wegner's canonical generator, and identified numerical problems related to that one: With a harmonic oscillator basis and a standard Coulomb interaction, we encountered again numerical instabilities for lower values of the oscillator frequency, as well as quite a large deviation from the corresponding Diffusion Monte Carlo results. We therefore continued with an effective interaction and a Hartree-Fock basis, and especially the latter resulted in a much smaller introduced error, less required CPU time and resolved the numerical instabilities for up to six particles. However, with increasing number of particles and lower values of the oscillator frequency, we found out that Wegner's generator leads to stiff systems of differential equations, causing numerical stability problems and non-convergence of the ground state energy. These features lead eventually to a much higher CPU expenditure.
This motivated us to combine IM-SRG with White's generator, that introduces similar decaying speeds for all matrix elements and is known to eliminate the problem of stiff equation systems [51, 52]. As a result, we indeed solved the stiffness problem and obtained converging results for systems with larger number of particles and lower oscillator frequencies, too. Moreover, we observed that computations with White's generator are much more efficient and require less CPU time, especially as the size of the basis is increased.
We finally performed our large calculations with White's generator and compared the results with other many-body methods. In general, we found that our results compare really well: Regarding the corresponding DMC results, IM-SRG(2) performs much better than Hartree-Fock, indicating how important it is to include correlations beyond the mean field approximation. Moreover, our results compare really well with the ones from CCSD. This feature of the SRG results has previously been observed in nuclear physics, too, see for example Ref. [27]. In particular, CCSD approximates the Diffusion Monte Carlo energy better than IM-SRG(2) up to $N = 6$ particles, whereas for all cases $N > 6$ IM-SRG(2) performs better. For the few cases where we have data for comparison available, our IM-SRG ground state energies compare also really well with the respective FCI data, although we noted that our energy systematically lies below the FCI one.
Finally we used our results to study the role of correlations between electrons in circular quantum dots. In particular, we found out that the impact of correlations beyond the mean-field approximation is larger for smaller number of particles and lower oscillator frequencies ω. For that reason, the Hartree-Fock method works better as the number of particles is increased. For smaller systems, correlations beyond one-particle-one-hole excitations, which are included in IM-SRG(2), are needed to obtain results with a high accuracy.

Future work and perspectives With our work, we have developed a flexible SRG code and extensions are possible in many directions. One aspect of great interest would be to extend IM-SRG(2) to IM-SRG(3), where all operators are truncated one a three-body instead of a two-body level. A comparison of both results would give insight into the significance of higher correlations and allow to analyse the convergence behaviour of the IM-SRG hierarchy of truncation.
Moreover, we have applied the SRG flow equations exclusively in m-scheme in this thesis. A next step would be to perform the calculations in an angular momentum coupled scheme,

which allows for further simplifications of the flow equations and makes the code even more efficient.
Since our code has been written in a flexible object-oriented way, it can easily be extended to look at other systems than circular, two-dimensional quantum dots. From the physical point of view, it would be interesting to analyse how SRG performs when applied to more difficult systems, like the double well quantum dots examined by Y.M. Wang [77]. An extension from quantum dots to atoms, molecules or nuclei seems possible, too. To our knowledge, atoms and molecules have not been studied with IM-SRG before. From a computational point of view, the structure of our code makes this extension rather straightforward, and the study of those systems would open up an even larger range of applications.

Appendix A

Basic commutation relations

When deriving the flow equations for IM-SRG, both the generator $\hat{\eta}$ and the final flow equations require the evaluation of commutators between operators. These operators are given in normal-ordered form, such that Wick's generalized theorem can be applied, see section 3.2. As an example, we remind the reader of the expression for Wegner's canonical generator, $\hat{\eta} = \left[\hat{H}^{\mathrm{d}}, \hat{H}\right]$, which suggests computing

$$\hat{\eta} = \left[\sum_{pq} f^d_{pq}\{a^\dagger_p a_q\} + \frac{1}{4}\sum_{pqrs} v^d_{pqrs}\{a^\dagger_p a^\dagger_q a_s a_r\}, \sum_{pq} f_{pq}\{a^\dagger_p a_q\} + \frac{1}{4}\sum_{pqrs} v_{pqrs}\{a^\dagger_p a^\dagger_q a_s a_r\}.\right],$$

The amplitudes of the diagonal Hamiltonian are defined as

$$f^d_{pq} = f_{pq}\delta_{pq}, \qquad v^d_{pqrs} = v_{pqrs}\left(\delta_{pr}\delta_{qs} + \delta_{ps}\delta_{qr}\right).$$

For instance, for the first term, $\left[\sum_{pq} f^d_{pq}\{a^\dagger_p a_q\}, \sum_{pq} f_{pq}\{a^\dagger_p a_q\}\right]$, Wick's theorem yields the following contributions for the fully contracted part:

$$\begin{aligned}
\left[\sum_{pq}\{a^\dagger_p a_q\} f^d_{pq}, \sum_{pq}\{a^\dagger_p a_q\} f_{pq}\right] &= \left[\sum_{pq}\{a^\dagger_p a_q\} f^d_{pq}, \sum_{rs}\{a^\dagger_r a_s\} f_{rs}\right] \\
&= \sum_{pqrs} f^d_{pq} f_{rs}\left[\{a^\dagger_p a_q\}, \{a^\dagger_r a_s\}\right] \\
&= \sum_{pqrs} f^d_{pq} f_{rs}\left(\{a^\dagger_p a_q\}\{a^\dagger_r a_s\} - \{a^\dagger_r a_s\}\{a^\dagger_p a_q\}\right) \\
&= \sum_{pqrs} f^d_{pq} f_{rs}\left(\{a^\dagger_p \overbracket{a_q a^\dagger_r} a_s\} - \{a^\dagger_r \overbracket{a_s a^\dagger_p} a_q\}\right) + \text{non-relevant contractions} \\
&= \sum_{ia}\left(f^d_{ia} f_{ai} - f^d_{ai} f_{ia}\right) + \text{non-relevant contractions},
\end{aligned}$$

where we collect all not fully contracted terms in "non-relevant contractions". As before, we use the notations that indices $\{i, j, k, \dots\}$ denote hole states below the Fermi level, $\{a, b, c, \dots\}$

denote particle states above the Fermi level, and $\{p, q, r, \dots\}$ are used as general indices.
An analogous procedure can be applied for obtaining the one-body terms, two-body terms etc. In the following, we list all the basic commutation relations that are needed to derive the IM-SRG equations, up to second-order level.
Orienting on our notation for the Hamiltonian, we denote the one- and two-body operators by

$$\begin{aligned} \hat{F} &= \sum_{pq} f_{pq}\{a_p^\dagger a_q\}, \\ \hat{V} &= \frac{1}{(2!)^2}\sum_{pqrs} v_{pqrs}\{a_p^\dagger a_q^\dagger a_s a_r\}, \end{aligned} \tag{A.1}$$

and we assume antisymmetrized elements:

$$v_{pqrs} = -v_{qprs} = -v_{pqsr} = v_{qpsr}. \tag{A.2}$$

With the permutation operator

$$\hat{P}_{pq} f(p,q) = f(q,p),$$

the fully contracted terms are

$$[\hat{F}^A, \hat{F}^B]^{(0)} = \sum_{ia}\Big((1-\hat{P}_{ia}) f_{ia}^A f_{ai}^B\Big) \tag{A.3}$$

$$[\hat{V}^A, \hat{V}^B]^{(0)} = \frac{1}{4}\sum_{ijab}\left(v_{ijab}^A v_{abij}^B - v_{ijab}^B v_{abij}^A\right) \tag{A.4}$$

the one-body terms are

$$[\hat{F}^A, \hat{F}^B]^{(1)} = \sum_{pq}\{a_p^\dagger a_q\}\sum_r\left(f_{pr}^A f_{rq}^B - f_{pr}^B f_{rq}^A\right) \tag{A.5}$$

$$[\hat{F}, \hat{V}]^{(1)} = \sum_{pq}\{a_p^\dagger a_q\}\sum_{ia}(1-\hat{P}_{ia}) f_{ia} v_{apiq} \tag{A.6}$$

$$\begin{aligned} [\hat{V}^A, \hat{V}^B]^{(1)} = \sum_{pq}\{a_p^\dagger a_q\}\Bigg\{&\frac{1}{2}\sum_{abi}\left(v_{ipab}^A v_{abiq}^B - v_{ipab}^B v_{abiq}^A\right) \\ &+\frac{1}{2}\sum_{ija}\left(v_{apij}^A v_{ijaq}^B - v_{apij}^B v_{ijaq}^A\right)\Bigg\} \end{aligned} \tag{A.7}$$

and the two-body terms

$$[\hat{F}, \hat{V}]^{(2)} = \frac{1}{4}\sum_{pqrs}\{a_p^\dagger a_q^\dagger a_s a_r\}\sum_t\Big((1-\hat{P}_{pq}) f_{pt} v_{tqrs} - (1-\hat{P}_{rs}) f_{tr} v_{pqts}\Big) \tag{A.8}$$

$$\begin{aligned} [\hat{V}^A, \hat{V}^B]^{(2)} = \frac{1}{4}\sum_{pqrs}\{a_p^\dagger a_q^\dagger a_s a_r\}\Bigg\{&\frac{1}{2}\sum_{ab}\left(v_{pqab}^A v_{abrs}^B - v_{pqab}^B v_{abrs}^A\right) \\ &-\frac{1}{2}\sum_{ij}\left(v_{pqij}^A v_{ijrs}^B - v_{pqij}^B v_{ijrs}^A\right) \\ &+\sum_{ia}(1-\hat{P}_{ia})(1-\hat{P}_{pq})(1-\hat{P}_{rs}) v_{ipar}^A v_{aqis}^B\Bigg\} \end{aligned} \tag{A.9}$$

Appendix B

Tables

B.1 Free-Space SRG

N	ω	R	$\hat{\eta}_1$	$\hat{\eta}_2$	FCI
2	0.01	2	nc	nc	-
		3	0.07624389770	0.07624389770	0.07624389770
		4	0.07429635508	0.07429635508	0.07429635508
		5	0.07383643326	0.07383643325	0.07383643325
		6	0.07383537264	0.07383537264	0.07383537264
		7	0.07383515617	0.07383515617	0.07383515617
		8	0.07383513707	0.07383513707	0.07383513707
		9	0.07383512937	0.07383512937	0.07383512937
		10	0.07383512679	0.07383512679	0.07383512679
	0.1	2	0.5125198414	0.5125198414	0.5125198414
		3	0.4421887603	0.4421887603	0.4421887603
		4	0.4418679942	0.4418679942	0.4418679942
		5	0.4416137068	0.4416137068	0.4416137068
		6	0.4414466720	0.4414466720	0.4414466720
		7	0.4413297357	0.4413297357	0.4413297357
		8	0.4412461536	0.4412461536	0.4412461536
		9	0.4411834870	0.4411834870	0.4411834870
		10	0.4411351270	0.4411351270	0.4411351270

Table B.1: The ground state energy E_0 (in atomic units), obtained with free-space SRG for $N = 2$ particles, is compared between runs with generator $\hat{\eta}_1 = \left[\hat{T}_{\text{rel}}, \hat{V}\right]$, generator $\hat{\eta}_2 = \left[\hat{H}^{\text{d}}, \hat{H}^{\text{od}}\right]$ and exact diagonalization (FCI). The large number of specified digits demonstrates that exactly the same results are obtained.

N	ω	R	$\hat{\eta}_1$	$\hat{\eta}_2$	FCI
2	0.28	2	1.127251038	1.127251038	1.127251038
		3	1.032681412	1.032681412	1.032681412
		4	1.028803672	1.028803672	1.028803672
		5	1.026588059	1.026588059	1.026588059
		6	1.025448813	1.025448813	1.025448813
		7	1.024705875	1.024705875	1.024705875
		8	1.024199606	1.024199606	1.024199606
		9	1.023830251	1.023830251	1.023830251
		10	1.023550577	1.023550577	1.023550577
	0.5	2	1.786913530	1.786913530	1.786913530
		3	1.681631996	1.681631996	1.681631996
		4	1.673872389	1.673872389	1.673872389
		5	1.669498218	1.669498218	1.669498218
		6	1.667257181	1.667257181	1.667257181
		7	1.665799351	1.665799351	1.665799351
		8	1.664806939	1.664806939	1.664806939
		9	1.664083215	1.664083215	1.664083215
		10	1.663535219	1.663535219	1.663535219
	1.0	2	3.152328007	3.152328007	3.152328007
		3	3.038604576	3.038604576	3.038604576
		4	3.025230582	3.025230582	3.025230582
		5	3.017606230	3.017606230	3.017606230
		6	3.013626129	3.013626129	3.013626129
		7	3.011019984	3.011019984	3.011019984
		8	3.009235721	3.009235721	3.009235721
		9	3.007929461	3.007929461	3.007929461
		10	3.006937178	3.006937178	3.006937178

Table B.2: Continuation of table B.1. See corresponding caption for explanation.

B.2 Code validation of IM-SRG(2)

ω	R	Test code IM-SRG(2)	Free-space SRG
0.1	3	0.4421887603	0.4421887603
	5	0.4416137068	0.4416137068
	7	0.4413297357	0.4413297357
0.28	3	1.032681412	1.032681412
	5	1.026588059	1.026588059
	7	1.024705875	1.024705875
0.5	3	1.681631996	1.681631996
	5	1.669498218	1.669498218
	7	1.665799351	1.665799351
1.0	3	3.038604576	3.038604576
	5	3.017606230	3.017606230
	7	3.011019984	3.011019984

Table B.3: Comparison of the ground state energy E_0 (in atomic units) obtained with free-space SRG and IM-SRG(2), for $N = 2$ particles and Wegner's generator. As explained in section 9.2, the IM-SRG(2) code has for this purpose of code validation been slightly modified. The high number of specified digits emphasizes that exactly the same results are obtained. All runs have been performed with a harmonic oscillator basis and a standard Coulomb interaction.

		Test code IM-SRG(2)		Free-space SRG
ω	R	HO basis	HF basis	
0.1	3	0.442188760	0.442188760	0.442188760
	5	0.441613707	0.441613707	0.441613707
	7	0.441329736	0.441329736	0.441329736
0.28	3	1.03268141	1.03268141	1.03268141
	5	1.02658806	1.02658806	1.02658806
	7	1.02470588	1.02470588	1.02470588
0.5	3	1.68163200	1.68163200	1.68163200
	5	1.66949822	1.66949822	1.66949822
	7	1.66579935	1.66579935	1.66579935
1.0	3	3.03860458	3.03860458	3.03860458
	5	3.01760623	3.01760623	3.01760623
	7	3.01101998	3.01101998	3.01101998

Table B.4: Comparison of the ground state energy E_0 (in atomic units) obtained with free-space SRG and IM-SRG(2), for $N = 2$ particles and White's generator. As explained in section 9.2, the IM-SRG(2) code has for this purpose of code validation been slightly modified. The high number of specified digits emphasizes that exactly the same results are obtained. All runs have been performed with a harmonic oscillator basis and a standard Coulomb interaction.

B.3 IM-SRG(2) results with Wegner's generator

	Harm. oscillator basis				Hartree-Fock basis			
R	$\omega = 0.1$	$\omega = 0.28$	$\omega = 0.5$	$\omega = 1.0$	$\omega = 0.1$	$\omega = 0.28$	$\omega = 0.5$	$\omega = 1.0$
2	nc	1.089121	1.767956	3.143526	nc	1.089121	1.767956	3.143526
3	0.4252721	1.022391	1.674052	3.033884	0.4354154	1.027421	1.677713	3.036218
4	0.4254326	1.014839	1.663100	3.018304	0.4267331	1.018093	1.665905	3.020183
5	0.4284273	1.016052	1.661176	3.012019	0.4362793	1.020519	1.664304	3.013951
6	0.4354808	1.017138	1.660178	3.008581	0.4371191	1.020532	1.662897	3.010398
7	0.4368345	1.017620	1.659615	3.006510	0.4382300	1.020842	1.662244	3.008294
8	0.4367619	1.017846	1.659181	3.005059	0.4384303	1.020830	1.661674	3.006790
9	0.4368605	1.017956	1.658851	3.004009	0.4385010	1.020767	1.661239	3.005695
10	0.4371156	1.018009	1.658578	3.003198	0.4385023	1.020669	1.660874	3.004841
11	0.4371817	1.018017	1.658347	3.002555	0.4384803	1.020565	1.660568	3.004162
12	0.4372050	1.018006	1.658150	3.002031	0.4384559	1.020467	1.660310	3.003607
DMC	0.44087(3)	1.02166(3)	1.65976(2)	3.00000(3)	0.44081(1)	1.02166(3)	1.65975(2)	3.00000(3)

Table B.5: Ground state energies (in atomic units) obtained with IM-SRG(2) for $N = 2$ electrons. All calculations have been performed with Wegner's generator and bare Coulomb interaction. For benchmarking, we also included the Diffusion Monte Carlo (DMC) results, where the number in brackets denotes the standard error.

	Harm. oscillator basis				Hartree-Fock basis			
R	$\omega = 0.1$	$\omega = 0.28$	$\omega = 0.5$	$\omega = 1.0$	$\omega = 0.1$	$\omega = 0.28$	$\omega = 0.5$	$\omega = 1.0$
2	0.4552503	1.060039	1.708803	3.060394	0.4552503	1.060039	1.708804	3.060394
3	0.4325924	1.018938	1.660647	3.006047	0.4360931	1.020688	1.662112	3.007167
4	0.4339930	1.016126	1.656843	3.001313	0.4330903	1.017199	1.658105	3.002366
5	0.4327919	1.016113	1.656178	2.999413	0.4363536	1.018151	1.657803	3.000586
6	0.4360176	1.016597	1.656036	2.998598	0.4369135	1.018444	1.657671	2.999826
7	0.4368824	1.016830	1.656023	2.998209	0.4376575	1.018825	1.657765	2.999499
8	0.4367201	1.016996	1.656036	2.997970	0.4379233	1.019021	1.657810	2.999289
9	0.4367887	1.017110	1.656056	2.997824	0.4380649	1.019142	1.657844	2.999158
10	0.4369907	1.017197	1.656072	2.997719	0.4381416	1.019214	1.657860	2.999059
11	0.4370701	1.017252	1.656082	2.997641	0.4381797	1.019255	1.657864	2.998982
12	0.4371174	1.017292	1.656087	2.997581	0.4382028	1.019280	1.657862	2.998920
DMC	0.44081(1)	1.02166(3)	1.65976(2)	3.00000(3)	0.44081(1)	1.02166(3)	1.65976(2)	3.00000(3)

Table B.6: Ground state energies (in atomic units) obtained with IM-SRG(2) for $N = 2$ electrons. All calculations have been performed with Wegner's generator and effective interaction. For benchmarking, we also included the Diffusion Monte Carlo (DMC) results, where the number in brackets denotes the standard error.

	Harm. oscillator basis				Hartree-Fock basis			
R	$\omega = 0.1$	$\omega = 0.28$	$\omega = 0.5$	$\omega = 1.0$	$\omega = 0.1$	$\omega = 0.28$	$\omega = 0.5$	$\omega = 1.0$
3	nc	nc	12.87740	21.40509	nc	nc	12.89292	21.41679
4	nc	7.82997	12.01971	20.40296	nc	7.84871	12.03399	20.41322
5	nc	7.61752	11.88042	20.29962	3.57515	7.68673	11.90040	20.31070
6	nc	nc	11.80833	20.23782	3.54968	7.61780	11.83243	20.25264
7	nc	nc	11.79448	20.21363	3.54587	7.60919	11.81643	20.22692
8	nc	nc	11.78199	20.19841	3.54887	7.60720	11.80874	20.21170
9	nc	nc	11.77924	20.18864	3.54941	7.60536	11.80349	20.20133
10	nc	nc	11.77722	20.18178	3.55024	7.60405	11.79986	20.19410
11	nc	nc	11.77510	20.17660	3.55046	7.60300	11.79714	20.18868
12	nc	nc	11.77334	20.17258	3.55052	7.60213	11.79500	20.18448
DMC	-	7.6001(2)	11.7855(8)	20.1598(4)	3.5539(1)	7.6001(2)	11.7855(8)	20.1598(4)

Table B.7: Ground state energies (in atomic units) obtained with IM-SRG(2) for $N = 6$ electrons. All calculations have been performed with Wegner's generator and bare Coulomb interaction. The label 'nc' denotes non-converging runs. For benchmarking, we also included the Diffusion Monte Carlo (DMC) results, where the number in brackets denotes the standard error. Extension of table 9.10.

	Harm. oscillator basis				Hartree-Fock basis			
R	$\omega = 0.1$	$\omega = 0.28$	$\omega = 0.5$	$\omega = 1.0$	$\omega = 0.1$	$\omega = 0.28$	$\omega = 0.5$	$\omega = 1.0$
3	nc	nc	12.36436	20.85382	nc	8.07826	12.36955	20.86051
4	nc	7.69353	11.85851	20.20328	3.67022	7.70061	11.86345	20.20863
5	nc	7.63149	11.81564	20.18527	3.60074	7.64366	11.82329	20.19127
6	nc	7.54028	11.77229	20.16104	3.55268	7.59622	11.78635	20.17093
7	nc	nc	11.76773	20.15487	3.54689	7.59392	11.78248	20.16458
8	nc	nc	11.76193	20.15073	3.54726	7.59418	11.78119	20.16110
9	nc	nc	11.76187	20.14857	3.54753	7.59421	11.78039	20.15884
10	nc	nc	11.76192	20.14719	3.54828	7.59429	11.77994	20.15744
11	nc	nc	11.76156	20.14620	3.54868	7.59436	11.77964	20.15647
12	nc	nc	11.76121	20.14547	3.54891	7.59439	11.77941	20.15577
DMC	-	7.6001(2)	11.7855(8)	20.1598(4)	3.5539(1)	7.6001(2)	11.7855(8)	20.1598(4)

Table B.8: Ground state energies (in atomic units) obtained with IM-SRG(2) for $N = 6$ electrons. All calculations have been performed with Wegner's generator and effective interaction. The label 'nc' denotes non-converging runs. For benchmarking, we also included the Diffusion Monte Carlo (DMC) results, where the number in brackets denotes the standard error.

B.4 IM-SRG(2) results with White's generator

ω	R	$N=6$				
		Standard		Effective		
		HF	SRG	HF	SRG	DMC
1.0	3	21.59320	21.41211	20.98230	20.85864	20.1598(4)
	4	20.76692	20.40701	20.46414	20.20510	
	5	20.74840	20.30093	20.53188	20.18511	
	6	20.72026	20.24258	20.55552	20.16379	
	7	20.72013	20.21740	20.58382	20.15726	
	8	20.71925	20.20197	20.60309	20.15334	
	9	20.71925	20.19147	20.61800	20.15074	
	10	20.71922	20.18407	20.62949	20.14904	
	11	20.71922	20.17852	20.63866	20.14781	
	12	20.71922	20.17423	20.64613	20.14691	
	13	20.71922	20.17082	20.65234	20.14622	
	14	20.71922	20.16804	20.65758	20.14568	
	15	20.71922	20.16573	20.66205	20.14524	
	16	20.71922	20.16378	20.66593	20.14488	
	17	20.71922	20.16212	20.66693	20.14458	
	18	20.71922	20.16068	20.67229	20.14433	
	20	20.71922	20.15833	20.67730	20.14393	
0.5	3	13.05160	12.88169	12.47376	12.36581	11.7855(8)
	4	12.35747	12.02571	12.07198	11.85925	
	5	12.32513	11.87957	12.12564	11.81166	
	6	12.27150	11.81401	12.12696	11.77419	
	7	12.27138	11.79987	12.15198	11.77034	
	8	12.27136	11.79151	12.16956	11.76804	
	9	12.27134	11.78593	12.18262	11.76651	
	10	12.27133	11.78203	12.19272	11.76547	
	11	12.27132	11.77914	12.20075	11.76473	
	12	12.27132	11.77691	12.20730	11.76416	
	13	12.27132	11.77516	12.21273	11.76373	
	14	12.27132	11.77374	12.21732	11.76339	
	16	12.27132	11.77158	12.22463	11.76289	
	18	12.27132	11.77003	12.23021	11.76254	
	20	12.27132	11.76885	12.23459	11.76228	

Table B.9: Ground state energy E_0 (in atomic units), obtained with IM-SRG(2), for a system with $N = 6$ particles. All calculations have been performed with Hartree-Fock basis and White's generator. The variable R represents the number of oscillator shells. The left-most two columns list the Hartree-Fock (HF) and SRG result with standard Coulomb interaction, the next two colums with effective interaction. For benchmarking, the last column shows the results that we get with Diffusion Monte Carlo (DMC).

ω	R	$N=6$				
		Standard		Effective		
		HF	SRG	HF	SRG	DMC
0.28	3	8.72502	nc	8.17707	8.06895	7.6001(2)
	4	8.13972	7.83914	7.86602	7.69623	
	5	8.09588	7.64560	7.90831	7.62391	
	6	8.02196	7.58885	7.89287	7.57813	
	7	8.02057	7.58345	7.91470	7.57577	
	8	8.01963	7.58020	7.92991	7.57422	
	9	8.01961	7.57795	7.94147	7.57311	
	10	8.01957	7.57634	7.95035	7.57230	
	11	8.01957	7.57519	7.95744	7.57170	
	12	8.01957	7.57430	7.96321	7.57125	
	13	8.01957	7.57362	7.96799	7.57090	
	14	8.01957	7.57307	7.97203	7.57063	
	16	8.01957	7.57226	7.97847	7.57022	
	18	8.01957	7.57169	7.98337	7.56993	
	20	8.01957	7.57126	7.98723	7.56972	
0.1	3	4.43574	nc	3.95078	nc	3.5539(1)
	4	4.01979	3.77883	3.76475	3.66606	
	5	3.96315	nc	3.79200	3.52508	
	6	3.87062	3.48576	3.76124	3.51726	
	7	3.86314	3.48124	3.77636	3.50595	
	8	3.85288	3.49061	3.78317	3.50579	
	9	3.85259	3.49250	3.79223	3.50465	
	10	3.85239	3.49329	3.79914	3.50349	
	11	3.85239	3.49397	3.80465	3.50270	
	12	3.85238	3.49436	3.80911	3.50203	
	13	3.85238	3.49473	3.81280	3.50152	
	14	3.85238	3.49501	3.81590	3.50112	
	16	3.85238	3.49550	3.82084	3.50055	
	18	3.85238	3.495893	3.82460	3.50017	
	20	3.85238	3.496203	3.82755	3.49991	

Table B.10: Continuation of table B.9, see corresponding caption for explanation. The label 'nc' denotes non-converging runs.

ω	R	$N = 12$				
		Standard		Effective		
		HF	SRG	HF	SRG	DMC
1.0	4	70.67385	70.29589	69.10357	68.80616	65.699(3)
	5	67.56993	67.00566	66.70680	66.28884	
	6	67.29687	66.47442	66.70832	66.07020	
	7	66.93474	65.99242	66.51446	65.74626	
	8	66.92309	65.91119	66.58001	65.72667	
	9	66.91224	65.85942	66.62251	65.71204	
	10	66.91204	65.82642	66.65962	65.70372	
	11	66.91136	65.80346	66.68776	65.69828	
	12	66.91136	65.78621	66.71056	65.69424	
	13	66.91132	65.77311	66.72909	65.69139	
	14	66.91132	65.76275	66.74450	65.68925	
	15	66.91132	65.75437	66.75751	65.68760	
	16	66.91132	65.74746	66.76863	65.68629	
	18	66.91132	65.73671	66.78665	65.68437	
	20	66.91132	65.72874	66.80063	65.68304	
0.5	4	43.66327	43.24100	42.13168	nc	39.162(2)
	5	41.10885	40.62285	40.24607	39.92061	
	6	40.75051	39.98939	40.17461	39.63910	
	7	40.30272	39.42398	39.90763	39.34639	
	8	40.26375	39.30358	39.95054	39.19076	
	9	40.21669	39.23755	39.96118	39.15404	
	10	40.21625	39.21784	39.99480	39.14955	
	11	40.21619	39.20457	40.02056	39.14673	
	12	40.21617	39.19486	40.04086	39.14471	
	13	40.21614	39.18748	40.05731	39.14323	
	14	40.21614	39.18171	40.07093	39.14214	
	16	40.21614	39.17328	40.09216	39.14062	
	18	40.21614	39.16742	40.10795	39.13965	
	20	40.21614	39.16313	40.12017	39.13899	

Table B.11: Ground state energy E_0 (in atomic units) for a system with $N = 12$ particles. All calculations have been performed with Hartree-Fock basis and White's generator. The variable R represents the number of oscillator shells, the label 'nc' denotes non-converging runs. The left-most two columns list the Hartree-Fock (HF) and SRG result with standard Coulomb interaction, the next two colums with effective interaction. For benchmarking, the last column shows the results that we get with Diffusion Monte Carlo (DMC).

ω	R	$N = 12$				
		Standard		Effective		
		HF	SRG	HF	SRG	DMC
0.28	4	29.73595	nc	28.25314	nc	25.636(1)
	5	27.59611	27.17675	26.73363	26.47883	
	6	27.19490	26.49848	26.62070	26.17776	
	7	26.72253	25.92512	26.33443	25.78604	
	8	26.65115	25.74255	26.35283	25.68518	
	9	26.55970	25.65997	26.32718	25.62207	
	10	26.55443	25.64839	26.35564	25.61791	
	11	26.55004	25.64090	26.37645	25.61530	
	12	26.55004	25.63619	26.39496	25.61416	
	13	26.55003	25.63261	26.40984	25.61330	
	14	26.55003	25.62985	26.42207	25.61265	
	16	26.55002	25.62587	26.44104	25.61177	
	18	26.55002	25.62316	26.45508	25.61121	
	20	26.55002	25.62121	26.46590	25.61084	
0.1	4	15.61925	nc	14.27995	nc	12.272(2)
	5	14.09824	13.78429	13.25936	13.09657	
	6	13.70045	nc	13.12793	nc	
	7	13.27086	12.63665	12.87326	12.53349	
	8	13.15107	nc	12.85545	12.33824	
	9	13.00015	12.25154	12.78357	12.27393	
	10	12.96987	12.21380	12.79405	12.23766	
	11	12.93358	12.21849	12.79009	12.23059	
	12	12.92922	12.21819	12.80393	12.22779	
	13	12.92495	12.22015	12.81407	12.22741	
	14	12.92476	12.22080	12.82426	12.22692	
	16	12.92466	12.22152	12.83980	12.22616	
	18	12.92466	12.22196	12.85111	12.22565	
	20	12.92466	12.22229	12.85971	12.22533	

Table B.12: Continuation of table B.11, see corresponding caption for explanation.

ω	R	$N = 20$				
		Standard		Effective		
		HF	SRG	HF	SRG	DMC
1.0	5	169.3217	nc	166.0855	nc	155.8822(1)
	6	161.3397	160.5668	159.4178	158.8262	
	7	159.9587	158.7748	158.6710	157.7382	
	8	158.4002	156.9626	157.4910	156.3245	
	9	158.2260	156.5807	157.5091	156.1342	
	10	158.0177	156.2704	157.4356	155.9363	
	11	158.0103	156.1920	157.5071	155.9170	
	12	158.0050	156.1371	157.5613	155.9026	
	13	158.0048	156.0968	157.6069	155.8927	
	14	158.0043	156.0665	157.6437	155.8857	
	16	158.0043	156.0232	157.7001	155.8761	
	18	158.0043	155.9944	157.7413	155.8703	
	20	158.0043	155.9737	157.7725	155.8665	
0.5	5	106.2185	nc	103.0094	nc	93.8752(1)
	6	99.75460	99.08287	97.79030	97.31453	
	7	98.19348	97.12898	96.87681	96.09675	
	8	96.55322	95.29265	95.62660	94.68694	
	9	96.22321	94.70851	95.51816	94.34604	
	10	95.83332	94.21776	95.28718	93.98470	
	11	95.78579	94.09619	95.32660	93.92273	
	12	95.73460	94.02036	95.34072	93.88021	
	13	95.73331	93.99475	95.38265	93.87463	
	14	95.73278	93.97612	95.41640	93.87089	
	16	95.73274	93.95049	95.46757	93.86612	
	18	95.73274	93.93370	95.50430	93.86328	
	20	95.73274	93.92186	95.53202	93.86145	

Table B.13: Ground state energy E_0 (in atomic units) for a system with $N = 20$ particles. All calculations have been performed with Hartree-Fock basis and White's generator. The variable R represents the number of oscillator shells, the label 'nc' denotes non-converging runs. The left-most two columns list the Hartree-Fock (HF) and SRG result with standard Coulomb interaction, the next two colums with effective interaction. The DMC results were provided by the fellow master student J. Høgberget.

ω	R	$N = 20$				
		Standard		Effective		
		HF	SRG	HF	SRG	DMC
0.28	5	73.23369	nc	70.08685	nc	61.9268(1)
	6	67.90736	67.31049	65.92851	65.53328	
	7	66.33661	65.35869	64.99235	nc	
	8	64.75479	63.66640	63.79051	63.04603	
	9	64.30909	62.94660	63.58599	62.61893	
	10	63.80561	62.35022	63.25879	62.17051	
	11	63.69533	62.13682	63.25071	62.03406	
	12	63.56727	62.01239	63.20160	61.93754	
	13	63.55277	61.98829	63.23310	61.92605	
	14	63.53940	61.97226	63.25569	61.91856	
	16	63.53881	61.95845	63.30323	61.91600	
	18	63.53880	61.94968	63.33695	61.91456	
	20	63.53880	61.94362	63.36208	61.91368	
0.1	5	39.20839	nc	36.27986	nc	29.9779(1)
	6	35.57216	35.06703	33.64794	nc	
	7	34.23072	32.11004	32.89292	nc	
	8	32.90761	32.11003	31.88419	31.42950	
	9	32.37905	nc	31.59774	nc	
	10	31.82309	30.70206	31.22119	30.54339	
	11	31.60656	30.26807	31.13172	30.27244	
	12	31.35975	30.10850	30.98747	30.10963	
	13	31.28027	29.99937	30.97266	30.01856	
	14	31.19017	29.96846	30.93558	29.97700	
	16	31.14599	29.95255	30.95087	29.95585	
	18	31.13953	29.95236	30.97752	29.95391	
	20	31.13922	29.95263	30.99927	29.95345	

Table B.14: Continuation of table B.13, see corresponding caption for explanation.

ω	R	$N = 30$				
		Standard		Effective		
		HF	SRG	HF	SRG	DMC
1.0	6	339.1696	nc	333.4968	nc	308.5627(2)
	7	322.6847	321.6779	319.0806	318.2878	
	8	318.4354	316.8561	315.9902	314.7115	
	9	314.0800	312.1890	312.3136	310.7714	
	10	313.1707	310.8837	311.8164	309.9124	
	11	312.1390	309.6182	311.0721	308.9258	
	12	312.0104	309.3140	311.1098	308.7701	
	13	311.8694	309.0741	311.0937	308.6278	
	14	311.8639	308.9924	311.1723	308.6068	
	16	311.8603	308.8829	311.2896	308.5793	
	18	311.8600	308.8143	311.3732	308.5635	
	20	311.8600	308.7673	311.4352	308.5536	
0.5	6	215.2093	nc	209.5522	nc	187.0426(2)
	7	202.1003	201.2100	198.3991	197.7436	
	8	197.7891	nc	195.2410	nc	
	9	193.5541	191.9332	191.6749	190.4457	
	10	192.2256	190.2075	190.8081	189.2440	
	11	190.8102	189.2440	189.7173	187.9475	
	12	190.4624	187.9951	189.5752	187.5663	
	13	190.0695	187.5046	189.3391	187.1936	
	14	190.0072	187.3669	189.3729	187.1177	
	16	189.9396	187.2400	189.4335	187.0548	
	18	189.9376	187.1958	189.5105	187.0458	
	20	189.9376	187.1671	189.5672	187.0408	

Table B.15: Ground state energy E_0 (in atomic units) for a system with $N = 30$ particles. All calculations have been performed with Hartree-Fock basis and White's generator. The variable R represents the number of oscillator shells, the label 'nc' denotes non-converging runs. The left-most two columns list the Hartree-Fock (HF) and SRG result with standard Coulomb interaction, the next two columns with effective interaction. The DMC results were provided by the fellow master student J. Høgberget.

ω	R	$N = 30$				
		Standard		Effective		
		HF	SRG	HF	SRG	DMC
0.28	6	149.7016	nc	144.1397	nc	123.9683(2)
	7	139.0730	138.2659	135.3429	nc	
	8	135.0536	nc	132.4451	nc	
	9	131.1536	129.7463	129.1753	128.1780	
	10	129.6246	127.8474	128.1216	nc	
	11	128.0639	126.1378	126.8957	125.4827	
	12	127.5162	125.3208	126.5854	124.9271	
	13	126.9189	124.6269	126.1722	124.3786	
	14	126.7486	124.3451	126.1218	124.1826	
	16	126.5257	124.1040	126.0557	123.9999	
	18	126.4898	124.0579	126.1046	123.9763	
	20	126.4878	124.0410	126.1573	123.9733	
0.1	6	81.23805	nc	76.06436	nc	60.4205(2)
	7	74.16367	nc	70.51280	nc	
	8	71.05573	nc	68.45012	nc	
	9	68.01272	66.91276	65.95888	nc	
	10	66.51783	nc	64.91648	nc	
	11	65.02427	63.62582	63.70699	62.82948	
	12	64.28835	nc	63.22067	nc	
	13	63.53191	61.81515	62.65837	61.54376	
	14	63.16953	61.23935	62.45200	61.14597	
	16	62.11632	60.65165	62.61035	60.65351	
	18	62.36082	60.47087	61.99956	60.47429	
	20	62.27164	60.43186	61.98690	60.43000	

Table B.16: Continuation of table B.15, see corresponding caption for explanation.

ω	R	$N = 42$				
		Standard		Effective		
		HF	SRG	HF	SRG	DMC
1.0	7	604.3198	nc	595.3531	nc	542.9428(8)
	8	574.7947	573.5344	568.7993	567.7852	
	9	564.9986	nc	560.8229	nc	
	10	555.3932	553.0438	552.2744	550.3358	
	11	552.4725	549.5708	550.0972	547.6678	
	12	549.3884	546.1358	547.5249	544.7653	
	13	548.6881	545.0728	547.1565	544.0414	
	14	547.9075	544.0770	546.6254	543.2717	
	16	547.6913	543.5977	546.6919	543.0178	
	18	547.6834	543.4402	546.8498	542.9766	
	20	547.6832	543.3398	546.9657	542.9528	
0.5	7	386.8765	nc	377.9255	nc	330.6306(2)
	8	363.7667	362.6384	357.6144	nc	
	9	354.5736	nc	350.2031	nc	
	10	345.7212	343.6797	342.3674	340.7850	
	11	342.1232	nc	339.5582	nc	
	12	338.5124	335.7230	336.4875	334.2717	
	13	337.2062	334.0219	335.5831	333.0112	
	14	335.8230	332.4212	334.5036	331.7032	
	16	334.9900	331.2554	334.0420	330.8460	
	18	334.8158	330.9641	334.0635	330.6687	
	20	334.8026	330.8885	334.1666	330.6485	
0.28	7	270.9290	nc	262.1197	nc	219.8426(2)
	8	252.4173	nc	246.2234	nc	
	9	244.2610	nc	239.7866	nc	
	10	236.4193	nc	232.9101	nc	
	11	232.7016	nc	229.9802	nc	
	12	229.0335	226.6320	226.8320	225.0394	
	13	227.3522	nc	225.5821	nc	
	14	225.6415	222.7101	224.1989	221.9471	
	16	224.1895	220.8758	223.1976	220.5309	
	18	223.6466	220.1885	222.9137	219.9997	
	20	223.5045	220.0226	222.9158	219.8836	

Table B.17: Ground state energy E_0 (in atomic units) for a system with $N = 42$ particles. All calculations have been performed with Hartree-Fock basis and White's generator. The variable R represents the number of oscillator shells, the label 'nc' denotes non-converging runs. The left-most two columns list the Hartree-Fock (HF) and SRG result with standard Coulomb interaction, the next two columns with effective interaction. The DMC results were provided by the fellow master student J. Høgberget.

	$N = 42$		
ω	R	HF	SRG
0.1	7	140.2161	nc
	8	130.3217	nc
	10	121.0367	nc
	12	115.8748	nc
	14	112.9496	nc
	16	111.3669	109.7069
	18	110.5420	108.6210

Table B.18: Same caption as table B.17. Here with $\omega = 0.1$ and exclusively using an effective interaction.

B.5 Additional material

B.5.1 Extract of an output file obtained with free-space SRG

Listing B.1: Output of free-space SRG for a system with $N = 12$ particles, $R = 6$ shells and $\omega = 0.1$. The first column shows the flow evolution parameter λ, the second column the ground state energy E_0, and the third column the required CPU time in s.

```
3           13.12004385         343.3
2.5         13.11600348         403.24
2           13.10884015         436.59
1.5         13.09449178         476.79
1           13.06059554         533.71
0.5         12.97131436         667.5
0.4         12.94435764         700.97
0.3         12.91639611         1123.59
0.2         12.88884077         1264.37
0.1         12.85558490         2052.84
0.08        12.84697661         2563.64
0.06        12.84011433         3741.36
0.04        12.83643950         7089.55
0.02        12.83246403         26688.27
0.016       12.83127608         41823.58
0.012       12.83019680         75560.03
0.008       12.82971288         166720.54
```

B.5.2 Comparison of IM-SRG(2) with Coupled Cluster results

N	ω	Δ_{SRG}	Δ_{CCSD}	
6	1.0	0.016	0.016	
	0.5	0.023	0.020	
	0.28	0.030	0.025	
	0.1	0.053	0.026	
12	1.0	0.016	0.042	
	0.5	0.023	0.057	
	0.28	0.025	0.073	
	0.1	0.047	0.076	
20	1.0	0.016	0.078	
	0.5	0.014	0.114	
	0.28	0.013	0.139	
	0.1	0.132	0.292	(R=12)
30	1.0	0.009	0.319	
	0.5	0.002	0.309	
	0.28	0.005	0.653	(R=14)
	0.1	0.725	1.214	(R=14)
42	1.0	0.010	0.560	
	0.5	1.073	2.038	(R=14)
	0.28	2.105	3.148	(R=14)

Table B.19: Comparison between IM-SRG(2) and Coupled Cluster (CCSD) ground state energies (in atomic units) with respect to Diffusion Monte Carlo. In particular, we compute for $R = 20$ the absolute differences $\Delta_{\mathrm{SRG}} = |E_{SRG} - E_{DMC}|$ and $\Delta_{\mathrm{CCSD}} = |E_{CCSD} - E_{DMC}|$. All results are obtained with effective interaction and Hartree-Fock basis. In the IM-SRG(2) calculations, we use White's generator. The CCSD results are taken from [24]. In the cases where we give a shell number $R < 20$, we have, due to convergence problems of the Coupled Cluster calculations, no CCSD data available.

Bibliography

[1] S. M. Reimann and M. Manninen. Electronic structure of quantum dots. *Rev. Mod. Phys.*, 74:1238, 2002.

[2] Hans-Andreas Engel, Vitaly N. Golovach, Daniel Loss, L. M. K. Vandersypen, J. M. Elzerman, R. Hanson, and L. P. Kouwenhoven. Measurement efficiency and n-shot readout of spin qubits. *Phys. Rev. Lett.*, 93:106804, 2004.

[3] S. Tarucha, D. G. Austing, T. Honda, R. J. van der Hage, and L. P. Kouwenhoven. Shell filling and spin effects in a few electron quantum dot. *Phys. Rev. Lett.*, 77:3613–3616, 1996.

[4] S. Strauf, K. Hennessy, M.T. Rakher, Y.-S. Choi, A. Badolato, L. C. Andreani, E. L. Hu, P.M. Petroff, and D. Bouwmeester. Self-tuned quantum dot gain in photonic crystal lasers. *Phys. Rev. Lett.*, 96:127404, 2010.

[5] Z. Mi, Jun Yang, Pallab Bhattacharya, Guoxuan Qin, and Zhenqiang Ma. High-performance quantum dot lasers and integrated optoelectronics on si. *Proc. IEEE*, 97:1239.

[6] Steven Jenks and Robert Gilmore. Quantum dot solar cell: Materials that produce two intermediate bands. *JRSE*, 2:013111, 2010.

[7] A. J. Nozik, M. C. Beard, J. M. Luther, M. Law, R. J. Ellingson, and J. C. Johnson. Semiconductor quantum dots and quantum dot arrays and applications of multiple exciton generation to third-generation photovoltaic solar cells. *Chem. Rev.*, 110:6873, 2010.

[8] Daniel Loss and David P. DiVincenzo. Quantum computation with quantum dots. *Phys. Rev. A*, 57:120, 1998.

[9] Elia T. Ben-Ari. Nanoscale quantum dots hold promise for cancer applications. *Journal of the National Cancer Institute*, 95:502–504, 2003.

[10] M. Taut. Two electrons in an external oscillator potential: Particular analytic solutions of a coulomb correlation problem. *Phys. Rev. A*, 48:3561, 1993.

[11] A. D. Güçlü, Jian-Sheng Wang, and Hong Guo. Disordered quantum dots: a diffusion quantum monte carlo study. *Phys. Rev. B*, 68:035304, 2003.

[12] Francesco Pederiva, C. J. Umrigar, and E. Lipparini. Diffusion monte carlo study of circular quantum dots. *Phys. Rev. B*, 62:8120, 2000.

[13] M. Pedersen Lohne, G. Hagen, M. Hjorth-Jensen, S. Kvaal, and F. Pederiva. *Ab initio* computation of the energies of circular quantum dots. *Phys. Rev. B*, 84:115302, 2011.

[14] F. Bolton. Fixed-phase quantum monte carlo method applied to interacting electrons in a quantum dot. *Phys. Rev. B*, 54:4780, 1996.

[15] Mikio Eto. Electronic structures of few electrons in a quantum dot under magnetic fields. *JJAP*, 36:3924, 1997.

[16] T. Ezaki, N. Mori, and C. Hamaguchi. Electronic structures in circular, elliptic, and triangular quantum dots. *Phys. Rev. B*, 56:6428, 1997.

[17] S. Kvaal. Open source FCI code for quantum dots and effective interactions. 2008.

[18] Massimo Rontani, Carlo Cavazzoni, Devis Bellucci, and Guido Goldoni. Full configuration interaction approach to the few-electron problem in artificial atoms. *J. Chem. Phys.*, 124:124102, 2006.

[19] Thomas M. Henderson, Keith Runge, and Rodney J. Bartlett. Excited states in artificial atoms via equation-of-motion coupled cluster theory. *Phys. Rev. B*, 67:045320, 2003.

[20] Ideh Heidari, Sourav Pal, B. S. Pujari, and D. G. Kanhere. Electronic structure of spherical quantum dots using coupled cluster method. *J. Chem. Phys.*, 127:114708, 2007.

[21] Andrew J. Williamson, Jeffrey C. Grossman, Randolph Q. Hood, Aaron Puzder, and Giulia Galli. Quantum monte carlo calculations of nanostructure optical gaps: Application to silicon quantum dots. *Phys. Rev. Lett.*, 89:196803, 2002.

[22] V.K.B. Olsen. "Full Configuration Interaction Simulation of Quantum Dots". Master's thesis, Department of Physics, University of Oslo, 2012. [Online]. Available: https://www.duo.uio.no/handle/123456789/34217.

[23] P.J. Reynolds, D.M. Ceperley, B.J. Alder, and W.A. Lester. Fixed-node quantum Monte Carlo for molecules. *J. Chem. Phys.*, 77:5593, 1982.

[24] C. Hirth. "Studies of quantum dots, Ab initio coupled-cluster analysis using OpenCL and GPU programming". Master's thesis, Department of Physics, University of Oslo, 2012. [Online]. Available: http://urn.nb.no/URN:NBN:no-31657.

[25] I. Shavitt and R.J. Bartlett. *Many-Body Methods in Chemistry and Physics: MBPT and Coupled-Cluster Theory*. Cambridge Molecular Science. Cambridge University Press, 2009.

[26] K. Tsukiyama, S. K. Bogner, and A. Schwenk. In-medium similarity renormalization group for nuclei. *Phys. Rev. Lett.*, 106:222502, 2011.

[27] H. Hergert, S.K. Bogner, S. Binder, S. Calci, J. Langhammer, R. Roth, and A. Schwenk. In-medium similarity renormalization group with chiral two- plus three-nucleon interactions. *Submitted to Phys. Rev. C*, 2012.

[28] E.R. Anderson, S.K. Bogner, R.J. Furnstahl, and R.J. Perry. Operator Evolution via the Similarity Renormalization Group I: The Deuteron. *Phys. Rev.C82:054001*, 2010.

[29] S. K. Bogner, R. J. Furnstahl, and R. J. Perry. Similarity renormalization group for nucleon-nucleon interactions. *Phys. Rev. C*, 75:061001, 2007.

[30] O. Åkerlund, E.J. Lindgren, J. Bergsten, B. Grevholm, P. Lerner, R. Linscott, C. Forssen, and L. Platter. The similarity renormalization group for three-body interactions in one dimension. *Eur. Phys. J. A*, 47:1, 2011.

[31] R.L. Liboff. *Introductory quantum mechanics*. Addison-Wesley Pub. Co., 1992.

[32] D.J. Griffiths. *Introduction to quantum mechanics*. Pearson Prentice Hall, 2005.

[33] M.R. Gaberdiel. Quantenmechanik I, Lecture notes. Department of Theoretical Physics, ETH-Hönggerberg, Zuerich, 2006.

[34] G. Rudolph. Vorlesungen zur Quantenmechanik I. Department of Theoretical Physics, University of Leipzig, 2009.

[35] D. Hilbert, J.v. Neumann, and L. Nordheim. Über die Grundlagen der Quantenmechanik. *Mathematische Annalen*, 98, 1928.

[36] P.A.M. Dirac. A new notation for quantum mechanics. *Math. Proc. Cambridge*, 35:416, 1939.

[37] W. Nolting. *Grundkurs Theoretische Physik 7: Viel-Teilchen-Theorie*. Springer-Lehrbuch. Springer, 2009.

[38] J. C. Slater. The Theory of Complex Spectra. *Phys. Rev.*, 34:1293, 1929.

[39] G. C. Wick. The Evaluation of the Collision Matrix. *Phys. Rev.*, 80:268, 1950.

[40] Shampine and Gordon ODE Solver. [Online], http://people.sc.fsu.edu/ jburkardt/cpp_src/ode/ode.html.

[41] L.F. Shampine and M.K. Gordon. *Computer Solution of Ordinary Differential Equations: The Initial Value Problem*. Freeman, 1975.

[42] J. W. Daniel and R. E. Moore. *Computation and theory in ordinary differential equations*. Freeman, 1970.

[43] E. Hairer, S.P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I: Non-stiff Problems*. Springer Series in Computational Mathematics. Springer, 2011.

[44] Simen Kvaal. Harmonic oscillator eigenfunction expansions, quantum dots, and effective interactions. *Phys. Rev. B*, 80:045321, 2009.

[45] Stanisław D. Głazek and Kenneth G. Wilson. Renormalization of Hamiltonians. *Phys. Rev. D*, 48:5863, 1993.

[46] Stanislaw D. Glazek and Kenneth G. Wilson. Perturbative renormalization group for hamiltonians. *Phys. Rev. D*, 49:4214, 1994.

[47] Franz J. Wegner. Flow-equations for Hamiltonians. *Ann. Phys.*, 3:77, 1994.

[48] Franz J. Wegner. Flow equations for Hamiltonians. *Phys. Rep.*, 348:77, 2001.

[49] S. Kehrein. *The Flow Equation Approach to Many-Particle Systems*. Number 217 in Springer Tracts in Modern Physics. Springer, 2006.

[50] S.K. Bogner, R.J. Furnstahl, P. Maris, R.J. Perry, A. Schwenk, and J.P. Vary. Convergence in the no-core shell model with low-momentum two-nucleon interactions. *Nuclear Physics A*, 801:21, 2008.

[51] S.R. White. A numerical canonical transformation approach to quantum many body problems. *J. Chem. Phys.*, 117, 2002.

[52] K. Tsukiyama. *In-Medium Similarity Renormalization Group for Nuclear Many-Body Systems*. PhD thesis, Department of Physics, The University of Tokyo, 2011.

[53] Robert Roth, Sven Binder, Klaus Vobig, Angelo Calci, Joachim Langhammer, and Petr Navrátil. Medium-mass nuclei with normal-ordered chiral nn+3n interactions. *Phys. Rev. Lett.*, 109:052501, 2012.

[54] R. J. Bartlett. Many-body perturbation-theory and coupled cluster theory for electron correlation in molecules. *Annu. Rev. Phys. Chem.*, 32:359, 1981.

[55] J. Thijssen. *Computational Physics*. Cambridge University Press, 2007.

[56] B.L. Hammond, J. W. A. Lester, and P.J. Reynolds. *Monte Carlo Methods in Ab Initio Quantum Chemistry*. World Scientific Lecture and Course Notes in Chemistry ; Vol. 1. World Scientific, 1994.

[57] M. Hjorth-Jensen. Computational Physics, Lecture Notes. Department of Physics, University of Oslo, 2011.

[58] Daniel Andres Nissenbaum. *The stochastic gradient approximation: an application to li nanoclusters*. PhD thesis, Northeastern University, 2008.

[59] H. Risken. *The Fokker-Planck Equation: Methods of Solution and Applications*. Springer-Verlag, 1996.

[60] J.A. Nocadal and S.J. Wright. *Numerical Optimization*. Springer Series in Operations Research Series. Springer-Verlag, 1999.

[61] J.R. Shewchuck. An introduction to the conjugate gradient method without the agonizing pain. *School of Computer Science, Carnegie Mellon University. Pittburgh*, 1994.

[62] C++ language tutorial - C++ documentation. [Online].
Available: http://www.cplusplus.com/.

[63] The C++ Programming Language. [Online].
Available: http://www.tutorialspoint.com/cplusplus/.

[64] Fred T. Krogh. A variable-step, variable-order multistep method for the numerical solution of ordinary differential equations. In *IFIP Congress (1)*, page 194, 1968.

[65] C.W. Gera. The numerical integration of ordinary differential equations. *Math. Comp.*, 21:146, 1967.

[66] M.J. Quinn. *Parallel programming in C with MPI and OpenMP*. McGraw-Hill Higher Education. McGraw-Hill Higher Education, 2004.

[67] M.H. Jørgensen. "Many-Body Approaches to Quantum Dots". Master's thesis, Department of Physics, University of Oslo, 2011. [Online]. Available: http://urn.nb.no/URN:NBN:no-28880.

[68] Spring 2012. Course FYS4411 (Computational physics II: Quantum mechanical systems), University of Oslo.

[69] W.T. Vetterling. *Numerical Recipes Example Book (C++)*. Numerical recipes series. Cambridge University Press, 2002.

[70] C.J. Umrigar, M.P. Nightingale, and K.J. Runge. A diffusion Monte Carlo algorithm with very small time-step errors. *J. Chem. Phys.*, 99, 1993.

[71] B.H. Wells. In *Methods of Computational Chemistry*, volume 1. Wiley, New York, S. Wilson edition, 1987.

[72] Ioan Kosztin, Byron Faber, and Klaus Schulten. Introduction to the Diffusion Monte Carlo method. *Am. J. Phys.*, 64:633, 1996.

[73] Armadillo, C++ linear algebra library. [Online]. Available: http://arma.sourceforge.net/.

[74] M.P. Lohne. "Coupled-Cluster Studies of Quantum Dots". Master's thesis, Department of Physics, University of Oslo, 2010. [Online]. Available: http://urn.nb.no/URN:NBN:no-28527.

[75] Noritaka Shimizu, Yutaka Utsuno, Takahiro Mizusaki, Michio Honma, Yusuke Tsunoda, et al. Variational procedure for nuclear shell-model calculations and energy-variance extrapolation. *Phys. Rev. C*, 85:054301, 2012.

[76] Andrew G. Taube and Rodney J. Bartlett. New perspectives on unitary coupled-cluster theory. *Int. J. Quant. Chem.*, 106:3393, 2006.

[77] Y.M. Wang. "Coupled-Cluster Studies of Double Quantum Dots". Master's thesis, Department of Physics, University of Oslo, 2011. [Online]. Available: http://hdl.handle.net/10852/11049.